# Robust Training and Verification of Deep Neural Networks 

Présentée le 18 août 2023

Faculté informatique et communications
Laboratoire de systèmes d'information et d'inférence
Programme doctoral en informatique et communications
pour l'obtention du grade de Docteur ès Sciences
par

## Fabian Ricardo LATORRE GOMEZ

Acceptée sur proposition du jury
Prof. L. Zdeborová, présidente du jury
Prof. V. Cevher, directeur de thèse
Prof. R. Vidal, rapporteur
Prof. T. Goldstein, rapporteur
Prof. D. Kuhn, rapporteur

## Acknowledgements

I would like to thank my advisor Prof. Volkan Cevher for his support and multiple discussions during my Ph.D., which lead to many of the results in this work. I would also like to thank the Swiss Data Science Center for supporting me financially during the last four years of my Ph.D. Special thanks to my lab's secretary Gosia Baltaian who helped me multiple times with logistical, financial and burocratic matters regarding my life as a Ph.D. student.

I would like to thank my girlfriend Noémie for her continuous support and affection during all these years in Switzerland. I also thank my family in Colombia who supported me from a distance, as well as my dogs Caramel and Luna for their unconditional love. Special thanks to my old friends from Colombia and the new friends I made in Switzerland and Europe: Steve, Nicola, Simon Z, Doménico, Simón O, Monica, Nicolas and Tea in Berlin, Delio, Bruno, Sena, Matteo, Anna, Paritosh, Jakob, Julie, Francisco in Philadelphia, David in Toronto, and many more. Special thanks to Isabel who gave me free soup during my initial years of the Ph.D.

I would like to thank G-Research for awarding me the second place of the EPFL Maths and Data Science Ph.D. prize. The monetary prize helped me pay for my family's travel to Europe for my graduation ceremony (public defense). I want to extend my gratitude to Laura Montoya and all the team from the LatinX in AI coalition who gave me the chance of volunteering for their cause on multiple editions of the workshop at ICML and NeurIPS.

I would like to thank all current and former colleagues in the lab for making this experience much more enjoyable through countless jokes, discussions and trips: Paul, Thomas S, Leello, Thomas P, Ilija, Alp, Ali K, Ali R, Chaehwan, Yongtao, Wanyun, Zhenyu, Luca Toni, Zechen, Ahmet, Maria, Bang, Pedro, Andrej, Igor, Nadav, Kamal, Fatih, Arda, Ya-Ping, Baran, Grigoris, Stratis, Kimon, Antoine and Puya. Special thanks to Ilija for hosting my very first project in the lab which helped to shape my later research. Also, my hosts and colleagues at Amazon during my internship: Matt, Chris, Milton and Harvineet. Special thanks to my collaborators in UPenn, Hamed and Alex, who alongside my friend Francisco, hosted me in Philadelphia, where we had fruitful discussion that lead to a chapter of this thesis.

Finally I would like to thank my thesis committee René Vidal, Daniel Kuhn, Tom Goldstein and Lenka Zdeborová for taking the time to review and approve this manuscript.

## Abstract

According to the proposed Artificial Intelligence Act by the European Comission (expected to pass at the end of 2023), the class of High-Risk AI Systems (Title III) comprises several important applications of Deep Learning like autonomous driving vehicles or robot-assisted surgery, which rely on supervised learning with image data. According to Article 15 in the aforementioned legal framework, such systems must be resilient to errors, faults or inconsistencies that may occur within the the environment, and to attempts by unauthorised third parties to alter their performance by exploiting the system vulnerabilities. Non-compliance can result in fines and a forced withdrawal from the market for infringing products and companies.

In this work, we develop theory and algorithms to train and certify robust Deep Neural Networks. Our theoretical results and proposed algorithms provide resiliency in different scenarios like the presence of adversarial perturbations or injection of random noise in the input features. In this way, our framework allows compliance with the requirements of the AI Act, and is a step towards a safe rollout of High-Risk AI systems based on Deep Learning.

To summarize, the main contributions of this Ph.D. thesis are: (I) first algorithm for certifying the robustness of Deep Neural Networks with the use of Polynomial Optimization, by upper bounding their Lipschitz constant (Latorre et al., 2020a), (II) first algorithm with guarantees for performing 1-path-norm regularization for Shallow Networks (Latorre et al., 2020c), and proof of its relation with the robustness to adversarial perturbations, (III) extension of 1-path-norm regularization methods to Deep Neural Networks, (IV) first generalization bounds and robustness analysis of Deep Polynomial Networks, and a novel regularization scheme to improve their robustness (Zhu et al., 2022), (V) first theoretically correct descent method for Adversarial Training, the most common algorithm for training robust networks (Latorre et al., 2023), (VI) first theoretically correct formulation of Adversarial Training as a bilevel optimization problem, which provides a solution of the robust overfitting phenomenon, (VII) ADMM algorithm with guarantees of fast convergence, for the problem of Denoising Adversarial Examples using Generative Adversarial Networks as a prior (Latorre et al., 2019) and (VIII) an explicit regularization scheme for Quadratic Neural Networks with guaranteed improvement in the robustness to random noise, compared to SVMs (Latorre et al., 2021).

## Résumé

En accord avec le projet législatif Artificial Intelligence Act proposé par la Commission Européenne (qui devrait passer à la fin de l'année 2023), plusieurs applications importantes d'apprentissage profond par réseau neuronal comme les véhicules autonomes ou la chirurgie assistée par des robots, qui utilisent des techniques d'apprentissage supervisé à partir d'images, seront inlcluses dans la classe de Systèmes d'IA à Haut-Risque (Titre III). Selon l'article 15 du cadre juridique mentionné, de tels systèmes doivent être résistants aux erreurs, fautes ou inconsistences qui peuvent survenir dans leur environnement, ainsi qu'aux tentatives d'altération de leur fonctionnement par des acteurs non-autorisés, qui pourraient exploiter les vulnerabilités du système. Leur non-conformité peut entraîner des amendes et un retrait de force du marché pour tout produit ou compagnie impliqué.

Dans cette thèse, nous développons la théorie concernant l'entraînement et l'évaluation de Réseaux Neuronaux Robustes. Nos résultats théoriques et algorithmes proposés fournissent une résistance à divers scenarios tels que la présence de perturbations adverses ou l'injection de bruit aléatoire dans les données d'entrée. De cette façon, notre cadre permet de satisfaire les exigences de l'AI Act, et constitue un pas vers un déploiement sûr des systèmes à haut-risque basés sur l'apprentisage profond.

Pour résumer, les principales contributions de cette thèse doctorale sont : (I) premier algorithme pour vérifier la robustesse de Réseaux Neuronaux Profonds basé sur l'optimisation polynomiale et une borne superiéure de leur constante de Lipschitz (Latorre et al., 2020a), (II) premier algorithme avec des garanties théoriques pour effectuer la régularisation de type 1-path-norm sur des réseaux neuronaux à une couche (Latorre et al., 2020c) et une preuve de sa relation avec la robustesse aux perturbations adverses, (III) extension du cadre de régularisation de type 1-path-norm pour des Réseaux Neuronaux profonds, (IV) premières bornes sur l'erreur de généralisation, une analyse de robustesse des Réseaux Polynomiaux profonds et un nouveau cadre de régularisation pour augmenter leur robustesse (Zhu et al., 2022), (V) première méthode correcte de descente pour l'apprentissage contradictoire (Adversarial Training), l'algorithme le plus commun pour l'entraînement de réseaux neuronaux robustes (Latorre et al., 2023), (VI) première formulation correcte de l'apprentissage contradictoire comme un problème d'optimisation à deux niveaux, fournissant une solution au problème du sur-ajustment robuste (robust overfitting), (VII) algorithme de type ADMM avec garanties des convergence rapide, pour le problème du débruitage des perturbations adverses avec des

Réseaux Antagonistes Génératifs (Generative Adversarial Networks) comme a priori (Latorre et al., 2019) et (VIII) un cadre de régularisation explicite pour des Réseaux Neuronaux avec activation quadratique, avec garantie d'amélioration de robustesse aux bruits aléatoires, en comparaison avec les SVMs (Latorre et al., 2021).

List of Keywords. Adversarial Training, Robustness, Adversarial Perturbations, Non-convex optimization, Explicit Regularization, Deep Neural Networks, Deep Polynomial Networks, Generative Adversarial Networks, Generative Models, Generalization error, Polynomial Optimization, Lipschitz constant, bilevel optimization, ADMM, AI

## Contents

Acknowledgements ..... i
Abstract (English/Français) ..... iii
1 Introduction ..... 1
1.1 Outline of the thesis ..... 4
1.1.1 Robustness through Lipschitz Regularization ..... 4
1.1.2 Upending the Adversarial Training Paradigm ..... 7
1.1.3 Robustness by Denoising the input data ..... 8
1.1.4 Random Noise Robustness through Regularization ..... 8
2 Lipschitz constant estimation of Neural Networks via sparse polynomial optimiza- tion ..... 11
2.1 Introduction ..... 11
2.2 Polynomial optimization formulation ..... 13
2.3 Hierarchical solution based on a Polynomial Positivity certificate ..... 14
2.4 Reducing the number of variables ..... 16
2.5 QCQP reformulation and Shor's SDP relaxation ..... 19
2.6 Related work ..... 20
2.7 Experiments ..... 21
2.7.1 Experiments on random networks ..... 22
2.7.2 Experiments on trained networks ..... 23
2.7.3 Estimating local Lipschitz constants with LiPopt ..... 24
2.8 Appendix: Proof of Theorem 2.1 ..... 25
2.9 Appendix: Proof of Proposition Proposition 2.1 ..... 26
2.10 Appendix: Experiments on random networks ..... 27
2.11 Bibliographic Note ..... 27
3 Efficient Proximal Mapping of the 1-path-norm of Shallow Networks ..... 29
3.1 Introduction ..... 29
3.2 Problem Setup ..... 31
3.3 The Prox-Grad Method ..... 33
3.4 Computing the Proximal Mapping ..... 34
3.5 Multi-Output ..... 38
3.6 Related Work ..... 41
3.7 Experiments ..... 43
3.7.1 Convergence of SGD vs Proximal-SGD ..... 43
3.7.2 The robustness-accuracy trade-off ..... 44
3.8 Appendix: Proof of Theorem 3.1 ..... 44
3.9 Appendix: Proof of Theorem 3.2 ..... 46
3.10 Appendix: Single output proximal map computation ..... 47
3.10.1 Moving to an Equivalent Easier Problem ..... 47
3.10.2 Solving the Prox Problem ..... 48
3.11 Appendix: Multi-output proximal map computation ..... 53
3.11.1 Solving the prox problem ..... 53
3.12 Appendix: Experimental details and other plots ..... 63
3.12.1 sparsity per iteration ..... 64
3.12.2 Robustness vs accuracy tradeoff ..... 64
3.13 Bibliographic Note ..... 66
4 How to train your 1-path-norm regularized Deep Neural Network ..... 67
4.1 Introduction ..... 67
4.2 Preliminaries and Problem Statement ..... 69
4.3 On the hardness of computing the exact Proximal Mapping ..... 72
4.4 Approximate Proximal Gradient Algorithm for the General Case ..... 72
4.5 Related Work ..... 75
4.6 Experiments ..... 76
4.7 Appendix: Proof of theorem 4.1 ..... 80
4.8 Appendix: Proof of lemma 4.4 ..... 82
4.9 Appendix: Proof of lemma 4.1 ..... 86
4.10 Appendix: Experimental Setup Details ..... 87
4.11 Bibliographic Note ..... 89
5 Controlling the Complexity and Lipschitz Constant Improves Polynomial Nets ..... 91
5.1 Introduction ..... 91
5.2 Rademacher Complexity and Lipschitz constant bounds for Polynomial Nets ..... 93
5.2.1 Coupled CP-Decomposition of Polynomial Nets (CCP model) ..... 94
5.2.2 Nested Coupled CP-Decomposition (NCP model) ..... 95
5.3 Algorithms ..... 96
5.4 Numerical Evidence ..... 99
5.4.1 Experimental Setup ..... 99
5.4.2 Evaluation of the robustness of PNs ..... 100
5.4.3 Adversarial training (AT) on PNs ..... 101
5.5 Related Work ..... 104
5.6 Appendix: introduction ..... 105
5.7 Appendix: Background ..... 105
5.7.1 Notation ..... 106
5.7.2 Definitions ..... 106
5.7.3 Well-known Lemmas ..... 108
5.8 Appendix: Details on polynomial networks ..... 110
5.8.1 Re-parametrization of CCP model ..... 110
5.8.2 Reparametrization of the NCP model ..... 112
5.9 Appendix: Result of the CCP model ..... 112
5.9.1 Proof of theorem 5.1: Rademacher Complexity bound of CCP under $\ell_{\infty}$ norm ..... 112
5.9.2 Proof of lemma 5.1 ..... 114
5.9.3 Rademacher Complexity bound under $\ell_{2}$ norm ..... 115
5.9.4 Lipschitz constant bound of the CCP model ..... 117
5.10 Appendix: Result of the NCP model ..... 119
5.10.1 Proof of Theorem 5.3: Rademacher Complexity of NCP under $\ell_{\infty}$ norm ..... 119
5.10.2 Proof of lemma 5.2 ..... 120
5.10.3 Rademacher Complexity under $\ell_{2}$ norm ..... 120
5.10.4 Lipschitz constant bound of the NCP model ..... 122
5.11 Appendix: Relationship between a Convolutional layer and a Fully Connected layer ..... 124
5.11.1 Proof of theorem 5.10 ..... 125
5.11.2 Proof of theorem 5.11 ..... 125
5.11.3 Proof of theorem 5.5 ..... 126
5.12 Appendix: Auxiliary numerical evidence ..... 126
5.12.1 Theoretical and algorithmic bound ..... 127
5.12.2 Ablation study on the hidden size ..... 127
5.12.3 Ablation study on the effect of adversarial steps ..... 129
5.12.4 Evaluation of the accuracy of PNs ..... 129
5.12.5 Experimental results on additional datasets ..... 131
5.12.6 Experimental results of more types of attacks ..... 132
5.12.7 Experimental results in NCP model ..... 132
5.12.8 Layer-wise bound ..... 132
5.12.9 Adversarial defense method ..... 141
5.13 Bibliographic Note ..... 141
6 Finding Actual Descent Directions for Adversarial Training ..... 145
6.1 Introduction ..... 145
6.2 A Counterexample to Madry et al. (2018a, Corollary C.2) ..... 147
6.3 A Counterexample at a point that is not locally optimal ..... 149
6.4 Danskin's Descent Direction ..... 150
6.5 Experiments ..... 153
6.5.1 Existence of multiple optimal adversarial solutions ..... 153
6.5.2 Exploring the Optimization Landscape of DDi and Standard Adversarial Training ..... 154
6.5.3 Accuracy/Robustness comparison of DDi vs Adversarial Training ..... 154
6.6 Related Work ..... 156
6.7 Appendix: More on counterexamples ..... 157
6.8 Appendix: Experiments ..... 158
6.8.1 Multiple attacks ..... 158
6.8.2 Justifying our modifications ..... 159
6.8.3 Further details on synthetic experiments ..... 160
6.8.4 Improving the running time ..... 160
6.9 Appendix: Proof of theorem 6.2. ..... 162
6.10 Appendix: Proof of theorem 6.3. ..... 163
6.11 Appendix: Proof of lemma 6.1 ..... 164
6.12 Bibliographic Note ..... 164
7 Adversarial Training Should Be Cast As a Non-Zero-Sum Game ..... 165
7.1 Introduction ..... 165
7.2 The promises and pitfalls of adversarial training ..... 167
7.2.1 Preliminaries: Training DNNs with surrogate losses ..... 167
7.2.2 The pervasive setting of adversarial examples ..... 168
7.2.3 Surrogate-based approaches to robustness ..... 169
7.3 Non-zero-sum formulation of adversarial training ..... 170
7.3.1 Fundamental limitations of surrogates in adversarial learning ..... 170
7.3.2 Decoupling adversarial attacks and defenses ..... 171
7.3.3 Putting the pieces together: Non-zero-sum adversarial training ..... 172
7.4 Algorithms ..... 173
7.5 Experiments ..... 174
7.6 Related work ..... 176
7.7 Appendix: Proof of proposition 7.1 ..... 178
7.8 Appendix: Smooth reformulation of the lower level ..... 179
7.9 Bibliographic Note ..... 181
8 Fast and Provable ADMM for Learning with Generative Priors ..... 183
8.1 Introduction ..... 183
8.2 Algorithm ..... 185
8.3 Optimization Guarantees ..... 186
8.4 Related Work ..... 191
8.5 Experiments ..... 193
8.6 Appendix: Statistical Learning with Generative Priors ..... 195
8.6.1 Generalization Error ..... 197
8.7 Appendix: Proof of Proposition 8.1 ..... 200
8.8 Appendix: Proof of Theorem 8.1 ..... 202
8.9 Appendix: Proof of Lemma 8.3 ..... 212
8.10 Appendix: Proof of Lemma 8.4 ..... 213
8.11 Appendix: Relation with Gradient Descent ..... 216
8.12 Appendix: Proof of Lemma 8.1 ..... 217
8.13 Appendix: Experimental Setup Details ..... 217
8.13.1 Per-Iteration Computational Complexity ..... 217
8.13.2 Parameter Tuning ..... 218
8.13.3 Fast Exact Augmented Lagrangian Minimization with Respect to Primal Variable $w$ ..... 219
8.13.4 Per-Iteration Computational Complexity ..... 219
8.14 Appendix: Pseudocode for EADMM ..... 220
8.15 Bibliographic Note ..... 220
9 The Effect of the Intrinsic Dimension on the Generalization of Quadratic Classifiers ..... 221
9.1 Introduction ..... 221
9.1.1 Related work ..... 223
9.2 Rademacher complexity bounds ..... 224
9.2.1 Improvement upon previous work ..... 228
9.3 Computable Generalization error bounds ..... 229
9.4 Frobenius vs Nuclear-norm constraint ..... 230
9.5 Experiments ..... 232
9.6 Appendix: Proof of lemma 9.2 ..... 233
9.7 Appendix: Proof of theorem 9.3 ..... 235
9.8 Appendix: Proof of lemma 9.1 and lemma 9.5 ..... 238
9.9 Appendix: Additional experiments ..... 239
9.9.1 A synthetic experiment maximizing the train/test loss gap ..... 239
9.9.2 Additional details on the isotropic normalization effects on real data ..... 240
9.9.3 Computational details on the Nuclear Norm Constrained Classifier ..... 241
9.10 Bibliographic Note ..... 242
10 Conclusions and Future Work ..... 243
Bibliography ..... 247
Curriculum Vitae ..... 279

## 1 Introduction

We are currently undergoing a fourth industrial revolution, characterized by a push towards automation of labor, device interconnectivity and human-machine interaction. This paradigm shift has been enabled thanks to the massive adoption of internet services and smart devices worldwide, which has exponentially increased the amount of available data compared to previous decades. Other instrumental developments are the increased accessibility to cloud computing and the evolution of highly parallel processor architectures like the Graphics Processing Unit (GPU).

The key technology transforming such abundant data and computational resources intro practical applications is Machine Learning (ML), which lies at the intersection between statistics and computer science. The data-driven approach of ML can solve problems like image classification (Deng et al., 2009), spam email filtering (Dada et al., 2019) or image segmentation (Kirillov et al., 2023), that are hard to model as a rules-based system based on human expertise.

Among ML techniques, the subfield of Deep Learning ( $D L$ ) has made striking advances in performance, even matching that of humans in various tasks across Computer Vision (CV) (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan and Zisserman, 2015) and Natural Language Processing (NLP) (Brown et al., 2020; Touvron et al., 2023), to name a few. Nowadays most, if not all, contemporary mainstream applications of ML are based on Deep Neural Networks. Indeed, the field of Artificial Intelligence (AI) has become synonymous with DL, despite the rich pre-DL era of symbolic (rule-based) AI (Nilsson, 1982).

The promise of near-human performance of AI systems based on Deep Learning, has prompted a gold-rush where many companies as well as governments have invested large amounts of capital, with the goal of quickly developing ML/DL-based applications. The rising cost of human labor in most of the world due to increased productivity and aging demographics, has further added to the pressure. For example, hypothetical autonomous driving systems with super-human capabilities could leave taxi drivers worldwide without a job, making it a highly profitable endeavour, if succesful. How far are we from this hypothetical scenario?

Good performance of ML/DL methods in curated benchmark test sets suggests we are close, and might lead to higher confidence in the quality of trained models deployed in real-world applications. This is evidenced by recent rollouts of autonomous driving vehicles by companies like Motional, Cruise, Waymo, and many others. Whereas Waymo takes a more careful approach by providing services only in well-tested and monitored routes within pre-mapped zones, the Full Self-Driving (FSD) mode in Tesla vehicles allows autonomous driving on any public road and is available to any owner in the US that passes a safety scoring system.

Unfortunately, the bold approach of letting AI systems take decisions outside of strictly controlled environments, has led to the discovery of multiple failure cases in ML/DL applications. Oftentimes, this has exposed users to health and economic hazards. Perhaps the most wellknown and controversial example is the aforementioned FSD system, which has been linked to erratic behaviour and fatal accidents. As of writing, the website https://www.tesladeaths.com records 31 deaths related to Tesla's Autopilot technologies. Other examples of unexpected failure cases are discriminatory practices against racial minorities in ML-based hiring (Köchling and Wehner, 2020) or credit approval (Bartlett et al., 2019; Fuster et al., 2017). This paints a grim picture of the current state and future of ML/DL applications.

We have come to the realization of a sad truth: only in the most simple or toy applications can we expect the distribution of our training and testing data to match, or to represent reality in an accurate way. The training data usually has flaws, and often the performance of ML/DL models degrades in production. Even worse, it is not uncommon that such drops in performance are severe (Zech et al., 2018; Schulam and Saria, 2017), leading to unexpected and catastrophic events. Indeed, accuracy or mean error on a curated benchmark test set is one of the poorest indicators of future reliability, safety and fairness of the decisions taken by autonomous AI systems. To understand what can go wrong, we describe a non-exhaustive list of failure cases that have to be addressed before safely deploying any ML based application to production:
$\triangleright$ Robustness to natural distribution shifts. The distribution of real-world data is always morphing due to changes in human behaviour, device specifications, quality of sensors, natural phenomena like weather conditions, and fat-tail events that occur with extremely low probability (Quionero-Candela et al., 2009; Recht et al., 2019). However, at any point in time, AI systems are limited to learn from data collected in the past. Moreover, the data collection process can suffer from biases (Torralba and Efros, 2011; Ntoutsi et al., 2020) like observing only data from sensors in pristine operational conditions (Zhu and $\mathrm{Wu}, 2004$ ). In reality, sensors like cameras can unexpectedly introduce noise due to faulty components or wear. Hence, AI systems should be tested on natural perturbations of the training data like the addition of random noise, or more structured variations like lightning conditions (Taori et al., 2020), changes in weather or geographical location (Robey et al., 2021b).
$\triangleright$ Robustness to adversarial perturbations. The real-world is a multi-agent system where
economic incentives change the behaviour of its participants. For example, a person performing a phishing attack which has been filtered by an AI system will probably make modifications to the email text until it is no longer classified as malicious (Liang et al., 2016). In cases like this, the test set performance observed during development quickly decays, as agents try to optimize their outcome. In fact, AI systems based on Deep Neural Networks are highly vulnerable to adversarial examples (Biggio et al., 2013a; Szegedy et al., 2014a), which can be defined as imperceptible modifications of data that are tailored to manipulate the decision of the network. In the context of supervised learning, this means making the network output an incorrect class. For example, it is of particular concern that traffic signs can be manipulated by third parties with the purpose of inducing errors in autonomous driving systems (Sitawarin et al., 2018; Eykholt et al., 2018a; Li et al., 2021), as this can potentially lead to fatal accidents.

The failures and risks caused by the rush to deploy AI systems in user-facing applications has not gone unnoticed by governments and regulators. Elected officials have started to draft legislation protecting their citizens from such threats. The European Union has taken the lead by introducing the Artificial Intelligence Act ${ }^{1}$. According to the draft proposed by the European Comission (expected to pass at the end of 2023), applications like autonomous driving vehicles based on Deep Learning would be classified as a High-Risk AI System (Title III). Article 15 in the aforementioned legal framework states that such systems must at least comply with the following:
$\triangleright$ High-risk AI systems shall be designed and developed in such a way that they achieve, in the light of their intended purpose, an appropriate level of accuracy, robustness and cybersecurity, and perform consistently in those respects throughout their lifecycle.
$\triangleright$ High-risk AI systems shall be resilient as regards errors, faults or inconsistencies that may occur within the system or the environment in which the system operates, in particular due to their interaction with natural persons or other systems.
$\triangleright$ High-risk AI systems shall be resilient as regards attempts by unauthorised third parties to alter their use or performance by exploiting the system vulnerabilities. The technical solutions to address AI specific vulnerabilities shall include, where appropriate, measures to prevent and control for attacks trying to manipulate the training dataset (data poisoning), inputs designed to cause the model to make a mistake (adversarial examples), or model flaws.

This regulation has extraterritorial scope (i.e., it will also apply to users outside the EU if the product is used in the EU) and will impose high fines of up to $€ 30$ million or up to $6 \%$ of the company's total worldwide annual sales for the preceding financial year (Article 71). Hence, products based on Deep Learning components that fail to achieve and certify high robustness

[^0]will not be able to reach the market. In case of failures, the responsible companies will be exposed to unexpected financial penalties and prosecution.

Is such legislation enough to prevent future accidents? Not quite. The main issue with regulations in a nascent industry like contemporary AI based on Deep Learning, is its lack of clear technical guidelines i.e., it does not describe how to prove the desired requirement of, for example, being resilient to errors. In contrast, centuries-old industries like aviation have standardized tests and protocols that can be followed in a clear manner to verify the security of a plane. Hence, we are left with general guidelines and objectives regarding the safety of AI systems, but without a clear algorithmic and theoretical basis of it means to be robust. Developing such a framework is a task delegated to industry representatives and academics alike. This leads to the main research questions of this thesis:

## How can we train Deep Neural Networks that are robust to natural/adversarial perturbations at test time? How can we verify and quantify their robustness?

In this work, we attempt to answer these questions from a Mathematical Optimization and/or Statistical Learning perspective. We model the properties under consideration as solutions of (continuous) maximization/minimization problems, and we quantify their relation with the size of the training dataset. In this way, we are able to leverage a rich algorithmic framework to achieve the desired properties in a computationally and statistically efficient and manner.

### 1.1 Outline of the thesis

Throughout, we focus on the supervised classification task with score-based classifiers. The score function is a map $f: \mathbb{R}^{d} \rightarrow \mathbb{R}^{K}$, where $K$ is the number of classes, that defines a classifier:

$$
\begin{equation*}
\hat{f}: \mathbb{R}^{d} \rightarrow\{1, \ldots, K\}, \quad \hat{f}(x)=\underset{j=1, \ldots, K}{\operatorname{argmax}} f(x)_{j} \tag{1.1}
\end{equation*}
$$

For the most part, we choose to parametrize the score function as a Deep Neural Network:

$$
\begin{equation*}
f(x):=W_{D} \sigma\left(W_{D-1} \sigma\left(\cdots W_{2} \sigma\left(W_{1} x\right) \cdots\right)\right) \tag{1.2}
\end{equation*}
$$

where $\left[W_{\ell}\right]_{i=1}^{D}$ are matrices of appropriate size, $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ is the so-called activation function (applied element-wise), and we refer to $D$ as the depth of the network.

### 1.1.1 Robustness through Lipschitz Regularization

Perhaps the simplest way to define the robustness of a classifier at a point $x$, is to determine the largest neighborhood around $x$ where the predicted label remains constant (Szegedy et al.,

2014a). An upper bound on the Lipschitz constant of the score function:

$$
\begin{equation*}
L(f)=\sup _{x_{0}, x_{1}} \frac{\left\|f\left(x_{0}\right)-f\left(x_{1}\right)\right\|}{\left\|x_{0}-x_{1}\right\|} \tag{1.3}
\end{equation*}
$$

leads to such a certificate of robustness. For example, assuming all the norms are chosen as the euclidean $\ell_{2}$-norm, we can assert the following:

Lemma 1.1. (Tsuzuku et al., 2018a, Proposition 1)

$$
\begin{equation*}
f(x)_{y}-\max _{j \neq y} f(x)_{j} \geq \sqrt{2} \epsilon L(f) \Rightarrow \underset{j=1, \ldots, K}{\operatorname{argmax}} f(x+\delta)_{j}=y, \quad \forall \delta:\|\delta\| \leq \epsilon \tag{1.4}
\end{equation*}
$$

That is, the prediction does not change inside a ball with radius proportional to the margin at $x$, and inversely proportional to the Lipschitz constant (or an upper bound thereof). Analogous bounds like (1.4) exist for different choices of $\ell_{p}$-norms.

In chapter 2 (Latorre et al., 2020a) we present an approach to compute tight upper bounds on the Lipschitz constant of a Deep Neural Network via polynomial optimization (POP) (Lasserre, 2015). Exactly computing the Lipschitz constant (1.3), even for shallow networks, is unfortunately NP-hard (Virmaux and Scaman, 2018), which leaves only upper bounds as a tractable proxy. Moreover, up to the publication of Latorre et al. (2020a), only poor upper bounds were available for Deep Neural Networks in the $\ell_{\infty}$-norm case (Cisse et al., 2017a). Our framework was further refined in other works (Chen et al., 2020b) and extended to the case of Deep Equilibrium Models (Chen et al., 2021a).

Despite providing tight bounds, the main issue with the polynomial optimization approach is its computational cost, as it involves solving a large Linear (LP) or Semidefinite (SDP) program whose number of variables grows exponentially with the depth of the network. If the goal is to certify the robustness of the network (Zhang et al., 2018; Raghunathan et al., 2018a; Lecuyer et al., 2019), such cost is bearable as the LP/SDP problem is solved only once. On the other hand, encouraging a small Lipschitz constant during training i.e., Lipschitz Regularization (Terjék, 2020; Liu et al., 2022; Kuhn et al., 2019), is a costlier endeavor that promotes robustness in the sense of lemma 1.1 as well as robustness to distribution shifts in the Wasserstein metric c.f., Kuhn et al. (2019, Theorem 5).

Indeed, given a labelled dataset $\left\{\left(x_{i}, y_{i}\right)\right\}_{i=1}^{n}$, such goal can be achieved through a regularized empirical risk minimization problem with loss $\ell$ and using and upper-bound on the Lipschitz constant $\hat{L}(f)$ as regularizer:

$$
\begin{equation*}
\min _{f} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f, x_{i}, y_{i}\right)+\lambda \hat{L}(f) \tag{1.5}
\end{equation*}
$$

Unfortunately, solving this problem with first-order methods requires computing $\hat{L}(f)$ at each iteration, which renders this approach impractical. Hence, one would rather trade tightness of the bound for a more efficient and fast way to compute it.

## Chapter 1. Introduction

In chapter 3 (Latorre et al., 2020c) we resolve the tightness vs. complexity trade-off that appears when computing upper bounds on the Lipschitz constant of a Neural Network. We demonstrate that the 1-path-norm of a Shallow Neural Network (Neyshabur et al., 2015c) achieves a sweetspot i.e., it provides an upper bound on the Lipschitz constant of the network that is tighter than the trivial product-of-weight-norms bound (Bartlett et al., 2017a; Cisse et al., 2017a), whilst its computational cost remains proportional to a forward pass over the network. Additionaly, we show that it allows a closed form proximal operator (Parikh et al., 2014) which can be efficiently computed despite its non-smoothness and non-convexity, allowing the use of stochastic proximal-gradient-type methods to obtain solutions of the regularized empirical risk minimization problem:

$$
\begin{equation*}
\min _{f} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f, x_{i}, y_{i}\right)+\lambda P_{1}(f) \tag{1.6}
\end{equation*}
$$

where $P_{1}(f)$ denotes the 1-path-norm of the Neural Network $f$. Up until this date, it is the only algorithm designed to handle this type of regularization. Due to the properties of the 1-path-norm, (1.6) is a type of Lipschitz regularization, inducing robustness in the network.

The arguments leading to the efficient proximal operator in the shallow case do not extend naturally to the setting with multiple layers, where one needs to solve a system of non-linear equations without apparent efficient solution. Unfortunately, contemporary network architectures used in industrial state-of-the-art applications are always composed of multiple layers, which limits the impact of the method developed in chapter 3 (Latorre et al., 2020c).

In chapter 4 we extend the 1-path-norm regularization framework to multilayer networks and study the feasibility of using automatic differentiation modules to either directly optimize the objective eq. (1.6), or approximately solve the proximal operator with first-order methods. Surprisingly, we find that despite the lack of optimization guarantees of such methods, they are able to increase the accuracy of Deep Neural Networks. Moreover, we show that the 1-path-norm has a positive effect on the robustness of the network to the presence of random uniform noise in the data at inference time. This adds to the existing empirical evidence in favor of the 1-path-norm as a regularizer (Jiang et al., 2020; Dziugaite et al., 2020).

In chapter 5 (Zhu et al., 2022) we derive the first bounds on the Rademacher Complexity of a various Polynomial Network architectures(Chrysos et al., 2020), under different boundedness assumptions on the input. Such bounds lead to the first known generalization error bounds for this class of models. To complement our understanding, we also derive upper bounds on their Lipschitz constant allowing Lipschitz regularization of such models with robustness guarantees.

The concept of Polynomial Neural Networks (Chrysos et al., 2020) takes the idea of multiplicative interactions (Jayakumar et al., 2020) to the extreme, and replaces all activation functions by multiplications between variables. This type of architecture has been succesful tasks like image classification Wang et al. (2018), image generation Chrysos and Panagakis (2020), sequence
models Su et al. (2020) and face verification Kemelmacher-Shlizerman et al. (2016). Hence, their use in applications warrants an study of their theoretical properties of generalization and robustness.

Indeed, recent research on Deep Learning architectures has moved beyond simple compositions of linear layers, instead exploring layers with multiplicative interactions between variables. Beyond Polynomial Networks, an important example of this trend is the introduction of the self-attention mechanism (Vaswani et al., 2017a), whose key component, the attention matrix is nothing but a second-degree polynomial of the input variables. Results in previous chapters are not directly applicable in this case, which highlights the need for theory and algorithms that are tailored to other network architectures that depart from the traditional feed-forward structure in eq. (1.2).

### 1.1.2 Upending the Adversarial Training Paradigm

Using upper bounds on the Lipschitz constant provides strong guarantees of robustness, as it measures a worst-case rate of change of the network's output as a function of the inputperturbation size. Unfortunately, this worst-case approach can lead to loose upper bounds and suboptimal performance. In contrast, Adversarial Training (Madry et al., 2018b) has as objective the average loss on the perturbed data:

$$
\begin{equation*}
\min _{f} \sum_{i=1}^{n} \max _{\left\|\delta_{i}\right\| \leq e} \ell\left(f, x_{i}+\delta_{i}, y_{i}\right) \tag{1.7}
\end{equation*}
$$

whose solution can provide better performance when evaluating the average error rate on a perturbed dataset. Indeed, AT is usually part of the best performing models in the robustness benchmark RobustBench (Croce et al., 2020a).

In chapter 6 (Latorre et al., 2023) we scrutinize the structure of the AT algorithm, and we find that its theoretical optimization foundation i.e., Madry et al. (2018b, Corollary C.2.), is false, and we present two counterexamples. More precisely, we show that finding a single optimal adversarial perturbation i.e., a solution of the inner-maximization problem in the zero-sum formulation of AT, does not necessarily yield a descent direction for the robust loss. Rather the gradient (with respect to the model parameters) at such point could constitute an ascent direction, leading to an increase in the robust loss during training. Based on correct optimization results, we derive new variations of the AT algorithm that can certify their updates as descent directions and thus are able to mitigate some of the problems of the vanilla AT approach.

In chapter 7 we uncover a more concerning issue in the the zero-sum structure of the original AT optimization formulation (Madry et al., 2018b) (1.7): a solution of this problem does not provide any guarantee of robustness to adversarial perturbations, when the loss function $\ell$ is chosen as the cross-entropy or other common surrogate losses used in practice. Thus, the occurrence of issues like robust overfitting (Rice et al., 2020), whereby more iterations of AT
lead to lower robustness, is not surprising at all. Rather, they are an artifact of the flawed AT formulation using losses like cross-entropy. We present a correct formulation of AT as a bilevel optimization formulation (Bard, 2013) and show how it provides guarantees of robustness and solves the robust overfitting issue.

### 1.1.3 Robustness by Denoising the input data

In chapter 8 (Latorre et al., 2019), we show how to achieve robustness to adversarial perturbations by modifying the data that is fed to the model. This is a different approach when compared to previous sections where we induced robustness by means of choosing better parameters for the model. Thanks to the success of generative models like Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) or Difussion models (Ho et al., 2020), it is possible to learn the distribution of clean, unperturbed samples and denoise the input data to the model by means of a non-convex projection onto the range of the generative model. In this way, any possible adversarial perturbation present in the input is partially removed, boosting the robustness of the model.

This strategy has proven succesful (Ilyas et al., 2017; Samangouei et al., 2018). It requires solving a non-convex and possibly non-smooth projection problem of the form $\min _{z}\|x-G(z)\|$ where $G$ is the generator. However, common norms used in adversarial perturbations like the $\ell_{\infty^{-}}$ norm (Madry et al., 2018b), are non-smooth and have a highly sparse (sub)gradient that leads to poor optimization of the objective. In chapter 8 (Latorre et al., 2019), we solve this problem using a splitting approach, where we adapt the ADMM algorithm to the non-convex projection problem previously mentioned. This allows the use of fast proximal operators of non-smooth norms like the $\ell_{\infty}$-norm, improving the convergence. Indeed, we observe improved robustness against $\ell_{\infty}$-bounded adversarial perturbations, when denoising using the same norm.

### 1.1.4 Random Noise Robustness through Regularization

In chapter 9 (Latorre et al., 2021) we focus on a different type of robustness, where we assume a zero-mean random variable $x$ is contaminated with Gaussian noise $\epsilon \sim \mathscr{N}(0, \sigma I)$. If we let $\Sigma=\mathbb{E}\left[x x^{\top}\right]$ be the covariance matrix of the data $x$, and $x_{\sigma}=x+\epsilon$ be the noisy variable, we have that $\Sigma_{\sigma}=\mathbb{E}\left[x_{\sigma} x_{\sigma}^{\top}\right]=\Sigma+\sigma I$. In this case we have that the intrinsic dimension (c.f. definition 9.2) of the noisy variable $x_{\epsilon}$ has the following property:

$$
\begin{equation*}
\lim _{\sigma \rightarrow \infty} \frac{\left\|\Sigma_{\sigma}\right\|_{\mathrm{tr}}}{\left\|\Sigma_{\sigma}\right\|_{2}} \rightarrow d \tag{1.8}
\end{equation*}
$$

where $d$ is the dimension of the space. Simply put, this means that when the magnitude of the random noise increases, the resulting variable does not concentrate on any linear subspace. This property is usually known as isotropy. To understand and improve the performance of a Neural Networks classifiers when Gaussian noise is present in the input, we study the class of
neural networks with quadratic activations (Du and Lee, 2018; Mannelli et al., 2020), which can be rewritten as:

$$
\begin{align*}
f(x)=v^{T} \sigma(W x) & =\sum_{i=1}^{m} v_{i} \sigma\left(w_{i}^{\top} x\right) \\
& =\sum_{i=1}^{m} v_{i} x^{\top} w_{i} w_{i}^{\top} x  \tag{1.9}\\
& =x^{\top}\left(\sum_{i=1}^{m} v_{i} w_{i} w_{i}^{\top}\right) x \\
& =x^{\top} Q x
\end{align*}
$$

Hence, we can study properties of neural networks with quadratic activations by analyzing quadratic functions of the form $x^{\top} Q x$. Indeed, eq. (1.9) states that we have the factorization $Q=\operatorname{diag} \nu W W^{\top}$. Now, recall the variational formulation of the nuclear norm of a matrix:

$$
\begin{equation*}
\|Q\|_{\mathrm{tr}}=\min _{Q=U V^{\top}}\|U\|_{F}\|V\|_{F} \tag{1.10}
\end{equation*}
$$

given that $Q=\operatorname{diag}(\nu) W W^{\top}$ we have:

$$
\begin{equation*}
\|Q\|_{\mathrm{tr}} \leq\|\operatorname{diag}(v) W\|_{F}\|W\|_{F} \leq\|v\|_{2}\|W\|_{F}^{2} \tag{1.11}
\end{equation*}
$$

The quantities on the right-hand-side of eq. (1.11) are in fact used as the most common regularization in neural networks, the so-called weight decay, which penalizes the $\ell_{2}$-norm of the vector $v$ and the Frobenius norm of the matrix $W$. Hence, in order to understand neural networks with quadratic activations and weight decay, it suffices to study the class of quadratics of the form $x^{\top} Q x$ with a nuclear norm constraint on the matrix $Q$.

Precisely, In chapter 9 (Latorre et al., 2021) we find that such class generalizes better in the presence of isotropic data, compared to kernel methods like Support Vector Machines (SVMs). This explains recent empirical observations that neural networks require fewer samples than SVMs in high-noise environments Ghorbani et al. (2020).

## 2 Lipschitz constant estimation of Neural Networks via sparse polynomial optimization

Fabian Latorre, Paul Rolland and Volkan Cevher. International Conference on Learning Representations (ICLR) 2020.


#### Abstract

We introduce LiPopt, a polynomial optimization framework for computing increasingly tighter upper bounds on the Lipschitz constant of neural networks. The underlying optimization problems boil down to either linear (LP) or semidefinite (SDP) programming. We show how to use the sparse connectivity of a network, to significantly reduce the complexity of computation. This is specially useful for convolutional as well as pruned neural networks. We conduct experiments on networks with random weights as well as networks trained on MNIST, showing that in the particular case of the $\ell_{\infty}$-Lipschitz constant, our approach yields superior estimates, compared to baselines available in the literature.


### 2.1 Introduction

We consider a neural network $f_{d}$ defined by the recursion:

$$
\begin{equation*}
f_{1}(x):=W_{1} x \quad f_{i}(x):=W_{i} \sigma\left(f_{i-1}(x)\right), \quad i=2, \ldots, d \tag{2.1}
\end{equation*}
$$

for an integer $d$ larger than 1 , matrices $\left\{W_{i}\right\}_{i=1}^{d}$ of appropriate dimensions and an activation function $\sigma$, understood to be applied element-wise. We refer to $d$ as the depth, and we focus on the case where $f_{d}$ has a single real value as output.

In this work, we address the problem of estimating the Lipschitz constant of the network $f_{d}$. A function $f$ is Lipschitz continuous with respect to a norm $\|\cdot\|$ if there exists a constant $L$ such that for all $x, y$ we have $|f(x)-f(y)| \leq L\|x-y\|$. The minimum over all such values satisfying this condition is called the Lipschitz constant of $f$ and is denoted by $L(f)$.

The Lipschitz constant of a neural network is of major importance in many successful applications of deep learning. In the context of supervised learning, Bartlett et al. (2017b) show how it directly correlates with the generalization ability of neural network classifiers, suggesting it
as model complexity measure. It also provides a measure of robustness against adversarial perturbations (Szegedy et al., 2014a) and can be used to improve such metric (Cisse et al., 2017a). Moreover, an upper bound on $L\left(f_{d}\right)$ provides a certificate of robust classification around data points (Weng et al., 2018).

Another example is the discriminator network of the Wasserstein GAN (Arjovsky et al., 2017), whose Lipschitz constant is constrained to be at most 1 . To handle this constraint, researchers have proposed different methods like heuristic penalties (Gulrajani et al., 2017), upper bounds (Miyato et al., 2018a), choice of activation function (Anil et al., 2019), among many others. This line of work has shown that accurate estimation of such constant is key to generating high quality images.

Lower bounds or heuristic estimates of $L\left(f_{d}\right)$ can be used to provide a general sense of how robust a network is, but fail to provide true certificates of robustness to input perturbations. Such certificates require true upper bounds, and are paramount when deploying safety-critical deep reinforcement learning applications (Berkenkamp et al., 2017; Jin and Lavaei, 2018). The trivial upper bound given by the product of layer-wise Lipschitz constants is easy to compute but rather loose and overly pessimistic, providing poor insight into the true robustness of a network (Huster et al., 2018).

Indeed, there is a growing need for methods that provide tighter upper bounds on $L\left(f_{d}\right)$, even at the expense of increased complexity. For example Raghunathan et al. (2018a); Jin and Lavaei (2018); Fazlyab et al. (2019a) derive upper bounds based on semidefinite programming (SDP). While expensive to compute, these type of certificates are in practice surprisingly tight. Our work belongs in this vein of research, and aims to overcome some limitations in the current state-of-the-art.

## Our Contributions.

$\triangleright$ We present LiPopt, a general approach for upper bounding the Lipschitz constant of a neural network based on a relaxation to a polynomial optimization problem (POP) (Lasserre, 2015). This approach requires only that the unit ball be described with polynomial inequalities, which covers the common $\ell_{2}$ - and $\ell_{\infty}$-norms.
$\triangleright$ Based on a theorem due to Weisser et al. (2018), we exploit the sparse connectivity of neural network architectures to derive a sequence of linear programs (LPs) of considerably smaller size than their vanilla counterparts. We provide an asymptotic analysis of the size of such programs, in terms of the number of neurons, depth and sparsity of the network.
$\triangleright$ Focusing on the $\ell_{\infty}$-norm, we experiment on networks with random weights and networks trained on MNIST (Lecun et al., 1998). We evaluate different configurations of depth, width and sparsity and we show that the proposed sequence of LPs can provide tighter upper bounds on $L\left(f_{d}\right)$ compared to other baselines available in the literature.

Notation. We denote by $n_{i}$ the number of columns of the matrix $W_{i}$ in the definition (2.1) of the network. This corresponds to the size of the $i$-th layer, where we identify the input as the first layer. We let $n=n_{1}+\ldots+n_{d}$ be the total number of neurons in the network. For a vector $x, \operatorname{Diag}(x)$ denotes the square matrix with $x$ in its diagonal and zeros everywhere else. For an array $X, \operatorname{vec}(X)$ is the flattened array. The support of a sequence $\operatorname{supp}(\alpha)$ is defined as the set of indices $j$ such that $\alpha_{j}$ is nonzero. For $x=\left[x_{1}, \ldots, x_{n}\right]$ and a sequence of nonnegative integers $\gamma=\left[\gamma_{1}, \ldots, \gamma_{n}\right]$ we denote by $x^{\gamma}$ the monomial $x_{1}^{\gamma_{1}} x_{2}^{\gamma_{2}} \ldots x_{n}^{\gamma_{n}}$. The set of nonnegative integers is denoted by $\mathbb{N}$.

Remark. The definition of network (2.1) covers typical architectures composed of dense and convolutional layers. In general, our proposed approach can be readily extended with minor modifications to any directed acyclic computation graph e.g., residual network architectures (He et al., 2016).

### 2.2 Polynomial optimization formulation

In this section we derive an upper bound on $L\left(f_{d}\right)$ given by the value of a POP, i.e. the minimum value of a polynomial subject to polynomial inequalities. Our starting point is the following theorem, which casts $L(f)$ as an optimization problem:

Theorem 2.1. Let $f$ be a differentiable and Lipschitz continuous function on an open, convex subset $\mathscr{X}$ of an euclidean space. Let $\|\cdot\|_{*}$ be the dual norm. The Lipschitz constant of $f$ is given by

$$
\begin{equation*}
L(f)=\sup _{x \in \mathscr{\mathscr { C }}}\|\nabla f(x)\|_{*} \tag{2.2}
\end{equation*}
$$

For completeness, we provide a proof in appendix 2.8. In our setting, we assume that the activation function $\sigma$ is Lipschitz continuous and differentiable. In this case, the assumptions of Theorem 2.1 are fulfilled because $f_{d}$ is a composition of activations and linear transformations. The differentiability assumption rules out the common ReLU activation $\sigma(x)=\max \{0, x\}$, but allows many others such as the exponential linear unit (ELU) (Clevert et al., 2015) or the softplus.

Using the chain rule, the compositional structure of $f_{d}$ yields the following formula for its gradient:

$$
\begin{equation*}
\nabla f_{d}(x)=W_{1}^{T} \prod_{i=1}^{d-1} \operatorname{Diag}\left(\sigma^{\prime}\left(f_{i}(x)\right)\right) W_{i+1}^{T} \tag{2.3}
\end{equation*}
$$

For every $i=1, \ldots, d-1$ we introduce a variable $s_{i}=\sigma^{\prime}\left(f_{i}(x)\right)$ corresponding to the derivative of $\sigma$ at the $i$-th hidden layer of the network. For activation functions like ELU or softplus, their derivative is bounded between 0 and 1 , which implies that $0 \leq s_{i} \leq 1$. This bound together with the definition of the dual norm $\|x\|_{*}:=\sup _{\|t\| \leq 1} t^{T} x$ implies the following upper bound of
$L\left(f_{d}\right):$

$$
\begin{equation*}
L\left(f_{d}\right) \leq \max \left\{t^{T} W_{1}^{T} \prod_{i=1}^{d-1} \operatorname{Diag}\left(s_{i}\right) W_{i+1}^{T}: 0 \leq s_{i} \leq 1,\|t\| \leq 1\right\} \tag{2.4}
\end{equation*}
$$

We will refer to the polynomial objective of this problem as the norm-gradient polynomial of the network $f_{d}$, a central object of study in this work.

For some frequently used $\ell_{p}$-norms, the constraint $\|t\|_{p} \leq 1$ can be written with polynomial inequalities. In the rest of this work, we use exclusively the $\ell_{\infty}$-norm for which $\|t\|_{\infty} \leq 1$ is equivalent to the polynomial inequalities $-1 \leq t_{i} \leq 1$, for $i=1, \ldots, n_{1}$. However, note that when $p \geq 2$ is a positive even integer, $\|t\|_{p} \leq 1$ is equivalent to a single polynomial inequality $\|t\|_{p}^{p} \leq 1$, and our proposed approach can be adapted with minimal modifications.

In such cases, the optimization problem in the right-hand side of (4.2) is a POP. Optimization of polynomials is a NP-hard problem and we do not expect to have efficient algorithms for solving (4.2) in this general form. In the next sections we describe LiPopt: a systematic way of obtaining an upper bound on $L\left(f_{d}\right)$ via tractable approximation methods of the POP (4.2).

Local Lipschitz constant. In many practical escenarios, we have additional bounds on the input of the network. For example, in the case of image classification tasks, valid input is constrained in a hypercube. In the robustness certification task, we are interested in all possible input in a $\epsilon$-ball around some data point. In those cases, it is interesting to compute a local Lipschitz constant, that is, the Lipschitz constant of a function restricted to a subset.

We can achieve this by deriving tighter bounds $0 \leq l_{i} \leq s_{i} \leq u_{i} \leq 1$, as a consequence of the restricted input (see for example, Algorithm 1 in Wong and Kolter (2018a)). By incorporating this knowledge in the optimization problem (4.2) we obtain bounds on local Lipschitz constants of $f_{d}$. We study this setting and provide numerical experiments in subsection 2.7.3.

Choice of norm. We highlight the importance of computing good upper bounds on $L\left(f_{d}\right)$ with respect to the $\ell_{\infty}$-norm. It is one of the most commonly used norms to assess robustness in the adversarial examples literature. Moreover, it has been shown that, in practice, $\ell_{\infty}$-norm robust networks are also robust in other more plausible measures of perceptibility, like the Wasserstein distance (Wong et al., 2019). This motivates our focus on this choice.

### 2.3 Hierarchical solution based on a Polynomial Positivity certificate

For ease of exposition, we rewrite (4.2) as a POP constrained in $[0,1]^{n}$ using the substitution $s_{0}:=(t+1) / 2$. Denote by $p$ the norm-gradient polynomial, and let $x=\left[s_{0}, \ldots, s_{d-1}\right]$ be the concatenation of all variables. Polynomial optimization methods (Lasserre, 2015) start from the observation that a value $\lambda$ is an upper bound for $p$ over a set $K$ if and only if the polynomial $\lambda-p$ is positive over $K$.

In LiPopt, we will employ a well-known classical result in algebraic geometry, the so-called Krivine's positivity certificate ${ }^{1}$, but in theory we can use any positivity certificate like sum-of-squares (SOS). The following is a straightforward adaptation of Krivine's certificate to our setting:

Theorem 2.2. (Adapted from Krivine (1964); Stengle (1974); Handelman (1988)) If the polynomial $\lambda-p$ is strictly positive on $[0,1]^{n}$, then there exist finitely many positive weights $c_{\alpha \beta}$ such that

$$
\begin{equation*}
\lambda-p=\sum_{(\alpha, \beta) \in \mathbb{N}^{2 n}} c_{\alpha \beta} h_{\alpha \beta}, \quad h_{\alpha \beta}(x):=\prod_{j=1}^{n} x_{j}^{\alpha_{j}}\left(1-x_{j}\right)^{\beta_{j}} \tag{2.5}
\end{equation*}
$$

By truncating the degree of Krivine's positivity certificate (Theorem 2.2) and minimizing over all possible upper bounds $\lambda$ we obtain a hierarchy of LP problems (Lasserre, 2015, Section 9):

$$
\begin{equation*}
\theta_{k}:=\min _{c \geq 0, \lambda}\left\{\lambda: \lambda-p=\sum_{(\alpha, \beta) \in \mathbb{N}_{k}^{2 n}} c_{\alpha \beta} h_{\alpha \beta}\right\} \tag{2.6}
\end{equation*}
$$

where $\mathbb{N}_{k}^{2 n}$ is the set of nonnegative integer sequences of length $2 n$ adding up to at most $k$. This is indeed a sequence of LPs as the polynomial equality constraint can be implemented by equating coefficients in the canonical monomial basis. For this polynomial equality to be feasible, the degree of the certificate has to be at least that of the norm-gradient polynomial $p$, which is equal to the depth $d$. This implies that the first nontrivial bound $\left(\theta_{k}<\infty\right)$ corresponds to $k=d$.

The sequence $\left\{\theta_{k}\right\}_{k=1}^{\infty}$ is non-incresing and converges to the maximum of the upper bound (4.2). Note that for any level of the hierarchy, the solution of the LP (2.6) provides a valid upper bound on $L\left(f_{d}\right)$.

An advantage of using Krivine's positivity certificate over SOS is that one obtains an LP hierarchy (rather than SDP), for which commercial solvers can reliably handle a large instances. Other positivity certificates offering a similar advantage are the DSOS and SDSOS hierarchies (Ahmadi and Majumdar, 2019), which boil down to LP or second order cone programming (SOCP), respectively.

Drawback. The size of the LPs given by Krivine's positivity certificate can become quite large. The dimension of the variable $c$ is $\left|\mathbb{N}_{k}^{2 n}\right|=\mathscr{O}\left(n^{k}\right)$. For reference, if we consider the MNIST dataset and a one-hidden-layer network with 100 neurons we have $\left|\mathbb{N}_{2}^{2 n}\right| \approx 1.5 \times 10^{6}$ while $\left|\mathbb{N}_{3}^{2 n}\right| \approx 9.3 \times 10^{8}$. To make this approach more scalable, in the next section we exploit the sparsity of the polynomial $p$ to find LPs of drastically smaller size than (2.6), but with similar approximation properties.

Remark. In order to compute upper bounds for local Lipschitz constants, first obtain tighter bounds $0 \leq l_{i} \leq s_{i} \leq u_{i}$ and then perform the change of variables $\widetilde{s}_{i}=\left(s_{i}-l_{i}\right) /\left(u_{i}-l_{i}\right)$ to

[^1]rewrite the problem (4.2) as a POP constrained on $[0,1]^{n}$.

### 2.4 Reducing the number of variables

Many neural network architectures, like those composed of convolutional layers, have a highly sparse connectivity between neurons. Moreover, it has been empirically observed that up to $90 \%$ of network weights can be pruned (set to zero) without harming accuracy (Frankle and Carbin, 2019). In such cases their norm-gradient polynomial has a special structure that allows polynomial positivity certificates of smaller size than the one given by Krivine's positivity certificate (Theorem 2.2).

In this section, we describe an implementation of LiPopt (Algorithm 2.1) that exploits the sparsity of the network to decrease the complexity of the LPs (2.6) given by the Krivine's positivity certificate. In this way, we obtain upper bounds on $L\left(f_{d}\right)$ that require less computation and memory. Let us start with the definition of a valid sparsity pattern:

Definition 2.1. Let $I=\{1, \ldots, n\}$ and $p$ be a polynomial with variable $x \in \mathbb{R}^{n}$. A valid sparsity pattern of $p$ is a sequence $\left\{I_{i}\right\}_{i=1}^{m}$ of subsets of $I$, called cliques, such that $\bigcup_{i=1}^{m} I_{i}=I$ and:

$$
\triangleright p=\sum_{i=1}^{m} p_{i} \text { where } p_{i} \text { is a polynomial that depends only on the variables }\left\{x_{j}: j \in I_{i}\right\}
$$

$\triangleright$ for all $i=1, \ldots, m-1$ there is an $l \leq i$ such that $\left(I_{i+1} \cap \cup_{r=1}^{i} I_{r}\right) \subseteq I_{l}$

When the polynomial objective $p$ in a POP has a valid sparsity pattern, there is an extension of Theorem 2.2 due to Weisser et al. (2018), providing a smaller positivity certificate for $\lambda-p$ over $[0,1]^{n}$. We refer to it as the sparse Krivine's certificate and we include it here for completeness:

Theorem 2.3 (Adapted from Weisser et al. (2018)). Let a polynomial p have a valid sparsity pattern $\left\{I_{i}\right\}_{i=1}^{m}$. Define $N_{i}$ as the set of sequences $(\alpha, \beta) \in \mathbb{N}^{2 n}$ where the support of both $\alpha$ and $\beta$ is contained in $I_{i}$. If $\lambda-p$ is strictly positive over $K=[0,1]^{n}$, there exist finitely many positive weights $c_{\alpha \beta}$ such that

$$
\begin{equation*}
\lambda-p=\sum_{i=1}^{m} h_{i}, \quad h_{i}=\sum_{(\alpha, \beta) \in N_{i}} c_{\alpha \beta} h_{\alpha \beta} \tag{2.7}
\end{equation*}
$$

where the polynomials $h_{\alpha \beta}$ are defined as in (2.5).

The sparse Krivine's certificate can be used like the general version (Theorem 2.2) to derive a sequence of LPs approximating the upper bound on $L\left(f_{d}\right)$ stated in (4.2). However, the number of different polynomials $h_{\alpha \beta}$ of degree at most $k$ appearing in the sparse certificate can be drastically smaller, the amount of which determines how good the sparsity pattern is.

We introduce a graph that depends on the network $f_{d}$, from which we will extract a sparsity pattern for the norm-gradient polynomial of a network.


Figure 2.1: Sparsity pattern of Proposition 2.1 for a network of depth three.


Figure 2.2: Structure of one set in the sparsity pattern from Proposition 2.1 for a network with 2D convolutional layers with $3 \times 3$ filters.

Definition 2.2. Let $f_{d}$ be a network with weights $\left\{W_{i}\right\}_{i=1}^{d}$. Define a directed graph $G_{d}=(V, E)$ as:

$$
\begin{gather*}
V=\left\{s_{i, j}: 0 \leq i \leq d-1,1 \leq j \leq n_{i}\right\} \\
E=\left\{\left(s_{i, j}, s_{i+1, k}\right): 0 \leq i \leq d-2,\left[W_{i}\right]_{k, j} \neq 0\right\} \tag{2.8}
\end{gather*}
$$

which we call the computational graph of the network $f_{d}$.

In the graph $G_{d}$ the vertex $s_{(i, j)}$ represents the $j$-th neuron in the $i$-th layer. There is a directed edge between two neurons in consecutive layers if they are joined by a nonzero weight in the network. The following result shows that for fully connected networks we can extract a valid sparsity pattern from this graph. We relegate the proof to appendix 2.9.

Proposition 2.1. Let $f_{d}$ be a dense network (all weights are nonzero). The following sets, indexed by $i=1, \ldots, n_{d}$, form a valid sparsity pattern for the norm-gradient polynomial of the network $f_{d}$ :

$$
\begin{equation*}
I_{i}:=\left\{s_{(d-1, i)}\right\} \cup\left\{s_{(j, k)}: \text { there exists a directed path from } s_{(j, k)} \text { to } s_{(d-1, i)} \text { in } G_{d}\right\} \tag{2.9}
\end{equation*}
$$

We refer to this as the sparsity pattern induced by $G_{d}$. An example is depicted in in Figure 2.1.
Remark. When the network is not dense, the the second condition (Definition 2.1) for the sparsity pattern (2.9) to be valid might not hold. In that case we lose the guarantee that the values of the corresponding LPs converge to the maximum of the POP (4.2). Nevertheless, it still provides a valid positivity certificate that we use to upper bound $L\left(f_{d}\right)$. In Section 2.7 we show that in practice it provides upper bounds of good enough quality. If needed, a valid sparsity pattern can be obtained via a chordal completion of the correlative sparsity graph of the POP (Waki et al., 2006).

We now quantify how good this sparsity pattern is. Let $s$ be the size of the largest clique in a sparsity pattern, and let $N_{i, k}$ be the subset of $N_{i}$ (defined in Theorem 2.3) composed of sequences summing up to $k$. The number of different polynomials for the $k$-th LP in the
hierarchy given by the sparse Krivine's certificate can be bounded as follows:

$$
\begin{equation*}
\left|\bigcup_{i=1}^{m} N_{i, k}\right| \leq \sum_{i=1}^{m}\binom{2\left|I_{i}\right|+k}{k}=\mathscr{O}\left(m s^{k}\right) \tag{2.10}
\end{equation*}
$$

We immediately see that the dependence on the number of cliques $m$ is really mild (linear) but the size of the cliques as well as the degree of the hierarchy can really impact the size of the optimization problem. Nevertheless, this upper bound can be quite loose; polynomials $h_{\alpha \beta}$ that depend only on variables in the intersection of two or more cliques are counted more than once.

The number of cliques given in the sparsity pattern induced by $G_{d}$ is equal to the size of the last layer $m=n_{d}$ and the size of each clique depends on the particular implementation of the network. We now study different architectures that could arise in practice, and determine the amount of polynomials in their sparse Krivine's certificate.

Fully connected networks. Even in the case of a network with all nonzero connections, the sparsity pattern induced by $G_{d}$ decreases the size of the LPs when compared to Krivine's certificate. In this case the cliques have size $n_{1}+\ldots+n_{d-1}+1$ but they all have the same common intersection equal to all neurons up to the second-to-last hidden layer. A straightforward counting argument shows that the total number of polynomials is $\mathscr{O}\left(n\left(n_{1}+\ldots+n_{d-1}+1\right)^{k-1}\right)$, improving the upper bound (2.10).

Unstructured sparsity. Sparsity in a Neural Network can appear implicitely during training: it has been observed that many weights at convergence have negligible magnitude. This is precisely the backbone of pruning methods (Hanson and Pratt, 1989) which work by setting to exact zero a large percentage of parameters with smallest magnitude, suffering only a very small drop in performance. An alternative way of inducing unstructure sparsity is by choosing the network architecture randomly from a distribution over graphs (Xie et al., 2019). In this case, the sparsity pattern can be arbitrary, and the size of the resulting LPs varies at runtime. Under the layer-wise assumption that any neuron is connected to at most $r$ neurons in the previous layer, the size of the cliques in (2.9) is bounded as $s=\mathscr{O}\left(r^{d}\right)$. This estimate has an exponential dependency on the depth but ignores that many neurons might share connections to the same inputs in the previous layer, thus being potentially loose. The bound (2.10) implies that the number of different polynomials is $\mathscr{O}\left(n_{d} r^{d k}\right)$.

2D Convolutional networks. The sparsity in the weight matrices of convolutional layers has a certain local structure; neurons are connected to contiguous inputs in the previous layer. Adjacent neurons also have many input pixels in common (see Figure 2.2). Assuming a constant number of channels per layer, the size of the cliques in (2.9) is $\mathscr{O}\left(d^{3}\right)$. Intuitively, such number is proportional to the volume of the pyramid depicted in Figure 2.2 where each dimension depends linearly on $d$. Using (2.10) we get that there are $\mathscr{O}\left(n_{d} d^{3 k}\right)$ different polynomials in the sparse Krivine's certificate. This is a drastic decrease in complexity when compared to the unstructured sparsity case.

The use of sparsity in polynomial optimization preceeds Theorem 2.3 (Weisser et al., 2018). First studied in the context of sum-of-squares by Kojima et al. (2005) and further refined in Waki et al. (2006); Lasserre (2006) (and references therein), it has found applications in safety verification (Yang et al., 2016; Zhang et al., 2018), sensor localization Wang et al. (2006), optimal power flow (Ghaddar et al., 2015) and many others. Our work fits precisely into this set of important applications.

```
Algorithm 2.1 LiPopt for ELU activations and sparsity pattern
    Input: matrices \(\left\{W_{i}\right\}_{i=1}^{d}\), sparsity pattern \(\left\{I_{i}\right\}_{i=1}^{m}\), hierarchy degree \(k\).
    \(p \leftarrow\left(2 s_{0}-1\right)^{T} W_{1}^{T} \Pi_{i=1}^{d-1} \operatorname{Diag}\left(s_{i}\right) W_{i+1}^{T} \quad \triangleright\) compute norm-gradient polynomial
    \(x \leftarrow\left[s_{0}, \ldots, s_{d-1}\right]\)
    \(b \leftarrow\left[b_{\gamma}: \gamma \in \mathbb{N}_{k}^{n}\right]\) where \(p(x)=\sum_{\gamma \in \mathbb{N}_{k}^{n}} b_{\gamma} x^{\gamma} \quad \triangleright\) compute coefficients of \(p\) in basis
    for \(i=1, \ldots, m\) do
        \(N_{i, k} \leftarrow\left\{(\alpha, \beta) \in \mathbb{N}_{k}^{2 n}: \operatorname{supp}(\alpha) \cap \operatorname{supp}(\beta) \subseteq I_{i}\right\}\)
    end for
    \(\widetilde{N}_{k} \leftarrow \cup_{i=1}^{m} N_{i, k}\)
    \(h \leftarrow \sum_{(\alpha, \beta) \in \tilde{N}} c_{\alpha \beta} h_{\alpha \beta} \quad \triangleright\) compute positivity certificate
    \(c \leftarrow\left[c_{\alpha \beta}:(\alpha, \beta) \in \widetilde{N}_{k}\right] ; y \leftarrow[\lambda, c] \quad \triangleright\) linear program variables
    \(Z \leftarrow\left[z_{\gamma}\right]_{\gamma \in \mathbb{N}_{k}^{n}}\) where \(\lambda-h(x)=\sum_{\gamma \in \mathbb{N}_{k}^{n}}\left(z_{\gamma}^{T} y\right) x^{\gamma} \quad \triangleright\) compute coefficients of \(\lambda-h\) in basis
    return \(\min \{\lambda: b=Z y, y=[\lambda, c], c \geq 0\} \quad \triangleright\) solve LP
```


### 2.5 QCQP reformulation and Shor's SDP relaxation

Another way of upper bounding $L\left(f_{d}\right)$ comes from a further relaxation of (4.2) to an SDP. We consider the following equivalent formulation where the variables $s_{i}$ are normalized to lie in the interval $[-1,1]$, and we rename $t=s_{0}$ :

$$
\begin{equation*}
L\left(f_{d}\right) \leq \max \left\{\frac{1}{2^{d-1}} s_{0}^{T} W_{1}^{T} \prod_{i=1}^{d-1} \operatorname{Diag}\left(s_{i}+1\right) W_{i+1}^{T}:-1 \leq s_{i} \leq 1\right\} \tag{2.11}
\end{equation*}
$$

Any polynomial optimization problem like (2.11) can be cast as a (possibly non-convex) quadratically constrained quadratic program (QCQP) by introducing new variables and quadratic constraints. This is a well-known procedure described in Park and Boyd (2017, Section 2.1). When $d=2$ problem (2.11) is already a QCQP (for the $\ell_{\infty}$ and $\ell_{2}$-norm cases) and no modification is necessary.

QCQP reformulation. We illustrate the case $d=3$ where we have the variables $s_{1}$, $s_{2}$ corresponding to the first and second hidden layer and a variable $s_{0}$ corresponding to the input. The norm-gradient polynomial in this case is cubic, and it can be rewritten as a quadratic polynomial by introducing new variables corresponding to the product of the first and second hidden layer variables.

More precisely the introduction of a variable $s_{1,2}$ with quadratic constraint $s_{1,2}=\operatorname{vec}\left(s_{1} s_{2}^{T}\right)$
allows us to write the objective (2.11) as a quadratic polynomial. The problem then becomes a QCQP with variable $y=\left[1, s_{0}, s_{1}, s_{2}, s_{1,2}\right]$ of dimension $1+n+n_{1} n_{2}$.

SDP relaxation. Any quadratic objective and constraints can then be relaxed to linear constraints on the positive semidefinite variable $y y^{T}=X \succcurlyeq 0$ yielding the so-called Shor's relaxation of (2.11) (Park and Boyd, 2017, Section 3.3). When $d=2$ the resulting SDP corresponds precisely to the one studied in Raghunathan et al. (2018a). This resolves a common misconception (Raghunathan et al., 2018c) that this approach is only limited to networks with one hidden layer.

Note that in our setting we are only interested in the optimal value rather than the optimizers, so there is no need to extract a solution for (2.11) from that of the SDP relaxation.

Drawback. This approach includes a further relaxation step from (2.11), thus being fundamentally limited in how tightly it can upper bound the value of $L\left(f_{d}\right)$. Moreover when compared to LP solvers, off-the-shelf semidefinite programming solvers are, in general, much more limited in the number of variables they can efficiently handle.

In the case $d=2$ this relaxation provides a constant factor approximation to the original QCQP (Ye, 1999). Further approximation quality results for such hierarchical optimization approaches to NP-hard problems are out of the scope of this work.

Relation to sum-of-squares. The QCQP approach might appear fundamentaly different to the hierarchical optimization approaches to POPs, like the one described in Section 2.3. However, it is known that Shor's SDP relaxation corresponds exactly to the first degree of the SOS hierarchical SDP solution to the QCQP relaxation (Lasserre, 2000). Thus, the approach in section 2.3 and the one in this section are, in essence, the same; they only differ in the choice of polynomial positivity certificate.

### 2.6 Related work

Estimation of $L\left(f_{d}\right)$ with $\ell_{2}$-norm is studied by Virmaux and Scaman (2018); Combettes and Pesquet (2019); Fazlyab et al. (2019a); Jin and Lavaei (2018). The method SeqLip proposed in Virmaux and Scaman (2018) has the drawback of not providing true upper bounds. It is in fact a heuristic method for solving (4.2) but which provides no guarantees and thus can not be used for robustness certification. In contrast the LipSDP method of Fazlyab et al. (2019a) provides true upper bounds on $L\left(f_{d}\right)$ and in practice shows superior performance over both SeqLip and CPLip (Combettes and Pesquet, 2019).

Despite the accurate estimation of LipSDP, its formulation is limited to the $\ell_{2}$-norm. The only estimate available for other $\ell_{p}$-norms comes from the equivalence of norms in euclidean spaces. For instance, we can obtain an upper bound for the $\ell_{\infty}$-norm after multiplying the $\ell_{2}$ Lipschitz constant upper bound by the square root of the input dimension. The resulting bound can be rather loose and our experiments in section 2.7 confirm the issue. In contrast,
our proposed approach LiPopt can acommodate any norm whose unit ball can be described via polynomial inequalities.

Let us point to one key advantage of LiPopt, compared to LipSDP (Jin and Lavaei, 2018; Fazlyab et al., 2019a). In the context of robustness certification we are given a sample $x^{\natural}$ and a ball of radius $\epsilon$ around it. Computing an upper bound on the local Lipschitz constant in this subset, rather than a global one, can provide a larger region of certified robustness. Taking into account the restricted domain we can refine the bounds in our POP (see remark in section 8.1). This potentially yields a tighter estimate of the local Lipschitz constant. On the other hand, it is not clear how to include such additional information in LipSDP, which only computes one global bound on the Lipschitz constant for the unconstrained network.

Raghunathan et al. (2018a) find an upper bound for $L\left(f_{d}\right)$ with $\ell_{\infty}$ metric starting from problem (4.2) but only in the context of one-hidden-layer networks ( $d=2$ ). To compute such bound they use its corresponding Shor's relaxation and obtain as a byproduct a differentiable regularizer for training networks. They claim such approach is limited to the setting $d=2$ but, as we remark in section 2.5 , it is just a particular instance of the SDP relaxation method for QCQPs arising from a polynomial optimization problem. We find that this method fits into the LiPopt framework, using SOS certificates instead of Krivine's. We expect that the SDP-based bounds described in 2.5 can also be used as regularizers promoting robustness.

Weng et al. (2018) provide an upper bound on the local Lipschitz constant for networks based on a sequence of ad-hoc bounding arguments, which are particular to the choice of ReLU activation function. In contrast, our approach applies in general to activations whose derivative is bounded.

### 2.7 Experiments

We consider the following estimators of $L\left(f_{d}\right)$ with respect to the $\ell_{\infty}$ norm:

| Name | Description |
| :---: | :--- |
| SDP | Upper bound arising from the solution of the SDP relaxation described <br> in Section 2.5 |
| LipOpt-k | Upper bound arising from the $k$-th degree of the LP hierarchy (2.6) <br> based on the sparse Krivine Positivstellenstatz. |
| Lip-SDP | Upper bound from Fazlyab et al. (2019a) multiplied $\sqrt{d}$ where $d$ is the <br> input dimension of the network. |
| UBP | Upper bound determined by the product of the layer-wise Lipschitz <br> constants with $\ell_{\infty}$ metric |
| LBS | Lower bound obtained by sampling 50000 random points around zero, <br> and evaluating the dual norm of the gradient |

### 2.7.1 Experiments on random networks

We compare the bounds obtained by the algorithms described above on networks with random weights and either one or two hidden layers. We define the sparsity level of a network as the maximum number of neurons any neuron in one layer is connected to in the next layer. For example, the network represented on Figure 2.1 has sparsity 2. The non-zero weights of network's $i$-th layer are sampled uniformly in $\left[-\frac{1}{\sqrt{n_{i}}}, \frac{1}{\sqrt{n_{i}}}\right]$ where $n_{i}$ is the number of neurons in layer $i$.

For different configurations of width and sparsity, we generate 10 random networks and average the obtained Lipschitz bounds. For better comparison, we plot the relative error. Since we do not know the true Lipschitz constant, we cannot compute the true relative error. Instead, we take as reference the lower bound given by LBS. Figures 2.3 and 2.5 show the relative error, i.e., $\left(\hat{L}-L_{L B S}\right) / L_{L B S}$ where $L_{L B S}$ is the lower bound computed by LBS and $\hat{L}$ is the estimated upper bound. Figures 2.9 and 2.10 in Appendix 2.10 we show the values of the computed Lipschitz bounds for 1 and 2 hidden layers respectively.

When the chosen degree for LiPopt-k is the smallest as possible, i.e., equal to the depth of the network, we observe that the method is already competitive with the SDP method, especially in the case of 2 hidden layers. When we increment the degree by l, LiPopt-k becomes uniformly better than SDP over all tested configurations. We remark that the upper bounds given by UBP are too large to be shown in the plots. Similarly, for the 1-hidden layer networks, the bounds from LipSDP are too large to be plotted.

Finally, we measured the computation time of the different methods on each tested network (Figures 2.4 and 2.6). We observe that the computation time for LiPopt-k heavily depends on the network sparsity, which reflects the fact that such structure is exploited in the algorithm. In contrast, the time required for SDP does not depend on the sparsity, but only on the size of the network. Therefore as the network size grows (with fixed sparsity level), LipOpt-k obtains a better upper bound and runs faster. Also, with our method, we see that it is possible to increase the computation power in order to compute tighter bounds when required, making it more flexible than SDP in terms of computation/accuracy tradeoff. LiPopt uses the Gurobi LP solver, while SDP uses Mosek. All methods run on a single machine with Core i7 2.8Ghz quad-core processor and 16 Gb of RAM.

Remark 2.1. We observe that a high level of sparsity in the network has two different positive effects. First, it allows efficient computation of a bound on its Lipschitz constant, by reducing the number of variables in our proposed optimization problems. More interestingly, a higher sparsity also induces a smaller value of the Lipschitz constant, making the network more robust to adversarial perturbations in the $\ell_{\infty}$ norm. This relation between the sparsity of the network and a smaller Lipschitz constant was made more explicit by Muthukumar and Sulam (2022), after the publication of this chapter.


Figure 2.3: Lipschitz approximated relative error for 1-hidden layer networks


Figure 2.4: Computation times for 1-hidden layer networks (seconds)


Figure 2.5: Lipschitz approximated relative error for 2-hidden layer networks


Figure 2.6: Computation times for 2-hidden layer networks (seconds)

### 2.7.2 Experiments on trained networks

Similarly, we compare these methods on networks trained on MNIST. The architecture we use is a fully connected network with two hidden layers with 300 and 100 neurons respectively, and with one-hot output of size 10 . Since the output is multi-dimensional, we restrict the network to a single output, and estimate the Lipschitz constant with respect to label 8.

Moreover, in order to improve the scalability of our method, we train the network using the pruning strategy described in Han et al. (2015) ${ }^{2}$. After training the full network using a standard

[^2]technique, the weights of smallest magnitude are set to zero. Then, the network is trained for additional iterations, only updating the nonzero parameters. Doing so, we were able to remove $95 \%$ of the weights, while preserving the same test accuracy. We recorded the Lipschitz bounds for various methods in Table 2.7.2. We observe clear improvement of the Lipschitz bound obtained from LiPopt-k compared to SDP method, even when using $k=3$. Also note that the input dimension is too large for the method Lip-SDP to provide competitive bound, so we do not provide the obtained bound for this method.

| Algorithm | LBS | LiPopt-4 | LiPopt-3 | SDP | UBP |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Lipschitz bound | 84.2 | 88.3 | 94.6 | 98.8 | 691.5 |

### 2.7.3 Estimating local Lipschitz constants with LiPopt

We study the improvement on the upper bound obtained by LiPopt, when we incorporate tighter upper and lower bounds on the variables $s_{i}$ of the polynomial optimization problem (4.2). Such bounds arise from the limited range that the pre-activation values of the network can take, when the input is limited to an $\ell_{\infty}$-norm ball of radius $\epsilon$ centered at an arbitrary point $x_{0}$.

The algorithm that computes upper and lower bounds on the pre-activation values is fast (it has the same complexity as a forward pass) and is described, for example, in Wong and Kolter (2018a). The variables $s_{i}$ correspond to the value of the derivative of the activation function. For activations like ELU or ReLU, their derivative is monotonically increasing, so we need only evaluate it at the upper and lower bounds of the pre-activation values to obtain corresponding bounds for the variables $s_{i}$.

We plot the local upper bounds obtained by LiPopt-3 for increasing values of the radius $\epsilon$, the bound for the global constant (given by LiPopt-3) and the lower bound on the local Lipschitz constant obtained by sampling in the $\epsilon$-neighborhood (LBS). We sample 15 random networks and plot the average values obtained. We observe clear gap between both estimates, which shows that larger certified balls could be obtained with such method in the robustness certification applications.


Figure 2.7: Global vs local Lipschitz constant bounds for 1-hidden layer networks

[^3]

Figure 2.8: Global vs local Lipschitz constant bounds for 2-hidden layer networks

### 2.8 Appendix: Proof of Theorem 2.1

Theorem. Let $f$ be a differentiable and Lipschitz continuous function on an open, convex subset $\mathscr{X}$ of a euclidean space. Let $\|\cdot\|$ be the dual norm. The Lipschitz constant of $f$ is given by

$$
\begin{equation*}
L(f)=\sup _{x \in \mathscr{X}}\|\nabla f(x)\|_{*} \tag{2.12}
\end{equation*}
$$

Proof. First we show that $L(f) \leq \sup _{x \in \mathscr{X}}\|\nabla f(x)\|_{*}$.

$$
\begin{aligned}
|f(y)-f(x)| & =\left|\int_{0}^{1} \nabla f((1-t) x+t y)^{T}(y-x) d t\right| \\
& \leq \int_{0}^{1}\left|\nabla f((1-t) x+t y)^{T}(y-x)\right| d t \\
& \leq \int_{0}^{1}\|\nabla f((1-t) x+t y)\|_{*} d t\|y-x\| \\
& \leq \sup _{x \in \mathscr{X}}\|\nabla f(x)\|_{*}\|y-x\|
\end{aligned}
$$

were we have used the convexity of $\mathscr{X}$.
Now we show the reverse inequality $L(f) \geq \sup _{x \in \mathscr{\mathscr { C }}}\|\nabla f(x)\|_{*}$. To this end, we show that for any positive $\epsilon$, we have that $L(f) \geq \sup _{x \in \mathscr{X}}\|\nabla f(x)\|_{*} \begin{gathered}x \in \mathscr{X} \\ \text {. }\end{gathered}$

Let $z \in \mathscr{X}$ be such that $\|\nabla f(z)\|_{*} \geq \sup _{x \in \mathscr{X}}\|\nabla f(x)\|_{*}-\epsilon$. Because $\mathscr{X}$ is open, there exists a sequence $\left\{h_{k}\right\}_{k=1}^{\infty}$ with the following properties:

1. $\left\langle h_{k}, \nabla f(z)\right\rangle=\left\|h_{k}\right\|\|\nabla f(z)\|_{*}$
2. $z+h_{k} \in \mathscr{X}$
3. $\lim _{k \rightarrow \infty} h_{k}=0$.

By definition of the gradient, there exists a function $\delta$ such that $\lim _{h \rightarrow 0} \delta(h)=0$ and the
following holds:

$$
f(z+h)=f(z)+\langle h, \nabla f(z)\rangle+\delta(h)\|h\|
$$

For our previously defined iterates $h_{k}$ we then have

$$
\Rightarrow\left|f\left(z+h_{k}\right)-f(z)\right|=\left|\left\|h_{k}\right\|\|\nabla f(z)\|_{*}+\delta\left(h_{k}\right)\left\|h_{k}\right\|\right|
$$

Dividing both sides by $\left\|h_{k}\right\|$ and using the definition of $L(f)$ we finally get

$$
\begin{aligned}
& \Rightarrow L(f) \geq\left|\frac{f\left(z+h_{k}\right)-f(z)}{\left\|h_{k}\right\|}\right|=\left|\|\nabla f(z)\|_{*}+\delta\left(h_{k}\right)\right| \\
& \Rightarrow L(f) \geq \lim _{k \rightarrow \infty}\left|\|f(z)\|_{*}+\delta\left(h_{k}\right)\right|=\|\nabla f(z)\|_{*} \\
& \Rightarrow L(f) \geq \sup _{x \in \mathscr{C}}\|\nabla f(x)\|_{*}-\epsilon
\end{aligned}
$$

### 2.9 Appendix: Proof of Proposition Proposition 2.1

Proposition. Let $f_{d}$ be a dense network (all weights are nonzero). The following sets, indexed by $i=1, \ldots, n_{d}$, form a valid sparsity pattern for the norm-gradient polynomial of the network $f_{d}$ :

$$
\begin{equation*}
I_{i}:=\left\{s_{(d-1, i)}\right\} \cup\left\{s_{(j, k)}: \text { there exists a directed path from } s_{(j, k)} \text { to } s_{(d-1, i)} \text { in } G_{d}\right\} \tag{2.13}
\end{equation*}
$$

Proof. First we show that $\cup_{i=1}^{m} I_{i}=I$. This comes from the fact that any neuron in the network is connected to at least one neuron in the last layer. Otherwise such neuron could be removed from the network altogether.

Now we show the second property of a valid sparsity pattern. Note that the norm-gradient polynomial is composed of monomials corresponding to the product of variables in a path from input to a final neuron. This imples that if we let $p_{i}$ be the sum of all the terms that involve the neuron $s_{(d-1, i)}$ we have that $p=\sum_{i} p_{i}$, and $p_{i}$ only depends on the variables in $I_{i}$.

We now show the last property of the valid sparsity pattern. This is the only part where we use that the network is dense. For any network architecture the first two conditions hold. We will use the fact that the maximal cliques of a chordal graph form a valid sparsity pattern (see for example Lasserre (2006)).

Because the network is dense, we see that the clique $I_{i}$ is composed of the neuron in the last layer $s_{(d-1, i)}$ and all neurons in the previous layers. Now consider the extension of the
computational graph $\hat{G}_{d}=(V, \hat{E})$ where

$$
\left.\hat{E}=E \cup\left\{\left(s_{j, k}, s_{l, m}\right): j, l \leq d-2\right)\right\}
$$

which consists of adding all the edges between the neurons that are not in the last layer. We show that this graph is chordal. Let $\left(a_{1}, \ldots, a_{r}, a_{1}\right)$ be a cycle of length at least $4(r \geq 4)$. notice that because neurons in the last layer are not connected between them in $\hat{G}$, no two consecutive neurons in this cycle belong to the last layer. This implies that in the subsequence $\left(a_{1}, a_{2}, a_{3}, a_{4}, a_{5}\right)$ at most three belong to the last layer. A simple analysis of all cases implies that it contains at least two nonconsecutive neurons not in the last layer. Neurons not in the last layer are always connected in $\hat{G}$. This constitutes a chord. This shows that $\hat{G}_{d}$ is a chordal graph. Its maximal cliques correspond exactly to the sets in proposition.

### 2.10 Appendix: Experiments on random networks



Figure 2.9: Lipschitz bound comparison for 1-hidden layer networks


Figure 2.10: Lipschitz bound comparison for 2-hidden layer networks

### 2.11 Bibliographic Note

The candidate came up with the idea of expressing the Lipschitz constant computation as a POP, and all the theoretical results i.e., theorem 2.1, proposition 2.1 and eq. (4.2), which leads to the algorithm. I designed the experiments and obtained the numerical results with the help of P. Rolland who also contributed by scaling the algorithm in the case of sparse networks.

# 3 Efficient Proximal Mapping of the 1-path-norm of Shallow Networks 

Fabian Latorre, Paul Rolland, Nadav Hallak and Volkan Cevher. International Conference on Machine Learning (ICML) 2020.


#### Abstract

We demonstrate two new important properties of the 1-path-norm of shallow neural networks. First, despite its non-smoothness and non-convexity it allows a closed form proximal operator which can be efficiently computed, allowing the use of stochastic proximal-gradient-type methods for regularized empirical risk minimization. Second, when the activation functions is differentiable, it provides an upper bound on the Lipschitz constant of the network. Such bound is tighter than the trivial layer-wise product of Lipschitz constants, motivating its use for training networks robust to adversarial perturbations. In practical experiments we illustrate the advantages of using the proximal mapping and we compare the robustness-accuracy trade-off induced by the 1-path-norm, L1-norm and layer-wise constraints on the Lipschitz constant (Parseval networks).


### 3.1 Introduction

Neural networks are the backbone of contemporary applications in machine learning and related fields, having huge influence and significance both in theory and practice. Among the most important and desirable attributes of a trained network are robustness and sparsity. Robustness, is often defined as stability to adversarial perturbations, such as in supervised classification methods. The apparent brittleness of neural networks to adversarial attacks in this context has been considered in the literature for some time, see e.g., (Biggio et al., 2013a; Szegedy et al., 2013; Madry et al., 2018b) and references therein.

A fundamental question in this regard is how to measure robustness, or more importantly, how to encourage it. One prominent approach supported by theory and practice (Raghunathan et al., 2018b; Cisse et al., 2017b), is to use the Lipschitz constant of the network function to quantize robustness, and regularization to encourage it.

This approach is also supported theoretically with generalization bounds in terms of the layer-

## Chapter 3. Efficient Proximal Mapping of the 1-path-norm of Shallow Networks

wise product of spectral norms (Bartlett et al., 2017b; Miyato et al., 2018a), which particularly upper-bounds the Lipschitz constant. However, a recent empirical study (Jiang et al., 2020) has found in practice a negative correlation of this measure with generalization. This casts doubts on its usefulness and signals the fact that it is a rather loose upper bound for the Lipschitz constant (Latorre et al., 2020b).

Current methods that compute upper bounds on the Lipschitz constant of neural networks can be roughly classified into two classes: (i) the class of product bounds, comprising all upper bounds obtained by the multiplication of layer-wise matrix norms; and, (ii) the class of convex-optimization-based bounds, which addresses the network as a whole entity (Raghunathan et al., 2018b; Fazlyab et al., 2019a; Latorre et al., 2020b).

A trade-off between computational complexity and quality of the upper bound seems apparent. An ideal bound would achieve a balance between both properties: it should provide a good estimate of the constant while being fast and easy to minimize with iterative first-order algorithms.

Recently, the path-norm of the network (Neyshabur et al., 2015c) has emerged as a complexity measure that is highly-correlated with generalization (Jiang et al., 2020). Thus, its use as a regularizer holds an increasing interest for researchers in the field.

Despite existing generalization bounds (Neyshabur et al., 2015c), our understanding of the optimization aspects of the path-norm-regularized objective is lacking. Jiang et al. (2020) refrained from using automatic-differentiation methods in this case because, as they argue, the optimization could fail, thus providing no conclusion about its qualities.

It is then natural to ask: how do we properly optimize the path-norm-regularized objective with theoretical guarantees? What conclusions can we draw about the robustness and sparsity of path-norm-regularized networks? We focus on the 1-path-norm and provide partial answers to those questions, further advancing our understanding of this measure. Let us summarize our main contributions:

Optimization. We show a striking property of the 1-path-norm, that makes it a strong candidate for explicit regularization: despite its non-convexity, it admits an efficient proximal mapping (Algorithm 3.3). This allows the use of proximal-gradient type methods which are, as of now, the only first-order optimization algorithms to provide guarantees of convergence for composite non-smooth and non-convex problems (Bolte et al., 2013).

Indeed, automatic differentiation modules of popular deep learning frameworks like PyTorch (Paszke et al., 2019) or TensorFlow (Abadi et al., 2015) may not compute the correct gradient for compositions of non-smooth functions, at points where these are differentiable (Kakade and Lee, 2018; Bolte and Pauwels, 2019). Our proposed optimization algorithm avoids such issue altogether by using differentiable activation functions like ELU (Clevert et al., 2015) and our novel proximal mapping of the 1-path-norm.

Upper bounds. We show that the 1-path-norm (Neyshabur et al., 2015c) achieves a sweet spot in the computation-quality trade-off observed among upper bounds of the Lipschitz constant: it has a simple closed formula in terms of the weights of the network, and it provides an upper bound on the ( $\ell_{\infty}, \ell_{1}$ )-Lipschitz constant (cf., theorem 3.1), which is always better than the product bound.

Sparsity. Neural network regularization schemes promoting sparsity in a principled way are of great interest in the growing field of compression in Deep Learning (Han et al., 2016; Cheng et al., 2017).

Our analysis provides a formula (cf. Lemma lemma 3.4) for choosing the strength of the regularization, which enforces a desired bound on the sparsity level of the iterates generated by the proximal gradient method. This is a suprising, yet intuitive, result, as the sparsity-inducing properties of non-smooth regularizers have been observed before in convex optimization and signal processing literature, see e.g., (Bach et al., 2012; Eldar and Kutyniok, 2012).

Experiments. In section 3.7, we present numerical evidence that our approach (i) converges faster and to lower values of the objective function, compared to plain SGD; (ii) generates sparse iterates; and, (iii) the magnitude of the regularization parameter of the 1-path-norm allows a better accuracy-robustness trade-off than the common $\ell_{1}$ regularization or constraints on layer-wise matrix norms.

### 3.2 Problem Setup

We consider the so-called shallow neural networks with $n$ hidden neurons and $p$ outputs $h: \mathbb{R}^{m} \rightarrow \mathbb{R}^{p}$ given by

$$
\begin{equation*}
h_{V, W}(x)=V^{T} \sigma(W x) \tag{3.1}
\end{equation*}
$$

where $V \in \mathbb{R}^{n \times p}, W \in \mathbb{R}^{n \times m}$ and $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ is some differentiable activation function with derivative globally bounded between zero and one. This condition is satisfied, for example, by the ELU or softplus activation functions. To control the robustness of the network to perturbations of its input $x$, we want to regularize training using its Lipschitz constant as a function of the weights $V$ and $W$.

To properly define this constant, we utilize the $\ell_{\infty}$-norm for the input space, and the $\ell_{1}$-norm for the output space. Exact computation of such constant is a hard task. A simple and easily computable upper bound can be derived by the product of the layer-wise Lipschitz constants, however, it can be quite loose.

We derive an improved upper bound which is still easy to compute. In the following, we denote with $\|W\|_{\infty}$ the operator norm of a matrix $W$ with respect to the $\ell_{\infty}$ norm for both input and output space; it is equal to the maximum $\ell_{1}$-norm of its rows. We denote with $\|V\|_{\infty, 1}$ the operator norm of the matrix $V$ with respect to the $\ell_{\infty}$ norm in input space and $\ell_{1}$-norm in output space; it is equal to the sum of the $\ell_{1}$ norm of its columns.

## Chapter 3. Efficient Proximal Mapping of the 1-path-norm of Shallow Networks

Theorem 3.1. Let $h_{V, W}(x)=V^{T} \sigma(W x)$ be a network such that the derivative of the activation $\sigma$ is globally bounded between zero and one. Choose the $\ell_{\infty^{-}}$and $\ell_{1}$-norm for input and output space, respectively. The Lipschitz constant of the network, denoted by $L_{V, W}$ is bounded as follows:

$$
\begin{equation*}
L_{V, W} \leq \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i j} V_{i k}\right| \leq\left\|V^{T}\right\|_{\infty, 1}\|W\|_{\infty} \tag{3.2}
\end{equation*}
$$

The proof is provided in appendix 3.8. The term in the middle of inequality (4.2) belongs to the family of path-norms, introduced in Neyshabur et al. (2015c, Eq. (7)). Throughout, we refer to it as the 1-path-norm.

Notice that although the path-norm and layer wise product bounds can be equal, this only happens in the following worst case: For the weight matrix in the first layer, the 1-norms of every row are equal. Thus, in practice the bounds can differ drastically.

Remark 3.1. In practice, one might want to regularize each ouput of the network in a different way according to some weighting scheme (Raghunathan et al., 2018b). Precisely, the 1-pathnorm of the network is equal to the sum (with equal weight) of the 1-path-norm of each output. A weighted version of the 1-path-norm can be defined to account for such a weighting scheme. All our results can be adapted to this scenerio, with minor changes.

We now turn to the task of minimizing an empirical risk functional regularized by the improved upper bound on the Lipschitz constant given in (4.2):

$$
\begin{equation*}
\min _{V, W} \mathbb{E}_{(x, y)}\left[\ell\left(h_{V, W}(x), y\right)\right]+\lambda \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i j} V_{i k}\right| \tag{3.3}
\end{equation*}
$$

The objective function in problem (3.3) is composed of an expectation of a nonconvex smooth loss, and a nonconvex nonsmooth regularizer, meaning that it is essentially a composite problem (cf. (Beck, 2017, Ch. 10)). That is, the objective function (3.3) can be cast as use these notation hereafter)

$$
\begin{equation*}
\min _{V, W} \mathscr{F}(V, W) \equiv f(V, W)+\lambda g(V, W), \quad g(V, W)=\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i j} V_{i k}\right| \tag{3.4}
\end{equation*}
$$

where $f$ is a nonconvex continuously differentiable function, and $g$ is a continuous, nonconvex, nonsmooth, function. We assume that the objective function is bounded below, i.e., $\inf \mathscr{F}:=\mathscr{F}_{*}>0$.

A natural choice for a scheme to obtain critical points for (3.4) is the proximal-gradient framework. However, for a nonconvex $g$, solving the proximal gradient problem is a hard problem in general. In Section 3.4 we develop a method that computes the proximal gradient with respect to $g$ efficiently.

To streamline our approach and techniques in a compact and user-friendly manner, we will
illustrate the majority of our results and proofs via the particular single-output scenario in which $h$ and $g$ are reduced to

$$
h_{\nu, W}(x)=v^{T} \sigma(W x), g(\nu, W) \equiv \lambda\|\operatorname{vec}(\operatorname{Diag}(\nu) W)\|_{1} .
$$

The multi-output case follows from the same techniques and insights, however, requires more tedious computations and arguments, on which we elaborate in Section 3.5, and detail in the appendix.

### 3.3 The Prox-Grad Method

Assume that $f$ has a Lipschitz continuous gradient with Lipschitz constant $L>0$, that is

$$
\|\nabla f(z)-\nabla f(u)\| \leq L\|z-u\|, \quad \forall z, u \in \mathbb{R}^{n} .
$$

The prox-grad method is described by Algorithm 3.1; since $g$ is nonconvex, the prox in (3.5) can be a set of solutions.

```
Algorithm 3.1 Prox-Grad Method
Input: }\mp@subsup{z}{}{0}\equiv\operatorname{vec}(\mp@subsup{V}{}{0},\mp@subsup{W}{}{0})\in\mp@subsup{\mathbb{R}}{}{p\cdotn+n\cdotm},{\mp@subsup{\eta}{}{k}\mp@subsup{}}{k\geq0}{}
    for }k=0,1,\ldots\mathrm{ do
        Compute G}\mp@subsup{}{}{k}=\nablaf(\mp@subsup{z}{}{k}
        z
    end for
```

Theoretical guarantees for the prox-grad method with respect to a nonconvex regularizer were established by Bolte et al. (2013) (for a more general prox-grad type scheme).

Theorem 3.2 (Convergence guarantees). Let $\left\{z^{k}\right\}_{k \geq 0}$ be a sequence generated by Algorithm 3.1 with $\left\{\eta^{k}\right\}_{k \geq 0} \subseteq(0,1 / L)$. Then

1. Any accumulation point of $\left\{z^{k}\right\}_{k \geq 0}$ is a critical point of (3.4).
2. If f satisfies the Kurdyka-Lojasiewicz (KL) property, then $\left\{z^{k}\right\}_{k \geq 0}$ converges to a critical point.
3. Suppose that $\eta_{k}$ is chosen such that there exists $c>0$ such that $\sum_{k=0}^{K} \frac{1}{\eta_{k}} \geq c K$ for any $K \geq 0$. Then

$$
\min _{k=0, \ldots, K}\left\|z^{k+1}-z^{k}\right\|_{2} \leq \sqrt{\frac{2\left(\mathscr{F}\left(z^{0}\right)-\mathscr{F}_{*}\right)}{(c-L) K}} .
$$

Proof. See Section 3.9 in the appendix.

Remark 3.2 (On KL related convergence rate). A convergence rate result under the KL property can be derived with respect to the desingularizing function; see Bolte et al. (2013) for additional details.

Remark 3.3 (On the stochastic prox-grad method). The literature does not provide any theoretical guarantees for a prox-grad type method that uses stochastic gradients (i.e., replacing $G^{k}$ with an approximation of $\left.\nabla f\left(z^{k}\right)\right)$ under our setting. Recently, Metel and Takeda (2019) studied stochastic prox-grad methods, however, their results rely on the assumption that the regularizer is Lipschitz continuous, which is not satisfied by our robust-sparsity regularizer.

### 3.4 Computing the Proximal Mapping

Throughout this section we assume the single-output setting. The path-norm regularizer we propose is a nonconvex nonsmooth function, suggesting that the prox-grad scheme in Algorithm 3.1 is intractable.

In this section we will not only prove that in fact it is tractable in the single output case, but that it can also be implemented efficiently with complexity of $O(m \log (m)$ ); we prove the stated in detail in Section 3.10, and provide here a concise version.

Denote the given pair $(x, Y)$ by $z$. The proximal mapping with respect to $\lambda g$ at $z$ is defined as

$$
\begin{equation*}
\operatorname{prox}_{\lambda g}(z)=\underset{u}{\operatorname{argmin}} \lambda g(u)+\frac{1}{2}\|\operatorname{vec}(u-z)\|_{2}^{2} . \tag{3.5}
\end{equation*}
$$

By the choice of $g$, the objective function in (3.5) is coercive and lower bounded, implying that there exists an optimal solution (cf. (Beck, 2014, Thm. 2.32)).

Remark 3.4. The derivations in this section can be easily adapted and used with adaptive gradient methods like Adagrad (Duchi et al., 2011), by a careful handling of the per-coordinate scaling coefficients.

Lemma 3.1 (Well-posedness of (3.5)). For any $\lambda \geq 0$ and any ( $u, z$ ), the problem (3.5) has a global optimal solution.

Additionally, we have that (3.5) is separable with respect to the $i$-th entry of the vector $v$ and the $i$-th row of the matrix $W$, meaning that problem (3.5) can be solved in a distributed manner by applying the same solution procedure coordinate-wise for $v$ and row-wise for $W$. In light of this, let us consider the $i$-th row related problem

$$
\begin{equation*}
\min _{v, w \in \mathbb{R} \times \mathbb{R}^{m}} \frac{1}{2}(\nu-x)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-y_{j}\right)^{2}+\lambda|\nu| \sum_{j=1}^{m}\left|w_{j}\right| . \tag{3.6}
\end{equation*}
$$

The signs of the elements of the decision variables in (3.6) are determined by the signs of ( $x, y$ ),
and consequently, the problem in (3.6) is equivalent to

$$
\begin{equation*}
\min _{\nu, w \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}} \frac{1}{2}(\nu-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda v \sum_{j=1}^{m} w_{j} . \tag{3.7}
\end{equation*}
$$

Lemma 3.2. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{n}$ be an optimal solution of $(3.7)$. Then $\left(\operatorname{sign}(x) \cdot v^{*}, \operatorname{sign}(y) \circ\right.$ $w^{*}$ ) is an optimal solution of problem (3.6).

Denote

$$
h_{\lambda}(\nu, w ; x, y)=\frac{1}{2}(\nu-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda v \sum_{j=1}^{m} w_{j} .
$$

Although $h_{\lambda}$ is nonconvex, we will show that a global optimum to (3.7) can be obtained efficiently by utilizing several tools, the first being the first-order optimality conditions of (3.7) (cf. (Beck, 2014, Ch. 9)) given below.
Lemma 3.3 (Stationarity conditions). Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.7) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$. Then

$$
\begin{aligned}
w_{j}^{*} & =\max \left\{0,\left|y_{j}\right|-\lambda v^{*}\right\} \text { for any } j=1,2, \ldots, m, \\
v^{*} & =\max \left\{0,|x|-\lambda \sum_{j=1}^{m} w_{j}^{*}\right\} .
\end{aligned}
$$

A key insight following Lemma 3.3 is that: the elements of any solution to (3.7), satisfy a monotonic relation in magnitude, correlated with the magnitude of the elements of $y$; this is formulated by the next corollary.

Corollary 3.1. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.7) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$. Then

1. The vector $w^{*}$ satisfies that for any $j, l \in\{1,2, \ldots, m\}$ it holds that $w_{j}^{*} \geq w_{l}^{*}$ only if $\left|y_{j}\right| \geq$ $\left|y_{l}\right|$.
2. Let $\bar{y}$ be the sorted vector of $y$ in descending magnitude order. Suppose that $v^{*}>0$ and let $s=\left|\left\{j: s w_{j}^{*}>0\right\}\right|$. Then,

$$
\begin{equation*}
v^{*}=\frac{1}{1-s \lambda^{2}}\left(|x|-\lambda \sum_{j=1}^{s}\left|\bar{y}_{j}\right|\right), \tag{3.8}
\end{equation*}
$$

where we use the convention that $\sum_{j=1}^{0}\left|\bar{y}_{j}\right|=0$.

Proof. The first part follows trivially from the stationarity conditions on $w^{*}$ given in Lemma 3.3.

From the first part and the conditions in Lemma 3.3 we have that $\sum_{j=1}^{m} w_{j}^{*}=\sum_{j=1}^{s}\left|\bar{y}_{j}\right|-\lambda s v^{*}$. Plugging the latter to the stationarity condition on $v^{*}$ (given in Lemma 3.3) then implies the required.

Remark 3.5. Corollary 3.1 implies that the solution vector $w^{*}$ is ordered in the same way as $|y|$. Thus, the s non-zero entries of $w^{*}$ are precisely the ones corresponding with the s largest entries $o f|y|$.

Without loss of generality, we assume hereafter that the input y is already sorted in decreasing order, such that the s non-zero entries of $w^{*}$ are always the first s entries.

To supplement the results above, we now show that we can actually upper-bound the sparsity level of the prox-grad output by adjusting the value of $\lambda$.

Lemma 3.4 (Sparsity bound). Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.7) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$. Suppose that $v^{*}>0$ (i.e., non-trivial), ${ }^{1}$ and denote $S=\left\{j: w_{j}^{*}>0\right\}$. Then $|S| \leq \lambda^{-2}$.

Proof. Since ( $v^{*}, w^{*}$ ) is an optimal solution of (3.7) and the objective function in (3.7) is twice continuously differentiable, $\left(v^{*}, w^{*}\right)$ satisfies the second order necessary optimality conditions (Bertsekas, 1999, Ex. 2.1.10). That is, for any $d \in \mathbb{R} \times \mathbb{R}^{m}$ satisfying that ( $\left.\nu^{*}, w^{*}\right)+d \in$ $\mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ and $d^{T} \nabla h_{\lambda}\left(v^{*}, w^{*} ; x, y\right)=0$ it holds that

$$
d^{T} \nabla^{2} h_{\lambda}\left(v^{*}, w^{*} ; x, y\right) d=d^{T}\left(\begin{array}{cccc}
1 & \lambda & \cdots & \lambda \\
\lambda & 1 & 0 & 0 \\
\vdots & 0 & \ddots & 0 \\
\lambda & 0 & 0 & 1
\end{array}\right) d \geq 0
$$

where the first row/column corresponds to $v$ and the others correspond to $w$. Noting that for any $j \in S$ it holds that $\frac{\partial h_{\lambda}}{\partial w_{j}}\left(v^{*}, w^{*} ; x, y\right)=0$, we have that the submatrix of $\nabla^{2} h_{\lambda}\left(v^{*}, w^{*} ; x, y\right)$ containing the rows and columns corresponding to the positive coordinates in ( $\nu^{*}, w^{*}$ ) must be positive semidefinite.

Since the the minimal eigenvalue of this submatrix equals $1-\lambda \sqrt{|S|}$, we have that $\lambda^{-2} \geq|S|$.

Moreover, the function $h_{\lambda}$ is monotonically decreasing in the sparsity level, which implies that instead of exhaustively checking the value of $h_{\lambda}$ for any sparsity level, we can employ a binary search. Denote for any $s \in\{0, \ldots, m\}$ the $m+1$ possible solutions:

$$
\begin{aligned}
v^{(s)} & =\frac{1}{1-s \lambda^{2}}\left(|x|-\lambda \sum_{j=1}^{s}\left|y_{j}\right|\right) \\
w_{j}^{(s)} & =\left|y_{j}\right|-\lambda v^{(s)} \text { for } j \in[s], \text { and } w_{j}^{(s)}=0 \text { otherwise. }
\end{aligned}
$$

Lemma 3.5. Let $\bar{s}=\left\lfloor\lambda^{-2}\right\rfloor$. For all integer $s \in\{2,3, \ldots, \bar{s}\}$, we have that

$$
\begin{equation*}
h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right)<h_{\lambda}\left(v^{(s-1)}, w^{(s-1)} ; x, y\right) \tag{3.9}
\end{equation*}
$$

[^4]Lemma 3.5 follows from algebraic considerations, and thus its proof is deferred to Section 3.10. Its substantial implication is the following.

Corollary 3.2. Suppose that there exists a non-trivial optimal solution of (3.7). Denote $\bar{s}=$ $\min \left(\left\lfloor\lambda^{-2}\right\rfloor, m\right)$ and let

$$
s^{*}=\max \left\{s \in\{0, \ldots, \bar{s}\}: v^{(s)}, w_{s}^{(s)}>0\right\}
$$

Then $\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$ is an optimal solution of (3.7).

Note that since, by definition, the $s$ first entries of the vector $w^{(s)}$ are ordered in decreasing order, the constrained $w_{s}^{(s)}>0$ ensures that the full vector $w^{(s)}$ has exactly $s$ nonzero entries, which are all strictly positive.

The final ingredient required for designing an efficient algorithm is the following monotone property of the feasibility criterion in problem (3.2):

Lemma 3.6. For any $k \in[\bar{s}]$, we have

$$
v^{(k)}>0, w^{(k)}>0 \Rightarrow v^{(i)}>0, w^{(i)}>0, \quad \forall i<k .
$$

This property, whose proof is also deferred to Section 3.10, implies that the optimal sparsity parameter $s^{*}$ can be efficiently found using a binary search approach.

We conclude this section by combining all the ingredients above to develop Algorithm 3.2, and to prove that it yields a solution to (3.5).

```
Algorithm 3.2 Single-output robust-sparse proximal mapping
Input: \(x \in \mathbb{R}, y \in \mathbb{R}^{m}\) sorted in decreasing magnitude order, \(\lambda>0\).
    \(v^{*}=0, w^{*}=|y|\)
    \(s_{\mathrm{lb}} \leftarrow 0, s_{\mathrm{ub}} \leftarrow \min \left(\left\lfloor\lambda^{-2}\right\rfloor, m\right), s \leftarrow\left\lceil\left(s_{\mathrm{lb}}+s_{\mathrm{ub}}\right) / 2\right\rceil\)
    while \(s_{\mathrm{lb}} \neq s_{\mathrm{ub}}\) do
        \(v^{(s)}=\frac{1}{1-s \lambda^{2}}\left(|x|-\lambda \sum_{j=1}^{s}\left|y_{j}\right|\right)\)
        \(w_{j}^{(s)}=\left|y_{j}\right|-\lambda v^{(s)}, j \in[s]\) and \(w_{j}^{(s)}=0\) otherwise
        if \(v>0, w_{s}>0\) then
            \(s_{\mathrm{lb}} \leftarrow s, s \leftarrow\left\lceil\left(s_{\mathrm{lb}}+s_{\mathrm{ub}}\right) / 2\right\rceil\)
            \(\left(v^{*}, w^{*}\right) \leftarrow(v, w)\)
        else if \(v<0\) then \(s_{\mathrm{ub}} \leftarrow s, s \leftarrow\left\lceil\left(s_{\mathrm{lb}}+s_{\mathrm{ub}}\right) / 2\right\rceil\)
        else \(s_{\mathrm{lb}} \leftarrow s, s \leftarrow\left\lceil\left(s_{\mathrm{lb}}+s_{\mathrm{ub}}\right) / 2\right\rceil\)
        end if
    end while
    return \(\left(\operatorname{sign}(x) \cdot v^{*}, \operatorname{sign}(y) \circ w^{*}\right)\)
```

Theorem 3.3 (Prox computation). Let $\left(v_{i}^{*}, W_{i,:}^{*}\right)$ be the output of Algorithm 3.2 with input $x_{i}, Y_{i,:}, \lambda$, assuming that each $Y_{i,:}$ is sorted in decreasing magnitude order. Then ( $\left.v^{*}, W^{*}\right)$ is a solution to (3.5).

Proof. For any $i=1,2, \ldots, n$, let $\left(v_{i}^{*}, W_{i,:}^{*}\right)$ be the output of Algorithm 3.2 with input $x_{i}, Y_{i,:}, \lambda$. We will show that $\left(v^{*}, W^{*}\right)$ is an optimal solution to (3.5) by arguing that Algorithm 3.2 chooses the point with the smallest $h_{\lambda}$ value out of a feasible set of solutions containing an optimal solution of (3.5).

For simplicity, and without loss of generality, let us consider the one-coordinate-one-row case, that is, $\left(v_{i}^{*}, W_{i,:}^{*}\right) \equiv\left(v^{*}, w^{*}\right),\left(x_{i}, Y_{i,:}\right) \equiv(x, y)$; the proof for the general case is a trivial replication.

By Lemma 3.2 it is sufficient to prove that $\left(\left|v^{*}\right|,\left|w^{*}\right|\right)$ is an optimal solution of (3.7), as this will imply the optimality of ( $v^{*}, w^{*}$ ); Recall that Lemma 3.1 establishes that there exists an optimal solution to (3.7).

If the trivial solution is the only optimal solution to (3.7), then obviously it will be the output of Algorithm 3.2. Otherwise, the point described in Corollary 3.2 is an optimal solution. Assume that Algorithm 3.2 returned the point ( $v^{\left(s_{\text {out }}\right)}, w^{\left(s_{\text {out }}\right)}$ ) for some $s_{\text {out }} \in[\bar{s}]$, meaning in particular that $\left(\nu^{\left(s_{\text {out }}\right)}, w^{\left(s_{\text {out }}\right)}\right)>0$. By definition, $s^{*} \geq s_{\text {out }}$. If $s_{\text {out }}<s^{*}$, then at some $s<s^{*}$ we had that $v^{(s)}<0$. Since the value of $v^{(i)}$ is monotonic decreasing in the sparsity level, this implies that $v^{\left(s^{*}\right)}<0$, which is a contradiction.

Hence, if Algorithm 3.2 did not return the trivial solution, then $\left(v^{*}, w^{*}\right)=\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$, meaning that $\left(\operatorname{sign}(x) \cdot v^{*}, \operatorname{sign}(y) \circ w^{*}\right)$ is a solution to (3.5).

Time complexity of Algorithm 3.2. In the worst case where $m \leq \lambda^{-2}$, the number of searches for finding $s^{*}$ is at $\operatorname{most}^{\log _{2}(m)}$. Each search requires to compute $v^{(s)}$, and in particular $\sum_{j=1}^{s}\left|y_{j}\right|$, as well as $w_{j}^{(s)}, j=1, \ldots, s$, each taking $\mathscr{O}(s)$ steps. Thus, the overall loop complexity is $\mathscr{O}(m)$.

Moreover, this algorithm assumes that the input vector $y$ is already sorted in decreasing magnitude order. This can easily be achieved by a sorting procedure in time $\mathscr{O}(m \log m)$.

### 3.5 Multi-Output

The efficient computation of the robust-sparse proximal mapping we derived for the singleoutput scenario will now be generalized to the multi-output case. Although we use similar arguments and insights, the analysis is much more complicated and requires more delicate and advanced treatment. Due to the tedious computations that accompany the analysis, the proofs are deferred to appendix 3.11.

When the network has multiple-output, the proximal operator $\operatorname{prox}_{\lambda g}(X, Y)$ can be written as the solution set of

$$
\begin{equation*}
\min _{V, W}\|V-X\|_{F}+\|W-Y\|_{F}+2 \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i j} V_{i k}\right| \tag{3.10}
\end{equation*}
$$

where $V \in \mathbb{R}^{n \times p}$ and $W \in \mathbb{R}^{n \times m}$. As in the single-output case, we observe that the proximal mapping (3.10) is separable with respect to the $i$-th rows of the matrices $V$ and $W$, and that the signs of the decision variables are determined by the signs of $(X, Y)$. Therefore, it is enough to consider the problem related to the $i$-th row of $V$, denoted as $x$, and $i$-th row of $W$, denoted as $y$, i.e.,

$$
\begin{equation*}
\min _{v, w \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}} h_{\lambda}(\nu, w ; x, y) \tag{3.11}
\end{equation*}
$$

where we redefine $h_{\lambda}(v, w ; x, y)$ to include the multi-output case: $h_{\lambda}(v, w ; x, y)=\frac{1}{2} \sum_{k=1}^{p}\left(v_{k}-\right.$ $\left.\left|x_{k}\right|\right)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda \sum_{k=1}^{p} v_{k} \sum_{j=1}^{m} w_{j}$. To improve readability, we will abuse notation and just write $h_{\lambda}(\nu, w)$, assuming that $(x, y)$ are understood from context.

Using the same observations we exploited to enumerated all stationary points of the proximal mapping in the single-output setup, we can identify the stationary points depending on the number of non zero elements of $v$ and $w$.

Lemma 3.7. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.17) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$. Then

1. The vector $w^{*}$ satisfies that for any $j, l \in[m]$ it holds that $w_{j}^{*} \geq w_{l}^{*}$ only if $\left|y_{j}\right| \geq\left|y_{l}\right|$.
2. The vector $v^{*}$ satisfies that for any $k, l \in[p]$ it holds that $v_{k}^{*} \geq v_{l}^{*}$ only if $\left|x_{k}\right| \geq\left|x_{l}\right|$.
3. Let $\bar{x}, \bar{y}$ be the sorted vectors in descending magnitude order of $x$ and $y$ respectively. Let $s_{v}=\left|\left\{k: v_{k}^{*}>0\right\}\right|$ and $s_{w}=\left|\left\{j: w_{j}^{*}>0\right\}\right|$. If $v^{*}, w^{*} \neq 0$, then we have that for any $k \in\left\{k: v_{k}^{*}>0\right\}$ and $j \in\left\{j: w_{j}^{*}>0\right\}$, it holds that $v^{*}=v^{\left(s_{v}, s_{w}\right)}$ and $w^{*}=w^{\left(s_{v}, s_{w}\right)}$ where

$$
\begin{align*}
& v_{k}^{\left(s_{v}, s_{w}\right)}=\left|x_{k}\right|+\mu\left(\lambda^{2} s_{w} \sum_{l=1}^{s_{v}}\left|\bar{x}_{l}\right|-\lambda \sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|\right)  \tag{3.12}\\
& w_{j}^{\left(s_{v}, s_{w}\right)}=\left|y_{j}\right|+\mu\left(\lambda^{2} s_{v} \sum_{l=1}^{s_{w}}\left|\bar{y}_{l}\right|-\lambda \sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|\right) \tag{3.13}
\end{align*}
$$

and $\mu=\left(1-s_{v} s_{w} \lambda^{2}\right)^{-1}$.

From the two first points in Lemma 3.7, the argument in Remark 3.5 is also valid in the multioutput case, and so we assume hereafter that the input vectors $x, y$ are sorted in decreasing magnitude order.

Using the second order stationary conditions, we can generalize our sparsity bound in the single-output scenario, given in Lemma 3.4, to an upper bound on the product of the sparsities of the solutions based on the value of $\lambda$; indeed, $s_{v}=1$ yields the bound in Lemma 3.4.

Lemma 3.8 (Sparsity bound). Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.11) for a given $(x, y) \in \mathbb{R}^{p} \times \mathbb{R}^{m}$. Denote $s_{\nu}=\left|\left\{j: w_{j}^{*}>0\right\}\right|$ and $s_{w}=\left|\left\{j: w_{j}^{*}>0\right\}\right|$. Then $s_{\nu} s_{w} \leq \lambda^{-2}$.

A possible algorithm for computing this proximal mapping would thus be to compute the value of $h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)}\right)$ for each pair of sparsities $\left(s_{v}, s_{w}\right) \in\{0, \ldots, p\} \times\{0, \ldots, m\}$ satisfying $s_{v} s_{w} \leq \lambda^{-2}$ and return the pair achieving the smallest value.

However, such an approach would be computationally inefficient. In order to avoid computing the value of $h_{\lambda}$ at each pair, we show the following monotonicity property of $h_{\lambda}$ in the sparsity levels, which generalizes the same property in the single-output case.

Lemma 3.9. Given $(x, y) \in \mathbb{R}^{p} \times \mathbb{R}^{m}$, for all $s_{v}, s_{w} \in\{0, \ldots, p\} \times\{0, \ldots, m\}$ satisfying $s_{v} s_{w}<\lambda^{-2}$, we have

$$
\begin{aligned}
& h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)}\right)<h_{\lambda}\left(v^{\left(s_{v}, s_{w}-1\right)}, w^{\left(s_{v}, s_{w}-1\right)}\right), \\
& h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)}\right)<h_{\lambda}\left(v^{\left(s_{v}-1, s_{w}\right)}, w^{\left(s_{v}-1, s_{w}\right)}\right) .
\end{aligned}
$$

Moreover, the feasibility criterion $v \geq 0, w \geq 0$ also has a monotonic property:
Lemma 3.10. Let $(k, l) \in[p] \times[m]$ be such that $k l \leq \lambda^{-2}$.
If $v^{(k, l)} \geq 0$ and $w^{(k, l)} \geq 0$, then, $v^{(i, j)} \geq 0$ and $w^{(i, j)} \geq 0 \forall i=1, \ldots, k$ and $\forall j=1, \ldots, l$.

To properly address the complications arising from handling two intertwining sparsity levels at the same time, we introduce the notion of maximal feasibility boundary (MFB) which acts a frontier of possible sparsity levels.

Definition 3.1 (Maximal feasibility boundary). We say that a sparsity pair $\left(s_{v}, s_{w}\right) \in\{0, \ldots, p\} \times$ $\{0, \ldots, m\}$ is on the maximal feasibility boundary (MFB) if incrementing either $s_{v}$ or $s_{w}$ results with a non-stationary point. That is, if both of the following conditions hold:

- $v_{s_{v}+1}^{\left(s_{v}+1, s_{w}\right)}<0$ or $w_{s_{w}}^{\left(s_{v}+1, s_{w}\right)}<0 \operatorname{or}\left(s_{\nu}+1\right) s_{w}>\lambda^{-2}$,
- $v_{s_{v}}^{\left(s_{v}, s_{w}+1\right)}<0$ or $w_{s_{w}+1}^{\left(s_{v}, s_{w}+1\right)}<0$ or $s_{v}\left(s_{w}+1\right)>\lambda^{-2}$.

The efficient computation of the multi-output robust-sparse proximal mapping is based on the fact that we only need to compute the value of $h_{\lambda}$ for sparsity levels that are at the frontier of the MFB. This allows us to find the optimal sparsity in time $\mathscr{O}(p+m)$, improving upon the $\mathscr{O}(p m)$ complexity of the exhaustive search. Algorithm 3.3 implements the above by employing a binary search type procedure defined in Algorithm 3.5 to calculate the MFB.

Theorem 3.4 (Multi-output prox computation). Let $\left(V_{:, i}^{*}, W_{i,:}^{*}\right)$ be the output of Algorithm 3.3 with input $X_{:, i}, Y_{i,:}, \lambda$, where each $X_{:, i}, Y_{i,:}$ are sorted in decreasing magnitude order. Then $\left(V^{*}, W^{*}\right)$ is a solution to (3.5).

Time complexity of Algorithm 3.3. It is easy to see that the maximal feasibility boundary contains at most $\min (m, p)$ pairs, and Algorithm 3.5 finds them all in time $\mathscr{O}(m+p)$. Then, for

```
Algorithm 3.3 Multi-output robust-sparse proximal mapping
Input: \(x \in \mathbb{R}^{p}, y \in \mathbb{R}^{m}\) ordered in decreasing magnitude order, \(\lambda>0\).
    Employ Algorithm 3.5: Find the set of sparsity pairs \(S=\left\{\left(s_{v}, s_{w}\right)\right\}\) that are on the MFB
    \(h_{o p t} \leftarrow \infty\)
    for \(\left(s_{v}, s_{w}\right) \in S\) do
        Compute \(v^{\left(s_{v}, s_{w}\right)}\) and \(w^{\left(s_{v}, s_{w}\right)}\) as given in equations (3.12), (3.13)
        if \(h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, v^{\left(s_{v}, s_{w}\right)} ;|x|,|y|\right)<h_{o p t}\) then
            \(h_{o p t}=h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, v^{\left(s_{v}, s_{w}\right)} ;|x|,|y|\right)\)
            \(v^{*} \leftarrow v^{\left(s_{v}, s_{w}\right)}, w^{*} \leftarrow w^{\left(s_{v}, s_{w}\right)}\)
        end if
    end for
    return \(\left(\operatorname{sign}(x) \circ v^{*}, \operatorname{sign}(y) \circ w^{*}\right)\)
```

each such pair ( $s_{v}, s_{w}$ ), we must compute $v^{\left(s_{v}, s_{w}\right)}$ and $w^{\left(s_{v}, s_{w}\right)}$ and $h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)}\right.$ ), which takes time $\mathscr{O}(m+p)$. The total complexity of Algorithm 3.3 is thus $\mathscr{O}(\min (m, p)(m+p))$. In most practical application, the output layer size $p$ can be considered $\mathscr{O}(1)$, so that the complexity of computing this proximal mapping is comparable to the complexity of computing one stochastic gradient.

### 3.6 Related Work

The path regularization approach to train neural networks can be traced back to the seminal paper by Neyshabur et al. (2015c), who introduced the $p$-path-norm as a heuristic proxy to control the capacity of the network.

In this paper (cf. Theorem 3.1), we took a step forward by moving from heuristic explanations to rigorous arguments by establishing a new connection between the 1-path-norm and the Lipschitz constant of the network. This result also reads as a relation between the 1-pathnorm and the product bound, of which variants have been found to be useful in deriving generalization bounds (Bartlett et al., 2017b).

Generalization bounds in terms of the $p$-path-norm were also derived in (Neyshabur et al., 2015c), but the question of how to methodologically exploit these as regularizers remains open; our algorithmic contribution is a first step in this direction. Additionally, issues regarding optimization with path-norm regularization were reported by Ravi et al. (2019), which examined a conditional gradient method in the context of path-norm regularization.

A growing collection of works have focused on the task of network compression, doing so via sparsity-inducing regularizers (Alvarez and Salzmann, 2016; Yoon and Hwang, 2017; Scardapane et al., 2017; Lemhadri et al., 2019). They have achieved a great level of success by setting the regularization term in an $a d-h o c$ manner. In contrast, we follow a principled regularization approach with theoretical properties of generalization and robustness, and as a consequence, we are able to quantify the sparsity of the resulting networks.


Figure 3.1: value of regularized cross-entropy loss across iterations.


Figure 3.2: Misclassification test error (left) and robust test error (right) as a function of the percentage of nonzero weights.


Figure 3.3: Misclassification test error (left) and robust test error (right) on the test set, as a function of the regularization parameter $\lambda$.

Moreover, the aforementioned works only use convex regularizers for which efficient proximal mappings are available (see Bach et al. (2012) and references therein). We tackle the much harder non-convex regularization task, and derive a new method to compute the proximal mapping in this case. The merits of non-convex non-smooth regularization, and difficulties regarding their optimization, have been extensively studied in the imaging and signal sciences, see e.g., (Ochs et al., 2015) and the recent survey (Wen et al., 2018).

Layer-wise constraints or regularization with matrix-norms, which are also motivated by the product bound, have been used for robust training (Cisse et al., 2017b; Tsuzuku et al., 2018a) and generative models (Miyato et al., 2018a). These focus on robustness with respect to the $\ell_{2}$-norm, which requires a careful handling of operations on the singular values of the weight matrices, and does not have the extra benefit of inducing sparsity.

In section 3.7 we compare to this class of methods for the $\ell_{\infty}$-norm case, in which a simple rescaling of the rows in the weight matrices yields a numerically stable procedure (Duchi et al., 2008b; Condat, 2016).

### 3.7 Experiments

We empirically evaluate shallow neural networks trained by regularized empirical risk minimization (3.4) using cross-entropy loss. In terms of the weight matrices $V$ and $W$ of the network (3.1), the following regularizers are considered:
$\ell_{1}$ regularization. We penalize the $\ell_{1}$-norm of the parameters of the network, i.e., $g(V, W)=$ $\|\operatorname{vec}(V)\|_{1}+\|\operatorname{vec}(W)\|_{1}$ in the objective function (3.4).

1-path-norm regularization. We set $g(V, W)$ as $\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i j} V_{k i}\right|$ in the objective function (3.4).

Layer-wise regularization (Parseval Networks). we minimize the cross-entropy loss with a hard constrain on the $\ell_{\infty}$-operator-norm of the weight matrices i.e., $\|W\|_{\infty} \leq \lambda^{-1}$ and $\|V\|_{\infty} \leq \lambda^{-1}$, as described by Cisse et al. (2017b). The projection on such set is achieved by projecting each row of the matrices onto an $\ell_{1}$-ball using efficient algorithms (Duchi et al., 2008b; Condat, 2016).

Remark. We will refer (incorrectly) to the training loop defined by PyTorch's SGD optimizer as Stochastic gradient descent (SGD) (see the discussion in section 3.1).

Experimental setup. Our benchmarks are the MNIST (LeCun and Cortes, 2010a), FashionMNIST (Xiao et al., 2017a) and Kuzushiji-MNIST (Clanuwat et al., 2018a). For a wide range of learning rates, number of hidden neurons and regularization parameters $\lambda$, we train networks with SGD and Proximal-SGD (with constant learning rate). We do so for 20 epochs and with batch size set to 100 . For each combination of parameters we train 6 networks with the default random initialization. Details and further experiments are reported in appendix 3.12.

### 3.7.1 Convergence of SGD vs Proximal-SGD

Due to the non-differentiability of the $\ell_{1}$ - and path-norm regularizers, we expect ProximalSGD to converge faster, and to lower values of the regularized loss, when compared to SGD. This is examined in Figure 3.1, where we plot the value of the loss function across iterations. For both SGD and Proximal-SGD, the loss function decays rapidly in the first few epochs. We then enter a second regime where SGD suffers from slow convergence, whereas ProximalSGD continues to reduce the loss at a fast rate. At the end of the 20 -epochs, Proximal-SGD consistently achieves a lower value of the loss.

An advantage of Proximal-SGD over plain SGD is that the proximal mappings of both the $\ell_{1}$ - and path-norm regularizers can set many weights to exactly zero. In Figure 3.2 we plot the average error and robust test error obtained, as a fuction of the sparsity of the network. Compared to $\ell_{1}$ regularization, the sparsity pattern induced by the 1-path-norm correlates with the robustness to a higher degree. As a drawback, it appears that in more difficult datasets like KMNIST, the 1-path-norm struggles to obtain good accuracy and sparsity simultaneously.

### 3.7.2 The robustness-accuracy trade-off

The relation between the Lipschitz constant of a network and its robustness to adversarial perturbations has been extensively studied in the literature. In theorem 3.1 we have shown that the 1-path-norm of a single-output network is a tighter upper bound of its Lipschitz constant, compared to the corresponding product bound.

To the best of our knowledge, the $\ell_{1}$-norm regularizer only provides an upper boud on the already loose product bound (Neyshabur et al., 2015c, Eq. (4)), which makes it less attractive as a regularizer, despite its sparsity-inducing properties. Hence, the 1-path-norm regularizer is, in theory, a better proxy for robustness than the other regularization schemes.

Subsection 3.12.2 shows the misclassification error on clean and adversarial examples as a function of $\lambda$, and corresponds to the learning rate minimizing the error on clean samples. The adversarial perturbations were obtained by PGD (Madry et al., 2018b).

Any training procedure which promotes robustness of a classifier may decrease its accuracy, and this effect is consistently observed in practice (Tsipras et al., 2019). Hence, the merits of a regularizer should be measured by how efficiently it can trade-off accuracy for robustness. We observe that for all three regularization schemes, there exists choices of $\lambda$ that attain the best possible error on clean samples.

On the other hand, the error obtained by the $\ell_{1}$ regularization degrades significantly. The layerwise and 1-path-norm regularization achieve a noticeably low error on adversarial examples. Comparing the latter schemes, the 1-path-norm regularization shows only a slight advantage over the layer-wise methods, which merits further investigation.

### 3.8 Appendix: Proof of Theorem 3.1

We will first prove a particular case of Theorem 3.1, the single-output case ( $p=1$ ).

Proposition 3.1. Let $h_{V, W}(x)=V^{T} \sigma(W x): \mathbb{R}^{m} \rightarrow \mathbb{R}$ be a neural network where $V \in \mathbb{R}^{n \times 1}$ and $W \in \mathbb{R}^{n \times m}$. Suppose that that the derivative of the activation is globally bounded between zero and one. Its Lipschitz constant with respect to the $\ell_{\infty}$ norm (for the input space) and the $\ell_{1}$-norm (for the output space) is bounded as follows:

$$
\begin{equation*}
L_{V, W} \leq \sum_{i=1}^{n} \sum_{j=1}^{m}\left|W_{i, j} V_{i, 1}\right| \leq\|V\|_{1}\|W\|_{\infty} \tag{3.14}
\end{equation*}
$$

First, note that because the output space is $\mathbb{R}$, the $\ell_{1}$-norm is just the absolute value of the output. In this case the Lipschitz constant of the single-output function $h$ is equal to the supremum of the $\ell_{1}$-norm of its gradient, over its domain (c.f., Latorre et al. (2020b, Theorem 1)).

Proof.

$$
\begin{aligned}
& L_{V, W}=\sup _{x}\left\|\nabla h_{V, W}(x)\right\|_{1} \\
= & \sup _{x} \sup _{\|t\|_{\infty} \leq 1} t^{T} \nabla h_{V, W}(x) \\
= & \sup _{x} \sup _{\|t\|_{\infty} \leq 1} t^{T} W^{T} \sigma^{\prime}(W x) V \\
\leq & \sup _{0 \leq s \leq 1} \sup _{\|t\|_{\infty} \leq 1} t^{T} W^{T} \operatorname{Diag}(s) V \\
= & \sup _{0 \leq s \leq 1} \sup _{\|t\|_{\infty} \leq 1} \sum_{i=1}^{n} \sum_{j=1}^{m} t_{i}\left(W^{T} \operatorname{Diag}(V)\right)_{i, j} s_{j} \\
\leq & \sum_{i=1}^{n} \sum_{j=1}^{m} \sup _{0 \leq s_{j} \leq 1} \sup \sup _{1 \leq t_{i} \leq 1} t_{i}\left(W^{T} \operatorname{Diag}(V)\right)_{i, j} s_{j} \\
= & \sum_{i=1}^{n} \sum_{j=1}^{m}\left|W^{T} \operatorname{Diag}(V)\right|_{i, j}=\sum_{i=1}^{n} \sum_{j=1}^{m}\left|W_{i, j} V_{i, 1}\right|
\end{aligned}
$$

This shows the first inequality in (3.14). We now show the second inequality. Denote the $i$-th row of the matrix $W$ as $w_{i}$ :

$$
\begin{aligned}
\sum_{i=1}^{n} \sum_{j=1}^{m}\left|W_{i, j} V_{i, 1}\right| & =\sum_{i=1}^{n}\left|V_{i, 1}\right| \sum_{j=1}^{m}\left|W_{i, j}\right| \\
& =\sum_{i=1}^{n}\left|V_{i, 1}\right|\left\|w_{i}\right\|_{1} \\
& \leq \sum_{i=1}^{n}\left|V_{i, 1}\right| \max _{j=1, \ldots, m}\left\|w_{j}\right\|_{1} \\
& =\sum_{i=1}^{n}\left|V_{i, 1}\right|\|W\|_{\infty} \\
& =\|V\|_{1}\|W\|_{\infty}
\end{aligned}
$$

In the fourth line we have used the fact that the $\ell_{\infty}$ operator norm of a matrix is equal to the maximum $\ell_{1}$-norm of the rows.

Proof of Theorem 3.1. We now proceed with the general case where $V \in \mathbb{R}^{n \times p}, W \in \mathbb{R}^{n \times m}$ and $h_{V, W}(x)=V^{T} \sigma(W x)$.

Proof. Denote the columns of $V$, in order, as $V_{1}, \ldots, V_{p}$. Using Proposition 3.1 we have

$$
\begin{aligned}
\left\|V^{T} \sigma(W x)-V^{T} \sigma(W y)\right\|_{1} & =\sum_{k=1}^{p}\left|V_{k}^{T} \sigma(W x)-V_{k}^{T} \sigma(W y)\right| \\
& \leq \sum_{k=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{m} \mid W_{i, j} V_{i, k}\|x-u\|_{\infty} \\
& \leq \sum_{k=1}^{p}\left\|V_{k}\right\|_{1}\|W\|_{\infty}\|x-y\|_{\infty} \\
& =\left\|V^{T}\right\|_{\infty, 1}\|W\|_{\infty}\|x-y\|_{\infty}
\end{aligned}
$$

where in the fourth line we have used the fact that the $\left(\ell_{\infty}, \ell_{1}\right)$ operator norm of a matrix $V^{T}$ is equal to the sum of the $\ell_{1}$ norm of its rows i.e., the columns of $V$. This shows that $L_{V, W} \leq \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p}\left|W_{i, j} V_{i, k}\right| \leq\left\|V^{T}\right\|_{\infty, 1}\|W\|_{\infty}$

### 3.9 Appendix: Proof of Theorem 3.2

In this section we prove the theoretical guarantees stated in Theorem 3.2 of the prox-grad method described by Algorithm 3.1. The first and second parts of Theorem 3.2 follow immediately from the results establish by Bolte et al. (2013). Part two in Theorem 3.2 states that Algorithm 3.1 is globally convergent under the celebrated Kurdyka-Lojasiewicz (KL) property Attouch et al. (2010). The broad classes of semi-algebraic and subanalytic functions, widely used in optimization, satisfy the KL property (see e.g. (Bolte et al., 2013, Section 5)), and in particular, most convex functions encountered in finite dimensional applications satisfy it (see (Bolte et al., 2013, Section 5.1)). We refer the reader to the works Attouch et al. (2010, 2011); Bolte et al. (2013), in particular to (Bolte et al., 2013, Sections 3.2-3.5) for additional information and results.

For Part three we require the sufficient decrease property stated next.
Lemma 3.11 (Sufficient decrease property (Bolte et al., 2013, Lemma 2)). Let $\Psi: \mathbb{R}^{n} \rightarrow \mathbb{R}$ be a continuously differentiable function with gradient assumed $L_{\psi}$-Lipschitz continuous, and let $\sigma: \mathbb{R}^{n} \rightarrow(-\infty, \infty]$ be a proper l.s.c function satisfying that $\inf \sigma>-\infty$. Fix any $t \in(0,1 / L \Psi)$. Then, for any $\mathbf{u} \in \operatorname{dom} \sigma$ and any $\mathbf{u}^{+} \in \mathbb{R}^{n}$ defined by

$$
\mathbf{u}^{+} \in \operatorname{prox}_{\sigma t}(\mathbf{u}-t \nabla \Psi(\mathbf{u}))
$$

we have

$$
\Psi(\mathbf{u})+\sigma(\mathbf{u})-\Psi\left(\mathbf{u}^{+}\right)-\sigma(\mathbf{u}) \geq \frac{1-t L_{\Psi}}{2 t}\left\|\mathbf{u}^{+}-\mathbf{u}\right\|^{2} .
$$

Proof of Theorem 3.2. The first and second parts follow from the results established by Bolte
et al. (2013). We will now prove the third part. By Lemma 3.11 we have that

$$
\begin{equation*}
\left.\mathscr{F}\left(z^{k}\right)-\mathscr{F}\left(z^{k+1}\right)=f\left(z^{k}\right)+\lambda g\left(z^{k}\right)-f\left(z^{k+1}\right)-\lambda g\left(z^{k+1}\right)\right) \geq \frac{1-L \eta_{k}}{2 \eta_{k}}\left\|z^{k+1}-z^{k}\right\|^{2} \tag{3.15}
\end{equation*}
$$

Hence $\left\{f\left(z^{k}\right)+\lambda g\left(z^{k}\right)\right\}_{k \geq 0}$ is a non-increasing sequence that strictly decreasing unless a critical point is obtained in a finite number of steps. By summing (3.15) over $k=0,1, \ldots, K$ and using the fact that $\left\{f\left(z^{k}\right)+\lambda g\left(z^{k}\right)\right\}_{k \geq 0}$ is non-increasing and is bounded below by $\mathscr{F}_{*}$, we obtain that

$$
\begin{aligned}
\mathscr{F}\left(z^{0}\right)-\mathscr{F}_{*} & \geq \sum_{k=0}^{K} \frac{1-L \eta_{k}}{2 \eta_{k}}\left\|z^{k+1}-z^{k}\right\|^{2} \\
& \geq \frac{1}{2}(c-L) K \min _{k=0, \ldots, K}\left\|z^{k+1}-z^{k}\right\|_{2}^{2}
\end{aligned}
$$

Consequently,

$$
\min _{k=0, \ldots, K}\left\|z^{k+1}-z^{k}\right\|_{2} \leq \sqrt{\frac{2\left(\mathscr{F}\left(z^{0}\right)-\mathscr{F}_{*}\right)}{(c-L) K}}
$$

### 3.10 Appendix: Single output proximal map computation

This section provides the theoretical background and the required intermediate results to prove Theorem 3.3.

### 3.10.1 Moving to an Equivalent Easier Problem

We are interested in minimizing the nonconvex twice continuously differentiable function

$$
\begin{equation*}
\min _{\nu, w \in \mathbb{R} \times \mathbb{R}^{m}} \frac{1}{2}(\nu-x)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-y_{j}\right)^{2}+\lambda|v| \sum_{j=1}^{m}\left|w_{j}\right| . \tag{3.16}
\end{equation*}
$$

The signs of the elements of the decision variables in (3.16) are determined by the signs of ( $x, y$ ), and consequently, the problem in (3.16) is equivalent to problem (3.17); this is (partly) formulated by Lemma 3.12.

$$
\begin{equation*}
\min _{v, w \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}} h_{\lambda}(\nu, w ; x, y) \equiv \frac{1}{2}(\nu-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda \nu \sum_{j=1}^{m} w_{j} \tag{3.17}
\end{equation*}
$$

Lemma 3.12. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{n}$ be an optimal solution of problem (3.17). Then $(\operatorname{sign}(x) \cdot$ $\left.v^{*}, \operatorname{sign}(y) \circ w^{*}\right)$ is an optimal solution of problem (3.16).

Proof. We have that

$$
\begin{aligned}
\tilde{h}_{\lambda}(v, w ; x, y) & \equiv \frac{1}{2}(v-x)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-y_{j}\right)^{2}+\lambda|v| \sum_{j=1}^{m}\left|w_{j}\right| \\
& =\frac{1}{2}(\operatorname{sign}(x) v-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(\operatorname{sign}\left(y_{j}\right) w_{j}-\left|y_{j}\right|\right)^{2}+\lambda|v| \sum_{j=1}^{m}\left|w_{j}\right| \\
& \geq \frac{1}{2}(|v|-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(\left|w_{j}\right|-\left|y_{j}\right|\right)^{2}+\lambda v \sum_{j=1}^{m} w_{j} \\
& \geq h_{\lambda}\left(v^{*}, w^{*} ;|x|,|y|\right)
\end{aligned}
$$

where the last inequality follows from the fact that ( $v^{*}, w^{*}$ ) is an optimal solution of (3.17). Since equality with the lower bound is attained by setting $(v, w)=\left(\operatorname{sign}(x) \cdot v^{*}, \operatorname{sign}(y) \circ w^{*}\right)$, we conclude that $\left(\operatorname{sign}(x) \cdot v^{*}, \operatorname{sign}(y) \circ w^{*}\right)$ is an optimal solution of (3.16).

To summarize, we have established that, finding an optimal solution to (3.17) and then changing signs accordingly, yields an optimal solution to (3.16). We will now focus on obtaining an optimal solution for (3.16).

### 3.10.2 Solving the Prox Problem

First we note that problem (3.17) is well-posed.

Lemma 3.13 (Well-posedness of (3.17)). For any $\lambda \geq 0$ and any $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$, the problem (3.17) has a global optimal solution.

Proof. The claim follows from the fact that the objective function is coercive, cf. (Beck, 2014, Thm. 2.32).

In light of Lemma 4.5, and due to the fact that in (3.17) we minimize a continuously differentiable function over a closed convex set, the set of optimal solutions of (3.17) is a nonempty subset of the set of stationary points of (3.17). These satisfy the following conditions (cf. (Beck, 2014, Ch. 9.1)).

Lemma 3.14 (Stationarity conditions). Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.17) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$. Then

$$
\begin{aligned}
w_{j}^{*} & =\max \left\{0,\left|y_{j}\right|-\lambda v^{*}\right\} \text { for any } j=1,2, \ldots, m \\
v^{*} & =\max \left\{0,|x|-\lambda \sum_{j=1}^{m} w_{j}^{*}\right\}
\end{aligned}
$$

Proof. The stationarity (first-order) conditions of (3.17) (cf. (Beck, 2014, Ch. 9.1)) state that

$$
\frac{\partial h_{\lambda}}{\partial v}\left(v^{*}, w^{*} ; x, y\right)\left\{\begin{array} { l l } 
{ = 0 , } & { v ^ { * } > 0 , } \\
{ \geq 0 , } & { v ^ { * } = 0 , }
\end{array} \quad \frac { \partial h _ { \lambda } } { \partial w _ { j } } ( v ^ { * } , w ^ { * } ; x , y ) \left\{\begin{array}{ll}
=0, & w_{j}^{*}>0 \\
\geq 0, & w_{j}^{*}=0
\end{array}\right.\right.
$$

which translates to

$$
v^{*}-|x|+\lambda \sum_{j=1}^{m} w_{j}^{*}\left\{\begin{array}{ll}
=0, & v^{*}>0, \\
\geq 0, & v^{*}=0,
\end{array} \quad w_{j}^{*}-\left|y_{j}\right|+\lambda v^{*} \begin{cases}=0, & w_{j}^{*}>0, \\
\geq 0, & w_{j}^{*}=0,\end{cases}\right.
$$

and the required follows.

The stationarity conditions given in Lemma 4.7 imply a solution form that we exploit in Algorithm 3.2; this is described by Corollary 4.1, where we use the convention that $\sum_{j=1}^{0} a_{j} \equiv 0$ for any $\left\{a_{j}\right\} \subseteq \mathbb{R}$.

Corollary 3.3. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.17) for a given $(x, y) \in \mathbb{R} \times \mathbb{R}^{m}$.

1. The vector $w^{*}$ satisfies that for any $j, l \in\{1,2, \ldots, m\}$ it holds that $w_{j}^{*} \geq w_{l}^{*}$ only if $\left|y_{j}\right| \geq$ $\left|y_{l}\right|$.
2. If $v^{*}=0$, then $w^{*}=y$.
3. If $v^{*}>0$, and $s=\left|\left\{j: w_{j}^{*}>0\right\}\right|$, then we have that

$$
\begin{equation*}
v^{*}=\frac{1}{1-s \lambda^{2}}\left(|x|-\lambda \sum_{j=1}^{s}\left|\bar{y}_{j}\right|\right), \tag{3.18}
\end{equation*}
$$

where $\bar{y}$ is the sorted vector of $y$ in descending magnitude order.

Proof. The first part follows trivially from the stationarity conditions on $w^{*}$ given in Lemma 4.7. The second part also follows trivially from the problem definition.

From the first part and the conditions in Lemma 4.7 we have that $\sum_{j=1}^{m} w_{j}^{*}=\sum_{j=1}^{s}\left|\bar{y}_{j}\right|-\lambda s v^{*}$. Plugging the latter to the stationarity condition on $v^{*}$ (given in Lemma 4.7) then implies the required.

In our analysis, we strictly distinguish between the trivial solution $\left(v^{*}, w^{*}\right)=(0, y)$, and the non-trivial solution in which $v^{*}>0$. A practical point-of-view suggests that if $v^{*}=0$, then the corresponding succeeding weights should also be zero, even though the optimality conditions imply otherwise. However, to avoid hindering the training process, this observation is considered only in the end of the training.

Recall that our analysis so-far implies that the magnitude order of $y$ determines the order magnitude of $w$, effectively implying on set of non-zero entries in $w$ (cf. Remark 3.5). For clarity of indices, and without loss of generality, we assume throughout this section that the vector $y$ is already sorted in decreasing order of magnitude, that is $y \equiv \bar{y}$. We will use, without confusion, both notation to describe the same entity in order to maintain coherence with our procedures and results.

Denote

$$
\begin{align*}
v^{(s)} & =\frac{1}{1-s \lambda^{2}}\left(|x|-\lambda \sum_{j=1}^{s}\left|y_{j}\right|\right)  \tag{3.19}\\
w_{j}^{(s)} & =\left|y_{j}\right|-\lambda v^{(s)} \text { for } j=1,2, \ldots, s, \text { and } w_{j}^{(s)}=0 \text { otherwise. }
\end{align*}
$$

Lemma 3.5 which states the monotonicity property

$$
h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right)<h_{\lambda}\left(v^{(s-1)}, w^{(s-1)} ; x, y\right)
$$

is proved next.

Proof of Lemma 3.5. Recall that $h_{\lambda}(\nu, w ; x, y):=\frac{1}{2}(\nu-|x|)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda v \sum_{j=1}^{m} w_{j}$. By plugging $w^{(s)}$ defined in (3.19) to $h_{\lambda}$ we obtain that

$$
\begin{aligned}
h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right)= & \frac{1}{2}\left(v^{(s)}-|x|\right)^{2}+\frac{1}{2} \sum_{i=1}^{s}\left(\left|\bar{y}_{i}\right|-\left(\left|\bar{y}_{i}\right|-\lambda v^{(s)}\right)\right)^{2} \\
& +\frac{1}{2} \sum_{i=s+1}^{m}\left|\bar{y}_{i}\right|^{2}+\lambda v^{(s)} \sum_{i=1}^{s}\left(\left|\bar{y}_{i}\right|-\lambda v^{(s)}\right) \\
= & \frac{1}{2}\left(v^{(s)}-|x|\right)^{2}+\frac{\lambda^{2}}{2} s\left(v^{(s)}\right)^{2}+\frac{1}{2}\|y\|_{2}^{2} \\
& -\frac{1}{2} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|^{2}+\lambda v^{(s)} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|-\lambda^{2} s\left(v^{(s)}\right)^{2} .
\end{aligned}
$$

Consequently, plugging $v^{(s)}$, defined in (3.19), yields

$$
\begin{aligned}
h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right)= & \frac{1}{2}\left(\frac{\lambda^{2} s}{1-\lambda^{2} s}|x|-\frac{\lambda}{1-\lambda^{2} s} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}-\frac{\lambda^{2} s}{2\left(1-\lambda^{2} s\right)^{2}}\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2} \\
& +\frac{\lambda}{1-\lambda^{2} s} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)-\frac{1}{2} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|^{2}+\frac{1}{2}\|y\|_{2}^{2} \\
= & \frac{\lambda^{2} s}{2\left(1-\lambda^{2} s\right)^{2}} x^{2}\left(\lambda^{2} s-1\right)+\frac{\lambda^{2}}{2\left(1-\lambda^{2} s\right)^{2}}\left(\sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}\left(1-\lambda^{2} s-2\left(1-\lambda^{2} s\right)\right) \\
& +|x| \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\left(-\frac{\lambda^{3} s}{\left(1-\lambda^{2} s\right)^{2}}+\frac{\lambda^{3} s}{\left(1-\lambda^{2} s\right)^{2}}+\frac{\lambda}{1-\lambda^{2} s}\right)-\frac{1}{2} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|^{2}+\frac{1}{2}\|y\|_{2}^{2}
\end{aligned}
$$

$$
\left.\begin{array}{l}
\quad=\frac{1}{2\left(1-\lambda^{2} s\right)}\left(-\lambda^{2} s x^{2}-\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}+x^{2}\right)-\frac{1}{2} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|^{2}+\frac{1}{2}\|y\|_{2}^{2} \\
=-\frac{1}{2\left(1-\lambda^{2} s\right)}\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}+\frac{1}{2}\|x\|_{2}^{2}-\frac{1}{2} \sum_{i=1}^{s}\left|\bar{y}_{i}\right|^{2}+\frac{1}{2}\|y\|_{2}^{2} \\
\\
=-\left(1+\frac{\lambda^{2}}{1-\lambda^{2} s}\right) \frac{1}{2\left(1-\lambda^{2}(s-1)\right)}\left(|x|-\lambda \sum_{i=1}^{s-1}\left|\bar{y}_{i}\right|-\lambda\left|\bar{y}_{s}\right|\right)^{2} \\
+\frac{1}{2}\|x\|_{2}^{2}-\frac{1}{2} \sum_{i=1}^{s-1}\left|\bar{y}_{i}\right|^{2}-\frac{1}{2}\left|\bar{y}_{s}\right|^{2}+\frac{1}{2}\|y\|_{2}^{2} \\
=
\end{array} h_{\lambda}\left(\nu^{(s-1)}, w^{(s-1)} ; x, y\right)-\frac{1}{2\left(1-\lambda^{2} s+\lambda^{2}\right)}\left(-2 \lambda\left|\bar{y}_{s}\right|\left(|x|-\lambda \sum_{i=1}^{s-1}\left|\bar{y}_{i}\right|\right)+\lambda^{2}\left|\bar{y}_{s}\right|^{2}\right)\right)
$$

Therefore,

$$
\begin{aligned}
& h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right)-h_{\lambda}\left(v^{(s-1)}, w^{(s-1)} ; x, y\right) \\
= & -\frac{1}{2\left(1-\lambda^{2} s+\lambda^{2}\right)}\left(-2 \lambda\left|\bar{y}_{s}\right|\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)-\lambda^{2}\left|\bar{y}_{s}\right|^{2}+\frac{\lambda^{2}}{1-\lambda^{2} s}\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}+\left(1-\lambda^{2} s+\lambda^{2}\right)\left|\bar{y}_{s}\right|^{2}\right) \\
= & -\frac{1}{2\left(1-\lambda^{2} s+\lambda^{2}\right)}\left(\left(1-\lambda^{2} s\right)\left|\bar{y}_{s}\right|^{2}-2 \lambda\left|\bar{y}_{s}\right|\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)+\frac{\lambda^{2}}{1-\lambda^{2} s}\left(|x|-\lambda \sum_{i=1}^{s}\left|\bar{y}_{i}\right|\right)^{2}\right) \\
= & -\frac{1-\lambda^{2} s}{2\left(1-\lambda^{2} s+\lambda^{2}\right)}\left(\left|\bar{y}_{s}\right|^{2}-2 \lambda\left|\bar{y}_{s}\right| v^{(s)}+\lambda^{2}\left(v^{(s)}\right)^{2}\right) \\
= & -\frac{1-\lambda^{2} s}{2\left(1-\lambda^{2} s+\lambda^{2}\right)}\left(\left|\bar{y}_{s}\right|-\lambda v^{(s)}\right)^{2} \leq 0,
\end{aligned}
$$

meaning that

$$
h_{\lambda}\left(v^{(s)}, w^{(s)} ; x, y\right) \leq h_{\lambda}\left(v^{(s-1)}, w^{(s-1)} ; x, y\right)
$$

We can now prove our key result formulated in Corollary 3.2, that states that $\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$ is an optimal solution of (3.7) for

$$
s^{*}=\max \left\{s \in[\bar{s}]: v^{(s)}, w^{(s)}>0\right\}, \text { where } \bar{s}=\min \left(\left\lfloor\lambda^{-2}\right\rfloor, m\right)
$$

Proof of Corollary 3.2. By Lemma 3.3, $\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$ is a stationary point of (3.7). Moreover, according to Corollary 3.1 and Lemma $3.4,\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$ belongs to the set of $\bar{s}$ stationary points
that are candidates to be optimal solutions of (3.7). Invoking Lemma 3.5, we have that

$$
\begin{equation*}
h_{\lambda}\left(\nu^{\left(s^{*}\right)}, w^{\left(s^{*}\right)} ; x, y\right)<h_{\lambda}\left(\nu^{(j)}, w^{(j)} ; x, y\right), \quad \forall s^{*}>j \tag{3.20}
\end{equation*}
$$

Hence, $\left(v^{(j)}, w^{(j)}\right)$ is not an optimal solution for any $j<s^{*}$.
Let us now consider the complementary case. By Lemma 3.4, for any $i>\bar{s}$ the pair $\left(v^{(i)}, w^{(i)}\right)$ does not satisfy the second-order optimality conditions, and therefore is not an optimal solution. On the other hand, by the definition of $s^{*}$, for any $\bar{s}>i>s^{*}$ the pair $\left(v^{(i)}, w^{(i)}\right)$ is not a feasible solution, and subsequently not a stationary point. To conclude, $h_{\lambda}\left(v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)} ; x, y\right)<$ $h_{\lambda}\left(v^{(j)}, w^{(j)} ; x, y\right)$ holds for any $j \neq s^{*}$ such that $\left(v^{(j)}, w^{(j)}\right)$ is a stationary point, meaning that ( $\left.v^{\left(s^{*}\right)}, w^{\left(s^{*}\right)}\right)$ is an optimal solution of (3.7).

Finally, we will show that the problem of finding $s^{*}$ can be easily solved using binary search. To this end, we show that the feasibility criterion (i.e., $v^{(s)}>0$ and $w^{(s)}>0$ ) satisfies that

$$
\left(v^{(k)}, w^{(k)}\right) \text { is feasible } \Rightarrow\left(v^{(i)}, w^{(i)}\right) \text { is feasible } \forall i<k
$$

Proof of Lemma 4.6. Suppose that $\left(v^{(k)}, w^{(k)}\right)$ is feasible for some $k \in\{2, \ldots, \bar{s}\}$. By induction principle, it is sufficient to show that $\left(v^{(k-1)}, w^{(k-1)}\right)$ is feasible in order to prove the result.

By (3.19), we have:

$$
\left(1-k \lambda^{2}\right) v^{(k)}=|x|-\lambda \sum_{j=1}^{k}\left|y_{j}\right|=\left(1-k \lambda^{2}+\lambda^{2}\right) v^{(k-1)}-\left|y_{k}\right|
$$

which implies

$$
v^{(k-1)}=\frac{1}{\left(1-k \lambda^{2}+\lambda^{2}\right)}\left(\left(1-k \lambda^{2}\right) v^{(k)}+\left|y_{k}\right|\right) \geq 0 .
$$

For $w^{(k)}$, it is easy to see from (3.19) that since the vector $y$ is sorted in decreasing order of magnitude, the vector $w^{(k)}$ is also sorted in decreasing order, and thus $w^{(k)}$ is feasible if and only if $w_{k}^{(k)}>0$.

$$
\begin{aligned}
\left(1-k \lambda^{2}\right) w_{k}^{(k)}= & \left(1-k \lambda^{2}\right)\left|y_{k}\right|-\lambda|x|+\lambda^{2} \sum_{j=1}^{k}\left|y_{j}\right| \\
= & -\lambda|x|+\left(1-(k-1) \lambda^{2}\right)\left|y_{k-1}\right|+\lambda^{2} \sum_{j=1}^{k-1}\left|y_{j}\right|+\lambda^{2}\left|y_{k}\right| \\
& +\left(1-k \lambda^{2}\right)\left|y_{k}\right|-\left(1-(k-1) \lambda^{2}\right)\left|y_{k-1}\right| \\
= & \left(1-(k-1) \lambda^{2}\right) w_{k-1}^{(k-1)}+\left(1-k \lambda^{2}+\lambda^{2}\right)\left(\left|y_{k}\right|-\left|y_{k-1}\right|\right)
\end{aligned}
$$

where the last line uses the identity of the first line for $k-1$. We thus have:

$$
w_{k-1}^{(k-1)}=\frac{1}{\left(1-(k-1) \lambda^{2}\right)}\left(1-k \lambda^{2}\right) w_{k}^{(k)}+\left|y_{k-1}\right|-\left|y_{k}\right| \geq 0,
$$

since $\left|y_{k-1}\right| \geq\left|y_{k}\right|$ and $k \leq \lambda^{-2}$.
Therefore, there exists a value $s^{*}$ such that $v^{(k)}>0$ and $w^{(k)}>0 \forall k \geq s^{*}$ and $v^{(k)} \geq 0$ or $w^{(k)} \geq 0$ $\forall k>s^{*}$. This value of $s^{*}$ can thus efficiently be found using binary search.

### 3.11 Appendix: Multi-output proximal map computation

### 3.11.1 Solving the prox problem

Returning to the multi-output setting, recall that $h_{V, W}(x)=V^{T} \sigma(W x)$ with $V \in \mathbb{R}^{p \times n}, W \in$ $\mathbb{R}^{n \times m}$ and

$$
g(V, W)=\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p} W_{i j} V_{k i} .
$$

The proximal mapping can then be written as:

$$
\begin{aligned}
\operatorname{prox}_{\lambda g}(\bar{V}, \bar{W}) & =\underset{V, W}{\operatorname{argmin}} \frac{1}{2}\|V-\bar{V}\|_{F}+\frac{1}{2}\|W-\bar{W}\|_{F}+\lambda \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p} W_{i j} V_{k i} \\
& =\underset{V, W}{\operatorname{argmin}} \sum_{i=1}^{n}\left(\frac{1}{2} \sum_{k=1}^{p}\left(V_{k i}-\bar{V}_{k i}\right)^{2}+\sum_{j=1}^{p}\left(W_{i j}-\bar{W}_{i j}\right)^{2}+\sum_{j=1}^{m} \sum_{k=1}^{p} W_{i j} V_{k i}\right) .
\end{aligned}
$$

Noting that the proximal mapping is separable with respect to the columns of $W$ and the rows of $V$, and using the same sign trick applied in the single-output case, it is enough to solve for any $i=1, \ldots, n$,

$$
\begin{equation*}
\min _{v, w \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}} h_{\lambda}(v, w ; x, y) \equiv \frac{1}{2} \sum_{k=1}^{p}\left(v_{k}-\left|x_{k}\right|\right)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda \sum_{j=1}^{m} \sum_{k=1}^{p} v_{k} w_{j}, \tag{3.21}
\end{equation*}
$$

where $x$ denotes the $i$-th row of $V$ and $y$ the $i$-th column of $W$, in order to compute the prox operator.

The stationarity conditions for (3.21) are stated next; the arguments are the same as in the single-output case.

Lemma 3.15 (Stationarity conditions). Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.21)
for a given $(x, y) \in \mathbb{R}^{p} \times \mathbb{R}^{m}$. Then

$$
\begin{aligned}
& w_{j}^{*}=\max \left\{0,\left|y_{j}\right|-\lambda \sum_{k=1}^{p} v_{k}^{*}\right\} \text { for any } j=1,2, \ldots, m \\
& v_{k}^{*}=\max \left\{0,\left|x_{k}\right|-\lambda \sum_{j=1}^{m} w_{j}^{*}\right\} \text { for any } k=1,2, \ldots, p
\end{aligned}
$$

The next lemma restates the result in Lemma 3.7 which expands on the monotonic relation in magnitude originally established for single-output networks in Corollary 3.1.

Lemma 3.16. Let $\left(v^{*}, w^{*}\right) \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}$ be an optimal solution of (3.17) for a given $(x, y) \in \mathbb{R}^{p} \times \mathbb{R}^{m}$.

1. The vector $w^{*}$ satisfies that for any $j, l \in\{1,2, \ldots, m\}$ it holds that $w_{j}^{*} \geq w_{l}^{*}$ only if $\left|y_{j}\right| \geq$ $\left|y_{l}\right|$.
2. The vector $v^{*}$ satisfies that for any $k, l \in\{1,2, \ldots, p\}$ it holds that $v_{k}^{*} \geq v_{l}^{*}$ only if $\left|x_{k}\right| \geq\left|x_{l}\right|$.
3. Let $\bar{x}, \bar{y}$ be the sorted vector of $x$ and $y$ respectively in descending magnitude order. Let $s_{\nu}=\left|\left\{k: v_{k}^{*}>0\right\}\right|$ and $s_{w}=\left|\left\{j: w_{j}^{*}>0\right\}\right|$. If $v^{*}, w^{*} \neq 0$, then

$$
\begin{align*}
& v_{k}^{*}=\left|x_{k}\right|+\frac{1}{1-s_{v} s_{w} \lambda^{2}}\left(\lambda^{2} s_{w} \sum_{l=1}^{s_{v}}\left|\bar{x}_{l}\right|-\lambda \sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|\right)  \tag{3.22}\\
& w_{j}^{*}=\left|y_{j}\right|+\frac{1}{1-s_{v} s_{w} \lambda^{2}}\left(\lambda^{2} s_{v} \sum_{l=1}^{s_{w}}\left|\bar{y}_{l}\right|-\lambda \sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|\right) \tag{3.23}
\end{align*}
$$

Proof. The two first points are direct applications of the stationary conditions of Lemma 3.15.
From the conditions in Lemma 3.15 we have that

$$
\begin{aligned}
\sum_{j=1}^{m} w_{j}^{*} & =\sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|-\lambda s_{w} \sum_{k=1}^{p} v_{k}^{*} \\
\sum_{k=1}^{p} v_{k}^{*} & =\sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|-\lambda s_{v} \sum_{j=1}^{m} w_{j}^{*} \\
& =\sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|-\lambda s_{v} \sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|+\lambda^{2} s_{\nu} s_{w} \sum_{k=1}^{p} v_{k}^{*} \\
& =\frac{1}{1-\lambda^{2} s_{\nu} s_{w}}\left(\sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|-\lambda s_{v} \sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|\right)
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\sum_{j=1}^{m} w_{j}^{*} & =\sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|-\frac{\lambda s_{w}}{1-\lambda^{2} s_{\nu} s_{w}}\left(\sum_{k=1}^{s_{v}}\left|\bar{x}_{k}\right|-\lambda s_{v} \sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|\right) \\
& =\frac{1}{1-\lambda^{2} s_{v} s_{w}}\left(-\lambda s_{w} \sum_{k=1}^{s_{\nu}}\left|\bar{x}_{k}\right|+\sum_{j=1}^{s_{w}}\left|\bar{y}_{j}\right|\right)
\end{aligned}
$$

Plugging the latter to the stationarity condition on $v^{*}$ (given in Lemma 3.15) then implies the result.

Finally, we show, as in the single-output case, that second order stationarity condition constraints the ranges of sparsities of $v^{*}$ and $w^{*}$; this relation is given by Lemma 3.8, and is proved next.

Proof of Lemma 3.8. Since ( $v^{*}, w^{*}$ ) is an optimal solution of (3.21) and the objective function in (3.21) is twice continuously differentiable, $\left(v^{*}, w^{*}\right)$ satisfies the second order necessary optimality conditions. That is, for any $d \in \mathbb{R}^{p} \times \mathbb{R}^{m}$ satisfying that $\left(v^{*}, w^{*}\right)+d \in \mathbb{R}_{+}^{p} \times \mathbb{R}_{+}^{m}$ and $d^{T} \nabla h_{\lambda}\left(v^{*}, w^{*} ; x, y\right)=0$ it holds that

$$
d^{T} \nabla^{2} h_{\lambda}\left(v^{*}, w^{*} ; x, y\right) d=d^{T}\left(\begin{array}{cc}
I_{p \times p} & \Lambda_{p \times m} \\
\Lambda_{m \times p} & I_{m \times m}
\end{array}\right) d \geq 0
$$

where the first row/ column corresponds to $v$ and the others correspond to $w, I$ denotes the identity matrix and $\Lambda$ denotes a matrix completely filled with $\lambda$. Similarly as in the single output case, we require that the submatrix of $\nabla^{2} h_{\lambda}\left(v^{*}, w^{*} ; x, y\right)$ containing the rows and columns corresponding to the positive coordinates in $\left(v^{*}, w^{*}\right)$ is positive semidefinite. Since the minimal eigenvalue of this submatrix equals $1-\lambda \sqrt{\left|S_{v}\right|\left|S_{w}\right|}$, we have $\lambda^{-2} \geq\left|S_{v}\right|\left|S_{w}\right|$.

A possible way of solving this proximal problem is thus to exhaustively compute the value of $h_{\lambda}$ at each stationary point associated with sparsities $s_{v}=1, \ldots, p, s_{w}=1, \ldots, m$ such that $s_{v} s_{w} \leq \lambda^{-2}$. However, trying all possible pairs of sparsities $\left(s_{v}, s_{w}\right)$ is computationally costly. Similarly as is the single output case, we can exploit some structure of the objective function $h_{\lambda}$ in order to reduce the possible candidate pairs of sparsities.

Without loss of generality, we assume hereafter that the vectors $x, y$ are already sorted in decreasing order of magnitude.

Lemma 3.16 shows that for each pair $\left(s_{v}, s_{w}\right), s_{v}=0,1, \ldots, p, s_{w}=0,1, \ldots, m$, there exists a stationary point $\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)}\right)$ of $h_{\lambda}(\cdot, \cdot ; x, y)$ such that $\left|\left\{k: v_{k}^{\left(s_{v}, s_{w}\right)}>0\right\}\right|=s_{v}, \mid\left\{j: w_{j}^{\left(s_{v}, s_{w}\right)}>\right.$
$0\} \mid=s_{w}$, given by

$$
\begin{align*}
& v_{k}^{\left(s_{v}, s_{w}\right)}=\left|x_{k}\right|+\frac{1}{1-s_{v} s_{w} \lambda^{2}}\left(\lambda^{2} s_{w} \sum_{l=1}^{s_{v}}\left|x_{l}\right|-\lambda \sum_{j=1}^{s_{w}}\left|y_{j}\right|\right) \text { for } k=1,2, \ldots, s_{v}, \text { and } v_{k}^{\left(s_{v}, s_{w}\right)}=0 \text { else } \\
& w_{j}^{\left(s_{v}, s_{w}\right)}=\left|y_{j}\right|+\frac{1}{1-s_{\nu} s_{w} \lambda^{2}}\left(\lambda^{2} s_{\nu} \sum_{l=1}^{s_{w}}\left|y_{l}\right|-\lambda \sum_{k=1}^{s_{v}}\left|x_{k}\right|\right) \text { for } j=1,2, \ldots, s_{w}, \text { and } w_{j}^{\left(s_{v}, s_{w}\right)}=0 \text { else. } \tag{3.24}
\end{align*}
$$

We now move to prove the monotonicity property stated in Lemma 3.9.

Proof of Lemma 3.9. The proof follows exactly the same lines as in the single output case. We recall the definition of the objective function:

$$
h_{\lambda}(\nu, w ; x, y) \equiv \frac{1}{2} \sum_{k=1}^{p}\left(v_{k}-\left|x_{k}\right|\right)^{2}+\frac{1}{2} \sum_{j=1}^{m}\left(w_{j}-\left|y_{j}\right|\right)^{2}+\lambda\left(\sum_{k=1}^{p} v_{k}\right)\left(\sum_{j=1}^{m} w_{j}\right)
$$

Plugging the definitions from equation (3.24), we have

$$
\begin{align*}
& h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)} ; x, y\right)=\frac{s_{v}}{2}\left(\frac{1}{1-\lambda^{2} s_{v} s_{w}}\left(\lambda^{2} s_{w} \sum_{k=1}^{s_{v}}\left|x_{k}\right|-\lambda \sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\right)^{2}+\frac{1}{2} \sum_{k=s_{v}+1}^{p} x_{k}^{2} \\
& \quad+\frac{s_{w}}{2}\left(\frac{1}{1-\lambda^{2} s_{\nu} s_{w}}\left(\lambda^{2} s_{v} \sum_{j=1}^{s_{w}}\left|y_{j}\right|-\lambda \sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)\right)^{2}+\frac{1}{2} \sum_{j=s_{w}+1}^{m} y_{j}^{2} \\
& \quad+\frac{\lambda}{\left(1-\lambda^{2} s_{v} s_{w}\right)^{2}}\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|-\lambda s_{v} \sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\left(-\lambda s_{w} \sum_{k=1}^{s_{\nu}}\left|x_{k}\right|+\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right) \\
& =\frac{1}{2\left(1-\lambda^{2} s_{v} s_{w}\right)^{2}}\left(\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}\left(\lambda^{4} s_{\nu} s_{w}^{2}+\lambda^{2} s_{w}-2 \lambda^{2} s_{w}\right)+\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)^{2}\left(\lambda^{2} s_{\nu}+\lambda^{4} s_{\nu}^{2} s_{w}-2 \lambda^{2} s_{v}\right)\right. \\
& \left.\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\left(-2 \lambda^{3} s_{\nu} s_{w}-2 \lambda^{3} s_{\nu} s_{w}+2 \lambda+2 \lambda^{3} s_{\nu} s_{w}\right)\right)+\frac{1}{2} \sum_{k=s_{v}+1}^{p} x_{k}^{2}+\frac{1}{2} \sum_{j=s_{w}+1}^{m} y_{j}^{2} \tag{3.25}
\end{align*}
$$

$$
\begin{align*}
= & \frac{1}{2\left(1-\lambda^{2} s_{v} s_{w}\right)}\left(-\lambda^{2} s_{w}\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}-\lambda^{2} s_{v}\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)^{2}+2 \lambda\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\right)  \tag{3.26}\\
& +\frac{1}{2} \sum_{k=s_{v}+1}^{p} x_{k}^{2}+\frac{1}{2} \sum_{j=s_{w}+1}^{m} y_{j}^{2}  \tag{3.28}\\
= & \left(1+\frac{\lambda^{2} s_{v}}{1-\lambda^{2} s_{v} s_{w}}\right) \frac{1}{2\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(-\lambda^{2}\left(s_{w}-1\right)\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}-\lambda^{2}\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}\right. \\
& \left.-\lambda^{2} s_{v}\left(\left(\sum_{j=1}^{s_{w}-1}\left|y_{j}\right|\right)^{2}+2 \lambda\left|y_{s_{w}}\right| \sum_{j=1}^{s_{w}-1}\left|y_{j}\right|+y_{s_{w}}^{2}\right)+2 \lambda \sum_{k=1}^{s_{v}}\left|x_{k}\right|\left(\sum_{j=1}^{s_{w}-1}\left|y_{j}\right|+\left|y_{s_{w}}\right|\right)\right)  \tag{3.29}\\
& +\frac{1}{2} \sum_{k=s_{v}+1}^{p} x_{k}^{2}+\frac{1}{2} \sum_{j=s_{w}-1+1}^{m} y_{j}^{2}-\frac{1}{2} y_{s_{w}}^{2} . \tag{3.30}
\end{align*}
$$

By applying equation (3.25) at $s_{v}, s_{w}-1$, we can express the right hand side of equation (3.29) in terms of $h_{\lambda}\left(\nu^{\left(s_{\nu}, s_{w}-1\right)}, w^{\left(s_{\nu}, s_{w}-1\right)} ; x, y\right)$ as:

$$
\begin{aligned}
& h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)} ; x, y\right)=h_{\lambda}\left(\nu^{\left(s_{v}, s_{w}-1\right)}, w^{\left(s_{v}, s_{w}-1\right)} ; x, y\right)+\frac{1}{2\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(-\lambda^{2}\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}\right. \\
& \left.\quad-\lambda^{2} s_{\nu}\left|y_{s_{w}}\right|\left(2 \sum_{j=1}^{s_{w}-1}\left|y_{j}\right|+\left|y_{s_{w}}\right|\right)+2 \lambda\left|y_{s_{w}}\right| \sum_{k=1}^{s_{\nu}}\left|x_{k}\right|\right) \\
& \quad+\frac{\lambda^{2} s_{v}}{2\left(1-\lambda^{2} s_{v} s_{w}\right)\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(-\lambda^{2} s_{w}\left(\sum_{k=1}^{s_{\nu}}\left|x_{k}\right|\right)^{2}\right. \\
& \left.\quad-\lambda^{2} s_{\nu}\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)^{2}-2 \lambda\left(\sum_{k=1}^{s_{\nu}}\left|x_{k}\right|\right)\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\right)-\frac{1}{2} y_{s_{w}}^{2} .
\end{aligned}
$$

Therefore:

$$
\begin{aligned}
& h_{\lambda}\left(v^{\left(s_{v}, s_{w}\right)}, w^{\left(s_{v}, s_{w}\right)} ; x, y\right)-h_{\lambda}\left(v^{\left(s_{v}, s_{w}-1\right)}, w^{\left(s_{v}, s_{w}-1\right)} ; x, y\right) \\
& =-\frac{1}{2\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(-2 \lambda\left|y_{s_{w}}\right|\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|-\lambda s_{v} \sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)-\lambda^{2} s_{v}\left|y_{s_{w}}\right|^{2}+\lambda^{2}\left(\sum_{k=1}^{s_{v}}\left|x_{k}\right|\right)^{2}\right. \\
& +\frac{\lambda^{2} s_{\nu}}{1-\lambda^{2} s_{\nu} s_{w}}\left(\lambda^{2} s_{w}\left(\sum_{k=1}^{s_{\nu}}\left|x_{k}\right|\right)^{2}+\lambda^{2} s_{\nu}\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)^{2}-2 \lambda\left(\sum_{k=1}^{s_{\nu}}\left|x_{k}\right|\right)\left(\sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)\right) \\
& \left.+\left(1-\lambda^{2} s_{v} s_{w}+\lambda^{2} s_{v}\right)\left|y_{s_{w}}\right|\right) \\
& =-\frac{1}{2\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(\left(1-\lambda^{2} s_{\nu} s_{w}\right) y_{s_{w}}^{2}-2 \lambda\left|y_{s_{w}}\right|\left(1-\lambda^{2} s_{v} s_{w}\right) \sum_{k=1}^{s_{v}} v_{k}^{\left(s_{v}, s_{w}\right)}\right. \\
& \left.+\frac{\lambda^{2}}{1-\lambda^{2} s_{\nu} s_{w}}\left(\sum_{k=1}^{s_{\nu}}\left|x_{k}\right|-\lambda s_{\nu} \sum_{j=1}^{s_{w}}\left|y_{j}\right|\right)^{2}\right) \\
& =-\frac{1-\lambda^{2} s_{v} s_{w}}{2\left(1-\lambda^{2} s_{v}\left(s_{w}-1\right)\right)}\left(\left|y_{s_{w}}\right|-\lambda \sum_{k=1}^{s_{v}} v_{k}^{\left(s_{v}, s_{w}\right)}\right)^{2} .
\end{aligned}
$$

The second result is obtain directly by symmetry between $v$ and $w$.

In order to derive an efficient algorithm, we will again exploit the monotone property of the feasibility criterion $v^{\left(s_{v}, s_{w}\right)}>0, w^{\left(s_{v}, s_{w}\right)}>0$ restated from Lemma 4.9:

Lemma 3.17 (Restatement of Lemma 4.9). Let $(k, l) \in[p] \times[m]$ be such that $k l \leq \lambda^{-2}$. Suppose that

$$
v^{(k, l)} \geq 0 \text { and } w^{(k, l)} \geq 0
$$

Then for any $i=1, \ldots, k$ and any $j=1, \ldots, l$, it holds that

$$
v^{(i, j)} \geq 0 \text { and } w^{(i, j)} \geq 0
$$

Proof of Lemma 4.9. Since the first $k$ entries of $v^{(k, l)}$ are ordered in decreasing order, we have that $v^{(k, l)} \geq 0$ if and only if $v_{k}^{(k, l)} \geq 0$. Similarly, $w^{(k, l)} \geq 0$ if and only if $w_{l}^{(k, l)} \geq 0$.

Suppose that $v^{(k, l)} \geq 0$ and $w^{(k, l)} \geq 0$. By induction, in order to prove the result, it is sufficient to prove that $v_{k-1}^{(k-1, l)} \geq 0, v_{k}^{(k, l-1)} \geq 0, w_{l}^{(k-1, l)} \geq 0$ and $w_{l-1}^{(k, l-1)} \geq 0$. We only prove the result for $v$, as the proof for $w$ is identical.

Using equation (3.24), we have:

$$
\begin{align*}
\left(1-k l \lambda^{2}\right) v_{k}^{(k, l)}= & \left(1-k l \lambda^{2}\right)\left|x_{k}\right|+\lambda^{2} l \sum_{i=1}^{k}\left|x_{i}\right|-\lambda \sum_{j=1}^{l}\left|y_{j}\right|  \tag{3.31}\\
= & \left(1-k l \lambda^{2}\right)\left|x_{k}\right|+\left(1-(k-1) l \lambda^{2}\right)\left|x_{k-1}\right|-\left(1-(k-1) l \lambda^{2}\right)\left|x_{k-1}\right|  \tag{3.32}\\
& \quad+\lambda^{2} l \sum_{i=1}^{k-1}\left|x_{i}\right|+\lambda^{2} l\left|x_{k}\right| \lambda \sum_{j=1}^{l}\left|y_{j}\right| \\
= & \left(1-(k-1) l \lambda^{2}\right) v_{k-1}^{(k-1, l)}+\left(1-(k-1) l \lambda^{2}\right)\left(\left|x_{k}\right|-\left|x_{k-1}\right|\right) .
\end{align*}
$$

Therefore:

$$
v_{k-1}^{(k-1, l)}=\frac{1-(k-1) l \lambda^{2}}{1-k l \lambda^{2}} v_{k}^{(k, l)}+\left|x_{k-1}\right|-\left|x_{k}\right| \geq 0
$$

since the vector $x$ is ordered in decreasing order of magnitude, and thus $\left|x_{k-1}\right|-\left|x_{k}\right| \geq 0$.
Using again equation (3.31), we have:

$$
\begin{aligned}
\left(1-k l \lambda^{2}\right) v_{k}^{(k, l)} & =\left(1-k l \lambda^{2}\right)\left|x_{k}\right|+\left(1-k(l-1) \lambda^{2}\right)\left|x_{k}\right|-\left(1-k(l-1) \lambda^{2}\right)\left|x_{k}\right| \\
& +\lambda^{2}(l-1) \sum_{i=1}^{k}\left|x_{i}\right|+\lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right|-\lambda \sum_{j=1}^{l-1}\left|y_{j}\right|-\lambda\left|y_{l}\right| \\
& =\left(1-k(l-1) \lambda^{2}\right) v_{k}^{(k, l-1)}-k \lambda^{2}\left|x_{k}\right|+\lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right|-\lambda\left|y_{l}\right|
\end{aligned}
$$

where the last equality follows (again) from equation (3.31) for $v_{k}^{(k, l-1)}$. Thus,

$$
\begin{equation*}
\left(1-k(l-1) \lambda^{2}\right) v_{k}^{(k, l-1)}=\left(1-k l \lambda^{2}\right) v_{k}^{(k, l)}+k \lambda^{2}\left|x_{k}\right|-\lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right|+\lambda\left|y_{l}\right| \tag{3.33}
\end{equation*}
$$

From the definition of $v_{k}^{(k, l)}$ (equation (3.24)), we have that $v_{k}^{(k, l)} \geq 0$ is equivalent to the condition:

$$
\left|x_{k}\right| \geq \frac{\lambda \sum_{j=1}^{l}\left|y_{j}\right|-l \lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right|}{1-k l \lambda^{2}}
$$

Plugging this inequality in equation (3.33), we obtain:

$$
\begin{align*}
\left(1-k(l-1) \lambda^{2}\right) v_{k}^{(k, l-1)} \geq & \left(1-k l \lambda^{2}\right) v_{k}^{(k, l)}+\frac{k \lambda^{2}}{1-k l \lambda^{2}}\left(\lambda \sum_{j=1}^{l}\left|y_{j}\right|-l \lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right|\right)+\lambda\left|y_{l}\right|-\lambda^{2} \sum_{i=1}^{k}\left|x_{i}\right| \\
= & \left(1-k l \lambda^{2}\right) v_{k}^{(k, l)}+\frac{\lambda}{1-k l \lambda^{2}}(  \tag{3.34}\\
& \left.\quad k \lambda^{2} \sum_{j=1}^{l}\left|y_{j}\right|-k l \lambda^{3} \sum_{i=1}^{k}\left|x_{i}\right|+\left(1-k l \lambda^{2}\right)\left|y_{l}\right|-\lambda\left(1-k l \lambda^{2}\right) \sum_{i=1}^{k}\left|x_{i}\right|\right) \\
= & \left(1-k l \lambda^{2}\right) v_{k}^{(k, l)}+\frac{\lambda}{1-k l \lambda^{2}}\left(k \lambda^{2} \sum_{j=1}^{l}\left|y_{j}\right|+\left(1-k l \lambda^{2}\right)\left|y_{l}\right|-\lambda \sum_{i=1}^{k}\left|x_{i}\right|\right) \tag{3.35}
\end{align*}
$$

From the definition of $w_{l}^{(k, l)}$ (equation (3.24)), we have that $w_{l}^{(k, l)} \geq 0$ is equivalent to the condition:

$$
\begin{equation*}
\left(1-k l \lambda^{2}\right)\left|y_{l}\right|+k \lambda^{2} \sum_{j=1}^{l}\left|y_{j}\right|-\lambda \sum_{i=1}^{k}\left|x_{i}\right| \geq 0 \tag{3.36}
\end{equation*}
$$

Since the expression of equation (3.36) is exactly the same as the one inside the parentheses of equation (3.35), plugging this relation to (3.33) thus shows that ( $\left.1-k(l-1) \lambda^{2}\right) \nu_{k}^{(k, l-1)} \geq 0$, i.e. $v_{k}^{(k, l-1)} \geq 0$.

We now introduce the efficient procedure to compute the maximal feasibility boundary (MFB), and prove that it indeed delivers, as promised, all sparsity pairs in the MFB set.

Lemma 3.18. The set $S$ returned by Algorithm 3.5 contains all, and only, the sparsity pairs that are on the maximal feasibility boundary.

Proof. First recall that the MFB is defined as all pairs $\left(s_{v}, s_{w}\right) \in\{0, \ldots, p\} \times\{0, \ldots, m\}$ satisfying the conditions:

1. $v_{s_{v}}^{\left(s_{v}, s_{w}\right)}>0$ and $w_{s_{w}}^{\left(s_{v}, s_{w}\right)}>0$ and $s_{v} s_{w} \leq \lambda^{-2}$,
2. $v_{s_{v}+1}^{\left(s_{v}+1, s_{w}\right)} \leq 0$ or $w_{s_{w}}^{\left(s_{v}+1, s_{w}\right)} \leq 0$ or $\left(s_{v}+1\right) s_{w}>\lambda^{-2}$ or $s_{v}=p$,
3. $v_{s_{v}}^{\left(s_{v}, s_{w}+1\right)} \leq 0$ or $w_{s_{w}+1}^{\left(s_{v}, s_{w}+1\right)} \leq 0$ or $s_{v}\left(s_{w}+1\right)>\lambda^{-2}$ or $s_{w}=m$.

Algorithm 3.5 plays on the properties of feasibility-infeasibility of the sparsity levels to build the MFB. We say that a pair of the sparsity levels of $v$ and $w\left(s_{v}, s_{w}\right)$ is feasible if $v_{s_{v}}^{\left(s_{v}, s_{w}\right)} \geq 0$, $w_{s_{w}}^{\left(s_{v}, s_{w}\right)} \geq 0$ and $s_{v} s_{w}<\lambda^{-2}$, and denote this by the property $P(i, j)$, i.e.

$$
(i, j) \text { is feasible } \Leftrightarrow P(i, j)
$$

```
Algorithm 3.5 Finding sparsity pairs on the maximal feasibility boundary
Input: \(x \in \mathbb{R}^{p}, y \in \mathbb{R}^{m}\) ordered in decreasing magnitude order, \(\lambda>0\).
    \(s_{v} \leftarrow 0, s_{w} \leftarrow m\)
    \(S \leftarrow \varnothing\)
    maximal \(\leftarrow\) True
    while \(s_{\nu} \leq p\) and \(s_{w} \geq 0\) do
        Compute \(v_{s_{v}}^{\left(s_{v}, s_{w}\right)}\) and \(w_{s_{w}}^{\left(s_{v}, s_{w}\right)}\) as shown in equation (3.24)
        if \(v_{s_{v}}^{\left(s_{v}, s_{w}\right)}<0\) or \(w_{s_{w}}^{\left(s_{v}, s_{w}\right)}<0\) or \(s_{v} s_{w} \geq \lambda^{-2}\) then
            if maximal then
                \(S \leftarrow S \cup\left\{\left(s_{v}-1, s_{w}\right)\right\}\)
                maximal \(\leftarrow\) False
            end if
            \(s_{w} \leftarrow s_{w}-1\)
        else
            \(s_{v} \leftarrow s_{v}+1\)
            maximal \(\leftarrow\) True
        end if
    end while
    if \(s_{v}==p+1\) then
        \(S \leftarrow S \cup\left\{\left(s_{\nu}-1, s_{w}\right)\right\}\)
    end if
    return S
```

Our claim can be read as: Let $(i, j) \in\{0, \ldots, p\} \times\{0, \ldots, m\}$, then $(i, j)$ is added to $S$ by Algorithm 3.5 if and only if $(i, j)$ belongs to the MFB, i.e.,

$$
(i, j) \in \operatorname{MFB} \Leftrightarrow(i, j) \in S
$$

Obviously, only feasible sparsity pairs belong to the MFB, and it is quite easy to see that only feasible sparsity pairs will belong to an output $S$ of Algorithm 3.5. Indeed, Algorithm 3.5 monotonically decrements $s_{w}$ starting from $s_{w}=m$ and increments $s_{v}$ starting from $s_{v}=0$. For each value of $s_{w}$, it increases $s_{v}$ while the current pair ( $s_{v}, s_{w}$ ) is feasible (lines 12-15). Once it reaches an infeasible point $\left(i, s_{w}\right)$, and in the case where $s_{v}$ has been increased at least once for this particular value of $s_{w}$, it adds to $S$ the pair encountered just before, i.e., $\left(i-1, s_{w}\right)$, and then decrements $s_{w}$ (lines 6-11).

We first prove the $\Rightarrow$ statement. Suppose that some pair $(i, j)$ belongs to the MFB. Let us first leave aside the corner cases, and assume that $i<p$ and $j<m$.

Suppose first that $s_{w}$ reaches $j$ before $s_{v}$ reaches $i$, i.e., $s_{\nu}<i$. Since the pair $(i, j)$ is feasible, and due to the monotonicity property of the feasibility condition (Lemma 3.9), all pairs ( $k, s_{w}$ ) with $k \leq i$ must be feasible. Therefore, $s_{\nu}$ will be increased until reaching $i+1$. By definition of the MFB, the pair $(i+1, j)$ must be infeasible. Since $s_{\nu}$ has necessarily been increased at least once for this value of $s_{w}=j$, and so the pair $(i+1-1, j)=(i, j)$ will be added to $S$ before

## Chapter 3. Efficient Proximal Mapping of the 1-path-norm of Shallow Networks

decrementing $s_{w}$.
In the special case where $i=p$, no infeasible point will be found. The loop will thus finish with $s_{w}=j$ and $s_{v}=p+1$. The condition at line 17 will thus hold, and the pair $(p, j)$ will be added to $S$.

Suppose now that $s_{v}$ reaches $i$ before $s_{w}$ reaches $j$, i.e., $s_{w}>j$. Since $(i, j)$ is in the MFB, then the pair $(i, j+1)$ must be infeasible. Thanks to the monotonicity property of the feasibility condition (Lemma 3.9), all pairs ( $s_{v}, k$ ) with $k \geq i$ must also be infeasible. Therefore, $s_{w}$ will be decreased until reaching $s_{w}=j$. Then, similarly as in the previous case, since $(i, j)$ is feasible, $s_{v}$ will be increased, and the pair $(i, j)$ added to $S$.

We now prove the $\Leftarrow$ statement. We show that if $(i, j)$ is added to $S$, then it must belong to the MFB, i.e., it satisfies all three properties recalled in the beginning of the proof.

Let us first show that for each pair $\left(s_{v}, s_{w}\right)$ encountered during the algorithm, the pair ( $s_{v}-1, s_{w}$ ) is always feasible (or $s_{\nu}=0$ ). We can show that this property is conserved each time the algorithm either increases $s_{v}$ or decreases $s_{w}$. First note that the pair $(0, m)$ is always feasible. The algorithm will then necessarily first goes to the pair $(1, m)$ and $P(1, m)$ is true. Then suppose that $P\left(s_{v}, s_{w}\right)$ is true for some pair $\left(s_{v}, s_{w}\right)$ encountered during the algorithm. Then, if $s_{v}$ is increases, it means that the pair $\left(s_{v}, s_{w}\right)$ is feasible. The next encountered pair is then $\left(s_{v}+1, s_{w}\right)$ and $P\left(s_{v}+1, s_{w}\right)$ is true. On the other hand, suppose that $s_{w}$ is decreased. The next encountered pair is thus $\left(s_{v}, s_{w}-1\right)$. Since $P\left(s_{v}, s_{w}\right)$ is true, it means that $\left(s_{v}-1, s_{w}\right)$ is feasible. By Lemma 3.9, it implies that ( $s_{v}-1, s_{w}-1$ ) is also feasible, and thus $P\left(s_{v}, s_{w}-1\right)$ is true. We thus proved that $P\left(s_{v}, s_{w}\right)$ is true for any pair $\left(s_{v}, s_{w}\right)$ encountered during the algorithm. Therefore, since any pair added to $S$ is of the form $\left(s_{v}-1, s_{w}\right)$ for some pair ( $s_{v}, s_{w}$ ) encountered during the algorithm, then any pair added to $S$ must be feasible.

The second property of the MFB is straightforward to show. Indeed, if $(i-1, j)$ is added to $S$, it means that the pair $(i, j)$ is infeasible due to condition on line 6.

Finally, the third property follows from the fact that, when reaching $s_{w}=j, s_{v}$ must be increased at least once for adding a pair of the form $(i, j)$ to $S$. Let $s_{v}^{(j)}$ be the value of $s_{v}$ when the algorithm reaches $s_{w}=j$. We necessarily have $s_{v}^{(j)} \leq i$. This implies that the pair $\left(s_{v}^{(j)}, j+1\right)$ is infeasible, otherwise $s_{v}$ would have been increased to a greater value at the previous value $s_{w}=j+1$. By Lemma 3.9, and since $s_{v}^{(j)} \leq i$ this implies that the pair $(i, j+1)$ is also infeasible, hence the result.

Time complexity of Algorithm 3.5. At each iteration of the loop, either $s_{v}$ is incremented by 1 or $s_{w}$ is decremented by 1 . Since $s_{v}$ starts from 0 and $s_{w}$ from $m$, and that the stopping criterion is $s_{v}>p$ or $s_{w}<0$, it follows that the maximal number of iterations inside the loop is $m+p$. At each iteration, we must compute $\nu_{s_{v}}^{\left(s_{v}, s_{w}\right)}$ and $w_{s_{w}}^{\left(s_{v}, s_{w}\right)}$, which requires in particular to compute
$\sum_{k=1}^{s_{v}}\left|x_{k}\right|$ and $\sum_{j=1}^{s_{w}}\left|y_{j}\right|$. However, these cumulative sums can be efficiently computed before the loop in time $\mathscr{O}(m+p)$, so that computing $v_{s_{v}}^{\left(s_{v}, s_{w}\right)}$ and $w_{s_{w}}^{\left(s_{v}, s_{w}\right)}$ inside the loop can be done in constant time. The overall complexity of this algorithm is thus $\mathscr{O}(m+p)$.

Moreover, we can see that each time we add a pair to $S$, we must both decrement $s_{w}$ by 1 (just after adding the element in the algorithm), and increment $s_{v}$ by 1 (in order for the boolean maximal to become true again). Therefore, there can be at most $\min (m, p)$ pairs in the final set $s$ at the end of the algorithm.

Merging all previous results, we can finally prove Theorem 3.4.

Proof of Theorem 3.4. Thanks to the separability argument, it is sufficient to prove that Algorithm 3.3 returns a solution of problem (3.11).

Lemma 3.7 states that given the number of nonzero elements $s_{\nu}=\left|\left\{k: v_{k}^{*}>0\right\}\right|, s_{w}=\mid\left\{j: w_{j}^{*}>\right.$ $0\} \mid$, the optimal solution $\left(v^{*}, w^{*}\right)$ can be obtained in close form (equations (3.12), (3.13)).

Due the monotonicity property of the objective function $h_{\lambda}$ (Lemma 3.9), it follows that the sparsity pair $\left(s_{v}, s_{w}\right)$ of the optimal solution must lie on the MFB. Indeed, if it does not lie on the MFB, then it means that the candidate solution associated with either the sparsity pair $\left(s_{v}+1, s_{w}\right)$ or $\left(s_{v}, s_{w}+1\right)$ must be feasible. According to Lemma 3.9, this pair would then yield a lower value of $h_{\lambda}$, and would then be a better solution.

Algorithm 3.3 computes the candidate solution associated with all sparsity pair lying on the MFB, and returns the one achieving the lowest value of $h_{\lambda}$. Therefore, the returned solution must necessarily be the optimal solution.

### 3.12 Appendix: Experimental details and other plots

We consider the following values for the parameters that determine the training loop:

```
\triangleright batch size: }10
\triangleright epochs: }2
\triangleright learning rate: 1e-1, 1e-2, 1e-3, 1e-4, 5e-1, 5e-2, 5e-3, 5e-4
| dataset:mnist, fmnist, kmnist
\triangleright hidden neurons: 200
\triangleright ~ l a m b d a ~ ( ~ \lambda ) : 0 . , ~ 1 e - 5 , ~ 1 e - 4 , ~ 1 e - 3 , ~ 1 e - 2 , ~ 1 e - 1 , ~ 1 e 0 , ~ 1 e 1 , ~ 1 e 2 , ~ 2 e - 5 , ~ 2 e - 4 ,
    2e-3, 2e-2, 2e-1, 2e0, 2e1, 2e2, 3e-5, 3e-4, 3e-3, 3e-2, 3e-1, 3e0, 3e1,
    3e2, 4e-5, 4e-4, 4e-3, 4e-2, 4e-1, 4e0, 4e1, 4e2, 5e-5, 5e-4, 5e-3, 5e-2,
    5e-1, 5e0, 5e1, 5e2
```



Figure 3.4: percentage of nonzero weights in the network, as a function of iteration count (path regularization - fmnist dataset).

The $\ell_{\infty}$-bounded adversarial examples used to evaluate the robustness of the networks were generated using the PGD method described in (Madry et al., 2018b) and implemented in the advertorch toolbox (https:// github.com/BorealisAI/advertorch) using the following parameters:

```
\triangleright epsilon: 0.05, 0.1, 0.15, 0.2, 0.25, 0.3
 iterations: 40
\triangleright step size: epsilon / 20
random initialization: True
```


### 3.12.1 sparsity per iteration

One advantage of the proximal mapping of the 1-path-norm and the $\ell_{1}$-norm is that they can set many weights to exactly zero. This has the effect of providing sparse networks from early iterations. This is in contrast to SGD with a constant stepsize which does not generate sparse iterates. In Figures 4, 5, 6 and 7 we plot the percentage of nonzero weights as a function of the iteration count, for both plain SGD and proximal SGD. We observe that in fact this is the case, and that the sparsity of the $\ell_{1}$ and 1-path-norm regularized network can be controlled with the regularization parameter $\lambda$.

### 3.12.2 Robustness vs accuracy tradeoff

For all possible values of $\lambda$, in Figure 8 we plot the data corresponding to the lerning rate with least error. We plot the value of the error on clean samples and the error on adversarial


Figure 3.5: percentage of nonzero weights in the network, as a function of iteration count (path regularization - kmnist dataset).


Figure 3.6: percentage of nonzero weights in the network, as a function of iteration count ( $\ell_{1}$ regularization - fmnist dataset).


Figure 3.7: percentage of nonzero weights in the network, as a function of iteration count ( $\ell_{1}$ regularization - kmnist dataset).


Figure 3.8: Robustness vs accuracy tradeoff for the different regularizers studied.
examples. This allows us to understand the tradeoff between accuracy and robustness that is controlled by the regularization paramter $\lambda$.

### 3.13 Bibliographic Note

The candidate proposed the original idea of this paper and derived one of the main results (theorem 3.1). The candidate also wrote code together with P. Rolland, and obtained all the numerical results.

# 4 How to train your 1-path-norm regularized Deep Neural Network 

Fabian Latorre, Antoine Bonnet, Nadav Hallak, Paul Rolland and Volkan Cevher. Under Submission.


#### Abstract

The so-called path-norm measure is considered one of the best indicators for good generalization of neural networks. However, its direct implementation in mathematical programming for the purpose of training deep networks is still unexplored. This paper presents different alternatives to optimize the regularized objective, including automatic differentiation (AD) and a proposed proximal gradient framework applicable to general deep architectures. We address the resulting nonconvex nonsmooth optimization model by transforming the intractable induced proximal operation to an equivalent differentiable program. This facilitates the use of first-order algorithms with momentum to obtain a suboptimal - but good enough - solution. We demonstrate the advantages of this approach over AD in the task of solving the proximal mapping objective. Additionaly, numerical experiments on FashionMNIST and CIFAR10 show that for fully-connected architectures, 1-path-norm regularization outperforms weight-decay in terms of accuracy and robustness to noisy labels or noisy data. The differentiable reformulation of the prox shows an advantage in the presence of large perturbations of the data, where its unique path-sparsification properties become important.


### 4.1 Introduction

In the realm of supervised deep learning, assessing the generalization ability of neural networks is essential. This is reflected by the interest of the community in finding complexity measures to predict generalization, e.g., Jiang et al. (2020).

One measure gaining prominence is the path-norm measure Neyshabur et al. (2015c), which quantifies the complexity and length of the paths taken by the network during inference. Comprehensive numerical experiments Jiang et al. (2020) found that among norm-based and margin-based measures, the path-norm of a neural network is the most positively-correlated with generalization (see Table 2 therein). Supplementing this, Dziugaite et al. (2020) and

Vakanski and Xian (2021) show that the path-norm is a robust generalization measure. The positive empirical observations on the path-norm are further supported from a theoretical perspective by formal excess-risk bounds obtained via a Rademacher complexity analysis of path-norm-bounded networks; see Neyshabur et al. (2015c); Barron and Klusowski (2019) on ReLU networks, and E et al. (2020) on Residual Networks.

Among the family of path-norm complexity measures, the so-called 1-path-norm stands out as the only one providing width-independent generalization bounds for ReLU networks Neyshabur et al. (2015c). This is a striking property, as any excess-risk bound depending on the width of the network is otherwise incapable of explaining the good generalization performance of overparametrized networks with increasing width e.g., Neyshabur et al. (2015b, 2019a); Novak et al. (2018).

The discussion above provides clear motivation for taking into account the path-norm measure in the training process. A natural and sound manner to do so is to employ a regularization element in the optimization model. However, the fact that the 1-path-norm is a non-smooth function poses great difficulty to the automatic-differentiation (AD) paradigm: SGD/Adam are in theory designed to minimize differentiable objectives. Moreover, AD modules of deep learning software such as PyTorch Paszke et al. (2019), or TensorFlow Abadi et al. (2015), may compute incorrect gradient information of compositions of non-differentiable functions Kakade and Lee (2018); Bolte and Pauwels (2019). Overall, the frameworks that are commonly used by practitioners are, in theory, problematic to utilize for 1-path-norm regularization.

For example, the authors of Jiang et al. (2020); Dziugaite et al. (2020); Vakanski and Xian (2021) refrain from using AD methods to perform explicit regularization. As it is argued in Jiang et al. (2020), the optimization algorithm could fail, thus providing no conclusion about the qualities of the regularization scheme. In contrast, Proximal gradient methods (Parikh et al., 2014) have recently been gaining traction in the context of deep learning Yang et al. (2020), and can provide stronger guarantees for the minimization of nonconvex and nonsmooth losses (Bolte et al., 2013).

Proximal gradient methods (Parikh et al., 2014) require access to the so-called proximal mapping operation. Notwithstanding, the path-norm expression is nonconvex and nonsmooth, and obtaning an efficient closed-form solution of its proximal operator is a hard problem in and of itself. Currently, an efficient proximal mapping for the 1-path-norm is only available for shallow networks Latorre et al. (2020c). This limits the impact of 1-path-norm regularization, as modern deep neural network architectures are composed of more than one hidden layer. In contrast, the sharpness of minima Keskar et al. (2017), which is another measure that stands out as a good indicator of generalization Dziugaite et al. (2020), has specialized algorithms to implement it Foret et al. (2021) and is compatible with arbitrary deep architectures.

In this work, our goal is to achieve generalization by employing 1-path-norm regularization, and assess empirically whether the resulting objective can be succesfully minimized using AD or proximal gradient -based schemes. Since the path-norm is nonsmooth and nonconvex, its
related proximal map cannot be handled directly. To tackle this, our approach plays on the properties of the 1-path-norm and the definition of the proximal mapping.

Our contributions. We demonstrate theoretical and practical benefits of 1-path-norm regularization of Deep Neural Networks, and we perform an experimental study about the effectiveness of certain algorithms for its optimization. To summarize:

- We establish a theoretical connection that relates the 1-path-norm to the Lipschitz constant of networks with arbitrary depth with either differentiable or ReLU activations (Theorem 4.1). This generalizes the result of (Latorre et al., 2020c, Theorem 1) for shallow networks with differentiable activations, and corresponds to the multi-layer architectures used in contemporary deep learning.
- We develop a tractable procedure to compute the non-convex non-smooth proximal mapping operator of the 1-path-norm for networks of unit-width and arbitrary depth, with optimality guarantees section 4.8. In synthetic data we illustrate it provides faster optimization convergence c.f., figure 4.1.
- For networks of arbitrary width and depth, we introduce an approximate proximal gradient scheme that requires only forward and backward passes through a slight modification of the network architecture (Algorithm 4.4). The scheme is based on a differentiable reformulation of the proximal mapping problem, and uses first-order methods to avoid the hardness of computing the exact proximal mapping. We verify that this reformulation leads to better solutions of the proximal mapping c.f., figure 4.2.
- We perform an experimental study about the effects of 1-path-norm regularization on the FashionMNIST (Xiao et al., 2017a) and CIFAR10 (Krizhevsky, 2009) image classification tasks. On multilayer fully-connected architectures (MLPs) we show how 1-path-norm regularization improves the classification error and leads to more robust models in the presence of noisy training labels or perturbations in the data, compared to L2 (weight-decay) or no regularization. Whereas for plain accuracy all 1-path-norm optimization methods (including AD) achieve similar performance, the proximal methods enjoy an edge in the robustness tasks. For convolutional architectures (CNNs) we confirm the 1-path-norm is not so effective, given that by designs CNNs already have a greatly reduced effective number of paths.

Remark. the related algorithm Path-SGD Neyshabur et al. (2015a) is not an algorithm for 1-path-norm regularization. Rather, it is designed to be a rescaling-invariant optimization algorithm. For this reason, it is not meaningful to compare Path-SGD with our proposed algorithms.

### 4.2 Preliminaries and Problem Statement

For an $L$-layer feedforward neural network $f_{W}(x):=W^{L} \sigma\left(W^{L-1} \sigma\left(\cdots \sigma\left(W^{1} x\right) \cdots\right)\right)$ with a differentiable activation function $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ and weight matrices with dimensions determined
by a sequence of layer sizes ${ }^{1} d_{0}, \ldots, d_{L-1}, d_{L}$, its 1-path-norm (Neyshabur et al., 2015c) can be defined as:

$$
\begin{equation*}
P_{1}(W):=\mathbb{1}^{T}\left|W^{L}\right|\left|W^{L-1}\right| \cdots\left|W^{1}\right| \mathbb{1}:=\sum_{i \in I}\left|\prod_{\ell=1}^{L} W^{\ell}\left[i_{\ell}, i_{\ell-1}\right]\right| \tag{4.1}
\end{equation*}
$$

where $I$ is the set $\left[d_{0}\right] \times \cdots \times\left[d_{L}\right]$ with $\left[d_{\ell}\right]:=\left\{0,1, \ldots, d_{\ell}-1\right\},\left|W^{\ell}\right|$ is the matrix obtained by entry-wise application of the absolute value function, the symbol $\mathbb{1}$ denotes an all-ones column vector with dimension inferred by the context, and the $i, j$-th entry of a matrix $W^{\ell}$ is denoted as $W^{\ell}[i, j]$.

One of the key properties of the 1-path-norm, is that it controls the smoothness of the network as it upper bounds its Lipschitz constant. This is known in the case of shallow networks with differentiable activation functions Latorre et al. (2020c). As we now show, this result is more general and holds for networks of arbitrary depth and or networks with ReLU activations.
Theorem 4.1. Let $f_{W}: \mathbb{R}^{d_{0}} \rightarrow \mathbb{R}, f_{W}(x):=W^{L} \sigma\left(W^{L-1} \sigma\left(\cdots \sigma\left(W^{1} x\right) \cdots\right)\right)$ be a network such that the gradient of the activation $\sigma$ is globally bounded between zero and one, i.e., $0 \leq \sigma(x) \leq 1$ or $\sigma(x)=\operatorname{ReLU}(x)$. Choose the $\ell_{\infty}$-norm for the input space and $|\cdot|$ for the output space. The Lipschitz constant of the network, denoted by $L_{W}$ is bounded as follows:

$$
\begin{equation*}
L_{W} \leq P_{1}(W) \leq \prod_{\ell=1}^{L}\left\|W^{\ell}\right\|_{\infty} \tag{4.2}
\end{equation*}
$$

The right-hand-side of Equation (4.2) is usually referred to as the trivial bound based on the product of the norms of each weight matrix (Cisse et al., 2017b). Precisely, Theorem 4.1 states that the 1-path-norm is a better estimator of the $\ell_{\infty}$-Lipschitz constant of the network, than the trivial product bound, in the case of single output. The proof is provided in Section 4.7. This also motivates the use of the 1-path-norm as a regularizer, given that the the Lipschitz constant is related to the generalization and robustness of the network (Sokolić et al., 2017; Anil et al., 2018; Gouk et al., 2021; Pauli et al., 2022).

We can succintly describe the 1-path-norm of a network (4.1) as the $\ell_{1}$-norm of a vector containing the product of weights along each input-output path in the network. The following is the 1-path-norm regularized empirical risk minimization problem on $n$ labeled training samples $\left(x_{i}, y_{i}\right) \in \mathbb{R}^{d_{0}} \times \mathbb{R}^{d_{L}}$, loss function $\mathscr{L}$ and regularization parameter $\lambda \in \mathbb{R}_{\geq 0}$ :

$$
\begin{equation*}
\min _{W} \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}\left(f_{W}\left(x_{i}\right), y_{i}\right)+\lambda P_{1}(W), \quad W:=\left[W_{\ell} \in \mathbb{R}^{d_{\ell} \times d_{\ell-1}}\right]_{\ell=1}^{L} \tag{4.3}
\end{equation*}
$$

When $L \geq 2$, common choices of loss function $\mathscr{L}$, such as the cross-entropy loss, lead to a composite optimization objective in (4.3) that is non-convex and non-smooth with a non-convex non-smooth regularizer due to the presence of absolute values and products. Obviously, such a model cannot be solved globally. Thus, instead of global optimality, we turn to the task of devising algorithms with non-asymptotic rates of convergence to first-order stationarity

[^5]```
Algorithm 4.1 (Stochastic) Proximal Gradient Descent
Input: Initialization \(W_{0}\), batch size \(b\), step size \(\gamma\), regularization parameter \(\lambda \geq 0\), iterations \(T\)
Output: Trained model parameters \(W_{T}\).
    for \(t=1, \ldots, T\) do
        Sample \(i_{1}, \ldots, i_{b} \sim \operatorname{Unif}[n]\)
        \(W_{t+1 / 2} \leftarrow W_{t}-\gamma \nabla_{W} \frac{1}{b} \sum_{j=1}^{b} \mathscr{L}\left(f_{W_{t}}\left(x_{i_{j}}\right), y_{i_{j}}\right)\)
        \(W_{t} \leftarrow \operatorname{prox}_{\gamma \lambda P_{1}}\left(W_{t+1 / 2}\right)\)
    end for
    return \(W_{T}\)
```

via the proximal gradient approach. Let us recall the definition of the proximal mapping, a well-known concept in optimization (Parikh et al., 2014), in the context of the 1-path-norm: (Bauschke and Combettes, 2011, Definition 12.23):

$$
\begin{equation*}
\operatorname{prox}_{\lambda P_{1}(W)} \in \underset{Z}{\operatorname{argmin}} \frac{1}{2}\|Z-W\|_{F}^{2}+\lambda P_{1}(Z) . \tag{4.4}
\end{equation*}
$$

For the type of problem in consideration, eq. (4.4) constitutes a highly challenging task because it involves solving a non-convex and non-smooth problem. Indeed, in the case of the 1-pathnorm, an efficient implementation of the proximal mapping is only known for the case of linear functions and shallow neural networks (Latorre et al., 2020c), which hinders the applicability of 1-path-norm regularization for contemporary deep network architectures used in practice.

Coupled with a stochastic first-order oracle for the smooth elements of the objective function, guarantees for the stochastic proximal gradient (Algorithm 4.1) approach with a non-convex non-smooth regularizer either require structural assumptions (Lipschitz continuity / weak convexity) on the regularizer (Davis and Drusvyatskiy, 2019; Metel and Takeda, 2019), or restrictive conditions on the variance (Xu et al., 2019a; Hallak et al., 2021) via a mini-batch mechanism. Recently some significant advances have been made in this regard Xu et al. (2019a); Davis and Drusvyatskiy (2019); Xu et al. (2019b); Metel and Takeda (2019); Hallak et al. (2021); Tran-Dinh et al. (2021), but the stochastic prox-grad approach (Algorithm 4.1) with the 1-path-norm regularizer is still without any controllable guarantees, even if the prox operator could be computed exactly.

Nevertheless, proximal-gradient type methods are, as of now, the only first-order algorithms with convergence guarantees for composite non-smooth non-convex problems (Bolte et al., 2013). Hence, it is worth exploring their use for 1-path-norm regularization of Deep Neural Networks, and decide if in practice they are a better choice than automatic differentiation. In the following, we use the proximal gradient method as a template (Algorithm 4.1) to perform 1-path-norm regularization. When the prox cannot be obtained exactly, we propose a practical heuristic that approximates its mechanism.

### 4.3 On the hardness of computing the exact Proximal Mapping

To illustrate the difficulty in deriving an exact proximal mapping, we first consider a single term in the sum defining the 1-path-norm (Equation (4.1)). This is equivalent to a fully connected network architecture with one hidden neuron per layer and arbitrary depth. Although limited in practice, this architecture provides two insights: (i) In this simple setup, although tractable, it is already hard to derive an exact solution of the proximal mapping; (ii) Given access an exact proximal mapping oracle, it is possible to optimize Equation (4.3), and similar, in a more efficient manner than employing automatic differentiation (backpropagation). Let $\left(w_{1}, \ldots, w_{L}\right) \in \mathbb{R}^{L}$ be the weights of a unit-width neural network. The proximal mapping of the 1-path-norm of this network is the solution of the following:

$$
\begin{equation*}
\operatorname{prox}_{\lambda P_{1}}(w) \in \underset{z \in \mathbb{R}^{L}}{\operatorname{argmin}} \frac{1}{2}\|z-w\|^{2}+\lambda \prod_{i=1}^{L}\left|z_{i}\right| \tag{4.5}
\end{equation*}
$$

This problem is coercive and bounded from below, meaning that it is has at least one optimal solution. To obtain a solution for eq. (4.5), we introduce Algorithm 4.5 in section 4.8, which uses the first-order optimality conditions of eq. (4.5) to reduce the overall problem into a simple procedure of computing all solutions of a univariate non-linear equation on a compact interval. Solving this one-dimensional optimization problem can be done to any arbitrary sub-optimality using a brute-force search method or more complicated protocols; see Aaid et al. (2017) and references therein.

In section 4.8 we formally show that algorithm 4.5 indeed returns a solution to (4.5), and we explain in detail the inner workings of the algorithm. To show the benefits of having access to the proximal mapping, we consider the toy problem of minimizing the 1-path-norm of a unit-width neural network i.e. $\left|w_{1} \cdot \ldots \cdot w_{L}\right|$, which is equivalent to eq. (4.3) after choosing the loss $\mathscr{L}_{W}=0$ to be identically zero, and setting $\lambda=1$. In this case we know the minimum is 0 , and we observe that automatic differentiation struggles to find a minimizer of this simple problem c.f., Figure 4.1. Unfortunately, the arguments showing that Algorithm 4.5 provides a solution of the proximal operator objective, do not generalize for networks of arbitrary depth and width. The interactions between variables that are present in more than one input-output path in the network lead to an apparently untractable system of equations. For this reason, in the next section we will look at alternatives that instead try to solve the proximal mapping via first-order methods.

### 4.4 Approximate Proximal Gradient Algorithm for the General Case

In this section, we propose three strategies to optimize the objective in eq. (4.3). The first strategy is to simply use automatic differentiation directly on such objective (Algorithm 4.2), together with an off-the-shelf first-order optimization algorithm like SGD or Adam (Kingma and $\mathrm{Ba}, 2014$ ). It has the advantage of being simple, and can be easily extended to any feedforward architecture.


Figure 4.1: Value of the 1-path-norm of a unit-width network across iterations, for different values of depth. Two step sizes and four different optimizers are considered, SGD with either constant or decreasing step-size, Adam (Kingma and Ba, 2014) and the proximal gradient algorithm based on Algorithm 4.5 (prox) in section 4.8. 1000 repeated runs with random initialization were performed for each parameter combination.

The second strategy is to follow the Proximal Gradient Template (Algorithm 4.1), and replace the proximal mapping by a few steps of automatic differentiation applied to the non-smooth objective defining it (4.4). This leads to Algorithm 4.3 (Prox-AD), where the approximation of the prox occurs between section 4.4 and section 4.4. This is a refined approach when compared to the first strategy, but it is still simple to implement. The third strategy we propose relies on the following result:

Lemma 4.1. Let $P$ be a function satisfying $P(W)=P(|W|)$. Its proximal mapping satisfies

$$
\begin{equation*}
\operatorname{prox}_{P}(W)=\operatorname{sign}(W) \odot \operatorname{prox}_{P}^{+}(|W|), \quad \operatorname{prox}_{P}^{+}(X):=\underset{Z \in \mathbb{R}_{+}^{d}}{\operatorname{argmin}} \frac{1}{2}\|X-Z\|_{F}^{2}+P(Z) \tag{4.6}
\end{equation*}
$$

We present a proof of Lemma 4.1 in Section 4.9. Clearly, the 1-path-norm $P_{1}$ satisfies the conditions of Lemma 4.1, as it defined on the absolute values of the weight matrices $\left|W^{L}\right|, \ldots,\left|W^{1}\right|$. This result states that instead of solving prox $\lambda_{\lambda P_{1}}$, we can alternatively solve prox ${ }_{\lambda P_{1}}^{+}$which is a constrained optimization problem over the nonnegative orthant, with a differentiable objective. Indeed, over nonnegative weight matrices, the 1-path-norm is identical to the function $\mathbb{1}^{T} W_{L} \cdots W_{1} \mathbb{1}$ which is infinitely differentiable.

Hence, the key advantage of this third approach over the first two strategies is that it removes all the non-smoothness difficulties and effects originating from the absolute value function. Our third strategy Prox-DIF (Algorithm 4.4) follows the proximal gradient template, but computes an approximation of the proximal mapping of the 1-path-norm using Projected Gradient Descent on the objective that defines prox ${ }_{\lambda P_{1}}^{+}$(c.f., eq. (4.6), Section 4.4 in Algorithm 4.4), and uses the sign of the weights to recover an approximate solution of $\operatorname{prox}_{\lambda P_{1}}$ (c.f., Section 4.4 in

```
Algorithm 4.2 1-path-norm regularization using AD (Path-AD)
Input: Initialization \(W_{0}\), batch size \(b\), step size \(\gamma\), regularization parameter \(\lambda \geq 0\), iterations \(T\)
Output: Trained model parameters \(W_{T}\)
    for \(t=1, \ldots, T\) do
        Sample \(i_{1}, \ldots, i_{b} \sim \operatorname{Unif}[n]\)
        \(W_{t+1} \leftarrow W_{t}-\gamma \nabla_{W}\left[\frac{1}{b} \sum_{j=1}^{b} \mathscr{L}\left(f_{W_{t}}\left(x_{i_{j}}\right), y_{i_{j}}\right)+\lambda P_{1}\left(W_{t}\right)\right]\)
    end for
    return \(W_{T}\)
```

```
Algorithm 4.3 1-path-norm regularization using AD for the Proximal Mapping. (Prox-AD)
Input: Initialization \(W_{0}\), batch size \(b\), step size \(\gamma\), inner step size \(\gamma^{\prime}\) regularization parameter
\(\lambda \geq 0\), iterations \(T\), inner iterations \(T^{\prime}\), skip-prox parameter \(B\).
Output: Trained model parameters \(W_{T}\)
    for \(t=0, \ldots, T-1\) do
        Sample \(i_{1}, \ldots, i_{b} \sim \operatorname{Unif}[n]\)
        \(W_{t+1 / 2} \leftarrow W_{t}-\gamma \nabla_{W} \frac{1}{b} \sum_{j=1}^{b} \mathscr{L}\left(f_{W_{t}}\left(x_{i_{j}}\right), y_{i_{j}}\right)\)
        if \(t=0(\bmod B)\) then
            \(Z_{0}=W_{t+1 / 2}\)
            for \(t^{\prime}=0, \ldots, T^{\prime}-1\) do
                \(Z_{t^{\prime}+1}=Z_{t}-\gamma^{\prime} \nabla_{Z}\left[\frac{1}{2}\left\|W_{t+1 / 2}-Z_{t^{\prime}}\right\|_{2}^{2}+\lambda \gamma P_{1}\left(Z_{t^{\prime}}\right)\right]\)
            end for
            \(W_{t+1}=Z_{T^{\prime}}\)
        else
            \(W_{t+1}=W_{t+1 / 2}\)
        end if
    end for
    return \(W_{T}\)
```


## Algorithm 4.4), as Lemma 4.1 indicates.

Efficiently computing the 1-path-norm for arbitrary architectures. All the algorithms for 1-path-norm regularization (Algorithms 4.2 to 4.4) require evaluating the 1-path-norm of the network to obtain its gradient through automatic differentiation. The defining expression given by eq. (4.1) is stated for fully connected feed-forward neural networks via the weight matrices $W^{L}, \ldots, W^{1}$. Nevertheless, we can efficiently evaluate this expression for convolutional networks or in general feed-forward architectures. Note that the computation of the 1-path-norm is equivalent to the following process: (1) replace the activations in the network with the identity $\sigma(x)=x$; (2) replace the weights by their absolute values; (3) remove biases; (4) compute the forward pass of the resulting (linear) network on the vector of all-ones $\mathbb{1}$; and (5) sum the outputs. For example, in the case of CNNs, this avoids transforming the kernel matrices into their equivalent representation as a huge doubly circulant matrix (c.f., Sedghi et al. (2019)), which would be inefficient memory-wise.

Sparsification properties of Prox-DIF. The main difference between Prox-DIF (algorithm 4.4)

```
Algorithm 4.4 Differentiable Proximal training of 1-path-norm regularized NNs (Prox-DIF )
Input: Initialization \(W_{0}\), batch size \(b\), step size \(\gamma\), inner step size \(\gamma^{\prime}\) regularization parameter
\(\lambda \geq 0\), iterations \(T\), inner iterations \(T^{\prime}\), skip-prox parameter \(B\).
Output: Trained model parameters \(W_{T}\)
    for \(t=0, \ldots, T-1\) do
        Sample \(i_{1}, \ldots, i_{b} \sim \operatorname{Unif}[n]\)
        \(W_{t+1 / 2} \leftarrow W_{t}-\gamma \nabla_{W} \frac{1}{b} \sum_{j=1}^{b} \mathscr{L}\left(f_{W_{t}}\left(x_{i_{j}}\right), y_{i_{j}}\right)\)
        if \(t=0(\bmod B)\) then
            \(Z_{0}=\left|W_{t+1 / 2}\right|\)
            for \(t^{\prime}=0, \ldots, T^{\prime}-1\) do
                \(Z_{t^{\prime}+1 / 2}=Z_{t}-\gamma^{\prime} \nabla_{Z}\left[\frac{1}{2}\left\|\left|W_{t+1 / 2}\right|-Z_{t^{\prime}}\right\|_{2}^{2}+\lambda \gamma P_{1}\left(Z_{t^{\prime}}\right)\right]\)
            \(Z_{t^{\prime}+1}=\max \left(0, Z_{t^{\prime}+1 / 2}\right)\)
            end for
            \(W_{t+1}=\operatorname{sign}\left(W_{t+1 / 2}\right) \odot Z_{T^{\prime}}\)
        else
            \(W_{t+1}=W_{t+1 / 2}\)
        end if
    end for
    return \(W_{T}\)
```

and prox-AD/path-AD (algorithms 4.2 and 4.3) is the presence of the projection operator section 4.4. This clamping operator has the benefit of inducing real sparsity, as it sets weights to exactly zero whenever they become negative during the inner loop corresponding to the approximate solution of prox ${ }_{\lambda P_{1}}^{+}$.

### 4.5 Related Work

The 1-path-norm is a positively homogeneous regularizer, a setting that has been studied by Haeffele and Vidal (2015); Haeffele et al. (2014); Haeffele and Vidal (2020). Haeffele and Vidal (2020); Haeffele et al. (2014) study a setting with objective function of the form:

$$
\begin{equation*}
\min _{U, V} \ell\left(Y, U V^{T}\right)+\lambda \Theta(U, V) \tag{4.7}
\end{equation*}
$$

where $Y$ is some data and $U, V$ are the model parameters, usually corresponding to a matrix factorization task $X=U V^{T}$. The fundamental difference between our work and Haeffele et al. (2014); Haeffele and Vidal (2020) is that eq. (4.7) results in a multi-block multi-convex objective that allows convergence of the alternating optimization method from Xu and Yin (2013). In contrast, due to the presence of non-linear activation functions in the Neural Network, our optimization objective no longer follows the template eq. (4.7) (in the shallow network case). Hence, the alternating optimization scheme of Xu and Yin (2013), which would have eliminated the need for the full proximal mapping of the 1-path-norm, no longer enjoys convergence guarantees to a critical point. In contrast, the full proximal map in Latorre et al.
(2020c) and in this work, allows the implementation of the non-convex proximal gradient scheme of Bolte et al. (2013), which ensures convergence to a critical point. Indeed, the setting most similar to our work is that of Haeffele and Vidal (2015) where the objective function is of the form

$$
\begin{equation*}
\min _{W_{1}, \ldots, W_{d}} \ell\left(Y, \Phi\left(W_{1}, \ldots, W_{d}\right)\right)+\lambda \Theta\left(W_{1}, \ldots, W_{d}\right) \tag{4.8}
\end{equation*}
$$

In this formulation $\Phi$ is a non-multi-linear map that destroys the multi-block multi-convex structure of the objective of eq. (4.7). Indeed, in this case Haeffele and Vidal (2015) do not provide an algorithm with guarantees of convergence to a critical point. A different work providing guarantees for block-coordinate methods for non-convex functions (Razaviyayn et al., 2013) requires access to tight upper bounds (Assumption 1 therein) that are hard to obtain for general non-convex functions, or at least, not readily available in our setting.

### 4.6 Experiments

Proximal Mapping approximation: Algorithm 4.3 vs. Algorithm 4.4. We compare the performance of the two proposed algorithms that approximate the proximal mapping of the 1-path-norm. This corresponds to section 4.4 in algorithm 4.3, and section 4.4 in algorithm 4.4. We sample fully-connected networks and CNNs of different width and depth, and for different values of the regularization parameter $\lambda$ we plot how the value of the proximal objective evolves across iterations. In Figure 4.2, we observe that for larger values of the regularization parameter $\lambda$, Prox-DIF (Algorithm 4.4) has an advantage over the baseline Prox-AD (Algorithm 4.3), which uses automatic differentiation directly on the non-smooth objective eq. (4.4). Prox-DIF reaches lower values of the objective much faster. The differentiability of the Prox-DIF objective allows the use of momentum more effectively.

In the case of shallow networks, where the optimum is known (cf. (Latorre et al., 2020c, Algorithm 2)), Prox-DIF almost reaches such value, suggesting a possible unknown theoretical proximity relation with the optimum. For deeper networks where achieving the optimal value is intractable, Prox-DIF still shows a noticeable advantage. Nevertheless, as observed in the bottom row of figure 4.2, for really small values of $\lambda$, the difference between the two methods is pretty small. This suggests that for problems where only a small amount of regularization is needed, there might be no advantage for Prox-DIF .

Impact on the Generalization Error. We train fully-connected networks and CNNs on the FashionMNIST and CIFAR10 benchmark datasets. In every experiment, we perform grid search and select the best parameters for each regularization method based on validation accuracy. All models are trained with early stopping using accuracy on the validation set and then evaluated on a held-out test set. In the case of Prox-AD (Algorithm 4.3) and Prox-DIF (Algorithm 4.4), we tune the inner loop parameters on randomly initialized network weights, so as to minimize the objective of the proximal operator (4.4). Exact details on the training parameters are available in Section 4.10. With these parameters at hand, we train the networks on the full training set, and plot the validation accuracy as a function of the training epoch


Figure 4.2: For randomly initialized $W$ (network parameters), we plot the objective value of the proximal mapping $0.5\|W-Z\|^{2}+\lambda P_{1}(Z)$ as a function of iteration, using the inner-most loop in the algorithms Prox-AD (Algorithm 4.3) and Prox-DIF (Algorithm 4.4). Left column correspond to shallow networks (MLP1), for which we plot the theoretical optimal value obtained with the exact proximal algorithm from Latorre et al. (2020c). Middle column corresponds to fully-connected networks with 3 hidden layers (MLP3). Right column corresponds to CNNs with 4 convolutional layers and 2 fully-connected layers (CNN6). Top row corresponds to high regularization parameter $\lambda=1 \mathrm{e}-2$. Bottom row corresponds to low regularization parameter $\lambda=1 \mathrm{e}-6$. All values are averaged over 5 random initializations, with standard error shown as shaded area.


Figure 4.3: Validation accuracy as a function of training epoch, for different training algorithms. First and second panels: fully-connected architectures with 2 and 3 hidden layers (MLP2 and MLP3) on the FashionMNIST dataset. Third panel: fully-connected architecture with 3 hidden layers (MLP3) on the CIFAR10 dataset. Fourth panel: CNN with 6 hidden layers (CNN6) on the CIFAR10 dataset. All values are averaged over 5 independent runs, with standard error shown as shaded area.

Table 4.1: Comparison of test set accuracy (\%) for networks trained with different regularization.

| Model | Dataset | Regularization |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | None | L2 | Path-AD | Prox-AD | Prox-DIF |
| MLP2 | FMNIST | $85.6 \pm 1.2$ | $85.4 \pm 1.2$ | $\mathbf{8 9 . 0} \pm 0.1$ | $88.6 \pm 0.1$ | $88.6 \pm 0.2$ |
| MLP3 | FMNIST | $84.4 \pm 1.1$ | $85.9 \pm 0.3$ | $88.86 \pm 0.02$ | $\mathbf{8 8 . 9} \pm 0.1$ | $\mathbf{8 8 . 9} \pm 0.1$ |
| MLP3 | CIFAR10 | $53.5 \pm 0.1$ | $53.8 \pm 0.2$ | $54.7 \pm 0.1$ | $\mathbf{5 4 . 9} \pm 0.1$ | $54.2 \pm 0.2$ |
| CNN6 | CIFAR10 | $66.6 \pm 0.6$ | $\mathbf{7 4 . 6} \pm 0.2$ | $73.4 \pm 0.3$ | $73.7 \pm 0.6$ | $73.4 \pm 0.3$ |

in Figure 4.3. Finally, the test accuracy for each model, dataset and regularization method is reported in Table 4.1. We observe that for fully connected networks, the 1-path-norm is a significantly better choice than no-regularization or weight-decay (L2 regularization).

Regarding the optimization methods for 1-path-norm regularization, we do not observe major differences other than Prox-DIF achieving slightly less accuracy, or Prox-AD achieving a slightly more accurate MLP in CIFAR10. This is probably due to the fact that the values between $\lambda=10^{-6}$ and $\lambda=10^{-3}$, which we find lead to higher accuracy, are really small. This is a common feat in computer vision tasks where only small amounts of regularization is needed. Moreover, given that the error on the test set is a stochastic estimate, better optimization of the training loss does not necessarily lead to gains in test accuracy (Bottou and Bousquet, 2007). In such setting, the simplest approach of just using automatic differentiation on the 1-path-norm objective works well, despite lacking theory. This has the advantage that AD is the most common approach among practitioners. Prox-DIF might work better in settings where a high value of regularization strength is required to improve performance, like data with low signal-to-noise ratio. We synthetically test this setting later on.

On the other hand, for CNNs, 1-path-norm regularization struggles to improve over weight decay. This can be attributed to the fact that CNNs are highly-sparse fully connected networks


Figure 4.4: Absolute difference in test accuracy with regards to the unregularized model as a function of percentage of noisy labels in the training set (top), or as a function of image noise level (bottom), for different training algorithms. First and second columns: fully-connected architectures with 2 and 3 hidden layers (MLP2 and MLP3) on the FashionMNIST dataset. Third column: fully-connected architecture with 3 hidden layers (MLP3) on the CIFAR10 dataset. Fourth column: CNN with 6 hidden layers (CNN6) on the CIFAR10 dataset. All values are averaged over 5 independent runs, with standard error shown as shaded area.
(Sedghi et al., 2019), given the parameter sharing structure of the convolutional layers. Indeed, due to the reduced parameter count in CNNs, the number of different paths from input to output is small by design. Recall that the 1-path-norm controls complexity by reducing the number of paths and hence, it can be expected to perform poorly for CNNs. As such, 1-pathnorm regularization might be more suitable to architectures like MLP-mixer (Tolstikhin et al., 2021).

Robustness to Noise. We study the effect of the regularization on the model performance in the case of noisy data or noisy labels. In the former case, a uniform random noise sampled from the $\ell_{\infty}$-ball of varying radius (referred to as the noise level) is added to the images at inference time. In the latter case, the learning with noisy labels task (Song et al., 2022) consists in training a model on a dataset where a fraction of the labels are corrupted, then evaluating its accuracy on the correctly labelled test set. In the noisy labels regime (figure 4.4-top), 1-path-norm regularization improves robustness in the FashionMNIST dataset. In contrast, in CIFAR10 1-path-norm does not induce noisy-labels robustness for MLPs, while it does for CNNs, which is not consistent. For CNNs, however, L2 provides better robustness to noisy labels when the noise is small, and 1-path-norm is better for higher values of the noise. Nevertheless, we stress that 1-path-norm is not designed to induce robustness to noisy labels, but it is always useful to identify whether it has unexpected benefits. In this case, the effect is not always positive but rather it depends on the architecture and dataset. Similarly as in the previous experiment, we do not see a big difference between the different optimization algorithms used for 1-path-norm regularization.

In the noisy data regime the 1-path-norm regularization improves all fully-connected networks and achieves significantly higher accuracy than weight-decay or no regularization on the majority of noise levels, for both datasets. In MLPs we observe that the proximal methods Prox-AD and Prox-DIF are better than Path-AD consistently, in particular for CIFAR10 it is crucial to use them to achieve higher robustness, with Prox-DIF achieving higher robustness over Prox-AD for large levels of the noise, in particular visible for FashionMNIST-MLP2 and CIFAR10-MLP3. This is consistent with the sparsity-inducing property of Prox-DIF (section 4.4). For CNNs 1-path-norm again fails to provide benefits, probably due to the already highly sparse structure of the CNNs. We stress that the goal is not to provide state-of-the-art numbers for robustness, rather, regularization can be used together with Adversarial Training (Madry et al., 2018b) to enhance robustness, as is done for example in (Zhu et al., 2022)

### 4.7 Appendix: Proof of theorem 4.1

We prove theorem 4.1 by first showing that the 1-path-norm is upper bounded by the product-of-norms lemma 4.2 and then showing that the 1-path-norm upper bounds the Lipschtiz constant of the network lemma 4.3.
Lemma 4.2. Let $W=\left[W^{1}, \ldots, W^{L}\right]$ be the weight matrices of a fully-connected network with a single output. Then

$$
\begin{equation*}
P_{1}(W) \leq \Pi_{\ell=1}^{L}\left\|W^{\ell}\right\|_{\infty} \tag{4.9}
\end{equation*}
$$

Proof. We proceed by induction. The base case corresponds to shallow networks and is already known as Latorre et al. (2020c, Theorem 1.). Assume that the result is true for $L$ layers. Let $W=\left[W^{1}, \ldots, W^{L+1}\right]$ where $W^{L+1} \in \mathbb{R}^{1, d_{L}}$ is a matrix with a single row. In this case we have

$$
\begin{align*}
P_{1}(W) & =\mathbb{1}^{T}\left|W^{L+1}\right|\left|W^{L}\right|\left|W^{L-1}\right| \ldots\left|W^{1}\right| \mathbb{1} \\
& =\sum_{i=1}^{d_{L}}\left|W_{i}^{L+1}\right| \| W^{L}[i,:]| | W^{L-1}|\ldots| W^{1} \mid \mathbb{1} \\
& =\sum_{i=1}^{d_{L}}\left|W_{i}^{L+1}\right| P_{1}\left(W^{L}[i,:], \ldots, W^{1}\right) \\
& \leq \sum_{i=1}^{d_{L}}\left|W^{L+1}[1, i]\right| \underbrace{\Pi_{\ell}}_{\leq\left\|W^{L}[i,:]\right\|_{1}} \Pi_{\ell=1}^{L-1}\left\|W^{\ell}\right\|_{\infty}  \tag{4.10}\\
& \leq \sum_{i=1}^{d_{L}}\left|W^{L+1}[1, i]\right|\left\|W^{L}\right\|_{\infty} \Pi_{\ell=1}^{L-1}\left\|W^{\ell}\right\|_{\infty} \\
& =\left\|W^{L+1}[1,:]\right\|_{1}\left\|W^{L}\right\|_{\infty} \Pi_{\ell=1}^{L-1}\left\|W^{\ell}\right\|_{\infty} \\
& =\Pi_{\ell=1}^{L+1}\left\|W^{\ell}\right\|_{\infty}
\end{align*}
$$

where the second equality is due to the definition of the path-norm, the first inequality is due to the induction hypothesis and the second inequality is due to the definition of the $\infty$-operator norm $\|W\|_{\infty}=\max _{i=1}^{d}\|W[i,:]\|_{1}$ is the maximum $\ell_{1}$-norm of the rows. Note that
in the last two lines $\left\|W^{L+1}[1,:]\right\|_{1}$ denotes the $\ell_{1}$-norm as a vector while $\left\|W^{L+1}\right\|_{\infty}$ denotes the $\infty$-operator norm of $W^{L+1}$ as a matrix with a single row.

Lemma 4.3. Let $f_{W}: \mathbb{R}^{d_{0}} \rightarrow \mathbb{R}, f_{W}(x)=W^{L} \sigma\left(W^{L-1} \sigma\left(\ldots \sigma\left(W^{1} x\right) \ldots\right)\right.$ be a neural network where $W^{L} \in \mathbb{R}^{d_{L} \times 1}$, i.e., a single-output network. In the case of a differentiable activation function suppose that that the derivative of the activation is globally bounded between zero and one. For ReLU, we have that the subgradient satisfiest $0 \leq \sigma^{\prime}(x) \leq 1$. Its Lipschitz constant, denoted as $L_{W}$, with respect to the $\ell_{\infty}$ norm (for the input space) and absolute-value (for the output space) satisfies the inequality $L_{W} \leq P_{1}(W)$.

Proof. In the case of differentiable activation functions, we have that the $\ell_{\infty}$-Lipschitz constant of $f_{W}$ is equal to the supremum of the $\ell_{1}$-norm of its gradient, over its domain (c.f., Latorre et al. (2020c, Theorem 1)). In the case of ReLU activations, Chen et al. (2020b, Lemma 1) ensures that the $\ell_{\infty}$-Lipschitz constant is still equal to the supremum of the $\ell_{1}$-norm of the "gradient" computed with the backpropagation algorithm, which applies the chain-rule even in the presence of the non-differentiable ReLU activation. Despite not being the true gradient, we will denote this element as $\nabla f_{W}(x)$. We use the notation $D f=\nabla^{\top} f$ for the Jacobian (transpose of gradient).

Denote by $f_{W}^{\ell}(x)$ the value of the $\ell$-th hidden layer (pre-activation) in the forward pass over the network, that is:

$$
\begin{equation*}
f_{W}^{\ell}(x)=W^{\ell} \sigma\left(W^{\ell-1} \sigma\left(\ldots \sigma\left(W_{1} x\right) \ldots\right)\right) \tag{4.11}
\end{equation*}
$$

We now upper bound the Lipschitz constant as follows, using Latorre et al. (2020c, Theorem 1) and the chain rule:

$$
\begin{align*}
L_{W} & =\sup _{x}\left\|\nabla f_{W}(x)\right\|_{1}  \tag{4.12}\\
& =\sup _{x} \sup _{\|t\|_{\infty} \leq 1} D f_{W}(x) t  \tag{4.13}\\
& =\sup _{x} \sup _{\|t\|_{\infty} \leq 1} W^{L} \operatorname{Diag}\left(\sigma^{\prime}\left(f_{W}^{L-1}(x)\right)\right) W^{L-1} \operatorname{Diag}\left(\sigma^{\prime}\left(f_{W}^{L-2}(x)\right)\right) \ldots W^{1} t \\
& \leq \sup _{0 \leq s^{s} \leq 1\|t\|_{\infty} \leq 1} W^{L} \operatorname{Diag}\left(s^{L}\right) W^{L-1} \operatorname{Diag}\left(s^{L-1}\right) \ldots W^{2} \operatorname{Diag}\left(s^{2}\right) W^{1} t  \tag{4.14}\\
& \leq \sup _{0 \leq s^{s} \leq 1\|t\|_{\infty} \leq 1} \widehat{W}^{L} \widehat{W}^{L-1} \widehat{W}^{L-2} \ldots \widehat{W}^{2} W^{1} t \tag{4.15}
\end{align*}
$$

where $\widehat{W^{\ell}}=W^{\ell} \operatorname{Diag}\left(s^{\ell}\right)$. We see tthe right hand side of eq. (4.12) is equivalent to evaluating a fully connected linear neural network on the input $t$, and then taking the supremum over all possible values of $t$ i.e., $-1 \leq t \leq 1$ and $s^{\ell}=\sigma^{\prime}\left(f_{W}^{L-1}(x)\right)$, so $0 \leq s^{\ell} \leq 1$ by our assumption on the gradient of the activation (or subgradient in the case of ReLU). The output of a linear neural network precisely corresponds to summing the product of weights over all input-output paths.

```
Algorithm 4.5 Single path proximal operator
Input: Weights along the path \(w=\left(w_{1}, \ldots, w_{L}\right) \in \mathbb{R}^{L}\), regularization parameter \(\lambda \geq 0\)
Output: \(\operatorname{prox}_{\lambda g}(w)\)
    \(: \sigma \leftarrow \operatorname{argsort}(|w|) \quad \triangleright\) decreasing order
    \(\tilde{w}_{i} \leftarrow|w|_{\sigma(i)} \quad \triangleright\) sort
    Let \(f_{\lambda}(\tilde{w}, \beta):=2^{2-L} \lambda \prod_{i=2}^{L-1}\left(\tilde{w}_{i}+\sqrt{4 \beta^{2}+\tilde{w}_{i}^{2}-\tilde{w}_{1}^{2}}\right)\)
    Find the set \(B\) of values \(\beta\) satisfying:
                    \(\tilde{w}_{1}=2 \beta+2 \tilde{w}_{L} f_{\lambda}(\tilde{w}, \beta)-\left(2 \beta+\tilde{w}_{1}\right) f_{\lambda}^{2}(\tilde{w}, \beta)\)
subject to \(\frac{1}{2} \sqrt{\tilde{w}_{1}^{2}-\tilde{w}_{L}^{2}} \leq \beta \leq \frac{1}{2} \tilde{w}_{1}\).
5: For each value of \(\beta\), compute:
\(\triangleright z_{1}^{\beta} \leftarrow \beta+\frac{\tilde{w}_{1}}{2}\)
\(\triangleright z_{i}^{\beta} \leftarrow \frac{1}{2}\left(\tilde{w}_{i}+\sqrt{\tilde{w}_{i}^{2}-4 z_{1}^{\beta}\left(\tilde{w}_{1}-z_{1}^{\beta}\right)}\right), i=2, \ldots, L\)
\(\triangleright z^{\beta} \leftarrow \operatorname{sign}(w) \odot \sigma^{-1}\left(z^{\beta}\right)\), where \(\odot\) stands for element-wise multiplication
6: \(z_{i}^{0}=0\) for \(i=\underset{k}{\operatorname{argmin}}\left|w_{k}\right|, z_{j}^{0}=w_{j}\) for \(j \neq i\)
7: return \(z \in\left\{z^{\beta}: \beta \in B\right\} \cup\left\{z^{0}\right\}\) achieving the smallest objective value in (4.5)
```

We arrive at the following:

$$
\begin{align*}
& L_{W} \leq \sup _{0 \leq s^{\ell} \leq 1\|t\|_{\infty} \leq 1} \sup _{\left(i_{0}, \ldots, i_{L}\right)} W^{1}\left[i_{1}, i_{0}\right] t_{i_{0}} \Pi_{\ell=2}^{L} \widehat{W}^{\ell}\left[i_{\ell}, i_{\ell-1}\right]  \tag{4.17}\\
& L_{W} \leq \sup _{0 \leq s^{\ell} \leq 1} \sup _{\|t\|_{\infty} \leq 1} \sum_{\left(i_{0}, \ldots, i_{L}\right)} W^{1}\left[i_{1}, i_{0}\right] t_{i_{0}} \Pi_{\ell=2}^{L} W^{\ell}\left[i_{\ell}, i_{\ell-1}\right] s_{i_{\ell-1}}^{\ell}  \tag{4.18}\\
& L_{W} \leq \sum_{\left(i_{0}, \ldots, i_{L}\right)} \Pi_{\ell=1}^{L}\left|W^{\ell}\left[i_{\ell}, i_{\ell-1}\right]\right|=P_{1}(W) \tag{4.19}
\end{align*}
$$

### 4.8 Appendix: Proof of lemma 4.4

We will now show the following:
Lemma 4.4. Let $z^{\beta^{*}}$ be the solution returned by Algorithm 4.5. Then, $z^{\beta^{*}}$ is a solution of eq. (4.5).

Recall that we want to solve the following problem:

$$
\begin{equation*}
\underset{z \in \mathbb{R}^{L}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{L}\left(z_{i}-w_{i}\right)^{2}+\lambda\left|z_{1} \cdot z_{2} \cdots z_{L}\right|, \tag{4.21}
\end{equation*}
$$

Since our domain is the entire space, and the proximal gradient is a solution to a symmetric
problem, we can just ignore the absolute value and write the problem (4.21) as

$$
\begin{equation*}
\underset{z \in \mathbb{R}_{+}^{m}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{m}\left(z_{i}-\left|w_{i}\right|\right)^{2}+\lambda z_{1} \cdots z_{L} \tag{4.22}
\end{equation*}
$$

The problem in (4.22) is coercive and bounded from below, meaning that it has an optimal solution.

Lemma 4.5. There exist an optimal solution for eq. (4.22).

Additionally, since the regularizer in eq. (4.22) is symmetric, the order of the elements with respect to magnitude is maintained by the optimal solution.

Lemma 4.6. Suppose that $\left|w_{1}\right| \geq\left|w_{2}\right| \geq \cdots \geq\left|w_{L}\right|$, and let $z^{*}$ be an optimal solution of eq. (4.21). Then

$$
\begin{equation*}
z_{1}^{*} \geq z_{2}^{*} \geq \cdots \geq z_{L}^{*} \tag{4.23}
\end{equation*}
$$

Proof. Assume the contrary, that there exist an optimal solution of (4.22) such that (4.22) does not hold. Without loss of generality, suppose that $z_{1}^{*}<z_{2}^{*}$, and consider the solution $\tilde{z}$ given by

$$
\tilde{z}_{i}= \begin{cases}z_{i}^{*}, & i=3,4, \ldots, L \\ z_{1}^{*}, & i=2 \\ z_{2}^{*}, & i=1\end{cases}
$$

Then by the optimality of $z^{*},\left|w_{1}\right| \geq\left|w_{2}\right|$, and our assumption that $z_{1}^{*}<z_{2}^{*}$, we obtain that

$$
\begin{aligned}
0 & \geq \frac{1}{2} \sum_{i=1}^{l}\left(z_{i}^{*}-\left|w_{i}\right|\right)^{2}+\lambda z_{1}^{*} \cdot z_{2}^{*} \cdots z_{L}^{*}-\frac{1}{2} \sum_{i=1}^{l}\left(\tilde{z}_{i}-\left|w_{i}\right|\right)^{2}-\lambda \tilde{z}_{1} \cdot \tilde{z}_{2} \cdots \tilde{z}_{L} \\
& =\frac{1}{2}\left[\left(z_{1}^{*}-\left|w_{1}\right|\right)^{2}+\left(z_{2}^{*}-\left|w_{2}\right|\right)^{2}-\left(\tilde{z}_{1}-\left|w_{1}\right|\right)^{2}-\left(\tilde{z}_{2}-\left|w_{2}\right|\right)^{2}\right] \\
& =\frac{1}{2}\left[\left(z_{1}^{*}-\left|w_{1}\right|\right)^{2}+\left(z_{2}^{*}-\left|w_{2}\right|\right)^{2}-\left(z_{2}^{*}-\left|w_{1}\right|\right)^{2}-\left(z_{1}^{*}-\left|w_{2}\right|\right)^{2}\right] \\
& =\left(\left|w_{1}\right|-\left|w_{2}\right|\right)\left(z_{2}^{*}-z_{1}^{*}\right)>0
\end{aligned}
$$

which is a contradiction.

It is not hard to derive from (4.22) that if one weight is set to zero then the regularizer has no influence on the solution, and thus the solution is trivial; from Lemma 4.6 this must be the element corresponding to the smallest $w_{i}$.

Lemma 4.7. If there exists an optimal solution $z^{*}$ of (4.21) with $z_{i}^{*}=0$ for some $i$. Then $\left|w_{i}\right|=\min _{j}\left|w_{j}\right|$, and the optimal solution satisfies that $z_{j}^{*}=\left|w_{j}\right|$ for all $j \neq i$.

To focus on the main difficulty in finding a solution to (4.21), let us make the following conventions until the end of this section:
A. Order: It holds that $\left|w_{1}\right| \geq\left|w_{2}\right| \geq \cdots \geq\left|w_{L}\right|$;
B. Nontrivial Solution: There exists a nontrivial solution $z^{*}$ of (4.21), that is, satisfying that $z_{i}^{*}>0$ for all $i=1,2, \ldots, L$.

We will now solve (4.22) under the assumptions above and we will casually regard any solution as positive, i.e., $z>0$. Once possible solutions satisfying $z>0$ are found, they must be compared to the trivial possible solution of Lemma 4.7 in terms of the objective value in (4.21).

Lemma 4.8 (First-order optimality conditions). Let $z^{*}$ be an optimal solution of (4.21). Then:

$$
\begin{equation*}
z_{i}^{*}-\left|w_{i}\right|+\lambda \frac{z_{1}^{*} \cdot z_{2}^{*} \cdots z_{L}^{*}}{z_{i}^{*}}=0, \quad i=1,2, \ldots, L \tag{4.24}
\end{equation*}
$$

Proof. This set of equations is obtained by setting the gradient of the objective of (4.22) to 0 .

The optimality conditions imply the following useful result.
Corollary 4.1. Let $z^{*}$ be an optimal solution of (4.21). Then:

$$
\begin{equation*}
z_{i}^{*}\left(\left|w_{i}\right|-z_{i}^{*}\right)=z_{j}^{*}\left(\left|w_{j}\right|-z_{j}^{*}\right), \quad \forall i, j=1,2, \ldots, L . \tag{4.25}
\end{equation*}
$$

Corollary 4.1 suggests that the elements of the optimal solution $z_{1}^{*}, z_{2}^{*}, \ldots, z_{L}^{*}$ must satisfy that their values yield value equality of all their corresponding parabolas. Analyzing this graphically yields a very interesting phenomenon. Observe the illustration in Figure 4.5 which depicts two parabolas with different $\left|w_{i}\right|$.


Figure 4.5: Red lines depict parabolas for $\left|w_{2}\right|=1,\left|w_{1}\right|=2$, blue line depicts maximal value for the two parabolas to have the same value.

For the two parabolas in Figure 4.5 to have the same value, $z_{1}$ must satisfy that $z_{1}\left(\left|w_{1}\right|-z_{1}\right) \leq$ $\frac{\left|w_{2}\right|^{2}}{4}$, which means intuitively that its possible values are close to the extremes of its parabola.

Moreover, since Lemma 4.6 implies that $z_{1} \geq z_{2}, z_{1}$ must be on the right side of the parabola, i.e., $z_{1}>\frac{\left|w_{1}\right|}{2}$. By the same argument, we can deduce that $z_{i} \geq \frac{\left|w_{i}\right|}{2}$ for all $i=1,2, \ldots, L-1$; note that $z_{L}$ is not constrained by this argument.

We now formulate and prove all of this discussion properly.
Lemma 4.9 (Properties of solutions for (4.21)). Let $z^{*}$ be an optimal solution of (4.21) (such that $z^{*}>0$ ). Then:

1. For any $i=1,2, \ldots, L$ it holds that

$$
z_{i}^{*}\left(\left|w_{i}\right|-z_{i}^{*}\right) \leq \frac{\left|w_{L}\right|^{2}}{4}
$$

2. For any $i=1,2, \ldots, L-1$, the element $z_{i}^{*}$ satisfies that

$$
\begin{equation*}
\frac{1}{2}\left|w_{i}\right|+\frac{1}{2} \sqrt{\left|w_{i}\right|^{2}-\left|w_{L}\right|^{2}} \leq z_{i}^{*} \leq\left|w_{i}\right| \tag{4.26}
\end{equation*}
$$

3. For any $i=1,2, \ldots, L-1$, the element $z_{i}^{*}$ satisfies that

$$
z_{i}^{*}=\frac{1}{2}\left(\left|w_{i}\right|+\sqrt{\left|w_{i}\right|^{2}-4 z_{1}^{*}\left(\left|w_{1}\right|-z_{1}^{*}\right)}\right)
$$

and

$$
z_{L}^{*}=\left|w_{L}\right|-\lambda z_{1}^{*} \cdot z_{2}^{*} \cdots z_{L-1}^{*} .
$$

4. It holds that

$$
\begin{gather*}
\left.\left|w_{1}\right|=z_{1}^{*}+\frac{\lambda}{2^{l-2}}\left(\left|w_{L}\right|-\frac{\lambda}{2^{l-2}} z_{1}^{*} \prod_{i=2}^{L-1}\left(\left|w_{i}\right|+\sqrt{\left|w_{i}\right|^{2}-4 z_{1}^{*}\left(\left|w_{1}\right|-z_{1}^{*}\right.}\right)\right)\right)  \tag{4.27}\\
\cdot \prod_{i=2}^{L-1}\left(\left|w_{i}\right|+\sqrt{\left|w_{i}\right|^{2}-4 z_{1}^{*}\left(\left|w_{1}\right|-z_{1}^{*}\right)}\right)
\end{gather*}
$$

Proof. 1. This claim immediately follows from Corollary 4.1, as for any $i=1,2, \ldots, L$, it holds that

$$
z_{i}^{*}\left(\left|w_{i}\right|-z_{i}^{*}\right)=z_{L}^{*}\left(\left|w_{L}\right|-z_{L}^{*}\right) \leq \frac{\left|w_{L}\right|^{2}}{4}
$$

2. First, it is obvious that $z_{i}^{*} \leq\left|w_{i}\right|$ for any $i=1,2, \ldots, L$. We now proceed to prove the lower bound on $z_{i}^{*}$.
By the first part, $z_{i}^{*}\left(\left|w_{i}\right|-z_{i}^{*}\right) \leq \frac{\left|w_{L}\right|^{2}}{4}$ for all $i=1,2, \ldots, L$, which implies that:

$$
z_{i}^{*} \leq \frac{1}{2}\left(\left|w_{i}\right|-\sqrt{\left|w_{i}\right|^{2}-\left|w_{L}\right|^{2}}\right) \text { or } z_{i}^{*} \geq \frac{1}{2}\left(\left|w_{i}\right|+\sqrt{\left|w_{i}\right|^{2}-\left|w_{L}\right|^{2}}\right)
$$

Suppose that $z_{i}^{*}<\frac{1}{2}\left|w_{i}\right|$. Then from the properties of the parabola (illustrated by Figure 4.5) we can derive that in order for

$$
z_{i}^{*}\left(\left|w_{i}\right|-z_{i}^{*}\right)=z_{L}^{*}\left(\left|w_{L}\right|-z_{L}^{*}\right)
$$

we must have that $z_{i}^{*}<z_{L}^{*}$, which by Lemma 4.6 is a contradiction to our assumption that $\left|w_{i}\right| \geq\left|w_{L}\right|$.
3. The equality for $i=1,2, \ldots, L-1$ follows from Corollary 4.1 (by finding roots of the order two polynomial) together with the second part (that excludes one of the two solutions). The equality for $z_{L}^{*}$ trivially follows from Lemma 4.8 .
4. By Lemma 4.8 , we have that

$$
\left|w_{1}\right|=z_{1}^{*}+\lambda z_{2}^{*} \cdot z_{3}^{*} \cdots z_{L}^{*}
$$

The desired then follows by plugging the formulas in part 3 to the latter.

Lemma 4.9 suggests that candidates for an optimal solution can be found by finding feasible roots in the nonlinear univariate function (4.27) over a specific bounded interval (4.26). This is considered an easy task in optimization (there are some well-known search procedures in the literature). Algorithm 4.5 is thus obtained by making the change of variable $\beta=z_{1}^{*}-\frac{\left|w_{1}\right|}{2}$.

Complexity of Algorithm 4.5: The first step in the procedure involves sorting the weights along the path, and has complexity $O(L \log L)$. Although we do not theoretically bound the number of solutions of equation (4.27), we empirically observe that this equation has a finite number of solutions (at most 4 in practice), independently of the path's depth. Since evaluating the RHS of equation (4.20) takes time $O(L)$, and since grid search can be trivially parallelized on GPU, applying grid search over a bounded domain has the same complexity $O(L)$. Overall, we thus conclude that the total complexity of Algorithm 4.5 is $O(L \log L)$.

### 4.9 Appendix: Proof of lemma 4.1

Let $P$ be a function satisfying $P(W)=Q(|W|)$. Denote by $\odot$ the element-wise multiplication operation. First note that

$$
\begin{equation*}
\|\operatorname{sign}(X \odot W) \odot X-W\|^{2} \leq\|X-W\|^{2} \tag{4.28}
\end{equation*}
$$

Too see this there are two cases. In the first case $X_{i}$ has the same sign as $W_{i} \operatorname{so} \operatorname{sign}\left(X_{i} W_{i}\right)=1$ and $\left(\operatorname{sign}\left(X_{i} W_{i}\right) X_{i}-W_{i}\right)^{2}=\left(X_{i}-W_{i}\right)^{2}$. In the second case, they have opposite signs, then $\left|\operatorname{sign}\left(X_{i} W_{i}\right) X_{i}-W_{i}\right|=\left|-X_{i}-W_{i}\right| \leq\left|X_{i}\right|+\left|W_{i}\right|=\left|X_{i}-W_{i}\right|$. Now, due to the assumption we
also have $P(\operatorname{sign}(X \odot W) \odot X)=P(X)$ i.e., $P$ doesn't change after changing signs of variables. With these observations we have:

$$
\begin{equation*}
\frac{1}{2}\|\operatorname{sign}(X \odot W) \odot X-W\|^{2}+P(\operatorname{sign}(X \odot W) \odot X) \leq \frac{1}{2}\|X-W\|^{2}+P(X) \tag{4.29}
\end{equation*}
$$

this implies

$$
\begin{equation*}
\min _{X} \frac{1}{2}\|\operatorname{sign}(X \odot W) \odot X-W\|^{2}+P(\operatorname{sign}(X \odot W) \odot X) \leq \min _{X} \frac{1}{2}\|X-W\|^{2}+P(X) \tag{4.30}
\end{equation*}
$$

Letting $\operatorname{sign}(X \odot W) \odot X=\widehat{X}$ we see that the opposite inequality also holds. Hence,

$$
\begin{equation*}
\min _{X} \frac{1}{2}\|\operatorname{sign}(X \odot W) \odot X-W\|^{2}+P(\operatorname{sign}(X \odot W) \odot X)=\min _{X} \frac{1}{2}\|X-W\|^{2}+P(X) \tag{4.31}
\end{equation*}
$$

Now, we modify the objective function in the left hand side as follows:

$$
\begin{array}{r}
\frac{1}{2}\|\operatorname{sign}(X \odot W) \odot X-W\|^{2}+P(\operatorname{sign}(X \odot W) \odot X) \\
=\frac{1}{2}\|\operatorname{sign}(W) \odot(\operatorname{sign}(X) \odot X-|W|)\|^{2}+P(\operatorname{sign}(X \odot W) \odot X)  \tag{4.32}\\
=\frac{1}{2}\|\operatorname{sign}(X) \odot X-|W|\|^{2}+P(\operatorname{sign}(X) \odot X)
\end{array}
$$

Noting that $\operatorname{sign}(X) \odot X$ is a matrix with nonnegative entries, we have the following:

$$
\begin{equation*}
\min _{X \geq 0} \frac{1}{2}\|X-|W|\|^{2}+P(X)=\min _{X} \frac{1}{2}\|X-W\|^{2}+P(X) \tag{4.33}
\end{equation*}
$$

Hence we see that both objective functions defining $\operatorname{prox}_{P}(W)$ and $\operatorname{prox}_{P}^{+}(|W|)$ have the same value. By the previous arguments the way to transform an element of prox ${ }_{P}^{+}(|W|)$ into an optimal solution of $\operatorname{prox}_{P}(W)$ is by multiplying by the sign of $W$, this follows from the last equation in eq. (4.32). This concludes the result.

### 4.10 Appendix: Experimental Setup Details

Model Architectures. We now detail the specific model architectures referred to in section 4.6. The dimensions of the multilayer perceptrons trained on both Fashion-MNIST and CIFAR-10 are listed in the table below. A single convolutional neural network architecture (CNN6) was trained on CIFAR-1. The CNN model consists of an input layer followed by four convolutional layers and two fully connected layers. The convolutional layers each have a kernel of size $3 \times 3$, and respectively had $32,32,64,64$ output channels. A max pooling layer with $2 \times 2$ kernel size was applied after each pair of convolutional layers. Then, two fully-connected layers were added, respectively of widths 1600 and 512. ReLu activation was applied after each convolutional layer except the last fully connected layer. No dropout was used for any of the models, as we aimed to isolate the effects of each regularization method. All models were trained with batches of 64 samples.

Table 4.2: Layer dimensions of fully-connected networks.

| Model | Dataset | Dimensions |  |  |
| :--- | :---: | :---: | :---: | :---: |
|  |  | Input | Hidden layers | Output |
| MLP2 | FMNIST | $[28,28,1]$ | $[500,500]$ | 10 |
| MLP3 | FMNIST | $[28,28,1]$ | $[500,500,500]$ | 10 |
| MLP3 | CIFAR10 | $[32,32,3]$ | $[1024,512,256]$ | 10 |

Hyperparameter selection. Training hyperparameters, such as step size $\gamma$, momentum $\beta$ and regularization strength $\lambda$ were selected through independent grid search for each regularization method, model architecture and dataset. An initial round of grid search was run on a wide grid of parameters for 50 training epochs on 3 independent runs. The best hyperparameters were then selected for each model, then the models were retrained for 200 training epochs on 5 independent runs. Model training included early stopping, by which each model's training was halted when maximal validation accuracy was reached. All grid searches included both stochastic gradient descent (SGD) and Adam optimizers.

Additional parameters were required by the Prox-AD (algorithm 4.3) and Prox-DIF (algorithm 4.4) regularization methods for the optimizer of the proximal objective; namely the inner step size $\gamma^{\prime}$ and inner momentum $\beta^{\prime}$. These parameters were pre-tuned by minimizing the proximal objective eq. (4.4) on randomly initialized network weights and a wide range of regularization strengths $\lambda^{\prime}=\lambda \cdot \gamma$ on for each architecture (refer to algorithm 4.2 for notation and Figure 4.2 for examples). During grid search for the outer loop parameters, the inner loop optimizer and parameters were selected for both Prox-AD and Prox-DIF as those minimizing the average final proximal objective after 250 iterations of Prox-AD over 5 independent runs. Once the best combination of the above parameters were selected, a third and final round of grid search was performed to tune the number of inner iterations $T^{\prime}$ and skip-prox parameter $B$. The values for the skip-prox parameter were chosen as divisors of the total number of batches so that every epoch training would end with a proximal step. All selected parameters are shown in the table below.

Noise robustness. Training hyperparameters were selected at noiseless conditions then reused for the noise robustness experiments at all noise levels, since the same parameters were often found to yield the highest performance under all noise conditions. The robustness to noisy data of different regularization methods was compared by evaluating the accuracy of each trained model on the same test set at different noise levels, as shown in (figure 4.4-bottom). Examples of images at varying amounts of image corruption with uniform random noise are shown below.


Figure 4.6: Examples of noisy images generated from the Fashion-MNIST dataset (top) and the CIFAR-10 dataset (bottom), at various noise levels ranging from $0 \%$ to $90 \%$.

### 4.11 Bibliographic Note

The candidate proposed the original idea of this paper and derived theorem 4.1 and lemma 4.1. The candidate also wrote code for algorithm 4.5 together with P. Rolland, and obtained all the numerical results in section 4.3. The candidate designed the experiments in section 4.6 and supervised Antoine Bonnet who obtained the numerical results in that section.

Table 4.3: Training parameters selected for each regularization method.

| Model | Dataset | Regularization | Hyperparameters |
| :---: | :---: | :---: | :---: |
| MLP2 | FMNIST | None | SGD; $\gamma=.1, \beta=.6$ |
|  |  | L2 | SGD; $\gamma=.1, \beta=.6, \lambda=10^{-4}$ |
|  |  | Path-AD | SGD; $\gamma=.1, \beta=.9, \lambda=10^{-6}$ |
|  |  | Prox-AD | Outer: SGD; $\gamma=.1, \beta=.9, \lambda=10^{-3}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-4}, T^{\prime}=250, B^{\prime}=125$ |
|  |  | Prox-DIF | Outer: SGD; $\gamma=.1, \beta=.9, \lambda=10^{-3}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-4}, T^{\prime}=250, B^{\prime}=125$ |
| MLP3 | FMNIST | None | SGD; $\gamma=10^{-2}, \beta=.9$ |
|  |  | L2 | SGD; $\gamma=.1, \beta=.5, \lambda=10^{-3}$ |
|  |  | Path-AD | SGD; $\gamma=.1, \beta=.9, \lambda=10^{-6}$ |
|  |  | Prox-AD | Outer: SGD; $\gamma=.1, \beta=.8, \lambda=10^{-4}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-3}, T^{\prime}=250, B^{\prime}=125$ |
|  |  | Prox-DIF | Outer: SGD; $\gamma=.1, \beta=.9, \lambda=10^{-3}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-3}, T^{\prime}=250, B^{\prime}=125$ |
| MLP3 | FMNIST | None | SGD; $\gamma=.1, \beta=.1$ |
|  |  | L2 | SGD; $\gamma=.1, \beta=.1, \lambda=10^{-4}$ |
|  |  | Path-AD | SGD; $\gamma=.1, \beta=.8, \lambda=10^{-7}$ |
|  |  | Prox-AD | Outer: SGD; $\gamma=.1, \beta=.7, \lambda=10^{-3}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-3}, T^{\prime}=250, B^{\prime}=125$ |
|  |  | Prox-DIF | Outer: SGD; $\gamma=.1, \beta=.7, \lambda=10^{-3}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-3}, T^{\prime}=250, B^{\prime}=125$ |
| MLP3 | FMNIST | None | SGD; $\gamma=10^{-2}, \beta=.9$ |
|  |  | L2 | SGD; $\gamma=10^{-2}, \beta=.9, \lambda=10^{-4}$ |
|  |  | Path-AD | SGD; $\gamma=10^{-2}, \beta=.9, \lambda=10^{-9}$ |
|  |  | Prox-AD | Outer: SGD; $\gamma=10^{-2}, \beta=.9, \lambda=10^{-5}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-4}, T^{\prime}=200, B^{\prime}=125$ |
|  |  | Prox-DIF | Outer: SGD; $\gamma=10^{-2}, \beta=.9, \lambda=10^{-5}$ |
|  |  |  | Inner: Adam; $\gamma^{\prime}=10^{-4}, T^{\prime}=200, B^{\prime}=125$ |

# 5 Controlling the Complexity and Lipschitz Constant Improves Polynomial Nets 


#### Abstract

Zhenyu Zhu, Fabian Latorre, Grigorios G Chrysos and Volkan Cevher. International Conference on Learning Representations (ICLR) 2022.


#### Abstract

While the class of Polynomial Nets demonstrates comparable performance to neural networks (NN), it currently has neither theoretical generalization characterization nor robustness guarantees. To this end, we derive new complexity bounds for the set of Coupled CP-Decomposition (CCP) and Nested Coupled CP-decomposition (NCP) models of Polynomial Nets in terms of the $\ell_{\infty}$-operator-norm and the $\ell_{2}$-operator norm. In addition, we derive bounds on the Lipschitz constant for both models to establish a theoretical certificate for their robustness. The theoretical results enable us to propose a principled regularization scheme that we also evaluate experimentally in six datasets and show that it improves the accuracy as well as the robustness of the models to adversarial perturbations. We showcase how this regularization can be combined with adversarial training, resulting in further improvements.


### 5.1 Introduction

Recently, high-degree Polynomial Nets (PNs) have been demonstrating state-of-the-art performance in a range of challenging tasks like image generation (Karras et al., 2019; Chrysos and Panagakis, 2020), image classification (Wang et al., 2018), reinforcement learning (Jayakumar et al., 2020), non-euclidean representation learning (Chrysos et al., 2020) and sequence models (Su et al., 2020). In particular, in public benchmarks like the Face verification on MegaFace task ${ }^{1}$ (Kemelmacher-Shlizerman et al., 2016), Polynomial Nets are currently the top performing model.

A major advantage of Polynomial Nets over traditional Neural Networks ${ }^{2}$ is that they are compatible with efficient Leveled Fully Homomorphic Encryption (LFHE) protocols (Brakerski et al., 2014). Such protocols allow efficient computation on encrypted data, but they only

[^6]support addition or multiplication operations i.e., polynomials. This has prompted an effort to adapt neural networks by replacing typical activation functions with polynomial approximations (Gilad-Bachrach et al., 2016; Hesamifard et al., 2018). Polynomial Nets do not need any adaptation to work with LFHE.

Without doubt, these arguments motivate further investigation about the inner-workings of Polynomial Nets. Surprisingly, little is known about the theoretical properties of such highdegree polynomial expansions, despite their success. Previous work on PNs (Chrysos et al., 2020; Chrysos and Panagakis, 2020) have focused on developing the foundational structure of the models as well as their training, but do not provide an analysis of their generalization ability or robustness to adversarial perturbations.

In contrast, such type of results are readily available for traditional feed-forward Deep Neural Networks, in the form of high-probability generalization error bounds (Neyshabur et al., 2015d; Bartlett et al., 2017a; Neyshabur et al., 2017; Golowich et al., 2018) or upper bounds on their Lipschitz constant (Scaman and Virmaux, 2018; Fazlyab et al., 2019b; Latorre et al., 2020a). Despite their similarity in the compositional structure, theoretical results for Deep Neural Networks ${ }^{2}$ do not apply to Polynomial Nets, as they are essentialy two non-overlapping classes of functions.

Why are such results important? First, they provide key theoretical quantities like the sample complexity of a hypothesis class: how many samples are required to succeed at learning in the PAC-framework. Second, they provide certified performance guarantees to adversarial perturbations (Szegedy et al., 2013; Goodfellow et al., 2015) via a worst-case analysis c.f. Scaman and Virmaux (2018). Most importantly, the bounds themselves provide a principled way to regularize the hypothesis class and improve their accuracy or robustness.

For example, Generalization and Lipschitz constant bounds of Deep Neural Networks that depend on the operator-norm of their weight matrices (Bartlett et al., 2017a; Neyshabur et al., 2017) have layed out the path for regularization schemes like spectral regularization (Yoshida and Miyato, 2017; Miyato et al., 2018b), Lipschitz-margin training (Tsuzuku et al., 2018b) and Parseval Networks (Cisse et al., 2017c), to name a few.

Indeed, such schemes have been observed in practice to improve the performance of Deep Neural Networks. Unfortunately, similar regularization schemes for Polynomial Nets do not exist due to the lack of analogous bounds. Hence, it is possible that PNs are not yet being used to their fullest potential. We believe that theoretical advances in their understanding might lead to more resilient and accurate models. In this work, we aim to fill the gap in the theoretical understanding of PNs. We summarize our main contributions as follows:

Rademacher Complexity Bounds. We derive bounds on the Rademacher Complexity of the Coupled CP-decomposition model (CCP) and Nested Coupled CP-decomposition model (NCP) of PNs, under the assumption of a unit $\ell_{\infty}$-norm bound on the input (theorems 5.1 and 5.3), a natural assumption in image-based applications. Analogous bounds for the $\ell_{2}-$
norm are also provided (subsections 5.9.3 and 5.10.3). Such bounds lead to the first known generalization error bounds for this class of models.

Lipschitz constant Bounds. To complement our understanding of the CCP and NCP models, we derive upper bounds on their $\ell_{\infty}$-Lipschitz constants (theorems 5.2 and 5.4), which are directly related to their robustness against $\ell_{\infty}$-bounded adversarial perturbations, and provide formal guarantees. Analogous results hold for any $\ell_{p}$-norm (subsections 5.9.4 and 5.10.4).

Regularization schemes. We identify key quantities that simultaneously control both Rademacher Complexity and Lipschitz constant bounds that we previously derived, i.e., the operator norms of the weight matrices in the Polynomial Nets. Hence, we propose to regularize the CCP and NCP models by constraining such operator norms. In doing so, our theoretical results indicate that both the generalization and the robustness to adversarial perturbations should improve. We propose a Projected Stochastic Gradient Descent scheme (algorithm 5.1), enjoying the same per-iteration complexity as vanilla back-propagation in the $\ell_{\infty}$-norm case, and a variant that augments the base algorithm with adversarial traning (algorithm 5.2).

Experiments. We conduct experimentation in five widely-used datasets on image recognition and on dataset in audio recognition. The experimentation illustrates how the aforementioned regularization schemes impact the accuracy (and the robust accuracy) of both CCP and NCP models, outperforming alternative schemes such as Jacobian regularization and the $L_{2}$ weight decay. Indeed, for a grid of regularization parameters we observe that there exists a sweet-spot for the regularization parameter which not only increases the test-accuracy of the model, but also its resilience to adversarial perturbations. Larger values of the regularization parameter also allow a trade-off between accuracy and robustness. The observation is consistent across all datasets and all adversarial attacks demonstrating the efficacy of the proposed regularization scheme.

### 5.2 Rademacher Complexity and Lipschitz constant bounds for Polynomial Nets

Notation. The symbol $\circ$ denotes the Hadamard (element-wise) product, the symbol • is the face-splitting product, while the symbol $\star$ denotes a convolutional operator. Matrices are denoted by uppercase letters e.g., $V$. Due to the space constraints, a detailed notation is deferred to section 5.7.

Assumption on the input distribution. Unless explicitly mentioned otherwise, we assume an $\ell_{\infty}$-norm unit bound on the input data i.e., $\|z\|_{\infty} \leq 1$ for any input $z$. This is the most common assumption in image-domain applications in contemporary deep learning, i.e., each pixel takes values in $[-1,1]$ interval. Nevertheless, analogous results for $\ell_{2}$-norm unit bound assumptions are presented in subsections 5.9.3, 5.9.4, 5.10.3 and 5.10.4.

We now introduce the basic concepts that will be developed throughout the paper i.e., the


Figure 5.1: Schematic of CCP model (left) and NCP model (right), where $\circ$ denotes the Hadamard product. Blue boxes correspond to learnable parameters. Green and red boxes denote input and output, respectively. Yellow boxes denote operations.

Rademacher Complexity of a class of functions (Bartlett and Mendelson, 2002) and the Lipschitz constant.

Definition 5.1 (Empirical Rademacher Complexity). Let $Z=\left\{z_{1}, \ldots, z_{n}\right\} \subseteq \mathbb{R}^{d}$ and let $\sigma_{j}, j=$ $1, \ldots, n$ be independent Rademacher random variables i.e., taking values uniformly in $\{-1,+1\}$. Let $\mathscr{F}$ be a class of real-valued functions over $\mathbb{R}^{d}$. The Empirical Rademacher complexity of $\mathscr{F}$ with respect to $Z$ is defined as follows:

$$
\mathscr{R}_{Z}(\mathscr{F}):=\mathbb{E}_{\sigma}\left[\sup _{f \in \mathscr{F}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right)\right] .
$$

Definition 5.2 (Lipschitz constant). Given two normed spaces $\left(\mathscr{X},\|\cdot\|_{\mathscr{X}}\right)$ and $(\mathscr{Y},\|\cdot\| \mathscr{Y}), a$ function $f: \mathscr{X} \rightarrow \mathscr{Y}$ is called Lipschitz continuous with Lipschitz constant $K \geq 0$ iffor all $x_{1}, x_{2}$ in $\mathscr{X}$ :

$$
\left\|f\left(x_{1}\right)-f\left(x_{2}\right)\right\| \mathscr{Y} \leq K\left\|x_{1}-x_{2}\right\|_{\mathscr{X}} .
$$

### 5.2.1 Coupled CP-Decomposition of Polynomial Nets (CCP model)

The Coupled CP-Decomposition (CCP) model of PNs (Chrysos et al., 2020) leverages a coupled CP Tensor decomposition (Kolda and Bader, 2009) to vastly reduce the parameters required to describe a high-degree polynomial, and allows its computation in a compositional fashion, much similar to a feed-forward pass through a traditional neural network. The CCP model was used in Chrysos and Panagakis (2020) to construct a generative model. CCP can be succintly defined as follows:

$$
\begin{equation*}
f(z)=C \circ \circ_{i=1}^{k} U_{i} z \tag{ССР}
\end{equation*}
$$

where $z \in \mathbb{R}^{d}$ is the input data, $f(z) \in \mathbb{R}^{o}$ is the output of the model and $U_{n} \in \mathbb{R}^{m \times d}, C \in \mathbb{R}^{o \times m}$ are the learnable parameters, where $m \in \mathbb{N}$ is the hidden rank. In figure 5.1 we provide a schematic of the architecture, while in subsection 5.8 .1 we include further details on the original CCP formulation (and how to obtain our equivalent re-parametrization) for the interested reader.

In theorem 5.1 we derive an upper bound on the complexity of CCP models with bounded $\ell_{\infty}$-operator-norms of the face-splitting product of the weight matrices. Its proof can be found in subsection 5.9.1. For a given CCP model, we derive an upper bound on its $\ell_{\infty}$-Lipschitz
constant in theorem 5.2 and its proof is given in subsection 5.9.4.

Theorem 5.1. Let $Z=\left\{z_{1}, \ldots, z_{n}\right\} \subseteq \mathbb{R}^{d}$ and suppose that $\left\|z_{j}\right\|_{\infty} \leq 1$ for all $j=1, \ldots, n$. Let

$$
\mathscr{F}_{C C P}^{k}:=\left\{f(z)=\left\langle c, \circ_{i=1}^{k} U_{i} z\right\rangle:\|c\|_{1} \leq \mu,\left\|\bullet k{ }_{i=1}^{k} U_{i}\right\|_{\infty} \leq \lambda\right\} .
$$

The Empirical Rademacher Complexity of $C C P_{k}$ ( $k$-degree CCP polynomials) with respect to $Z$ is bounded as:

$$
\mathscr{R}_{Z}\left(\mathscr{F}_{C C P}^{k}\right) \leq 2 \mu \lambda \sqrt{\frac{2 k \log (d)}{n}}
$$

Proof sketch of theorem 5.1. We now describe the core steps of the proof. For the interested reader, the complete and detailed proof steps are presented in subsection 5.9.1. First, Hölder's inequality is used to bound the Rademacher complexity as:

$$
\begin{equation*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{CCP}}^{k}\right)=\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\left\langle c, \sum_{j=1}^{n}\left[\sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right]\right\rangle \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right]\right\|_{\infty} \tag{5.1}
\end{equation*}
$$

This shows why the factor $\|c\|_{1} \leq \mu$ appears in the final bound. Then, using the mixed product property (Slyusar, 1999) and its extension to repeated Hadamard products (lemma 5.7 in subsection 5.7.3), we can rewrite the summation in the right-hand-side of (5.1) as follows:

$$
\sum_{j=1}^{n} \sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)=\sum_{j=1}^{n} \sigma_{j} \bullet{ }_{i=1}^{k}\left(U_{i}\right) *_{i=1}^{k}\left(z_{j}\right)=\bullet \bullet_{i=1}^{k}\left(U_{i}\right) \sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)
$$

This step can be seen as a linearization of the polynomial by lifting the problem to a higher dimensional space. We use this fact and the definition of the operator norm to further bound the term inside the $\ell_{\infty}$-norm in the right-hand-side of (5.1). Such term is bounded as the product of the $\ell_{\infty}$-operator norm of $\bullet{ }_{i=1}^{k}\left(U_{i}\right)$, and the $\ell_{\infty}$-norm of an expression involving the Rademacher variables $\sigma_{j}$ and the vectors $*_{i=1}^{k}\left(z_{j}\right)$. Finally, an application of Massart's Lemma (Shalev-Shwartz and Ben-David (2014a), Lemma 26.8) leads to the final result.

Theorem 5.2. The Lipschitz constant (with respect to the $\ell_{\infty}$-norm) of the function defined in eq. (CCP), restricted to the set $\left\{z \in \mathbb{R}^{d}:\|z\|_{\infty} \leq 1\right\}$ is bounded as:

$$
\operatorname{Lip}_{\infty}(f) \leq k\|C\|_{\infty} \prod_{i=1}^{k}\left\|U_{i}\right\|_{\infty}
$$

### 5.2.2 Nested Coupled CP-Decomposition (NCP model)

The Nested Coupled CP-Decomposition (NCP) model leverages a joint hierarchical decomposition, which provided strong results in both generative and discriminative tasks in Chrysos et al. (2020). A slight re-parametrization of the model (subsection 5.8.2) can be expressed with
the following recursive relation:

$$
\begin{equation*}
x_{1}=\left(A_{1} z\right) \circ\left(s_{1}\right), \quad x_{n}=\left(A_{n} z\right) \circ\left(S_{n} x_{n-1}\right), \quad f(z)=C x_{k}, \tag{NCP}
\end{equation*}
$$

where $z \in \mathbb{R}^{d}$ is the input vector and $C \in \mathbb{R}^{o \times m}, A_{n} \in \mathbb{R}^{m \times d}, S_{n} \in \mathbb{R}^{m \times m}$ and $s_{1} \in \mathbb{R}^{m}$ are the learnable parameters. In figure 5.1 we provide a schematic of the architecture.

In theorem 5.3 we derive an upper bound on the complexity of NCP models with bounded $\ell_{\infty}$-operator-norm of a matrix function of its parameters. Its proof can be found in subsection 5.10.1. For a given NCP model, we derive an upper bound on its $\ell_{\infty}$-Lipschitz constant in theorem 5.4 and its proof is given in subsection 5.10.4.

Theorem 5.3. Let $Z=\left\{z_{1}, \ldots, z_{n}\right\} \subseteq \mathbb{R}^{d}$ and suppose that $\left\|z_{j}\right\|_{\infty} \leq 1$ for all $j=1, \ldots, n$. Define the matrix $\Phi\left(A_{1}, S_{1}, \ldots, A_{n}, S_{n}\right):=\left(A_{k} \bullet S_{k}\right) \prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}$. Consider the class offunctions:

$$
\mathscr{F}_{N C P}^{k}:=\left\{f(z) \text { as in (NCP) }:\|C\|_{\infty} \leq \mu,\left\|\Phi\left(A_{1}, S_{1}, \ldots, A_{k}, S_{k}\right)\right\|_{\infty} \leq \lambda\right\}
$$

where $C \in \mathbb{R}^{1 \times m}$ (single output), thus, we will write it as $c$, and the corresponding bound also becomes $\|c\|_{1} \leq \mu$. The Empirical Rademacher Complexity of $N C P_{k}$ ( $k$-degree NCP polynomials) with respect to $Z$ is bounded as:

$$
\mathscr{R}_{Z}\left(\mathscr{F}_{N C P}^{k}\right) \leq 2 \mu \lambda \sqrt{\frac{2 k \log (d)}{n}} .
$$

Theorem 5.4. The Lipschitz constant (with respect to the $\ell_{\infty}$-norm) of the function defined in eq. (NCP), restricted to the set $\left\{z \in \mathbb{R}^{d}:\|z\|_{\infty} \leq 1\right\}$ is bounded as:

$$
\operatorname{Lip}_{\infty}(f) \leq k\|C\|_{\infty} \prod_{i=1}^{k}\left(\left\|A_{i}\right\|_{\infty}\left\|S_{i}\right\|_{\infty}\right)
$$

### 5.3 Algorithms

By constraining the quantities in the upper bounds on the Rademacher complexity (theorems 5.1 and 5.3), we can regularize the empirical loss minimization objective (Mohri et al., 2018a, Theorem 3.3). Such method would prevent overfitting and can lead to an improved accuracy. However, one issue with the quantities involved in theorems 5.1 and 5.3, namely

$$
\|\bullet k, \quad\|\left(A_{k} \cdot S_{k}\right) \prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i} \|_{\infty}
$$

is that projecting onto their level sets correspond to a difficult non-convex problem. Nevertheless, we can control an upper bound that depends on the $\ell_{\infty}$-operator norm of each weight matrix:

Lemma 5.1. It holds that $\left\|\bullet{ }_{i=1}^{k} U_{i}\right\|_{\infty} \leq \prod_{i=1}^{k}\left\|U_{i}\right\|_{\infty}$.

Lemma 5.2. It holds that $\left\|\left(A_{k} \bullet S_{k}\right) \prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right\|_{\infty} \leq \prod_{i=1}^{k}\left\|A_{i}\right\|_{\infty}\left\|S_{i}\right\|_{\infty}$.

The proofs of lemmas 5.1 and 5.2 can be found in subsection 5.9.2 and subsection 5.10.2. These results mean that by constraining the operator norms of each weight matrix, we can control the overall complexity of the CCP and NCP models.

Projecting a matrix onto an $\ell_{\infty}$-operator norm ball is a simple task that can be achieved by projecting each row of the matrix onto an $\ell_{1}$-norm ball, for example, using the well-known algorithm from Duchi et al. (2008a). The final optimization objective for training a regularized CCP is the following:

$$
\begin{equation*}
\min _{C, U_{1}, \ldots, U_{k}} \frac{1}{n} \sum_{i=1}^{n} L\left(C, U_{1}, \ldots, U_{k} ; x_{i}, y_{i}\right) \quad \text { subject to }\|C\|_{\infty} \leq \mu,\left\|U_{i}\right\|_{\infty} \leq \lambda \tag{5.2}
\end{equation*}
$$

where $\left(x_{i}, y_{i}\right)_{i=1}^{n}$ is the training dataset, $L$ is the loss function (e.g., cross-entropy) and $\mu, \lambda$ are the regularization parameters. We notice that the constraints on the learnable parameters $U_{i}$ and $C$ have the effect of simultaneously controlling the Rademacher Complexity and the Lipschitz constant of the CCP model. For the NCP model, an analogous objective function is used.

To solve the optimization problem in eq. (5.2) we will use a Projected Stochastic Gradient Descent method (algorithm 5.1). We also propose a variant that combines Adversarial Training with the projection step (algorithm 5.2) with the goal of increasing robustness to adversarial examples.

```
Algorithm 5.1 Projected SGD
Input: dataset \(Z\), learning rate \(\left\{\gamma_{t}>0\right\}_{t=0}^{T-1}\), iterations \(T\), hyper-parameters \(R, f\), Loss \(L\).
Output: model with parameters \(\theta\).
    Initialize \(\theta\).
    for \(t=0\) to \(T-1\) do
        Sample ( \(x, y\) ) from \(Z\)
        \(\theta=\theta-\gamma_{t} \nabla_{\theta} L(\theta ; x, y)\).
        if \(t \bmod f=0\) then
            \(\theta=\prod_{\left\{\theta:\|\theta\|_{\infty} \leq R\right\}}(\theta)\)
        end if
    end for
```

```
Algorithm 5.2 Projected SGD + Adversarial Training
Input: dataset \(Z\), learning rate \(\left\{\gamma_{t}>0\right\}_{t=0}^{T-1}\), iterations \(T\) and \(n\), hyper-parameters \(R, f, \epsilon\) and
\(\alpha\), Loss \(L\)
Output: model with parameters \(\theta\).
    Initialize \(\theta\).
    for \(t=0\) to \(T-1\) do
        Sample ( \(x, y\) ) from \(Z\)
        for \(i=0\) to \(n-1\) do
            \(x^{\text {adv }}=\prod_{\left\{x^{\prime}:\left\|x^{\prime}-x\right\|_{\infty} \leq \epsilon\right\}}\left\{x+\alpha \nabla_{x} L(\theta ; x, y)\right\}\)
        end for
        \(\theta=\theta-\gamma_{t} \nabla_{\theta} L\left(\theta ; x^{\text {adv }}, y\right)\)
        if \(t \bmod f=0\) then
            \(\theta=\prod_{\left\{\theta:\|\theta\|_{\infty} \leq R\right\}}(\theta)\)
        end if
    end for
```

In algorithms 5.1 and 5.2 the parameter $f$ is set in practice to a positive value, so that the projection (denoted by $\Pi$ ) is made only every few iterations. The variable $\theta$ represents the weight matrices of the model, and the projection in the last line should be understood as applied independently for every weight matrix. The regularization parameter $R$ corresponds to the variables $\mu, \lambda$ in eq. (5.2).

Convolutional layers. Frequently, convolutions are employed in the literature, especially in the image-domain. It is important to understand how our previous results extend to this case, and how the proposed algorithms work in that case. Below, we show that the $\ell_{\infty}$-operator norm of the convolutional layer (as a linear operator) is related to the $\ell_{\infty}$-operator norm of the kernel after a reshaping operation. For simplicity, we consider only convolutions with zero padding.

We study the cases of 1D, 2D and 3D convolutions. For clarity, we mention below the result for the 3D convolution, since this is relevant to our experimental validation, and we defer the other two cases to section 5.11.

Theorem 5.5. Let $A \in \mathbb{R}^{n \times m \times r}$ be an input image and let $K \in \mathbb{R}^{h \times h \times r \times o}$ be a convolutional kernel with o output channels. For simplicity assume that $k \leq \min (n, m)$ is odd. Denote by $B=K \star A$ the output of the convolutional layer. Let $U \in \mathbb{R}^{n m o \times n m r}$ be the matrix such that $\operatorname{vec}(B)=U \operatorname{vec}(A)$ i.e., $U$ is the matrix representation of the convolution. Let $M(K) \in \mathbb{R}^{o \times h h r}$ be the matricization of $K$, where each row contains the parameters of a single output channel of the convolution. It holds that: $\|U\|_{\infty}=\|M(K)\|_{\infty}$.

Thus, we can control the $\ell_{\infty}$-operator-norm of a convolutional layer during training by controlling that of the reshaping of the kernel, which is done with the same code as for fully
connected layers. It can be seen that when the padding is non-zero, the result still holds.

### 5.4 Numerical Evidence

The generalization properties and the robustness of PNs are numerically verified in this section. We evaluate the robustness to three widely-used adversarial attacks in sec. 5.4.2. We assess whether the compared regularization schemes can also help in the case of adversarial training in sec. 5.4.3. Experiments with additional datasets, models (NCP models), adversarial attacks (APGDT, PGDT) and layer-wise bound (instead of a single bound for all matrices) are conducted in section 5.12 due to the restricted space. The results exhibit a consistent behavior across different adversarial attacks, different datasets and different models. Whenever the results differ, we explicitly mention the differences in the main body below.

### 5.4.1 Experimental Setup

The accuracy is reported as as the evaluation metric for every experiment, where a higher accuracy translates to better performance.

Datasets and Benchmark Models: We conduct experiments on the popular datasets of Fashion-MNIST (Xiao et al., 2017b), E-MNIST (Cohen et al., 2017) and CIFAR-10 (Krizhevsky et al., 2014). The first two datasets include grayscale images of resolution $28 \times 28$, while CIFAR10 includes 60,000 RGB images of resolution $32 \times 32$. Each image is annotated with one out of the ten categories. We use two popular regularization methods from the literature for comparison, i.e., Jacobian regularization (Hoffman et al., 2019) and $L_{2}$ regularization (weight decay).

Models: We report results using the following three models: 1) a $4^{\text {th }}$-degree CCP model named "PN-4", 2) a $10^{\text {th }}$-degree CCP model referenced as "PN-10" and 3) a $4^{\text {th }}$-degree Convolutional CCP model called "PN-Conv". In the PN-Conv, we have replaced all the $U_{i}$ matrices with convolutional kernels. None of the variants contains any activation functions.

Hyper-parameters: Unless mentioned otherwise, all models are trained for 100 epochs with a batch size of 64 . The initial value of the learning rate is 0.001 . After the first 25 epochs, the learning rate is multiplied by a factor of 0.2 every 50 epochs. The SGD is used to optimize all the models, while the cross-entropy loss is used. In the experiments that include projection or adversarial training, the first 50 epochs are pre-training, i.e., training only with the crossentropy loss. The projection is performed every ten iterations.

Adversarial Attack Settings: We utilize two widely used attacks: a) Fast Gradient Sign Method (FGSM) and b) Projected Gradient Descent (PGD). In FGSM the hyper-parameter $\epsilon$ represents the step size of the adversarial attack. In PGD there is a triple of parameters ( $\epsilon_{\text {total }}, n_{\text {iters }}, \epsilon_{\text {iter }}$ ), which represent the maximum step size of the total adversarial attack, the number of steps to perform for a single attack, and the step size of each adversarial attack step respectively. We

|  | Method | No proj. | Our method Fashion | Jacobian n-MNIST | $L_{2}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| PN-4 | Clean | $87.28 \pm 0.18 \%$ | 87.32 $\pm$ 0.14\% | $86.24 \pm 0.14 \%$ | $87.31 \pm 0.13 \%$ |
|  | FGSM-0.1 | $12.92 \pm 2.74 \%$ | $46.43 \pm 0.95 \%$ | $17.90 \pm 6.51 \%$ | $13.80 \pm 3.65 \%$ |
|  | PGD-(0.1, 20, 0.01) | $5.64 \pm 1.76 \%$ | $49.58 \pm 0.59 \%$ | $12.23 \pm 5.63 \%$ | $5.01 \pm 2.44 \%$ |
|  | PGD-(0.3, 20, 0.03) | $0.18 \pm 0.16 \%$ | $28.96 \pm 2.31 \%$ | $1.27 \pm 1.29 \%$ | $0.28 \pm 0.18 \%$ |
| PN-10 | Clean | $88.48 \pm 0.17 \%$ | 88.72 $\pm$ 0.12\% | $88.12 \pm 0.11 \%$ | $88.46 \pm 0.19 \%$ |
|  | FGSM-0.1 | $15.96 \pm 1.00 \%$ | $44.71 \pm 1.24 \%$ | $19.52 \pm 1.14 \%$ | $16.51 \pm 2.33 \%$ |
|  | PGD-(0.1, 20, 0.01) | $1.94 \pm 0.82 \%$ | $47.94 \pm 2.29 \%$ | $5.44 \pm 0.81 \%$ | $2.16 \pm 0.95 \%$ |
|  | PGD-(0.3, 20, 0.03) | $0.02 \pm 0.03 \%$ | $\mathbf{3 0 . 5 1} \pm \mathbf{1 . 2 2 \%}$ | $0.05 \pm 0.02 \%$ | $0.01 \pm 0.02 \%$ |
| PN-Conv | Clean | $86.36 \pm 0.21 \%$ | $86.38 \pm 0.26 \%$ | $84.69 \pm 0.44 \%$ | $\mathbf{8 6 . 4 5} \pm \mathbf{0 . 2 1 \%}$ |
|  | FGSM-0.1 | $10.80 \pm 1.82 \%$ | $48.15 \pm 1.23 \%$ | $10.62 \pm 0.77 \%$ | $10.73 \pm 1.58 \%$ |
|  | PGD-(0.1, 20, 0.01) | $9.37 \pm 1.00 \%$ | $46.63 \pm 3.68 \%$ | $10.20 \pm 0.32 \%$ | $8.96 \pm 0.83 \%$ |
|  | PGD-(0.3, 20, 0.03) | $1.75 \pm 0.83 \%$ | $28.94 \pm 1.20 \%$ | $8.26 \pm 1.05 \%$ | $2.03 \pm 0.99 \%$ |
|  |  | E-MNIST |  |  |  |
| PN-4 | Clean | $84.27 \pm 0.26 \%$ | $\mathbf{8 4 . 3 4} \pm \mathbf{0 . 3 1 \%}$ | $81.99 \pm 0.33 \%$ | $84.22 \pm 0.33 \%$ |
|  | FGSM-0.1 | $8.92 \pm 1.99 \%$ | 27.56 $\pm$ 3.32\% | $14.96 \pm 1.32 \%$ | $8.18 \pm 3.48 \%$ |
|  | PGD-(0.1, 20, 0.01) | $6.24 \pm 1.43 \%$ | $29.46 \pm 2.73 \%$ | $6.75 \pm 2.92 \%$ | $5.93 \pm 1.97 \%$ |
|  | PGD-(0.3, 20, 0.03) | $1.22 \pm 0.85 \%$ | $19.07 \pm 0.98 \%$ | $3.06 \pm 0.53 \%$ | $1.00 \pm 0.76 \%$ |
| PN-10 | Clean | $89.31 \pm 0.09 \%$ | $\mathbf{9 0 . 5 6} \pm \mathbf{0 . 1 0 \%}$ | $89.19 \pm 0.07 \%$ | $89.23 \pm 0.13 \%$ |
|  | FGSM-0.1 | $15.56 \pm 1.16 \%$ | $\mathbf{3 7 . 1 1} \pm \mathbf{2 . 8 1 \%}$ | $24.21 \pm 1.89 \%$ | $16.30 \pm 1.82 \%$ |
|  | PGD-(0.1, 20, 0.01) | $2.63 \pm 0.65 \%$ | $37.89 \pm 2.91 \%$ | $9.18 \pm 1.09 \%$ | $2.33 \pm 0.43 \%$ |
|  | PGD-(0.3, 20, 0.03) | $0.00 \pm 0.00 \%$ | $20.47 \pm 0.96 \%$ | $0.11 \pm 0.08 \%$ | $0.02 \pm 0.03 \%$ |
| PN-Conv | Clean | $91.49 \pm 0.29 \%$ | 91.57 $\pm$ 0.19\% | $90.38 \pm 0.13 \%$ | $91.41 \pm 0.18 \%$ |
|  | FGSM-0.1 | $4.28 \pm 0.55 \%$ | $\mathbf{3 5 . 3 9} \pm 7.51 \%$ | $3.88 \pm 0.04 \%$ | $4.13 \pm 0.41 \%$ |
|  | PGD-(0.1, 20, 0.01) | $3.98 \pm 0.82 \%$ | $33.75 \pm 7.17 \%$ | $3.86 \pm 0.01 \%$ | $4.83 \pm 0.87 \%$ |
|  | PGD-(0.3, 20, 0.03) | $3.24 \pm 0.76 \%$ | $\mathbf{2 8 . 1 0} \pm \mathbf{3 . 2 7 \%}$ | $3.84 \pm 0.01 \%$ | $2.76 \pm 0.65 \%$ |

Table 5.1: Comparison of regularization techniques on Fashion-MNIST (top) and E-MNIST (bottom). In each dataset, the base networks are PN-4, i.e., a $4^{\text {th }}$ degree polynomial, on the top four rows, $\mathrm{PN}-10$, i.e., a $10^{\text {th }}$ degree polynomial, on the middle four rows and PN-Conv, i.e., a $4^{\text {th }}$ degree polynomial with convolutions, on the bottom four rows. Our projection method exhibits the best performance in all three attacks, with the difference on accuracy to stronger attacks being substantial.
consider the following hyper-parameters for the attacks: a) FGSM with $\epsilon=0.1$, b) PGD with parameters ( $0.1,20,0.01$ ), c) PGD with parameters ( $0.3,20,0.03$ ).

### 5.4.2 Evaluation of the robustness of PNs

In the next experiment, we assess the robustness of PNs under adversarial noise. That is, the method is trained on the train set of the respective dataset and the evaluation is performed on the test set perturbed by additive adversarial noise. That is, each image is individually perturbed based on the respective adversarial attack. The proposed method implements algorithm 5.1.

The quantitative results in both Fashion-MNIST and E-MNIST using PN-4, PN-10 and PNConv under the three attacks are reported in table 5.1. The column 'No-proj' exhibits the plain SGD training (i.e., without regularization), while the remaining columns include the proposed regularization, Jacobian and the $L_{2}$ regularization respectively. The results without regularization exhibit a substantial decrease in accuracy for stronger adversarial attacks. The proposed regularization outperforms all methods consistently across different adversarial attacks. Interestingly, the stronger the adversarial attack, the bigger the difference of the proposed regularization scheme with the alternatives of Jacobian and $L_{2}$ regularizations.

Next, we learn the networks with varying projection bounds. The results on Fashion-MNIST and E-MNIST are visualized in figure 5.2, where the $x$-axis is plotted in log-scale. As a reference point, we include the clean accuracy curves, i.e., when there is no adversarial noise. Projection bounds larger than 2 (in the log-axis) leave the accuracy unchanged. As the bounds decrease, the results gradually improve. This can be attributed to the constraints the projection bounds impose into the $U_{i}$ matrices.

Similar observations can be made when evaluating the clean accuracy (i.e., no adversarial noise in the test set). However, in the case of adversarial attacks a tighter bound performs better, i.e., the best accuracy is exhibited in the region of 0 in the log-axis. The projection bounds can have a substantial improvement on the performance, especially in the case of stronger adversarial attacks, i.e., PGD. Notice that all in the aforementioned cases, the intermediate values of the projection bounds yield an increased performance in terms of the test-accuracy and the adversarial perturbations.

Beyond the aforementioned datasets, we also validate the proposed method on CIFAR-10 dataset. The results in figure 5.3 and table 5.2 exhibit similar patterns as the aforementioned experiments. Although the improvement is smaller than the case of Fashion-MNIST and EMNIST, we can still obtain about $10 \%$ accuracy improvement under three different adversarial attacks.

### 5.4.3 Adversarial training (AT) on PNs

Adversarial training has been used as a strong defence against adversarial attacks. In this experiment we evaluate whether different regularization methods can work in conjunction with adversarial training that is widely used as a defence method. Since multi-step adversarial attacks are computationally intensive, we utilize the FGSM attack during training, while we evaluate the trained model in all three adversarial attacks. For this experiment we select PN-10 as the base model. The proposed model implements algorithm 5.2.

The accuracy is reported in table 5.3 with Fashion-MNIST on the top and E-MNIST on the bottom. In the FGSM attack, the difference of the compared methods is smaller, which is expected since similar attack is used for the training. However, for stronger attacks the difference becomes pronounced with the proposed regularization method outperforming
-- Clean ---- FGSM_0.1
--4-- PGD_0.1_0.01_20


Figure 5.2: Adversarial attacks during testing on Fashion-MNIST (top), E-MNIST (bottom) with the x -axis is plotted in log-scale. Note that intermediate values of projection bounds yield the highest accuracy. The patterns are consistent in both datasets and across adversarial attacks.
$\rightarrow$ Clean --+ FGSM_0.1 - -- PGD_0.1_0.01_20 $\rightarrow$ PGD_0.3_0.03_20


Figure 5.3: Adversarial attacks during testing on CIFAR-10.

| Model | PN-Conv |  |  |  |
| :--- | :---: | :---: | :---: | :---: |
| Projection | No-proj | Our method | Jacobian | $L_{2}$ |
| Clean accuracy | $65.09 \pm 0.14 \%$ | $\mathbf{6 5 . 2 2} \pm \mathbf{0 . 1 3} \%$ | $64.43 \pm 0.19 \%$ | $65.11 \pm 0.08 \%$ |
| FGSM-0.1 | $6.00 \pm 0.53 \%$ | $\mathbf{1 5 . 1 3} \pm \mathbf{0 . 8 1} \%$ | $3.34 \pm 0.40 \%$ | $1.27 \pm 0.10 \%$ |
| PGD-(0.1, 20, 0.01) | $7.08 \pm 0.68 \%$ | $\mathbf{1 5 . 1 7} \pm \mathbf{0 . 8 8} \%$ | $1.74 \pm 0.14 \%$ | $1.05 \pm 0.05 \%$ |
| PGD-(0.3, 20, 0.03) | $0.41 \pm 0.09 \%$ | $\mathbf{1 1 . 7 1} \pm \mathbf{1 . 1 1} \%$ | $0.04 \pm 0.02 \%$ | $0.51 \pm 0.04 \%$ |

Table 5.2: Evaluation of the robustness of PN models on CIFAR-10. Each line refers to a different adversarial attack. The projection offers an improvement in the accuracy in each case; in PGD attacks the projection improves the accuracy by a significant margin.
both the Jacobian and the $L_{2}$ regularization methods.

| Method | AT | Our method + AT | Jacobian + AT | $L_{2}+$ AT |
| :--- | :---: | :---: | :---: | :---: |
|  | Adversarial training (AT) with PN-10 on Fashion-MNIST |  |  |  |
| FGSM-0.1 | $65.33 \pm 0.46 \%$ | $\mathbf{6 5 . 6 4} \pm \mathbf{0 . 3 5 \%}$ | $62.04 \pm 0.22 \%$ | $65.62 \pm 0.15 \%$ |
| PGD-(0.1, 20, 0.01) | $57.45 \pm 0.35 \%$ | $\mathbf{5 9 . 8 9} \pm \mathbf{0 . 2 2 \%}$ | $57.42 \pm 0.24 \%$ | $57.40 \pm 0.36 \%$ |
| PGD-(0.3, 20, 0.03) | $24.46 \pm 0.45 \%$ | $\mathbf{3 9 . 7 9} \pm \mathbf{1 . 4 0 \%}$ | $25.59 \pm 0.20 \%$ | $24.99 \pm 0.57 \%$ |
| Adversarial training (AT) |  |  |  |  |
| with PN-10 on $E-M N I S T$ |  |  |  |  |
| FGSM-0.1 | $78.30 \pm 0.18 \%$ | $\mathbf{7 8 . 6 1} \pm \mathbf{0 . 5 8 \%}$ | $70.11 \pm 0.18 \%$ | $78.31 \pm 0.32 \%$ |
| PGD-(0.1, 20, 0.01) | $68.40 \pm 0.32 \%$ | $\mathbf{6 8 . 5 1} \pm \mathbf{0 . 1 9 \%}$ | $64.61 \pm 0.16 \%$ | $68.41 \pm 0.37 \%$ |
| PGD-(0.3, 20, 0.03) | $35.58 \pm 0.33 \%$ | $\mathbf{4 2 . 2 2} \pm \mathbf{0 . 6 0 \%}$ | $39.83 \pm 0.24 \%$ | $35.17 \pm 0.46 \%$ |

Table 5.3: Comparison of regularization techniques on (a) Fashion-MNIST (top) and (b) EMNIST (bottom) along with adversarial training (AT). The base network is a PN-10, i.e., $10^{\text {th }}$ degree polynomial. Our projection method exhibits the best performance in all three attacks.

The limitations of the proposed work are threefold. Firstly, theorem 5.1 relies on the $\ell_{\infty^{-}}$ operator norm of the face-splitting product of the weight matrices, which in practice we relax in lemma 5.1 for performing the projection. In the future, we aim to study if it is feasible to compute the non-convex projection onto the set of PNs with bounded $\ell_{\infty_{\infty}}$-norm of the face-splitting product of the weight matrices. This would allow us to let go off the relaxation argument and directly optimize the original tighter Rademacher Complexity bound (theorem 5.1).

Secondly, the regularization effect of the projection differs across datasets and adversarial attacks, a topic that is worth investigating in the future.

Thirdly, our bounds do not take into account the algorithm used, which corresponds to a variant of the Stochastic Projected Gradient Descent, and hence any improved generalization properties due to possible uniform stability (Bousquet and Elisseeff, 2002) of the algorithm or implicit regularization properties (Neyshabur, 2017), do not play a role in our analysis.

### 5.5 Related Work

Rademacher Complexity: Known bounds for the class of polynomials are a consequence of more general result for kernel methods (Mohri et al., 2018a, Theorem 6.12). Support Vector Machines (SVMs) with a polynomial kernel of degree $k$ effectively correspond to a general polynomial with the same degree. In contrast, our bound is tailored to the parametric definition of the CCP and the NCP models, which are a subset of the class of all polynomials. Hence, they are tighter than the general kernel complexity bounds.

Bounds for the class of neural networks were stablished in (Bartlett et al., 2017a; Neyshabur et al., 2017), but they require a long and technical proof, and in particular it assumes an $\ell_{2}$-bound on the input, which is incompatible with image-based applications. This bound
also depend on the product of spectral norms of each layer. In contrast, our bounds are more similar in spirit to the path-norm-based complexity bounds (Neyshabur et al., 2015d), as they depend on interactions between neurons at different layers. This interaction precisely corresponds to the face-splitting product between weight matrices that appears in theorem 5.1.

Lipschitz constant: A variety of methods have been proposed for estimating the Lipschitz constant of neural networks. For example, Scaman and Virmaux (2018) (SVD), Fazlyab et al. (2019b) (Semidefinite programming) and Latorre et al. (2020a) (Polynomial Optimization) are expensive optimization methods to compute tighter bounds on such constant. These methods are unusable in our case as they would require a non-trivial adaptation to work with Polynomial Nets. In contrast we find an upper bound that applies to such family of models, and it can be controlled with efficient $\ell_{\infty}$-operator-norm projections. However, our bounds might not be the tightest. Developing tighter methods to bound and control the Lipschitz constant for Polynomial Nets is a promising direction of future research.

### 5.6 Appendix: introduction

The Appendix is organized as follows:
$\triangleright$ The related work is summarized in section 5.5.
$\triangleright$ In section 5.7 the notation and the core Lemmas from the literature are described.
$\triangleright$ Further details on the Polynomial Nets are provided in section 5.8.
$\triangleright$ The proofs on the CCP and the NCP models are added in section 5.9 and section 5.10 respectively.
$\triangleright$ The extensions of the theorems for convolutional layers and their proofs are detailed in section 5.11.
$\triangleright$ Additional experiments are included in section 5.12.

### 5.7 Appendix: Background

Below, we develop a detailed notation in subsection 5.7.1, we include related definitions in subsection 5.7.2 and Lemmas required for our proofs in subsection 5.7.3. The goal of this section is to cover many of the required information for following the proofs and the notation we follow in this work. Readers familiar with the different matrix/vector products, e.g., KhatriRao or face-splitting product, and with basic inequalities, e.g., Hölder's inequality, can skip to the next section.

Table 5.4: Symbols for various matrix products. The precise definitions of the products are included in subsection 5.7.2 for completion.

| Symbol | Definition |
| :---: | :---: |
| $\circ$ | Hadamard (element-wise) product. |
| $*$ | Column-wise Khatri-Rao product. |
| $\bullet$ | Face-splitting product. |
| $\otimes$ | Kronecker product. |
| $\star$ | Convolution. |

### 5.7.1 Notation

Different matrix products and their associated symbols are referenced in table 5.4, while matrix operations on a matrix $A$ are defined on table 5.5. Every matrix product, e.g., Hadamard product, can be used in two ways: a) $A \circ B$, which translates to Hadamard product of matrices $A$ and $B, \mathrm{~b}) \circ{ }_{i=1}^{N} A_{i}$ abbreviates the Hadamard products $\underbrace{A_{1} \circ A_{2} \circ \ldots A_{N}}_{N \text { products }}$.

Table 5.5: Operations and symbols on a matrix $A$.

| Symbol | Definition |
| :---: | :---: |
| $\\|A\\|_{\infty}$ | $\ell_{\infty}$-operator-norm; corresponds to the maximum $\ell_{1}$-norm of its rows. |
| $A^{i}$ | $i^{\text {th }}$ row of $A$. |
| $a_{i, j}$ | $(i, j)^{\text {th }}$ element of $A$. |
| $A_{i, j}$ | $(i, j)^{\text {th }}$ block of a block-matrix $A$ |
| $A_{i}$ | The i-th matrix in a set of matrices $\left\{A_{1}, \cdots, A_{N}\right\}$. |

The symbol $x_{i}^{j}$ refers to $j^{\text {th }}$ element of vector $x_{i}$.

### 5.7.2 Definitions

For thoroughness, we include the definitions of the core products that we use in this work. Specifically, the definitions of the Hadamard product (definition 5.3), Kronecker product (definition 5.4), the Khatri-Rao product (definition 5.5), column-wise Khatri-Rao product (definition 5.6) and the face-splitting product (definition 5.7) are included.

Definition 5.3 (Hadamard product). For two matrices $A$ and $B$ of the same dimension $m \times n$, the Hadamard product $A \circ B$ is a matrix of the same dimension as the operands, with elements given by

$$
(a \circ b)_{i, j}=a_{i, j} b_{i, j}
$$

Definition 5.4 (Kronecker product). If $A$ is an $m \times n$ matrix and $B$ is a $p \times q$ matrix, then the

Kronecker product $A \otimes B$ is the $p m \times q n$ block matrix, given as follows:

$$
A \otimes B=\left[\begin{array}{ccc}
a_{1,1} B & \cdots & a_{1, n} B \\
\vdots & \ddots & \vdots \\
a_{m, 1} B & \cdots & a_{m, n} B
\end{array}\right]
$$

Example: the Kronecker product of the matrices $A \in \mathbb{R}^{2 \times 2}$ and $B \in \mathbb{R}^{2 \times 2}$ is computed below:

$$
\underbrace{\left[\begin{array}{ll}
a_{1,1} & a_{1,2} \\
a_{2,1} & a_{2,2}
\end{array}\right]}_{A} \otimes \underbrace{\left[\begin{array}{ll}
b_{1,1} & b_{1,2} \\
b_{2,1} & b_{2,2}
\end{array}\right]}_{B}=\underbrace{\left[\begin{array}{llll}
a_{1,1} b_{1,1} & a_{1,1} b_{1,2} & a_{1,2} b_{1,1} & a_{1,2} b_{1,2} \\
a_{1,1} b_{2,1} & a_{1,1} b_{2,2} & a_{1,2} b_{2,1} & a_{1,2} b_{2,2} \\
a_{2,1} b_{1,1} & a_{2,1} b_{1,2} & a_{2,2} b_{1,1} & a_{2,2} b_{1,2} \\
a_{2,1} b_{2,1} & a_{2,1} b_{2,2} & a_{2,2} b_{2,1} & a_{2,2} b_{2,2}
\end{array}\right]}_{A \otimes B} .
$$

Definition 5.5 (Khatri-Rao product). The Khatri-Rao product is defined as:

$$
A * B=\left(A_{i, j} \otimes B_{i, j}\right)_{i, j}
$$

in which the $(i, j)$-th block is the $m_{i} p_{i} \times n_{j} q_{j}$ sized Kronecker product of the corresponding blocks of $A$ and $B$, assuming the number of row and column partitions of both matrices is equal. The size of the product is then $\left(\sum_{i} m_{i} p_{i}\right) \times\left(\sum_{i} n_{j} q_{j}\right)$.

Example: if $A$ and $B$ both are $2 \times 2$ partitioned matrices e.g.:

$$
\begin{aligned}
& A=\left[\begin{array}{c|c}
A_{1,1} & A_{1,2} \\
\hline A_{2,1} & A_{2,2}
\end{array}\right]=\left[\begin{array}{ll|l}
a_{1,1} & a_{1,2} & a_{1,3} \\
a_{2,1} & a_{2,2} & a_{2,3} \\
\hline a_{3,1} & a_{3,2} & a_{3,3}
\end{array}\right], \\
& B=\left[\begin{array}{l|l}
B_{1,1} & B_{1,2} \\
\hline B_{2,1} & B_{2,2}
\end{array}\right]=\left[\begin{array}{l|ll}
b_{1,1} & b_{1,2} & b_{1,3} \\
\hline b_{2,1} & b_{2,2} & b_{2,3} \\
b_{3,1} & b_{3,2} & b_{3,3}
\end{array}\right],
\end{aligned}
$$

then we obtain the following:

$$
A * B=\left[\begin{array}{c|c}
A_{1,1} \otimes B_{1,1} & A_{1,2} \otimes B_{1,2} \\
\hline A_{2,1} \otimes B_{2,1} & A_{2,2} \otimes B_{2,2}
\end{array}\right]=\left[\begin{array}{cc|cc}
a_{1,1} b_{1,1} & a_{1,2} b_{1,1} & a_{1,3} b_{1,2} & a_{1,3} b_{1,3} \\
a_{2,1} b_{1,1} & a_{2,2} b_{1,1} & a_{2,3} b_{1,2} & a_{2,3} b_{1,3} \\
\hline a_{3,1} b_{2,1} & a_{3,2} b_{2,1} & a_{3,3} b_{2,2} & a_{3,3} b_{2,3} \\
a_{3,1} b_{3,1} & a_{3,2} b_{3,1} & a_{3,3} b_{3,2} & a_{3,3} b_{3,3}
\end{array}\right] .
$$

Definition 5.6 (Column-wise Khatri-Rao product). A column-wise Kronecker product of two matrices may also be called the Khatri-Rao product. This product assumes the partitions of the matrices are their columns. In this case $m_{1}=m, p_{1}=p, n=q$ and for each $j: n_{j}=p_{j}=1$.

The resulting product is a $m p \times n$ matrix of which each column is the Kronecker product of the corresponding columns of $A$ and $B$.

Example: the Column-wise Khatri-Rao product of the matrices $A \in \mathbb{R}^{2 \times 3}$ and $B \in \mathbb{R}^{3 \times 3}$ is computed below:

$$
\underbrace{\left[\begin{array}{lll}
a_{1,1} & a_{1,2} & a_{1,3} \\
a_{2,1} & a_{2,2} & a_{2,3}
\end{array}\right]}_{A} * \underbrace{\left[\begin{array}{lll}
b_{1,1} & b_{1,2} & b_{1,3} \\
b_{2,1} & b_{2,2} & b_{2,3} \\
b_{3,1} & b_{3,2} & b_{3,3}
\end{array}\right]}_{B}=\underbrace{\left[\begin{array}{lll}
a_{1,1} b_{1,1} & a_{1,2} b_{1,2} & a_{1,3} b_{1,3} \\
a_{1,1} b_{2,1} & a_{1,2} b_{2,2} & a_{1,3} b_{2,3} \\
a_{1,1} b_{3,1} & a_{1,2} b_{3,2} & a_{1,3} b_{3,3} \\
a_{2,1} b_{1,1} & a_{2,2} b_{1,2} & a_{2,3} b_{1,3} \\
a_{2,1} b_{2,1} & a_{2,2} b_{2,2} & a_{2,3} b_{2,3} \\
a_{2,1} b_{3,1} & a_{2,2} b_{3,2} & a_{2,3} b_{3,3}
\end{array}\right]}_{A * B} .
$$

From here on, all $*$ refer to Column-wise Khatri-Rao product.
Definition 5.7 (Face-splitting product). The alternative concept of the matrix product, which uses row-wise splitting of matrices with a given quantity of rows. This matrix operation was named the face-splitting product of matrices or the transposed Khatri-Rao product. This type of operation is based on row-by-row Kronecker products of two matrices.

Example: the Face-splitting product of the matrices $A \in \mathbb{R}^{3 \times 2}$ and $B \in \mathbb{R}^{3 \times 2}$ is computed below:

$$
\underbrace{\left[\begin{array}{ll}
a_{1,1} & a_{1,2} \\
a_{2,1} & a_{3,2} \\
a_{3,1} & a_{3,2}
\end{array}\right]}_{A} \cdot \underbrace{\left[\begin{array}{ll}
b_{1,1} & b_{1,2} \\
b_{2,1} & b_{2,2} \\
b_{3,1} & b_{3,2}
\end{array}\right]}_{B}=\underbrace{\left[\begin{array}{llll}
a_{1,1} b_{1,1} & a_{1,2} b_{1,1} & a_{1,1} b_{1,2} & a_{1,2} b_{1,2} \\
a_{2,1} b_{2,1} & a_{2,2} b_{2,1} & a_{2,1} b_{2,2} & a_{2,2} b_{2,2} \\
a_{3,1} b_{3,1} & a_{3,2} b_{3,1} & a_{3,1} b_{3,2} & a_{3,2} b_{3,2}
\end{array}\right]}_{A \bullet B}
$$

### 5.7.3 Well-known Lemmas

In this section, we provide the details on the Lemmas required for our proofs along with their proofs or the corresponding citations where the Lemmas can be found as well.

Lemma 5.3. (Federer, 2014) Let $g$, $h$ be two composable Lipschitz functions. Then $g \circ h$ is also $\operatorname{Lipschitz}$ with $\operatorname{Lip}(g \circ h) \leq \operatorname{Lip}(g) \operatorname{Lip}(h)$. Here and only here $\circ$ represents function composition.
Lemma 5.4. (Federer, 2014) Let $f: \mathscr{X} \subseteq \mathbb{R}^{n} \rightarrow \mathbb{R}^{m}$ be differentiable and Lipschitz continuous. Let $J_{f}(x)$ denote its total derivative (Jacobian) at $x$. Then $\operatorname{Lip}_{p}(f)=\sup _{x \in \mathscr{X}}\left\|J_{f}(x)\right\|_{p}$ where $\left\|J_{f}(x)\right\|_{p}$ is the induced operator norm on $J_{f}(x)$.
Lemma 5.5 (Hölder's inequality). (Cvetkovski, 2012) Let $(S, \Sigma, \mu)$ be a measure space and let $p, q \in[1, \infty]$ with $\frac{1}{p}+\frac{1}{q}=1$. Then, for all measurable real-valued functions $f$ and $g$ on $S$, it holds that:

$$
\|f g\|_{1} \leq\|f\|_{p}\|g\|_{q} .
$$

Lemma 5.6 (Mixed Product Property 1). (Slyusar, 1999) The following holds:

$$
\left(A_{1} B_{1}\right) \circ\left(A_{2} B_{2}\right)=\left(A_{1} \bullet A_{2}\right)\left(B_{1} * B_{2}\right)
$$

Lemma 5.7 (Mixed Product Property 2). The following holds:

$$
\circ_{i=1}^{N}\left(A_{i} B_{i}\right)=\bullet \bullet_{i=1}^{N}\left(A_{i}\right) *_{i=1}^{N}\left(B_{i}\right) .
$$

Proof. We prove this lemma by induction on $N$.
Base case $(N=1): A_{1} B_{1}=A_{1} B_{1}$.
Inductive step: Assume that the induction hypothesis holds for a particular $k$, i.e., the case $N=k$ holds. That can be expressed as:

$$
\begin{equation*}
\circ_{i=1}^{k}\left(A_{i} B_{i}\right)=\bullet \bullet_{i=1}^{k}\left(A_{i}\right) *_{i=1}^{k}\left(B_{i}\right) \tag{5.3}
\end{equation*}
$$

Then we will prove that it holds for $N=k+1$ :

$$
\begin{aligned}
& \circ_{i=1}^{k+1}\left(A_{i} B_{i}\right) \\
& =\left[\bullet_{i=1}^{k}\left(A_{i} B_{i}\right)\right] \circ\left(A_{k+1} B_{k+1}\right) \\
& =\left[\bullet{ }_{i=1}^{k}\left(A_{i}\right) *_{i=1}^{k}\left(B_{i}\right)\right] \circ\left(A_{k+1} B_{k+1}\right) \quad \text { use inductive hypothesis (eq. (5.3)) } \\
& =\left[\bullet{ }_{i=1}^{k}\left(A_{i}\right) \bullet A_{k+1}\right]\left[*_{i=1}^{k}\left(B_{i}\right) * B_{k+1}\right] \quad \text { lemma } 5.6 \text { [Mixed product property 1] } \\
& =\bullet_{i=1}^{k+1}\left(A_{i}\right) *_{i=1}^{k+1}\left(B_{i}\right)
\end{aligned}
$$

That is, the case $N=k+1$ also holds true, establishing the inductive step.

Lemma 5.8 (Massart Lemma. Lemma 26.8 in Shalev-Shwartz and Ben-David (2014a)). Let $\mathscr{A}$ $=\left\{a_{1}, \cdots, a_{N}\right\}$ be a finite set of vectors in $\mathbb{R}^{m}$. Define $\bar{a}=\frac{1}{N} \sum_{i=1}^{N} a_{i}$. Then:

$$
\mathscr{R}(\mathscr{A}) \leq \max _{a \in A}\|a-\bar{a}\| \frac{\sqrt{2 \log N}}{m}
$$

Definition 5.8 (Consistency of a matrix norm). A matrix norm is called consistent on $\mathbb{C}^{n, n}$, if

$$
\|A B\| \leq\|A\|\|B\|
$$

holds for $A, B \in \mathbb{C}^{n, n}$.
Lemma 5.9 (Consistency of the operator norm). (Lyche, 2020) The operator norm is consistent if the vector norm $\|\cdot\|_{\alpha}$ is defined for all $m \in \mathbb{N}$ and $\|\cdot\|_{\beta}=\|\cdot\|_{\alpha}$

Proof.

$$
\begin{aligned}
\|A B\| & =\max _{B x \neq 0} \frac{\|A B x\|_{\alpha}}{\|x\|_{\alpha}}=\max _{B x \neq 0} \frac{\|A B x\|_{\alpha}}{\|B x\|_{\alpha}} \frac{\|B x\|_{\alpha}}{\|x\|_{\alpha}} \\
& \leq \max _{y \neq 0} \frac{\|A y\|_{\alpha}}{\|y\|_{\alpha}} \max _{x \neq 0} \frac{\|B x\|_{\alpha}}{\|x\|_{\alpha}}=\|A\|\|B\| .
\end{aligned}
$$

Lemma 5.10. (Rao, 1970)

$$
(A C) *(B D)=(A \otimes B)(C * D) .
$$

### 5.8 Appendix: Details on polynomial networks

In this section, we provide further details on the two most prominent parametrizations proposed in Chrysos et al. (2020). This re-parametrization creates equivalent models, but enables us to absorb the bias terms into the input terms. Firstly, we provide the re-parametrization of the CCP model in subsection 5.8.1 and then we create the re-parametrization of the NCP model in subsection 5.8.2.

### 5.8.1 Re-parametrization of CCP model

The Coupled CP-Decomposition (CCP) model of PNs (Chrysos et al., 2020) leverages a coupled CP Tensor decomposition. A $k$-degree CCP model $f(\zeta)$ can be succinctly described by the following recursive relations:

$$
\begin{equation*}
y_{1}=V_{1} \zeta, \quad y_{n}=\left(V_{n} \zeta\right) \circ y_{n-1}+y_{n-1}, \quad f(\zeta)=Q y_{k}+\beta \tag{5.4}
\end{equation*}
$$

where $\zeta \in \mathbb{R}^{\delta}$ is the input data with $\delta \in \mathbb{N}, f(\zeta) \in \mathbb{R}^{o}$ is the output of the model and $V_{n} \in \mathbb{R}^{\mu \times \delta}$, $Q \in \mathbb{R}^{0 \times \mu}$ and $\beta \in \mathbb{R}^{o}$ are the learnable parameters, where $\mu \in \mathbb{N}$ is the hidden rank. In order to simplify the bias terms in the model, we will introduce a minor re-parametrization in lemma 5.11 that we will use to present our results in the subsequent sections.

Lemma 5.11. Let $z=\left[\zeta^{\top}, 1\right]^{\top} \in \mathbb{R}^{d}, x_{n}=\left[y_{n}^{\top}, 1\right]^{\top} \in \mathbb{R}^{m}, C=[Q, \beta] \in \mathbb{R}^{0 \times m}, d=\delta+1, m=\mu+1$. Define:

$$
U_{1}=\left[\begin{array}{cc}
V_{1} & 0 \\
0^{\top} & 1
\end{array}\right] \in \mathbb{R}^{m \times d}, \quad U_{i}=\left[\begin{array}{cc}
V_{i} & 1 \\
0^{\top} & 1
\end{array}\right] \in \mathbb{R}^{m \times d} \quad(i>1) .
$$

where the boldface numbers 0 and 1 denote all-zeros and all-ones column vectors of appropriate size, respectively. The CCP model in eq. (5.4) can be rewritten as $f(z)=C \circ_{i=1}^{k} U_{i} z$, which is the one used in eq. (ССР).

As a reminder before providing the proof, the core symbols for this proof are summarized in
table 5.6.
Table 5.6: Core symbols in the proof of lemma 5.11.

| Symbol | Dimensions | Definition |
| :---: | :---: | :---: |
| $\circ$ | - | Hadamard (element-wise) product. |
| $\zeta$ | $\mathbb{R}^{\delta}$ | Input of the polynomial expansion. |
| $f(\zeta)$ | $\mathbb{R}^{o}$ | Output of the polynomial expansion. |
| $k$ | $\mathbb{N}$ | Degree of polynomial expansion. |
| $m$ | $\mathbb{N}$ | Hidden rank of the expansion. |
| $V_{n}$ | $\mathbb{R}^{\mu \times \delta}$ | Learnable matrices of the expansion. |
| $Q$ | $\mathbb{R}^{o \times \mu}$ | Learnable matrix of the expansion. |
| $\beta$ | $\mathbb{R}^{o}$ | Bias of the expansion. |
| $z$ | $\mathbb{R}^{d}$ | Re-parametrization of the input. |
| $C$ | $\mathbb{R}^{o \times m}$ | $C=(Q, \beta)$. |

Proof. By definition, we have:

$$
\begin{aligned}
x_{1} & =\left[y_{1}^{\top}, 1\right]^{\top}=\left[\begin{array}{c}
y_{1} \\
1
\end{array}\right]=\left[\begin{array}{c}
V_{1} \zeta \\
1
\end{array}\right]=\left[\begin{array}{ll}
V_{1} & 0 \\
0^{\top} & 1
\end{array}\right]\left[\begin{array}{l}
\zeta \\
1
\end{array}\right]=\left[\begin{array}{cc}
V_{1} & 0 \\
0^{\top} & 1
\end{array}\right]\left[\zeta^{\top}, 1\right]^{\top}=U_{1} z . \\
x_{n} & =\left[y_{n}^{\top}, 1\right]^{\top}=\left[\begin{array}{c}
y_{n} \\
1
\end{array}\right]=\left[\begin{array}{c}
\left(V_{n} \zeta\right) \circ y_{n-1}+y_{n-1} \\
1
\end{array}\right]=\left[\begin{array}{c}
V_{n} \zeta+1 \\
1
\end{array}\right] \circ\left[\begin{array}{c}
y_{n-1} \\
1
\end{array}\right] \\
& =\left[\begin{array}{ll}
V_{n} & 1 \\
0^{\top} & 1
\end{array}\right]\left[\begin{array}{c}
\zeta \\
1
\end{array}\right] \circ\left[\begin{array}{c}
y_{n-1} \\
1
\end{array}\right]=\left[\begin{array}{ll}
V_{n} & 1 \\
0^{\top} & 1
\end{array}\right]\left[\zeta^{\top}, 1\right]^{\top} \circ\left[y_{n-1}^{\top}, 1\right]^{\top}=U_{n} z \circ x_{n-1} .
\end{aligned}
$$

Hence, it holds that:

$$
\begin{aligned}
& f(z)=Q y_{k}+\beta=(Q, \beta)\left[\begin{array}{c}
y_{k} \\
1
\end{array}\right]=C x_{k} \\
&=C U_{k} z \circ x_{k-1} \\
&=C U_{k} z \circ\left(U_{k-1} z\right) \circ x_{k-2} \\
&=\cdots \\
&=C U_{k} z \circ\left(U_{k-1} z\right) \circ \cdots \circ\left(U_{2} z\right) \circ x_{1} \\
&=C U_{k} z \circ\left(U_{k-1} z\right) \circ \cdots \circ\left(U_{2} z\right) \circ\left(U_{1} z\right) \\
&=C \circ k \\
& i=1 \\
&\left(U_{i} z\right) .
\end{aligned}
$$

### 5.8.2 Reparametrization of the NCP model

The nested coupled CP decomposition (NCP) model of PNs (Chrysos et al., 2020) leverages a joint hierarchical decomposition. A $k$-degree $N C P$ model $f(\zeta)$ is expressed with the following recursive relations:

$$
\begin{equation*}
y_{1}=\left(V_{1} \zeta\right) \circ\left(b_{1}\right), \quad y_{n}=\left(V_{n} \zeta\right) \circ\left(U_{n} y_{n-1}+b_{n}\right), \quad f(\zeta)=Q y_{k}+\beta \tag{5.5}
\end{equation*}
$$

where $\zeta \in \mathbb{R}^{\delta}$ is the input data with $\delta \in \mathbb{N}, f(\zeta) \in \mathbb{R}^{o}$ is the output of the model and $V_{n} \in \mathbb{R}^{\mu \times \delta}$, $b_{n} \in \mathbb{R}^{\mu}, U_{n} \in \mathbb{R}^{\mu \times \mu}, Q \in \mathbb{R}^{o \times \mu}$ and $\beta \in \mathbb{R}^{o}$ are the learnable parameters, where $\mu \in \mathbb{N}$ is the hidden rank. In order to simplify the bias terms in the model, we will introduce a minor re-parametrization in lemma 5.12 that we will use to present our results in the subsequent sections.

Lemma 5.12. Let $z=\left[\zeta^{\top}, 1\right]^{\top} \in \mathbb{R}^{d}, x_{n}=\left[y_{n}^{\top}, 1\right]^{\top} \in \mathbb{R}^{m}, C=[Q, \beta] \in \mathbb{R}^{o \times m}, d=\delta+1, m=\mu+1$. Let:

$$
s_{1}=\left[b_{1}^{\top}, 1\right]^{\top} \in \mathbb{R}^{m}, \quad S_{i}=\left[\begin{array}{cc}
U_{i} & b_{i} \\
0^{\top} & 1
\end{array}\right] \in \mathbb{R}^{m \times m}(i>1), \quad A_{i}=\left[\begin{array}{cc}
V_{i} & 0 \\
0^{\top} & 1
\end{array}\right] \in \mathbb{R}^{m \times d}
$$

where the boldface numbers 0 and 1 denote all-zeros and all-ones column vectors of appropriate size, respectively. The NCP model in eq. (5.5) can be rewritten as

$$
\begin{equation*}
x_{1}=\left(A_{1} z\right) \circ\left(s_{1}\right), \quad x_{n}=\left(A_{n} z\right) \circ\left(S_{n} x_{n-1}\right), \quad f(z)=C x_{k} \tag{5.6}
\end{equation*}
$$

In the aforementioned eq. (5.6), we have written $S_{n}$ even for $n=1$, when $s_{1}$ is technically a vector, but this is done for convenience only and does not change the end result.

### 5.9 Appendix: Result of the CCP model

### 5.9.1 Proof of theorem 5.1: Rademacher Complexity bound of CCP under $\ell_{\infty}$ norm

To facilitate the proof below, we include the related symbols in table 5.7. Below, to avoid cluttering the notation, we consider that the expectation is over $\sigma$ and omit the brackets as well.

Table 5.7: Core symbols for proof of theorem 5.1.

| Symbol | Dimensions | Definition |
| :---: | :---: | :---: |
| $\circ$ | - | Hadamard (element-wise) product. |
| $\bullet$ | - | Face-splitting product. |
| $*$ | - | Column-wise Khatri-Rao product. |
| $z$ | $\mathbb{R}^{d}$ | Input of the polynomial expansion. |
| $f(z)$ | $\mathbb{R}$ | Output of the polynomial expansion. |
| $k$ | $\mathbb{N}$ | Degree of polynomial expansion. |
| $m$ | $\mathbb{N}$ | Hidden rank of the expansion. |
| $U_{i}$ | $\mathbb{R}^{m \times d}$ | Learnable matrices. |
| $c$ | $\mathbb{R}^{1 \times m}$ | Learnable matrix. |
| $\mu$ | $\mathbb{R}$ | $\\|c\\|_{1} \leq \mu$. |
| $\lambda$ | $\mathbb{R}$ | $\left\\|\bullet k . k\left(U_{i}\right)\right\\|_{\infty} \leq \lambda$. |

Proof.

$$
\begin{align*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{CCP}}^{k}\right) & =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n}\left(\sigma_{j}\left\langle c, \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right\rangle\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\left\langle c, \sum_{j=1}^{n}\left[\sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right]\right\rangle \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j} \stackrel{1}{i=1}_{k}\left(U_{i} z_{j}\right)\right]\right\|_{\infty} \quad \text { (lemma 5.5) } \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j} \bullet_{i=1}^{k}\left(U_{i}\right) *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty} \quad \text { (lemma 5.7) }  \tag{5.7}\\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\bullet_{i=1}^{k}\left(U_{i}\right) \sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty} \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty}\left\|\bullet \bullet_{i=1}^{k}\left(U_{i}\right)\right\|_{\infty} \\
& \leq \frac{\mu \lambda}{n} \mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty} .
\end{align*}
$$

Next, we compute the bound of $\mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty}$.
Let $Z_{j}=*_{i=1}^{k}\left(z_{j}\right) \in \mathbb{R}^{d^{k}}$. For each $l \in\left[d^{k}\right]$, let $v_{l}=\left(Z_{1}^{l}, \ldots, Z_{n}^{l}\right) \in \mathbb{R}^{n}$. Note that $\left\|v_{l}\right\|_{2} \leq$ $\sqrt{n} \max _{j}\left\|Z_{j}\right\|_{\infty}$. Let $V=\left\{v_{1}, \ldots, v_{d^{k}}\right\}$. Then, it is true that:

$$
\begin{equation*}
\mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty}=\mathbb{E}\left\|\sum_{j=1}^{n} \sigma_{j} Z_{j}\right\|_{\infty}=\mathbb{E} \max _{l=1}^{d^{k}}\left|\sum_{j=1}^{n} \sigma_{j}\left(\nu_{l}\right)_{j}\right|=n \mathscr{R}(V) . \tag{5.8}
\end{equation*}
$$

Using lemma 5.8 [Massart Lemma] we have that:

$$
\begin{equation*}
\mathscr{R}(V) \leq 2 \max _{j}\left\|Z_{j}\right\|_{\infty} \sqrt{2 \log \left(d^{k}\right) / n} \tag{5.9}
\end{equation*}
$$

Then, it holds that:

$$
\begin{align*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{CCP}}^{k}\right) & =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right) \\
& \leq \frac{\mu \lambda}{n} \mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{\infty} \\
& =\frac{\mu \lambda}{n} n \mathscr{R}(V) \\
& \leq 2 \mu \lambda \max _{j}\left\|Z_{j}\right\|_{\infty} \sqrt{2 \log \left(d^{k}\right) / n}  \tag{5.10}\\
& =2 \mu \lambda \max _{j}\left\|*_{i=1}^{k}\left(z_{j}\right)\right\|_{\infty} \sqrt{2 \log \left(d^{k}\right) / n} \\
& \leq 2 \mu \lambda\left(\max _{j}\left\|z_{j}\right\|_{\infty}\right)^{k} \sqrt{2 \log \left(d^{k}\right) / n} \\
& \leq 2 \mu \lambda \sqrt{2 k \log (d) / n} .
\end{align*}
$$

### 5.9.2 Proof of lemma 5.1

Table 5.8: Core symbols in the proof of lemma 5.1.

| Symbol | Dimensions | Definition |
| :---: | :---: | :---: |
| $\otimes$ | - | Kronecker product. |
| $\bullet$ | - | Face-splitting product. |
| $z$ | $\mathbb{R}^{d}$ | Input of the polynomial expansion. |
| $f(z)$ | $\mathbb{R}$ | Output of the polynomial expansion. |
| $k$ | $\mathbb{N}$ | Degree of polynomial expansion. |
| $m$ | $\mathbb{N}$ | Hidden rank of the expansion. |
| $U_{i}$ | $\mathbb{R}^{m \times d}$ | Learnable matrices. |
| $U_{i}^{j}$ | $\mathbb{R}^{d}$ | $j^{\text {th }}$ row of $U_{i}$. |
| $\lambda_{i}$ | $\mathbb{R}$ | $\left\\|U_{i}\right\\|_{\infty} \leq \lambda_{i}$ for $i=1,2, \ldots, k$. |

Proof.

$$
\begin{aligned}
& \left\|\bullet{ }_{i=1}^{k}\left(U_{i}\right)\right\|_{\infty}=\max _{j=1}^{m}\left\|\left[\bullet \bullet_{i=1}^{k}\left(U_{i}\right)\right]^{j}\right\|_{1} \\
& =\max _{j=1}^{m}\left\|\otimes_{i=1}^{k}\left[U_{i}^{j}\right]\right\|_{1} \quad \text { Definition of Face-splitting product } \\
& =\max _{j=1}\left[\prod_{i=1}^{k}\left\|U_{i}^{j}\right\|_{1}\right] \quad \text { Multiplicativity of absolute value } \\
& \leq \prod_{i=1}^{k}\left[\max _{j=1}^{m}\left\|U_{i}^{j}\right\|_{1}\right] \\
& =\prod_{i=1}^{k}\left\|U_{i}\right\|_{\infty} .
\end{aligned}
$$

### 5.9.3 Rademacher Complexity bound under $\ell_{2}$ norm

Theorem 5.6. Let $Z=\left\{z_{1}, \ldots, z_{n}\right\} \subseteq \mathbb{R}^{d}$ and suppose that $\left\|z_{j}\right\|_{\infty} \leq 1$ for all $j=1, \ldots, n$. Let

$$
\mathscr{F}_{C C P}^{k}:=\left\{f(z)=\left\langle c, \circ_{i=1}^{k} U_{i} z\right\rangle:\|c\|_{2} \leq \mu,\left\|\bullet \frac{k}{i=1} U_{i}\right\|_{2} \leq \lambda\right\} .
$$

The Empirical Rademacher Complexity of $C C P_{k}$ ( $k$-degree CCP polynomials) with respect to $Z$ is bounded as:

$$
\mathscr{R}_{Z}\left(\mathscr{F}_{C C P}^{k}\right) \leq \frac{\mu \lambda}{\sqrt{n}}
$$

To facilitate the proof below, we include the related symbols in table 5.9. Below, to avoid cluttering the notation, we consider that the expectation is over $\sigma$ and omit the brackets as well.

Table 5.9: Core symbols for proof of theorem 5.6.
$\left.\begin{array}{ccc}\hline \text { Symbol } & \text { Dimensions } & \text { Definition } \\ \hline \circ & - & \text { Hadamard (element-wise) product. } \\ \bullet & - & \text { Face-splitting product. } \\ * & - & \text { Column-wise Khatri-Rao product. } \\ z & \mathbb{R}^{d} & \text { Input of the polynomial expansion. } \\ f(z) & \mathbb{R} & \text { Output of the polynomial expansion. } \\ k & \mathbb{N} & \text { Degree of polynomial expansion. } \\ m & \mathbb{N} & \text { Hidden rank of the expansion. } \\ U_{i} & \mathbb{R}^{m \times d} & \text { Learnable matrices. } \\ c & \mathbb{R}^{1 \times m} & \text { Learnable matrix. } \\ \hline \mu & \mathbb{R} & \|c\|_{2} \leq \mu . \\ \lambda & \mathbb{R} & \| \bullet k \\ i=1\end{array} U_{i}\right) \|_{2} \leq \lambda$.

Proof.

$$
\begin{align*}
& \mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{CCP}}^{k}\right)=\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CP}}^{k}} \frac{1}{n} \sum_{j=1}^{n}\left(\sigma_{j}\left\langle c, \mathrm{o}_{i=1}^{k}\left(U_{i} z_{j}\right)\right\rangle\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\text {CCP }}^{k}} \frac{1}{n}\left\langle c, \sum_{j=1}^{n}\left[\sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right]\right\rangle \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\text {CCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j} \circ_{i=1}^{k}\left(U_{i} z_{j}\right)\right]\right\|_{2} \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\text {CCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j} \bullet_{i=1}^{k}\left(U_{i}\right) * *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2} \quad \text { (lemma 5.7) } \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\text {CCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\bullet{ }_{i=1}^{k}\left(U_{i}\right) \sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2} \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\text {CCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2}\left\|\bullet{ }_{i=1}^{k}\left(U_{i}\right)\right\|_{2} . \\
& \mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2}=\mathbb{E} \sqrt{\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2}^{2}} \\
& \leq \sqrt{\mathbb{E}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2}^{2}} \quad \text { Jensen's inequality } \\
& =\sqrt{\mathbb{E} \sum_{s, j}^{n}\left[\sigma_{s} \sigma_{j}\left\langle *_{i=1}^{k}\left(z_{s}\right), *_{i=1}^{k}\left(z_{j}\right)\right\rangle\right]}  \tag{5.11}\\
& =\sqrt{\sum_{j=1}^{n}\left[\left\|*_{i=1}^{k}\left(z_{j}\right)\right\|_{2}^{2}\right]} \\
& =\sqrt{\sum_{j=1}^{n}\left(\prod_{i=1}^{k}\left\|z_{j}\right\|_{2}^{2}\right)} \\
& \leq \sqrt{n} \text {. }
\end{align*}
$$

So:

$$
\begin{aligned}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{CCP}}^{k}\right) & \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right]\right\|_{2}\left\|\cdot \bullet_{i=1}^{k}\left(U_{i}\right)\right\|_{2} \\
& \leq \frac{\sup _{f \in \mathscr{F}_{\mathrm{CCP}}^{k}}\|c\|_{2}\left\|\cdot{ }_{i=1}^{k}\left(U_{i}\right)\right\|_{2}}{\sqrt{n}} \\
& \leq \frac{\mu \lambda}{\sqrt{n}} .
\end{aligned}
$$

### 5.9.4 Lipschitz constant bound of the CCP model

We will first prove a more general result about the $\ell_{p}$-Lipschitz constant of the CCP model.

Theorem 5.7. The Lipschitz constant (with respect to the $\ell_{p}$-norm) of the function defined in eq. (CCP), restricted to the set $\left\{z \in \mathbb{R}^{d}:\|z\|_{p} \leq 1\right\}$ is bounded as:

$$
\operatorname{Lip}_{p}(f) \leq k\|C\|_{p} \prod_{i=1}^{k}\left\|U_{i}\right\|_{p}
$$

Proof. Let $g(x)=C x$ and $h(z)=\circ_{i=1}^{k}\left(U_{i} z\right)$. Then it holds that $f(z)=g(h(z))$. By lemma 5.3, we have: $\operatorname{Lip}_{p}(f) \leq \operatorname{Lip}_{p}(g) \operatorname{Lip}_{p}(h)$. We will compute an upper bound of each function individually.

Let us first consider the function $g(x)=C x$. By lemma 5.4, because $g$ is a linear map represented by a matrix $C$, its Jacobian is $J_{g}(x)=C$. So:

$$
\operatorname{Lip}_{p}(g)=\|C\|_{p}:=\sup _{\|x\|_{p}=1}\|C x\|_{p}
$$

where $\|C\|_{p}$ is the operator norm on matrices induced by the vector $p$-norm.
Now, let us consider the function $h(z)=0_{i=1}^{k} U_{i} z$. Its Jacobian is given by:

$$
\frac{d h}{d z}=\sum_{i=1}^{k} \operatorname{diag}\left(॰_{j \neq i} U_{j} z\right) U_{i}
$$

## Chapter 5. Controlling the Complexity and Lipschitz Constant Improves Polynomial Nets

Using lemma 5.4 we have:

$$
\begin{array}{rlr}
\operatorname{Lip}_{p}(h) & \leq \sup _{z:\| \| \|_{p} \leq 1}\left\|\sum_{i=1}^{k}\left[\operatorname{diag}\left(\circ_{j \neq i}\left(U_{j} z\right)\right) U_{i}\right]\right\|_{p} & \\
& \leq \sup _{z:\|z\|_{p} \leq 1} \sum_{i=1}^{k}\left\|\operatorname{diag}\left(\circ_{j \neq i}\left(U_{j} z\right)\right) U_{i}\right\|_{p} & \text { Triangle inequality } \\
& \leq \sup _{z:\|z\|_{p} \leq 1} \sum_{i=1}^{k} \| \operatorname{diag}\left(o_{j \neq i}\left(U_{j} z\right)\left\|_{p}\right\| U_{i} \|_{p} \quad\right. \text { lemma 5.9 [consistency] } \\
& \leq \sup _{z:\|z\|_{p} \leq 1} \sum_{i=1}^{k}\left\|o_{j \neq i}\left(U_{j} z\right)\right\|_{p}\left\|U_{i}\right\|_{p} & \\
& \leq \sup _{z:\| \| \|_{p} \leq 1} \sum_{i=1}^{k} \prod_{j \neq i}\left(\left\|U_{j} z\right\|_{p}\right)\left\|U_{i}\right\|_{p} & \\
& \leq \sup _{z:\|z\|_{p} \leq 1} \sum_{i=1}^{k} \prod_{j \neq i}\left(\left\|U_{j}\right\|_{p}\|z\|_{p}\right)\left\|U_{i}\right\|_{p} & \\
& \leq \sup _{z:\|z\|_{p} \leq 1} \sum_{i=1}^{k} \prod_{j=1}^{k}\left(\left\|U_{j}\right\|_{p}\right) \\
& =k \prod_{j=1}^{k}\left\|U_{j}\right\|_{p}
\end{array}
$$

So:

$$
\begin{aligned}
\operatorname{Lip}_{p}\left(\mathscr{F}_{L}\right) & \leq \operatorname{Lip}_{p}(g) \operatorname{Lip}_{p}(h) \\
& \leq k\|C\|_{p} \prod_{i=1}^{k}\left\|U_{i}\right\|_{p} .
\end{aligned}
$$

## Proof of theorem 5.2

Proof. This is a particular case of theorem 5.7 when $p=\infty$.

### 5.10 Appendix: Result of the NCP model

### 5.10.1 Proof of Theorem 5.3: Rademacher Complexity of NCP under $\ell_{\infty}$ norm

 Proof.$$
\begin{align*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{NCP}}^{k}\right) & =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n}\left(\sigma_{j}\left\langle c, x_{k}\left(z_{j}\right)\right\rangle\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\left\langle c, \sum_{j=1}^{n}\left[\sigma_{j} x_{k}\left(z_{j}\right)\right]\right\rangle \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j} x_{k}\left(z_{j}\right)\right]\right\|_{\infty}  \tag{lemma5.5}\\
& =\underset{f \in \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j}\left(\left(A_{k} z_{j}\right) \circ\left(S_{k} x_{k-1}\left(z_{j}\right)\right)\right)\right]\right\|_{\infty}}{ } \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\sum_{j=1}^{n}\left[\sigma_{j}\left(\left(A_{k} \bullet S_{k}\right)\left(z_{j} * x_{k-1}\left(z_{j}\right)\right)\right)\right]\right\|_{\infty} \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\left(A_{k} \bullet S_{k}\right) \sum_{j=1}^{n}\left[\sigma_{j}\left(z_{j} * x_{k-1}\left(z_{j}\right)\right)\right]\right\|_{\infty} . \tag{5.12}
\end{align*}
$$

Now, because of the recursive definition of the eq. (NCP), we obtain:

$$
\begin{align*}
\sum_{j=1}^{n} \sigma_{j}\left(z_{j} * x_{k-1}\left(z_{j}\right)\right) & =\sum_{j=1}^{n} \sigma_{j}\left(z_{j} *\left(A_{k-1} z_{j}\right) \circ\left(S_{k-1} x_{k-2}\left(z_{j}\right)\right)\right) \\
& =\sum_{j=1}^{n} \sigma_{j}\left(z_{j} *\left(\left(A_{k-1} \bullet S_{k-1}\right)\left(z_{j} * x_{k-2}\left(z_{j}\right)\right)\right)\right) \quad \text { (lemma 5.7) } \\
& =\sum_{j=1}^{n} \sigma_{j}\left(I \otimes\left(A_{k-1} \bullet S_{k-1}\right)\right)\left(z_{j} *\left(z_{j} * x_{k-2}\left(z_{j}\right)\right)\right) \quad \text { (lemma 5.10) }  \tag{5.13}\\
& =I \otimes\left(A_{k-1} \bullet S_{k-1}\right) \sum_{j=1}^{n}\left[\sigma_{j}\left(z_{j} *\left(z_{j} * x_{k-2}\left(z_{j}\right)\right)\right) .\right.
\end{align*}
$$

recursively applying this argument we have:

$$
\begin{equation*}
\sum_{j=1}^{n} \sigma_{j}\left(z_{j} * x_{k-1}\left(z_{j}\right)\right)=\left(\prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right) \sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right) . \tag{5.14}
\end{equation*}
$$

Combining the two previous equations (eqs. (5.13) and (5.14)) inside eq. (5.12) we finally obtain

$$
\begin{align*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{NCP}}^{k}\right) & \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\left(A_{k} \bullet S_{k}\right)\left(\prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right) \sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right\|_{\infty} \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{1}\left\|\left(A_{k} \bullet S_{k}\right)\left(\prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right)\right\|_{\infty}\left\|\sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right\|_{\infty} \\
& \leq \frac{\mu \lambda}{n} \mathbb{E}\left\|\sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right\|_{\infty} \\
& =\frac{\mu \lambda}{n} n \mathscr{R}(V)=\mu \lambda \mathscr{R}(V) . \tag{5.8}
\end{align*}
$$

following the same arguments as in eq. (5.10) it follows that:

$$
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{NCP}}^{k}\right) \leq 2 \mu \lambda \sqrt{\frac{2 k \log (d)}{n}}
$$

### 5.10.2 Proof of lemma 5.2

Proof.

$$
\begin{array}{rlrl}
\left\|\left(A_{k} \bullet S_{k}\right) \prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right\|_{\infty} & \leq\left\|A_{k} \bullet S_{k}\right\|_{\infty} \prod_{i=1}^{k-1}\left\|I \otimes A_{i} \bullet S_{i}\right\|_{\infty} & \quad \text { (lemma 5.9) } \\
& =\prod_{i=1}^{k}\left\|A_{i} \bullet S_{i}\right\|_{\infty} \\
& \leq \prod_{i=1}^{k}\left\|A_{i}\right\|_{\infty}\left\|S_{i}\right\|_{\infty} & \quad \text { (lemma 5.1) }
\end{array}
$$

### 5.10.3 Rademacher Complexity under $\ell_{2}$ norm

Theorem 5.8. Let $Z=\left\{z_{1}, \ldots, z_{n}\right\} \subseteq \mathbb{R}^{d}$ and suppose that $\left\|z_{j}\right\|_{\infty} \leq 1$ for all $j=1, \ldots, n$. Define the matrix $\Phi\left(A_{1}, S_{1}, \ldots, A_{n}, S_{n}\right):=\left(A_{k} \bullet S_{k}\right) \prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}$. Consider the class offunctions:

$$
\mathscr{F}_{N C P}^{k}:=\left\{f(z) \text { as in }(\mathrm{NCP}):\|C\|_{2} \leq \mu,\left\|\Phi\left(A_{1}, S_{1}, \ldots, A_{k}, S_{k}\right)\right\|_{2} \leq \lambda\right\}
$$

where $C \in \mathbb{R}^{1 \times m}$ (single output case). The Empirical Rademacher Complexity of $N C P_{k}(k$-degree NCP polynomials) with respect to $Z$ is bounded as:

$$
\mathscr{R}_{Z}\left(\mathscr{F}_{N C P}^{k}\right) \leq \frac{\mu \lambda}{\sqrt{n}} .
$$

Proof.

$$
\begin{align*}
\mathscr{R}_{Z}\left(\mathscr{F}_{\mathrm{NCP}}^{k}\right) & =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n} \sigma_{j} f\left(z_{j}\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n} \sum_{j=1}^{n}\left(\sigma_{j}\left\langle c, x_{k}\left(z_{j}\right)\right\rangle\right) \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\left\langle c, \sum_{j=1}^{n}\left[\sigma_{j} x_{k}\left(z_{j}\right)\right]\right\rangle \\
& \leq \mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j} x_{k}\left(z_{j}\right)\right]\right\|_{2}  \tag{lemma5.5}\\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\text {NCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j}\left(\left(A_{k} z_{j}\right) \circ\left(S_{k} x_{k-1}\left(z_{j}\right)\right)\right)\right]\right\|_{2} \\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{2}\left\|\sum_{j=1}^{n}\left[\sigma_{j}\left(\left(A_{k} \bullet S_{k}\right)\left(z_{j} * x_{k-1}\left(z_{j}\right)\right)\right)\right]\right\|_{2}  \tag{lemma5.7}\\
& =\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{N}}^{k}} \frac{1}{n}\|c\|_{2}\left\|\left(A_{k} \bullet S_{k}\right) \sum_{j=1}^{n}\left[\sigma_{j}\left(z_{j} * x_{k-1}\left(z_{j}\right)\right)\right]\right\|_{2} \\
& \left.=\mathbb{E} \sup _{f \in \mathscr{F}_{\mathrm{NCP}}^{k}} \frac{1}{n}\|c\|_{2} \|\left(A_{k} \bullet S_{k}\right)\left(\prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right) \sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right] \|_{2}  \tag{5.14}\\
& \left.\leq \mathbb{E} \sup _{f \in \mathscr{F}_{\text {NCP }}^{k}} \frac{1}{n}\|c\|_{2}\left\|\left(A_{k} \bullet S_{k}\right)\left(\prod_{i=1}^{k-1} I \otimes A_{i} \bullet S_{i}\right)\right\|\| \|_{2}^{n} \sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right] \|_{2} \\
& \leq \frac{\mu \lambda}{n} \mathbb{E}\left\|\sum_{j=1}^{n} \sigma_{j} *_{i=1}^{k}\left(z_{j}\right)\right\| \|_{2} \\
& \leq \frac{\mu \lambda}{\sqrt{n}} .
\end{align*}
$$

### 5.10.4 Lipschitz constant bound of the NCP model

Theorem 5.9. Let $\mathscr{F}_{L}$ be the class offunctions defined as

$$
\begin{array}{r}
\mathscr{F}_{L}:=\left\{x_{1}=\left(A_{1} z\right) \circ\left(S_{1}\right), x_{n}=\left(A_{n} z\right) \circ\left(S_{n} x_{n-1}\right), f(z)=C x_{k}:\right. \\
\left.\|C\|_{p} \leq \mu,\left\|A_{i}\right\|_{p} \leq \lambda_{i},\left\|S_{i}\right\|_{p} \leq \rho_{i},\|z\|_{p} \leq 1\right\} .
\end{array}
$$

The Lipschitz Constant of $\mathscr{F}_{\mathscr{L}}$ (k-degree NCP polynomial) under $\ell_{p}$ norm restrictions is bounded as:

$$
\operatorname{Lip}_{p}\left(\mathscr{F}_{L}\right) \leq k \mu \prod_{i=1}^{k}\left(\lambda_{i} \rho_{i}\right)
$$

Proof. Let $g(x)=C x, h(z)=\left(A_{n} z\right) \circ\left(S_{n} x_{n-1}(z)\right)$. Then it holds that $f(z)=g(h(z))$.
By lemma 5.3, we have: $\operatorname{Lip}(f) \leq \operatorname{Lip}(g) \operatorname{Lip}(h)$. This enables us to compute an upper bound of each function (i.e., $g, h$ ) individually.

Let us first consider the function $g(x)=C x$. By lemma 5.4, because $g$ is a linear map represented by a matrix $C$, its Jacobian is $J_{g}(x)=C$. So:

$$
\operatorname{Lip}_{p}(g)=\|C\|_{p}:=\sup _{\|x\|_{p}=1}\|C x\|_{p}= \begin{cases}\sigma_{\max }(C) & \text { if } p=2 \\ \max _{i} \sum_{j}\left|C_{(i, j)}\right| & \text { if } p=\infty\end{cases}
$$

where $\|C\|_{p}$ is the operator norm on matrices induced by the vector p-norm, and $\sigma_{\max }(C)$ is the largest singular value of $C$.

Now, let us consider the function $x_{n}(z)=h(z)=\left(A_{n} z\right) \circ\left(S_{n} x_{n-1}(z)\right)$. Its Jacobian is given by:

$$
J_{x_{n}}=\operatorname{diag}\left(A_{n} z\right) S_{n} J_{x_{n-1}}+\operatorname{diag}\left(S_{n} x_{n-1}\right) A_{n}, \quad J_{x_{1}}=\operatorname{diag}\left(S_{1}\right) A_{1}
$$

$$
\begin{aligned}
\operatorname{Lip}_{p}(h) & =\sup _{z:\|z\|_{p} \leq 1}\left\|J_{x_{n}}\right\|_{p} \\
& =\sup _{z:\|z\|_{p} \leq 1}\left\|\operatorname{diag}\left(A_{n} z\right) S_{n} J_{x_{n-1}}+\operatorname{diag}\left(S_{n} x_{n-1}\right) A_{n}\right\|_{p} \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|\operatorname{diag}\left(A_{n} z\right) S_{n} J_{x_{n-1}}\right\|_{p}+\left\|\operatorname{diag}\left(S_{n} x_{n-1}\right) A_{n}\right\|_{p} \quad \text { (Triangle inequality) } \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|\operatorname{diag}\left(A_{n} z\right)\right\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|\operatorname{diag}\left(S_{n} x_{n-1}\right)\right\|_{p}\left\|A_{n}\right\|_{p} \quad \text { (lemma 5.9) } \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n} z\right\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|S_{n} x_{n-1}\right\|_{p}\left\|A_{n}\right\|_{p} \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|S_{n}\right\|_{p}\left\|x_{n-1}\right\|_{p}\left\|A_{n}\right\|_{p} \\
& =\sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|S_{n}\right\|_{p}\left\|\left(A_{n-1} z\right) \circ\left(S_{n-1} x_{n-2}\right)\right\|_{p}\left\|A_{n}\right\|_{p} \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|S_{n}\right\|_{p}\left\|A_{n-1} z\right\|_{p}\left\|S_{n-1} x_{n-2}\right\|_{p}\left\|A_{n}\right\|_{p} \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left\|J_{x_{n-1}}\right\|_{p}+\left\|S_{n}\right\|_{p}\left\|A_{n-1}\right\|_{p}\|z\|_{p}\left\|S_{n-1}\right\|_{p}\left\|x_{n-2}\right\|_{p}\left\|A_{n}\right\|_{p} \\
& =\sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left(\left\|J_{x_{n-1}}\right\|_{p}+\left\|A_{n-1}\right\|_{p}\left\|S_{n-1}\right\|_{p}\left\|x_{n-2}\right\|_{p}\right) \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left(\left\|J_{x_{n-1}}\right\|_{p}+\prod_{i=1}^{n-1}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right)\|z\|_{p}^{n-2}\right) .
\end{aligned}
$$

Then we proof the result by induction.
Inductive hypothesis:

$$
\sup _{z:\|z\|_{p} \leq 1}\left\|J_{x_{n}}\right\|_{p} \leq n \prod_{i=1}^{n}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right) .
$$

Case $k=1$ :

$$
\begin{aligned}
\operatorname{Lip}_{p}(h) & =\sup _{z:\| \|_{p} \leq 1}\left\|J_{x_{1}}\right\|_{p} \\
& =\left\|\operatorname{diag}\left(S_{1}\right) A_{1}\right\|_{p} \\
& \leq\left\|\operatorname{diag}\left(S_{1}\right)\right\|_{p}\left\|A_{1}\right\|_{p} \\
& \leq\left\|S_{1}\right\|_{p}\left\|A_{1}\right\|_{p} .
\end{aligned}
$$

Case $k=n$ :

$$
\begin{aligned}
\operatorname{Lip}_{p}(h) & =\sup _{z:\|z\|_{p} \leq 1}\left\|J_{x_{n}}\right\|_{p} \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left(\left\|J_{x_{n-1}}\right\|_{p}+\prod_{i=1}^{n-1}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right)\|z\|_{p}^{n-2}\right) \\
& \leq \sup _{z:\|z\|_{p} \leq 1}\left\|A_{n}\right\|_{p}\|z\|_{p}\left\|S_{n}\right\|_{p}\left((n-1) \prod_{i=1}^{n-1}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right)+\prod_{i=1}^{n-1}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right)\|z\|_{p}^{n-2}\right) \\
& \leq n \prod_{i=1}^{n}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right) .
\end{aligned}
$$

So:

$$
\begin{aligned}
\operatorname{Lip}_{p}\left(\mathscr{F}_{L}\right) & \leq \operatorname{Lip}_{p}(g) \operatorname{Lip}_{p}(h) \\
& \leq k\|C\|_{p} \prod_{i=1}^{k}\left(\left\|S_{i}\right\|_{p}\left\|A_{i}\right\|_{p}\right) .
\end{aligned}
$$

proof of theorem 5.4
Proof. This is particular case of theorem 5.9 with $p=\infty$.

### 5.11 Appendix: Relationship between a Convolutional layer and a Fully Connected layer

In this section we discuss various cases of input/output types depending on the dimensionality of the input tensor and the output tensor. We also provide the proof of theorem 5.5.

Theorem 5.10. Let $A \in \mathbb{R}^{n}$. Let $K \in \mathbb{R}^{h}$ be a 1-D convolutional kernel. For simplicity, we assume $h$ is odd and $h \leq n$. Let $B \in \mathbb{R}^{n}, B=K \star A$ be the output of the convolution. Let $U$ be the convolutional operator i.e., the linear operator (matrix) $U \in \mathbb{R}^{n \times n}$ such that $B=K \star A=U A$. It holds that $\|U\|_{\infty}=\|K\|_{1}$.

Theorem 5.11. Let $A \in \mathbb{R}^{n \times m}$, and let $K \in \mathbb{R}^{h \times h}$ be a $2-D$ convolutional kernel. For simplicity assume $h$ is odd number and $h \leq \min (n, m)$. Let $B \in \mathbb{R}^{n \times m}, B=K \star A$ be the output of the convolution. Let $U$ be the convolutional operator i.e., the linear operator (matrix) $U \in \mathbb{R}^{n m \times n m}$ such that $\operatorname{vec}(B)=U \operatorname{vec}(A)$. It holds that $\|U\|_{\infty}=\|\operatorname{vec}(K)\|_{1}$.

### 5.11.1 Proof of theorem 5.10

Proof. From, $B=K \star A=U A$ we can obtain the following:

$$
\left(\begin{array}{cccc}
u_{1,1} & u_{1,2} & \cdots & u_{1, n} \\
u_{2,1} & u_{2,2} & \cdots & u_{2, n} \\
\vdots & \vdots & \ddots & \vdots \\
u_{n, 1} & u_{n, 2} & \cdots & u_{n, n}
\end{array}\right)\left(A^{1}, \cdots, A^{n}\right)^{\top}=\left(K^{1}, \cdots, K^{h}\right)^{\top} \star\left(A^{1}, \cdots, A^{n}\right)^{\top}
$$

We observe that:

$$
u_{i, j}= \begin{cases}K^{\frac{h+1}{2}+j-i} & \text { if }|i-j| \leq \frac{h-1}{2} \\ 0 & \text { if }|i-j|>\frac{h-1}{2}\end{cases}
$$

Then, it holds that:

$$
\|U\|_{\infty}=\max _{i=1}^{n} \sum_{j=1}^{n}\left|u_{i, j}\right| \leq \max _{i=1}^{n} \sum_{j=1}^{n}\left|K^{j}\right|=\|K\|_{1}
$$

### 5.11.2 Proof of theorem 5.11

Proof. We partition $U$ into $n \times n$ partition matrices of shape $m \times m$. Then the $(i, j)^{\text {th }}$ partition matrix $U_{(i, j)}$ describes the relationship between the $B^{i}$ and the $A^{j}$. So, $U_{(i, j)}$ is also similar to the Toeplitz matrix in the previous result.

$$
U_{(i, j)}= \begin{cases}M_{\frac{h+1}{2}+j-i} & \text { if }|i-j| \leq \frac{h-1}{2} \\ 0 & \text { if }|i-j|>\frac{h-1}{2}\end{cases}
$$

Meanwhile, the matrix $M$ satisfies:

$$
m_{i(s, l)}= \begin{cases}k_{\left(i, \frac{h+1}{2}+l-s\right)} & \text { if }|s-l| \leq \frac{h-1}{2} \\ 0 & \text { if }|s-l|>\frac{h-1}{2}\end{cases}
$$

Then, we have the following:

$$
\|U\|_{\infty}=\max _{i=1}^{n \times m} \sum_{j=1}^{n \times m}\left|u_{i, j}\right| \leq \max _{i=1}^{n} \sum_{j=1}^{n}\left\|U_{i, j}\right\|_{\infty} \leq \max _{i=1}^{n} \sum_{j=1}^{h}\left\|K^{j}\right\|_{1}=\sum_{i=1}^{h}\left\|K^{i}\right\|_{1}=\|\operatorname{vec}(K)\|_{1}
$$

In addition, by $h \leq n, h \leq m$ we have:

$$
\begin{equation*}
\left\|U^{\frac{h+1}{2}+M\left(\frac{h-1}{2}\right)}\right\|_{1}=\sum_{i=1}^{h}\left\|K^{i}\right\|_{1} . \tag{5.15}
\end{equation*}
$$

Then, it holds that $\|U\|_{\infty}=\sum_{i=1}^{h}\left\|K^{i}\right\|_{1}$.

### 5.11.3 Proof of theorem 5.5

Proof. We partition $U$ into $o \times r$ partition matrices of shape $n m \times n m$. Then the $(i, j)^{\text {th }}$ partition of the matrix $U_{(i, j)}$ describes the relationship between the $i^{\text {th }}$ channel of $B$ and the $j^{\text {th }}$ channel of $A$. Then, the following holds: $\left\|U_{(i, j)}\right\|_{\infty}=\sum_{i=1}^{h}\left\|K_{i j}^{i}\right\|_{1}$, where $K_{i j}$ means the twodimensional tensor obtained by the third dimension of $K$ takes $j$ and the fourth dimension of $K$ takes $i$.

$$
\begin{aligned}
\|U\|_{\infty} & ={\underset{i=1}{n \times m \times o} \sum_{j=1}^{n \times m \times r} \sum_{j=1}^{n a n}\left|u_{(i, j)}\right|}=\frac{\max _{l=0}^{o-1} \max _{i=1}^{n \times m} \sum_{s=0}^{r-1} \sum_{j=1}^{n \times m}\left|u_{(i+n m l, j+n m s)}\right|}{} \\
& \leq \max _{l=0}^{o-1} \sum_{s=0}^{r-1}\left(\max _{i=1}^{n \times m} \sum_{j=1}^{n \times m}\left|u_{(i+n m l, j+n m s)}\right|\right) \\
& =\max _{l=0}^{o-1} \sum_{s=0}^{r-1}\left\|U_{(l+1, s+1)}\right\|_{\infty} \\
& =\max _{l=1}^{o} \sum_{s=1}^{r}\left\|U_{(l, s)}\right\|_{\infty} \\
& =\max _{l=1}^{o} \sum_{s=1}^{r} \sum_{i=1}^{n}\left\|K_{l s}^{i}\right\|_{1} \\
& \leq \max _{l=1}^{o}\left\|\hat{K}^{l}\right\|_{1} \\
& =\|\hat{K}\|_{\infty} .
\end{aligned}
$$

Similar to eq. (5.15): for every $n m$ rows, we choose $\frac{k+1}{2}^{\text {th }}$ row. Then its 1 -norm is equal to this $n m$ rows of the $\hat{K}$ 's $\infty$-norm. So the equation holds.

### 5.12 Appendix: Auxiliary numerical evidence

A number of additional experiments are conducted in this section. Unless explicitly mentioned otherwise, the experimental setup remains similar to the one in the main paper. The following experiments are conducted below:

1. The difference between the theoretical and the algorithmic bound and their evolution during training is studied in subsection 5.12.1.
2. An ablation study on the hidden size is conducted in subsection 5.12.2.
3. An ablation study is conducted on the effect of adversarial steps in subsection 5.12.3.
4. We evaluate the effect of the proposed projection into the testset performance in subsection 5.12.4.
5. We conduct experiments on four new datasets, i.e., MNIST, K-MNIST, E-MNIST-BY, NSYNTH in subsection 5.12.5. These experiments are conducted in addition to the datasets already presented in the main paper.
6. In subsection 5.12.6 experiments on three additional adversarial attacks, i.e., FGSM-0.01, APGDT and TPGD, are performed.
7. We conduct an experiment using the NCP model in subsection 5.12.7.
8. The layer-wise bound (instead of a single bound for all matrices) is explored in subsection 5.12.8.
9. The comparison with adversarial defense methods is conducted in subsection 5.12.9.

### 5.12.1 Theoretical and algorithmic bound

As mentioned in section 5.3 , projecting the quantity $\theta=\left\|\bullet{ }_{i=1}^{k} U_{i}\right\|_{\infty}$ onto their level set corresponds to a difficult non-convex problem. Given that we have an upper bound

$$
\theta=\left\|\bullet{ }_{i=1}^{k} U_{i}\right\|_{\infty} \leq \Pi_{i=1}^{k}\left\|U_{i}\right\|_{\infty}=: \gamma
$$

we want to understand in practice how tight is this bound. In figure 5.4 we compute the ratio $\frac{\gamma}{\theta}$ for PN-4. In figure 5.5 the ratio is illustrated for randomly initialized matrices (i.e., untrained networks).

### 5.12.2 Ablation study on the hidden size

Initially, we explore the effect of the hidden rank of PN-4 and PN-10 on Fashion-MNIST. figure 5.6 exhibits the accuracy on both the training and the test-set for both models. We observe that PN-10 has a better accuracy on the training set, however the accuracy on the test set is the same in the two models. We also note that increasing the hidden rank improves the accuracy on the training set, but not on the test set.


Figure 5.4: Visualization of the difference between the bound results on Fashion-MNIST (top row) and on K-MNIST (bottom row). Specifically, in (a) and (c) we visualize the ratio $\frac{\gamma}{\theta}=\frac{\prod_{i=1}^{k}\left\|U_{i}\right\|_{\infty}}{\left\|\cdot{ }_{i=1}^{k} U_{i}\right\|_{\infty}}$ for different log bound values for PN-4. In (b), (d) the exact values of the two bounds are computed over the course of the unregularized training. Notice that there is a gap between the two bounds, however importantly the two bounds are increasing at the same rate, while their ratio is close to 1 .


Figure 5.5: Visualization of the ratio $\frac{\prod_{i=1}^{k}\left\|U_{i}\right\|_{\infty}}{\left\|\cdot 0_{i=1}^{k} U_{i}\right\|_{\infty}}$ in a randomly initialized network (i.e., using normal distribution random matrices). Specifically, in (a) we visualize the ratio for different log hidden rank values for PN-10. In (b) we visualize the ratio for different depth values for hidden rank $=16$. Neither of two plots contain any regularization.


Figure 5.6: Accuracy of PN-4 and PN-10 when the hidden rank varies (plotted in log-scale).

### 5.12.3 Ablation study on the effect of adversarial steps

Our next experiment scrutinizes the effect of the number of adversarial steps on the robust accuracy. We consider in all cases a projection bound of 1 , which provides the best empirical results. We vary the number of adversarial steps and report the accuracy in figure 5.7. The results exhibit a similar performance both in terms of the dataset (i.e., Fashion-MNIST and K-MNIST) and in terms of the network (PN-4 and PN-Conv). Notice that when the adversarial attack has more than 10 steps the performance does not vary significantly from the performance at 10 steps, indicating that the projection bound is effective for stronger adversarial attacks.

### 5.12.4 Evaluation of the accuracy of PNs

In this experiment, we evaluate the accuracy of PNs. We consider three networks, i.e., PN-4, $\mathrm{PN}-10$ and $\mathrm{PN}-\mathrm{Conv}$, and train them under varying projection bounds using algorithm 5.1. Each model is evaluated on the test set of (a) Fashion-MNIST and (b) E-MNIST.

The accuracy of each method is reported in figure 5.8 , where the $x$-axis is plotted in log-scale (natural logarithm). The accuracy is better for bounds larger than 2 (in the log-axis) when compared to tighter bounds (i.e., values less than 0 ). Very tight bounds stifle the ability of the network to learn, which explains the decreased accuracy. Interestingly, PN-4 reaches similar accuracy to PN-10 and PN-Conv in Fashion-MNIST as the bound increases, while in E-MNIST it cannot reach the same performance as the bound increases. The best bounds for all three models are observed in the intermediate values, i.e., in the region of 1 in the log-axis for PN-4 and PN-10.

We scrutinize further the projection bounds by training the same models only with crossentropy loss (i.e., no bound regularization). In table 5.10, we include the accuracy of the three networks with and without projection. Note that projection consistently improves the accuracy, particularly in the case of larger networks, i.e., PN-10.


Figure 5.7: Ablation study on the effect of adversarial steps in Fashion-MNIST (top row) and K-MNIST (bottom row). All methods are run by considering a projection bound of 1 .


Figure 5.8: Accuracy of PN-4, PN-10 and PN-Conv under varying projection bounds (x-axis in log-scale) learned on (a) Fashion-MNIST, (b) E-MNIST. Notice that the performance increases for intermediate values, while it deteriorates when the bound is very tight.

| Method | PN-4 | PN-10 <br> Fashion-MNIST | PN-Conv |
| :--- | :---: | :---: | :---: |
| No projection | $87.28 \pm 0.18 \%$ | $88.48 \pm 0.17 \%$ | $86.36 \pm 0.21 \%$ |
| Projection | $87.32 \pm 0.14 \%$ | $88.72 \pm 0.12 \%$ | $86.38 \pm 0.26 \%$ |
| $E-M N I S T$ |  |  |  |
| No projection | $84.27 \pm 0.26 \%$ | $89.31 \pm 0.09 \%$ | $91.49 \pm 0.29 \%$ |
| Projection | $84.34 \pm 0.31 \%$ | $90.56 \pm 0.10 \%$ | $91.57 \pm 0.19 \%$ |

Table 5.10: The accuracy of different PN models on Fashion-MNIST (top) and E-MNIST (bottom) when trained only with SGD (first row) and when trained with projection (last row).

### 5.12.5 Experimental results on additional datasets

To validate even further we experiment with additional datasets. We describe the datasets below and then present the robust accuracy in each case. The experimental setup remains the same as in subsection 5.4.2 in the main paper. As a reminder, we are evaluating the robustness of the different models under adversarial noise.

Dataset details: There are six datasets used in this work:

1. Fashion-MNIST (Xiao et al., 2017b) includes grayscale images of clothing. The training set consists of 60,000 examples, and the test set of 10,000 examples. The resolution of each image is $28 \times 28$, with each image belonging to one of the 10 classes.
2. E-MNIST (Cohen et al., 2017) includes handwritten character and digit images with a training set of 124,800 examples, and a test set of 20,800 examples. The resolution of each image is $28 \times 28$. E-MNIST includes 26 classes. We also use the variant EMNIST-BY that includes 62 classes with 697,932 examples for training and 116, 323 examples for testing.
3. K-MNIST (Clanuwat et al., 2018b) depicts grayscale images of Hiragana characters with a training set of 60,000 examples, and a test set of 10,000 examples. The resolution of each image is $28 \times 28$. K-MNIST has 10 classes.
4. MNIST (Lecun et al., 1998) includes handwritten digits images. MNIST has a training set of 60,000 examples, and a test set of 10,000 examples. The resolution of each image is $28 \times 28$.
5. CIFAR-10 (Krizhevsky et al., 2014) depicts images of natural scenes. CIFAR-10 has a training set of 50,000 examples, and a test set of 10,000 examples. The resolution of each RGB image is $32 \times 32$.
6. NSYNTH (Engel et al., 2017) is an audio dataset containing 305, 979 musical notes, each with a unique pitch, timbre, and envelope.

We provide a visualization ${ }^{3}$ of indicative samples from MNIST, Fashion-MNIST, K-MNIST and E-MNIST in figure 5.9.

We originally train PN-4, PN-10 and PN-Conv without projection bounds. The results are reported in table 5.11 (columns titled 'No proj') for MNIST and K-MNIST, table 5.13 (columns titled 'No proj') for E-MNIST-BY and table 5.14 (columns titled 'No proj’) for NSYNTH. Next, we consider the performance under varying projection bounds; the accuracy in each case is depicted in figure 5.10 for K-MNIST, MNIST and E-MNIST-BY and figure 5.11 for NSYNTH. The figures (and the tables) depict the same patterns that emerged in the two main experiments, i.e., the performance can be vastly improved for intermediate values of the projection bound. Similarly, we validate the performance when using adversarial training. The results in table 5.12 demonstrate the benefits of using projection bounds even in the case of adversarial training.

### 5.12.6 Experimental results of more types of attacks

To further verify the results of the main paper, we conduct experiments with three additional adversarial attacks: a) FGSM with $\epsilon=0.01$, b) Projected Gradient Descent in Trades (TPGD) (Zhang et al., 2019a), c) Targeted Auto-Projected Gradient Descent (APGDT) (Croce and Hein, 2020a). In TPGD and APGDT, we use the default parameters for a one-step attack.

The quantitative results are reported in table 5.15 for four datasets and the curves of FashionMNIST and E-MNIST are visualized in figure 5.12 and the curves of K-MNIST and MNIST are visualized in figure 5.13. The results in both cases remain similar to the attacks in the main paper, i.e., the proposed projection improves the performance consistently across attacks, types of networks and adversarial attacks.

### 5.12.7 Experimental results in NCP model

To complement, the results of the CCP model, we conduct an experiment using the NCP model. That is, we use a $4^{\text {th }}$ degree polynomial expansion, called NCP-4, for our experiment. We conduct an experiment in the K-MNIST dataset and present the result with varying bound in figure 5.14. Notice that the patterns remain similar to the CCP model, i.e., intermediate values of the projection bound can increase the performance significantly.

### 5.12.8 Layer-wise bound

To assess the flexibility of the proposed method, we assess the performance of the layer-wise bound. In the previous sections, we have considered using a single bound for all the matrices, i.e., $\left\|U_{i}\right\|_{\infty} \leq \lambda$, because the projection for a single matrix has efficient projection algorithms. However, lemma 5.1 enables each matrix $U_{i}$ to have a different bound $\lambda_{i}$. We assess the performance of having different bounds for each matrix $U_{i}$.

[^7]

Figure 5.9: Samples from the datasets used for the numerical evidence. MNIST (top left), Fashion-MNIST (top right), K-MNIST (bottom left), E-MNIST (bottom right). Below each image, the class name and the class number are denoted.
$\rightarrow$ Clean --*- FGSM_0.1 --4- PGD_0.1_0.01_20 $\rightarrow$ PGD_0.3_0.03_20


Figure 5.10: Adversarial attacks during testing on K-MNIST (top row), MNIST (middle row), E-MNIST-BY (bottom row) with the $x$-axis is plotted in log-scale. Note that intermediate values of projection bounds yield the highest accuracy. The patterns are consistent in all datasets and across adversarial attacks.


Figure 5.11: Adversarial attacks during testing on NSYNTH.
$\ldots$ Clean ....+... FGSM0.01 --↔-- APGDT --*-- TPGD


Figure 5.12: Three new adversarial attacks during testing on Fashion-MNIST (top row), EMNIST (bottom row) with the x -axis is plotted in log-scale. Note that intermediate values of projection bounds yield the highest accuracy. The patterns are consistent in both datasets and across adversarial attacks.



Figure 5.13: Three new adversarial attacks during testing on K-MNIST (top row), MNIST (bottom row) with the x -axis is plotted in log-scale. Note that intermediate values of projection bounds yield the highest accuracy. The patterns are consistent in both datasets and across adversarial attacks.


Figure 5.14: Experimental result of K-MNIST in NCP model.

| Method | No proj. | Our method |  | Jacobian | $L_{2}$ |
| :--- | :--- | :---: | :---: | :---: | :---: |
|  |  |  | $K-M N I S T$ |  |  |

Table 5.11: Comparison of regularization techniques on K-MNIST (top) and MNIST (bottom). In each dataset, the base networks are $\mathrm{PN}-4$, i.e., a $4^{\text {th }}$ degree polynomial, on the top four rows, PN-10, i.e., a $10^{\text {th }}$ degree polynomial, on the middle four rows and PN-Conv, i.e., a $4^{\text {th }}$ degree polynomial with convolutions, on the bottom four rows. Our projection method exhibits the best performance in all three attacks, with the difference on accuracy to stronger attacks being substantial.

We experiment on PN-4 that we set a different projection bound for each matrix $U_{i}$. Specifically, we use five different candidate values for each $\lambda_{i}$ and then perform the grid search on the Fashion-MNIST FGSM-0.01 attack. The results on Fashion-MNIST in table 5.16 exhibit how the layer-wise bounds outperform the previously used single bound ${ }^{4}$. The best performing values for PN-4 are $\lambda_{1}=1.5, \lambda_{2}=2, \lambda_{3}=1.5, \lambda_{4}=2, \mu=0.8$. The values of $\lambda_{i}$ in the first few layers are larger, while the value in the output matrix $C$ is tighter.

To scrutinize the results even further, we evaluate whether the best performing $\lambda_{i}$ can improve the performance in different datasets and the FGSM-0.1 attack. In both cases, the best

[^8]| Method | AT | Our method + AT <br> Adversarial training (AT) | Jacobian + AT | $L_{2}+$ AT |  |
| :--- | :---: | :---: | :---: | :---: | :---: |
|  | PN-10 on $K$ K-MNIST |  |  |  |  |

Table 5.12: Comparison of regularization techniques on (a) K-MNIST (top) and (b) MNIST (bottom) along with adversarial training (AT). The base network is a PN -10, i.e., $10^{\text {th }}$ degree polynomial. Our projection method exhibits the best performance in all three attacks.

|  | Method | PN-4, PN-10 and PN-Conv on E-MNIST-BY <br> No proj. |  |
| :--- | :--- | :---: | :---: |
| PN-4 |  | $80.18 \pm 0.19 \%$ | $\mathbf{8 0 . 2 6} \pm \mathbf{0 . 1 7 \%}$ |
|  | FGSM-0.1 | $3.65 \pm 0.76 \%$ | $\mathbf{1 6 . 5 8} \pm \mathbf{3 . 8 7 \%}$ |
|  | PGD-(0.1, 20, 0.01) | $4.57 \pm 1.98 \%$ | $\mathbf{1 9 . 7 7} \pm \mathbf{4 . 4 2 \%}$ |
|  | PGD-(0.3, 20, 0.03) | $0.59 \pm 0.40 \%$ | $\mathbf{1 0 . 1 3} \pm \mathbf{2 . 0 8 \%}$ |
| PN-10 | Clean | $84.17 \pm 0.06 \%$ | $\mathbf{8 5 . 3 2} \pm \mathbf{0 . 0 4 \%}$ |
|  | FGSM-0.1 | $11.67 \pm 1.21 \%$ | $\mathbf{3 2 . 3 7} \pm \mathbf{2 . 5 8 \%}$ |
|  | PGD-(0.1, 20, 0.01) | $2.48 \pm 0.66 \%$ | $\mathbf{3 1 . 2 2} \pm \mathbf{2 . 3 2 \%}$ |
|  | PGD-(0.3, 20, 0.03) | $0.03 \pm 0.05 \%$ | $\mathbf{1 3 . 7 4} \pm \mathbf{0 . 7 7 \%}$ |
| PN-Conv | Clean | $85.92 \pm 0.08 \%$ | $\mathbf{8 6 . 0 3} \pm \mathbf{0 . 0 8 \%}$ |
|  | FGSM-0.1 | $0.65 \pm 0.17 \%$ | $\mathbf{2 9 . 0 7} \pm \mathbf{2 . 7 2 \%}$ |
|  | PGD-(0.1,20,0.01) | $1.57 \pm 1.40 \%$ | $\mathbf{3 1 . 0 6} \pm \mathbf{4 . 7 0 \%}$ |
|  | PGD-(0.3,20, 0.03) | $0.33 \pm 0.06 \%$ | $\mathbf{2 3 . 9 3} \pm \mathbf{6 . 3 2 \%}$ |

Table 5.13: Comparison of regularization techniques on E-MNIST-BY. The base network are PN-4, i.e., $4^{\text {th }}$ degree polynomial, on the top four rows, PN-10, i.e., $10^{\text {th }}$ degree polynomial, on the middle four rows and PN -Conv, i.e., a $4^{\text {th }}$ degree polynomial with convolution, on the bottom four rows. Our projection method exhibits the best performance in all three attacks, with the difference on accuracy to stronger attacks being substantial.

| Model | PN-4 |  |
| :--- | :---: | :---: |
| Projection | No-proj | Proj |
| Clean accuracy | $80.25 \pm 0.27 \%$ | $\mathbf{8 0 . 3 3} \pm \mathbf{0 . 2 6} \%$ |
| FGSM-0.1 | $0.91 \pm 0.14 \%$ | $\mathbf{2 2 . 2 5} \pm \mathbf{0 . 0 4} \%$ |
| PGD-(0.1, 20, 0.01) | $0.31 \pm 0.11 \%$ | $\mathbf{2 2 . 2 7} \pm \mathbf{0 . 0 0} \%$ |
| PGD-(0.3, 20, 0.03) | $0.46 \pm 0.29 \%$ | $\mathbf{2 0 . 3 8} \pm \mathbf{2 . 3 0} \%$ |

Table 5.14: Evaluation of the robustness of PN models on NSYNTH. Each line refers to a different adversarial attack. The projection offers an improvement in the accuracy in each case; in PGD attacks projection improves the accuracy by a remarkable margin.
performing $\lambda_{i}$ can improve the performance of the single bound.

### 5.12.9 Adversarial defense method

One frequent method used against adversarial perturbations are the so called adversarial defense methods. We assess the performance of adversarial defense methods on the PNs when compared with the proposed method.

We experiment on PN-4 in Fashion-MNIST. We chose three different methods: gaussian denoising, median denoising and guided denoising (Liao et al., 2018). Gaussian denoising and median denoising are the methods of using gaussian filter and median filter for feature denoising (Xie et al., 2019). The results in table 5.17 show that in both attacks our method performs favourably to the adversarial defense methods.

### 5.13 Bibliographic Note

The candidate proposed the original idea of this paper and derived the main results for the Rademacher Complexity and Lipschtiz constant of the CCP Decomposition (theorems 5.1 and 5.2). Theorem 5.6 and theorem 5.7 where derived by Z. Zhu under the candidate's supervision and feedback.

|  | Method | No proj. Fash | Our method <br> n-MNIST |
| :---: | :---: | :---: | :---: |
| PN-4 | FGSM-0.01 | $26.49 \pm 3.13 \%$ | $58.09 \pm 1.63 \%$ |
|  | APGDT | $16.59 \pm 4.35 \%$ | $50.83 \pm 1.55 \%$ |
|  | TPGD | $26.88 \pm 6.78 \%$ | $59.03 \pm 1.45 \%$ |
| PN-10 | FGSM-0.01 | $18.59 \pm 1.82 \%$ | 60.56 $\pm$ 1.06\% |
|  | APGDT | $8.76 \pm 1.14 \%$ | $51.93 \pm 1.91 \%$ |
|  | TPGD | $14.53 \pm 1.49 \%$ | $63.33 \pm 0.51 \%$ |
| PN-Conv | FGSM-0.01 | $15.30 \pm 3.10 \%$ | $\mathbf{5 5 . 9 0} \pm \mathbf{2 . 6 0 \%}$ |
|  | APGDT | $11.88 \pm 1.33 \%$ | $53.49 \pm 0.72 \%$ |
|  | TPGD | $14.50 \pm 1.59 \%$ | 58.72 $\pm$ 1.87\% |
|  |  | E-MNIST |  |
| PN-4 | FGSM-0.01 | $13.40 \pm 5.16 \%$ | $\mathbf{3 2 . 8 3} \pm \mathbf{2 . 0 8 \%}$ |
|  | APGDT | $9.33 \pm 4.00 \%$ | $\mathbf{2 6 . 3 8} \pm \mathbf{2 . 7 0} \%$ |
|  | TPGD | $17.40 \pm 3.11 \%$ | 34.68 $\pm$ 1.92\% |
| PN-10 | FGSM-0.01 | $14.47 \pm 1.80 \%$ | $48.28 \pm 3.06 \%$ |
|  | APGDT | $10.13 \pm 0.93 \%$ | $41.72 \pm 4.05 \%$ |
|  | TPGD | $13.97 \pm 0.88 \%$ | $47.44 \pm 3.62 \%$ |
| PN-Conv | FGSM-0.01 | $4.71 \pm 1.10 \%$ | $\mathbf{3 9 . 3 7} \pm \mathbf{5 . 4 3} \%$ |
|  | APGDT | $3.58 \pm 0.66 \%$ | 30.43 $\pm 4.87 \%$ |
|  | TPGD | $4.08 \pm 0.33 \%$ | $\mathbf{3 5 . 8 5} \pm \mathbf{1 0 . 2 0} \%$ |
|  |  | K-MNIST |  |
| PN-4 | FGSM-0.01 | $23.31 \pm 5.34 \%$ | $43.74 \pm 5.97 \%$ |
|  | APGDT | $17.02 \pm 6.97 \%$ | $39.43 \pm 1.89 \%$ |
|  | TPGD | $23.45 \pm 7.67 \%$ | $48.46 \pm 3.84 \%$ |
| PN-10 | FGSM-0.01 | $26.87 \pm 2.14 \%$ | $50.99 \pm 3.52 \%$ |
|  | APGDT | $16.23 \pm 1.32 \%$ | $41.46 \pm 3.85 \%$ |
|  | TPGD | $22.63 \pm 0.99 \%$ | $49.91 \pm 1.37 \%$ |
| PN-Conv | FGSM-0.01 | $12.31 \pm 2.03 \%$ | $52.58 \pm 6.80 \%$ |
|  | APGDT | $13.47 \pm 2.19 \%$ | $42.94 \pm 1.68 \%$ |
|  | TPGD | $14.25 \pm 2.51 \%$ | $48.19 \pm 3.02 \%$ |
|  |  | MNIST |  |
| PN-4 | FGSM-0.01 | $34.14 \pm 7.63 \%$ | 73.95 $\pm$ 5.18\% |
|  | APGDT | $29.88 \pm 9.47 \%$ | 71.26 $\pm 4.88 \%$ |
|  | TPGD | $27.01 \pm 9.77 \%$ | $76.88 \pm 1.98 \%$ |
| PN-10 | FGSM-0.01 | $32.34 \pm 4.67 \%$ | $78.83 \pm 1.63 \%$ |
|  | APGDT | $19.55 \pm 1.72 \%$ | $75.22 \pm 2.05 \%$ |
|  | TPGD | $28.11 \pm 3.87 \%$ | 79.74 $\pm 2.07 \%$ |
| PN-Conv | FGSM-0.01 | $22.73 \pm 3.10 \%$ | $69.83 \pm 8.91 \%$ |
|  | APGDT | $17.95 \pm 3.39 \%$ | $64.94 \pm 8.96 \%$ |
|  | TPGD | $21.82 \pm 3.07 \%$ | $66.47 \pm 11.83 \%$ |

Table 5.15: Evaluation of the robustness of PN models on four datasets with three new types of attacks. Each line refers to a different adversarial attack. The projection offers an improvement in the accuracy in each case.

| Method | No proj. | Jacobian | $L_{2}$ <br> Fashion-MNIST | Single bound | Layer-wise bound |
| :--- | :---: | :---: | :---: | :---: | :---: |
| FGSM-0.01 | $26.49 \pm 3.13 \%$ | $39.88 \pm 4.59 \%$ | $24.36 \pm 1.95 \%$ | $58.09 \pm 1.63 \%$ | $\mathbf{6 3 . 9 5} \pm \mathbf{1 . 2 6 \%}$ |
| FGSM-0.1 | $12.92 \pm 2.74 \%$ | $17.90 \pm 6.51 \%$ | $13.80 \pm 3.65 \%$ | $46.43 \pm 0.95 \%$ | $\mathbf{5 5 . 1 4} \pm \mathbf{3 . 6 5 \%}$ |
| K-MNIST |  |  |  |  |  |
| FGSM-0.01 | $23.31 \pm 5.34 \%$ | $25.46 \pm 3.51 \%$ | $27.85 \pm 7.62 \%$ | $43.74 \pm 5.97 \%$ | $\mathbf{4 9 . 6 1} \pm \mathbf{1 . 4 4 \%}$ |
| FGSM-0.1 | $18.86 \pm 2.61 \%$ | $22.61 \pm 1.30 \%$ | $22.05 \pm 2.76 \%$ | $35.84 \pm 1.67 \%$ | $\mathbf{4 7 . 5 4} \pm \mathbf{3 . 7 4 \%}$ |
| MNIST |  |  |  |  |  |
| FGSM-0.01 | $34.14 \pm 7.63 \%$ | $32.78 \pm 6.94 \%$ | $29.31 \pm 3.95 \%$ | $73.95 \pm 5.18 \%$ | $\mathbf{7 9 . 2 3} \pm \mathbf{3 . 6 5 \%}$ |
| FGSM-0.1 | $20.96 \pm 5.16 \%$ | $33.59 \pm 8.46 \%$ | $26.07 \pm 5.64 \%$ | $64.09 \pm 2.41 \%$ | $\mathbf{7 4 . 9 7} \pm \mathbf{5 . 6 0 \%}$ |

Table 5.16: Evaluation of our layer-wise bound versus our single bound. To avoid confusion with previous results, note that 'single bound' corresponds to 'Our method' in the rest of the tables in this work. The different $\lambda_{i}$ values are optimized on Fashion-MNIST FGSM-0.01 attack. Then, the same $\lambda_{i}$ values are used for training the rest of the methods. The proposed layer-wise bound outperforms the single bound by a large margin, improving even further by baseline regularization schemes.

| Method | Fashion-MNIST |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| FGSM-0.01 | $26.49 \pm 3.13 \%$ | $58.09 \pm 1.63 \%$ | $63.95 \pm 1.26 \%$ | $18.80 \pm 3.08 \%$ | $19.68 \pm 3.20 \%$ | $29.69 \pm 5.37 \%$ |
| FGSM-0.1 | $12.92 \pm 2.74 \%$ | $46.43 \pm 0.95 \%$ | $\mathbf{5 5 . 1 4} \pm \mathbf{3 . 6 5 \%}$ | $14.14 \pm 2.77 \%$ | $14.02 \pm 1.95 \%$ | $22.94 \pm 5.65 \%$ |

Table 5.17: Comparison of the proposed method against adversarial defense methods on feature denoising (Xie et al., 2019) and guided denoising (Liao et al., 2018). Notice that the single bound (cf. subsection 5.12 .8 for details) already outperforms the proposed defense methods, while the layer-wise bounds further improves upon our single bound case.

## 6 Finding Actual Descent Directions for Adversarial Training

Fabian Latorre, Igor Krawczuk, Leello Dadi, Thomas Pethick and Volkan Cevher. International Conference on Learning Representations (ICLR) 2023.


#### Abstract

Adversarial Training using a strong first-order adversary (PGD) is the gold standard for training Deep Neural Networks that are robust to adversarial examples. We show that, contrary to the general understanding of the method, the gradient at an optimal adversarial example may increase, rather than decrease, the adversarially robust loss. This holds independently of the learning rate. More precisely, we provide a counterexample to a corollary of Danskin's Theorem presented in the seminal paper of Madry et al. (2018a) which states that a solution of the inner maximization problem can yield a descent direction for the adversarially robust loss. Based on a correct interpretation of Danskin's Theorem, we propose Danskin's Descent Direction ( DDi ) and we verify experimentally that it provides better directions than those obtained by a PGD adversary. Using the CIFAR10 dataset we further provide a real world example showing that our method achieves a steeper increase in robustness levels in the early training stages of smooth-activation networks without BatchNorm, and is more stable than the PGD baseline. As a limitation, PGD training of ReLU+BatchNorm networks still performs better, but current theory is unable to explain this.


### 6.1 Introduction

Adversarial Training (AT) (Goodfellow et al., 2015; Madry et al., 2018a) has become the de-facto algorithm used to train Neural Networks that are robust to adversarial examples (Szegedy et al., 2014b). Variations of AT together with data augmentation yield the best-performing models in public benchmarks (Croce et al., 2020a). Despite lacking optimality guarantees for the inner-maximization problem, the simplicity and performance of AT are enough reasons to embrace its heuristic nature.

From an optimization perspective, the consensus is that AT is a sound algorithm: based on Danskin's Theorem, Madry et al. (2018a, Corollary C.2) posit that by finding a maximizer of
the inner non-concave maximization problem, i.e., an optimal adversarial example, one can obtain a descent direction for the adversarially robust loss. What if this is not true? are we potentially overlooking issues in its algorithmic framework?

As mentioned in (Dong et al., 2020, Section 2.3), Corollary C. 2 in Madry et al. (2018a) can be considered the theoretical optimization foundation of the non-convex non-concave minmax optimization algorithms that we now collectively refer to as Adversarial Training. It justifies the two-stage structure of the training loop: first we find one approximately optimal adversarial example and then we update the model using the gradient (with respect to the model parameters) at the perturbed input.

The only drawbacks of a first-order adversary seem to be its computational complexity and its approximate suboptimal solver nature. Ignoring the computational complexity issue, suppose we have access to a theoretical oracle that provides a single solution of the inner-maximization problem. In such idealized setting, can we safely assume AT is decreasing the adversarially robust loss on the data sample? According to the aforementioned theoretical results, it would appear so.

In this work, we scrutinize the optimization paradigm on which Adversarial Training (AT) has been founded, and we posit that finding multiple solutions of the inner-maximization problem is necessary to find good descent directions of the adversarially robust loss. In doing so, we hope to improve our understanding of the non-convex/non-concave min-max optimization problem that underlies the Adversarial Training methodology, and potentially improving its performance.

Our contributions: We present two counterexamples to Madry et al. (2018a, Corollary C.2), the motivation behind AT. They show that using the gradient (with respect to the parameters of the model) evaluated at a single solution of the inner-maximization problem, can increase the robust loss, i.e., it can harm the robustness of the model. In particular, in counterexample 6.2 many descent directions exist, but they cannot be found if we only compute a single solution of the inner-maximization problem. In section 6.2 we explain that the flaw in the proof is due to a misunderstanding of the directional derivative notion that is used in the original work of Danskin (1966).

Based on our findings, we propose Danskin's Descent Direction (DDi, algorithm 6.1). It aims to overcome the problems of the single adversarial example paradigm of AT by exploiting multiple adversarial examples, obtaining better update directions for the network. For a data-label pair, DDi finds the steepest descent direction for the robust loss, assuming that (i) there exists a finite number of solutions of the inner-maximization problem and (ii) they can be found with first-order methods.

In section 2.7 we verify experimentally that: $(i)$ it is unrealistic to assume a unique solution of the inner-maximization problem, hence making a case for our method DDi, ( $i i$ ) our method can achieve more stable descent dynamics than the vanilla AT method in synthetic scenarions


Figure 6.1: (a) and (b): comparison of our method (DDi) and the single-adversarial-example method (PGD) on a synthetic min-max problem. Using a single example may increase the robust loss. DDi computes 10 examples and can avoid this. (c): similar improvement over PGD training shown on CIFAR10, where DDi with 10 examples speeds up convergence. More details in section 2.7
and ( iii ) on the CIFAR10 dataset DDi is more stable and achieves higher robustness levels in the early stages of traning, compared with a PGD adversary of equivalent complexity. This is observed in a setting where the conditions of Danskin's Theorem holds, i.e., using differentiable activation functions and removing BatchNorm. As a limitation, PGD training of ReLU+BatchNorm networks still performs better, but there is no theory explaining this.

Remark. The fact that (Madry et al., 2018a, Corollary C.2) is false, might be well-known in the optimization field. In the convex setting it corresponds to the common knowledge that a negative subgradient of a non-smooth convex function might not be a descent direction c.f., (Boyd, 2014, Section 2.1). However, we believe this is not well-known in the AT community given that $(i)$ its practical implications i.e., methods deriving steeper descent updates using multiple adversarial examples, have not been previously introduced, and (ii) the results in Madry et al. (2018a) have been central in the development of AT. Hence, our contribution can be understood as raising awareness about the issue, and demonstrating its practical implications for AT.

### 6.2 A Counterexample to Madry et al. (2018a, Corollary C.2)

Preliminaries. Let $\theta \in \mathbb{R}^{d}$ be the parameters of a model, $(x, y) \sim \mathscr{D}$ a data-label distribution, $\delta$ a perturbation in a compact set $\mathscr{S}_{0}$ and $L$ a loss function. The optimization objective of AT is:

$$
\begin{equation*}
\min _{\theta} \rho(\theta), \quad \text { where } \rho(\theta):=\mathbb{E}_{(x, y) \sim \mathscr{D}}\left[\max _{\delta \in \mathscr{S}_{0}} L(\theta, x+\delta, y)\right] \tag{6.1}
\end{equation*}
$$

In this setting $\rho(\theta)$ is referred to as the adversarial loss or robust loss. In order to optimize eq. (6.1) via iterative first-order methods, we need access to an stochastic gradient of the adversarial loss $\rho$ or at least, the weaker notion of stochastic descent direction i.e., a direction
along which the function

$$
\begin{equation*}
\phi(\theta):=\max _{\delta \in \mathscr{S}:=\mathscr{S}_{0}^{k}}\left\{g(\theta, \delta):=\frac{1}{k} \sum_{i=1}^{k} L\left(\theta, x_{i}+\delta_{i}, y_{i}\right)\right\} \tag{6.2}
\end{equation*}
$$

decreases in value. We have collected the perturbations $\delta_{i} \in \mathscr{S}_{0}$ on the batch $\left\{\left(x_{i}, y_{i}\right)\right\}_{i=1}^{k}$ as the columns of a matrix $\delta=\left[\delta_{1}, \ldots, \delta_{k}\right] \in \mathscr{S}:=\mathscr{S}_{0}^{k}$ which is also a compact set. To obtain a descent direction for partial maximization functions like $\phi$ we resort to Danskin's Theorem:

Theorem 6.1 (Danskin (1966)). Let $\mathscr{S}$ be a compact topological space, and let $g: \mathbb{R}^{d} \times \mathscr{S}$ be a continuous function such that $g(\cdot, \delta)$ is differentiable for all $\delta \in \mathscr{S}$ and $\nabla_{\theta} g(\theta, \delta)$ is continuous on $\mathbb{R}^{d} \times \mathscr{S}$. Let

$$
\begin{equation*}
\phi(\theta):=\max _{\delta \in \mathscr{S}} g(\theta, \delta), \quad \mathscr{S}^{\star}(\theta):=\underset{\delta \in \mathscr{S}}{\operatorname{argmax}} g(\theta, \delta) \tag{6.3}
\end{equation*}
$$

Let $\gamma \in \mathbb{R}^{d}$ with $\|\gamma\|_{2}=1$ be an arbitrary unit vector. The directional derivative $D_{\gamma} \phi(\theta)$ of $\phi$ in the direction $\gamma$ at the point $\theta$ exists, and is given by the formula

$$
\begin{equation*}
D_{\gamma} \phi(\theta)=\max _{\delta \in \mathscr{S}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle \tag{6.4}
\end{equation*}
$$

Remark. $\quad \gamma \neq 0$ is called a descent direction of $\phi$ at $\theta$ if and only if $D_{\gamma} \phi(\theta)<0$, i.e., if the directional derivative is strictly negative.
corollary 6.1 is an equivalent rephrasing of Madry et al. (2018a, Corollary C.2.), and was originally claimed to be a consequence of theorem 6.1 . Unfortunately counterexample 6.1 shows that the corollary is false. As theorem 6.1 (Danskin's Theorem) is true, this means that there is some mistake in the proof of the corollary provided in Madry et al. (2018a).

Corollary 6.1. Let $\delta^{\star} \in \mathscr{S}^{\star}(\theta) . I f-\nabla_{\theta} g\left(\theta, \delta^{\star}\right) \neq 0$, then it is a descent direction for $\phi$ at $\theta$.

Counterexample 6.1. Let $\mathscr{S}:=[-1,1]$ and $g(\theta, \delta)=\theta \delta$. The conditions of Danskin's theorem clearly hold in this case, and

$$
\begin{equation*}
\phi(\theta):=\max _{\delta \in[-1,1]} \theta \delta=|\theta| \tag{6.5}
\end{equation*}
$$

Note that at $\theta=0$, we have $\mathscr{S}^{\star}(0)=[-1,1]$. Choosing $\delta=1 \in \mathscr{S}^{\star}(0)$ we have that $g(\theta, 1)=\theta$ and so $-\nabla_{\theta} g(0,1)=-1 \neq 0$. Hence, corollary 6.1 would imply that -1 is a descent direction for $\phi(\theta)=|\theta|$. However, $\theta=0$ is a global minimizer of the absolute value function, which means that there exists no descent direction. This is a contradiction.

To cast more clarity on why corollary 6.1 is false, we explain what is the mistake in the proof provided in Madry et al. (2018a). The main issue is the definition of the directional derivative, a concept in multivariable calculus that is defined in slightly different ways in the literature.

Definition 6.1. Let $\phi: \mathbb{R}^{d} \rightarrow \mathbb{R}$. For a nonzero vector $\gamma \in \mathbb{R}^{d}$, the one-sided directional derivative
of $\phi$ in the direction $\gamma$ at the point $\theta$ is defined as the one-sided limit:

$$
\begin{equation*}
D_{\gamma} \phi(\theta):=\lim _{t \rightarrow 0^{+}} \frac{\phi(\theta+t \gamma)-\phi(\theta)}{t\|\gamma\|_{2}} \tag{6.6}
\end{equation*}
$$

The two-sided directional derivative is defined as the two-sided limit:

$$
\begin{equation*}
\hat{D}_{\gamma} \phi(\theta):=\lim _{t \rightarrow 0} \frac{\phi(\theta+t \gamma)-\phi(\theta)}{t\|\gamma\|_{2}} \tag{6.7}
\end{equation*}
$$

Unfortunately, it is not always clear which one of the two notions is meant when the term directional derivative is used. Indeed, as our notation suggests, the one-sided definition eq. (6.6) is the one used in the statement of Danskin's Theorem (Danskin, 1966). However, the proof of corollary 6.1 provided in Madry et al. (2018a) mistakenly assumes the twosided definition eq. (6.7), and inadvertently uses the following property that holds for $\hat{D}_{\gamma} \phi(\theta)$ (eq. (6.7)) but not for $D_{\gamma} \phi(\theta)$ (eq. (6.6)):
Lemma 6.1. For the two-sided directional derivative definition (6.7) it holds that- $\hat{D}_{\gamma} \phi(\theta)=$ $\hat{D}_{-\gamma} \phi(\theta)$ provided that $\hat{D}_{\gamma}$ exists. In particular, if $\hat{D}_{\gamma} \phi(\theta)>0$ then $\hat{D}_{-\gamma} \phi(\theta)<0$. However this is not true for the one-sided directional derivative (6.6), as the example $\phi(\theta)=|\theta|$ at $\theta=0$ shows (both directional derivatives are strictly positive).

We provide a proof of this fact in section 6.11. The (flawed) proof of corollary 6.1 provided in Madry et al. (2018a) starts by noting that for a solution $\bar{\delta}$ of the inner-maximization problem, the directional derivative in the direction $\gamma=\nabla_{\theta} g(\theta, \bar{\delta})$ is positive, as implied by Danskin's Theorem:

$$
\begin{equation*}
D_{\gamma} \phi(\theta)=\max _{\delta \in \mathscr{S}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle \geq\left\langle\nabla_{\theta} g(\theta, \bar{\delta}), \nabla_{\theta} g(\theta, \bar{\delta})\right\rangle=\left\|\nabla_{\theta} g(\theta, \bar{\delta})\right\|^{2}>0 \tag{6.8}
\end{equation*}
$$

assuming that $\nabla_{\theta} g(\theta, \bar{\delta})$ is non-zero. The mistake in the proof lies in concluding that $D_{-\gamma} \phi(\theta)<0$. Following lemma 6.1, this property does not hold for the one-sided directional derivative definition eq. (6.6), the one used in Danskin's Theorem.

### 6.3 A Counterexample at a point that is not locally optimal

The question remains whether a slightly modified version of corollary 6.1 holds true: it might be the case that by adding some mild assumption, we exclude all possible counterexamples. In the particular case of counterexample 6.1, $\theta=0$ is a local optimum of the function $\phi(\theta)=|\theta|$. At such points, descent directions do not exist. However, in the trajectory of an iterative optimization algorithm we are mostly concerned with non-locally-optimal points. Hence, we explore whether adding the assumption that $\theta$ is not locally optimal can make corollary 6.1 true. Unfortunately, we will show that this is not the case.

To this end we construct a family of counterexamples to corollary 6.1 with the following
properties: ( $i$ ) there exists a descent direction at a point $\theta$ (that is, $\theta$ is not locally optimal) and ( $i i$ ), it does not coincide with $-\nabla_{\theta} g(\theta, \delta)$, for any optimal $\delta \in \mathscr{S}^{\star}(\theta)$. Moreover, all the directions $-\nabla_{\theta} g(\theta, \delta)$ are in fact ascent directions i.e., they lead to an increase in the function $\phi(\theta)$.

Counterexample 6.2. Let $\mathscr{S}:=[0,1]$ and let $u, v \in \mathbb{R}^{2}$ be unit vectors such that $-1<\langle u, v\rangle<0$. That is, $u$ and $v$ form an obtuse angle. Let

$$
\begin{equation*}
g(\theta, \delta)=\delta\langle\theta, u\rangle+(1-\delta)\langle\theta, v\rangle+\delta(\delta-1) \tag{6.9}
\end{equation*}
$$

Clearly, the function satisfies all conditions of theorem 6.1. At $\theta=0$, we have that $\mathscr{S}^{\star}(0)=$ $\operatorname{argmax} \delta(\delta-1)=\{0,1\}$. At $\delta=0$ we have $\nabla_{\theta} g(\theta, 0)=\nabla_{\theta}\langle\theta, v\rangle=v$ and at $\delta=1$ we have $\delta \in[0,1]$ $\nabla_{\theta} g(\theta, 1)=\nabla_{\theta}\langle\theta, u\rangle=u$. We compute the value of the directional derivatives in the negative direction of such vectors. According to Danskin's Theorem we have

$$
\begin{equation*}
D_{-v} \phi(0)=\max _{\delta \in\{0,1\}}\left\langle-v, \nabla_{\theta} g(\theta, \delta)\right\rangle=\max (\langle-v, v\rangle,\langle-v, u\rangle) \geq-\langle v, u\rangle>0 \tag{6.10}
\end{equation*}
$$

where $-\langle\nu, u\rangle>0$ holds by construction. Analogously, $D_{-u} \phi(0)>0$. This means that all such directions are ascent directions. However, for the direction $\gamma=-(u+v)$ we have

$$
\begin{align*}
D_{\gamma} \phi(\theta) & \propto \max _{\delta \in\{0,1\}}\left\langle-(u+v), \nabla_{\theta} g(\theta, \delta)\right\rangle  \tag{6.11}\\
& =\max (\langle-u-v, u\rangle,\langle-u-v, v\rangle)=-1-\langle u, v\rangle<0
\end{align*}
$$

where the last inequality also follows by construction. Hence, $-(u+v)$ is a descent direction.

As counterexample 6.2 shows, Adversarial Training has the following problem: even if we are able to compute one solution of the inner-maximization problem $\bar{\delta} \in \mathscr{S}$ it can be the case that moving in the direction $-\nabla_{\theta} g(\theta, \bar{\delta})$ increases the robust training loss i.e., the classifier becomes less, rather than more, robust. This can happen at any stage, independently of the local optimality of $\theta$.

For a non-locally-optimal $\theta \in \mathbb{R}^{d}$, the construction of the counterexamples relies on the following: if for any gradient computed at one inner-max solution, there exist another gradient (at a different inner-max solution) forming an obtuse angle, then no single inner-max solution yields a descent direction. Consequently, it suffices to ensure that for any gradient that can be found by solving the inner problem, there exists another one that has a negative inner product with it. Precisely, our counterexample 6.2 is carefully crafted so that this property holds.

### 6.4 Danskin's Descent Direction

Danskin's Theorem implies that the directional derivative depends on all the solutions of the inner-max problem $\mathscr{S}^{\star}(\theta)$ c.f., eq. (6.4). One possible issue in Adversarial Training is relying on a single solution, as it does not necessarily lead to a descent direction c.f. counterexample 6.2.

To fix this, we design an algorithm that uses multiple adversarial perturbations per data sample. In theory, we can obtain the steepest descent direction for the robust loss on a batch $\left\{\left(x_{i}, y_{i}\right): i=1, \ldots, k\right\}$ by solving the following min-max problem:

$$
\begin{equation*}
\gamma^{\star} \in \underset{\gamma:\|\gamma\|_{2}=1}{\operatorname{argmin}} \max _{\delta \in \mathscr{S}_{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle, \quad g(\theta, \delta):=\frac{1}{k} \sum_{i=1}^{k} L\left(\theta, x_{i}+\delta_{i}, y_{i}\right) \tag{6.12}
\end{equation*}
$$

On the one hand, if the set of maximizers $\mathscr{S}^{\star}(\theta)$ is infinite, eq. (6.12) would be out of reach for computationally tractable methods. On the other hand, the solution is trivial if there is a single maximizer, but we verify experimentally in section 2.7 that such assumption is wrong in practice. In conclusion, a compromise has to be made in order to devise an tractable algorithm that is relevant in practical scenarios. First, we assume that the set of optimal adversarial perturbations is finite:

$$
\begin{equation*}
\mathscr{S}^{\star}(\theta):=\underset{\delta \in \mathscr{S}}{\operatorname{argmax}} g(\theta, \delta)=\mathscr{S}_{m}^{\star}(\theta)=\left\{\delta^{(1)}, \ldots, \delta^{(m)}\right\}, \quad m \geq 1, m \in \mathbb{Z} \tag{6.13}
\end{equation*}
$$

Under such assumption, it is possible to compute the steepest descent direction in eq. (6.12) efficiently.

Theorem 6.2. Let $\Delta^{m}$ be the m-dimensional simplex i.e., $\alpha \geq 0, \sum_{i=1}^{m} \alpha_{i}=1$. Suppose that $\mathscr{S}^{\star}(\theta)=\mathscr{S}_{m}^{\star}(\theta):=\left\{\delta^{(1)}, \ldots, \delta^{(m)}\right\}$ and denote by $\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right)$ the matrix with columns $\nabla_{\theta} g\left(\theta, \delta^{(i)}\right)$ for $i=1, \ldots, m$. As long as $\theta$ is not a local minimizer of the robust loss $\phi(\theta)=\max _{\delta \in \mathscr{S}} g(\theta, \delta)$, then the steepest descent direction of $\phi$ at $\theta$ can be computed as:

$$
\begin{equation*}
\gamma^{\star}:=-\frac{\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha^{\star}}{\left\|\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha^{\star}\right\|}, \quad \alpha^{\star} \in \underset{\alpha \in \Delta^{m}}{\operatorname{argmin}}\left\|\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha\right\|_{2}^{2} \tag{6.14}
\end{equation*}
$$

We present the proof of theorem 6.2 in section 6.9. We now relax our initial finiteness assumption eq. (6.13), as it might not hold in practice. We show that it might suffice to approximate the (possibly infinite) set of maximizers $\mathscr{S}^{\star}(\theta)$ with a finite set $\mathscr{S}_{m}^{\star}(\theta)$. If the direction $\gamma^{\star}$ defined in eq. (6.14) satisfies an additional inequality involving the finite set $\mathscr{S}_{m}^{\star}(\theta)$, it will be a certified descent direction.

Theorem 6.3. Suppose that $\nabla_{\theta} g(\theta, \delta)$ is L-Lipschitz as a function of $\delta$, i.e., $\left\|\nabla_{\theta} g(\theta, \delta)-\nabla_{\theta} g\left(\theta, \delta^{\prime}\right)\right\|_{2} \leq$ $L\left\|\delta-\delta^{\prime}\right\|_{2}$. Let $\mathscr{S}^{\star}(\theta)$ be the set of solutions of the inner maximization problem, and let $\mathscr{S}_{m}^{\star}(\theta):=\left\{\delta^{(1)}, \ldots, \delta^{(m)}\right\}$ be a finite set that $\epsilon$-approximates $\mathscr{S}^{\star}(\theta)$ in the following sense: for any $\delta \in \mathscr{S}^{\star}(\theta)$ there exists $\delta^{(i)} \in \mathscr{S}_{m}^{\star}(\theta)$ such that $\left\|\delta-\delta^{(i)}\right\|_{2} \leq \epsilon$. Let $\gamma^{\star}$ be as in eq. (6.14). If $\max _{\delta \in \mathscr{S}_{m}^{\star}(\theta)}\left\langle\gamma^{\star}, \nabla_{\theta} g(\theta, \delta)\right\rangle<-$ Le then $\gamma^{\star}$ is a descent direction for $\phi$ at $\theta$.

The Lipschitz gradient assumption in theorem 6.3 is standard in the optimization literature. We provide a proof of theorem 6.3 in section 6.10. This results motivate Danskin's Descent Direction (algorithm 6.1). We assume an oracle providing a finite set of adversarial perturbations $\mathscr{S}_{m}^{\star}(\theta)$ that satifies the approximation assumption in theorem 6.3. In particular, this does not require solving the inner-maximization problem to optimality, which is out of reach

## Chapter 6. Finding Actual Descent Directions for Adversarial Training

for computationally tractable methods and requires expensive branch-and-bound or MIP techniques (Zhang et al., 2022a; Tjeng et al., 2019; Palma et al., 2021; Wang et al., 2021). Given $\mathscr{S}_{m}^{\star}(\theta)$, we compute $\gamma^{\star}$ as in eq. (6.14), which corresponds to algorithm 6.1 of algorithm 6.1. If the values of $L$ and $\epsilon$ in theorem 6.3 are not available (they might be hard to compute), we cannot certify that $\gamma^{\star}$ is a descent direction. However, note that given a set of adversarial examples $\mathscr{S}_{m}^{\star}(\theta), \gamma^{\star}$ is still the best choice as it ensures we improve the loss on all elements of $\mathscr{S}_{m}^{\star}(\theta)$.

The optimization problem defining $\alpha^{\star}$ and $\gamma^{\star}$ can be solved to arbitrary accuracy efficiently: It corresponds to the minimization of a smooth objective subject to the convex constraint $\alpha \in \Delta^{m}$. We use the accelerated PGD algorithm proposed in (Parikh et al., 2014, section 4.3) and pair it with the efficient simplex projection algorithm given in Duchi et al. (2008a). As the problem is smooth, a fixed step-size choice guarantees convergence. We set it as the inverse of the spectral norm of $\nabla_{\theta} g\left(\theta, \mathscr{S}^{\star}(\theta)\right)^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}^{\star}(\theta)\right)$ and run the algorithm for a fixed number of iterations. Alternatively, one can consider Frank-Wolfe with away steps (Lacoste-Julien and Jaggi, 2015).

Remark 6.1. In general, in the case of Deep Neural Netwworks, the set $\mathscr{S}_{m}^{\star}(\theta)$ cannot be computed, as the maximization problem is non-concave. Moreover, the dimension of the variable $\delta$ is fixed and hence one cannot employ overparametrization arguments to show convergence of gradient ascent to an optimal solution (Neyshabur et al., 2019b) . Vanishing suboptimality has only been shown in the severely restricted scenario of two-layer networks with LeakyReLU activations (Mianjy and Arora, 2023). Finally, even if we could find an algorithm that generates an optimal adversarial perturbation, this does not guarantee that it can find all optimal adversarial perturbations, a requirement for our results to follow.

In practice, the theoretical oracle algorithm that computes the set $\mathscr{S}_{m}^{\star}(\theta)$ is replaced by heuristics like performing multiple runs of the Fast Gradient Sign Method (FGSM) or Iterative FGSM (Kurakin et al., 2017) (referred to as PGD in Madry et al. (2018a)). In this sense, our proposed practical algorithm in the setting of Deep Neural Networks is a heuristic, as we currently cannot guarantee the optimality of the perturbations $\delta$ generated by gradient ascent. Despite this drawback, our goal is to produce a heuristic motivated by correct theoretical arguments, in hopes that this will lead to better practical performance in empirical benchmarks.

The complexity of an iteration in algorithm 6.1 depends on the choice of heuristic to generate adversarial perturbations. In section 2.7 we explore different choices and how it affects the the practical performance of the method.

```
Algorithm 6.1 Danskin's Descent Direction (DDi)
    Input: Batch size \(k \geq 1\), number of adversarial examples \(m\), initial iterate \(\theta_{0} \in \mathbb{R}^{d}\), number
    of iterations \(T \geq 1\), step-sizes \(\left\{\beta_{t}\right\}_{t=1}^{T}\).
    for \(t=0\) to \(T-1\) do
        Draw \(\left(x_{1}, y_{1}\right), \ldots,\left(x_{k}, y_{k}\right)\) from data distribution \(\mathscr{D}\)
        \(g(\theta, \delta) \leftarrow \frac{1}{k} \sum_{i=1}^{k} L\left(\theta, x+\delta_{i}, y_{i}\right)\)
        \(\delta^{(1)}, \ldots, \delta^{(m)} \leftarrow\) MAXIMIZE \(_{\delta \in \mathscr{S}} g\left(\theta_{t}, \delta\right) \quad \triangleright\) Using a heuristic like PGD
        \(M \leftarrow\left[\nabla_{\theta} g\left(\theta_{t}, \delta^{(i)}\right): i=1, \ldots, m\right] \in \mathbb{R}^{d \times m}\)
        \(\alpha^{\star} \leftarrow\) MINIMIZE \(_{\alpha \in \Delta^{m}}\|M \alpha\|_{2}^{2} \quad \triangleright\) To \(\epsilon\)-suboptimality
        \(\gamma^{\star} \leftarrow \frac{M \alpha^{\star}}{\left\|M \alpha^{\star}\right\|_{2}}\)
        \(\theta_{t+1} \leftarrow \theta_{t}+\beta_{t} \gamma^{\star}\)
    end for
    return \(\theta_{T}\)
```


### 6.5 Experiments

### 6.5.1 Existence of multiple optimal adversarial solutions

This section provides evidence that the set of optimal adversarial examples for a given sample is not a singleton. The hypothesis is tested by using a ResNet-18 pretrained on CIFAR10 and computing multiple randomly initialized PGD-7 attacks for each image with $\varepsilon=\frac{8}{255}$. We compute all pairwise $\ell_{2}$-distances between attacks for a given image and plot a joint histogram for 10 examples in Figure 6.2. There is a clear separation away from zero for all pairwise distances indicating that the attacks are indeed distinct in the input space. Additionally, we plot a histogram over the adversarial losses for each image. An example is provided in Figure 6.2, which is corroborated by similar results for other images (see figure 6.6§section 6.8). We find that the adversarial losses all concentrate with low variance far away from the clean loss. This confirms that all perturbations are in fact both strong and distinct.


Figure 6.2: Non-uniqueness of an optimal adversarial perturbation. (left) Pairwise $\ell_{2}$-distances between PGD-based perturbations are bounded away from zero by a large margin, showing that they are distinct. (right) The losses of multiple perturbations on the same sample concentrate around a value much larger than the clean loss (see figure 6.7 for zoomed-in version).

### 6.5.2 Exploring the Optimization Landscape of DDi and Standard Adversarial Training

Having established that there exist multipe adversarial examples, we now show that the gradients computed can exhibit the behaviors discussed in Section 6.3. In a first synthetic example we borrow from (Orabona, 2019, Chapter 6), we consider the function $g(\theta, \delta)=$ $\delta\left(\theta_{1}^{2}+\left(\theta_{2}+1\right)^{2}\right)+(1-\delta)\left(\theta_{1}^{2}+\left(\theta_{2}-1\right)^{2}\right)$ where $\theta \in \mathbb{R}^{2}$ and $\delta \in[0,1]$. As can be seen from Figure 6.1a and Figure 6.1b, following a gradient computed at a single example leads to a increase in the objective and an unstable optimization behavior despite the use of a decaying step-size.

In a second synthetic examples, we consider robust binary classification with a feed-forward neural network on a synthetic 2-dimensional dataset, trained with batch gradient descent. We observe that during training, after an initial phase where all gradients computed at different perturbations point roughly in the same direction, we begin to observe pairs of gradients with negative inner-products (see Figure 6.3 (left)). That means that following one of those gradients would lead to an increase of the robust loss, as shown by the different optimization behavior (see Figure 6.3 (center)). Therefore, the benefits DDi kick in later in training, once the loss has stabilized and the inner-solver starts outputting gradients with negative inner products. Indeed, we see that in the middle of training (iteration 250), DDi finds a descent direction of the (linearized) robust objective, whereas all individual gradients lead to an increase.


Figure 6.3: Count of negative inner products pairs among the 10 gradients computed per iteration(left), corresponding robust loss behavior along optimization (center). At iteration 250, comparison of the direction obtained by DDi and individual gradients.(right).

### 6.5.3 Accuracy/Robustness comparison of DDi vs Adversarial Training

We compare the robust test and training error of Adversarial Training vs our proposed method DDi, on the CIFAR10 benchmark. As baseline we use $\ell_{\infty}$-PGD with $\epsilon=8 / 255, \alpha=2 / 255, n_{\text {inner }}=$ 7. We train a ResNet18 with SGD, using the settings from Pang et al. (2021), Table 1 except for some modifications noted below. This means SGD with hyperparameters $\operatorname{lr}=0.1$, momentum=0.0 (not the default 0.9, we explain why below), batch_size= 128 and weight_decay= $5 e-4$. We run for 200 epochs, no warmup, decreasing lr by a factor of 0.1 at $50 \%$ and $75 \%$ of the epochs.


Figure 6.4: (left) Evolution of the robust accuracy on the CIFAR10 validation set, using a standard PGD-20 adversary for evaluation and DDi/PGD-7 during training. (right) an ablation testing the effect of adding the elements not covered by theory (BN,ReLU,momentum) back into our setting.

Satisfying theoretical assumptions:Real world architectures are often not covered by theory while simple toy examples are often far removed from practice. To demonstrate the real world impact of our results, we therefore study a setting where the conditions of Danskin's Theorem hold, but which also uses standard building blocks used by practitioners, specifically replacing ReLU with CELU(Barron, 2017), replacing BatchNorm (BN) (Ioffe and Szegedy, 2015) with GroupNorm (GN) (Wu and He, 2018) and removing momentum. This ensures differentiability, removes intra-batch dependencies and ensures each update depends only on the descent direction found at that step respecively. We present more detailed justification in subsection 6.8.2 due to space constraints and additionally show an ablation study on the effect of our modifications in (subsection 6.5.3) ${ }^{1}$.

Our main results can be seen in subsection 6.5.3. The robust accuracy of the DDi-trained model increases much more rapidly in the early stages, it increases more after the first drop in the learning rate, and is more stable when compared to the baseline. Subsection 6.5.3 also gives evidence that our method has (generally positive or neutral) effects in all settings. Using ReLU instead of CELU re-introduces the characteristic bump in robust accuracy that has led to early stopping becoming standard practice in robust training. It also diminishes the benefit of DDi, but DDi remains on par with PGD in terms of training speed and decays slightly less towards the end of the training. Adding momentum does not help either method in terms of training speed and makes them behave almost identically.

Finally, BN seems to significantly ease the optimisation for both methods, raising overall performance and amplifying the bump on both methods. Here, PGD actually reaches a higher maximum robust accuracy and rises faster initially, but then converges to a lower value. This implies that some benefits of DDi remain even outside the setting covered by the theory.

[^9]
## Chapter 6. Finding Actual Descent Directions for Adversarial Training

Although these are promising results indicating that DDi can give real world benefits in terms of iterations and reduce the need for early stopping, it is worth asking whether once could get the same benefit with a simpler or cheaper method. The final robust accuracies obtained are very close, and the increased convergence rate in terms of steps comes at a more than 10x slowdown due to having to perform 10 independent forward-backward passes and then solving an additional inner problem. Additionally, it could be argued that these results are to be expected and trivial: we are spending 10 x the compute to get 10 x the gradients.

One might even say there is no need to solve the inner product and a simpler method to select the best adversary would suffice. In figure 6.5 a we address these concerns by comparing subsection 6.5.3 to the results of the following variants attempting to match the computational complexity: PGD-70 runs a single PGD adversary for 10 x the number of steps, $P G D-70-\frac{1}{t}$ runs a single PGD adversary for 10x the number of steps, using a $1 / t$ learning rate decay after leaving the "standard" PGD regime (i.e. after 8 adversary steps) to converge closer to an optimal adversarial example, PGD-max-10 runs ten parallel, independent PGD adversaries for each image and select the adversarial example that induces the largest loss. Finally, PGD-min10 runs ten parallel, independent PGD adversaries for each image, then computes the gradients and selects the one with the lowest norm. This is an approximation of DDi that avoids solving algorithm 6.1 in algorithm 6.1.

In figure 6.5b we create a DDi variant based on the FAST adversary (Wong et al., 2020) (using $\epsilon=$ $8 / 255, \alpha=10 / 255$ ). Using PGD for the evaluation attack, we compare against vanilla FAST in our setting (no BN, momentum and using CELU) as well as a FAST-max-10 variant analoguous to PGD-max-10. As we can see in figure 6.5a, every step of the pipeline of DDi seems to be necessary, with none of the PGD variants achieving the fast initial rise in robustness. PGD-$70-\frac{1}{t}$ and PGD-min-10 reach a higher final robust accuracy, which we attribute to the higher quality adversarial example and informed selection respectively. This is corroborated in figure 6.5b. Using a single step adversary is sufficient to speed up convergence in the early stages of training, but does not reach the same final robust accuracy.

PGD and DDi seem to behave similarly in the later stages of training. We would suggest a computationally cheaper DDi variant which uses single ascent steps (FAST) in the beginning of training and PGD in the later stages. In any case, the bulk of the overhead lies in the subroutine in algorithm 6.1 of algorithm 6.1. A faster approximate solution could also speed up the method significantly. Such incremental improvements are left for future work Neverthelss, in subsection 6.8 .4 we explore some modifications that can reduce the runtime of algorithm 6.1 by at least $70 \%$ while retaining its benefits.

### 6.6 Related Work

Wang et al. (2019) derive suboptimality bounds for the robust training problem, under a locally strong concavity assumption on the inner-maximization problem. However, such results do not extend to Neural Networks, as the inner-maximization problem is not strongly concave, in


Figure 6.5: (a) Ablations comparing PGD-variants matching the number of adversarial gradients/steps used for DDi. (b) Ablation over single-step adversaries (FAST/DDi-FAST).
general. In contrast, we do not make unrealistic assumptions like strong concavity, and we deal with the existence multiple solutions of the inner-maximization problem.

In Nouiehed et al. (2019), it is shown that if the inner-maximization problem is unconstrained and satisfies the PL-condition, it is differentiable, and the gradient can be computed after obtaining a single solution of the problem. However, in the robust learning problem the adversary is usually constrained to a compact set, and the PL condition does not hold generically. This renders such assumptions hard to justify in the AT setting.

Tramer and Boneh (2019); Maini et al. (2020) study robustness to multiple perturbation types, which might appear similar to our approach, but is not. Such works strike to train models that are simultaneously robust against $\ell_{\infty^{-}}$and $\ell_{2}$-bounded perturbations, for example. In contrast, we focus on a single perturbation type, and we study how to use multiple adversarial examples of the same sample to improve the update directions of the network parameters.

Finally, we back our claim that the falseness of Madry et al. (2018a, Corollary C.2.) is not well-known in the literature on Adversarial Training. For example, such result is included in the textbook (Vorobeychik et al., 2018, Proposition 8.1). It has also been either reproduced or mentioned in conference papers like Liu et al. (2020, Section 2), Viallard et al. (2021, Appendix B), Wei and Ma (2020, Section 5) and possibly many others. This supports our claim that raising awareness about the mistake in the proof is an important contribution.

### 6.7 Appendix: More on counterexamples

Here we give more details on the construction of the counterexamples. First observe that for a given point $\theta_{0}$, and a direction $\gamma$, if there exists a $\delta_{0} \in \mathscr{S}^{\star}\left(\theta_{0}\right)$ such that $\left\langle\gamma, \nabla_{\theta} g\left(\theta_{0}, \delta\right)\right\rangle>0$, then $\gamma$ is not a descent direction since $D_{\gamma} \phi\left(\theta_{0}\right) \geq 0$.

In order to ensure that no descent directions can be recovered by solving the inner-maximization, it suffices to guarantee that for any $\delta \in \mathscr{S}^{\star}\left(\theta_{0}\right)$, there exists $\delta^{\prime} \in \mathscr{S}^{\star}\left(\theta_{0}\right)$ such that

$$
\begin{equation*}
\left\langle\nabla_{\theta} g\left(\theta_{0}, \delta^{\prime}\right), \nabla_{\theta} g\left(\theta_{0}, \delta\right)\right\rangle<0 \tag{6.15}
\end{equation*}
$$

This way, neither $-\nabla_{\theta} g\left(\theta_{0}, \delta\right)$ nor $-\nabla_{\theta} g\left(\theta_{0}, \delta^{\prime}\right)$ would be descent directions.
It easy to generate instances verifying the above using linear functions. More formally, by taking any family of vectors $\mathcal{V}=\left\{v_{1}, \ldots, v_{n}\right\}$ such that for any $i \in\{1, \ldots, n\}$ there exists $j \in\{1, \ldots, n\}$ such that $\left\langle v_{i}, v_{j}\right\rangle<0$, we can construct the objective $g(\theta, \delta)=\sum \delta_{i} v_{i}^{\top}\left(\theta-\theta_{0}\right)-H(\delta)$, where $\delta$ is in the $n$-dimensional Simplex and $H$ is the Shannon entropy. Solving the inner-maximization would yield any one of the vectors $\left\{v_{1}, \ldots, v_{n}\right\}$, and by construction, none of them are descent directions.

### 6.8 Appendix: Experiments

### 6.8.1 Multiple attacks



Figure 6.6: The losses of multiple perturbations on 9 different example all concentrate around a value much larger than the clean loss. See Subsection 6.5.1 for experimental details. The histograms have been enlarged in Figure 6.7.


Figure 6.7: The losses of multiple perturbations on 9 different example all concentrate around a value much larger than the clean loss (see Figure 6.6 for comparison with the clean loss).

### 6.8.2 Justifying our modifications

For Danskin's Theorem theorem 6.1 to hold, we require the function to be differentiable. To satisfy differentiability, we replace ReLU with CELU (Barron, 2017) , which has been found to have comparable performance and sometimes outperform ReLU (Dubey et al., 2022).

To operate on individual images and remove the batch-wise correlations across samples we replace BatchNorm (BN) (Ioffe and Szegedy, 2015) with GroupNorm (GN) (Wu and He, 2018) ${ }^{2}$.

Finally, to make each update depend only on the current state, we set momentum $=0.0$. Since momentum is standard practice in the CV community and works like Yan et al. (2018) argue that it can improve generalisation, we rely on our ablation to show that removing it is safe.

[^10]
### 6.8.3 Further details on synthetic experiments

The synthetic experiment in figure 6.1 a is conducted with the following settings. The innermaximization is approximated with 10 steps of projected gradient ascent in order to match the traditional AT setting. The outer iterations have a decaying $\frac{0.5}{\sqrt{k}}$ step-size schedule. We observe the same erratic behavior for PGD with a fixed outer stepsize, while DDi consitently remains well-behaved.

The synthetic experiment in figure 6.3 is conducted on a dataset of size 100 in dimension 2 where the coordinates are standard Gaussian. The neural network is a 2-layer network with ELU activation with a hidden layer of width 2. The inner solver is PGD with 10 steps with stepsize 0.1 and optimizes over the unit cube. The outer step-size is 0.01 and the weights are optimized with full batch gradient descent.

The linear approximation at iteration 250 of the robust loss consits of taking the 10 adversarial examples computed at iteration 250 and approximating it with

$$
\tilde{\phi}(\theta)=\max _{\delta_{1} \ldots \delta_{10}} \phi\left(\theta_{250}\right)+\left\langle\nabla_{\theta} g\left(\theta_{250}, \delta_{i}\right), \theta-\theta_{250}\right\rangle
$$

Interestingly we do not observe the same drastic improvement over PGD when observing the non-linearized loss at iteration 250.


### 6.8.4 Improving the running time

While the focus of this paper is not to obtain a state-of-the-art method, it does matter whether it is feasible to efficiently capture the benefit of DDi. The naive implementation has about a 10 - 12 times overhead compared to PGD, mainly due to three bottlenecks (in descending impact)

1. for $k$ - DDi , generating $k$ adversarial examples with PGD as the base attack involves a $k$ times overhead


Figure 6.8: (a) Epoch evolution of a more efficient implementation of DDi. (b) Wallclock evolution of the same methods.
2. then $k$ separate gradient samples need to be computed on these adversarial examples, which involes $k$ forward-backward passes
3. finally, one additional optimization problem needs to be solved.

While steps 1) and 2) can be somewhat parallelized, they still cause a massive increase in compute and memory. We therefore adopt two heuristic approaches to speed up the algorithm while (hopefully) maintaining it's benefits:

1. since later in training the benefits of DDi appear to diminish, we linearly decay the number of gradients sampled $k$ from 10 down to 1 along the 200 epochs (referred to as decay)
2. we also adopt a method of creating $k$ unique batches from only 2 independent adversarial attacks (described below in subsection 6.8.4, referred to as comb).

We evaluate this method using both PGD and FAST as base attacks and show the results in figure 6.8a and figure 6.8b. As can be seen, DA-PGD-decay-comb and DA-PGD-comb both enjoy a massive speedup in wallclock time (reducing the $12 \times$ overhead to about $3 \times$ ) while retaining the improved per-step progress of base $D D i$.

## Combinatorial batch construction

Suppose we have a batch of data-label pairs $\left(x_{i}, y_{i}\right)$ of size $B$. In order to construct $k \leq 2^{B}$ different gradients by computing only 2 adversarial examples per data sample $x_{i}$ in the batch we do the following:

1. for each $i=1, \ldots, B$ compute $\delta_{i, 0}, \delta_{i, 1}$ two adversarial examples using the data-label pair $\left(x_{i}, y_{i}\right)$ in the batch.
2. for each $j=1, \ldots, k$ repeat the following steps:
3. Define $\Delta=[]$ as an empty list.
4. generate a random bitvector $b \subseteq\{0,1\}^{B}$ of length $B$
5. when $b_{i}$ is 0 we append $\delta_{i, 0}$ to $\Delta$, otherwise when $b_{i}$ is 1 we append $\delta_{i, 1}$ to $\Delta$.
6. compute the gradient w.r.t. the network parameters using the perturbations in $\Delta$

While this still incurs overhead of computing $k$ gradients, it greatly reduces running time as seen in figure 6.8b and could further improved by e.g. reusing gradients from past epochs to construct the examples.

### 6.9 Appendix: Proof of theorem 6.2.

The steepest descent direction is computed, following eq. (6.4) as:

$$
\begin{equation*}
\gamma^{\star} \in \underset{\gamma:\|\gamma\|_{2}=1}{\operatorname{argmin}} D_{\gamma} \phi(\theta)=\underset{\gamma:\|\gamma\|_{2}=1}{\operatorname{argmin}} \max _{\delta \in \mathscr{S}_{m}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle \tag{6.16}
\end{equation*}
$$

Whenever $\theta$ is not a local optimum, there exists a non-zero descent direction. In this case we can relax the constraint that $\|\gamma\|_{2}=1$ to $\|\gamma\|_{2} \leq 1$ without changing the solutions or optimal value of (6.16), which is strictly negative:

$$
\begin{equation*}
\min _{\gamma:\|\gamma\|_{2}=1} \max _{\delta \in \mathscr{S}_{m}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle=\min _{\gamma:\|\gamma\|_{2} \leq 1} \max _{\delta \in \mathscr{S}_{m}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle<0 \tag{6.17}
\end{equation*}
$$

We can now transform (6.16) into a bilinear convex-concave min-max problem, subject to convex and compact constraints:

$$
\begin{align*}
\gamma^{\star} \in \underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} D_{\gamma} \phi(\theta) & =\underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} \max _{\delta \in \mathscr{S}_{m}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle \\
& =\underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} \max _{i=1, \ldots, m} \gamma^{\top} \nabla_{\theta} g\left(\theta, \delta^{(i)}\right)  \tag{6.18}\\
& =\underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} \max _{\alpha \in \Delta^{m}} \gamma^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha
\end{align*}
$$

By Sion's minimax Theorem Sion (1958), we can solve eq. (6.18) by swapping the operator order:

$$
\begin{align*}
\min _{\gamma:\|\gamma\|_{2} \leq 1} \max _{\alpha \in \Delta^{m}} \gamma^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha & =\max _{\alpha \in \Delta^{m}} \min _{\gamma:\|\gamma\|_{2} \leq 1} \gamma^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha \\
& =\max _{\alpha \in \Delta^{m}}-\left\|\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha\right\|_{2}  \tag{6.19}\\
& =-\min _{\alpha \in \Delta^{m}}\left\|\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha\right\|_{2}<0
\end{align*}
$$

Finally, by noting that squaring the objective function in the right-hand side of eq. (6.19) does not change the set of solutions, we arrive at the formula for $\alpha^{\star}$ in eq. (6.14). Indeed for a solution $\alpha^{\star}$ to this problem we have

$$
\begin{align*}
\underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} \max _{\alpha \in \Delta^{m}} \gamma^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha & =\underset{\gamma:\|\gamma\|_{2} \leq 1}{\operatorname{argmin}} \gamma^{\top} \nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha^{\star} \\
& =-\frac{\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha^{\star}}{\left\|\nabla_{\theta} g\left(\theta, \mathscr{S}_{m}^{\star}(\theta)\right) \alpha^{\star}\right\|} \tag{6.20}
\end{align*}
$$

where the denominator is nonnegative as the optimal objective value is nonzero c.f. eq. (6.19).

### 6.10 Appendix: Proof of theorem 6.3.

For any $\delta \in \mathscr{S}^{\star}(\theta)$ let $i(\delta) \in\{1, \ldots, m\}$ be such that $\left\|\delta^{(i(\delta))}-\delta\right\|_{2} \leq \epsilon$. That is, we map any maximizer $\delta$ to an index $i \in\{1, \ldots, m\}$ such that the corresponding perturbation $\delta^{(i)}$ in the finite set $\mathscr{S}_{m}^{\star}(\theta)$ is at most at an $\epsilon$ distance. This map can be constructed by the assumption on $\mathscr{S}_{m}^{\star}(\theta)$.

For any $\gamma$ such that $\|\gamma\|_{2}=1$ we have

$$
\begin{align*}
\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle & =\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)-\nabla_{\theta} g\left(\theta, \delta^{(i(\delta))}\right\rangle+\left\langle\gamma, \nabla_{\theta} g\left(\theta, \delta^{(i(\delta))}\right)\right\rangle\right. \\
& \leq \underbrace{\|\gamma\|_{2}\|\underbrace{}_{\Delta}\| \nabla_{\theta} g(\theta, \delta)-\nabla_{\theta} g\left(\theta, \delta^{(i(\delta))}\right) \|}_{=1}+\left\langle\gamma, \nabla_{\theta} g\left(\theta, \delta^{(i(\delta))}\right)\right\rangle  \tag{6.21}\\
& \leq\left\langle\gamma, \nabla_{\theta} g\left(\theta, \delta^{L i(\delta \delta))}\right)\right\rangle+L \epsilon \\
& \leq \sup _{\delta \in \mathcal{S}_{m}^{*}(\theta)}\left\langle\gamma, \nabla_{\theta} g\left(\theta, \delta^{(i)}\right)\right\rangle+L \epsilon
\end{align*}
$$

Taking the supremum over $\delta \in \mathscr{S}^{\star}(\theta)$ on the left-hand-side we obtain

$$
\begin{equation*}
D_{\gamma} \phi(\theta):=\sup _{\delta \in \mathscr{S}^{\star}(\theta)}\left\langle\gamma, \nabla_{\theta} g(\theta, \delta)\right\rangle \leq \sup _{\delta \in \mathscr{S}_{m}^{*}(\theta)}\left\langle\gamma, \nabla_{\theta} g\left(\theta, \delta^{(i)}\right)\right\rangle+L \epsilon \tag{6.22}
\end{equation*}
$$

Hence if the supremum on the right-hand-side is strictly smaller than $-L \varepsilon$ we have that $D_{\gamma} \phi(\theta)<0$, which yields the desired result.

### 6.11 Appendix: Proof of lemma 6.1

Assume the limit that defines $\hat{D}_{\gamma} \phi(\theta)$ exists (and is finite).

$$
\begin{align*}
\hat{D}_{-\gamma} \phi(\theta) & =\lim _{t \rightarrow 0} \frac{\phi(\theta+t(-\gamma))-\phi(\theta)}{t\|-\gamma\|_{2}} \\
& =\lim _{t \rightarrow 0} \frac{\phi(\theta+(-t) \gamma)-\phi(\theta)}{-(-t)\|\gamma\|_{2}} \\
& =\lim _{(-t) \rightarrow 0} \frac{\phi(\theta+(-t) \gamma)-\phi(\theta)}{-(-t)\|\gamma\|_{2}}  \tag{6.23}\\
& =\lim _{s \rightarrow 0}-\frac{\phi(\theta+s \gamma)-\phi(\theta)}{s\|\gamma\|_{2}} \quad(\text { let } s=(-t)) \\
& =-\lim _{s \rightarrow 0} \frac{\phi(\theta+s \gamma)-\phi(\theta)}{s\|\gamma\|_{2}} \\
& =-\hat{D}_{\gamma} \phi(\theta)
\end{align*}
$$

### 6.12 Bibliographic Note

The candidate came up with the original idea and derived all the theoretical results (theorems 6.2 and 6.3 ) as well as counterexample 6.1. Counterexample 6.2 was derived by the candidate and L . Dadi. The candidate proposed the structure of algorithms and designed some of the experiments.

# 7 Adversarial Training Should Be Cast As a Non-Zero-Sum Game 

Alexander Robey, Fabian Latorre, George Pappas, Hamed Hassani, Volkan Cevher. Under submission, 2023.


#### Abstract

One prominent approach toward resolving the adversarial vulnerability of deep neural networks is the two-player zero-sum paradigm of adversarial training, in which predictors are trained against adversarially-chosen perturbations of data. Despite the promise of this approach, algorithms based on this paradigm have not engendered sufficient levels of robustness, and suffer from pathological behaviour like robust overfitting. To understand this shortcoming, we first show that the commonly used surrogate-based relaxation used in adversarial training algorithms voids all guarantees on the robustness of trained classifiers. The identification of this pitfall informs a novel non-zero-sum bilevel formulation of adversarial training, wherein each player optimizes a different objective function. Our formulation naturally yields a simple algorithmic framework that matches and in some cases outperforms state-of-the-art attacks, attains comparable levels of robustness to standard adversarial training algorithms, and does not suffer from robust overfitting.


### 7.1 Introduction

A longstanding disappointment in the machine learning (ML) community is that deep neural networks (DNNs) remain vulnerable to seemingly innocuous changes to their input data including nuisances in visual data (Robey et al., 2020; Hendrycks and Dietterich, 2019; Hendrycks et al., 2021; Eykholt et al., 2018b), sub-populations (Santurkar et al., 2021; Sohoni et al., 2020; Koh et al., 2021), and distribution shifts (Xiao et al., 2021; Arjovsky et al., 2019; Sagawa et al., 2020; Robey et al., 2021b). Prominent amongst these vulnerabilities is the setting of adversarial examples, wherein it has been conclusively shown that imperceptible, adversarially-chosen perturbations can fool state-of-the-art classifiers parameterized by DNNs (Szegedy et al., 2013; Biggio et al., 2013b, 2012; Carlini and Wagner, 2017; Biggio et al., 2013a). In response, a plethora of research has proposed so-called adversarial training (AT) algorithms (Huang et al., 2015; Wong and Kolter, 2018b; Kurakin et al., 2017; Madry et al., 2018a; Goodfellow et al., 2015),
which are designed to improve robustness against adversarial examples.
AT is ubiquitously formulated as a two-player zero-sum game, where both players-often referred to as the defender and the adversary-respectively seek to minimize and maximize the classification error. However, this zero-sum game is not implementable in practice as the discontinuous nature of the classification error is not compatible with first-order optimization algorithms. To bridge this gap between theory and practice, it is commonplace to replace the classification error with a smooth surrogate loss (e.g., the cross-entropy loss) which is amenable to gradient-based optimization (Madry et al., 2018a; Zhang et al., 2019b). And while this seemingly harmless modification has a decades-long tradition in the ML literature due to the guarantees it imparts on non-adversarial objectives (Bartlett et al., 2006; Shalev-Shwartz and Ben-David, 2014b; nicolas le roux, 2017), there is a pronounced gap in the literature regarding the implications of this relaxation on the standard formulation of AT.

As the field of robust ML has matured, surrogate-based AT algorithms (see, e.g., Madry et al. (2018a); Zhang et al. (2019b); Goodfellow et al. (2015); Wang et al. (2020)) have collectively ushered in significant progress toward designing stronger attacks and obtaining more robust defenses (Croce et al., 2020a). However, despite these advances, recent years have witnessed a plateau in robustness measures on leaderboards such as RobustBench, resulting in the widely held beliefs that robustness and accuracy may be irreconcilable (Tsipras et al., 2019; Dobriban et al., 2020; Javanmard et al., 2020) and that robust generalization requires significantly more data (Schmidt et al., 2018; Chen et al., 2020a; Stutz et al., 2019). Moreover, various phenomena such as robust overfitting (Rice et al., 2020) and insufficient robustness evaluation (Carlini et al., 2019) have indicated that progress has been overestimated (Croce and Hein, 2020b). To combat these pitfalls, state-of-the-art algorithms increasingly rely on ad-hoc regularization schemes (Kannan et al., 2018; Zhang et al., 2019b; Chan et al., 2020; Hoffman et al., 2019; Finlay et al., 2018), weight perturbations (Wu et al., 2020; Sun et al., 2021; Foret et al., 2021), and heuristics such as multiple restarts (Madry et al., 2018a), carefully crafted learning rate schedules (Rice et al., 2020), and convoluted stopping conditions (Croce and Hein, 2020b), all of which contribute to an unclear set of best practices and a growing literature concerned with identifying flaws in various AT schemes (Latorre et al., 2023).

Motivated by these challenges, we argue that the pervasive surrogate-based zero-sum approach to AT suffers from a fundamental flaw. Our analysis of the standard minimax formulation of AT reveals that maximizing a surrogate like the cross-entropy provides no guarantee that the the classification error will increase, resulting in weak adversaries and ineffective AT algorithms. In identifying this shortcoming, we prove that to preserve guarantees on the optimality of the classification error objective, the defender and the adversary must optimize different objectives, resulting in a non-zero-sum game. This leads to a novel, yet natural bilevel formulation (Bard, 2013) of AT in which the defender minimizes an upper bound on the classification error, while the attacker maximizes a continuous reformulation of the classification error. We then propose an algorithm based on our formulation which is free from heuristics and ad hoc optimization techniques. Our empirical evaluations reveal that our approach
matches the test robustness achieved by the state-of-the-art, yet highly heuristic approaches such as AutoAttack, and that it eliminates the problem of robust overfitting.

Contributions. We summarize our contributions in the following bullets.
$\triangleright$ New formulation for adversarial robustness. Starting from the discontinuous minmax formulation of AT with respect to the 0-1 loss, we derive a novel continuous bilevel optimization formulation, the solution of which guarantees improved robustness against the optimal adversary.
$\triangleright$ New adversarial training algorithm. We derive a new, heuristic-free algorithm (Algorithm 7.2) based on our bilevel formulation, and show that offers strong robustness on CIFAR-10.
$\triangleright$ Elimination of robust overfitting. Without the need of heuristic modifications, our algorithm does not suffer from robust overfitting (RO). This suggest that RO is an artifact of the use of improper surrogates in the original AT paradigm, and that the use of a correct optimization formulation is enough to solve it.
$\triangleright$ State-of-the-art robustness evaluation. We show that our proposed optimization objective for the adversary yields a simple algorithm that matches the performance of the state-of-the-art, yet highly complex AutoAttack method, on classifiers trained on CIFAR-10.

### 7.2 The promises and pitfalls of adversarial training

### 7.2.1 Preliminaries: Training DNNs with surrogate losses

We consider a $k$-way classification setting, wherein data arrives in the form of instancelabel pairs $(X, Y)$ drawn i.i.d. from an unknown joint distribution $\mathscr{D}$ taking support over $\mathscr{X} \times \mathscr{Y} \subseteq \mathbb{R}^{d} \times[K]$, where $[K]:=\{1, \ldots, K\}$. Given a suitable hypothesis class $\mathscr{F}$, one fundamental goal in this setting is to select an element $f \in \mathscr{F}$ which correctly predicts the label $Y$ of a corresponding instance $X$. In practice, this hypothesis class $\mathscr{F}$ often comprises functions $f_{\theta}: \mathbb{R}^{d} \rightarrow \mathbb{R}^{K}$ which are parameterized by a vector $\theta \in \Theta \subset \mathbb{R}^{p}$, as is the case when training DNNs. In this scenario, the problem of learning a classifier that correctly predicts $Y$ from $X$ can written as follows:

$$
\begin{equation*}
\min _{\theta \in \Theta} \mathbb{E}\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X)_{i} \neq Y\right\} \tag{7.1}
\end{equation*}
$$

Here $f_{\theta}(X)_{i}$ denotes the $i^{\text {th }}$ component of the logits vector $f_{\theta}(X) \in \mathbb{R}^{K}$ and we use the notation $\{A\}$ to denote the indicator function of an event $A$, i.e., $\{A\}:=\mathbb{\square}_{A}(\cdot)$. In this sense, $\left\{\operatorname{argmax} f_{\theta}(X)_{i} \neq Y\right\}$ denotes the classification error of $f_{\theta}$ on the pair $(X, Y)$. $i \in[K]$

Prominent among the barriers to solving (7.1) in practice is the fact that the classification error is a discontinuous function of $\theta$, which in turn renders continuous first-order methods
(e.g., gradient descent) intractable. Fortunately, this pitfall can be resolved by minimizing a surrogate loss function $\ell:[k] \times[k] \rightarrow \mathbb{R}$ in place of the classification error (Shalev-Shwartz and Ben-David, 2014b, §12.3). For minimization problems, surrogate losses are chosen to be differentiable upper bounds of the classification error of $f_{\theta}$, in the sense that

$$
\begin{equation*}
\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X)_{i} \neq Y\right\} \leq \ell\left(f_{\theta}(X), Y\right) \tag{7.2}
\end{equation*}
$$

This inequality gives rise to a differentiable counterpart of (7.1) which is amenable to minimization via first-order methods and can be compactly expressed in the following optimization problem:

$$
\begin{equation*}
\min _{\theta \in \Theta} \mathbb{E} \ell\left(f_{\theta}(X), Y\right) \tag{7.3}
\end{equation*}
$$

Examples of commonly used surrogates are the hinge loss and the cross-entropy loss. Crucially, the inequality in (7.2) guarantees that the problem in (7.3) provides a solution that decreases the classification error (Bartlett et al., 2006), which, as discussed above, is the primary goal in supervised classification.

### 7.2.2 The pervasive setting of adversarial examples

For common hypothesis classes, it is well-known that classifiers obtained by solving (7.3) are sensitive to seemingly benign changes to their input data. Among these vulnerabilities, perhaps the most well-studied is the setting of adversarial examples, wherein a plethora of research has demonstrated that state-of-the-art classifiers can be fooled by small, adversariallychosen perturbations (Szegedy et al., 2013; Biggio et al., 2013b, 2012; Carlini and Wagner, 2017; Biggio et al., 2013c). In other words, given an instance label pair ( $X, Y$ ), it is relatively straightforward to find perturbations $\eta \in \mathbb{R}^{d}$ with small norm $\|\eta\| \leq \epsilon$ for some fixed $\epsilon>0$ such that

$$
\begin{equation*}
\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X)_{i}=Y \quad \text { and } \quad \underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{i} \neq \underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X)_{i} . \tag{7.4}
\end{equation*}
$$

The task of finding such perturbations $\eta$ which cause the classifier $f_{\theta}$ to misclassify perturbed data points $X+\eta$ can be compactly cast as the following maximization problem:

$$
\begin{equation*}
\eta^{\star} \in \underset{\eta:\|\eta\| \leq \epsilon}{\operatorname{argmax}}\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{i} \neq Y\right\} \tag{7.5}
\end{equation*}
$$

Here, if both of the expressions in (7.4) hold for the perturbation $\eta=\eta^{\star}$, then the perturbed instance $X+\eta^{\star}$ is called an adversarial example for $f_{\theta}$ with respect to the instance-label pair ( $X, Y$ ).

Due to prevalence of adversarial examples, there has been pronounced interest in solving the robust analog of (7.1), which is designed to find classifiers that are insensitive to small
perturbations. This robust analog is ubiquitously written as the following a two-player zerosum game with respect to the discontinuous classification error:

$$
\begin{equation*}
\min _{\theta \in \Theta} \mathbb{E}\left[\max _{\eta:\|\eta\| \leq \epsilon}\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{i} \neq Y\right\}\right] \tag{7.6}
\end{equation*}
$$

An optimal solution $\theta^{\star}$ for (7.6) yields a model $f_{\theta^{\star}}$ that achieves the lowest possible classification error despite the presence of adversarial perturbations. For this reason, this problemwherein the interplay between the maximization over $\eta$ and the minimization over $\theta$ comprises a two-player zero-sum game- is the starting point for numerous algorithms which aim to improve robustness.

### 7.2.3 Surrogate-based approaches to robustness

As discussed in $\S 7.2 .1$, the discontinuity of the classification error complicates the task of finding adversarial examples, as in (7.5), and of training against these perturbed instances, as in (7.6). One appealing approach toward overcoming this pitfall is to simply deploy a surrogate loss in place of the classification error inside (7.6), which gives rise to the following pair of optimization problems:

$$
\begin{equation*}
\eta^{\star} \in \underset{\eta:\|\eta\| \leq \varepsilon}{\operatorname{argmax}} \ell\left(f_{\theta}(X+\eta), Y\right) \tag{7.7}
\end{equation*}
$$

$$
\begin{equation*}
\min _{\theta \in \Theta} \mathbb{E}\left[\max _{\eta:\|\eta\| \leq \epsilon} \ell\left(f_{\theta}(X+\eta), Y\right)\right] \tag{7.8}
\end{equation*}
$$

Indeed, this surrogate-based approach is pervasive in practice. Madry et al.'s seminal paper on the subject of adversarial training employs this formulation (Madry et al., 2018a), which has subsequently been used as the starting point for numerous AT schemes (Huang et al., 2015; Wong and Kolter, 2018a; Kurakin et al., 2017; Madry et al., 2018a; Goodfellow et al., 2015).

Pitfalls of surrogate-based optimization. Despite the intuitive appeal of this paradigm, surrogate-based adversarial attacks are known to overestimate robustness (Mosbach et al., 2018; Croce et al., 2020b; Croce and Hein, 2020b), and standard adversarial training algorithms are known to fail against strong attacks. Furthermore, this formulation suffers from pitfalls such as robust overfitting (Rice et al., 2020) and trade-offs between robustness and accuracy (Tsipras et al., 2019). To combat these shortcomings, empirical adversarial attacks and defenses have increasingly relied on heuristics such as multiple restarts, variable learning rate schedules (Croce and Hein, 2020b), and carefully crafted initializations, resulting in a widening gap between the theory and practice of adversarial learning. In the next section, we argue that these pitfalls can be attributed to the fundamental limitations of (7.8).

### 7.3 Non-zero-sum formulation of adversarial training

### 7.3.1 Fundamental limitations of surrogates in adversarial learning

From an optimization perspective, the surrogate-based approaches to adversarial evaluation and training outlined in $\S 7.2 .3$ engenders two fundamental limitations.


#### Abstract

$\triangleright$ Limitation I: Weak attackers. In the adversarial evaluation problem of (7.7), the adversary maximizes an upper bound on the classification error. This means that any solution $\eta^{\star}$ to (7.7) is not guaranteed to increase the classification error in (7.5), resulting in weakened adversaries which are misaligned with the goal of finding adversarial examples that fool DNNs. Indeed, when the surrogate is an upper bound on the classification error, the only conclusion about the adversarial perturbation $\eta^{\star}$ obtained from (7.7) and its true objective (7.5) is:


$$
\begin{equation*}
\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}\left(X+\eta^{\star}\right)_{i} \neq Y\right\} \leq \max _{\eta:\|\eta\| \leq \varepsilon} \ell\left(f_{\theta}(X+\eta), Y\right) \tag{7.9}
\end{equation*}
$$

Notably, the right hand side of (7.9) can be arbitrarily large while the left hand side can simultaneously be equal to zero, i.e., the problem in (7.7) can fail to produce an adversarial example, even at optimality. Thus, while it is known empirically that attacks based on (7.7) tend to overestimate robustness (Croce and Hein, 2020b; Croce et al., 2020b; Gowal et al., 2019), we show that this shortcoming is evident a priori.
$\triangleright$ Limitation II: Ineffective defenders. Because attacks which seek to maximize upper bounds on the classification error are not proper surrogates for the classification error (c.f., Limitation I), training a model $f_{\theta}$ on such perturbations does not guarantee any improvement in robustness. Therefore, AT algorithms which seek to solve (7.8) are ineffective in that they do not optimize the worst-case classification error. For this reason, it should not be surprising that robust overfitting (Rice et al., 2020) occurs for models trained to solve eq. (7.8).

Both of these limitations arise directly by virtue of rewriting (7.7) and (7.8) with the surrogate loss $\ell$. Therefore, to summarize, there is a distinct tension between the efficient, yet misaligned paradigm of surrogate-based adversarial training with the principled, yet intractable paradigm of minimax optimization on the classification error. In the remainder of this section, we resolve this tension by decoupling the optimization problems of the adversary and the training algorithm.

### 7.3.2 Decoupling adversarial attacks and defenses

Our starting point is the two-player zero-sum formulation in (7.6). Observe that this minimax optimization problem can be equivalently cast as a bilevel optimization problem ${ }^{1}$ :

$$
\begin{array}{ll}
\min _{\theta \in \Theta} & \mathbb{E}\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}\left(X+\eta^{\star}\right)_{i} \neq Y\right\} \\
\text { subject to } & \eta^{\star} \in \underset{\eta:\|\eta\| \leq \varepsilon}{\operatorname{argmax}}\left\{\underset{i \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{i} \neq Y\right\} \tag{7.11}
\end{array}
$$

While this problem still constitutes a zero-sum game, the role of the attacker (the constraint in (7.11)) and the role of the defender (the objective in (7.10)) are now decoupled. From this perspective, the tension engendered by introducing surrogate losses is laid bare: the attacker ought to maximize a lower bound of the classification error (c.f., Limitation I), whereas the defender ought to minimize an upper bound on the classification error (c.f., Limitation II). This implies that to preserve guarantees on optimality, the attacker and defender must optimize separate objectives. In what follows, we discuss these objectives for the attacker and defender in detail.

The attacker's objective. We first address the role of the attacker. To do so, we define the negative margin $M_{\theta}(X, Y)$ of the classifier $f_{\theta}$ as follows:

$$
\begin{equation*}
M_{\theta}: \mathscr{X} \times \mathscr{Y} \rightarrow \mathbb{R}^{k}, \quad M_{\theta}(X, Y)_{j} \triangleq f_{\theta}(X)_{j}-f_{\theta}(X)_{Y} \tag{7.12}
\end{equation*}
$$

We call $M_{\theta}(X, Y)$ the negative margin because a positive value of (7.12) corresponds to a misclassification. As we show in the following proposition, the negative margin function (which is differentiable) provides an alternative characterization of the classification error.

Proposition 7.1. Given a fixed data pair $(X, Y)$, let $\eta^{\star}$ denote any maximizer of $M_{\theta}(X+\eta, Y)_{j}$ over the classes $j \in[K]-\{Y\}$ and perturbations $\eta \in \mathbb{R}^{d}$ satisfying $\|\eta\| \leq \epsilon$, i.e.,

$$
\begin{equation*}
\left(j^{\star}, \eta^{\star}\right) \in \underset{j \in[K]-\{Y\}, \eta:\|\eta\| \leq \epsilon}{\operatorname{argmax}} M_{\theta}(X+\eta, Y)_{j} . \tag{7.13}
\end{equation*}
$$

Then if $M_{\theta}\left(X+\eta^{\star}, Y\right)_{j^{\star}}>0, \eta^{\star}$ induces a misclassification and satisfies the constraint in (7.11), meaning that $X+\eta^{\star}$ is an adversarial example. Otherwise, if $M_{\theta}\left(X+\eta^{\star}, Y\right)_{j^{\star}} \leq 0$, then any $\eta:\|\eta\|<\epsilon$ satisfies (7.11), and no adversarial example exists for the pair $(X, Y)$. In summary, if $\eta^{\star}$ is as in eq. (7.13), then $\eta^{\star}$ solves the lower level problem in eq. (7.11).

We present a proof in section $7.7^{2}$. Proposition 7.1 implies that the non-differentiable constraint in (7.11) can be equivalently recast as an ensemble of $K$ differentiable optimization

[^11]problems that can be solved independently. This can collectively be expressed as
\[

$$
\begin{equation*}
\eta^{\star} \in \underset{\eta:\|\eta\|<\varepsilon}{\operatorname{argmax}} \max _{j \in[K]-\{Y\}} M_{\theta}(X+\eta, Y)_{j} . \tag{7.14}
\end{equation*}
$$

\]

Note that this does not constitute a relaxation; (7.11) and (7.14) are equivalent optimization problems. This means that the attacker can maximize the classification error directly using first-order optimization methods without resorting to a relaxation.

The defender's objective. Next, we consider the role of the defender. To handle the discontinuous upper-level problem in (7.10), note that this problem is equivalent to a perturbed version of the supervised learning problem in (7.1). As discussed in $\S 7.2 .1$, the strongest results for problems of this kind have historically been achieved by means of a surrogate-based relaxation. Subsequently, replacing the $0-1$ loss with a differentiable upper bound like the cross-entropy is a principled, guarantee-preserving approach for the defender.

### 7.3.3 Putting the pieces together: Non-zero-sum adversarial training

By combining the disparate problems discussed in the preceeding section, we arrive at a novel non-zero-sum (almost-everywhere) differentiable formulation of adversarial training:

$$
\begin{array}{ll}
\min _{\theta \in \Theta} & \mathbb{E} \ell\left(f_{\theta}\left(X+\eta^{\star}\right), Y\right) \\
\text { subject to } & \eta^{\star} \in \underset{\eta:\|\eta\| \leq \varepsilon}{\operatorname{argmax}} \max _{j \in[K]-\{Y\}} M_{\theta}(X+\eta, y)_{j} \tag{7.16}
\end{array}
$$

Notice that the second level of this bilevel problem remains non-smooth due to the maximization over the classes $j \in[K]-\{Y\}$. To impart smoothness on the problem without relaxing the constraint, observe that we can equivalently solve $K-1$ distinct smooth problems in the second level for each sample ( $X, Y$ ), resulting in the following equivalent optimization problem:

$$
\begin{array}{ll}
\min _{\theta \in \Theta} & \mathbb{E} \ell\left(f_{\theta}\left(X+\eta_{j^{\star}}^{\star}\right), Y\right) \\
\text { subject to } & \eta_{j}^{\star} \in \underset{\eta:\|\eta\| \leq \epsilon}{\operatorname{argmax}} M_{\theta}(X+\eta, y)_{j} \quad \forall j \in[K]-\{Y\} \\
& j^{\star} \in \underset{j \in[K]-\{Y\}}{\operatorname{argmax}} M_{\theta}\left(x+\eta_{j}^{\star}, y\right)_{j} \tag{7.19}
\end{array}
$$

Hence, in (7.19), we first obtain one perturbation $\eta_{j}^{\star}$ per class which maximizes the negative $\operatorname{margin} M_{\theta}\left(X+\eta_{j}^{\star}, Y\right)$ for that particular class. Next, in (7.18), we select the class index $j^{\star}$ corresponding to the perturbation $\eta_{j}^{\star}$ that maximized the negative margin. And finally, in the upper level, the surrogate minimization over $\theta \in \Theta$ is on the perturbed data pair $\left(X+\eta_{j^{\star}}^{\star}, Y\right)$. The result is a non-zero-sum formulation for AT that is amenable to gradient-based optimization, and preserves the optimality guarantees engendered by surrogate loss minimization

```
Algorithm 7.1 Best Targeted Attack (BETA)
Input: Data-label pair \((x, y)\), perturbation size \(\epsilon\), model \(f_{\theta}\), number of classes \(K\), iterations \(T\).
Output: Adversarial perturbation \(\eta^{\star}\)
    for \(j \in 1, \ldots, K\) do
        \(\eta_{j} \leftarrow \operatorname{Unif}[\max (X-\epsilon, 0), \min (X+\epsilon, 1)] \quad \triangleright\) (assume images in \([0,1]^{d}\) )
    end for
    for \(t=1, \ldots, T\) do
        for \(j \in 1, \ldots, K\) do
            \(\eta_{j} \leftarrow \operatorname{OPTIM}\left(\eta_{j}, \nabla_{\eta_{i}} M_{\theta}\left(x+\eta_{j}, y\right)_{j}\right) \quad \triangleright(\) optimizer step, e.g., RMSprop)
            \(\eta_{j} \leftarrow \Pi_{B_{\epsilon}(X) \cap[0,1]^{d}}\left(\eta_{j}\right) \quad \triangleright\) (Projection onto valid perturbation set)
        end for
    end for
    \(j^{\star} \leftarrow \underset{j \in[K]-\{y\}}{\operatorname{argmax}} M_{\theta}\left(x+\eta_{j}, y\right)\)
    return \(\eta_{j^{\star}}\)
```

```
Algorithm 7.2 BETA Adversarial Training (BETA-AT)
Input: Dataset \((X, Y)=\left(x_{i}, y_{i}\right)_{i=1}^{n}\), perturbation size \(\epsilon\), model \(f_{\theta}\), number of classes \(K\), itera-
tions \(T\), attack iterations \(T^{\prime}\).
Output: Robust model \(f_{\theta^{\star}}\)
    for \(t \in 1, \ldots, T\) do
        Sample \(i \sim \operatorname{Unif}[n]\)
        \(\eta^{\star} \leftarrow \operatorname{BETA}\left(x_{i}, y_{i}, \epsilon, f_{\theta}, T^{\prime}\right)\)
        \(L(\theta) \leftarrow \ell\left(f_{\theta}\left(x_{i}+\eta^{\star}\right), y_{i}\right)\)
        \(\theta \leftarrow \operatorname{OPTIM}(\theta, \nabla L(\theta)) \quad \triangleright\) (optimizer step, e.g., SGD)
    end for
    return \(f_{\theta}\)
```

without weakening the adversary.

### 7.4 Algorithms

Given the non-zero-sum formulation of AT in the previous section, the next question is how one should solve this bilevel optimization problem in practice. Our starting point is the empirical version of this bilevel problem, wherein we assume access to a finite dataset $\left\{\left(x_{i}, y_{i}\right)\right\}_{i=1}^{n}$ of $n$ instance-label pairs sampled i.i.d. from $\mathscr{D}$.

$$
\begin{array}{llr}
\min _{\theta \in \Theta} & \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_{\theta}\left(x_{i}+\eta_{i j^{\star}}^{\star}\right), y_{i}\right) & \\
\text { subject to } & \eta_{i j}^{\star} \in \underset{\eta:\|\eta\| \leq e}{\operatorname{argmax}} M_{\theta}\left(x_{i}+\eta, y_{i}\right)_{j} & \forall i, j \in[n] \times[K]-\{Y\} \\
& j^{\star} \in \underset{j \in[K]-\left\{y_{i}\right\}}{\operatorname{argmax}} M_{\theta}\left(x_{i}+\eta_{i j}^{\star}, y_{i}\right)_{j} & \forall i \in[n] \tag{7.22}
\end{array}
$$

To solve this empirical problem, we adopt a stochastic optimization based approach. That is, we first iteratively sample mini-batches from our dataset uniformly at random, and then obtain adversarial perturbations by solving the lower level problems in (7.21) and (7.22). Note that given the differentiability of the negative margin, the lower level problems can be solved iteratively with generic optimizers, e.g., Adam (Kingma and Ba, 2014) or RMSprop. This procedure is summarized in Algorithm 7.1, which we call the BEst Targeted Attack (BETA), given that it directly maximizes the classification error.

After obtaining such perturbations, we calculate the perturbed loss in (7.20), and then differentiate through this loss with respect to the model parameters. By updating the model parameters $\theta$ in the negative direction of this gradient, our algorithm seeks classifiers that are robust against perturbations found by BETA. We call the full adversarial training procedure based on this attack BETA Adversarial Training (BETA-AT), as it invokes BETA as a subroutine; see Algorithm 7.2 for details.

Smoothing the lower level. One potential limitation of the BETA-AT algorithm introduced in Algorithm 7.2 is its sample efficiency: BETA computes one adversarial perturbation per class, but only one of these perturbations is chosen for the upper level of the bilevel formulation (7.20). In this way, one could argue that there is wasted computational effort in discarding perturbations that achieve high values of the negative margin (7.12). This potential shortcoming is a byproduct of the non-smoothness of the max operator in (7.22). Fortunately, we can alleviate this limitation by using smooth under-approximations of the max (e.g., the softmax function), which is continuously differentiable. We explore this scheme in Appendix 7.8.

### 7.5 Experiments

In this section, we evaluate the performance of BETA and BETA-AT on CIFAR-10 (Krizhevsky, 2009). Throughout, we consider a range of AT algorithms, including PGD (Madry et al., 2018a), FGSM (Goodfellow et al., 2015), TRADES (Zhang et al., 2019b), MART (Wang et al., 2020), as well as a range of adversarial attacks, including APGD and AutoAttack (Croce and Hein, 2020b). We consider the standard perturbation budget of $\epsilon=8 / 255$, and all training and test-time attacks use a step size of $\alpha=2 / 255$. For both TRADES and MART, we set the trade-off parameter $\lambda=5$, which is consistent with the original implementations (Wang et al., 2020; Zhang et al., 2019b).

The bilevel formulation eliminates robust overfitting. Robust overfitting occurs when the robust test accuracy peaks immediately after the first learning rate decay, and then falls significantly in subsequent epochs as the model continues to train (Rice et al., 2020). This is illustrated in Figure 7.1a, in which we plot the learning curves (i.e., the clean and robust accuracies for the training and test sets) for a ResNet-18 (He et al., 2016) model trained using 10-step PGD against a 20-step PGD adversary. Notice that after the first learning rate decay step at epoch 100, the robust test accuracy spikes, before dropping off in subsequent epochs. On the other hand, BETA-AT does not suffer from robust overfitting, as shown in Figure 7.1b. We argue that this strength of our method is a direct result of our bilevel formulation, in which


Figure 7.1: BETA does not suffer from robust overfitting. We plot the learning curves against a PGD ${ }^{20}$ adversary for PGD ${ }^{10}$ and BETA-AT ${ }^{10}$. Observe that although PGD displays robust overfitting after the first learning rate decay step, BETA-AT does not suffer from this pitfall.
we train against a proper surrogate for the adversarial classification error.
BETA-AT outperforms baselines on the last iterate of training. We next compare the performance of ResNet-18 models trained using four different AT algorithms: FGSM, PGD, TRADES, MART, and BETA. PGD, TRADES, and MART used a 10-step adversary at training time. At test time, the models were evaluated against five different adversaries: FGSM, 10-step PGD, 40step PGD, 10-step BETA, and APGD. We report the performance of two different checkpoints for each algorithm: the best performing checkpoint chosen by early stopping on a held-out validation set, and the performance of the last checkpoint from training. Note that while BETA performs comparably to the baseline algorithms with respect to early stopping, it outperforms these algorithms significantly when the test-time adversaries attack the last checkpoint of training. This owes to the fact that BETA does not suffer from robust overfitting, meaning that the last and best checkpoints perform similarly.

BETA matches the robustness estimate of AutoAttack. AutoAttack is a state-of-the-art adversarial attack which is widely used to estimate the robustness of trained models on leaderboards such as RobustBench (Croce et al., 2020a; Croce and Hein, 2020b). In brief, AutoAttack comprises a collection of four disparate attacks: APGD-CE, APGD-T, FAB, and Square Attack. AutoAttack also involves several heuristics, including multiple restarts and variable stopping conditions. In Table 7.2, we compare the performance of the top-performing models on RobustBench against AutoAttack, APGD-T, and BETA with RMSprop. Both APGDT and BETA used thirty steps, whereas we used the default implementation of AutoAttack, which runs for 100 iterations. We also recorded the gap between AutoAttack and BETA. Notice

Table 7.1: Adversarial performance on CIFAR-10. We report the test accuracies of various AT algorithms against different adversarial attacks on the CIFAR-10 dataset.

| Training algorithm | Test accuracy |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Clean |  | FGSM |  | PGD ${ }^{10}$ |  | PGD ${ }^{40}$ |  | BETA ${ }^{10}$ |  | APGD |  |
|  | Best | Last | Best | Last | Best | Last | Best | Last | Best | Last | Best | Last |
| FGSM | 81.96 | 75.43 | 94.26 | 94.22 | 42.64 | 1.49 | 42.66 | 1.62 | 40.30 | 0.04 | 41.56 | 0.00 |
| PGD ${ }^{10}$ | 83.71 | 83.21 | 51.98 | 47.39 | 46.74 | 39.90 | 45.91 | 39.45 | 43.64 | 40.21 | 44.36 | 42.62 |
| TRADES ${ }^{10}$ | 81.64 | 81.42 | 52.40 | 51.31 | 47.85 | 42.31 | 47.76 | 42.92 | 44.31 | 40.97 | 43.34 | 41.33 |
| MART ${ }^{10}$ | 78.80 | 77.20 | 53.84 | 53.73 | 49.08 | 41.12 | 48.41 | 41.55 | 44.81 | 41.22 | 45.00 | 42.90 |
| BETA-AT ${ }^{5}$ | 87.02 | 86.67 | 51.22 | 51.10 | 44.02 | 43.22 | 43.94 | 42.56 | 42.62 | 42.61 | 41.44 | 41.02 |
| BETA-AT ${ }^{10}$ | 85.37 | 85.30 | 51.42 | 51.11 | 45.67 | 45.39 | 45.22 | 45.00 | 44.54 | 44.36 | 44.32 | 44.12 |
| BETA-AT ${ }^{20}$ | 82.11 | 81.72 | 54.01 | 53.99 | 49.96 | 48.67 | 49.20 | 48.70 | 46.91 | 45.90 | 45.27 | 45.25 |

Table 7.2: Estimated $\ell_{\infty}$ robustness (robust test accuracy). BETA+RMSprop (ours) vs APGDtargeted (APGD-T) vs AutoAttack (AA). CIFAR-10. BETA and APGD-T use 30 iterations + single restart. $\epsilon=8 / 255$. AA uses 4 different attacks with 100 iterations and 5 restarts.

| Model | BETA | APGD-T | AA | BETA/AA gap | Architecture |
| :--- | :---: | :---: | :---: | :---: | :---: |
| Wang et al. (2023) | 70.78 | 70.75 | 70.69 | 0.09 | WRN-70-16 |
| Wang et al. (2023) | 67.37 | 67.33 | 67.31 | 0.06 | WRN-28-10 |
| Rebuffi et al. (2021) | 66.75 | 66.71 | 66.58 | 0.17 | WRN-70-16 |
| Gowal et al. (2021) | 66.27 | 66.26 | 66.11 | 0.16 | WRN-70-16 |
| Huang et al. (2022) | 65.88 | 65.88 | 65.79 | 0.09 | WRN-A4 |
| Rebuffi et al. (2021) | 64.73 | 64.71 | 64.64 | 0.09 | WRN-106-16 |
| Rebuffi et al. (2021) | 64.36 | 64.27 | 64.25 | 0.11 | WRN-70-16 |
| Gowal et al. (2021) | 63.58 | 63.45 | 63.44 | 0.14 | WRN-28-10 |
| Pang et al. (2022) | 63.38 | 63.37 | 63.35 | 0.03 | WRN-70-16 |

that the 30-step BETA—a heuristic-free algorithm derived from our bilevel formulation of AT—performs almost identically to AutoAttack, despite the fact that AutoAttack runs for significantly more iterations and uses five restarts, which endows AutoAttack with an unfair computational advantage. That is, excepting for a negligible number of samples, BETA matches the robustness estimate of AutoPGD-targeted and AutoAttack, despite using an off-the-shelf optimizer.

### 7.6 Related work

Robust overfitting. Several recent papers (see, e.g., Rebuffi et al. (2021); Chen et al. (2021b); Yu et al. (2022); Dong et al. (2022); Wang et al. (2020); Lee et al. (2020)) have attempted to explain and resolve robust overfitting (Rice et al., 2020). However, none of these works point to a fundamental limitation of adversarial training as the cause of robust overfitting. Rather, much
of this past work has focused on proposing heuristics for algorithms specifically designed to reduce robust overfitting, rather than to improve adversarial training. In contrast, we posit that the lack of guarantees of the zero-sum surrogate-based AT paradigm (Madry et al., 2018a) is at fault, as this paradigm is not designed to maximize robustness with respect to classification error. And indeed, our empirical evaluations in the previous section confirm that our non-zero-sum formulation eliminates robust overfitting.

Estimating adversarial robustness. There is empirical evidence that attacks based on surrogates (e.g., PGD) overestimate the robustness of trained classifiers (Croce and Hein, 2020b; Croce et al., 2020b; Gowal et al., 2019). Indeed, this evidence served as motivation for the formulation of more sophisticated attacks like AutoAttack Croce and Hein (2020b), which empirically tend to provide more accurate estimates of robustness. In contrast, we provide solid, theoretical evidence that commonly used attacks overestimate robustness due to the misalignment between standard surrogate losses and the adversarial classification error. Moreover, we show that optimizing the BETA objective with a standard optimizer (e.g., RMSprop) achieves the same robustness as AutoAttack without employing ad hoc training procedures such as multiple restarts. convoluted stopping conditions, or adaptive learning rates.

One notable feature of past work is an observation made in (Gowal et al., 2019), which finds that multitargeted attacks tend to more accurately estimate robustness. However, their theoretical analysis only applies to linear functions, whereas our work extends these ideas to the nonlinear setting of DNNs. Moreover, (Gowal et al., 2019) do not explore training using a multitargeted attack, whereas we show that BETA-AT is an effective AT algorithm that mitigates the impact of robust overfitting.

Bilevel formulations of AT. Prior to our work, (Zhang et al., 2022b) proposed a different pseudobilevel $^{3}$ formulation for AT, wherein the main objective was to justify the Fast AT algorithm introduced in (Wong et al., 2020). More specifically, the formulation in (Zhang et al., 2022b) is designed to produce solutions that coincide with the iterates of Fast AT by linearizing the attacker's objective. In contrast, our bilevel formulation appears naturally following principled relaxations of the intractable classification error AT formulation. In this way, the formulation in Zhang et al. (2022b) applies only in the context of Fast AT, whereas our formulation deals more generally with the task of adversarial training.

Perhaps the closest variant related to our work is the Bilevel formulation presented in Mianjy and Arora (2023), where the upper level player minimizes an upper bound of the classification error (as in our case), whereas the adversary maximizes a lower bound thereof. Our work is different in that we show the adversary does not need to optimize lower bound, rather, it can

[^12]directly solve the objective via proposition 7.1, leading to a stronger adversary. Moreover, we focus on demonstrating the practical consequences of our reformulation, like the elimination of the Robust Overfitting phenomenon. Also note that Mianjy and Arora (2023) provide a method that works in binary classification whereas our approach works for the general multi-class setting.

Finally, Robey et al. (2021a) propose a constrained optimization formulation of Adversarial Training. Unfortunately in this case, the objective still defines the adversary as the maximization of a surrogate loss function like the cross-entropy. As such, this formulation still suffers from the pitfalls of the zero-sum formulation of adversarial training from Madry et al. (2018b).

### 7.7 Appendix: Proof of proposition 7.1

Suppose there exists $\hat{\eta}$ satisfying $\|\hat{\eta}\| \leq \epsilon$ such that for some $j \in[K], j \neq Y$ we have $M_{\theta}(X+$ $\hat{\eta}, Y)_{j}>0$, i.e., assume

$$
\begin{equation*}
\max _{j \in[K]-\{Y\}, \eta:\|\eta\| \leq \epsilon} M_{\theta}(X+\eta, Y)_{j}>0 \tag{7.23}
\end{equation*}
$$

for such $\hat{\eta}$ and such $j$ we have $f_{\theta}(X+\hat{\eta})_{j}>f_{\theta}(X+\hat{\eta})_{Y}$ and thus $\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}(X+\hat{\eta})_{j} \neq Y$. Hence, such $\hat{\eta}$ induces a misclassification error i.e.,

$$
\begin{equation*}
\hat{\eta} \in \underset{\eta:\|\eta\|_{2} \leq \epsilon}{\operatorname{argmax}}\left\{\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{j} \neq Y\right\} \tag{7.24}
\end{equation*}
$$

In particular if

$$
\begin{equation*}
\left(j^{\star}, \eta^{\star}\right) \in \underset{j \in[K]-\{Y\}, \eta:\|\eta\| \leq \epsilon}{\operatorname{argmax}} M_{\theta}(X+\eta, Y)_{j} \Rightarrow \eta^{\star} \in \underset{\eta:\|\eta\|_{2} \leq \epsilon}{\operatorname{argmax}}\left\{\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{j} \neq Y\right\} \tag{7.25}
\end{equation*}
$$

Otherwise, assume

$$
\begin{equation*}
\max _{j \in[K]-\{Y\}, \eta:\|\eta\| \leq \varepsilon} M_{\theta}(X+\eta, Y)_{j}<0 \tag{7.26}
\end{equation*}
$$

then for all $\eta:\|\eta\|<\epsilon$ and all $j \neq Y$ we have $f_{\theta}(X+\eta)_{j}<f_{\theta}(X+\eta)_{Y}$, so that $\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}(x+\eta)_{j}=$ $Y$ i.e., there is no adversarial example in the ball. In this case for any $\eta$, in particular In particular if

$$
\begin{equation*}
\left(j^{\star}, \eta^{\star}\right) \in \underset{j \in[K]-\{Y\}, \eta:\|\eta\| \leq e}{\operatorname{argmax}} M_{\theta}(X+\eta, Y)_{j} \tag{7.27}
\end{equation*}
$$

Then

$$
\begin{equation*}
0=\left\{\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}\left(X+\eta^{\star}\right)_{j} \neq Y\right\}=\max _{\eta:\|\eta\|_{2} \leq \epsilon}\left\{\underset{j \in[K]}{\operatorname{argmax}} f_{\theta}(X+\eta)_{j} \neq Y\right\} \tag{7.28}
\end{equation*}
$$

In conclusion, the solution

$$
\begin{equation*}
\left(j^{\star}, \eta^{\star}\right) \in \underset{j \in[K]-\{Y\}, \eta:\|\eta\| \leq \varepsilon}{\operatorname{argmax}} M_{\theta}(X+\eta, Y)_{j} \tag{7.29}
\end{equation*}
$$

always yields a maximizer of the misclassification error.

### 7.8 Appendix: Smooth reformulation of the lower level

First, note that the problem in eqs. (7.20) to (7.22) is equivalent to

$$
\begin{align*}
& \min _{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{K} \lambda_{i j}^{\star} \ell\left(f_{\theta}\left(x_{i}+\eta_{i j}^{\star}\right), y_{i}\right) \\
& \text { subject to } \lambda_{i j}^{\star}, \eta_{i j}^{\star} \in \underset{\substack{\left\|\eta_{i j}\right\| \leq e \\
\lambda_{i j} \geq 0,\left\|\lambda_{i}\right\|_{1}=1, \lambda_{i y}=0}}{\operatorname{argmax}} \sum_{j=1}^{K} \lambda_{i j} M_{\theta}\left(x_{i}+\eta_{i j}, y_{i}\right)_{j} \quad \forall i \in[n] \tag{7.30}
\end{align*}
$$

This is because the maximum over $\lambda_{i}$ in eq. (7.30) is always attained at the coordinate vector $\mathbf{e}_{j}$ such that $M_{\theta}\left(x_{i}+\eta_{i j}^{\star}, y_{i}\right)$ is maximum.

An alternative is to smooth the lower level optimization problem by adding an entropy regularization:

$$
\begin{align*}
\max _{\eta:\|\eta\| \leq e} \max _{j \in[K]-\{y\}} M_{\theta}\left(x+\eta_{j}, y\right)_{j} & =\max _{\eta:\|\eta\| \leq e} \max _{\lambda \geq 0,\|\lambda\|_{1}=1, \lambda_{y}=0}\left\langle\lambda, M_{\theta}\left(x+\eta_{j}, y\right)_{j=1}^{K}\right\rangle \\
& \geq \max _{\eta:\|\eta\| \leq e} \max _{\lambda \geq 0,\|\lambda\|_{1}=1, \lambda_{y}=0}\left\langle\lambda, M_{\theta}\left(x+\eta_{j}, y\right)_{j=1}^{K}\right\rangle-\frac{1}{\mu} \sum_{j=1}^{K} \lambda_{j} \log \left(\lambda_{j}\right) \\
& =\max _{\eta:\|\eta\| \leq e} \frac{1}{\mu} \log \left(\sum_{\substack{j=1 \\
j \neq y}}^{K} e^{\mu M_{\theta}(X+\eta, y)_{j}}\right) \tag{7.31}
\end{align*}
$$

where $\mu>0$ is some temperature constant. The inequality here is due to the fact that the entropy of a discrete probability $\lambda$ is positive. The innermost maximization problem in (7.31) has the closed-form solution:

$$
\begin{equation*}
\lambda_{j}^{\star}=\frac{e^{\mu M_{\theta}\left(x+\eta_{j}, y\right)_{j}}}{\sum_{\substack{j=1 \\ j \neq y}}^{K} e^{\mu M_{\theta}\left(x+\eta_{j}, y\right)_{j}}}: j \neq y, \quad \lambda_{y}^{\star}=0 \tag{7.32}
\end{equation*}
$$

Hence, after relaxing the second level maximization problem following eq. (7.31), and plugging
in the optimal values for $\lambda$ we arrive at:

$$
\begin{align*}
& \min _{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\
j \neq y_{i}}}^{K} \frac{e^{\mu M_{\theta}\left(x_{i}+\eta_{i j}, y_{i}\right)_{j}}}{\sum_{\substack{j=1 \\
j \neq y_{i}}}^{K} e^{\mu M_{\theta}\left(x_{i}+\eta_{i j}, y_{i}\right)_{j}}} \ell\left(f_{\theta}\left(x_{i}+\eta_{i j}^{\star}\right), y_{i}\right)  \tag{7.33}\\
& \text { subject to } \eta_{i j}^{\star} \in \underset{\left\|\eta_{i j}\right\| \leq \varepsilon}{\operatorname{argmax} M_{\theta}\left(x_{i}+\eta_{i j}, y_{i}\right)_{j} \quad \forall i \in[n], j \in[K]}
\end{align*}
$$

$$
\begin{array}{ll}
\min _{\theta \in \Theta} & \frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{j=1 \\
j \neq y_{i}}}^{K} \frac{e^{\mu M_{\theta}\left(x_{i}+\eta_{i j}^{\star}, y_{i}\right)_{j}}}{\sum_{\substack{j=1 \\
j \neq y_{i}}}^{K} e^{\mu M_{\theta}\left(x_{i}+\eta_{i j}^{\star}, y_{i}\right)_{j}}} \ell\left(f_{\theta}\left(x_{i}+\eta_{i j}^{\star}\right), y_{i}\right) \\
\text { subject to } & \eta_{i j}^{\star} \in \underset{\eta:\|\eta\| \leq \varepsilon}{\operatorname{argmax}} M_{\theta}\left(x_{i}+\eta, y_{i}\right)_{j} \quad \forall i \in[n] \tag{7.35}
\end{array}
$$

In this formulation, both upper- and lower-level problems are smooth (barring the possible use of nonsmooth components like ReLU). Most importantly (I) the smoothing is obtained through a lower bound of the original objective in eqs. (7.21) and (7.22), retaining guarantees that the adversary will increase the misclassification error and (II) all the adversarial perturbations obtained for each class now appear in the upper level (7.34), weighted by their corresponding negative margin. In this way, we make efficient use of all perturbations generated: if two perturbations from different classes achieve the same negative margin, they will affect the upper-level objective in fair proportion. This formulation gives rise to algorithm 7.3.

```
Algorithm 7.3 Smooth BETA Adversarial Training (SBETA-AT)
Input: Dataset \((X, Y)=\left(x_{i}, y_{i}\right)_{i=1}^{n}\), perturbation size \(\epsilon\), model \(f_{\theta}\), number of classes \(K\), itera-
tions \(T\), attack iterations \(T^{\prime}\), temperature \(\mu>0\)
Output: Robust model \(f_{\theta^{\star}}\)
    for \(t \in 1, \ldots, T\) do
        Sample \(i \sim \operatorname{Unif}[n]\)
        Initialize \(\eta_{j} \sim \operatorname{Unif}\left[\max \left(0, x_{i}-\epsilon\right), \min \left(x_{i}+\epsilon, 1\right)\right], \forall j \in[K]\)
        for \(j \in 1, \ldots, K\) do
            for \(t \in 1, \ldots, T^{\prime}\) do
                \(\eta_{j} \leftarrow \operatorname{OPTIM}\left(\eta_{j}, \nabla_{\eta} M_{\theta}\left(x_{i}+\eta_{j}, y_{i}\right)_{j}\right) \quad \triangleright(\) attack optimizer step, e.g., RMSprop)
                \(\eta_{j} \leftarrow \Pi_{B_{\epsilon}\left(x_{i}\right) \cap[0,1]^{d}}\left(\eta_{j}\right) \quad \triangleright\) (Projection onto valid perturbation set)
            end for
        end for
        Compute \(L(\theta)=\sum_{j=1, j \neq y_{i}}^{K} \frac{e^{\mu M_{\theta}\left(x_{i}+\eta_{j}, y_{i}\right)_{j}}}{\sum_{j=1, j \neq y_{i}}^{K} e^{\mu M_{\theta}\left(x_{i}+\eta_{j}, y_{i}\right)_{j}}} \ell\left(f_{\theta}\left(x_{i}+\eta_{j}\right), y_{i}\right) \quad \theta \leftarrow \operatorname{OPTIM}(\theta, \nabla L(\theta)) \triangleright\)
    (model optimizer step)
    end for
    return \(f_{\theta}\)
```


### 7.9 Bibliographic Note

The main ideas of this work were developed together with Alexander Robey. The candidate derived (Proposition 7.1) and performed the experiments summarized in table 7.2.

# 8 Fast and Provable ADMM for Learning with Generative Priors 

Fabian Latorre, Armin Eftekhari and Volkan Cevher. Advances in Neural Information Processing Systems (NeurIPS) 2019.


#### Abstract

In this work, we propose a (linearized) Alternating Direction Method-of-Multipliers (ADMM) algorithm for minimizing a convex function subject to a nonconvex constraint. We focus on the special case where such constraint arises from the specification that a variable should lie in the range of a neural network. This is motivated by recent successful applications of Generative Adversarial Networks (GANs) in tasks like compressive sensing, denoising and robustness against adversarial examples. The derived rates for our algorithm are characterized in terms of certain geometric properties of the generator network, which we show hold for feedforward architectures, under mild assumptions. Unlike gradient descent (GD), it can efficiently handle non-smooth objectives as well as exploit efficient partial minimization procedures, thus being faster in many practical scenarios.


### 8.1 Introduction

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) show great promise for faithfully modeling complex data distributions, such as natural images (Radford et al., 2015; Brock et al., 2019) or audio signals (Engel et al., 2019; Donahue et al., 2019). Understanding and improving the theoretical and practical aspects of their training has thus attracted significant interest (Lucic et al., 2018; Mescheder et al., 2018; Daskalakis et al., 2018; Hsieh et al., 2018; Gidel et al., 2019).

Researchers have also begun to leverage the modeling power of GANs and other generative models like Variational Auto-encoders (Kingma and Welling, 2013) in applications ranging from compressive sensing (Bora et al., 2017), to image denoising (Lipton and Tripathi, 2017; Tripathi et al., 2018), to robustness against adversarial examples (Ilyas et al., 2017; Samangouei et al., 2018).

These and other (Dhar et al., 2018; Ulyanov et al., 2018) applications model high-dimensional
data as the output of the generator network associated with a generative model, and often lead to a highly non-convex optimization problem of the form $\min _{z} f(G(z))$, where the the generator $G$ is nonlinear and $f$ is convex. We then find the optimal latent vector $z$, as illustrated in section 8.5 with several examples.

This optimization problem involving a generative model poses various difficulties for existing first-order algorithms. Indeed, to our knowledge, the only existing provable algorithm for solving (8.1) relies on the existence of a projection oracle, and is limited to the special case of compressive sensing with a generative prior (Shah and Hegde, 2018; Hegde, 2018), see section 8.4 for the details. The main computational bottleneck is of course the non-convex projection step, for which no convergence analysis in terms of the geometry of the underlying generator $G$ currently exists.

On the other hand, Gradient Descent (GD) and its adaptive variants (Kingma and Ba, 2014) cannot efficiently handle non-smooth objective functions, as they are entirely oblivious to the composite structure of the problem (Nesterov, 2013a). A simple example is denoising with the $\ell_{\infty}$-norm, for which subgradient descent (as the standard non-smooth alternative to GD) fails in practice, as observed in Section section 8.5.

With the explosion of generative models in popularity, there is consequently a pressing need for provable and flexible optimization algorithms to solve the resulting non-convex and (possibly) non-smooth problems. The present work addresses this need by focusing on the general optimization template

$$
\begin{array}{ll}
\underset{w, z}{\operatorname{minimize}} & F(w, z):=L(w)+R(w)+H(z)  \tag{8.1}\\
\text { subject to } & w=G(z)
\end{array}
$$

where $L: \mathbb{R}^{d} \rightarrow \mathbb{R}$ is convex and smooth, $R: \mathbb{R}^{d} \rightarrow \mathbb{R}$ and $H: \mathbb{R}^{s} \rightarrow \mathbb{R}$ are convex but not necessarily smooth, and $G: \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ is differentiable but often non-linear, corresponding to the generator network associated with a generative model. Even though $R$ and $H$ might not be smooth, we assume throughout that their proximal mappings can be efficiently computed (Parikh et al., 2014).

For brevity, we refer to (8.1) as optimization with a generative prior whenever $G$ is given by the generator network associated with a generative model (Kingma and Welling, 2013; Goodfellow et al., 2014). In this context, we make three key contributions, summarized below:

1. Algorithm: We propose an efficient and scalable (linearized) Alternating Direction Method-of-Multipliers (ADMM) framework to solve (8.1), see Algorithm 8.1. To our knowledge, this is the first non-convex and linearized ADMM algorithm for nonlinear constraints with provable fast rates to solve problem (8.1), see section 8.4 for a detailed literature review.

We evaluate this algorithm numerically in the context of denoising with GANs in the presence of adversarial or stochastic noise, as well as compressive sensing (Bora et al., 2017). In
particular, Algorithm 8.1 allows for efficient denoising with the $\ell_{\infty^{-}}$and $\ell_{1}$-norms, with applications in defenses against adversarial examples (Szegedy et al., 2014a) and signal processing, respectively.
2. Optimization guarantees: We prove fast approximate convergence for Algorithm 8.1 under the assumptions of smoothness and near-isometry of $G$, as well as strong convexity of $L$. That is, we distill the key geometric attributes of the generative network $G$ responsible for the success of Algorithm 8.1. We then show how some common neural network architectures satisfy these geometric assumptions.

We also establish a close relation between a variant of Algorithm 8.1 and the gradient descent in (Bora et al., 2017) and, in this sense, provide the first rates for it, albeit in a limit case detailed in section 8.3. Indeed, one key advantage of the primal-dual formulation studied in this paper is exactly this versatility, as well as the efficient handling of non-smooth objectives.

Lastly, we later relax the assumptions on $L$ to restricted strong convexity/smoothness, thus extending our results to the broader context of statistical learning with generative priors, which includes compressive sensing (Bora et al., 2017) as a special case.
3. Statistical guarantees: In the context of statistical learning with generative priors, where $L$ in (8.1) is replaced with an empirical risk, we provide the generalization error associated with Algorithm 8.1. That is, we use the standard notion of Rademacher complexity (Mohri et al., 2018c) to quantify the number of training data points required for Algorithm 8.1 to learn the true underlying parameter $w^{\natural}$.

### 8.2 Algorithm

In this section, we adapt the powerful Alternating Descent Method of Multipliers (ADMM) (Glowinski and Marroco, 1975; Gabay and Mercier, 1976; Boyd et al., 2011) to solve the non-convex problem (8.1). We define the corresponding augmented Lagrangian with the dual variable $\lambda \in \mathbb{R}^{p}$ as

$$
\begin{equation*}
\mathscr{L}_{\rho}(w, z, \lambda):=L(w)+\langle w-G(z), \lambda\rangle+\frac{\rho}{2}\|w-G(z)\|_{2}^{2} \tag{8.2}
\end{equation*}
$$

for a penalty weight $\rho>0$. By a standard duality argument, (8.1) is equivalent to

$$
\begin{equation*}
\min _{w, z} \max _{\lambda} \mathscr{L}_{\rho}(w, z, \lambda)+R(w)+H(z) \tag{8.3}
\end{equation*}
$$

Applied to (8.3), every iteration of ADMM would minimize the augmented Lagrangian with respect to $z$, then with respect to $w$, and then update the dual variable $\lambda$. Note that $\mathscr{L}_{\rho}(w, z, \lambda)$ is often non-convex with respect to $z$ due to the nonlinearity of the generator $G: \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ and, consequently, the minimization step with respect to $z$ in ADMM is often intractable.

To overcome this limitation, we next linearize ADMM. In the following, we let $\mathrm{P}_{R}$ and $\mathrm{P}_{H}$
denote the proximal maps of $R$ and $H$, respectively (Parikh et al., 2014).
The equivalence of problems (8.1) and (8.3) motivates us to consider the following algorithm for the penalty weight $\rho>0$, the primal step sizes $\alpha, \beta>0$, and the positive dual step sizes $\left\{\sigma_{t}\right\}_{t \geq 0}$ :

$$
\begin{align*}
z_{t+1} & =\mathrm{P}_{\beta H}\left(z_{t}-\beta \nabla_{z} \mathscr{L}_{\rho}\left(w_{t}, z_{t}, \lambda_{t}\right)\right) \\
w_{t+1} & =\mathrm{P}_{\alpha R}\left(w_{t}-\alpha \nabla_{w} \mathscr{L}_{\rho}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right)  \tag{8.4}\\
\lambda_{t+1} & =\lambda_{t}+\sigma_{t+1}\left(w_{t+1}-G\left(z_{t+1}\right)\right)
\end{align*}
$$

As opposed to ADMM, to solve (8.1), the linearized ADMM in (8.4) takes only one descent step in both $z$ and $w$, see Algorithm 8.1 for the summary. The particular choice of the dual step sizes $\left\{\sigma_{t}\right\}_{t}$ in Algorithm 8.1 ensures that the dual variables $\left\{\lambda_{t}\right\}_{t}$ remain bounded, see (Bertsekas, 1976) for a precedent in the convex literature.

EADMM. Let us introduce an important variant of Algorithm 8.1. In our setting, $\mathscr{L}_{\rho}(w, z, \lambda)$ is in fact convex with respect to $w$ and therefore EADMM replaces the second step in (8.4) with exact minimization over $w$. This exact minimization step can be executed with an off-the-shelf convex solver, or might sometimes have a closed-form solution. Moreover, EADMM gradually increases the penalty weight to emulate a multi-scale structure. More specifically, for an integer $K$, consider the sequences of penalty weights and primal step sizes $\left\{\rho_{k}, \alpha_{k}, \beta_{k}\right\}{ }_{k=1}^{K}$, specified as

$$
\begin{equation*}
\rho_{k}=2^{k} \rho, \quad \alpha_{k}=2^{-k} \alpha, \quad \beta_{k}=2^{-k} \beta, \quad k \leq K . \tag{8.5}
\end{equation*}
$$

Consider also a sequence of integers $\left\{n_{k}\right\}_{k=1}^{K}$, where

$$
\begin{equation*}
n_{k}=2^{k} n, \quad k \leq K \tag{8.6}
\end{equation*}
$$

for an integer $n$. At (outer) iteration $k$, EADMM executes $n_{k}$ iterations of Algorithm 8.1 with exact minimization over $w$. Then it passes the current iterates of $w, z$, and dual step size to the next (outer) iteration. Loosely speaking, EADMM has a multi-scale structure, allowing it to take larger steps initially and then slowing down as it approaches the solution. As discussed in Section 8.3, the theoretical guarantees for Algorithm 8.1 also apply to EADMM. The pseudocode for EADMM is given in section 8.14.

As the closing remark, akin to the convex case (He et al., 2000; Xu et al., 2017), it is also possible to devise a variant of Algorithm 8.1 with adaptive primal step sizes, which we leave for a future work.

### 8.3 Optimization Guarantees

Let us study the theoretical guarantees of Algorithm 8.1 for solving program (8.1), whose constraints are nonlinear and non-convex (since $G$ is specified by a neural network). The main

```
Algorithm 8.1 Linearized ADMM for solving problem (8.1)
Input: Differentiable \(L\), proximal-friendly convex regularizers \(R\) and \(H\), differentiable prior \(G\),
penalty weight \(\rho>0\), primal step sizes \(\alpha, \beta>0\), initial dual step size \(\sigma_{0}>0\), primal initialization
\(w_{0}\) and \(z_{0}\), dual initialization \(\lambda_{0}\), stopping threshold \(\tau_{c}>0\).
for \(t=0,1, \ldots, T-1\) do
    \(z_{t+1} \leftarrow \mathrm{P}_{\beta H}\left(z_{t}-\beta \nabla_{z} \mathscr{L}_{\rho}\left(w_{t}, z_{t}, \lambda_{t}\right)\right) \quad \triangleright\) primal updates
    \(w_{t+1} \leftarrow \mathrm{P}_{\alpha R}\left(w_{t}-\alpha \nabla_{w} \mathscr{L}_{\rho}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right)\)
    \(\sigma_{t+1} \leftarrow \min \left(\sigma_{0}, \frac{\sigma_{0}}{\left\|w_{t+1}-G\left(z_{t+1}\right)\right\|_{2} t \log ^{2}(t+1)}\right) \quad \triangleright\) dual step size
    \(\lambda_{t+1} \leftarrow \lambda_{t}+\sigma_{t+1}\left(w_{t+1}-G\left(z_{t+1}\right)\right) \quad \triangleright\) dual update
    \(s \leftarrow \frac{\left\|z_{t+1}-z_{t}\right\|_{2}^{2}}{\alpha}+\frac{\left\|w_{t+1}-w_{t}\right\|_{2}^{2}}{\beta}+\sigma_{t}\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2} \leq \tau_{c} \quad \triangleright\) stopping criterion
    if \(s \leq \tau_{c}\) then
        return \(\left(w_{t+1}, z_{t+1}\right)\)
    end if
    end for
    return \(\left(w_{T}, z_{T}\right)\)
```

contribution of this section is Theorem 8.1, which is inherently an optimization result stating that Algorithm 8.1 succeeds under certain assumptions on (8.1).

From an optimization perspective, to our knowledge, Theorem 8.1 is the first to provide (fast) rates for non-convex and linearized ADMM, see section 8.4 for a detailed literature review. The assumptions imposed below on $L$ and the generator $G$ ensure the success of Algorithm 8.1 and are shortly justified for our setup, where $G$ is a generator network.

Assumption 8.1. strong convexity / smoothness of $L$ : We assume that $L$ in (8.1) is both strongly convex and smooth, namely, there exist $0<\mu_{L} \leq v_{L}$ such that

$$
\begin{equation*}
\frac{\mu_{L}}{2}\left\|w-w^{\prime}\right\|^{2} \leq L\left(w^{\prime}\right)-L(w)-\left\langle w^{\prime}-w, \nabla L(w)\right\rangle \leq \frac{v_{L}}{2}\left\|w-w^{\prime}\right\|^{2}, \quad \forall w, w^{\prime} \in \mathbb{R}^{d} . \tag{8.7}
\end{equation*}
$$

Assumption 8.1 is necessary to establish fast rates for Algorithm 8.1, and is readily met for $L(w)=\|w-\widehat{w}\|_{2}^{2}$ with $\mu_{L}=v_{L}=1$, which renders Algorithm 8.1 applicable to $\ell_{2}$-denoising with generative prior in (Tripathi et al., 2018; Samangouei et al., 2018; Ilyas et al., 2017). Here, $\widehat{w}$ is the noisy image.

In section 8.6, we also relax the strong convexity/smoothness in Assumption 8.1 to restricted strong convexity/smoothness, which enables us to apply Theorem 8.1 in the context of statistical learning with a generative prior, for example in compressive sensing (Bora et al., 2017).

Under Assumption 8.1, even though $L$ and consequently the objective function of (8.1) are strongly convex, problem (8.1) might not have a unique solution, which is in stark contrast with convex optimization. Indeed, a simple example is minimizing $x^{2}+y^{2}$ with the constraint $x^{2}+y^{2}=1$. We next state our assumptions on the generator $G$.

Assumption 8.2. Strong smoothness of $G$ : Let $D G$ be the Jacobian of $G$. We assume that $G: \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ is strongly smooth, namely, there exists $v_{G} \geq 0$ such that

$$
\begin{equation*}
\left\|G\left(z^{\prime}\right)-G(z)-D G(z) \cdot\left(z^{\prime}-z\right)\right\|_{2} \leq \frac{v_{G}}{2}\left\|z^{\prime}-z\right\|_{2}^{2}, \quad \forall z, z^{\prime} \in \mathbb{R}^{s} \tag{8.8}
\end{equation*}
$$

Assumption 8.3. Near-isometry of $G$ : We assume that the generative prior $G$ is a near-isometric map, namely, there exist $0<\iota_{G} \leq \kappa_{G}$ such that

$$
\begin{equation*}
\iota_{G}\left\|z^{\prime}-z\right\|_{2} \leq\left\|G\left(z^{\prime}\right)-G(z)\right\|_{2} \leq \kappa_{G}\left\|z^{\prime}-z\right\|_{2}, \quad \forall z, z^{\prime} \in \mathbb{R}^{s} \tag{8.9}
\end{equation*}
$$

The invertibility of certain network architectures have been established before in (Ma et al., 2018; Hand and Voroninski, 2017). More concretely, Assumptions 8.2 and 8.3 hold for a broad class of generators, as summarized in Proposition 8.1 and proved in section 8.7.

Proposition 8.1. Let $G_{\Xi}: \mathscr{X} \subset \mathbb{R}^{d} \rightarrow \mathbb{R}^{s}$ be a feedforward neural network with weights $\Xi \in \mathbb{R}^{h}$, $k$ layers, non-decreasing layer sizes $s \leq s_{1} \leq \ldots s_{k} \leq d$, with $\omega_{i}$ as activation function in the $i$-th layer, and compact domain $\mathscr{X}$. For every layer $i$, suppose that the activation $\omega_{i}: \mathbb{R} \rightarrow \mathbb{R}$ is of class $C^{1}$ (continuously-differentiable) and strictly increasing. Then, after an arbitrarily small perturbation to the weights $\Xi$, Assumptions 8.2 and 8.3 hold almost surely with respect to the Lebesgue measure.

A few comments about the preceding result are in order.
Choice of the activation function: Strictly-increasing $C^{1}$ activation functions in Proposition 8.1, such as the Exponential Linear Unit (ELU) (Clevert et al., 2015) or softplus (Dugas et al., 2001), achieve similar or better performance compared to the commonly-used (but nonsmooth) Rectified Linear Activation Unit (ReLU) (Xu et al., 2015; Clevert et al., 2015; Gulrajani et al., 2017; Kumar et al., 2017; Kim et al., 2018).

In our experiments in Section 2.7, we found that using ELU activations for the generator $G$ does not adversely affect the representation power of the trained generator. Lastly, the activation function for the final layer of the generator is typically chosen as the sigmoid or tanh (Radford et al., 2015), for which the conditions in Proposition 8.1 are also met.

Compact domain: The compactness requirement in Proposition 8.1 is mild. Indeed, even though the Gaussian distribution is the default choice as the input for the generator in GANs, training has also been successful using compactly-supported distributions, such as the uniform distribution (Lipton and Tripathi, 2017).

Interestingly, even after training with Gaussian noise, limiting the resulting generator to a truncated Gaussian distribution can in fact boost the performance of GANs (Brock et al., 2019), as measured with common metrics like the Inception Score (Salimans et al., 2016) or Frechet Inception Distance (Heusel et al., 2017). This evidence suggests that obtaining a good generator $G$ with compact domain is straightforward. In the experiments of Section 2.7, we
use truncated Gaussian on an Euclidean ball centered at the origin.
Non-decreasing layer sizes: This is a standard feature of popular generator architectures such as the DCGAN (Radford et al., 2015) or infoGAN (Chen et al., 2016). This property is also exploited in the analysis of the optimization landscape of problem (8.1) by Hand and Voroninski (2017); Heckel et al. (2019) and for showing invertiblity of (de)convolutional generators (Ma et al., 2018).

Necessity of assumptions on $G$ : Assumptions 8.2 and 8.3 on the generator $G$ are necessary for the provable success of Algorithm 8.1. Loosely speaking, Assumption 8.2 controls the curvature of the generative prior, without which the dual iterations can oscillate without improving the objective.

On the other hand, the lower bound in (8.9) means that the generative prior $G$ must be stably injective: Faraway latent parameters should be mapped to faraway outputs under G. As a pathological example, consider the parametrization of a circle as $\{(\sin z, \cos z): z \in[0,2 \pi)\}$.

This stable injectivity property in (8.9) is necessary for the success of Algorithm 8.1 and is not an artifact of our proof techniques. Indeed, without this condition, the $z$ updates in Algorithm 8.1 might not reduce the feasibility gap $\|w-G(z)\|_{2}$. Geometric assumptions on nonlinear constraints have precedent in the optimization literature (Birgin et al., 2016; Flores-Bazán et al., 2012; Cartis et al., 2018) and to a lesser extent in the literature of neural networks too (Hand and Voroninski, 2017; Ma et al., 2018), which we further discuss in Section 8.4.

Having stated and justified our assumptions on $L$ and the generator $G$ in (8.1), we are now prepared to present the main technical result of this section. Theorem 8.1 states that Algorithm 8.1 converges linearly to a small neighborhood of a solution, see section 8.8 for the proof.

Theorem 8.1. (guarantees for Algorithm 8.1) Suppose that Assumptions 8.1-8.3 hold. Let ( $w^{*}, z^{*}$ ) be a solution of program (8.1) and let $\lambda^{*}$ be a corresponding optimal dual variable. Let also $\left\{w_{t}, z_{t}, \lambda_{t}\right\}_{t \geq 0}$ denote the output sequence of Algorithm 8.1. Suppose that the primal step sizes $\alpha, \beta$ satisfy

$$
\begin{equation*}
\alpha \leq \frac{1}{v_{\rho}}, \quad \beta \leq \frac{1}{\xi_{\rho}+2 \alpha \tau_{\rho}^{2}} . \quad \sigma_{0} \leq \sigma_{0, \rho} . \tag{8.10}
\end{equation*}
$$

Then it holds that

$$
\begin{gather*}
\frac{\left\|w_{t}-w^{*}\right\|_{2}^{2}}{\alpha}+\frac{\left\|z_{t}-z^{*}\right\|_{2}^{2}}{\beta} \leq 2\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{\bar{\eta}_{\rho}}{\rho},  \tag{8.11}\\
\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2} \leq \frac{4\left(1-\eta_{\rho}\right)^{t} \Delta_{0}}{\rho}+\frac{\tilde{\eta}_{\rho}}{\rho^{2}}, \tag{8.12}
\end{gather*}
$$

for every iteration $t$. Above, $\Delta_{0}=\mathscr{L}_{\rho}\left(w_{0}, z_{0}, \lambda_{0}\right)-\mathscr{L}_{\rho}\left(w^{*}, z^{*}, \lambda^{*}\right)$ is the initialization error, see (8.2). The convergence rate $1-\eta_{\rho} \in(0,1)$ and the quantities $v_{\rho}, \xi_{\rho}, \tau_{\rho}, \sigma_{0, \rho}, \bar{\eta}_{\rho}, \widetilde{\eta}_{\rho}$ above
depend on the parameters in Assumptions 8.1-8.3 and on $\lambda^{*}$, as specified in the proof. As an example, in the regime where $\mu_{L} \gg \rho$ and $\iota_{G}^{2} \gg v_{G}$, we can take

$$
\begin{gather*}
\alpha \approx \frac{1}{v_{L}}, \quad \beta \approx \frac{1}{\rho \kappa_{G}^{2}}, \quad \frac{\rho v_{G}}{\kappa_{G}^{2}} \lesssim \sigma_{0} \lesssim \rho \min \left(\frac{\mu_{L}^{2}}{v_{L}^{2}}, \frac{\iota_{G}^{4}}{\kappa_{G}^{4}}\right), \\
\eta_{\rho} \approx \min \left(\frac{\mu_{L}}{v_{L}}, \frac{\iota_{G}^{2}}{\kappa_{G}^{2}}\right), \quad \bar{\eta}_{\rho} \approx \widetilde{\eta}_{\rho} \approx \max \left(\frac{v_{L}}{\mu_{L}}, \frac{\kappa_{G}^{2}}{\iota_{G}^{2}}\right) . \tag{8.13}
\end{gather*}
$$

Above, for the sake of clarity, $\approx$ and $\lesssim$ suppress the universal constants, dependence on the initial dual $\lambda_{0}$ and the corresponding step size $\sigma_{0}$.

A few clarifying comments about Theorem 8.1 are in order.
Error: According to Theorem 8.1, if the primal and dual step sizes are sufficiently small and Assumptions 8.1-8.3 are met, Algorithm 8.1 converges linearly to a neighborhood of a solution ( $w^{*}, z^{*}$ ). The size of this neighborhood depends on the penalty weight $\rho$ in (8.2). For instance, in the example in Theorem 8.1, it is easy to verify that this neighborhood has a radius of $O(1 / \rho)$, which can be made smaller by increasing $\rho$.

Theorem 8.1 is however silent about the behavior of Algorithm 8.1 within this neighborhood. This is to be expected. Indeed, even in the simpler convex case, where $G$ in program (8.1) would have been an affine map, provably no first-order algorithm could converge linearly to the solution (Ouyang and Xu, 2018; Agarwal et al., 2010).

Investigating the behavior of Algorithm 8.1 within this neighborhood, while interesting, arguably has little practical value. For example, in the convex case, ADMM would converge slowly (sublinearly) in this neighborhood, which does not appeal to the practitioners.

As another example, when Algorithm 8.1 is applied in the context of statistical learning, there is no benefit in solving (8.1) beyond the statistical accuracy of the problem at hand (Agarwal et al., 2010), see the discussion in subsection 8.6.1. As such, we defer the study of the local behavior of Algorithm 8.1 to a future work.

Feasibility gap: Likewise, according to (8.24) in Theorem 8.1, the feasibility gap of Algorithm 8.1 rapidly reaches a plateau. In the example in Theorem 8.1 , the feasibility gap rapidly reaches $O(1 / \rho)$, where $\rho$ is the penalty weight in (8.2). As before, even in the convex case, no first-order algorithm could achieve exact feasibility at linear rate (Ouyang and Xu, 2018; Agarwal et al., 2010).

Intution: While the exact expressions for the quantities in Theorem 8.1 are given in section 8.8, the example provided in Theorem 8.1 highlights the simple but instructive regime where $\mu_{L} \gg \rho$ and $\iota_{G}^{2} \gg v_{G}$, see Assumptions 8.1-8.3. Intuitively, $\mu_{L} \gg \rho$ means that minimizing the objective of (8.1) is prioritized over reducing the feasibility gap, see (8.2). In addition, $l_{G}^{2} \gg v_{G}$
suggests that the generative prior $G$ is very smooth.
In this regime, the primal step size $\alpha$ for $w$ updates is determined by how smooth $L$ is, and the primal step size $\beta$ in the latent variable $z$ is determined by how smooth $G$ is, see (8.13). Similar restrictions are standard in first-order algorithms to avoid oscillations (Nesterov, 2013b).

As discussed earlier, the algorithm rapidly reaches a neighborhood of size $O(1 / \rho)$ of a solution and the feasibility gap plateaus at $O(1 / \rho)$. Note the trade-off here for the choice of $\rho$ : the larger the penalty weight $\rho$ is, the more accurate Algorithm 8.1 would be and yet increasing $\rho$ is restricted by the assumption $\rho \ll \mu_{L}$. Moreover, in this example, the rate $1-\eta_{\rho}$ of Algorithm 8.1 depends only on the regularity of $L$ and $G$ in program (8.1), see (8.13). Indeed, the more wellconditioned $L$ is and the more near-isometric $G$ is, the larger $\eta_{\rho}$ and the faster the convergence would be.

Generally speaking, increasing the penalty weight $\rho$ reduces the bias of Algorithm 8.1 at the cost of a slower rate. Beyond our work, such dependence on the geometry of the constraints has precedent in the literature of optimization (Birgin et al., 2016; Flores-Bazán et al., 2012; Cartis et al., 2018) and manifold embedding theory (Eftekhari and Wakin, 2015, 2017).

Relation to simple gradient descent: Consider a variant of Algorithm 8.1 that replaces the linearized update for $w$ in (8.4) with exact minimization with respect to $w$, which can be achieved with an off-the-shelf convex solver or might have a closed-form solution in some cases. The exact minimization over $w$ and Lemma 8.4 together guarantee that Theorem 8.1 also applies to this variant of Algorithm 1.

Moreover, as a special case of (8.1) where $R \equiv 0$ and $H \equiv 0$, this variant is closely related to GD (Bora et al., 2017), presented there without any rates. In Appendix 8.11, we establish that the updates of both algorithms match as the feasibility gap vanishes.

In this sense, Theorem 8.1 provides the first rates for GD, albeit in the limit case of vanishing feasibility gap. Indeed, one key advantage of the primal-dual formulation studied in this paper is exactly this versatility in providing a family of algorithms, such as Algorithms 8.1 and 2, that can be tuned for various scenarios and can also efficiently handle the non-smooth case where $R$ or $H$ are nonzero in (8.1).

### 8.4 Related Work

Bora et al. (2017) empirically tune gradient descent for compressive sensing with a generative prior

$$
\begin{equation*}
\min _{z}\|A \cdot G(z)-b\|_{2}^{2} \tag{8.14}
\end{equation*}
$$

which is a particular case of template (8.1) (without splitting). They also provide a statistical generalization error dependent on a certain set restricted isometry property on the matrix
A. More generally, Theorem 8.3 in Supplementary 8.6 provides statistical guarantees for Algorithm 8.1 using the standard notion of empirical Rademacher complexity (Mohri et al., 2018c).

Hand and Voroninski (2017) analyze the optimization landscape of (8.14) under the assumption that $G(i)$ is composed of linear layers and ReLU activation functions, $(i i)$ is sufficiently expansive at each layer and (iii) the network's weights have a Gaussian distribution or an equivalent deterministic weight distribution condition. Under such conditions, they show global existence of descent directions outside small neighborhoods around two points, but do not provide algorithmic convergence rates. Their analysis requires ReLU activation in all layers of the generator $G$, including the last one, which is often not met in practice.

On the other hand, our framework is not restricted to a particular network architecture and instead isolates the necessary assumptions on the network $G$ for the success of Algorithm 8.1. In doing so, we effectively decouple the learning task from the network structure $G$ and study them separately in Theorem 2.1 and Proposition 8.1, respectively. In particular, our theory in Section 8.3 (Supplementary 8.6) applies broadly to any nonlinear map $G$ that meets Assumptions 8.1-8.3 (Assumptions 8.2-8.5), respectively.

In turn, Proposition 8.1 establishes that the standard feed forward network with common differentiable activation functions almost surely meets these assumptions. In this sense, let us also point to the work of Oymak et al. (2018), which is limited to linear regression with a nonlinear constraint, with its convex analogue studied in (Agarwal et al., 2010; Giryes et al., 2016).

Heckel et al. (2019) provides a convergence proof for a modified version of gradient descent, limited to (8.14) and without specifying a rate. We provide the convergence rate for a broad range of learning problems, and study the statistical generalization. Hand et al. (2018) studied the phase retrieval problem, with a non-convex objective function that is not directly covered by (8.1).

For the problem (8.14), Shah and Hegde (2018); Hegde (2018) proposed to use Projected Gradient Descent (PGD) after splitting in a manner similar to our template (8.1). If the projection (onto the range of the prior $G$ ) is successful, and under certain additional conditions, the authors establish linear convergence of PGD to a minimizer of (8.14). However, the projection onto the nonlinear range of $G$ is itself a difficult non-convex program without any theoretical guarantees. In contrast, we can solve the same problem without any projections while still providing a convergence rate.

From an optimization perspective, there are no fast rates for linearized ADMM with nonlinear constraints to our knowledge, but convergence to a first-order stationary point and special cases in a few different settings have been studied (Liu et al., 2017; Shen et al., 2016; Chen and Gu, 2014; Qiao et al., 2016). Let us again emphasize that Assumptions 8.2 and 8.3 extract the key attributes of $G$ necessary for the success of Algorithm 8.1, which is therefore not limited
to a generator network. It is also worth noting another line of work that applies tools from statistical physics to inference with deep neural networks, see (Manoel et al., 2017; Rezende et al., 2014) and the references therein.

### 8.5 Experiments

In this section we evaluate our algorithms for image recovery tasks with a generative prior. The datasets we consider are the CelebA dataset of face images (Liu et al., 2015) and the MNIST dataset of handwritten digits (LeCun and Cortes, 2010b). We train a generator $G$ with ELU activation functions Clevert et al. (2015), in order to satisfy Assumption 8.2. The generators are trained using the Wasserstein GAN framework (Arjovsky et al., 2017). For the CelebA dataset we downsample the images to $64 \times 64$ pixels as in Gulrajani et al. (2017) and we use the same residual architecture (He et al., 2015) for the generator with four residual blocks followed by a convolutional layer. For MNIST, we use the same architecture as one in Gulrajani et al. (2017), which contains one fully connected layer followed by three deconvolutional layers.

We recover images on the range of the generator $G$, by choosing $z^{\star} \in \mathbb{R}^{s}$ and setting $w^{\star}:=G\left(z^{\star}\right)$ as the true image to be recovered. This sets the global minimum of our objective functions at zero, and allows us to illustrate and compare the convergence rates of various algorithms.

Our Algorithm 8.1 mantains iterates $\left\{w_{t}, z_{t}\right\}_{t}$ where $w_{t}$ might not be feasible, namely, $w_{t}$ might not be in the range of $G$. As the goal in the following tasks is to recover an element in the range of $G$ (feasible points of (8.1)), we plot the objective value at the point $G\left(z_{t}\right)$.

Baseline. We compare to the most widely-used algorithm in the current literature, the gradient descent algorithm (GD) as used in (Bora et al., 2017), where a fixed number of iterations with constant step size are performed for the function $L(G(z))$. We tune its learning rate to be as large as possible without overshooting. (See Supplementary 8.13 for details on the hyperparameter tuning).

Our goal is to illustrate our theoretical results and highlight scenarios where Algorithm 8.1 can have better performance than GD in optimization problems with a generative prior. Hence, we do not compare with sparsity-prior based algorithms, such as LASSO (Tibshirani, 1996), or argue about GAN vs. sparsity priors as in Bora et al. (2017).

Our algorithms. We will use ( $i$ ) (linearized) ADMM (Algorithm 8.1), and (ii) ADMM with exact minimization (EADMM), described in section 8.14. For both ADMM and EADMM, we choose a starting iterate (random $z_{0}$ and $w_{0}=G\left(z_{0}\right)$ ) and initial dual variable $\lambda_{0}=0$ (for GD we choose the same $z_{0}$ as initial iterate).. We carefully track the objective function value vs. computation time for a fair comparison.

Compressive sensing. The exact minimization step of EADMM involves the solution of a system of linear equations in each iteration. Performing Singular Value Decomposition (SVD) once on the measurement matrix $A$, and storing its components in memory, allows us to


Figure 8.1: Reconstruction error and measurement error vs time. MNIST (left) and CelebA (right)


Figure 8.2: Test error on denoised adversarial Figure 8.3: $\ell_{\infty}$ reconstruction error per iteraexamples vs computation time.
 tion for ADAM, GD, and EADMM.
solve such linear systems with a very low per-iteration complexity (see Supplementary 8.13.3). We plot the objective function value as well as the reconstruction error with $50 \%$ relative measurements in Figure 8.1(average over 20 images (MNIST) and 10 images (CelebA)).

Adversarial Denoising with $\ell_{\infty}$-norm Projection onto the range of a deep-net prior has been considered by Samangouei et al. (2018); Ilyas et al. (2017) as a defense mechanism against adversarial examples (Szegedy et al., 2014a). In their settings, samples are denoised with a generative prior, before being fed to a classifier. Even though the adversarial noise introduced is typically bounded in $\ell_{\infty}$-norm, the projection is done in $\ell_{2}$-norm. Such projection corresponds to $F(w, z)=\left\|w-w^{\natural}\right\|^{2}$ in (8.1).

We instead propose to project using the $\ell_{\infty}$-norm that bounds the adversarial perturbation. To this end we let $F(w, z)=\gamma\left\|w-w^{\natural}\right\|_{2}^{2}+\left\|w-w^{\natural}\right\|_{\infty}$ in the template (8.1), for some small value of $\gamma$. The proximal of the $\ell_{\infty}$ norm is efficiently computable (Duchi et al., 2008b), allowing us to split $F(w, z)$ in its components $L(w)=\gamma\left\|w-w^{\natural}\right\|_{2}^{2}$ and $R(w)=\left\|w-w^{\natural}\right\|_{\infty}$ (Note that the small $\gamma$ ensures that Assumption 8.1 holds)

We compare the ADAM optimizer (Kingma and Ba , 2014), GD and ADMM ( 450 iterations and for GD and ADAM, and 300 iterations for EADMM). We use ADAM to solve the $\ell_{2}$ projection, while ADMM solves the $\ell_{\infty}$ projection. We evaluate on a test set of 2000 adversarial examples from the MNIST dataset, obtained with the Projected Gradient Algorithm of Madry et al. (2018b) with 30 iterations, stepsize 0.01 and attack size 0.2 . For the classifier, we use a standard convolutional network trained on clean MNIST samples. We also test ADAM, GD (3000 iterations) and EADMM (2000 iterations) on the $\ell_{\infty}$ denoising task.

The test error as a function of computation time is in Figure 8.2. We observe that the $\ell_{\infty}$
denoising performs better when faced with $\ell_{\infty}$ bounded attacks, in the sense that it achieves a lower error with less computation time. In Figure 8.3, we plot the $\ell_{\infty}$ reconstruction error achieved by ADAM, GD and EADMM, averaged over 7 images. GD was unable to decrease the initial error, while ADAM takes a considerable number of iterations to do so. In contrast, our ADMM already achieves the final error of ADAM within its first 100 iterations.

### 8.6 Appendix: Statistical Learning with Generative Priors

So far, we have assumed $L$ to be strongly convex in (8.1), see Assumption 8.1 and Theorem 2.1. In this section, we relax this assumption on $L$ in the context of statistical learning with generative priors, thus extending Theorem 2.1 to applications such as compressive sensing. We also provide the corresponding generalization error in this section.

Here, we follow the standard setup in learning theory Mohri et al. (2018c). Consider the probability space $(\mathbb{X}, \chi)$, where $\mathbb{X} \subset \mathbb{R}^{d}$ is a compact set, equipped with the Borel sigma algebra, and $\chi$ is the corresponding probability measure. To learn an unknown parameter $w^{\natural} \in \mathbb{R}^{d}$, consider the optimization program

$$
\begin{equation*}
\min _{w \in \mathbb{R}^{p}} L(w), \quad L(w):=\mathbb{E}_{x \sim \chi} l(w, x), \tag{8.15}
\end{equation*}
$$

where $L: \mathbb{R}^{p} \rightarrow \mathbb{R}$ is the differentiable population risk and $l: \mathbb{R}^{d} \times \mathbb{R}^{p} \rightarrow \mathbb{R}$ is the corresponding loss function. We also assume that Program (8.15) has a unique solution $w^{\natural} \in \mathbb{R}^{p}$. The probability measure $\chi$ above is itself often unknown and we instead have access to $m$ samples drawn independently from $\chi$, namely, $\left\{x_{i}\right\}_{i=1}^{m} \sim \chi$. This allows us to form the empirical loss

$$
\begin{equation*}
L_{m}(w):=\frac{1}{m} \sum_{i=1}^{m} l\left(w, x_{i}\right) \tag{8.16}
\end{equation*}
$$

Often, $m \ll p$ and to avoid an ill-posed problem, we must leverage any inherent structure in $w^{\natural}$. In this work, we consider a differentiable map $G: \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ and we assume that $w^{\natural} \in G\left(\mathbb{R}^{s}\right)$. That is, there exists $z^{\natural} \in \mathbb{R}^{s}$ such that $w^{\natural}=G\left(z^{\natural}\right)$. While not necessary, we limit ourselves in this section to the important case where $G$ corresponds to a neural network, see Section 8.1.

To learn $w^{\natural}$ with the generative prior $w^{\natural}=G\left(z^{\natural}\right)$, we propose to solve the program

$$
\begin{array}{ll}
\underset{w, z}{\operatorname{minimize}} & L_{m}(w)+R(w)+H(z)  \tag{8.17}\\
\text { subject to } & w=G(z),
\end{array}
$$

where $R: \mathbb{R}^{p} \rightarrow \mathbb{R}$ and $H: \mathbb{R}^{s} \rightarrow \mathbb{R}$ are convex but not necessarily smooth. Depending on the specific problem at hand, the regularizers $R$ and $H$ allow us to impose additional structure on $w$ and $z$, such as sparsity or set inclusion. Throughout, we again require that the proximal maps (Parikh et al., 2014) for $R$ and $H$ can be computed efficiently, as detailed in Section 8.2.

Let us now state our assumptions, some of which differ from Section 8.3.

Assumption 8.4. Convexity / strong smoothness of loss: We assume that $l(\cdot, \cdot)$ is convex in both of its arguments. Moreover, we assume that $l(w, \cdot)$ is strongly smooth, namely, there exists $\sigma_{l} \geq 0$ such that for every $x, x^{\prime} \in \mathbb{X}$

$$
\begin{equation*}
D_{l}\left(x, x^{\prime} ; w\right) \leq \frac{\sigma_{l}}{2}\left\|x-x^{\prime}\right\|_{2}^{2} \tag{8.18}
\end{equation*}
$$

where $D_{l}$ stands for the Bregman divergence associated with $l(w, \cdot)$,

$$
D_{l}\left(x, x^{\prime} ; w\right)=l\left(w, x^{\prime}\right)-l(w, x)-\left\langle x^{\prime}-x, \nabla_{x} l(w, x)\right\rangle .
$$

Assumption 8.5. Strong convexity / smoothness of the population risk: We assume that the population risk $L$ defined as

$$
\begin{equation*}
L(w):=\mathbb{E}_{x \sim \chi} l(w, x), \tag{8.19}
\end{equation*}
$$

is both strongly convex and smooth, i.e., there exist $0<\zeta_{L} \leq \sigma_{L}$ such that

$$
\begin{align*}
& \frac{\zeta_{L}}{2}\left\|w-w^{\prime}\right\|^{2} \leq D_{L}\left(w, w^{\prime}\right) \leq \frac{\sigma_{L}}{2}\left\|w-w^{\prime}\right\|^{2}, \\
& D_{L}\left(w, w^{\prime}\right)=L\left(w^{\prime}\right)-L(w)-\left\langle w^{\prime}-w, \nabla L(w)\right\rangle \tag{8.20}
\end{align*}
$$

for every $w, w^{\prime} \in \mathbb{R}^{d}$. In the following we denote by $w^{\natural}$ the minimizer of (8.19). In view of our assumption, such minimizer is unique.

Assumptions 8.4 and 8.5 are standard in statistical learning Mohri et al. (2018c). For example, in linear regression, we might take

$$
\begin{gathered}
l(w, x)=\frac{1}{2}\left|\left\langle w-w^{\natural}, x\right\rangle\right|^{2}, \\
L_{m}(w)=\frac{1}{2 m} \sum_{i=1}^{m}\left|\left\langle w-w^{\natural}, x_{i}\right\rangle\right|^{2},
\end{gathered}
$$

for which both Assumptions 8.4 and 8.5 are met. Lastly, we require that the Assumptions 8.2 and 8.3 on $G$ hold in this section, see and Proposition 8.1 for when these assumptions hold for generative priors.

As a consequence of Assumption 8.4, we have that $L_{m}$ is convex. We additionally require $L_{m}$ to be strongly convex and smooth in the following restricted sense. Even though $L_{m}$ is random because of its dependence on the random training data $\left\{x_{i}\right\}_{i=1}^{m}$, we ensure later in this section that the next condition is indeed met with high probability when $m$ is large enough.

Definition 8.1. Restricted strong convexity / smoothness of empirical loss: We say that $L_{m}$ is strongly convex and smooth on the set $W \subset \mathbb{R}^{p}$ if there exist $0<\mu_{L} \leq v_{L}$ and $\bar{\mu}_{L}, \bar{v}_{L} \geq 0$ such
that

$$
\begin{gather*}
D_{L_{m}}\left(w, w^{\prime}\right) \geq \frac{\mu_{L}}{2}\left\|w^{\prime}-w\right\|_{2}^{2}-\bar{\mu}_{L}, \\
D_{L_{m}}\left(w, w^{\prime}\right) \leq \frac{v_{L}}{2}\left\|w^{\prime}-w\right\|_{2}^{2}+\bar{v}_{L},  \tag{8.21}\\
D_{L_{m}}\left(w, w^{\prime}\right):=L_{m}\left(w^{\prime}\right)-L_{m}(w)-\left\langle w^{\prime}-w, \nabla L_{m}(w)\right\rangle,
\end{gather*}
$$

for every $w, w^{\prime} \in W$.
Under the above assumptions, a result similar to Theorem 2.1 holds, which we state without proof.

Theorem 8.2. (guarantees for Algorithm 8.1) Suppose that Assumptions 8.2-8.5 hold. Let ( $w^{*}, z^{*}$ ) be a solution of program (8.1) and let $\lambda^{*}$ be a corresponding optimal dual variable. Let also $\left\{w_{t}, z_{t}, \lambda_{t}\right\}_{t \geq 0}$ denote the output sequence of Algorithm 8.1. Suppose that $L_{m}$ satisfies the restricted strong convexity and smoothness in Definition 8.1 for a set $W \subset \mathbb{R}^{p}$ that contains a solution $w^{*}$ of (8.1) and all the iterates $\left\{w_{t}\right\}_{t \geq 0}$ of Algorithm 8.1. ${ }^{1}$ Suppose also that the primal step sizes $\alpha, \beta$ in Algorithm 8.1 satisfy

$$
\begin{equation*}
\alpha \leq \frac{1}{v_{\rho}}, \quad \beta \leq \frac{1}{\xi_{\rho}+2 \alpha \tau_{\rho}^{2}} . \quad \sigma_{0} \leq \sigma_{0, \rho} \tag{8.2}
\end{equation*}
$$

Then it holds that

$$
\begin{gather*}
\frac{\left\|w_{t}-w^{*}\right\|_{2}^{2}}{\alpha}+\frac{\left\|z_{t}-z^{*}\right\|_{2}^{2}}{\beta} \leq 2\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{\bar{\eta}_{\rho}}{\rho},  \tag{8.23}\\
\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2} \leq \frac{4\left(1-\eta_{\rho}\right)^{t} \Delta_{0}}{\rho}+\frac{\widetilde{\eta}_{\rho}}{\rho^{2}}, \tag{8.24}
\end{gather*}
$$

for every iteration $t$. Above, $\Delta_{0}=\mathscr{L}_{\rho}\left(w_{0}, z_{0}, \lambda_{0}\right)-\mathscr{L}_{\rho}\left(w^{*}, z^{*}, \lambda^{*}\right)$ is the initialization error, see (8.2). The convergence rate $1-\eta_{\rho} \in(0,1)$ and the quantities $v_{\rho}, \xi_{\rho}, \tau_{\rho}, \sigma_{0, \rho}, \bar{\eta}_{\rho}, \tilde{\eta}_{\rho}$ above depend on the parameters in the Assumptions 8.2-8.5 and on $\lambda_{0}, \sigma_{0}$.

The remarks after Theorem 2.1 apply here too.

### 8.6.1 Generalization Error

Building upon the optimization guarantee in Theorem 8.3, our next result in this section is Theorem 8.3, which quantifies the convergence of the iterates $\left\{w_{t}\right\}_{t \geq 0}$ of Algorithm 8.1 to the true parameter $w^{\natural}$.

[^13]In other words, Theorem 8.3 below controls the generalization error of (8.1), namely, the error incurred by using the empirical risk $L_{m}$ in lieu of the population risk $L$. Indeed, Theorem 2.1 is silent about $\left\|w_{t}-w^{\natural}\right\|_{2}$. We address this shortcoming with the following result, proved in Section 8.12 of the supplementary material.
Lemma 8.1. Let $R=1_{W}$ be the indicator function on $W \subset \mathbb{R}^{p}$ and set $H=0$ in (8.1). ${ }^{2}$ Suppose that $w^{*}$ belongs to the relative interior of $W$. Then it holds that

$$
\begin{equation*}
\left\|w^{\natural}-w^{*}\right\|_{2} \leq \frac{1}{\zeta_{L}} \max _{w \in W}\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2} . \tag{8.25}
\end{equation*}
$$

Before bounding the right-hand side of (8.25), we remark that it is possible to extend Lemma 8.1 to the case where the regularizer $R$ is a decomposable norm, along the lines of Negahban et al. (2012). We will however not pursue this direction in the present work. Next note that (8.23) and Lemma 8.1 together imply that

$$
\begin{align*}
\frac{\left\|w_{t}-w^{\natural}\right\|_{2}^{2}}{\alpha^{2}} & \leq\left(\frac{\left\|w_{t}-w^{*}\right\|_{2}}{\alpha}+\frac{\left\|w^{*}-w^{\natural}\right\|_{2}}{\beta}\right)^{2} \quad \text { (triangle inequality) } \\
& \leq \frac{2\left\|w_{t}-w^{*}\right\|_{2}^{2}}{\alpha^{2}}+\frac{2\left\|w^{*}-w^{\natural}\right\|_{2}^{2}}{\beta^{2}} \quad\left((a+b)^{2} \leq 2 a^{2}+2 b^{2}\right) \\
& \leq 4\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2 \bar{\eta}_{\rho}}{\rho}+\frac{2}{\zeta_{L}^{2}} \max _{w \in W}\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2}^{2} \tag{8.26}
\end{align*}
$$

According to Theorem 2.1, the right-hand side of (8.26) depends on $\mu_{L}, \bar{\mu}_{L}, v_{L}, \bar{v}_{L}$, which were introduced in Definition 8.1. Note that $\mu_{L}, \bar{\mu}_{L}, v_{L}, \bar{v}_{L}$ and the right-hand side of (8.25) are all random variables because they depend on $L_{m}$ and thus on the randomly drawn training data $\left\{x_{i}\right\}_{i=1}^{m}$. To address this issue, we apply a basic result in statistical learning theory as follows. For every $w \in \mathbb{R}^{p}$ and every pair $x, x^{\prime} \in \mathbb{X}$, we use Assumption 8.4 to write that

$$
\begin{align*}
\left\|\nabla l(w, x)-\nabla l\left(w, x^{\prime}\right)\right\|_{2} & \leq \sigma_{l}\left\|x-x^{\prime}\right\|_{2} \quad(\text { see }(8.18)) \\
& \leq \sigma_{l} \operatorname{diam}(\mathbb{X}) \tag{8.27}
\end{align*}
$$

where $\operatorname{diam}(\mathbb{X})$ denotes the diameter of the compact set $\mathbb{X}$. Note also that

$$
\begin{equation*}
\mathbb{E}_{\left\{x_{i}\right\}_{i}} \nabla L_{m}(w)=\nabla L(w), \quad \forall w \in W \tag{8.28}
\end{equation*}
$$

where the expectation is over the training data $\left\{x_{i}\right\}_{i}$. Then, for $\varepsilon>0$ and except with a probability of at most $e^{-\varepsilon}$, it holds that

$$
\begin{align*}
& \left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2} \\
& \leq 2 \mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)+3 \sigma_{l} \operatorname{diam}(\mathbb{X}) \sqrt{\frac{\varepsilon+2}{2 m}} \\
& =: \Upsilon_{m, W}(\varepsilon), \tag{8.29}
\end{align*}
$$

[^14]for every $w \in W$ (Mohri et al., 2018c). Above,
\[

$$
\begin{equation*}
\mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)=\mathbb{E}_{E}\left[\max _{w \in W}\left\|\frac{1}{m} \sum_{i=1}^{m} e_{i} \nabla_{w} l\left(w, x_{i}\right)\right\|_{2}\right], \tag{8.30}
\end{equation*}
$$

\]

is the empirical Rademacher complexity and $E=\left\{e_{i}\right\}_{i}$ is a Rademacher sequence, namely, a sequence of independent random variables taking $\pm 1$ with equal probabilities. We can now revisit (8.26) and write that

$$
\begin{equation*}
\left\|w_{t}-w^{\natural}\right\|_{2}^{2} \leq 4 \alpha^{2}\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2 \alpha^{2} \bar{\eta}_{\rho}}{\rho}+\frac{2 \alpha^{2} \Upsilon_{m, W}^{2}(\varepsilon)}{\zeta_{L}^{2}} \tag{8.31}
\end{equation*}
$$

which holds except with a probability of at most $e^{-\varepsilon}$. In addition, for every $w, w^{\prime} \in W$, we may write that

$$
\begin{align*}
& \left\|\nabla L_{m}(w)-\nabla L_{m}\left(w^{\prime}\right)\right\|_{2} \\
& \leq\left\|\nabla L(w)-\nabla L\left(w^{\prime}\right)\right\|_{2}+\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2} \\
& \quad+\left\|\nabla L_{m}\left(w^{\prime}\right)-\nabla L\left(w^{\prime}\right)\right\|_{2} \quad \text { (triangle inequality) } \\
& \leq \sigma_{L}\left\|w-w^{\prime}\right\|_{2}+2 \Upsilon_{m, W}(\varepsilon), \quad \text { (see (8.20,8.29)) } \tag{8.32}
\end{align*}
$$

except with a probability of at most $e^{-\varepsilon}$. Likewise, for every $w, w^{\prime} \in W$, we have that

$$
\begin{align*}
& \left\|\nabla L_{m}(w)-\nabla L_{m}\left(w^{\prime}\right)\right\|_{2} \\
& \geq\left\|\nabla L_{m}(w)-\nabla L_{m}(w)\right\|_{2}-\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2} \\
& \quad-\left\|\nabla L_{m}\left(w^{\prime}\right)-\nabla L\left(w^{\prime}\right)\right\|_{2} \quad \text { (triangle inequality) } \\
& \geq \zeta_{L}\left\|w-w^{\prime}\right\|_{2}-2 \Upsilon_{m, W}(\varepsilon), \quad \text { (see }(8.20,8.29) \text { ) } \tag{8.33}
\end{align*}
$$

except with a probability of at most $e^{-\varepsilon}$. Therefore, $L_{m}$ satisfies the restricted strong convexity and smoothness in Definition 8.1 with

$$
\begin{align*}
& \mu_{L}=\sigma_{L}, \quad v_{L}=\zeta_{L}, \\
& \bar{\mu}_{L}=\bar{\zeta}_{L}=2 \Upsilon_{m, W}(\varepsilon) \tag{8.34}
\end{align*}
$$

Our findings in this section are summarized below.

Theorem 8.3. (generalization error) Suppose that Assumptions 8.2-8.5 hold and recall that the training samples $\left\{x_{i}\right\}_{i=1}^{m}$ are drawn independently from the probability space $(\mathbb{X}, \chi)$ for a compact set $\mathbb{X} \subset \mathbb{R}^{d}$ with diameter $\operatorname{diam}(\mathbb{X})$.

For a set $W \subset \mathbb{R}^{p}$, let $R=1_{W}$ be the indicator function on $W$, and set $H \equiv 0$ in (8.1). Suppose that solution $w^{*}$ of (8.1) belongs to the relative interior of $W$. For $\varepsilon>0$, evaluate the quantities in Theorem 8.2 with

$$
\mu_{L}=\sigma_{L}, \quad v_{L}=\zeta_{L}
$$

$$
\begin{align*}
\bar{\mu}_{L}=\bar{\zeta}_{L} & =4 \mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right) \\
& +6 \sigma_{l} \operatorname{diam}(\mathbb{X}) \sqrt{\frac{\varepsilon+2}{2 m}}, \tag{8.35}
\end{align*}
$$

where $\mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)$ in the empirical Rademacher complexity defined in (8.30). If the requirements on the step sizes in (8.22) hold, we then have that

$$
\begin{gather*}
\left\|w_{t}-w^{\natural}\right\|_{2}^{2} \leq 4 \alpha^{2}\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2 \alpha^{2} \bar{\eta}_{\rho}}{\rho}+\frac{8 \alpha^{2}}{\zeta_{L}^{2}} \mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)^{2} \\
+\frac{18 \alpha^{2} \sigma_{l}^{2} \operatorname{diam}(\mathbb{X})^{2}(\varepsilon+2)}{m} \tag{8.36}
\end{gather*}
$$

except with a probability of at most $e^{-\varepsilon}$.

Most of the remarks about Theorem 2.1 also apply to Theorem 8.3 and we note that $\left\|w_{t}-w^{\natural}\right\|_{2}$ reduces by increasing the number of training samples $m$, before asymptotically reaching the generalization error

$$
\begin{equation*}
2 \psi_{\rho}+\frac{8}{\zeta_{L}^{2}} \mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)^{2} \tag{8.37}
\end{equation*}
$$

Computing the Rademacher complexity above for specific choices of the network structure and loss is itself potentially a complicated task, which we will not pursue by the virtue of the generality of our results so far. The key technical challenge there is computing the corresponding entropy integral, which involves estimating the covering numbers of the set $W$ Mohri et al. (2018c). One last takeaway point from the statistical accuracy in (8.37) is the following. If

$$
\begin{equation*}
\bar{\eta}_{\rho}=O\left(\rho \cdot \mathscr{R}_{W}\left(x_{1}, \cdots, x_{m}\right)^{2} / \zeta_{L}^{2}\right) \tag{8.38}
\end{equation*}
$$

the asymptotic optimization error in Theorem 2.1 does not play an important role in determining the generalization error above. In words, if (8.38) holds, then Algorithm 8.1 converges to the ball of statistical accuracy around $w^{\natural}$. Here, $O$ stands for the standard Big-O notation.

### 8.7 Appendix: Proof of Proposition 8.1

The feedforward network $G=G_{\Xi}: \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ is a composition of linear maps and entry-wise applications of the activation functions, and hence is also of class $C^{1}$. Its Jacobian $D G$ : $\mathbb{R}^{s} \rightarrow \mathbb{R}^{d \times s}$ is thus a continuous function and its restriction to the compact subset $\mathscr{X} \subseteq \mathbb{R}^{s}$ is Lipschitz-continuous. Therefore, there exists $v_{G} \geq 0$ such that

$$
\left\|D G\left(z^{\prime}\right)-D G(z)\right\|_{2} \leq v_{G}\left\|z^{\prime}-z\right\|, \quad \forall z, z^{\prime} \in \mathscr{X}
$$

From standard arguments it then follows that Assumption 8.2 holds in the sense that

$$
\begin{aligned}
\left\|G\left(z^{\prime}\right)-G(z)-D G(z)\left(z^{\prime}-z\right)\right\|_{2} & =\left\|\int_{0}^{1}\left(D G\left(t z^{\prime}+(1-t) z\right)-D G(z)\right)\left(z^{\prime}-z\right) d t\right\|_{2} \\
& \leq \int_{0}^{1}\left\|D G\left(t z^{\prime}+(1-t) z\right)-D G(z)\right\|_{2}\left\|z^{\prime}-z\right\|_{2} d t \\
& \leq v_{G} \int_{0}^{1} t\left\|z^{\prime}-z\right\|^{2} d t=\frac{v_{G}}{2}\left\|z^{\prime}-z\right\|_{2}^{2}
\end{aligned}
$$

for every $z, z^{\prime} \in \mathbb{R}^{s}$.
In order to show that Assumption 8.3 (near-isometry) also holds, we will require the following simple fact:

Lemma 8.2. Let $G: \mathscr{X} \subseteq \mathbb{R}^{s} \rightarrow \mathbb{R}^{d}$ have a left inverse $H: G(\mathscr{X}) \subseteq \mathbb{R}^{d} \rightarrow \mathbb{R}^{s}$ which is Lipschitzcontinuous with constant $\iota_{G}>0$. Then it holds that

$$
\frac{1}{\iota_{G}}\left\|z^{\prime}-z\right\| \leq\left\|G\left(z^{\prime}\right)-G(z)\right\|, \quad \forall z^{\prime}, z \in D
$$

Proof.

$$
\left\|z^{\prime}-z\right\|=\left\|H\left(G\left(z^{\prime}\right)\right)-H(G(z))\right\| \leq \iota_{G}\left\|G\left(z^{\prime}\right)-G(z)\right\|
$$

We now proceed to show that Assumption 8.3 holds. We suppose $G$ is of the form

$$
G(z)=\omega_{k} W_{k}\left(\omega_{k-1} W_{k-1} \ldots\left(\omega_{1} W_{1} z\right) \ldots\right)
$$

for weight matrices $\left\{W_{k}\right\}_{k}$. First note that, by the compactness of the domain of $G$, the values of the hidden layers are always contained in a product of compact intervals, and so we can replace $\omega_{i}$ by its restriction to such sets. Each $\omega_{i}$ is continuous, defined on a product of intervals, and is stricly increasing so that they have a continuous left inverse $\omega_{i}^{-1}$ (Garling, 2014, Proposition 6.4.5). The assumption of non-decreasing layer sizes implies that the $W_{i}$ are tall matrices of dimensions ( $m_{i}, n_{i}$ ) with $m_{i} \geq n_{i}$, whose columns are almost surely linearly independent after an arbitrarily small perturbation. In such case they have a left matrix inverse $W_{i}^{-1}$, which as a bounded linear map, is continuous. It then follows that $G$ has a continuous left inverse of the form

$$
G^{-1}=W_{1}^{-1} \circ \omega_{1}^{-1} \ldots W_{k}^{-1} \circ \omega_{k}^{-1}
$$

which is a continuous mapping and is defined on $G(\mathscr{X})$ which by continuity of $G$ is compact, hence $G^{-1}$ is Lipschitz-continuous. The result then follows by the Lipschitz continuity of the map $G$ (restricted to the compact domain $\mathscr{X}$ ) and Lemma 8.2.

### 8.8 Appendix: Proof of Theorem 8.1

It is convenient throughout the supplementary material to use a slightly different notation for Lagrangian, compared to the body of the paper. To improve the readability of the proof, let us list here the assumptions on the empirical loss $L$ and prior $G$ that are used throughout this proof. For every iteration $t$, we assume that

$$
\begin{gather*}
L\left(w_{t}\right)-L\left(w^{*}\right)-\left\langle w_{t}-w^{*}, \nabla L\left(w^{*}\right)\right\rangle \\
\geq \frac{\mu_{L}}{2}\left\|w_{t}-w^{*}\right\|_{2}^{2}, \quad(\text { strong convexity of } L)  \tag{8.39}\\
L\left(w_{t+1}\right)-L\left(w_{t}\right)-\left\langle w_{t+1}-w_{t}, \nabla L\left(w_{t}\right)\right\rangle \\
\leq \frac{v_{L}}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2}, \quad(\text { strong smoothness of } L)  \tag{8.40}\\
\\
\left\|G\left(z^{\prime}\right)-G(z)-D G(z) \cdot\left(z^{\prime}-z\right)\right\|_{2}  \tag{8.41}\\
\leq \frac{v_{G}}{2}\left\|z^{\prime}-z\right\|_{2}^{2}, \quad(\text { strong smoothness of } G)  \tag{8.42}\\
\left.l_{G}\left\|z^{\prime}-z\right\|_{2} \leq\left\|G\left(z^{\prime}\right)-G(z)\right\|_{2} \leq \kappa_{G}\left\|z^{\prime}-z\right\|_{2}, \quad \quad \text { (near-isometry of } G\right)  \tag{8.43}\\
\left\|D G(z) \cdot\left(z^{\prime}-z\right)\right\|_{2} \leq \kappa_{G}\left\|z^{\prime}-z\right\|_{2}, \quad \quad(\text { Lipschitz continuty of } G)
\end{gather*}
$$

For the sake of brevity, let us set

$$
\begin{gather*}
\nu=(w, z) \in \mathbb{R}^{p+s}, \\
\mathscr{L}_{\rho}(\nu, \lambda):=\mathscr{L}_{\rho}(w, z, \lambda):=L(w)+R(w)+H(z)+\langle w-G(z), \lambda\rangle \\
+\frac{\rho}{2}\|w-G(z)\|_{2}^{2}, \quad \text { (augmented Lagrangian) }  \tag{8.44}\\
\mathscr{L}_{\rho}^{\prime}(\nu, \lambda):=\mathscr{L}_{\rho}^{\prime}(w, z, \lambda)=L(w)+\langle w-G(z), \lambda\rangle+\frac{\rho}{2}\|w-G(z)\|_{2}^{2},  \tag{8.45}\\
A(\nu)=A(w, z):=w-G(z) . \quad \text { (feasibility gap) } \tag{8.46}
\end{gather*}
$$

Let also $v^{*}=\left(w^{*}, z^{*}\right)$ be a solution of (8.1) and let $\lambda^{*}$ be a corresponding optimal dual variable. The first-order necessary optimality conditions for (8.1) are

$$
\left\{\begin{array}{l}
-\nabla_{v} \mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right) \in \partial R\left(w^{*}\right) \times \partial H\left(z^{*}\right)  \tag{8.47}\\
w^{*}=G\left(z^{*}\right)
\end{array}\right.
$$

where $\partial R\left(w^{*}\right)$ and $\partial H\left(z^{*}\right)$ are the subdifferentials of $R$ and $H$, respectively, at $w^{*}$ and $z^{*}$. Throughout the proof, we will also often use the notation

$$
\begin{gather*}
\Delta_{t}:=\mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right)  \tag{8.48}\\
\Delta_{t}^{\prime}:=\mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)  \tag{8.49}\\
\delta_{t}:=\left\|w_{t}-w^{*}\right\|_{2}, \quad \delta_{t}^{\prime}:=\left\|z_{t}-z^{*}\right\|_{2}  \tag{8.50}\\
A_{t}:=A\left(v_{t}\right)=w_{t}-G\left(z_{t}\right) \tag{8.51}
\end{gather*}
$$

In particular, with this new notation, the dual update can be rewritten as

$$
\begin{equation*}
\lambda_{t+1}=\lambda_{t}+\sigma_{t+1} A_{t+1} . \quad(\text { see Algorithm 8.1) } \tag{8.52}
\end{equation*}
$$

First, in Appendix 8.9, we control the smoothness of $\mathscr{L}_{\rho}^{\prime}$ over the trajectory of the algorithm.

Lemma 8.3. For every iteration $t$, it holds that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t+1}, z_{t+1}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\left\langle w_{t+1}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle \\
& \leq \frac{v_{\rho}}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2} \tag{8.53}
\end{align*}
$$

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1} \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)-\left\langle z_{t+1}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle \\
& \leq \frac{\xi_{\rho}}{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}  \tag{8.54}\\
& \left\|\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\|_{2} \leq \tau_{\rho}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \tag{8.55}
\end{align*}
$$

where

$$
\begin{gather*}
v_{\rho}:=v_{L}+\rho \\
\xi_{\rho}:=v_{G}\left(\lambda_{\max }+\rho \max _{i}\left\|A_{i}\right\|_{2}\right)+2 \rho \kappa_{G}^{2} \tag{8.57}
\end{gather*}
$$

$$
\begin{equation*}
\tau_{\rho}:=\rho \kappa_{G} . \tag{8.58}
\end{equation*}
$$

Second, in the following result we ensure that $\mathscr{L}_{\rho}$ and $\mathscr{L}_{\rho}^{\prime}$ are sufficiently regular along the trajectory of our algorithm, see Appendix 8.10 for the proof.

Lemma 8.4. For every iteration $t$, it holds that

$$
\begin{gather*}
\Delta_{t} \geq \frac{\mu_{\rho} \delta_{t}^{2}}{2}+\frac{\mu_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}-\bar{\mu}_{\rho}  \tag{8.59}\\
\Delta_{t}^{\prime}+\left\langle\nu^{*}-v_{t}, \nabla_{\nu} \mathscr{L}_{\rho}^{\prime}\left(v_{t}\right)\right\rangle \leq \frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2} \tag{8.60}
\end{gather*}
$$

where

$$
\begin{gather*}
\mu_{\rho}:=\mu_{L}-2 \rho, \quad \mu_{\rho}^{\prime}:=\frac{\rho l_{G}^{2}}{2}-v_{G}\left\|\lambda^{*}\right\|_{2}  \tag{8.61}\\
\bar{\mu}_{\rho}:=\frac{3}{\rho}\left(\lambda_{\max }^{2}+\left\|\lambda^{*}\right\|_{2}^{2}\right)  \tag{8.62}\\
\omega_{\rho}:=0, \quad \omega_{\rho}^{\prime}:=\frac{v_{G}}{2}\left(\lambda_{\max }+\rho\right) \tag{8.63}
\end{gather*}
$$

Having listed all the necessary technical lemmas above, we now proceed to prove Theorem 2.1. Using the smoothness of $\mathscr{L}_{\rho}^{\prime}$, established in Lemma 8.3, we argue that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(v_{t+1}, \lambda_{t+1}\right) \\
& =L\left(w_{t+1}\right)+\left\langle A_{t+1}, \lambda_{t+1}\right\rangle+\frac{\rho}{2}\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.45)) } \\
& =L\left(w_{t+1}\right)+\left\langle A_{t+1}, \lambda_{t}\right\rangle+\left(\frac{\rho}{2}+\sigma_{t+1}\right)\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.52)) } \\
& =\mathscr{L}_{\rho}^{\prime}\left(w_{t+1}, z_{t+1}, \lambda_{t}\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.44)) } \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\left\langle w_{t+1}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle+\frac{v_{\rho}}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2} \\
& \quad+\bar{v}_{\rho}+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad(\text { see }(8.53)) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\left\langle w_{t+1}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle+\frac{1}{2 \alpha}\left\|w_{t+1}-w_{t}\right\|_{2}^{2} \\
& \quad  \tag{8.64}\\
& \quad+\bar{v}_{\rho}+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2}
\end{align*}
$$

where the last line above holds if the step size $\alpha$ satisfies

$$
\begin{equation*}
\alpha \leq \frac{1}{v_{\rho}} . \tag{8.65}
\end{equation*}
$$

According to Algorithm 8.1, we can equivalently write the $w$ updates as

$$
\begin{equation*}
w_{t+1}=\arg \min _{w}\left\langle w-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle+\frac{1}{2 \alpha}\left\|w-w_{t}\right\|_{2}^{2}+R(w) . \tag{8.66}
\end{equation*}
$$

In particular, consider above the choice of $w=\theta w^{*}+(1-\theta) w_{t}$ for $\theta \in[0,1]$ to be set later. We can then bound the last line of (8.64) as

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(v_{t+1}, \lambda_{t+1}\right)+R\left(w_{t+1}\right) \\
& =\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\min _{w}\left\langle w-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle \\
& \quad+\frac{1}{2 \alpha}\left\|w-w_{t}\right\|_{2}^{2}+R(w)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad(\text { see }(8.64,8.66)) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{2 \alpha} \\
& \quad+\theta R\left(w^{*}\right)+(1-\theta) R\left(w_{t}\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad(\text { convexity of } R) \\
& =\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{2 \alpha} \\
& \quad+\theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle \\
& \quad+\theta R\left(w^{*}\right)+(1-\theta) R\left(w_{t}\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \tag{8.67}
\end{align*}
$$

The last inner product above can be controlled as

$$
\begin{align*}
& \theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle \\
& \leq \frac{\theta^{2} \delta_{t}^{2}}{2 \alpha}+\frac{\alpha}{2}\left\|\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\|_{2}^{2} \quad\left(2\langle a, b\rangle \leq\|a\|_{2}^{2}+\|b\|_{2}^{2}\right. \text { and (8.5 } \\
& \leq \frac{\theta^{2} \delta_{t}^{2}}{2 \alpha}+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}, \quad \text { (see (8.55)) } \tag{8.68}
\end{align*}
$$

which, after substituting in (8.67), yields that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(v_{t+1}, \lambda_{t+1}\right)+R\left(w_{t+1}\right) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{\alpha} \\
& \quad+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}+\theta R\left(w^{*}\right)+(1-\theta) R\left(w_{t}\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \tag{8.69}
\end{align*}
$$

Regarding the right-hand side above, the smoothness of $\mathscr{L}_{\rho}^{\prime}$ in Lemma 8.3 allows us to write that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)+\left\langle z_{t+1}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle \\
& \quad+\left(\frac{\xi_{\rho}}{2}+\alpha \tau_{\rho}^{2}\right)\left\|z_{t+1}-z_{t}\right\|_{2}^{2} . \quad \text { (see (8.54)) } \tag{8.70}
\end{align*}
$$

If we assume that the primal step sizes $\alpha, \beta$ satisfy

$$
\begin{equation*}
\frac{\xi_{\rho}}{2}+\alpha \tau_{\rho}^{2} \leq \frac{1}{2 \beta} \tag{8.71}
\end{equation*}
$$

we can simplify (8.70) as

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)+\left\langle z_{t+1}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{1}{2 \beta}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \tag{8.72}
\end{align*}
$$

From Algorithm 8.1, recall the equivalent expression of the $z$ updates as

$$
\begin{equation*}
z_{t+1}=\arg \min _{z}\left\langle z-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{1}{2 \beta}\left\|z-z_{t}\right\|_{2}^{2}+H(z), \tag{8.73}
\end{equation*}
$$

and consider the choice of $z=\theta z^{*}+(1-\theta) z_{t}$ above, with $\theta \in[0,1]$ to be set later. Combining (8.72,8.73) leads us to

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}+H\left(z_{t+1}\right) \\
& =\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)+\min _{z}\left\langle z-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{1}{2 \beta}\left\|z-z_{t}\right\|_{2}^{2}+H(z) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)+\theta\left\langle z^{*}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta}+H\left(\theta z^{*}+(1-\theta) z_{t}\right) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)+\theta\left\langle z^{*}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& \quad+\theta H\left(z^{*}\right)+(1-\theta) H\left(z_{t}\right) . \quad(\text { convexity of } H) \tag{8.74}
\end{align*}
$$

By combining ( $8.69,8.74$ ), we reach

$$
\begin{aligned}
& \mathscr{L}_{\rho}\left(v_{t+1}, \lambda_{t+1}\right) \\
& =\mathscr{L}_{\rho}^{\prime}\left(v_{t+1}, \lambda_{t+1}\right)+R\left(w_{t+1}\right)+H\left(z_{t+1}\right) \quad(\text { see }(8.44,8.45)) \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)+\theta\left\langle w^{*}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\alpha \tau_{\rho}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \\
& \quad+\theta R\left(w^{*}\right)+(1-\theta) R\left(w_{t}\right)+H\left(z_{t+1}\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.69)) } \\
& \leq \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)+\theta\left\langle v^{*}-v_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& \quad+\theta R\left(z^{*}\right)+(1-\theta) R\left(z_{t}\right)+\theta H\left(z^{*}\right)+(1-\theta) H\left(z_{t}\right) \\
& \quad+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad(\operatorname{see}(8.74)) \\
& =\mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)+\theta\left\langle v^{*}-v_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)\right\rangle+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& \quad+\theta\left(R\left(z^{*}\right)+H\left(z^{*}\right)-R\left(z_{t}\right)-H\left(z_{t}\right)\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.44,8.45)) }
\end{aligned}
$$

$$
\begin{align*}
\leq & \mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)+\theta\left(\frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}-\Delta_{t}^{\prime}\right)+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& +\theta\left(R\left(z^{*}\right)+H\left(z^{*}\right)-R\left(z_{t}\right)-H\left(z_{t}\right)\right)+\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad \text { (see (8.60)) } \\
= & \mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)+\theta\left(\frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}-\Delta_{t}\right)+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& +\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2} \quad(\text { see }(8.44,8.45)) \tag{8.75}
\end{align*}
$$

After recalling (8.48) and by subtracting $\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right)$ from both sides, (8.75) immediately implies that

$$
\begin{align*}
\Delta_{t+1} \leq \Delta_{t} & +\frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}+\theta\left(\bar{\omega}_{\rho}-\Delta_{t}\right)+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta} \\
& +\sigma_{t+1}\left\|A_{t+1}\right\|_{2}^{2}, \quad(\text { see }(8.48,8.75)) \tag{8.76}
\end{align*}
$$

where we also used the assumption that $\theta \leq 1$ above. To remove the feasibility gap $\left\|A_{t+1}\right\|_{2}$ from the right-hand side above, we write that

$$
\begin{array}{rlr}
\left\|A_{t+1}\right\|_{2} & =\left\|w_{t+1}-G\left(z_{t+1}\right)\right\|_{2} \quad(\operatorname{see}(8.51)) \\
& =\left\|w_{t+1}-w^{*}-\left(G\left(z_{t+1}\right)-G\left(z^{*}\right)\right)\right\|_{2} \quad\left(\left(w^{*}, z^{*}\right)\right. \text { is a solution of (8.1)) } \\
& \leq\left\|w_{t+1}-w^{*}\right\|_{2}+\left\|G\left(z_{t+1}\right)-G\left(z^{*}\right)\right\|_{2} \quad \quad \quad \text { (triangle inequality) } \\
& \left.\leq\left\|w_{t+1}-w^{*}\right\|_{2}+\kappa_{G}\left\|z_{t+1}-z^{*}\right\|_{2} \quad \text { (see }(8.42)\right) \\
& =\delta_{t+1}+\kappa_{G} \delta_{t+1}^{\prime}, \quad(\operatorname{see}(8.50)) \tag{8.77}
\end{array}
$$

which, after substituting in (8.76), yields that

$$
\begin{align*}
\Delta_{t+1} \leq & \Delta_{t}+\frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}+\theta\left(\bar{\omega}_{\rho}-\Delta_{t}\right)+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta}+2 \sigma_{t+1} \delta_{t+1}^{2}+2 \sigma_{t+1} \kappa_{G}^{2} \delta_{t+1}^{\prime 2} \\
& \left(\text { see (8.77) and }(a+b)^{2} \leq 2 a^{2}+2 b^{2}\right) \\
\leq & \Delta_{t}+\frac{\omega_{\rho} \delta_{t}^{2}}{2}+\frac{\omega_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}+\theta\left(\bar{\omega}_{\rho}-\Delta_{t}\right)+\frac{\theta^{2} \delta_{t}^{2}}{\alpha}+\frac{\theta^{2} \delta_{t}^{\prime 2}}{2 \beta}+2 \sigma_{0} \delta_{t+1}^{2}+2 \sigma_{0} \kappa_{G}^{2} \delta_{t+1}^{\prime 2} \\
& \quad\left(\sigma_{t+1} \leq \sigma_{0}\right. \text { in Algorithm 8.1) } \tag{8.78}
\end{align*}
$$

For every iteration $t$, suppose that

$$
\begin{equation*}
\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta} \geq \bar{\eta}_{\rho} \geq \frac{\bar{\mu}_{\rho}}{\min \left(\frac{\alpha \mu_{\rho}}{4}, \frac{\beta \mu_{\rho}^{\prime}}{2}\right)-\sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)}} \tag{8.79}
\end{equation*}
$$

for $\bar{\eta}_{\rho}$ to be set later. Consequently, it holds that

$$
\begin{align*}
\frac{\Delta_{t}}{\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta}} & \geq \frac{\frac{\mu_{\rho} \delta_{t}^{2}}{2}+\frac{\mu_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}-\bar{\mu}_{\rho}}{\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta}} \quad(\text { see (8.59)) } \\
& \geq \min \left(\frac{\alpha \mu_{\rho}}{4}, \frac{\beta \mu_{\rho}^{\prime}}{2}\right)-\frac{\bar{\mu}_{\rho}}{\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta^{\prime 2}}{\beta}} \\
& \geq \min \left(\frac{\alpha \mu_{\rho}}{4}, \frac{\beta \mu_{\rho}^{\prime}}{2}\right)-\frac{\bar{\mu}_{\rho}}{\bar{\eta}_{\rho}} \quad(\text { see (8.79)) } \\
& \geq \sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \tag{8.80}
\end{align*}
$$

We now set

$$
\begin{equation*}
\widehat{\theta}_{t}:=\min \left(\sqrt{\frac{\Delta_{t}^{2}}{\left(\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta}\right)^{2}}-\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)}, 1\right) \tag{8.81}
\end{equation*}
$$

which is well-defined, as verified in (8.80). From (8.80,8.81), it also immediately follows that

$$
\begin{gather*}
\widehat{\theta}_{t} \in[0,1], \quad \forall t,  \tag{8.82}\\
\Delta_{t} \geq 0, \quad \forall t, \tag{8.83}
\end{gather*}
$$

which we will use later on in the proof. Consider first the case where $\widehat{\theta}_{t}<1$. To study the choice of $\theta=\widehat{\theta}_{t}$ in (8.76), we will need the bound

$$
\begin{align*}
& -\widehat{\theta}_{t} \Delta_{t}+\widehat{\theta}_{t}^{2}\left(\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{2 \beta}\right) \\
& =-\sqrt{\frac{\Delta_{t}^{4}}{\left(\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta}\right)^{2}}-\Delta_{t}^{2} \max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \\
& \quad+\frac{\Delta_{t}^{2}}{\frac{4 \delta_{t}^{2}}{\alpha}+\frac{2 \delta_{t}^{\prime 2}}{\beta}}-\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)\left(\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{2 \beta}\right)  \tag{8.83}\\
& \leq-\frac{\Delta_{t}^{2}}{\frac{4 \delta_{t}^{2}}{\alpha}+\frac{2 \delta^{\prime 2}}{\beta}}+\Delta_{t} \sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \\
& \quad-\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)\left(\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{2 \beta}\right) \tag{8.84}
\end{align*}
$$

where the inequality above uses $\sqrt{a-b} \geq \sqrt{a}-\sqrt{b}$. Substituting (8.84) back into (8.78), we
reach

$$
\begin{align*}
\Delta_{t+1} \leq & \Delta_{t}-\frac{\Delta_{t}^{2}}{\frac{4 \delta_{t}^{2}}{\alpha}+\frac{2 \delta^{\prime 2}}{\beta}}+\Delta_{t} \sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \\
\leq & \Delta_{t}-\left(\min \left(\frac{\alpha \mu_{\rho}}{4}, \frac{\beta \mu_{\rho}^{\prime}}{2}\right)-\frac{\bar{\mu}_{\rho}}{\bar{\eta}_{\rho}}\right) \frac{\Delta_{t}}{2} \\
& +\Delta_{t} \sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \quad \text { (see third line of (8.80) and (8.83)) } \\
\leq & \left(1-\min \left(\frac{\alpha \mu_{\rho}}{8}, \frac{\beta \mu_{\rho}^{\prime}}{4}\right)+\frac{\bar{\mu}_{\rho}}{2 \bar{\eta}_{\rho}}+\sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)}\right) \Delta_{t} \\
= & \eta_{\rho, 1} \Delta_{t}, \quad \text { if } \Delta_{t}<\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta} \tag{8.85}
\end{align*}
$$

Next consider the case where $\widehat{\theta}_{t}=1$. With the choice of $\theta=\widehat{\theta}_{t}=1$ in (8.78), we find that

$$
\begin{align*}
\Delta_{t+1} & \leq\left(\frac{\omega_{\rho}}{2}+\frac{1}{\alpha}+\rho\right) \delta_{t}^{2}+\left(\frac{\omega_{\rho}^{\prime}}{2}+\frac{1}{2 \beta}+\rho \kappa_{G}^{2}\right) \delta_{t}^{\prime 2} \quad \text { (see (8.78)) } \\
& \leq \frac{1}{2}\left(1+\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)\right) \cdot\left(\frac{2 \delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta}\right) \\
& \leq \frac{1}{2} \sqrt{1+\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \Delta_{t} \quad \text { (see (8.81)) } \\
& =: \eta_{\rho, 2} \Delta_{t}, \quad \text { if } \Delta_{t} \geq \frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta} \tag{8.86}
\end{align*}
$$

To simplify the above expressions, let us assume that

$$
\begin{equation*}
\sqrt{\max \left(\frac{\alpha}{2}\left(\omega_{\rho}+4 \sigma_{0}\right), \beta\left(\omega_{\rho}^{\prime}+4 \sigma_{0} \kappa_{G}^{2}\right)\right)} \leq \min \left(\frac{\alpha \mu_{\rho}}{16}, \frac{\beta \mu_{\rho}^{\prime}}{8}\right) \leq \frac{1}{2} \tag{8.87}
\end{equation*}
$$

from which it follows that

$$
\begin{align*}
\max \left(\eta_{\rho, 1}, \eta_{\rho, 2}\right) & \leq 1-\min \left(\frac{\alpha \mu_{\rho}}{16}, \frac{\beta \mu_{\rho}^{\prime}}{8}\right)+\frac{\bar{\mu}_{\rho}}{2 \bar{\eta}_{\rho}} \\
& \leq 1-\min \left(\frac{\alpha \mu_{\rho}}{32}, \frac{\beta \mu_{\rho}^{\prime}}{16}\right) \\
& =: 1-\eta_{\rho} \in[0,1) \tag{8.88}
\end{align*}
$$

where the second line above holds if

$$
\begin{equation*}
\bar{\eta}_{\rho} \geq \frac{\bar{\mu}_{\rho}}{\min \left(\frac{\alpha \mu_{\rho}}{16}, \frac{\beta \mu_{\rho}^{\prime}}{8}\right)} \tag{8.89}
\end{equation*}
$$

Then, by unfolding ( $8.85,8.86$ ), we reach

$$
\begin{equation*}
\Delta_{t} \leq\left(1-\eta_{\rho}\right)^{t} \Delta_{0} \tag{8.90}
\end{equation*}
$$

Moreover, by combining (8.59,8.90), we can bound the error, namely,

$$
\begin{align*}
\frac{\delta_{t}^{2}}{\alpha}+\frac{\delta_{t}^{\prime 2}}{\beta} & \leq \max \left(\alpha \mu_{\rho}, \beta \mu_{\rho}^{\prime}\right)\left(\mu_{\rho} \delta_{t}^{2}+\mu_{\rho}^{\prime} \delta_{t}^{\prime 2}\right) \\
& \leq \mu_{\rho} \delta_{t}^{2}+\mu_{\rho}^{\prime} \delta_{t}^{\prime 2} \quad(\text { see }(8.65,8.71), \text { Lemmas 8.3 and 8.4) } \\
& \leq 2\left(\Delta_{t}+\bar{\mu}_{\rho}\right) \quad(\text { see }(8.59)) \\
& \leq 2\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2 \bar{\mu}_{\rho}}{\eta_{\rho}} \\
& \leq 2\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2 \bar{\mu}_{\rho}}{\min \left(\frac{\alpha \mu_{\rho}}{16}, \frac{\beta \mu_{\rho}^{\prime}}{8}\right)} \quad \text { (see (8.88)) } \\
& =: 2\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{\bar{\eta}_{\rho}}{\rho} \quad \quad \text { (this choice of } \bar{\eta}_{\rho} \text { satisfies (8.79,8.89)) } \tag{8.91}
\end{align*}
$$

It remains to bound the feasibility gap $\left\|A_{t}\right\|_{2}$, see (8.51). Instead of (8.77), we consider the following alternative approach to bound $\left\|A_{t}\right\|_{2}$. Using definition of $\Delta_{t}$ in (8.48), we write that

$$
\begin{align*}
\Delta_{t} & =\mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right) \quad(\operatorname{see}(8.48)) \\
& =\mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}\left(v_{t}, \lambda^{*}\right)+\mathscr{L}_{\rho}\left(v_{t}, \lambda^{*}\right)-\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right) \\
& =\left\langle A_{t}, \lambda_{t}-\lambda^{*}\right\rangle+\mathscr{L}\left(v_{t}, \lambda^{*}\right)-\mathscr{L}\left(v^{*}, \lambda^{*}\right)+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \tag{8.92}
\end{align*}
$$

where

$$
\begin{equation*}
\mathscr{L}(v, \lambda)=\mathscr{L}(w, z, \lambda):=L(w)+R(w)+H(z)+\langle w-G(z), \lambda\rangle . \tag{8.93}
\end{equation*}
$$

It is not difficult to verify that $\mathscr{L}\left(v^{*}, \lambda^{*}\right)=\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right)$ is the optimal value of problem (8.1) and that $\mathscr{L}\left(v_{t}, \lambda^{*}\right) \geq \mathscr{L}\left(v^{*}, \lambda^{*}\right)$, from which it follows that

$$
\begin{align*}
\Delta_{t} & \geq\left\langle A_{t}, \lambda_{t}-\lambda^{*}\right\rangle+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \quad(\text { see (8.92)) } \\
& \geq-\frac{\rho}{4}\left\|A_{t}\right\|_{2}^{2}-\frac{1}{\rho}\left\|\lambda_{t}-\lambda^{*}\right\|_{2}^{2}+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \quad\left(\text { Holder's inequality and } 2 a b \leq a^{2}+b^{2}\right) \\
& \geq-\frac{2}{\rho}\left\|\lambda_{t}\right\|_{2}^{2}-\frac{2}{\rho}\left\|\lambda^{*}\right\|_{2}^{2}+\frac{\rho}{4}\left\|A_{t}\right\|_{2}^{2} \quad\left((a+b)^{2} \leq 2 a^{2}+2 b^{2}\right) \\
& \geq-\frac{2 \lambda_{\max }^{2}}{\rho}-\frac{2\left\|\lambda^{*}\right\|_{2}^{2}}{\rho}+\frac{\rho}{4}\left\|A_{t}\right\|_{2}^{2} \quad \quad \text { (see (8.100)) } \tag{8.94}
\end{align*}
$$

which, in turn, implies that

$$
\begin{align*}
\left\|A_{t}\right\|_{2}^{2} & \leq \frac{4}{\rho}\left(\Delta_{t}+\frac{2 \lambda_{\text {max }}^{2}}{\rho}+\frac{2\left\|\lambda^{*}\right\|_{2}^{2}}{\rho}\right) \quad(\text { see (8.94)) } \\
& \leq \frac{4}{\rho}\left(\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{2\left(\bar{\eta}_{\rho, 1}+\bar{\eta}_{\rho, 2}\right)}{\eta_{\rho}}+\frac{2 \lambda_{\max }^{2}}{\rho}+\frac{2\left\|\lambda^{*}\right\|_{2}^{2}}{\rho}\right) \quad \text { (see (8.90)) } \\
& \leq \frac{4}{\rho}\left(\left(1-\eta_{\rho}\right)^{t} \Delta_{0}+\frac{\bar{\eta}_{\rho}+2 \lambda_{\max }^{2}+2\left\|\lambda^{*}\right\|_{2}^{2}}{\rho}\right) \quad \text { (see (8.91)) } \\
& =: \frac{4\left(1-\eta_{\rho}\right)^{t} \Delta_{0}}{\rho}+\frac{\widetilde{\eta}_{\rho}}{\rho^{2}} . \tag{8.95}
\end{align*}
$$

This completes the proof of Theorem 2.1.
Let us also inspect the special case where $\mu_{L} \gg \rho \gtrsim 1$ and $\iota_{G}^{2} \gg v_{G}$, where $\approx$ and $\gtrsim$ suppress any universal constants and dependence on the dual optimal variable $\lambda^{*}$, for the sake of simplicity. From Lemmas 8.3 and 8.4 , it is easy to verify that

$$
\begin{gather*}
v_{\rho} \approx v_{L}, \quad \xi_{\rho} \approx \rho \kappa_{G}^{2}, \quad \tau_{\rho}=\rho \kappa_{G} \\
\mu_{\rho} \approx \mu_{L}, \quad \mu_{\rho}^{\prime} \approx \rho l_{G}^{2}, \quad \bar{\mu}_{\rho} \approx \rho^{-1}, \quad \omega_{\rho}^{\prime} \approx \rho v_{G} \tag{8.96}
\end{gather*}
$$

We can then take

$$
\begin{align*}
\alpha & \approx \frac{1}{v_{L}}, \quad(\text { see (8.65)) } \\
\beta & \approx \frac{1}{\xi_{\rho}} \approx \frac{1}{\rho \kappa_{G}^{2}}, \quad(\text { see }(8.71)) \\
\eta_{\rho} & \approx \min \left(\frac{\mu_{L}}{v_{L}}, \frac{l_{G}^{2}}{\kappa_{G}^{2}}\right), \quad(\text { see }(8.88)) \\
\bar{\eta}_{\rho} & \approx \frac{\rho \bar{\mu}_{\rho}}{\min \left(\alpha \mu_{\rho}, \beta \mu_{\rho}^{\prime}\right)} \approx \max \left(\frac{v_{L}}{\mu_{L}}, \frac{\kappa_{G}^{2}}{l_{G}^{2}}\right), \quad(\text { see (8.91)) } \\
\widetilde{\eta}_{\rho} & \approx \bar{\eta}_{\rho} \approx \max \left(\frac{v_{L}}{\mu_{L}}, \frac{\kappa_{G}^{2}}{l_{G}^{2}}\right) . \quad(\text { see }(8.95)) \tag{8.97}
\end{align*}
$$

Lastly, for (8.87) to hold, it suffices that

$$
\begin{equation*}
\sigma_{0} \lesssim \rho \min \left(\frac{\mu_{L}^{2}}{v_{L}^{2}}, \frac{\iota_{G}^{4}}{\kappa_{G}^{4}}\right)=: \sigma_{0, \rho} \tag{8.98}
\end{equation*}
$$

### 8.9 Appendix: Proof of Lemma 8.3

To prove (8.53), we write that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t+1}, z_{t+1}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\left\langle w_{t+1}-w_{t}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)\right\rangle \\
& =L\left(w_{t+1}\right)-L\left(w_{t}\right)-\left\langle w_{t+1}-w_{t}, \nabla_{w} L\left(w_{t}\right)\right\rangle \\
& \quad \quad+\frac{\rho}{2}\left\|w_{t+1}-G\left(z_{t+1}\right)\right\|_{2}^{2}-\frac{\rho}{2}\left\|w_{t}-G\left(z_{t+1}\right)\right\|_{2}^{2}-2 \rho\left\langle w_{t+1}-w_{t}, w_{t}-G\left(z_{t+1}\right)\right\rangle \\
& \leq \\
& =\frac{v_{L}}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2}+\bar{v}_{L}+\frac{\rho}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2} \quad \text { (see (8.40)) }  \tag{8.99}\\
& =: \frac{v_{\rho}}{2}\left\|w_{t+1}-w_{t}\right\|_{2}^{2}+\bar{v}_{\rho} .
\end{align*}
$$

To prove (8.54), let us first control the dual sequence $\left\{\lambda_{t}\right\}_{t}$ by writing that

$$
\begin{align*}
\left\|\lambda_{t}\right\|_{2} & =\left\|\lambda_{0}+\sum_{i=1}^{t} \sigma_{i} A_{i}\right\|_{2} \quad(\text { see (8.52)) } \\
& \leq\left\|\lambda_{0}\right\|_{2}+\sum_{i=1}^{t} \sigma_{i}\left\|A_{i}\right\|_{2} \quad \text { (triangle inequality) } \\
& \leq\left\|\lambda_{0}\right\|_{2}+\sum_{t^{\prime}=1}^{t} \frac{\sigma_{0}}{i \log ^{2}(i+1)} \\
& \leq\left\|\lambda_{0}\right\|_{2}+c \sigma_{0} \\
& =: \lambda_{\max } \tag{8.100}
\end{align*}
$$

where

$$
\begin{equation*}
c \geq \sum_{t=1}^{\infty} \frac{1}{t \log ^{2}(t+1)} \tag{8.101}
\end{equation*}
$$

We now write that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)-\left\langle z_{t+1}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right. \\
&=-\left\langle G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), \lambda_{t}\right\rangle \\
&+\frac{\rho}{2}\left\|w_{t}-G\left(z_{t+1}\right)\right\|_{2}^{2}-\frac{\rho}{2}\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2} \\
& \quad+\rho\left\langle D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), w_{t}-G\left(z_{t}\right)\right\rangle . \quad \text { (see (8.45)) } \tag{8.102}
\end{align*}
$$

To bound the first inner product on the right-hand side above, we write that

$$
\begin{align*}
& \left\langle G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), \lambda_{t}\right\rangle \\
& \leq\left\|G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right)\right\|_{2} \cdot\left\|\lambda_{t}\right\|_{2} \quad \text { (Cauchy-Shwartz's inequality) } \\
& \leq \frac{v_{G} \lambda_{\max }}{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \quad(\text { see }(8.41,8.100)) \tag{8.103}
\end{align*}
$$

The remaining component on the right-hand side of (8.102) can be bounded as

$$
\begin{align*}
& \left\|w_{t}-G\left(z_{t+1}\right)\right\|_{2}^{2}-\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2}+2\left\langle D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), w_{t}-G\left(z_{t}\right)\right\rangle \\
& =\left\|w_{t}-G\left(z_{t+1}\right)\right\|_{2}^{2}-\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2}+2\left\langle G\left(z_{t+1}\right)-G\left(z_{t}\right), w_{t}-G\left(z_{t}\right)\right\rangle \\
& \quad-2\left\langle G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), w_{t}-G\left(z_{t}\right)\right\rangle \\
& =\left\|G\left(z_{t+1}\right)-G\left(z_{t}\right)\right\|_{2}^{2} \\
& \quad+2\left\langle G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right), w_{t}-G\left(z_{t}\right)\right\rangle \\
& \leq\left\|G\left(z_{t+1}\right)-G\left(z_{t}\right)\right\|_{2}^{2} \\
& \quad+2\left\|G\left(z_{t+1}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z_{t+1}-z_{t}\right)\right\|_{2} \cdot\left\|w_{t}-G\left(z_{t}\right)\right\|_{2} \quad \text { (Cauchy-Shwartz's inequality) } \\
& \leq \kappa_{G}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}+v_{G}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}\left\|w_{t}-G\left(z_{t}\right)\right\|_{2} \quad \text { (see (8.41,8.42)) } \\
& \leq \kappa_{G}^{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}+v_{G}\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \max _{i}\left\|A_{i}\right\|_{2} . \quad \text { (see (8.51)) } \tag{8.104}
\end{align*}
$$

Substituting the bounds in $(8.103,8.104)$ back into $(8.102)$, we find that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)-\left\langle z_{t+1}-z_{t}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right. \\
& \leq \frac{1}{2}\left(v_{G}\left(\lambda_{\max }+\rho \max _{i}\left\|A_{i}\right\|_{2}\right)+\rho \kappa_{G}^{2}\right)\left\|z_{t+1}-z_{t}\right\|_{2}^{2} \\
& =: \frac{\xi_{\rho}}{2}\left\|z_{t+1}-z_{t}\right\|_{2}^{2}+\bar{\xi}_{\rho} \tag{8.105}
\end{align*}
$$

which proves (8.54). To prove (8.55), we write that

$$
\begin{align*}
& \left\|\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t+1}, \lambda_{t}\right)-\nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(w_{t}, z_{t}, \lambda_{t}\right)\right\|_{2} \\
& =\rho\left\|G\left(z_{t+1}\right)-G\left(z_{t}\right)\right\|_{2} \quad(\text { see }(8.45)) \\
& \leq \rho \kappa_{G}\left\|z_{t+1}-z_{t}\right\|_{2} \quad(\text { see }(8.42)) \\
& =: \tau_{\rho}\left\|z_{t+1}-z_{t}\right\|_{2}+\bar{\tau}_{\rho} . \tag{8.106}
\end{align*}
$$

This completes the proof of Lemma 8.3.

### 8.10 Appendix: Proof of Lemma 8.4

For future reference, we record that

$$
\begin{aligned}
& \left\langle v_{t}-v^{*}, \nabla_{\nu} \mathscr{L}_{\rho}^{\prime}\left(v^{*}\right)\right\rangle \\
& =\left\langle w_{t}-w^{*}, \nabla_{w} \mathscr{L}_{\rho}^{\prime}\left(v^{*}\right)\right\rangle+\left\langle z_{t}-z^{*}, \nabla_{z} \mathscr{L}_{\rho}^{\prime}\left(v^{*}\right)\right\rangle \quad(v=(w, z)) \\
& =\left\langle w_{t}-w^{*}, \nabla L\left(w^{*}\right)+\lambda^{*}+\rho\left(w^{*}-G\left(z^{*}\right)\right\rangle-\left\langle D G\left(z^{*}\right)\left(z_{t}-z^{*}\right), \lambda^{*}+\rho\left(w^{*}-G\left(z^{*}\right)\right)\right\rangle\right. \\
& =\left\langle w_{t}-w^{*}, \nabla L\left(w^{*}\right)+\lambda^{*}\right\rangle-\left\langle D G\left(z^{*}\right)\left(z_{t}-z^{*}\right), \lambda^{*}\right\rangle,
\end{aligned}
$$

where the last line above uses the feasibility of $v^{*}$ in (8.1). To prove (8.59), we use the definition of $\mathscr{L}_{\rho}$ in (8.44) to write that

$$
\begin{align*}
& \mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right) \\
& =\mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)+R\left(w_{t}\right)-R\left(w^{*}\right)+L\left(z_{t}\right)-L\left(z^{*}\right) \quad \text { (see (8.44,8.45)) } \\
& \geq \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)-\left\langle v_{t}-v^{*}, \nabla_{v} \mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)\right\rangle \quad \text { (see (8.47)) } \\
& =L\left(w_{t}\right)-L\left(w^{*}\right)-\left\langle w_{t}-w^{*}, \nabla L\left(u^{*}\right)\right\rangle \\
& \quad+\left\langle A_{t}, \lambda_{t}\right\rangle-\left\langle w_{t}-w^{*}-D G\left(z^{*}\right)\left(z_{t}-z^{*}\right), \lambda^{*}\right\rangle+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \quad \text { (see (8.107)) } \\
& \geq \\
& \geq \frac{\mu_{L} \delta_{t}^{2}}{2}+\left\langle A_{t}, \lambda_{t}-\lambda^{*}\right\rangle+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \\
& \quad+\left\langle G\left(z_{t}\right)-G\left(z^{*}\right)-D G\left(z^{*}\right)\left(z_{t}-z_{k}^{*}\right), \lambda^{*}\right\rangle \quad \text { (see (8.39,8.50)) }  \tag{8.108}\\
& \geq \\
& \frac{\mu_{L} \delta_{t}^{2}}{2}+\left\langle A_{t}, \lambda_{t}-\lambda^{*}\right\rangle+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2}-\frac{v_{G} \delta_{t}^{2}}{2}\left\|\lambda^{*}\right\|_{2} \quad \quad \text { (see (8.41,8.50)) }
\end{align*}
$$

To control the terms involving $A_{t}$ in the last line above, we write that

$$
\begin{align*}
& \left\langle A_{t}, \lambda_{t}-\lambda^{*}\right\rangle+\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2} \\
& =\frac{\rho}{2}\left\|A_{t}-\frac{\lambda_{t}-\lambda^{*}}{\rho}\right\|_{2}^{2}-\frac{\left\|\lambda_{t}-\lambda^{*}\right\|_{2}^{2}}{2 \rho} \\
& =\frac{\rho}{2}\left\|w_{t}-w^{*}-\left(G\left(z_{t}\right)-G\left(z^{*}\right)\right)-\frac{\lambda_{t}-\lambda^{*}}{\rho}\right\|_{2}^{2}-\frac{\left\|\lambda_{t}-\lambda^{*}\right\|_{2}^{2}}{2 \rho} \quad(\text { see }(8.47,8.51)) \\
& \geq \frac{\rho}{4}\left\|G\left(z_{t}\right)-G\left(z^{*}\right)\right\|_{2}^{2}-\rho \delta_{t}^{2}-\frac{3\left\|\lambda_{t}-\lambda^{*}\right\|_{2}^{2}}{2 \rho} \quad\left(\|a-b-c\|_{2}^{2} \geq \frac{\|a\|_{2}^{2}}{2}-2\|b\|_{2}^{2}-2\|c\|_{2}^{2}\right) \\
& \geq \frac{\rho l_{G}^{2} \delta_{t}^{\prime 2}}{4}-\rho \delta_{t}^{2}-\frac{3\left\|\lambda_{t}-\lambda^{*}\right\|_{2}^{2}}{2 \rho} \quad(\text { see }(8.50,8.42)) \\
& \geq \frac{\rho \iota_{G}^{2} \delta_{t}^{\prime 2}}{4}-\rho \delta_{t}^{2}-\frac{3}{\rho}\left(\lambda_{\max }^{2}+\left\|\lambda^{*}\right\|_{2}^{2}\right), \quad\left((a+b)^{2} \leq 2 a^{2}+2 b^{2}\right. \text { and (8.100)) } \tag{8.109}
\end{align*}
$$

which, after substituting in (8.108), yields that

$$
\begin{align*}
& \mathscr{L}_{\rho}\left(v_{t}, \lambda_{t}\right)-\mathscr{L}_{\rho}\left(v^{*}, \lambda^{*}\right) \\
& \geq \frac{\mu_{L}-2 \rho}{2} \delta_{t}^{2}+\frac{1}{2}\left(\frac{\rho \iota_{G}^{2}}{2}-v_{G}\left\|\lambda^{*}\right\|_{2}\right) \delta_{t}^{\prime 2}-\frac{3}{\rho}\left(\lambda_{\max }^{2}+\left\|\lambda^{*}\right\|_{2}^{2}\right) \\
& \geq \frac{\mu_{\rho} \delta_{t}^{2}}{2}+\frac{\mu_{\rho}^{\prime} \delta_{t}^{\prime 2}}{2}-\bar{\mu}_{\rho} \tag{8.110}
\end{align*}
$$

where

$$
\begin{equation*}
\mu_{\rho}:=\mu_{L}-2 \rho, \quad \mu_{\rho}^{\prime}:=\frac{\rho l_{G}^{2}}{2}-v_{G}\left\|\lambda^{*}\right\|_{2} \tag{8.111}
\end{equation*}
$$

$$
\begin{equation*}
\bar{\mu}_{\rho}:=\frac{3}{\rho}\left(\lambda_{\max }^{2}+\left\|\lambda^{*}\right\|_{2}^{2}\right) \tag{8.112}
\end{equation*}
$$

This proves (8.59). To prove (8.60), we use the definition of $\mathscr{L}_{\rho}^{\prime}$ in (8.45) to write that

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)-\mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)-\left\langle v^{*}-v_{t}, \nabla_{v} \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)\right\rangle \\
& =L\left(w^{*}\right)-L\left(w_{t}\right)-\left\langle w^{*}-w_{t}, \nabla L\left(w_{t}\right)\right\rangle \\
& \quad-\left\langle A_{t}+D A\left(v_{t}\right)\left(v^{*}-v_{t}\right), \lambda_{t}\right\rangle \\
& \quad-\frac{\rho}{2}\left\langle A_{t}+2 D A\left(v_{t}\right)\left(v^{*}-v_{t}\right), A_{t}\right\rangle, \quad \text { (see (8.45)) } \tag{8.113}
\end{align*}
$$

where

$$
D A(\nu)=\left[\begin{array}{ll}
I_{d} & -D G(z) \tag{8.114}
\end{array}\right]
$$

is the Jacobian of the map $A$. The second inner product on the right-hand side of (8.113) can be bounded as

$$
\begin{align*}
& -\left\langle A_{t}+D A\left(v_{t}\right)\left(v^{*}-v_{t}\right), \lambda_{t}\right\rangle \\
& =-\left\langle w_{t}-G\left(z_{t}\right)+\left(w^{*}-w_{t}\right)-D G\left(z_{t}\right)\left(z^{*}-z_{t}\right), \lambda_{t}\right\rangle \quad(\text { see }(8.51,8.114)) \\
& =-\left\langle G\left(z^{*}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z^{*}-z_{t}\right), \lambda_{t}\right\rangle \quad\left(w^{*}=G\left(z^{*}\right)\right) \\
& \geq-\frac{v_{G} \delta_{t}^{\prime 2}}{2}\left\|\lambda_{t}\right\|_{2} \quad(\text { see }(8.41,8.50)) \\
& \geq-\frac{v_{G} \delta_{t}^{\prime 2}}{2} \lambda_{\max } . \quad(\text { see }(8.100)) \tag{8.115}
\end{align*}
$$

To control the last inner product on the right-hand side of (8.113), we write that

$$
\begin{align*}
& -\frac{\rho}{2}\left\langle A_{t}+2 D A\left(v_{t}\right)\left(v^{*}-v_{t}\right), A_{t}\right\rangle \\
& =\frac{\rho}{2}\left\|A_{t}\right\|_{2}^{2}-\rho\left\langle A_{t}+D A\left(v_{t}\right)\left(v^{*}-v_{t}\right), A_{t}\right\rangle \\
& \geq-\rho\left\|A_{t}+D A\left(v_{t}\right)\left(v^{*}-v_{t}\right)\right\|_{2}\left\|A_{t}\right\|_{2} \quad \text { (Holder's inequality) } \\
& \left.=-\rho\left\|\left(w^{*}-G\left(z^{*}\right)\right)-\left(w_{t}-G\left(z_{t}\right)\right)-\left(w^{*}-w_{t}\right)+D G\left(z_{t}\right)\left(z^{*}-z_{t}\right)\right\|_{2} \quad \text { (see (8.51,8.114) and } w^{*}=G\left(z^{*}\right)\right) \\
& =-\rho\left\|G\left(z^{*}\right)-G\left(z_{t}\right)-D G\left(z_{t}\right)\left(z^{*}-z_{t}\right)\right\|_{2} \\
& \geq-\frac{\rho v_{G}}{2}\left\|z^{*}-z_{t}\right\|_{2}^{2} \quad(\text { see }(8.41)) \\
& =-\frac{\rho v_{G} \delta_{t}^{\prime 2}}{2} . \quad(\text { see }(8.50)) \tag{8.116}
\end{align*}
$$

By substituting the bounds in $(8.115,8.116)$ back into (8.113) and also using the convexity of $L$, we reach

$$
\begin{align*}
& \mathscr{L}_{\rho}^{\prime}\left(v^{*}, \lambda^{*}\right)-\mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)-\left\langle v^{*}-v_{t}, \nabla_{v} \mathscr{L}_{\rho}^{\prime}\left(v_{t}, \lambda_{t}\right)\right\rangle \\
& \geq-\frac{v_{G}}{2}\left(\lambda_{\max }+\rho\right) \delta_{t}^{\prime 2} . \tag{8.117}
\end{align*}
$$

This proves (8.60), thus completing the proof of Lemma 8.4.

### 8.11 Appendix: Relation with Gradient Descent

Throughout this section, we set $R \equiv 0$ and $H \equiv 0$ in problem (8.1) and consider the updates in algorithm 8.2, namely,

$$
\begin{align*}
& z_{t+1}=z_{t}-\beta \nabla_{z} \mathscr{L}_{\rho}\left(w_{t}, z_{t}, \lambda_{t}\right) \\
& w_{t+1} \in \underset{w}{\operatorname{argmin}} \mathscr{L}_{\rho}\left(w, z_{t+1}, \lambda_{t}\right)  \tag{8.118}\\
& \lambda_{t+1}=\lambda_{t}+\sigma_{t+1}\left(w_{t+1}-G\left(z_{t+1}\right)\right)
\end{align*}
$$

From (8.2), recall that $\mathscr{L}_{\rho}(w, z, \lambda)$ is convex in $w$ and the second step in (8.118) is therefore often easy to implement with any over-the-shelf standard convex solver. Recalling (8.2), note also that the optimality condition for $w_{t+1}$ in (8.118) is

$$
\begin{equation*}
w_{t+1}-G\left(z_{t}\right)=-\frac{1}{\rho}\left(\nabla L_{m}\left(w_{t+1}\right)+\lambda_{t}\right) \tag{8.119}
\end{equation*}
$$

Using (8.2) again, we also write that

$$
\begin{align*}
& \nabla_{z} \mathscr{L}_{\rho}\left(w_{t+1}, z_{t}, \lambda_{t}\right) \\
& =-D G\left(z_{t}\right)^{\top}\left(\lambda_{t}+\rho\left(w_{t+1}-G\left(z_{t}\right)\right)\right. \\
& =-D G\left(z_{t}\right)^{\top}\left(\lambda_{t}-\lambda_{t-1}-\nabla L_{m}\left(w_{t}\right)\right) \\
& =-D G\left(z_{t}\right)^{\top}\left(\sigma_{t}\left(w_{t}-G\left(z_{t}\right)\right)-\nabla L\left(w_{t}\right)\right) \tag{8.120}
\end{align*}
$$

where the last two lines above follow from (8.119,8.118), respectively. Substituting back into the $z$ update in (8.118), we reach

$$
\begin{equation*}
z_{t+1}=z_{t}+\beta \sigma_{t} D G\left(z_{t}\right)^{\top}\left(w_{t}-G\left(z_{t}\right)\right)-\beta \nabla L\left(w_{t}\right) \quad(\text { see }(8.118,8.120)) \tag{8.121}
\end{equation*}
$$

from which it follows that

$$
\begin{align*}
& \left\|z_{t+1}-\left(z_{t}-\beta \nabla L\left(G\left(z_{t}\right)\right)\right)\right\|_{2} \\
& \leq \beta \sigma_{t}\left\|D G\left(z_{t}\right)^{\top}\left(w_{t}-G\left(z_{t}\right)\right)\right\|_{2}+\beta\left\|\nabla L\left(w_{t}\right)-\nabla L\left(G\left(z_{t}\right)\right)\right\|_{2} \quad \text { (see (8.121)) } \\
& \leq \beta\left(\sigma_{t} \kappa_{G}+v_{L}\right)\left\|w_{t}-G\left(z_{t}\right)\right\|_{2} . \quad \text { (see Assumptions 8.1 and 8.3) } \tag{8.122}
\end{align*}
$$

That is, as the feasibility gap vanishes in (8.24) in Theorem 2.1, the updates of Algorithm 2 match those of GD.

### 8.12 Appendix: Proof of Lemma 8.1

Recall that $R=1_{W}$ and $H \equiv 0$ for this proof. Using the optimality of $w^{*} \in \operatorname{relint}(W)$ in (8.17), we can write that

$$
\begin{align*}
\left\|\nabla L\left(w^{*}\right)\right\|_{2} & \leq\left\|\nabla L_{m}\left(w^{*}\right)\right\|_{2}+\left\|\nabla L_{m}\left(w^{*}\right)-\nabla L\left(w^{*}\right)\right\|_{2} \quad \text { (triangle inequality) } \\
& =\left\|\nabla L_{m}\left(w^{*}\right)-\nabla L\left(w^{*}\right)\right\|_{2} \quad\left(\nabla L_{m}\left(w^{*}\right)=0\right) \\
& \leq \max _{w \in W}\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2} \tag{8.123}
\end{align*}
$$

On the other hand, using the strong convexity of $L$ in (8.20), we can write that

$$
\begin{align*}
\left\|w^{\natural}-w^{*}\right\|_{2} & \leq \frac{1}{\zeta_{L}}\left\|\nabla L\left(w^{\natural}\right)-\nabla L\left(w^{*}\right)\right\|_{2} \quad \text { (see (8.20)) } \\
& =\frac{1}{\zeta_{L}}\left\|\nabla L\left(w^{*}\right)\right\| \quad\left(\nabla L\left(w^{\natural}\right)=0\right) \\
& \leq \frac{1}{\zeta_{L}} \max _{w \in W}\left\|\nabla L_{m}(w)-\nabla L(w)\right\|_{2}, \quad \text { (see (8.123)) } \tag{8.124}
\end{align*}
$$

which completes the proof of Lemma 8.1.

### 8.13 Appendix: Experimental Setup Details

### 8.13.1 Per-Iteration Computational Complexity

The gradient of the function

$$
\begin{equation*}
h(z)=\frac{1}{2}\|A G(z)-b\|_{2}^{2} \tag{8.125}
\end{equation*}
$$

follows the formula

$$
\begin{equation*}
\nabla h(z)=\nabla G(z) A^{\top}(A G(z)-b) \tag{8.126}
\end{equation*}
$$

which involves one forward pass through the network $G$, in order to compute $G(z)$, as well as one backward pass to compute $\nabla G(z)$, and finally matrix-vector products to compute the final result.

On the other hand our ADMM first computes the iterate $z_{t+1}$ with gradient descent on the augmented lagrangian (8.2) as

$$
\begin{equation*}
z_{t+1}=z_{t}-\beta \nabla_{z} \mathscr{L}_{\rho}\left(w_{t}, z_{t}, \lambda_{t}\right)=-\nabla G\left(z_{t}\right) \lambda_{t}^{\top}-\rho \nabla G\left(z_{t}\right)\left(w_{t}-G\left(z_{t}\right)\right)^{\top} \tag{8.127}
\end{equation*}
$$

which involves one forward and one backward pass on the network $G$, as well as matrixvector products. Then we perform the exact minimization procedure on the $w$ variable, which requires recomputing $G(z)$ on the new iterate $z_{t+1}$, involving one forward pass through the network, as well as the matrix-vector operations as described before. Recomputing the quantity $w_{t+1}-G\left(z_{t+1}\right)$ is immediate upon which the dual stepsize $\sigma_{t+1}$ can be computed at
negligible cost. Finally the dual variable update reads as

$$
\begin{equation*}
\lambda_{t+1}=\lambda_{t}+\sigma\left(w_{t+1}-G\left(z_{t+1}\right)\right) \tag{8.128}
\end{equation*}
$$

which involves only scalar products and vector additions of values already computed. All in all each GD iteration involves one forward and one backward pass, while ADMM computes two forward and one backward pass. Both algorithms require a few additional matrix-vector operations of similar complexity. For networks with multiple large layers, as usually encountered in practice, the complexity per iteration can then be estimated as the number of forward and backward passes, which are of similar complexity.

### 8.13.2 Parameter Tuning

We run a grid search for the gradient descent (GD) algorithm In order to do so we fix a number of iterations and compare the average objective function over a batch of 100 random images and choose the best performing parameters. We repeat the tuning in all possible escenarios in the experiments. The results figures $8.4-8.5$ (GD, Compressive sensing setup).


Figure 8.4: Performance of GD on the compressive sensing task for different step sizes. MNIST dataset. 156 (top) and 313 (bottom) linear measurements.


Figure 8.5: Performance of GD on the compressive sensing task for different step sizes. CelebA dataset. 2457 (top) and 4915 (bottom) linear measurements.

### 8.13.3 Fast Exact Augmented Lagrangian Minimization with Respect to Primal Variable $w$

In the compressive sensing setup, the augmented lagrangian takes the form

$$
\begin{equation*}
\mathscr{L}_{\rho}(w, z, \lambda):=\frac{1}{2}\|A w-b\|_{2}^{2}+\langle\lambda, w-G(z)\rangle+\frac{\rho}{2}\|w-G(z)\|_{2}^{2} \tag{8.129}
\end{equation*}
$$

with respect to $w$, this is a strongly convex function which admits a unique minimizer given by the first order optimality condition

$$
\begin{equation*}
\nabla_{w} \mathscr{L}_{\rho}(w, z, \lambda)=A^{\top}(A w-b)+\lambda+\rho(w-G(z))=0 \tag{8.130}
\end{equation*}
$$

with solution

$$
\begin{equation*}
w^{*}=\left(A^{\top} A+\rho I\right)^{-1}\left(-\lambda+G(z)+A^{\top} b\right) \tag{8.131}
\end{equation*}
$$

Given the SVD of $A=U S V^{\top}$ we have $A^{\top} A=V D V^{\top}$, where $D$ corresponds to the diagonal matrix with the eigenvalues of $A^{T} A$. We then have that $A^{\top} A+\rho I=V(D+\rho I) V^{\top}$ so that

$$
\begin{equation*}
w^{*}=V(D+\rho I) V^{\top}\left(-\lambda+G(z)+A^{\top} b\right) \tag{8.132}
\end{equation*}
$$

which involves only a fixed number of matrix-vector products per-iteration.

### 8.13.4 Per-Iteration Computational Complexity

The gradient of the function

$$
\begin{equation*}
h(z)=\frac{1}{2}\|A G(z)-b\|_{2}^{2} \tag{8.133}
\end{equation*}
$$

follows the formula

$$
\begin{equation*}
\nabla h(z)=\nabla G(z) A^{\top}(A G(z)-b) \tag{8.134}
\end{equation*}
$$

which involves one forward pass through the network $G$, in order to compute $G(z)$, as well as one backward pass to compute $\nabla G(z)$, and finally matrix-vector products to compute the final result.

On the other hand our ADMM first computes the iterate $z_{t+1}$ with gradient descent on the augmented lagrangian (8.129)

$$
\begin{equation*}
z_{t+1}=z_{t}-\beta \nabla_{z} \mathscr{L}_{\rho}\left(w_{t}, z_{t}, \lambda_{t}\right)=-\nabla G\left(z_{t}\right) \lambda_{t}^{\top}-\rho \nabla G\left(z_{t}\right)\left(w_{t}-G\left(z_{t}\right)\right)^{\top} \tag{8.135}
\end{equation*}
$$

which involves one forward and one backward pass on the network $G$, as well as matrixvector products. Then we perform the exact minimization procedure on the $w$ variable, as described in 8.13.3, which requires recomputing $G(z)$ on the new iterate $z_{t+1}$, involving one forward pass through the network, as well as the matrix-vector operations as described before. Recomputing the quantity $w_{t+1}-G\left(z_{t+1}\right)$ is immediate upon which the dual stepsize $\sigma_{t+1}$ can
be computed at negligible cost. Finally the dual variable update reads as

$$
\begin{equation*}
\lambda_{t+1}=\lambda_{t}+\sigma\left(w_{t+1}-G\left(z_{t+1}\right)\right) \tag{8.136}
\end{equation*}
$$

which involves only scalar products and vector additions of values already computed. All in all each GD iteration involves one forward and one backward pass, while ADMM computes two forward and one backward pass. Both algorithms require a few additional matrix-vector operations of similar complexity. For networks with multiple large layers, as usually encountered in practice, the complexity per iteration can then be estimated as the number of forward and backward passes, which are of similar complexity.

### 8.14 Appendix: Pseudocode for EADMM

```
Algorithm 8.2 Multi-scale Linearized ADMM with Exact Minimization (EADMM)
Input: Differentiable \(L\), proximal-friendly convex regularizers \(R\) and \(H\), differentiable prior \(G\),
penalty weight \(\rho>0\), primal step sizes \(\alpha, \beta>0\), initial dual step size \(\sigma_{0}>0\), primal initialization
\(w_{0}\) and \(z_{0}\), dual initialization \(\lambda_{0}\), stopping threshold \(\tau_{c}>0\), iterations parameter \(n\).
    \(z_{0,0} \leftarrow z_{0}, w_{0,0} \leftarrow w_{0}\)
    for \(\mathrm{k}=0, \ldots, \mathrm{~K}\) do
    \(\rho_{k} \leftarrow \rho 2^{k}, \alpha_{k} \leftarrow \alpha 2^{-k}, \beta_{k} \leftarrow \beta 2^{-k}\)
    \(z_{0} \leftarrow z_{0, k}, w_{0} \leftarrow w_{0, k}\)
    for \(t=0,1, \ldots, 2^{k} n\) do
            \(z_{t+1} \leftarrow \mathrm{P}_{\beta_{k} H}\left(z_{t}-\beta_{k} \nabla_{z} \mathscr{L}_{\rho_{k}}\left(w_{t}, z_{t}, \lambda_{t}\right)\right) \quad \triangleright\) primal updates
            \(w_{t+1} \in \underset{w}{\operatorname{argmin}} \mathscr{L}_{\rho}\left(w, z_{t+1}, \lambda_{t}\right)\)
            \(\sigma_{t+1} \leftarrow \min \left(\sigma_{0}, \frac{\sigma_{0}}{\left\|w_{t+1}-G\left(z_{t+1}\right)\right\|_{2} t \log ^{2}(t+1)}\right) \quad \triangleright\) dual step size
            \(\lambda_{t+1} \leftarrow \lambda_{t}+\sigma_{t+1}\left(w_{t+1}-G\left(z_{t+1}\right)\right) \quad \triangleright\) dual update
            \(s \leftarrow \frac{\left\|z_{t+1}-z_{t}\right\|_{2}^{2}}{\alpha_{k}}+\frac{\left\|w_{t+1}-w_{t}\right\|_{2}^{2}}{\beta_{k}}+\sigma_{t}\left\|w_{t}-G\left(z_{t}\right)\right\|_{2}^{2} \leq \tau_{c} \quad \triangleright\) stopping criterion
            if \(s \leq \tau_{c}\) then return \(\left(w_{t+1}, z_{t+1}\right)\)
            end if
            \(\left(w_{0, k+1}, z_{0, k+1}\right) \leftarrow\left(w_{t+1}, z_{t+1}\right)\)
    end for
end for
return \(\left(w_{0, K+1}, z_{0, K+1}\right)\)
```


### 8.15 Bibliographic Note

The candidate derived one of the main theoretical results (Proposition 8.1) which shows that indeed the proposed algorithm can be applied to Deep Neural Networks. A. Eftekhari contributed with Theorem 1. the candidate coded all the experiments and obtained all the numerical evidence.

# 9 The Effect of the Intrinsic Dimension on the Generalization of Quadratic Classifiers 

Fabian Latorre, Leello Dadi, Paul Rolland and Volkan Cevher. Advances in Neural Information Processing Systems 34 (NeurIPS) 2021.


#### Abstract

It has been recently observed that neural networks, unlike kernel methods, enjoy a reduced sample complexity when the distribution is isotropic (i.e., when the covariance matrix is the identity). We find that this sensitivity to the data distribution is not exclusive to neural networks, and the same phenomenon can be observed on the class of quadratic classifiers (i.e., the sign of a quadratic polynomial) with a nuclear-norm constraint. We demonstrate this by deriving an upper bound on the Rademacher Complexity that depends on two key quantities: (i) the intrinsic dimension, which is a measure of isotropy, and (ii) the largest eigenvalue of the second moment (covariance) matrix of the distribution. Our result improves the dependence on the dimension over the best previously known bound and precisely quantifies the relation between the sample complexity and the level of isotropy of the distribution.


### 9.1 Introduction

We revisit the problem of supervised classification using quadratic features of the data. We do so to highlight the influence of properties of data distrbution on the generalization error. Most of the existing results on this error only use a bound on the support of the distribution. By leveraging results from matrix concentration, we show an improved bound that uses more refined properties of the data distribution, like the second moment matrix.

The use of the second moment matrix in the error bound shows that the intrinsic dimension of the data distribution plays an important role. This is of particular interest because it is widely believed that real-world data distributions have nice properties that allow classifiers, namely neural networks, to avoid the worst-case sample complexities predicted by generalization bounds (Jiang et al., 2020).

Indeed, assumptions like the manifold hypothesis, which state that the data lies on lower dimensional embedded manifold, are often made to explain the practical success of some
generative methods. A recent paper by Pope et al. (2021) computes estimates of this true dimensionality of common machine learning datasets and shows that they are much lower than the ambient dimension of the pixel space $[0,1]^{d}$. It is therefore important that properties of the data distribution, going beyond simple bounds on the support, intervene in the study of generalization.

This influence of intrinsic dimension on generalization has been recently observed in the context of differentiating neural networks and from their kernel approximations, like the neural tangent kernel Jacot et al. (2018) or random feature models Yehudai and Shamir (2019). In particular, Ghorbani et al. (2020) observe that neural networks seem to require fewer samples than kernel methods to learn when the data distribution is isotropic.

We show that a similar phenomenon occurs in the simpler setting of quadratic classifiers, which leads to a better understanding of the causes. An improvement in sample complexity on isotropic data distributions can be proved when comparing nuclear-norm constrained quadratic classifiers and the corresponding kernel method (Frobenius norm constrained classifiers).

The study of quadratic classifiers can serve as an important first step in understanding how neural networks take advantage of the intrinsic dimension to learn with fewer samples Du and Lee (2018); Bai and Lee (2020). The nuclear-norm constraint is a natural one to study in this context. Indeed, when applying weight-decay (or $\ell_{2}$-regularization) on a single-hidden layer neural network with quadratic activations, the regularization is in effect encouraging a low nuclear norm of the coefficient matrix of the quadratic polynomial.

A better understanding of quadratics is also a worthwhile goal in its own right: complex architectures like those in Jayakumar et al. (2020) use quadratics as building blocks, attention layers Vaswani et al. (2017b), which have seen great success in language processing tasks, are multiplicative interactions.

For these reasons, we present theoretical and practical developments of nuclear-norm regularization for quadratic classification. We summarize our contributions as follows

Rademacher complexity bounds. We present a new bound on the Rademacher complexity of quadratic classifiers with a nuclear norm constraint c.f. Theorem 9.1. It improves upon the previously known bound, implied by the results by Kakade et al. (2012), by up to a square-root factor of the dimension, depending on the distribution of the data c.f. 9.4.

As a consequence of our bound, we draw attention to a clear difference between the complexity of nuclear-norm constrained and Frobenius-norm constrained quadratic classifiers. When the input data distribution is nearly-isotropic, the former enjoys a reduced dependency on the dimension. In contrast, the complexity of Frobenius-norm constrained classifiers has the same dependency on the dimension, independently of how isotropic the input data distribution is (9.2).

This observation motivates the use of data whitening pre-processing steps, which are commonly used in practice: such transformation might bring the second-order moment (covariance) matrix of the distribution close to the identity matrix and thus to nearly-isotropicity.

Computable generalization bounds. The refined Rademacher complexity bound that we obtain depends on the often unknown second-order moment of the distribution, rather than simple bounds on the diameter of the support as in (Kakade et al., 2012). Even though useful in theory, it is desirable in practice to obtain bounds that can be computed from a sample. We overcome this difficulty in Theorem 9.3, where we provide high-probability computable generalization error bounds for nuclear-norm constrained quadratic classifiers.

Experiments. We illustrate our theoretical results on synthetic data. We show how the isotropy of the input distribution plays a major role in the generalization properties of quadratic classifiers. As the dimension increases and the sample size remains proportional to it, we observe a constant generalization gap for the nuclear-norm constrained classifier. In contrast, for SVMs, the gap grows at a predicted $\sqrt{d}$ rate. In the case of anisotropic distributions, we observe similar performance for both regularization schemes.

### 9.1.1 Related work

Kakade et al. (2012) provides generalization error bounds for the more general problem of learning a linear classifier over matrices. An upper bound for quadratic classifiers with a nuclear norm constraint can be derived as a consequence of their results c.f. 9.1. To the best of our knowledge, it is the only known bound for the hypothesis class we study, and thus the one we compare to. Our analysis improves the dependency on the dimension. See subsection 9.2.1 for a technical discussion.

Because of the generality of the results in Kakade et al. (2012), it is only natural that the implied bound in some particular case is not the tightest. We precisely give a step towards tight complexity estimates for classification with quadratic polynomials. We look on our results as relevant, given the simplicity and widespread use of linear learning over features.

Wimalawarne et al. (2016) study linear classifiers over higher-order tensor spaces, using constraints on generalized notions of the nuclear norm. The problem we study is thus a particular case. Generalization error bounds via Rademacher complexity are provided, but they apply only under a highly restrictive assumption: the entries of the tensor are independent standard normal random variables. In contrast we only require a boundedness condition.

Srebro et al. (2005); Srebro and Shraibman (2005) develop the theory of nuclear-norm regularization for matrix completion. Bounds on the Rademacher complexity of the class of matrices with bounded nuclear-norm are obtained in Srebro (2005). However, in this setting matrices are understood as mappings from an index pair to a value, and the generalization error measures how well the missing entries of the target matrix can be predicted. In contrast, in our setting the matrix corresponds to the coefficients of a quadratic polynomial, so the
bounds are not comparable.
Pontil and Maurer (2013) study nuclear-norm regularization in the context of multi-task learning (Caruana, 1997). Rademacher complexity bounds are obtained for nuclear-norm constrained multi-task classifiers. In this setting, each row of the matrix corresponds to a linear classifier for a different task, and each task corresponds to a different distribution over data-label pairs.

Matrices with bounded trace norm also have been studied by Amit et al. (2007) and Yu et al. (2014) in the related multi-class classification and multi-label learning setting, respectively. Yu et al. (2014) remark essential differences between Rademacher complexity bounds of nuclearand Frobenius-norm constrained linear classifiers, similar to our conclusion in 9.2. In all such problems, however, the setting is not comparable to ours: the matrix acts as a linear mapping of the sample, rather than as a quadratic. Thus, the analysis is not analogous and requires a different set of tools in our case.

The analysis in Yu et al. (2014) is closest in spirit to ours, as their bound also depends on the intrinsic dimension of the distribution, and the largest eigenvalue of the second moment (covariance) matrix. However, we go the extra mile and achieve bounds that can be computed from the sample at hand (Theorem 9.3), as the true second moment (covariance) matrix is usually unknown.

The papers Du and Lee (2018); Bai and Lee (2020) establish a similar result as ours for a different norm for a non-convex parametrization of quadratic polynomials. The reparametrization consits of writing the coefficient matrix as a sum of $m$ rank 1 matrices which facilitates analogies to neural networks. It is straighforward to see that their studied norm $\|\cdot\|_{2,4}$ is, in essence, a way of upper bounding the nuclear-norm by using the reparametrization.

### 9.2 Rademacher complexity bounds

Notation. Throughout this section $\mathbf{x} \in \mathbb{R}^{d}$ is a random variable with distribution $\mu$, and $\mathbf{X}_{n}=\left(\mathbf{x}_{1}, \ldots \mathbf{x}_{n}\right)$ is a sample of i.i.d. random variables drawn from $\mu$. The second moment matrix of $\mu$ is denoted by $\Sigma:=\mathbb{E}\left[\mathbf{x} \mathbf{x}^{T}\right]$. Note that, for centered (mean zero) random variables, this notion coincides with the covariance matrix. For a square symmetric matrix A we denote with $\|\mathbf{A}\|_{F},\|\mathbf{A}\|_{2}$ and $\|\mathbf{A}\|_{\text {tr }}$ its Frobenius-norm, spectral-norm and nuclear-norm, respectively. The notation $\lesssim$ stands for less than or equal, but hides constants independent of the dimension or number of samples. The notation $x \approx y$ means that there exist constants $c, C>0$ such that $c y \leq x \leq C y$.

We consider a binary classifier obtained from a homogeneous quadratic polynomial. Such a function can be parametrized as $f_{\mathbf{A}}(\mathbf{x}):=\mathbf{x}^{T} \mathbf{A} \mathbf{x}$, where $\mathbf{A}$ is a square symmetric matrix containing the coefficients of the monomials. In order to control the complexity of a quadratic
polynomial we choose a matrix norm $\|\cdot\|$ and consider only elements in a constrained set:

$$
\begin{equation*}
\mathscr{Q}_{\|\cdot\|, \lambda}:=\left\{f_{\mathbf{A}}(\mathbf{x})=\mathbf{x}^{T} \mathbf{A x}:\|\mathbf{A}\| \leq \lambda\right\}, \quad \mathscr{Q}_{\|\cdot\|}:=\mathscr{Q}_{\|\cdot\|, 1} \tag{9.1}
\end{equation*}
$$

We quantify the complexity of such function classes using the classical notion of Rademacher complexity. It is well known that high probability generalization error bounds can be obtained in terms of this quantity (Koltchinskii and Panchenko, 2002; Bartlett and Mendelson, 2003). For this reason, we focus only on deriving upper bounds on this complexity measure.

Definition 9.1 (Rademacher complexity). Let $\sigma$ be uniformly distributed over the set $\{-1,1\}^{n}$ and let $\mathbf{X}_{n}=\left[\mathbf{x}_{1}, \ldots, \mathbf{x}_{n}\right] \subseteq \mathbb{R}^{d}$ be an i.i.d. sample drawn according to $\mu$. For a class of functions $\mathscr{F}: \mathbb{R}^{d} \rightarrow \mathbb{R}$ we define the empirical Rademacher complexity and the Rademacher complexity (with respect to $\mu$ ) of $\mathscr{F}$, respectively, as:

$$
\begin{equation*}
\hat{\mathscr{R}}\left(\mathscr{F} ; \mathbf{X}_{n}\right):=\mathbb{E}_{\sigma}\left[\sup _{f \in \mathscr{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_{i} f\left(\mathbf{x}_{i}\right)\right], \quad \mathscr{R}_{n, \mu}(\mathscr{F})=\mathbb{E}\left[\hat{\mathscr{R}}\left(\mathscr{F} ; \mathbf{X}_{n}\right)\right] \tag{9.2}
\end{equation*}
$$

Our bounds depend on the distribution through its intrinsic dimension (Tropp, 2015, Section 7), which measures how much the probability density concentrates near low-dimensional subspaces.

Definition 9.2. The intrinsic dimension of a distribution $\mu$ is the ratio $1 \leq r(\Sigma):=\operatorname{tr}(\Sigma) /\|\Sigma\|_{2} \leq$ $d$.

We are now ready to state our main results about the Rademacher complexity of homogeneous quadratic polynomials with nuclear norm constraint:

Theorem 9.1. Let $\mathbf{x} \in \mathbb{R}^{d}$ such that $\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$ almost surely and suppose $n \gtrsim r(\Sigma) \log d$. It holds that

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{t}, \lambda}\right) \lesssim \lambda \sqrt{\frac{r(\Sigma) \log d}{n}}\|\Sigma\|_{2} \tag{9.3}
\end{equation*}
$$

Now we proceed to prove Theorem 9.1. First, we only focus on the class $\mathscr{Q}_{\|\cdot\|}$ corresponding to the unit nuclear-norm ball. This is justified by well-known technical result 9.1 , whose proof is included for completeness in section 9.8.

Lemma 9.1. $\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|, \lambda}\right) \leq \lambda \mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|}\right)$.

The backbone of Theorem 9.1 is 9.3 , which relates the Rademacher complexity of a class of functions to concentration of empirical means to expectations with respect to the dual norm. It makes use of the technical 9.2, whose proof is included for completeness in section 9.6.

Lemma 9.2. For all even $n \in \mathbb{N}$, it holds that $\sum_{k=0}^{n}|2 k-n|\binom{n}{k}<\sqrt{n} 2^{n}$

Lemma 9.3. Denote by $\|\cdot\|_{*}$ the dual norm of $\|\cdot\|$. Define $M_{k}:=\mathbb{E}\left\|\Sigma_{k}-\Sigma\right\|_{*}, \Sigma_{k}:=\frac{1}{k} \sum_{i=1}^{k} \mathbf{x}_{i} \mathbf{x}_{i}^{T}$ The Rademacher complexity of the class $\mathscr{Q}_{\|\cdot\|}$ can be upper bounded as follows:

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|}\right) \leq \frac{1}{n 2^{n-1}} \sum_{k=1}^{n} k\binom{n}{k} M_{k}+\frac{\|\Sigma\|_{*}}{\sqrt{n}} \tag{9.4}
\end{equation*}
$$

Proof. We first compute an upper bound on the empirical Rademacher complexity. The result will follow after taking expectation of the bound over the sample $\mathbf{X}_{n}=\left[\mathbf{x}_{1}, \ldots, \mathbf{x}_{n}\right]$. By definition of the dual norm, using the basic algebraic identity $\mathbf{x}^{T} \mathbf{A x}=\left\langle\mathbf{A}, \mathbf{x x}^{T}\right\rangle$ we have

$$
\begin{equation*}
\hat{\mathscr{R}}\left(\mathscr{Q}_{\|\cdot\|} ; \mathbf{X}_{n}\right)=\frac{1}{n} \mathbb{E}_{\sigma} \sup _{\|\mathbf{A}\| \leq 1}\left\langle\mathbf{A}, \sum_{i=1}^{n} \sigma_{i} \mathbf{x}_{i} \mathbf{x}_{i}^{T}\right\rangle=\frac{1}{n} \mathbb{E}_{\sigma}\left\|\sum_{i=1}^{n} \sigma_{i} \mathbf{x}_{i} \mathbf{x}_{i}^{T}\right\|_{*} \tag{9.5}
\end{equation*}
$$

We now compute the expectation in Equation 9.5. There is a bijection between the possible configurations of the Rademacher variable $\sigma \in\{-1,1\}^{n}$ and the power set of [ $n$ ], namely $\sigma \mapsto\left\{i \in[n]: \sigma_{i}=1\right\}$. This allows us to write Equation 9.5 as:

$$
\begin{equation*}
\hat{\mathscr{R}}\left(\mathscr{Q}_{\|\cdot\|} ; \mathbf{X}_{n}\right)=\frac{1}{n} \mathbb{E}_{\sigma}\left\|\sum_{i=1}^{n} \sigma_{i} \mathbf{x}_{i} \mathbf{x}_{i}^{T}\right\|_{*}=\frac{1}{n 2^{n}} \sum_{B \subseteq[n]} \underbrace{\left\|\sum_{i \in B} \mathbf{x}_{i} \mathbf{x}_{i}^{T}-\sum_{i \in B^{c}} \mathbf{x}_{i} \mathbf{x}_{i}^{T}\right\|_{*}}_{:=D_{B}} \tag{9.6}
\end{equation*}
$$

Let $\Sigma_{B}:=|B|^{-1} \sum_{i \in B} \mathbf{x}_{i} \mathbf{x}_{i}^{T}$. Using the triangle inequality, we can bound $D_{B}$ as:

$$
\begin{equation*}
D_{B} \leq|B|\left\|\Sigma_{B}-\Sigma\right\|_{*}+\left||B|-\left|B^{c}\right|\right|\|\Sigma\|_{*}+\left|B^{c}\right|\left\|\Sigma_{B^{c}}-\Sigma\right\|_{*} \tag{9.7}
\end{equation*}
$$

To obtain a bound on the Rademacher complexity, we need now sum over $B \subseteq[n]$ the terms on the right hand side of Equation 9.7, and take expectation with respect to the sample $\mathbf{X}_{n}$. First, we will deal with the sum of the second term in Equation 9.7, as it is actually a deterministic value. We can sum over $B \subseteq[n]$ by grouping together subsets $B$ of the same cardinality $|B|=k$. We obtain:

$$
\begin{equation*}
\sum_{B \subseteq[n]}\left\|B\left|-\left|B^{c}\right|\right|\right\| \Sigma\left\|_{*}=\sum_{k=0}^{n}\binom{n}{k}|2 k-n|\right\| \Sigma\left\|_{*} \leq \sqrt{n} 2^{n}\right\| \Sigma \|_{*} \tag{9.8}
\end{equation*}
$$

where the last inequality follows from 9.2.
Finally, we compute the expectation of the sum over $B \subseteq[n]$ of the first and third term in Equation 9.7. After taking the sum, both terms become equal by symmetry. It suffices to bound the sum of the first term. Notice that because the variables $\mathbf{x}_{1}, \ldots, \mathbf{x}_{n}$ are i.i.d., the distribution of $\Sigma_{B}$ depends only on the size of the set $B$. Using the same counting argument as
in Equation 9.8 we arrive at:

$$
\begin{align*}
\mathbb{E} \sum_{B \subseteq[n]}|B|\left\|\Sigma_{B}-\Sigma\right\|_{*} & =\sum_{B \subseteq[n]}|B| \mathbb{E}\left\|\Sigma_{B}-\Sigma\right\|_{*} \\
& =\sum_{k=1}^{n} k\binom{n}{k} \underbrace{\mathbb{E}\left\|\Sigma_{k}-\Sigma\right\|_{*}}_{=M_{k}} \tag{9.9}
\end{align*}
$$

Combining the bounds in Equation 9.8 and Equation 9.9, and dividing by $n 2^{n}$ we obtain the result.

Note that the proof of 9.3 follows from technical arguments but the final result is not to be found in the literature, in this form or a similar one. In particular, it is completely unrelated to the result by Vershynin (2011) with which it only shares the fairly trivial split of Rademacher random variables preceding Equation 9.6.
9.3 provides a way to derive Rademacher complexity bounds from a bound on the expected deviations $M_{k}$ defined in 9.3, and might be of independent interest. In the particular case where the norm in consideration is the nuclear-norm, this lemma will be used to establish Theorem 9.1 as a simple application of a well-known non-asymptotic bound for the convergence of the empirical second moment (covariance) matrix to the true second moment (covariance) matrix.

Proof of Theorem 9.1. Recall that the dual norm of the nuclear-norm is the spectral-norm. The value of $M_{k}$ in 9.3 measures the average deviation of the empirical second moment matrix $\Sigma_{k}$ to the true $\Sigma$, in spectral-norm. Our assumption that $\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$ almost surely, implies the concentration result (Vershynin, 2018, Theorem 5.6.1), which concludes that

$$
\begin{equation*}
M_{k} \lesssim\left(\sqrt{\frac{r(\Sigma) \log d}{k}}+\frac{r(\Sigma) \log d}{k}\right)\|\Sigma\|_{2} \tag{9.10}
\end{equation*}
$$

Plug this in Equation 9.4, and use the bound $\sqrt{k} \leq \sqrt{n}$ for $k \leq n$ to obtain the inequality

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{t r}}\right) \lesssim\left(\sqrt{\frac{r(\Sigma) \log d}{n}}+\frac{r(\Sigma) \log d}{n}\right)\|\Sigma\|_{2} \tag{9.11}
\end{equation*}
$$

By assumption $n \gtrsim r(\Sigma) \log d$, so that the first term in Equation 9.11 is the largest. The second term is of smaller order and thus ends up hidden by the notation $\lesssim$. We conclude

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{t r}}\right) \lesssim \sqrt{\frac{r(\Sigma) \log d}{n}}\|\Sigma\|_{2} \tag{9.12}
\end{equation*}
$$

Invoking 9.1 we obtain the desired result.

### 9.2.1 Improvement upon previous work

We now show how our derived upper bound improves over the current best known bound by Kakade et al. (2012). To the best of our knowledge, the Rademacher complexity of quadratic classifiers with a nuclear-norm constraint has not been previously analyzed in a direct manner, as we do. Instead, the only existing bound (9.1) appears as a particular case of Theorem 9.2.

Theorem 9.2 (Kakade et al. (2012) page 1876). Let

$$
\begin{equation*}
\mathscr{G}_{\| \| \|_{\mathrm{t}}, \lambda}:=\left\{\mathrm{g}_{\mathbf{A}}(\mathbf{X}):=\langle\mathbf{A}, \mathbf{X}\rangle:\|\mathbf{A}\|_{\mathrm{tr}} \leq \lambda\right\} \tag{9.13}
\end{equation*}
$$

be the class of nuclear-norm constrained linear functions over square $d \times d$ matrices. Let $\mu$ be a distribution supported on $\mathscr{X} \subseteq \mathbb{R}^{d \times d}$. It holds that:

$$
\mathscr{R}_{n, \mu}\left(\mathscr{G}_{\|\cdot\| \|_{\mathrm{t}}, \lambda}\right) \lesssim \lambda X_{\infty} \sqrt{\frac{\log d}{n}}, \quad X_{\infty}=\sup _{\mathbf{X} \in \mathscr{\mathscr { C }}}\|\mathbf{X}\|_{2}
$$

Corollary 9.1. Let $\mathbf{x} \in \mathscr{X} \subseteq \mathbb{R}^{d}$ be a random variable with distribution $\mu$. The Rademacher complexity of the class in Equation 9.1 can be bounded as

$$
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{r}, \lambda}\right) \lesssim \lambda x_{\infty} \sqrt{\frac{\log d}{n}}, \quad x_{\infty}:=\sup _{\mathbf{x} \in \mathscr{C}}\|\mathbf{x}\|_{2}^{2}
$$

9.1 is a consequence of the fact that a function of the form $f_{\mathbf{A}}(\mathbf{x})=\mathbf{x}^{T} \mathbf{A x}$ can be written as a linear classifier on matrices, $g_{\mathbf{A}}(\mathbf{X})=\langle\mathbf{A}, \mathbf{X}\rangle$, where $\mathbf{X}=\mathbf{\mathbf { x } ^ { T }}$. In this case, it is easy to see that $X_{\infty}=\sup _{\mathbf{x} \in \mathscr{\mathscr { C }}}\left\|\mathbf{x} \mathbf{x}^{T}\right\|=\sup _{\mathbf{x} \in \mathscr{C}}\|\mathbf{x}\|_{2}^{2}=: x_{\infty}$. The only difference with our bound in Theorem 9.1 is that the term $x_{\infty}$ appears in place of $\sqrt{r(\Sigma)}\|\Sigma\|_{2}$.

In order to understand the difference between the two bounds, we turn to the analysis of the quotient between the bound in 9.1 and our bound Theorem 9.1. In 9.4 we show that this quotient can differ drastically, depending on the distribution.

Remark 9.1. The variables $X_{\infty}$ and $x_{\infty}$ defined respectively in Theorem 9.2 and 9.1 respectively, correspond to the supremum of a random variable. Because the Rademacher complexity arises as an expectation, it is clear that such quantities can be (and should be) replaced by the closely related measure-theoretic notion of essential supremum (denoted by ess sup): the least upper bound that holds almost surely. In this way the bounds are only tighter, and we believe this was the true intended definition by Kakade et al. (2012). In the following we will compare our bound to this tighter, modified bound.

Lemma 9.4. Let $\mathbf{x}$ be a random variable supported on a set $\mathscr{X} \subseteq \mathbb{R}^{d}$, and such that $\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$ almost surely, then:

$$
\begin{equation*}
\sqrt{r(\Sigma)} \lesssim \frac{\text { ess sup }_{\mathbf{x} \in \mathscr{X}}\|\mathbf{x}\|_{2}^{2}}{\sqrt{r(\Sigma)}\|\Sigma\|_{2}} \lesssim \sqrt{r(\Sigma)} \tag{9.14}
\end{equation*}
$$

Proof. By definition of essential supremum it holds that $\mathbb{E}\|\mathbf{x}\|_{2}^{2} \leq \operatorname{esss}_{\sup _{\mathbf{x} \in \mathscr{X}}\|\mathbf{x}\|_{2}^{2} \text {. Further, our }}$ assumption clearly implies that ess $\sup _{\mathbf{x} \in \mathscr{X}}\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$. The identity $\mathbb{E}\|\mathbf{x}\|_{2}^{2}=\operatorname{tr}(\Sigma)$ and 9.2 imply the result.

In summary, 9.4 shows that the baseline bound of Kakade et al. (2012) is larger by a square-root factor of the intrinsic dimension of the distribution (modulo global constants), compared to our bound in Theorem 9.1. Such factor ranges between 1 and the square root of the ambient dimension.

Precisely, when the intrinsic dimension of the distribution is equal to the ambient dimension, our bound enjoys a reduced dimension complexity. This is the case, for example, for isotropic distributions i.e., distributions such that their second moment matrix is the identity matrix. The dependency of the Rademacher complexity on the intrinsic dimension of the distribution is not revealed by the more general proof of Theorem 9.2 (Kakade et al., 2012).

Remark 9.2. The logarithmic term in Theorem 9.1 can be removed under the more restrictive dimension-independent L-subgaussianity assumption (Mendelson and Zhivotovskiy, 2018).

### 9.3 Computable Generalization error bounds

Let $\mathbf{y}=\left(y_{1}, \ldots, y_{n}\right) \in\{-1,1\}^{n}$ be the labels associated with the data sample, and let

$$
\begin{gather*}
L(f):=\mathbb{P}\{\operatorname{sign}(f(\mathbf{x})) \neq \mathbf{y}\}  \tag{9.15}\\
\hat{L}\left(f ; \mathbf{X}_{n}\right):=\frac{1}{n} \sum_{i=1}^{n} \min \left(1, \max \left(0,1-y_{i} f\left(\mathbf{x}_{i}\right)\right)\right) \tag{9.16}
\end{gather*}
$$

be the missclassification probability and the empirical margin loss of the classifier $f$, respectively. It is well-known (Mohri et al., 2018b, Theorem 5.8.) that with probability at least $1-\delta$, for all $f \in \mathscr{Q}_{\|\cdot\|_{t r}, \lambda}$ :

$$
\begin{equation*}
L(f) \lesssim \hat{L}\left(f ; \mathbf{X}_{n}\right)+\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{\mathrm{t}}, \lambda}\right)+\sqrt{\frac{\log \frac{1}{\delta}}{2 n}} \tag{9.17}
\end{equation*}
$$

This bound, together with the results in section 9.2, allow high probability uniform bounds on the misclassification error of a nuclear-norm constrained quadratic classifier.

However, the bound derived in this way is not actually computable: the Rademacher complexity bound in Theorem 9.1 depends on the second moment matrix of the distribution, which is unknown in practical applications. In the rest of this section we will overcome this drawback. Rewriting our bound in Equation 9.3 as:

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{t r}}\right) \lesssim \sqrt{\frac{\operatorname{tr} \Sigma\|\Sigma\|_{2} \log d}{n}} \tag{9.18}
\end{equation*}
$$

we observe that we need to estimate the trace and the largest eigenvalue of the second order moment matrix. Our hope is that the empirical estimators, based on the empirical second moment matrix $\Sigma_{n}$, will provide a good approximation of the true values.

We arrive at the following high probability bound on the Rademacher complexity in Equation 9.18, which is readily computable from samples and a bound on the diameter of the support of the distribution:

Theorem 9.3. Suppose that $\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$ almost surely. This implies that $\|\mathbf{x}\|_{2}^{2} \leq B$ for some $B>0$. Define

$$
\begin{equation*}
K_{1}:=\frac{\log d}{n-\sqrt{n d\left(\log d+\log \frac{1}{\delta}\right)}} \quad K_{2}:=\frac{\sqrt{2 \log \frac{1}{\delta}} \log d}{2 n\left(\sqrt{n}-\sqrt{d\left(\log d+\log \frac{1}{\delta}\right)}\right)} \tag{9.19}
\end{equation*}
$$

Let $\delta>0$ and $f_{\mathrm{A}} \in \mathscr{Q}_{\|\cdot\|_{\mathrm{tr}}, \lambda}$. Provided $n \gtrsim d\left(\log d+\log \frac{1}{\delta}\right)$, with probability at least $1-3 \delta$ it holds that

$$
\begin{equation*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{t r}}\right) \lesssim \underbrace{\sqrt{K_{1} \operatorname{tr} \Sigma_{n}\left\|\Sigma_{n}\right\|_{2}+B K_{2}\left\|\Sigma_{n}\right\|_{2}}}_{:=M(n, d, \delta)} \tag{9.20}
\end{equation*}
$$

Moreover, with probability at least $1-4 \delta$, uniformly for all $f_{\mathbf{A}} \in \mathscr{Q}_{\|\cdot\|_{\mathrm{r}}, \lambda}$ it holds that

$$
\begin{equation*}
L\left(f_{\mathbf{A}}\right) \lesssim \hat{L}\left(f_{\mathrm{A}} ; \mathbf{X}_{n}\right)+\lambda M(n, d, \delta)+\sqrt{\frac{\log \frac{1}{\delta}}{2 n}} \tag{9.21}
\end{equation*}
$$

Proof. See section 9.7.

### 9.4 Frobenius vs Nuclear-norm constraint

Perhaps the most common way to use quadratic features is to use a support vector machine (SVM) with the quadratic kernel $K(\mathbf{x}, \mathbf{y}):=\langle\mathbf{x}, \mathbf{y}\rangle^{2}$. 9.5, which is folklore in the kernel methods literature, precisely states that the RKHS norm constraint is equivalent to a Frobenius-norm constraint on the matrix of coefficients $\mathbf{A}$ of the underlying quadratic polynomial. Its proof is included for completeness in section 9.8.

Lemma 9.5. Let $\mathbb{H}$ be the Reproducing Kernel Hilbert Space associated to the symmetric, positive semidefinite polynomial kernel $K(\mathbf{x}, \mathbf{y})=\langle\mathbf{x}, \mathbf{y}\rangle^{2}$, and denote its induced norm by $\|\cdot\|_{\mathbb{H}}$. Then $f \in \mathbb{H}$ if and only if there exists a symmetric matrix $\mathbf{A}$ such that $f(\mathbf{x})=\mathbf{x}^{T} \mathbf{A} \mathbf{x}$ and $\|f\|_{\mathbb{H}}=\|\mathbf{A}\|_{F}$.

For this reason, we now turn to compare the qualities of nuclear-norm and Frobenius-norm constrained quadratic classifiers. As a consequence of our derived bound (Equation 9.3), we uncover a fundamental difference between both regularization schemes (9.2): as the dimension increases, the growth rate of the complexity of nuclear-norm constrained quadratics strongly depends on the intrinsic dimension of the distribution. In contrast, that of Frobeniusnorm constrained quadratics is insensitive to it.

In order to derive this rate, we need a way to argue about distributions across different dimensions and express our generalization bounds only in terms of dimension and number of samples. To this end, we introduce a natural boundedness assumption on the data distribution:

Assumption 9.1. $\|\mathbf{x}\|_{2}^{2} \approx d$ almost surely.

We now argue why this is a natural scaling order for this norm: if the entries of the random vector $\mathbf{x}$ are upper bounded as $\left|\mathbf{x}_{i}\right| \leq \alpha_{\text {max }}$ and their average magnitude is lower bounded as $0<\alpha_{\text {min }} \leq \frac{1}{d} \sum_{i=1}^{d}\left|\mathbf{x}_{i}\right|$. Then

$$
\alpha_{\min } \sqrt{d} \leq \frac{1}{\sqrt{d}}\|\mathbf{x}\|_{1} \leq\|\mathbf{x}\|_{2} \leq \sqrt{d}\|\mathbf{x}\|_{\infty} \leq \sqrt{d} \alpha_{\max }
$$

and hence, 9.1 is satisfied. If we think about distributions of pixel (natural image) data of increasing resolution, we indeed have an upper bound on the intensity of each pixel. A lower bounded average pixel intensity only means that images are not arbitrarily dark, which often holds in practice. Any distribution of similar characteristics (e.g., sensor data) will probably satisfy our assumption.

We also introduce a growth condition on the intrinsic dimension, which states that $r(\Sigma) \approx d^{s}$ for some $0 \leq s \leq 1$. Rather than being an assumption, this condition helps to understand how our bounds change as the distribution falls between the two possible extremes given by the bound $1 \leq r(\Sigma) \leq d$. Traditionally, distributions that attain the lower bound $(s=0, r(\Sigma)=1)$ or the upper bound ( $s=1, r(\Sigma)=d$ ) are called anisotropic or isotropic, respectively.

Corollary 9.2. Let 9.1 hold, and suppose that $r(\Sigma) \approx d^{s}$ for some $s \in[0,1]$ and $n \geq r(\Sigma) \log d$. Then

$$
\begin{align*}
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{\mathrm{t}}, \lambda}\right) & \lesssim \lambda \frac{d^{1-s / 2} \sqrt{\log d}}{\sqrt{n}}  \tag{9.22}\\
\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{F}, \lambda}\right) & \approx \lambda \frac{d}{\sqrt{n}} \tag{9.23}
\end{align*}
$$

Proof. For the first inequality, note that $\|\mathbf{x}\|_{2}^{2} \approx d$ implies that $\operatorname{tr} \Sigma=\mathbb{E}\|\mathbf{x}\|_{2}^{2} \approx d$. Thus, $\frac{\operatorname{tr} \Sigma}{\|\Sigma\|_{2}}=$ $r(\Sigma) \approx d^{s} \Longrightarrow\|\Sigma\|_{2} \approx d^{1-s}$ Using these two identities in the inequality Equation 9.3 we obtain the first result.

For the second identity, 9.5 implies the class of Frobenius-norm constrained quadratic functions is equal to the ball of radius $\lambda$ in the RKHS corresponding to the quadratic kernel. By Mohri et al. (2018b, Theorem 5.5) ${ }^{1}$, we have that

$$
\begin{equation*}
\hat{\mathscr{R}}\left(\mathscr{Q}_{\|\cdot\|_{F}, \lambda} ; \mathbf{X}_{n}\right) \approx \frac{\lambda}{n} \sqrt{\sum_{i=1}^{n} K\left(\mathbf{x}_{i}, \mathbf{x}_{i}\right)}=\frac{\lambda}{n} \sqrt{\sum_{i=1}^{n}\left\|\mathbf{x}_{i}\right\|_{2}^{4}} \approx \frac{\lambda d}{\sqrt{n}} \tag{9.24}
\end{equation*}
$$

[^15]where the final inequality comes from our assumption that $\|\mathbf{x}\|_{2}^{2} \approx d$. Taking expectation with respect to the sample we obtain that $\mathscr{R}_{n, \mu}\left(\mathscr{Q}_{\|\cdot\|_{F}, \lambda}\right) \approx \lambda \frac{d}{\sqrt{n}}$.
9.2 then states that the nuclear norm constraint adapts to the intrinsic dimension of the distribution. In the worst case $(s=0)$ it grows linearly ${ }^{2}$ with dimension; In the best case $(s=1)$ it is much slower, and grows as the square root of the dimension.

Note that we can bring any distribution to approximate isotropic position, given a good approximation of $\Sigma$. Hence, it makes sense to expect performance gains if we perform such normalization procedure before training nuclear-norm constrained quadratic classifiers. In section 2.7 we will put to test this claim in synthetic datasets, to illustrate the theory.

### 9.5 Experiments

The results derived in corollary 9.2 pertain to the worst case generalization gap. This worst case gap is not guaranteed to be attained by the function found through empirical loss minimization. Therefore, in order to better test our results, we try to find the function that attains this worst case generalization gap.

The experiment we propose to illustrate the difference in intrinsic dimension sensitivity consists of computing the quadratic function $f$ such that $\mathbb{E}[\ell(f(\mathbf{x}, y))]-\frac{1}{n} \sum_{i=1}^{n} \ell\left(f\left(\mathbf{x}_{i}, y_{i}\right)\right)$ is maximized. The expectation in the term above being generally difficult to compute in closed form, we replace the objective by an empirical estimate

$$
\frac{1}{n_{\text {test }}} \sum_{k=1}^{n_{\text {test }}} \ell\left(f\left(\mathbf{x}_{k}, y_{k}\right)\right)-\frac{1}{n_{\text {train }}} \sum_{i=1}^{n_{\text {train }}} \ell\left(f\left(\mathbf{x}_{i}, y_{i}\right)\right)
$$

We expect this quantity to more faithfully track the upperbounds derived in corollary 9.2. Indeed, what we evaluate here is the capacity of the class constraining $f$ to attain a low loss on $n_{\text {train }}$ samples while attaining a very high one on $n_{\text {test }}$ samples.

To illustrate that Frobenius and nuclear norm constrained classifiers exhibit different behaviors depending on the intrinsic dimension of the data, we compute the result of the maximization procedure on isotropic distributions on one hand and on anisotropic on the other. We set the radius $\lambda=1$ for both Nuclear and Frobenius norm constrained classifiers. We then observe the evolution of these quantities as the dimension increases.

Generating Isotropic Data. To generate isotropic data satisfying our assumptions, we sample a standard Gaussian random vector in $\mathbb{R}^{d}$ and normalize to obtain i.i.d samples $\mathbf{x}_{i}$ that are uniformly distributed on the sphere of radius $\sqrt{d}$.

Generating Anistropic Data. To generate anisotropic data, we first generate isotropic data

[^16]features by uniformly sampling points on the sphere of radius $\sqrt{d}$ as described previously. We then transform the samples into more anisotropic ones by multiplying each feature vector coordinate-wise by a vector $\alpha^{(s)}$ defined as $\alpha_{i}^{(s)}=C_{s} \frac{1}{i^{s}}, i=1, \ldots, d$, for $s \in[0,1]$ where $C_{s}$ is chosen such that $\left\|\alpha^{(s)}\right\|_{2}=\sqrt{d}$. That, way, $s=0$ induces no change on the data, since $\alpha^{(1)}=[1,1, \ldots, 1]^{T}$, and larger $s$ implies that the data will be more squashed along the first few coordinates, inducing smaller intrinsic dimension. This transform does not affect the norm of the vectors since $\mathbb{E}\left[\left\|\alpha^{(s)} \times \mathbf{x}_{i}\right\|^{2}\right]=\sum_{k=1}^{d}\left(\alpha_{k}^{(s)}\right)^{2} \mathbb{E}\left[\left[\mathbf{x}_{i}\right]_{k}^{2}\right]=\sum_{k=1}^{d}\left(\alpha_{k}^{(s)}\right)^{2}=\sqrt{d}$

Generating the labels. We test two approaches for generating the labels: (1) We generate a random matrix $A$ with i.i.d standard Gaussian coordinates and set $y_{i}=\operatorname{sgn}\left(\mathbf{x}_{i}^{\top} A \mathbf{x}_{i}\right)$, (2) We set the labels randomly with $y_{i} \sim \operatorname{Bern}(0.5)$.

## The results.



Figure 9.1: Maximized Train/Test loss gaps for each dimension $d$. The results are averaged over 5 independent runs. The error bars correspond to the standard error.

We observe that the maximal gaps do indeed evolve as the theory predicts. We indeed have the nuclear norm constrained classifier on isotropic data not exhibiting the same growth as the others, indicating that the maximal generalization gap for nuclear norm constrained classifiers is indeed sensitive to isotropy. Additional experiments and details are provided in section 9.9.

### 9.6 Appendix: Proof of lemma 9.2

For completeness, we include the following result, taken from https://math.stackexchange. com/questions/3209660/show-that-2m-choose-m-leq-frac22m-sqrt2m:

Lemma 9.6. For $m \in \mathbb{N}$ we have

$$
\begin{equation*}
\binom{2 m}{m}<\frac{2^{2 m}}{\sqrt{2 m+1}}<\frac{2^{2 m}}{\sqrt{2 m}} \tag{9.25}
\end{equation*}
$$

Proof. By induction on $m \in \mathbb{N}$. For $m=1$ the inequality holds. Suppose it holds for some
$m \in \mathbb{N}$

$$
\begin{equation*}
\frac{(2 m+2)!}{2^{2 m+2}((m+1)!)^{2}}=\frac{(2 m)!}{2^{2 m}(m!)^{2}} \cdot \frac{2 m+1}{2 m+2}<\frac{1}{\sqrt{2 m+1}} \cdot \frac{2 m+1}{2 m+2} \stackrel{?}{\leq} \frac{1}{\sqrt{2 m+3}} . \tag{9.26}
\end{equation*}
$$

So it remains to prove that

$$
\begin{equation*}
4 m^{2}+8 m+3=(\sqrt{2 m+1} \sqrt{2 m+3})^{2} \leq(2 m+2)^{2}=4 m^{2}+8 m+4 \tag{9.27}
\end{equation*}
$$

which holds.

Lemma 9.7. For all $n \in \mathbb{N}$ even:

$$
\sum_{k=0}^{n}\binom{n}{k}|2 k-n|=n\binom{n}{\frac{n}{2}}
$$

Proof. Suppose that $n=2 m$ is even.

$$
\begin{aligned}
\sum_{k=0}^{n}\binom{n}{k}|2 k-n| & =\sum_{k=0}^{m}\binom{2 m}{k}(2 m-2 k)+\sum_{k=m+1}^{2 m}\binom{2 m}{k}(2 k-2 m) \\
& =2 \sum_{k=0}^{m}\binom{2 m}{k}(m-k)+\sum_{k=0}^{m-1}\binom{2 m}{2 m-k}(2(2 m-k)-2 m) \\
& =2 \sum_{k=0}^{m}\binom{2 m}{k}(m-k)+2 \sum_{k=0}^{m-1}\binom{2 m}{k}(m-k) \\
& =4 \sum_{k=0}^{m}\binom{2 m}{k}(m-k)
\end{aligned}
$$

We now show by induction that $\forall m \in \mathbb{N}$,

$$
\begin{equation*}
\sum_{k=0}^{m}\binom{2 m}{k} k=m 2^{2 m-1} \tag{9.28}
\end{equation*}
$$

The result holds trivially true for $m=0$. Suppose it is true for some $m \geq 0$ and let's prove it remains true for $m+1$ :

$$
\begin{aligned}
\sum_{k=0}^{m+1}\binom{2 m+2}{k} k & =\sum_{k=0}^{m+1}\left(\binom{2 m}{k}+2\binom{2 m}{k-1}+\binom{2 m}{k-2}\right) k \\
& =\sum_{k=0}^{m+1}\binom{2 m}{k} k+2 \sum_{k=1}^{m+1}\binom{2 m}{k-1} k+\sum_{k=2}^{m+1}\binom{2 m}{k-2} k \\
& =\sum_{k=0}^{m+1}\binom{2 m}{k} k+2 \sum_{k=0}^{m}\binom{2 m}{k}(k+1)+\sum_{k=0}^{m-1}\binom{2 m}{k}(k+2) \\
& =4 \sum_{k=0}^{m}\binom{2 m}{k} k+\binom{2 m}{m+1}(m+1)+2 \sum_{k=0}^{m}\binom{2 m}{k}-\binom{2 m}{m} m+2 \sum_{k=0}^{m-1}\binom{2 m}{k} \\
& =4 m 2^{2 m-1}+2 \sum_{k=0}^{2 m}\binom{2 m}{k}
\end{aligned}
$$

$$
\begin{aligned}
& =m 2^{2(m+1)-1}+22^{2 m} \\
& =(m+1) 2^{2(m+1)-1} .
\end{aligned}
$$

We thus proved that Equation 9.28 is true for all $m \in \mathbb{N}$. Thus, we have:

$$
\begin{aligned}
\sum_{k=0}^{m}\binom{2 m}{k}(m-k) & =m \sum_{k=0}^{m}\binom{2 m}{k}-m 2^{2 m-1} \\
& =\frac{m}{2}\left(\sum_{k=0}^{m}\binom{2 m}{k}+\sum_{k=m}^{2 m}\binom{2 m}{k}\right)-m 2^{2 m-1} \\
& =\frac{m}{2}\left(\sum_{k=0}^{2 m}\binom{2 m}{k}+\binom{2 m}{m}\right)-m 2^{2 m-1} \\
& =\frac{m}{2}\left(2^{2 m}+\binom{2 m}{m}\right)-m 2^{2 m-1} \\
& =\frac{m}{2}\binom{2 m}{m}
\end{aligned}
$$

which concludes the proof.

Proof of 9.2. Combine 9.6 and 9.7.

### 9.7 Appendix: Proof of theorem 9.3

Lemma 9.8. Suppose that $n \gtrsim d(\log (d)+\log (1 / \delta))$ and that $\|\mathbf{x}\|_{2}^{2} \lesssim \mathbb{E}\|\mathbf{x}\|_{2}^{2}$ almost surely. With probability at least $1-2 \delta$ :

$$
\begin{equation*}
\|\Sigma\|_{2} \lesssim\left(1-\sqrt{\frac{d\left(\log d+\log \frac{1}{\delta}\right)}{n}}\right)^{-1}\left\|\Sigma_{n}\right\|_{2} \tag{9.29}
\end{equation*}
$$

Proof. The lemma is an application of the matrix Bernstein inequality. Recall that $\Sigma_{n}=$ $\sum_{i=1}^{n} \mathbf{x}_{i} \mathbf{x}_{i}^{T}$ where $\mathbf{x}_{i}$ for $i \in\{1, \ldots, n\}$ are independent, identically distributed and, by assumption, satisfy the inequality $\left\|\mathbf{x}_{i}\right\|_{2}^{2} \lesssim \mathbb{E}\left\|\mathbf{x}_{i}\right\|_{2}^{2}=\operatorname{tr}(\Sigma)$ almost surely. Spelling out the definition of $\lesssim$, we have that there exists a $C \geq 1$ such that

$$
\forall i \in\{1, \ldots, n\}, \quad\left\|\mathbf{x}_{i} \mathbf{x}_{i}^{T}\right\|_{2}=\left\|\mathbf{x}_{i}\right\|_{2}^{2} \leq C \operatorname{tr}(\Sigma) \quad \text { almost surely. }
$$

Consequently, by matrix Bernstein, for any $t \geq 0$,

$$
\mathbb{P}\left(\left\|\sum_{i=1}^{n}\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}-\Sigma\right)\right\|_{2} \geq t\right) \leq 2 d \exp \left(-\frac{\frac{t^{2}}{2}}{\sigma^{2}+K \frac{t}{3}}\right)
$$

where $K=C \operatorname{tr}(\Sigma)$ and $\sigma^{2}=\left\|\sum_{i=1}^{n} \mathbb{E}\left[\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}-\Sigma\right)^{2}\right]\right\|_{2}$.

Now for any $0<\delta \leq 1$, we take

$$
t=\sqrt{2}\left(\sigma \sqrt{\log d+\log \frac{1}{\delta}}+K\left(\log d+\log \frac{1}{\delta}\right)\right)
$$

The term inside the exponential on the right hand side of the Bernstein bound then becomes

$$
\begin{aligned}
-\frac{\frac{t^{2}}{2}}{\sigma^{2}+K \frac{t}{3}} & =-\frac{\sigma^{2}\left(\log d+\log \frac{1}{\delta}\right)+2 K \sigma \sqrt{\log d+\log \frac{1}{\delta}}\left(\log d+\log \frac{1}{\delta}\right)+K^{2}\left(\log d+\log \frac{1}{\delta}\right)^{2}}{\sigma^{2}+\frac{\sqrt{2} K}{3} \sigma \sqrt{\log d+\log \frac{1}{\delta}}+\frac{\sqrt{2} K^{2}}{3}\left(\log d+\log \frac{1}{\delta}\right)} \\
& =-\frac{\left(\log d+\log \frac{1}{\delta}\right)\left(\sigma^{2}+2 K \sigma \sqrt{\log d+\log \frac{1}{\delta}}+K^{2}\left(\log d+\log \frac{1}{\delta}\right)\right)}{\sigma^{2}+\frac{\sqrt{2} K}{3} \sigma \sqrt{\log d+\log \frac{1}{\delta}}+\frac{\sqrt{2} K^{2}}{3}\left(\log d+\log \frac{1}{\delta}\right)} \\
& \leq-\left(\log d+\log \frac{1}{\delta}\right) \quad\left(\text { since } \frac{\sqrt{2}}{3} \leq 1 \text { and } \log d+\log \frac{1}{\delta} \geq 0\right) .
\end{aligned}
$$

Consequently, the event

$$
\begin{equation*}
\left\|\sum_{i=1}^{n}\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}-\Sigma\right)\right\|_{2} \leq \sqrt{2}\left(\sigma \sqrt{\log d+\log \frac{1}{\delta}}+K\left(\log d+\log \frac{1}{\delta}\right)\right) \tag{9.30}
\end{equation*}
$$

holds with probability at least $1-2 \delta$. Now, by proceding just like Theorem 5.6.1 of Vershynin (2018), we can bound $\sigma^{2}$ :

$$
\begin{array}{rlr}
\sigma^{2} & =\left\|\sum_{i=1}^{n} \mathbb{E}\left[\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}-\Sigma\right)^{2}\right]\right\|_{2} \\
& =n\left\|\mathbb{E}\left[\left(\mathbf{x}_{1} \mathbf{x}_{1}^{T}-\Sigma\right)^{2}\right]\right\|_{2} & \\
& =n\left\|\mathbb{E}\left[\left(\mathbf{x}_{1} \mathbf{x}_{1}^{T}\right)^{2}\right]-\Sigma^{2}\right\|_{2} & \text { (by expanding the square) } \\
& \leq n\left\|\mathbb{E}\left[\left(\mathbf{x}_{1} \mathbf{x}_{1}^{T}\right)^{2}\right]\right\|_{2} & \text { (since } \Sigma \succeq 0 \text { ) } \\
& =n\left\|\mathbb{E}\left[\left\|\mathbf{x}_{1}\right\|_{2}^{2}\left(\mathbf{x}_{1} \mathbf{x}_{1}^{T}\right)\right]\right\|_{2} & \\
& \leq n C \operatorname{tr}(\Sigma)\left\|\mathbb{E}\left[\mathbf{x}_{1} \mathbf{x}_{1}^{T}\right]\right\|_{2} & \text { (by assumption) } \\
& =n C \operatorname{tr}(\Sigma)\|\Sigma\|_{2} . &
\end{array}
$$

Plugging this bound into (9.30) and dividing by $n$, we find that

$$
\left\|\Sigma_{n}-\Sigma\right\|_{2} \leq \sqrt{2}\left(\sqrt{n C \operatorname{tr}(\Sigma)\|\Sigma\|_{2}} \frac{\sqrt{\log d+\log \frac{1}{\delta}}}{n}+C \operatorname{tr}(\Sigma) \frac{\left(\log d+\log \frac{1}{\delta}\right)}{n}\right)
$$

Factorizing by $\|\Sigma\|_{2}$ on the RHS and using the assumption that $C d\left(\log d+\log \frac{1}{\delta}\right) \lesssim n$, we find

$$
\left\|\Sigma_{n}-\Sigma\right\|_{2} \lesssim \sqrt{\frac{r(\Sigma)\left(\log d+\log \frac{1}{\delta}\right)}{n}}\|\Sigma\|_{2}
$$

With this in hand, we simply use the triangle inequality to obtain

$$
\begin{aligned}
\|\Sigma\|_{2} & =\left\|\Sigma-\Sigma_{n}+\Sigma_{n}\right\|_{2} \\
& =\left\|\Sigma_{n}-\Sigma\right\|_{2}+\left\|\Sigma_{n}\right\|_{2} \\
& \lesssim \sqrt{\frac{r(\Sigma)\left(\log d+\log \frac{1}{\delta}\right)}{n}}\|\Sigma\|_{2}+\left\|\Sigma_{n}\right\|_{2}
\end{aligned}
$$

Isolating $\|\Sigma\|_{2}$ on the left hand side, we find that

$$
\|\Sigma\|_{2} \lesssim\left(1-\sqrt{\frac{r(\Sigma)\left(\log d+\log \frac{1}{\delta}\right)}{n}}\right)^{-1}\left\|\Sigma_{n}\right\|_{2} .
$$

Finally we have that $r(\Sigma) \leq d$.

Lemma 9.9. Suppose that $\|\mathbf{x}\|_{2}^{2} \leq B$ almost surely. With probability at least $1-\delta$ :

$$
\begin{equation*}
\operatorname{tr} \Sigma \leq \Sigma_{n}+\sqrt{\frac{B^{2} \log (1 / \delta)}{2 n}} \tag{9.31}
\end{equation*}
$$

Proof. We define

$$
\begin{equation*}
\sigma\left(\mathbf{x}_{1}, \ldots, \mathbf{x}_{n}\right)=\frac{1}{n} \sum_{i=1}^{n} \operatorname{tr}\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}\right)=\operatorname{tr} \Sigma_{n} \tag{9.32}
\end{equation*}
$$

The empirical second moment matrix is an unbiased estimator of the second moment matrix $\Sigma=\mathbb{E} \Sigma_{n}$. Hence

$$
\begin{equation*}
\operatorname{tr} \Sigma=\operatorname{tr} \mathbb{E} \Sigma_{n}=\mathbb{E} \operatorname{tr} \Sigma_{n}=\mathbb{E} \sigma\left(\mathbf{x}_{1}, \ldots, \mathbf{x}_{n}\right) \tag{9.33}
\end{equation*}
$$

We observe that the quantity we want to bound $\left|\operatorname{tr} \Sigma-\operatorname{tr} \Sigma_{n}\right|$ is precisely the deviation of $\sigma$ to its expected value.

In order to obtain a high-probability bound on this deviation, we will make use of McDiarmid's Inequality. We first show that $\sigma$ satisfies the bounded difference inequality

$$
\begin{equation*}
\left|\sigma\left(\mathbf{x}_{1}, \ldots, \mathbf{x}_{i}, \ldots, \mathbf{x}_{n}\right)-\sigma\left(\mathbf{x}_{1}, \ldots, \hat{\mathbf{x}}_{i}, \ldots, \mathbf{x}_{n}\right)\right|=\frac{1}{n}\left|\operatorname{tr}\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}-\hat{\mathbf{x}}_{i} \hat{\mathbf{x}}_{i}^{T}\right)\right|=\frac{1}{n}\left|\left\|\mathbf{x}_{i}\right\|^{2}-\left\|\hat{\mathbf{x}}_{i}^{2}\right\|\right| \leq \frac{B}{n} \tag{9.34}
\end{equation*}
$$

Hence by McDiarmid's inequality we conclude that

$$
\begin{equation*}
P\left(\operatorname{tr} \Sigma_{n}-\operatorname{tr} \Sigma<-t\right) \leq \exp \left(\frac{-2 n t^{2}}{B^{2}}\right) \tag{9.35}
\end{equation*}
$$

Letting $\delta=\exp \left(\frac{-2 n t^{2}}{B^{2}}\right)$ we have $t=\sqrt{\frac{B^{2} \log (11 / \delta)}{2 n}}$ so that with probability at least $1-\delta$

$$
\begin{equation*}
\operatorname{tr} \Sigma \leq \Sigma_{n}+\sqrt{\frac{B^{2} \log (1 / \delta)}{2 n}} \tag{9.36}
\end{equation*}
$$

Proof of Theorem 9.3. The first result follows after bounding the Rademacher Complexity using 9.8 and 9.9 in the inequality 9.18. The probability is obtained with a union bound.

From this result, the final computable uniform generalization bound in Equation 9.21 follows from Equation 9.17 and a union bound.

### 9.8 Appendix: Proof of lemma 9.1 and lemma 9.5

Proof of 9.1. Dividing by $n$ and taking expectation on the following inequality yields the result.

$$
\begin{equation*}
\sup _{\|A\| \leq \lambda} \sum_{i=1}^{n} \sigma_{i} x_{i}^{T} A x_{i}=\sup _{\|A\| \leq \lambda}\|A\| \sum_{i=1}^{n} \sigma_{i} x_{i}^{T}\left(\|A\|^{-1} A\right) x_{i} \leq \lambda \sup _{\|A\| \leq 1} \sum_{i=1}^{n} \sigma_{i} x_{i}^{T} A x_{i} \tag{9.37}
\end{equation*}
$$

Proof of 9.5. By the Moore-Aaronszajn Theorem, the RKHS corresponding to the kernel $K(x, y)=$ $\langle x, y\rangle^{2}$, denoted by $\mathbb{H}$, is built upon the vector space $\mathbb{H}_{0}$ corresponding to the linear span of functions of the form $K_{y}=K(\cdot, y)=\langle\cdot, y\rangle^{2}$ for $y \in \mathbb{R}^{d}$. We equip $\mathbb{H}_{0}$ with the inner product:

$$
\begin{equation*}
\left\langle\sum_{i=1}^{n} a_{i} K_{y_{i}}, \sum_{i=j}^{m} b_{j} K_{z_{j}}\right\rangle:=\sum_{i=1}^{n} \sum_{i=j}^{m} a_{i} b_{j} K\left(y_{i}, z_{j}\right) \tag{9.38}
\end{equation*}
$$

$\mathbb{H}$ is precisely the completion of $\mathbb{M}_{0}$, under the metric induced by this inner product. However, because any $K_{y}$ is a homogenous polynomial of second degree, $\mathbb{H}_{0}$ is finite-dimensional, hence closed and $\mathbb{H}=\mathbb{H}_{0}$. This shows that $\mathbb{H}$ is a subspace of the space of homogenous polynomials of second degree so $f \in \mathbb{H}$ can be represented as $f(x)=x^{T} A x$ for some symmetric matrix $A$. On the other hand, for any symmetric matrix we can represent the function $f(x)=x^{T} A x$ as

$$
\begin{equation*}
f(x)=x^{T} A x=x^{T} U S U^{T} x=\sum_{i=1}^{d} S_{i i}\left\langle x, u_{i}\right\rangle^{2}=\sum_{i=1}^{d} S_{i i} K\left(x, u_{i}\right) \tag{9.39}
\end{equation*}
$$

where $A=U S U^{T}$ is the SVD (orthogonal diagonalization) of the symmetric matrix A, and $u_{i}$ are the columns of $U$. So any homogeneous polynomial is in the linear span of the functions of the form $K_{y}=K(\cdot, y)$. We conclude that, as a set, $\mathbb{H}$ is equal to the space of homogeneous polynomials of second degree.

Now we show that its norm is equal to the Frobenius norm of the associated matrix of coefficients. Let $f(x)=x^{T} A x \in \mathbb{H}$, where $A$ is a symmetric matrix. Let $A=U S U^{T}$ be the SVD of $A$.

We know that $f=\sum_{i=1}^{d} S_{i i} K_{u_{i}}$ so that by definition of the RKHS norm

$$
\begin{align*}
\|f\|_{\mathbb{H}}^{2} & =\left\langle\sum_{i=1}^{d} S_{i i} K_{u_{i}}, \sum_{i=1}^{d} S_{i i} K_{u_{i}}\right\rangle_{\mathbb{H}}=\sum_{i=1}^{d} \sum_{j=1}^{d} S_{i i} S_{j j}\left\langle K_{u_{i}}, K_{u_{j}}\right\rangle \\
& =\sum_{i=1}^{d} \sum_{j=1}^{d} S_{i i} S_{j j} K\left(u_{i}, u_{j}\right)=\sum_{i=1}^{d} \sum_{j=1}^{d} S_{i i} S_{j j}\left\langle u_{i}, u_{j}\right\rangle^{2}=\sum_{i=1}^{d} S_{i i}^{2}=\|A\|_{F}^{2} \tag{9.40}
\end{align*}
$$

### 9.9 Appendix: Additional experiments

### 9.9.1 A synthetic experiment maximizing the train/test loss gap

The results derived in corollary 9.2 pertain to the worst case generalization gap. This worst case gap is not guaranteed to be attained by the function found through empirical loss minimization. Therefore, in order to better test our results, we try to find the function that attains this worst case generalization gap.

The experiment we propose consists of computing the quadratic function $f$ such that

$$
\mathbb{E}[\ell(f(\mathbf{x}, y))]-\frac{1}{n} \sum_{i=1}^{n} \ell\left(f\left(\mathbf{x}_{i}, y_{i}\right)\right)
$$

is maximized. The expectation in the term above being generally difficult to compute in closed form, we replace the objective by an empirical estimate

$$
\frac{1}{n_{\text {test }}} \sum_{k=1}^{n_{\text {test }}} \ell\left(f\left(\mathbf{x}_{k}, y_{k}\right)\right)-\frac{1}{n_{\text {train }}} \sum_{i=1}^{n_{\text {train }}} \ell\left(f\left(\mathbf{x}_{i}, y_{i}\right)\right)
$$

We expect this quantity to more faithfully track the upperbounds derived in corollary 9.2. Indeed, what we evaluate here is the capacity of the class constraining $f$ to attain a low loss on $n_{\text {train }}$ samples while attaining a very high one on $n_{\text {test }}$ samples.

To illustrate that Frobenius and nuclear norm constrained classifiers exhibit different behaviors depending on the intrinsic dimension of the data, we compute the result of the maximization procedure on isotropic distributions on one hand and on anisotropic on the other. We set the radius $\lambda=1$ for both Nuclear and Frobenius norm constrained classifiers. We then observe the evolution of these quantities as the dimension increases.

## Generating Isotropic Data.

To generate isotropic data satisfying our assumptions, we sample a standard Gaussian random
vector in $\mathbb{R}^{d}$ and normalize it as follows :

$$
\mathbf{x}_{i}=\sqrt{d} \times \frac{z_{i}}{\left\|z_{i}\right\|} \quad \text { where } z_{i} \sim \mathscr{N}\left(0, I_{d}\right)
$$

The $z_{i}$ 's are sampled independently. The procedure generates i.i.d samples $\mathbf{x}_{i}$ that are uniformly distributed on the sphere of radius $\sqrt{d}$.

## Generating Anistropic Data.

To generate anisotropic data, we first generate isotropic data features by uniformly sampling points on the sphere of radius $\sqrt{d}$ as described previously. We then transform the samples into more anisotropic ones.

The transform consists of multiplying each feature vector coordinate-wise by a vector $\alpha^{(s)}$ defined as $\alpha_{i}^{(s)}=C_{s} \frac{1}{i^{s}}, i=1, \ldots, d$, for $s \in[0,1]$ where $C_{s}$ is chosen such that $\left\|\alpha^{(s)}\right\|_{2}=\sqrt{d}$. That, way, $s=0$ induces no change on the data, since $\alpha^{(1)}=[1,1, \ldots, 1]^{T}$, and larger $s$ implies that the data will be more squashed along the first few coordinates, inducing smaller intrinsic dimension. This transform does not affect the norm of the vectors since

$$
\mathbb{E}\left[\left\|\alpha^{(s)} \times \mathbf{x}_{i}\right\|^{2}\right]=\sum_{k=1}^{d}\left(\alpha_{k}^{(s)}\right)^{2} \mathbb{E}\left[\left[\mathbf{x}_{i}\right]_{k}^{2}\right]=\sum_{k=1}^{d}\left(\alpha_{k}^{(s)}\right)^{2}=\sqrt{d}
$$

## Generating the labels.

We test two approaches for generating the labels :

1. We generate a random matrix $A$ with i.i.d standard Gaussian coordinates and set $y_{i}=$ $\operatorname{sgn}\left(\mathbf{x}_{i}^{\top} A \mathbf{x}_{i}\right)$.
2. We set the labels randomly with $y_{i} \sim \operatorname{Bern}(0.5)$.

## The results.

We observe that the maximal gaps do indeed evolve as the theory predicts. We indeed have the nuclear norm constrained classifier on isotropic data not exhibiting the same growth as the others, indicating that the maximal generalization gap for nuclear norm constrained classifiers is indeed sensitive to isotropy.

### 9.9.2 Additional details on the isotropic normalization effects on real data

Here we give additional details on the experimental setup. As mentioned in the main text, each dataset is randomly split with an 80/20 ratio to define training and testing sets. The training set is used to compute the normalizing factors $\Sigma_{n}$ and $v_{n}$. These factors are then used to normalize the entire dataset.


Figure 9.2: Maximized Train/Test loss gaps for each dimension $d$. The results are averaged over 5 independent runs. The error bars correspond to the standard error.

The nuclear- and Frobenius norm (or SVM) classifiers each have a hyperparameter $C$ and $\lambda$ that needs to be set before training. We search for the optimal hyperparameter among the grids $[0.0000001,0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10]$ for $C$, and $[0.0001,0.001,0.01$, $0.1,1,10,50,100,500,1000,10000$ ] for $\lambda$. The best hyperparameter is determined through 4 -fold cross-validation on the training set.

The SVM classifier results are computed using scikit learn's (Buitinck et al., 2013) liblinear (Fan et al., 2008) wrapper. Details on the nuclear norm classifier are given below.

### 9.9.3 Computational details on the Nuclear Norm Constrained Classifier

The classifier solves the following optimization problem

$$
\begin{array}{lr}
\min _{A \in \mathbb{R}^{d \times d}} & \frac{1}{n} \sum_{i=1}^{n} \ell\left(\mathbf{x}_{i}^{\top} A \mathbf{x}_{i}, y_{i}\right) \\
\text { subject to } & \|A\|_{\mathrm{tr}} \leq \lambda
\end{array}
$$

using accelerated projected gradient descent. The projection step makes use of the simplex projection algorithm proposed in Duchi et al. (2008b). The loss function used is the smoothed hinge loss, and its smoothness constant is upper bounded by $L=\frac{1}{n} \sum_{i=1}^{n}\left\|\mathbf{x}_{i}\right\|_{2}^{4}$. We use $1 / L$ as the stepsize for the accelerated projected gradient descent. The gradients are computed with JAX Bradbury et al. (2018). For the real dataset experiments, additional variables $b \in \mathbb{R}^{n}$ and $c \in \mathbb{R}$ are added in order to optimize over non-homogeneous quadratic polynomials like the SVM classifier.

### 9.10 Bibliographic Note

the candidate proposed the original idea for this paper, and derived the main result (Theorem 9.1) which uses the auxiliary results Lemma 9.2 (derived by P. Rolland) and Lemma 9.3 (derived by the candidate). The candidate also derived the secondary results Lemma 9.5, Corollary 9.2 and Theorem 9.3.

## 10 Conclusions and Future Work

In this thesis, we have provided algorithms with theoretical guarantees of robustness, either trough upper bounds of the Lipschitz constant (chapters 2, 3, 5 and 9), through Adversarial Training (chapters 6 and 7) or via denoising adversarial examples (chapter 8). We have also provided algorithms with improved generalization in the presence of noisy input data (chapter 9). We summarize the conclusions and possible extensions of each chapter as follows:

Chapter 2. We have introduced a general approach for computing an upper bound on the Lipschitz constant of neural networks. This approach is based on polynomial positivity certificates and generalizes some existing methods available in the literature. We have empirically demonstrated that it can tightly upper bound such constant. The resulting optimization problems are computationally expensive but the sparsity of the network can reduce this burden.

In order to further scale such methods to larger and deeper networks, we are interested in several possible directions: $(i)$ divide-and-conquer approaches splitting the computation on sub-networks in the same spirit of Fazlyab et al. (2019a), (ii) exploiting parallel optimization algorithms leveraging the structure of the polynomials, (iii) custom optimization algorithms with low-memory costs such as Frank-wolfe-type methods for SDP (Yurtsever et al., 2019) as well as stochastic handling of constraints (Fercoq et al., 2019) and (iv), exploting the symmetries in the polynomial that arise from weight sharing in typical network architectures to further reduce the size of the problems.

Chapter 3. We presented the first algorithm for 1-path-norm regularization of Shallow Neural Networks with guarantees of convergence to stationarity. The 1-path-norm is a powerful regularizer that non only has theoretical properties of good generalization, but also induces sparsity and higher robustness to adversarial perturbations. As such, it should be considered as a possible drop-in replacement for more common regularization methods like weight decay.

Of course, contemporary network architectures used in industrial applications like NLP and computer vision are much different to shallow networks. As such, there is limited impact
from our preliminary results, but they pave the way for further extensions to deeper and more involved network architectures. Since the number of paths is potentially large, this scenario requires more sophisticated treatment. Nonetheless, a trivial extension of our approach is to divide a multi-layered network into pairs of consecutive layers, and apply our method independently on such pairs.

Chapter 4. Our results indicate that 1-path-norm regularization clearly outperforms weightdecay on robustness and generalization tasks using fully-connected networks (MLPs). On the other hand, for highly sparse architectures like CNNs the effect is more muted or negative. Indeed, 1-path-norm works better when the number of effective paths in the network is large i.e., MLPs or MLP-Mixer (Tolstikhin et al., 2021). Nevertheless, the difference between different optimizers for 1-path-norm regularization seems to be small, but appear more noticeable when the sparsity of the network is crucial, e.g., robustness against noise in the data. Hence, the use of AD for 1-path-norm regularization is not discouraged despite its lack of theory.

1-path-norm regularization is not without drawbacks. First, an initial round of tuning is required for each model architecture to obtain satisfactory parameters for the proximal approximation method. Secondly, the computational cost of the proximal step might slow down training, so a balance struck by skipping the proximal map in some iterations (Mishchenko et al., 2022) or reducing the number of iterations for the proximal approximation step. Thirdly, 1-path-norm increases exponentially with regards to network depth, which might cause numerical problems for 1-path-norm regularization of really deep neural networks. This issue requires further attention and is left for future work. Finally, it does not consistently improve the robustness in the noisy-labels task.

Chapter 5. In this work, we explore the generalization properties of the Coupled CPdecomposition (CCP) and nested coupled CP-decomposition (NCP) models that belong in the class of Polynomial Nets (PNs). We derive bounds for the Rademacher complexity and the Lipschitz constant of the CCP and the NCP models. We utilize the computed bounds as a regularization during training and we showcase that such they can improve the generalization properties of CCP models. The regularization terms have also a substantial effect on the robustness of the model, i.e., when adversarial noise is added to the test set. Along with the recent empirical results on PNs, our derived bounds can further explain the benefits and drawbacks of using PNs.

A future direction of research is to obtain generalization bounds for this class of functions using stability notions, as it has been observed that the particular optimization algorithm can induce strong properties without the need of explicit regularization. One possible drawback of our approach is that it involves the products of norms of weight matrices. Similar to the case of traditional Deep Neural Networks, such bound can be loose, as it does not take into account the interactions between neurons in different layers. It would be of benefit to try directly regularizing the norm of the product of matrices that appears as an intermediate steps in the proof of theorem 5.1. This approach is more convoluted though, as it involves a non-convex
and non-smooth regularizer.
Chapter 6. We presented a formal proof, counter examples and evidence about the real world impact of the fact that a foundational corollary of the Adversarial Training literature is in fact false. Raising awareness about an incorrect claim that has been present in the Adversarial Training literature may provide opportunities to develop improved variants of the method. Indeed, we see some improvents in an implementable algorithm that align with our theoretical arguments: DDi exploits multiple approximate solutions of the innermaximization problem, yields better updates for the parameters of the network and improves the optimization dynamics.

However, it is important to remember the limitations and opportunities for future work: our algorithm requires multiple forward-backward passes and one additional optimization problem. Reducing the overhead over the vanilla PGD method would certainly make our results truly practical.

Non-smooth activations and the use of Batch Normalization or momentum still falls outside the scope of existing theory but might achieve better performance in benchmarks. To date, this requires using precise hyperparameters and tricks like early-stopping, that have only been found to work a-posteriori through extensive trial and error. Since we observe lower decay even in such setting, future work extending the analysis to cover this case might help alleviate this cost.

Chapter 7. We rigorously studied the standard zero-sum formulation of adversarial training. We argued that the surrogate-based relaxation commonly employed to improve the tractability of this problem voids guarantees on the ultimate robustness of trained classifiers, resulting in weak adversaries and ineffective AT algorithms. This shortcoming motivated the formulation of a novel, yet natural bilevel approach to adversrial training and evaluation. In our paradigm, the adversary and defender optimize separate objectives, which constitutes a non-zero-sum game that preserves guarantees on robustness. Based on this formulation, we developed a new adversarial attack algorithm-BETA, which stands for BEst Adversarial Attack-and a concomitant AT algorithm, which we call BETA-AT. In our experiments, we showed that BETA-AT eliminates robust overfitting, which we argued is a direct result of optimizing an objective which is aligned with the goal of finding true adversarial examples. We also showed that even when early stopping based model selection is used, BETA-AT performed comparably to AT. And finally, we showed that BETA provides almost identical estimates of robustness to AutoAttack, indicating that when the adversarial objective closely matches the true objective, one need not resort to heuristics like multiple restarts, variable stopping conditions, and adaptive learning rate schedules to accurately estimate robustness.

With regard to the bilevel formulation in this paper, future directions abound. One could imagine applying this framework to other changes in the data space, including the kinds of distribution shifts that are common in fields like domain adaptation and domain generalization. A convergence analysis of BETA and an analysis of the sample complexity of BETA-AT are two
more directions that we leave for future work. The prospect of applying more sophisticated bilevel optimization algorithmic techniques to this problem is also a promising avenue for future research.

Chapter 8. We have proposed a flexible linearized ADMM algorithm for the minimization of a convex function subject to a nonlinear constraint given by a neural network corresponding to a generative model. Under mild assumptions we demonstrate a fast convergence rate to a neighborhood of a solution of its Lagrangian formulation (8.3). Empirical evaluation shows how it can handle nonsmooth terms more efficiently when compared to gradient descent and its variants.

Some avenues of research are left open which could yield faster variants of our proposed approach. First, ADMM-type algorithms admit acceleration and restart schemes with faster convergence rates in the convex case (Goldstein et al., 2014) but their adaptation to the nonlinear constraint given by a generative model is non-trivial. Secondly, adaptivity in the choice of penalty parameter $\rho$ can potentially improve the performance of the method and reduce the need for tuning (He et al., 2000).

Chapter 9. Our result shows that given a fixed regularization parameter $\lambda$, nuclear-norm constrained classifiers can take advantage of the properties of the data distribution to shave off a $\sqrt{d}$ factor from the Rademacher complexity, whereas Frobenius-norm constrained ones cannot.

We rely on two main elements to show this: Hölder's inequality in equation (9.5) which introduces the dual norms and a bound on the expected deviations $M_{k}$, which quantifies how fast the empirical moment estimate converges to the true moment matrix. Both these elements admit extensions to higher order tensors which corresponds to polynomials of higher degree. This is an avenue of research worth exploring, as having a good analysis for higher-order polynomials can shed light on the behaviour for Neural Networks, by considering polynomial approximations of the activation functions.

The extension of Hölder's inequality is immediate. The main difficulty of extending our result lies in establishing the convergence rate of the deviations $\mathbb{E}\left[\left\|\frac{1}{k} \sum_{i=1}^{k} x_{i}^{\otimes m}-\mathscr{M}\right\|_{o p}\right]$, where $\mathscr{M}=\mathbb{E}\left[\mathbf{x}^{\otimes m}\right]$ is the tensor of $m$-th moments. For matrices, the rate is obtained through the matrix Bernstein inequality. Therefore, an extension can be derived by using the recently proved tensor Bernstein inequality in Luo et al. (2019) obtained by flattening the tensors and recycling the results for matrices.

## Bibliography

Aaid, D., Noui, A., and Ouanes, M. (2017). New technique for solving univariate global optimization. Archivum Mathematicum, 53(1):19-33. 72

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. 30, 68

Agarwal, A., Negahban, S., and Wainwright, M. J. (2010). Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In Advances in Neural Information Processing Systems, pages 37-45. 190, 192, 197

Ahmadi, A. and Majumdar, A. (2019). Dsos and sdsos optimization: More tractable alternatives to sum of squares and semidefinite optimization. SIAM Journal on Applied Algebra and Geometry, 3(2):193-230. 15

Alvarez, J. M. and Salzmann, M. (2016). Learning the number of neurons in deep networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems 29, pages 2270-2278. Curran Associates, Inc. 41

Amit, Y., Fink, M., Srebro, N., and Ullman, S. (2007). Uncovering shared structures in multiclass classification. In Proceedings of the 24th international conference on Machine learning, pages 17-24. 224

Anil, C., Lucas, J., and Grosse, R. (2019). Sorting out Lipschitz function approximation. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 291-301, Long Beach, California, USA. PMLR. 12

Anil, C., Lucas, J., and Grosse, R. B. (2018). Sorting out lipschitz function approximation. In International Conference on Machine Learning. 70

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2019). Invariant risk minimization. arXiv preprint arXiv:1907.02893. 165

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 214-223, International Convention Centre, Sydney, Australia. PMLR. 12, 193

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdykalojasiewicz inequality. Mathematics of Operations Research, 35(2):438-457. 46

Attouch, H., Bolte, J., and Svaiter, B. F. (2011). Convergence of descent methods for semialgebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods. Mathematical Programming, 137(1-2):91-129. 46

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. Found. Trends Mach. Learn., 4(1):1-106. 31, 42

Bai, Y. and Lee, J. D. (2020). Beyond linearization: On quadratic and higher-order approximation of wide neural networks. In International Conference on Learning Representations. 222, 224

Bard, J. F. (2013). Practical bilevel optimization: algorithms and applications, volume 30. Springer Science \& Business Media. 8, 166, 177

Barron, A. R. and Klusowski, J. M. (2019). Complexity, Statistical Risk, and Metric Entropy of Deep Nets Using Total Path Variation. arXiv e-prints, page arXiv:1902.00800. 68

Barron, J. T. (2017). Continuously differentiable exponential linear units. 155, 159
Bartlett, P., Foster, D. J., and Telgarsky, M. (2017a). Spectrally-normalized margin bounds for neural networks. 6, 92, 104

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017b). Spectrally-normalized margin bounds for neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems 30, pages 6240-6249. Curran Associates, Inc. 11, 30, 41

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. Journal of the American Statistical Association, 101(473):138-156. 166, 168

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research, 3(Nov):463-482. 94

Bartlett, P. L. and Mendelson, S. (2003). Rademacher and gaussian complexities: Risk bounds and structural results. J. Mach. Learn. Res., 3(null):463-482. 225

Bartlett, R., Morse, A., Stanton, R., and Wallace, N. (2019). Consumer-Lending Discrimination in the FinTech Era. NBER Working Papers 25943, National Bureau of Economic Research, Inc. 2

Bauschke, H. and Combettes, P. L. (2011). Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer. 71

Beck, A. (2014). Introduction to nonlinear optimization, volume 19 of MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. 34, 35, 48, 49

Beck, A. (2017). First-Order Methods in Optimization, volume 25. SIAM. 32
Benz, P., Zhang, C., and Kweon, I. S. (2021). Batch normalization increases adversarial vulnerability and decreases adversarial transferability: A non-robust feature perspective. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7818-7827. 159

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In Advances in Neural Information Processing Systems 30, pages 908-918. Curran Associates, Inc. 12

Bertsekas, D. P. (1976). On penalty and multiplier methods for constrained minimization. SIAM Journal on Control and Optimization, 14(2):216-235. 186

Bertsekas, D. P. (1999). Nonlinear programming. Athena scientific Belmont. 36
Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. (2013a). Evasion attacks against machine learning at test time. In ECML/PKDD. 3, 29, 165

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. (2013b). Evasion attacks against machine learning at test time. In ECML/PKKD. 165, 168

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013c). Evasion attacks against machine learning at test time. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13, pages 387-402. Springer. 168

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389. 165, 168

Birgin, E. G., Gardenghi, J., Martínez, J. M., Santos, S., and Toint, P. L. (2016). Evaluation complexity for nonlinear constrained optimization using unscaled kkt conditions and high-order models. SIAM Journal on Optimization, 26(2):951-967. 189, 191

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. Advances in neural information processing systems, 31. 159

Bolte, J. and Pauwels, E. (2019). Conservative set valued fields, automatic differentiation, stochastic gradient method and deep learning. 30, 68

Bolte, J., Sabach, S., and Teboulle, M. (2013). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Mathematical Programming, 146(1-2):459-494. 30, 33, 34, 46, 68, 71, 76

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed Sensing using Generative Models. arXiv:1703.03208 [cs, math, stat]. arXiv: 1703.03208. 183, 184, 185, 187, 191, 193

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, Advances in Neural Information Processing Systems, volume 20. Curran Associates, Inc. 78

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. Journal of Machine Learning Research, 2:499-526. 104

Boyd, S. (2014). Subgradient methods. https://web.stanford.edu/class/ee364b/lectures/ subgrad_method_notes.pdf. [Online; accessed 27-September-2022]. 147

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1):1-122. 185

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs. 241

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2014). (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):1-36. 91

Brock, A., De, S., Smith, S. L., and Simonyan, K. (2021). High-performance large-scale image recognition without normalization. In International Conference on Machine Learning, pages 1059-1071. PMLR. 159

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In International Conference on Learning Representations. 183, 188

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 1877-1901. Curran Associates, Inc. 1

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pages 108-122. 241

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. (2019). On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705. 166

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In 2017 ieee symposium on security and privacy (sp), pages 39-57. Ieee. 165, 168

Cartis, C., Gould, N. I., and Toint, P. L. (2018). Optimality of orders one to three and beyond: characterization and evaluation complexity in constrained nonconvex optimization. Journal of Complexity. 189, 191

Caruana, R. (1997). Multitask Learning. Machine Learning, 28(1):41-75. 224
Chan, A., Tay, Y., Ong, Y. S., and Fu, J. (2020). Jacobian adversarially regularized networks for robustness. ICLR. 166

Chen, L. and Gu, Y. (2014). The convergence guarantees of a non-convex approach for sparse recovery. IEEE Transactions on Signal Processing, 62(15):3754-3767. 192

Chen, L., Min, Y., Zhang, M., and Karbasi, A. (2020a). More data can expand the generalization gap between adversarially robust and standard models. In International Conference on Machine Learning, pages 1670-1680. PMLR. 166

Chen, T., Lasserre, J. B., Magron, V., and Pauwels, E. (2020b). Semialgebraic optimization for lipschitz constants of relu networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 19189-19200. Curran Associates, Inc. 5, 81

Chen, T., Lasserre, J. B., Magron, V., and Pauwels, E. (2021a). Semialgebraic representation of monotone deep equilibrium models and applications to certification. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pages 27146-27159. 5

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. (2021b). Robust overfitting may be mitigated by properly learned smoothening. In International Conference on Learning Representations. 176

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems 29, pages 2172-2180. Curran Associates, Inc. 189

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. 31

Chrysos, G., Moschoglou, S., Bouritsas, G., Panagakis, Y., Deng, J., and Zafeiriou, S. (2020). $\pi-$ nets: Deep polynomial neural networks. In Conference on Computer Vision and Pattern Recognition (CVPR). 6, 91, 92, 94, 95, 110, 112

Chrysos, G. G. and Panagakis, Y. (2020). NAPS: Non-adversarial polynomial synthesis. Pattern Recognit. Lett., 140:318-324. 6, 91, 92, 94

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017a). Parseval networks: Improving robustness to adversarial examples. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 854-863, International Convention Centre, Sydney, Australia. PMLR. 5, 6, 12

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017b). Parseval networks: Improving robustness to adversarial examples. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 854-863, International Convention Centre, Sydney, Australia. PMLR. 29, 42, 43, 70

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017c). Parseval networks: Improving robustness to adversarial examples. In International Conference on Machine Learning, pages 854-863. PMLR. 92

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018a). Deep learning for classical japanese literature. 43

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018b). Deep learning for classical japanese literature. arXiv preprint arXiv:1812.01718. 131

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). arXiv e-prints, page arXiv:1511.07289. 13, 30

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289. 188, 193

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1908.06571. 99, 131

Combettes, P. L. and Pesquet, J.-C. (2019). Lipschitz Certificates for Neural Network Structures Driven by Averaged Activation Operators. arXiv e-prints, page arXiv:1903.01014. 20

Condat, L. (2016). Fast projection onto the simplex and the 11 ball. Math. Program., 158(1-2):575-585. 42, 43

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. (2020a). Robustbench: a standardized adversarial robustness benchmark. arXiv preprint arXiv:2010.09670. 7, 145, 166, 175

Croce, F. and Hein, M. (2020a). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Proceedings of the 37th International Conference on Machine Learning. 132

Croce, F. and Hein, M. (2020b). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In International conference on machine learning, pages 2206-2216. PMLR. 166, 169, 170, 174, 175, 177

Croce, F., Rauber, J., and Hein, M. (2020b). Scaling up the randomized gradient-free adversarial attack reveals overestimation of robustness using established attacks. International Journal of Computer Vision, 128:1028-1046. 169, 170, 177

Cvetkovski, Z. (2012). Hölder's Inequality, Minkowski's Inequality and Their Variants, pages 95-105. Springer Berlin Heidelberg. 108

Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., and Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, 5(6):e01802. 1

Danskin, J. M. (1966). The theory of max-min, with applications. SIAM Journal on Applied Mathematics, 14(4):641-664. 146, 148, 149

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. (2018). Training GANs with optimism. In International Conference on Learning Representations. 183

Davis, D. and Drusvyatskiy, D. (2019). Stochastic model-based minimization of weakly convex functions. SIAM Journal on Optimization, 29(1):207-239. 71

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248-255. Ieee. 1

Dhar, M., Grover, A., and Ermon, S. (2018). Modeling sparse deviations for compressed sensing using generative models. In Dy, J. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 1214-1223, Stockholmsmässan, Stockholm Sweden. PMLR. 183

Dobriban, E., Hassani, H., Hong, D., and Robey, A. (2020). Provable tradeoffs in adversarially robust classification. arXiv preprint arXiv:2006.05161. 166

Donahue, C., McAuley, J., and Puckette, M. (2019). Adversarial audio synthesis. In Proceedings of the 7th International Conference on Learning Representations. 183

Dong, Y., Deng, Z., Pang, T., Zhu, J., and Su, H. (2020). Adversarial distributional training for robust deep learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 8270-8283. Curran Associates, Inc. 146

Dong, Y., Xu, K., Yang, X., Pang, T., Deng, Z., Su, H., and Zhu, J. (2022). Exploring memorization in adversarial training. In International Conference on Learning Representations. 176

Du, S. and Lee, J. (2018). On the power of over-parametrization in neural networks with quadratic activation. In Dy, J. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 1329-1338. PMLR. 9, 222, 224

Dubey, S. R., Singh, S. K., and Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing, 503:92-108. 159

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7). 34

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008a). Efficient projections onto the 11-ball for learning in high dimensions. In Proceedings of the 25th international conference on Machine learning, pages 272-279. 97, 152

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008b). Efficient projections onto the l1-ball for learning in high dimensions. In Proceedings of the 25th International Conference on Machine Learning, ICML 2008, page 272-279, New York, NY, USA. Association for Computing Machinery. 42, 43, 194, 241

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., and Garcia, R. (2001). Incorporating second-order functional knowledge for better option pricing. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, Advances in Neural Information Processing Systems 13, pages 472-478. MIT Press. 188

Dziugaite, G. K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., Wang, L., Mitliagkas, I., and Roy, D. M. (2020). In search of robust measures of generalization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 11723-11733. Curran Associates, Inc. 6, 67, 68

E, W., Ma, C., and Wang, Q. (2020). Rademacher complexity and the generalization error of residual networks. Communications in Mathematical Sciences, 18(6):1755-1774. Publisher Copyright: © 2020. International Press. All rights reserved. Copyright: Copyright 2020 Elsevier B.V., All rights reserved. 68

Eftekhari, A. and Wakin, M. B. (2015). New analysis of manifold embeddings and signal recovery from compressive measurements. Applied and Computational Harmonic Analysis, 39(1):67-109. 191

Eftekhari, A. and Wakin, M. B. (2017). What happens to a manifold under a bi-lipschitz map? Discrete \& Computational Geometry, 57(3):641-673. 191

Eldar, Y. C. and Kutyniok, G. (2012). Compressed sensing: theory and applications. Cambridge university press. 31

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). GANSynth: Adversarial neural audio synthesis. In International Conference on Learning Representations. 183

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. 131

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. (2019). Robustness (python library). 155

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018a). Robust physical-world attacks on deep learning visual classification. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1625-1634. 3

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2018b). Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1625-1634. 165

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. J. Mach. Learn. Res., 9:1871-1874. 241

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. (2019a). Efficient and accurate estimation of lipschitz constants for deep neural networks. In Advances in Neural Information Processing Systems 32, pages 11423-11434. Curran Associates, Inc. 12, 20, 21, 30, 243

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. J. (2019b). Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. Curran Associates Inc., Red Hook, NY, USA. 92, 105

Federer, H. (2014). Geometric measure theory. Springer. 108
Fercoq, O., Alacaoglu, A., Necoara, I., and Cevher, V. (2019). Almost surely constrained convex optimization. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 1910-1919, Long Beach, California, USA. PMLR. 243

Finlay, C., Calder, J., Abbasi, B., and Oberman, A. (2018). Lipschitz regularized deep neural networks generalize and are adversarially robust. arXiv preprint arXiv:1808.09540. 166

Flores-Bazán, F., Flores-Bazán, F., and Vera, C. (2012). A complete characterization of strong duality in nonconvex optimization with a single constraint. Journal of Global Optimization, 53(2):185-201. 189, 191

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2021). Sharpness-aware minimization for efficiently improving generalization. In International Conference on Learning Representations. 68, 166

Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In International Conference on Learning Representations. 16

Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., and Walther, A. (2017). Predictably unequal? the effects of machine learning on credit markets. CEPR Discussion Papers 12448, C.E.P.R. Discussion Papers. 2

Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Computers \& Mathematics with Applications, 2:17-40. 185

Garling, D. J. H. (2014). A Course in Mathematical Analysis, volume 1. Cambridge University Press. 201

Ghaddar, B., Marecek, J., and Mevissen, M. (2015). Optimal power flow as a polynomial optimization problem. IEEE Transactions on Power Systems, 31(1):539-546. 19

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. (2020). When do neural networks outperform kernel methods? 9, 222

Gidel, G., Hemmat, R. A., Pezeshki, M., Priol, R. L., Huang, G., Lacoste-Julien, S., and Mitliagkas, I. (2019). Negative momentum for improved game dynamics. In Chaudhuri, K. and Sugiyama, M., editors, Proceedings of Machine Learning Research, volume 89 of Proceedings of Machine Learning Research, pages 1802-1811. PMLR. 183

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Balcan, M. F. and Weinberger, K. Q., editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 201-210, New York, New York, USA. PMLR. 92

Giryes, R., Eldar, Y. C., Bronstein, A. M., and Sapiro, G. (2016). Tradeoffs between convergence speed and reconstruction accuracy in inverse problems. arXiv preprint arXiv:1605.09232. 192

Glowinski, R. and Marroco, A. (1975). Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique, 9(R2):41-76. 185

Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R. (2014). Fast alternating direction optimization methods. SIAM Journal on Imaging Sciences, 7(3):1588-1623. 246

Golowich, N., Rakhlin, A., and Shamir, O. (2018). Size-independent sample complexity of neural networks. 92

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (NeurIPS). 8

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. ArXiv e-prints. 183, 184

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y., editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 92, 145, 165, 166, 169, 174

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2021). Regularisation of neural networks by enforcing lipschitz continuity. Mach. Learn., 110(2):393-416. 70

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. (2021). Improving robustness using generated data. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems. 176

Gowal, S., Uesato, J., Qin, C., Huang, P.-S., Mann, T., and Kohli, P. (2019). An alternative surrogate loss for pgd-based adversarial testing. arXiv preprint arXiv:1910.09338. 170, 171, 177

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In Advances in neural information processing systems (NeurIPS), pages 5767-5777. 12, 188, 193

Haeffele, B., Young, E., and Vidal, R. (2014). Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In Xing, E. P. and Jebara, T., editors, Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 2007-2015, Bejing, China. PMLR. 75

Haeffele, B. D. and Vidal, R. (2015). Global optimality in tensor factorization, deep learning, and beyond. ArXiv, abs/1506.07540. 75, 76

Haeffele, B. D. and Vidal, R. (2020). Structured low-rank matrix factorization: Global optimality, algorithms, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(6):1468-1482. 75

Hallak, N., Mertikopoulos, P., and Cevher, V. (2021). Regret minimization in stochastic nonconvex learning via a proximal-gradient approach. In To appear in Proceedings of the 38th International Conference on Machine Learning. PMLR. 71

Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149. 23

Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. International Conference on Learning Representations, abs/1510.00149. 31

Hand, P., Leong, O., and Voroninski, V. (2018). Phase retrieval under a generative prior. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems 31, pages 9154-9164. Curran Associates, Inc. 192

Hand, P. and Voroninski, V. (2017). Global guarantees for enforcing deep generative priors by empirical risk. arXiv preprint arXiv:1705.07576. 188, 189, 192

Handelman, D. (1988). Representing polynomials by positive linear functions on compact convex polyhedra. Pacific J. Math., 132(1):35-62. 15

Hanson, S. J. and Pratt, L. Y. (1989). Comparing biases for minimal network construction with back-propagation. In Advances in Neural Information Processing Systems 1, pages 177-185. Morgan-Kaufmann. 18

He, B., Yang, H., and Wang, S. (2000). Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. Journal of Optimization Theory and applications, 106(2):337-356. 186, 246

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv e-prints, page arXiv:1512.03385. 193

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Conference on Computer Vision and Pattern Recognition, pages 770-778. 13, 174

Heckel, R., Huang, W., Hand, P., and Voroninski, V. (2019). Deep denoising: Rate-optimal recovery of structured signals with a deep prior. 189, 192

Hegde, C. (2018). Algorithmic Aspects of Inverse Problems Using Generative Models. ArXiv e-prints. 184, 192

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. (2021). The many faces of robustness: A critical analysis of out-of-distribution generalization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 8340-8349. 165

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In International Conference on Learning Representations. 165

Hesamifard, E., Takabi, H., Ghasemi, M., and Wright, R. N. (2018). Privacy-preserving machine learning as a service. Proc. Priv. Enhancing Technol., 2018(3):123-142. 92

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems 30, pages 6626-6637. Curran Associates, Inc. 188

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 6840-6851. Curran Associates, Inc. 8

Hoffman, J., Roberts, D. A., and Yaida, S. (2019). Robust learning with jacobian regularization. 99, 166

Hsieh, Y.-P., Liu, C., and Cevher, V. (2018). Finding Mixed Nash Equilibria of Generative Adversarial Networks. arXiv e-prints, page arXiv:1811.02002. 183

Huang, R., Xu, B., Schuurmans, D., and Szepesvari, C. (2015). Learning with a strong adversary. ArXiv, abs/1511.03034. 165, 169

Huang, S., Lu, Z., Deb, K., and Naresh Boddeti, V. (2022). Revisiting Residual Networks for Adversarial Robustness: An Architectural Perspective. arXiv e-prints, page arXiv:2212.11005. 176

Huster, T., Chiang, C.-Y. J., and Chadha, R. (2018). Limitations of the Lipschitz constant as a defense against adversarial examples. arXiv e-prints, page arXiv:1807.09705. 12

Ilyas, A., Jalal, A., Asteri, E., Daskalakis, C., and Dimakis, A. G. (2017). The Robust Manifold Defense: Adversarial Training using Generative Models. arXiv e-prints, page arXiv:1712.09196. 8, 183, 187, 194

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448-456. PMLR. 155, 159

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: convergence and generalization in neural networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 8580-8589. 222

Javanmard, A., Soltanolkotabi, M., and Hassani, H. (2020). Precise tradeoffs in adversarial training for linear regression. In Conference on Learning Theory, pages 2034-2078. PMLR. 166

Jayakumar, S. M., Czarnecki, W. M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., and Pascanu, R. (2020). Multiplicative interactions and where to find them. In International Conference on Learning Representations. 6, 91, 222

Jiang, Y., Foret, P., Yak, S., Roy, D. M., Mobahi, H., Karolina Dziugaite, G., Bengio, S., Gunasekar, S., Guyon, I., and Neyshabur, B. (2020). NeurIPS 2020 Competition: Predicting Generalization in Deep Learning. arXiv e-prints, page arXiv:2012.07976. 67

Jiang, Y., Neyshabur, B., Krishnan, D., Mobahi, H., and Bengio, S. (2020). Fantastic generalization measures and where to find them. In International Conference on Learning Representations. 6, 30, 67, 68, 221

Jin, M. and Lavaei, J. (2018). Stability-certified reinforcement learning: A control-theoretic perspective. arXiv e-prints, page arXiv:1810.11505. 12, 20, 21

Kakade, S. M. and Lee, J. D. (2018). Provably correct automatic sub-differentiation for qualified programs. In Advances in Neural Information Processing Systems 31, pages 7125-7135. Curran Associates, Inc. 30, 68

Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. (2012). Regularization techniques for learning with matrices. Journal of Machine Learning Research, 13(Jun):1865-1890. 222, 223, 228, 229

Kannan, H., Kurakin, A., and Goodfellow, I. (2018). Adversarial logit pairing. arXiv preprint arXiv:1803.06373. 166

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Conference on Computer Vision and Pattern Recognition (CVPR). 91

Kemelmacher-Shlizerman, I., Seitz, S. M., Miller, D., and Brossard, E. (2016). The megaface benchmark: 1 million faces for recognition at scale. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 4873-4882. 7, 91

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On largebatch training for deep learning: Generalization gap and sharp minima. In International Conference on Learning Representations. 68

Kim, Y., Kim, M., and Kim, G. (2018). Memorization precedes generation: Learning unsupervised GANs with memory networks. In International Conference on Learning Representations. 188

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv e-prints, page arXiv:1412.6980. 72, 73, 174, 184, 194

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. arXive-prints, page arXiv:1312.6114. 183, 184

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. (2023). Segment Anything. arXiv e-prints, page arXiv:2304.02643. 1

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. (2021). Wilds: A benchmark of in-the-wild distribution shifts. In International Conference on Machine Learning, pages 5637-5664. PMLR. 165

Kohler, J., Daneshmand, H., Lucchi, A., Hofmann, T., Zhou, M., and Neymeyr, K. (2019). Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In The 22nd International Conference on Artificial Intelligence and Statistics, pages 806-815. PMLR. 159

Kojima, M., Kim, S., and Waki, H. (2005). Sparsity in sums of squares of polynomials. Mathematical Programming, 103(1):45-62. 19

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. SIAM review, 51(3):455-500. 94

Koltchinskii, V. and Panchenko, D. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. Ann. Statist., 30(1):1-50. 225

Krivine, J.-L. (1964). Anneaux préordonnés. Journal d'analyse mathématique, 12:p. 307-326. 15

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report. 69, 174

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. online: http://www. cs. toronto. edu/kriz/cifar. html, 55. 99, 131

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc. 1

Kuhn, D., Mohajerin Esfahani, P., Nguyen, V. A., and Shafieezadeh-Abadeh, S. (2019). Wasserstein Distributionally Robust Optimization: Theory and Applications in Machine Learning. arXiv e-prints, page arXiv:1908.08729. 5

Kumar, A., Sattigeri, P., and Fletcher, T. (2017). Semi-supervised learning with gans: Manifold invariance with improved inference. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, Advances in Neural Information Processing Systems 30, pages 5534-5544. Curran Associates, Inc. 188

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2017). Adversarial machine learning at scale. In International Conference on Learning Representations. 152, 165, 169

Köchling, A. and Wehner, M. C. (2020). Discriminated by an algorithm: a systematic review of discrimination and fairness by algorithmic decision-making in the context of HR recruitment and HR development. Business Research, 13(3):795-848. 2

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of frank-wolfe optimization variants. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, page 496-504, Cambridge, MA, USA. MIT Press. 152

Lasserre, J. B. (2000). Convergent lmi relaxations for nonconvex quadratic programs. In Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187), volume 5, pages 5041-5046 vol.5. 20

Lasserre, J. B. (2006). Convergent sdp-relaxations in polynomial optimization with sparsity. SIAM Journal on Optimization, 17(3):822-843. 19, 26

Lasserre, J. B. (2015). An Introduction to Polynomial and Semi-Algebraic Optimization. Cambridge Texts in Applied Mathematics. Cambridge University Press. 5, 12, 14, 15

Latorre, F., Dadi, L. T., Rolland, P., and Cevher, V. (2021). The effect of the intrinsic dimension on the generalization of quadratic classifiers. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems, volume 34, pages 21138-21149. Curran Associates, Inc. iii, vi, 8, 9

Latorre, F., eftekhari, A., and Cevher, V. (2019). Fast and provable admm for learning with generative priors. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc. iii, vi, 8

Latorre, F., Krawczuk, I., Dadi, L. T., Pethick, T., and Cevher, V. (2023). Finding actual descent directions for adversarial training. In The Eleventh International Conference on Learning Representations. iii, v, 7, 166

Latorre, F., Rolland, P., and Cevher, V. (2020a). Lipschitz constant estimation of neural networks via sparse polynomial optimization. In International Conference on Learning Representations. iii, v, 5, 92, 105

Latorre, F., Rolland, P., and Cevher, V. (2020b). Lipschitz constant estimation of neural networks via sparse polynomial optimization. In International Conference on Learning Representations. 30, 44

Latorre, F., Rolland, P., Hallak, N., and Cevher, V. (2020c). Efficient proximal mapping of the 1-path-norm of shallow networks. In III, H. D. and Singh, A., editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 5651-5661. PMLR. iii, v, 6, 68, 69, 70, 71, 75, 76, 77, 80, 81

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324. 12

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324. 131

LeCun, Y. and Cortes, C. (2010a). MNIST handwritten digit database. 43
LeCun, Y. and Cortes, C. (2010b). MNIST handwritten digit database. 193

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. (2019). Certified robustness to adversarial examples with differential privacy. In 2019 IEEE Symposium on Security and Privacy (SP), pages 656-672. 5

Lee, S., Lee, H., and Yoon, S. (2020). Adversarial vertex mixup: Toward better adversarially robust generalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 176

Lemhadri, I., Ruan, F., and Tibshirani, R. (2019). A neural network with feature sparsity. arXiv e-prints, page arXiv:1907.12207. 41

Li, Y., Xu, X., Xiao, J., Li, S., and Shen, H. T. (2021). Adaptive square attack: Fooling autonomous cars with adversarial traffic signs. IEEE Internet of Things Journal, 8(8):6337-6347. 3

Liang, B., Su, M., You, W., Shi, W., and Yang, G. (2016). Cracking classifiers for evasion: A case study on the google's phishing pages filter. In Proceedings of the 25th International Conference on World Wide Web, WWW '16, page 345-356, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. 3

Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., and Zhu, J. (2018). Defense against adversarial attacks using high-level representation guided denoiser. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 141, 143

Lipton, Z. C. and Tripathi, S. (2017). Precise Recovery of Latent Vectors from Generative Adversarial Networks. arXiv:1702.04782 [cs, stat]. arXiv: 1702.04782. 183, 188

Liu, H.-T. D., Williams, F., Jacobson, A., Fidler, S., and Litany, O. (2022). Learning smooth neural functions via lipschitz regularization. In ACM SIGGRAPH 2022 Conference Proceedings, SIGGRAPH '22, New York, NY, USA. Association for Computing Machinery. 5

Liu, Q., Shen, X., and Gu, Y. (2017). Linearized admm for non-convex non-smooth optimization with convergence analysis. arXiv preprint arXiv:1705.02502. 192

Liu, S., Lu, S., Chen, X., Feng, Y., Xu, K., Al-Dujaili, A., Hong, M., and O’Reilly, U.-M. (2020). Min-max optimization without gradients: Convergence and applications to black-box evasion and poisoning attacks. In III, H. D. and Singh, A., editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 6282-6293. PMLR. 157

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV). 193

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are gans created equal? a large-scale study. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems 31, pages 700-709. Curran Associates, Inc. 183

Luo, Z., Qi, L., and Toint, P. L. (2019). Bernstein concentration inequalities for tensors via einstein products. arXiv preprint arXiv:1902.03056. 246

Lyche, T. (2020). Numerical Linear Algebra and Matrix Factorizations. Springer, Cham. 109
Ma, F., Ayaz, U., and Karaman, S. (2018). Invertibility of convolutional generative networks from partial measurements. In Advances in Neural Information Processing Systems, pages 9651-9660. 188, 189

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018a). Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations. ix, 145, 146, 147, 148, 149, 152, 157, 165, 166, 169, 174, 177

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018b). Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations. 7, 8, 29, 44, 64, 80, 178, 194

Maini, P., Wong, E., and Kolter, Z. (2020). Adversarial robustness against the union of multiple perturbation models. In III, H. D. and Singh, A., editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 6640-6650. PMLR. 157

Mannelli, S. S., Vanden-Eijnden, E., and Zdeborová, L. (2020). Optimization and generalization of shallow neural networks with quadratic activation functions. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA. Curran Associates Inc. 9

Manoel, A., Krzakala, F., Mézard, M., and Zdeborová, L. (2017). Multi-layer generalized linear estimation. In 2017 IEEE International Symposium on Information Theory (ISIT), pages 2098-2102. IEEE. 193

Mendelson, S. and Zhivotovskiy, N. (2018). Robust covariance estimation under $L_{4}-L_{2}$ norm equivalence. arXiv e-prints, page arXiv:1809.10462. 229

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for GANs do actually converge? In Dy, J. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 3481-3490, Stockholmsmässan, Stockholm Sweden. PMLR. 183

Metel, M. and Takeda, A. (2019). Simple stochastic gradient methods for non-smooth nonconvex regularized optimization. In International Conference on Machine Learning, pages 4537-4545. 34, 71

Mianjy, P. and Arora, R. (2023). Robustness guarantees for adversarially trained neural networks. 152, 177, 178

Mishchenko, K., Malinovsky, G., Stich, S., and Richtarik, P. (2022). ProxSkip: Yes! Local gradient steps provably lead to communication acceleration! Finally! In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 15750-15769. PMLR. 244

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018a). Spectral normalization for generative adversarial networks. In International Conference on Learning Representations. 12, 30, 42

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018b). Spectral normalization for generative adversarial networks. In International Conference on Learning Representations. 92

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018a). Foundations of Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2 edition. 96, 104

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018b). Foundations of Machine Learning. The MIT Press, 2nd edition. 229, 231

Mohri, M., Rostamizadeh, A., Talwalkar, A., and Bach, F (2018c). Foundations of Machine Learning. MIT Press. 185, 192, 195, 196, 199, 200

Mosbach, M., Andriushchenko, M., Trost, T., Hein, M., and Klakow, D. (2018). Logit pairing methods can fool gradient-based attacks. arXiv preprint arXiv:1810.12042. 169

Muthukumar, R. and Sulam, J. (2022). Adversarial robustness of sparse local Lipschitz predictors. arXiv e-prints, page arXiv:2202.13216. 22

Negahban, S. N., Ravikumar, P., Wainwright, M. J., Yu, B., et al. (2012). A unified framework for high-dimensional analysis of $m$-estimators with decomposable regularizers. Statistical Science, 27(4):538-557. 198

Nesterov, Y. (2013a). Gradient methods for minimizing composite functions. Mathematical Programming, 140(1):125-161. 184

Nesterov, Y. (2013b). Introductory Lectures on Convex Optimization: A Basic Course. Applied Optimization. Springer US. 191

Neyshabur, B. (2017). Implicit regularization in deep learning. arXiv preprint arXiv:1709.01953. 104

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017). A pac-bayesian approach to spectrallynormalized margin bounds for neural networks. arXiv preprint arXiv:1707.09564. 92, 104

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. (2019a). The role of overparametrization in generalization of neural networks. In International Conference on Learning Representations. 68

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. (2019b). The role of overparametrization in generalization of neural networks. In International Conference on Learning Representations. 152

Neyshabur, B., Salakhutdinov, R., and Srebro, N. (2015a). Path-sgd: Path-normalized optimization in deep neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS'15, page 2422-2430, Cambridge, MA, USA. MIT Press. 69

Neyshabur, B., Tomioka, R., and Srebro, N. (2015b). In search of the real inductive bias: On the role of implicit regularization in deep learning. In ICLR (Workshop). 68

Neyshabur, B., Tomioka, R., and Srebro, N. (2015c). Norm-based capacity control in neural networks. In Proceedings of The 28th Conference on Learning Theory, volume 40 of Proceedings of Machine Learning Research, pages 1376-1401, Paris, France. PMLR. 6, 30, 31, 32, 41, 44, 67, 68, 70

Neyshabur, B., Tomioka, R., and Srebro, N. (2015d). Norm-based capacity control in neural networks. In Conference on Learning Theory, pages 1376-1401. PMLR. 92, 105
nicolas le roux (2017). tighter bounds lead to improved classifiers. In international conference on learning representations. 166

Nilsson, N. J. (1982). Principles of artificial intelligence. Springer Science \& Business Media. 1
Nouiehed, M., Sanjabi, M., Huang, T., Lee, J. D., and Razaviyayn, M. (2019). Solving a class of non-convex min-max games using iterative first order methods. In Wallach, H., Larochelle, H., Beygelzimer, A., d’Alche Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc. 157

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. In International Conference on Learning Representations. 68

Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M.-E., Ruggieri, S., Turini, F., Papadopoulos, S., Krasanakis, E., Kompatsiaris, I., Kinder-Kurlanda, K., Wagner, C., Karimi, F., Fernández, M., Alani, H., Berendt, B., Kruegel, T., Heinze, C., Broelemann, K., Kasneci, G., Tiropanis, T., and Staab, S. (2020). Bias in data-driven artificial intelligence systems - an introductory survey. Wiley Interdiscip. Rev. Data Min. Knowl. Discov., 10(3). 2

Ochs, P., Dosovitskiy, A., Brox, T., and Pock, T. (2015). On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. SIAM Journal on Imaging Sciences, 8(1):331-372. 42

Orabona, F. (2019). A modern introduction to online learning. arXiv preprint arXiv:1912.13213. 154

Ouyang, Y. and Xu, Y. (2018). Lower complexity bounds of first-order methods for convexconcave bilinear saddle-point problems. arXiv preprint arXiv:1808.02901. 190

Oymak, S., Recht, B., and Soltanolkotabi, M. (2018). Sharp time-data tradeoffs for linear inverse problems. IEEE Transactions on Information Theory, 64(6):4129-4158. 192

Palma, A. D., Behl, H., Bunel, R. R., Torr, P., and Kumar, M. P. (2021). Scaling the convex barrier with active sets. In International Conference on Learning Representations. 152

Pang, T., Lin, M., Yang, X., Zhu, J., and Yan, S. (2022). Robustness and accuracy could be reconcilable by (Proper) definition. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 17258-17277. PMLR. 176

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2021). Bag of tricks for adversarial training. In International Conference on Learning Representations. 154

Parikh, N., Boyd, S., et al. (2014). Proximal algorithms. Foundations and trends® in Optimization, 1(3):127-239. 6, 68, 71, 152, 184, 186, 195

Park, J. and Boyd, S. (2017). General heuristics for nonconvex quadratically constrained quadratic programming. arXiv preprint arXiv:1703.07870. 19, 20

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8024-8035. Curran Associates, Inc. 30, 68

Pauli, P., Koch, A., Berberich, J., Kohler, P., and Allgöwer, F. (2022). Training robust neural networks using lipschitz bounds. IEEE Control Systems Letters, 6:121-126. 70

Pontil, M. and Maurer, A. (2013). Excess risk bounds for multitask learning with trace norm regularization. In Shalev-Shwartz, S. and Steinwart, I., editors, Proceedings of the 26th Annual Conference on Learning Theory, volume 30 of Proceedings of Machine Learning Research, pages 55-76, Princeton, NJ, USA. PMLR. 224

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. (2021). The intrinsic dimension of images and its impact on learning. In International Conference on Learning Representations. 222

Qiao, L., Zhang, B., Su, J., and Lu, X. (2016). Linearized alternating direction method of multipliers for constrained nonconvex regularized optimization. In Asian Conference on Machine Learning, pages 97-109. 192

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). Dataset Shift in Machine Learning. The MIT Press. 2

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ArXiv e-prints. 183, 188, 189

Raghunathan, A., Steinhardt, J., and Liang, P. (2018a). Certified defenses against adversarial examples. In International Conference on Learning Representations. 5, 12, 20, 21

Raghunathan, A., Steinhardt, J., and Liang, P. (2018b). Certified defenses against adversarial examples. In International Conference on Learning Representations. 29, 30, 32

Raghunathan, A., Steinhardt, J., and Liang, P. S. (2018c). Semidefinite relaxations for certifying robustness to adversarial examples. In Advances in Neural Information Processing Systems, pages 10877-10887. 20

Rao, C. R. (1970). Estimation of heteroscedastic variances in linear models. Journal of the American Statistical Association, 65(329):161-172. 110

Ravi, S. N., Dinh, T., Lokhande, V. S., and Singh, V. (2019). Explicitly imposing constraints in deep networks via conditional gradients gives improved generalization and faster convergence. In AAAI. 41

Razaviyayn, M., Hong, M., and Luo, Z.-Q. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM Journal on Optimization, 23(2):1126-1153. 76

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. (2021). Fixing data augmentation to improve adversarial robustness. arXiv preprint arXiv:2103.01946. 176

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. (2019). Do imagenet classifiers generalize to imagenet? In International Conference on Machine Learning. 2

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. arXiv preprint arXiv:1401.4082. 193

Rice, L., Wong, E., and Kolter, Z. (2020). Overfitting in adversarially robust deep learning. In III, H. D. and Singh, A., editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 8093-8104. PMLR. 7, 166, 169, 170, 174, 176

Robey, A., Chamon, L. F. O., Pappas, G. J., Hassani, H., and Ribeiro, A. (2021a). Adversarial robustness with semi-infinite constrained learning. In Neural Information Processing Systems. 178

Robey, A., Hassani, H., and Pappas, G. J. (2020). Model-based robust deep learning: Generalizing to natural, out-of-distribution data. arXiv preprint arXiv:2005.10247. 165

Robey, A., Pappas, G. J., and Hassani, H. (2021b). Model-based domain generalization. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems. 2, 165

Sagawa, S., Raghunathan, A., Koh, P. W., and Liang, P. (2020). An investigation of why overparameterization exacerbates spurious correlations. In International Conference on Machine Learning, pages 8346-8356. PMLR. 165

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved techniques for training gans. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems 29, pages 2234-2242. Curran Associates, Inc. 188

Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In International Conference on Learning Representations. 8, 183, 187, 194

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? Advances in neural information processing systems, 31. 159

Santurkar, S., Tsipras, D., and Madry, A. (2021). Breeds: Benchmarks for subpopulation shift. International Conference on Learning Representations. 165

Scaman, K. and Virmaux, A. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Advances in neural information processing systems (NeurIPS). 92, 105

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. (2017). Group sparse regularization for deep neural networks. Neurocomputing, 241:81-89. 41

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. (2018). Adversarially robust generalization requires more data. In Advances in neural information processing systems (NeurIPS). 166

Schulam, P. F. and Saria, S. (2017). Reliable decision support using counterfactual models. In NIPS. 2

Sedghi, H., Gupta, V., and Long, P. M. (2019). The singular values of convolutional layers. In International Conference on Learning Representations. 74, 79

Shah, V. and Hegde, C. (2018). Solving Linear Inverse Problems Using GAN Priors: An Algorithm with Provable Guarantees. arXiv:1802.08406 [cs, stat]. arXiv: 1802.08406. 184, 192

Shalev-Shwartz, S. and Ben-David, S. (2014a). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press. 95, 109

Shalev-Shwartz, S. and Ben-David, S. (2014b). Understanding machine learning: From theory to algorithms. Cambridge university press. 166, 168

Shen, X., Chen, L., Gu, Y., and So, H.-C. (2016). Square-root lasso with nonconvex regularization: An admm approach. IEEE Signal Processing Letters, 23(7):934-938. 192

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 1

Sion, M. (1958). On general minimax theorems. Pacific Journal of Mathematics, 8(1):171-176. 162

Sitawarin, C., Bhagoji, A. N., Mosenia, A., Chiang, M., and Mittal, P. (2018). Darts: Deceiving autonomous cars with toxic signs. ArXiv, abs/1802.06430. 3

Slyusar, V. (1999). A family of face products of matrices and its properties. Cybernetics and Systems Analysis, 35(3):379-384. 95, 109

Sohoni, N., Dunnmon, J., Angus, G., Gu, A., and Ré, C. (2020). No subclass left behind: Fine-grained robustness in coarse-grained classification problems. Advances in Neural Information Processing Systems, 33:19339-19352. 165

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. D. (2017). Robust large margin deep neural networks. Trans. Sig. Proc., 65(16):4265-4280. 70

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems, pages 1-19. 79

Srebro, N. (2005). Learning with matrix factorizations. PhD. Thesis. 223
Srebro, N., Rennie, J., and Jaakkola, T. S. (2005). Maximum-margin matrix factorization. In Saul, L. K., Weiss, Y., and Bottou, L., editors, Advances in Neural Information Processing Systems 17, pages 1329-1336. MIT Press. 223

Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. In Auer, P. and Meir, R., editors, Learning Theory, volume 3559, pages 545-560. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science. 223

Stengle, G. (1974). A nullstellensatz and a positivstellensatz in semialgebraic geometry. Mathematische Annalen, 207(2):87-97. 15

Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling adversarial robustness and generalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6976-6987. 166

Su, J., Byeon, W., Kossaifi, J., Huang, F., Kautz, J., and Anandkumar, A. (2020). Convolutional tensor-train lstm for spatio-temporal learning. Advances in neural information processing systems (NeurIPS). 7, 91

Sun, X., Zhang, Z., Ren, X., Luo, R., and Li, L. (2021). Exploring the vulnerability of deep neural networks: A study of parameter corruption. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 11648-11656. 166

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1-9. 1

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. 29, 92, 165, 168

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014a). Intriguing properties of neural networks. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. 3, 4, 12, 185, 194

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014b). Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y., editors, 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. 145

Taori, R., Dave, A., Shankar, V., Carlini, N., Recht, B., and Schmidt, L. (2020). Measuring robustness to natural distribution shifts in image classification. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, Advances in Neural Information Processing Systems, volume 33, pages 18583-18599. Curran Associates, Inc. 2

Terjék, D. (2020). Adversarial lipschitz regularization. In International Conference on Learning Representations. 5

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 58(1):267-288. 193

Tjeng, V., Xiao, K. Y., and Tedrake, R. (2019). Evaluating robustness of neural networks with mixed integer programming. In International Conference on Learning Representations. 152

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A. P., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). MLP-mixer: An all-MLP architecture for vision. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems. 79, 244

Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In CVPR 2011, pages 1521-1528. 2

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. arXiv e-prints, page arXiv:2302.13971. 1

Tramer, F. and Boneh, D. (2019). Adversarial training and robustness for multiple perturbations. In Wallach, H., Larochelle, H., Beygelzimer, A., d’ Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc. 157

Tran-Dinh, Q., Pham, N. H., Phan, D. T., and Nguyen, L. M. (2021). A hybrid stochastic optimization framework for composite nonconvex optimization. Mathematical Programming, pages 1-67. 71

Tripathi, S., Lipton, Z. C., and Nguyen, T. Q. (2018). Correction by Projection: Denoising Images with Generative Adversarial Networks. arXiv e-prints, page arXiv:1803.04477. 183, 187

Tropp, J. A. (2015). An introduction to matrix concentration inequalities. Foundations and Trends ${ }^{\circledR}$ in Machine Learning, 8(1-2):1-230. 225

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness may be at odds with accuracy. In International Conference on Learning Representations. 44, 166, 169

Tsuzuku, Y., Sato, I., and Sugiyama, M. (2018a). Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems 31, pages 6541-6550. Curran Associates, Inc. 5, 42

Tsuzuku, Y., Sato, I., and Sugiyama, M. (2018b). Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc. 92

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2018). Deep image prior. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9446-9454. 183

Vakanski, A. and Xian, M. (2021). Evaluation of Complexity Measures for Deep Learning Generalization in Medical Image Analysis. arXiv e-prints, page arXiv:2103.03328. 68

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. In Advances in neural information processing systems (NeurIPS), pages 5998-6008. 7

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017b). Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 6000-6010, Red Hook, NY, USA. Curran Associates Inc. 222

Vershynin, R. (2011). A simple decoupling inequality in probability theory (2011. http:// www-personal.umich.edu/~romanv/papers/decoupling-simple.pdf. [Online; accessed 1-February-2021]. 227

Vershynin, R. (2018). High-Dimensional Probability: An Introduction with Applications in Data Science. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. 227, 236

Viallard, P., Germain, P., Habrard, A., and Morvant, E. (2021). Self-bounding majority vote learning algorithms by the direct minimization of a tight pac-bayesian c-bound. In Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., and Lozano, J. A., editors, Machine Learning and Knowledge Discovery in Databases. Research Track, pages 167-183, Cham. Springer International Publishing. 157

Virmaux, A. and Scaman, K. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Advances in Neural Information Processing Systems 31, pages 3835-3844. Curran Associates, Inc. 5, 20

Vorobeychik, Y., Kantarcioglu, M., and Brachman, R. (2018). Adversarial Machine Learning. Morgan \& Claypool Publishers. 157

Waki, H., Kim, S., Kojima, M., and Muramatsu, M. (2006). Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM Journal on Optimization, 17(1):218-242. 17, 19

Wang, H., Zhang, A., Zheng, S., Shi, X., Li, M., and Wang, Z. (2022). Removing batch normalization boosts adversarial training. In International Conference on Machine Learning, pages 23433-23445. PMLR. 159

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. (2021). BetaCROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, Advances in Neural Information Processing Systems. 152

Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 7794-7803. 6, 91

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. (2019). On the convergence and robustness of adversarial training. In Chaudhuri, K. and Salakhutdinov, R., editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 6586-6595. PMLR. 156

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. (2020). Improving adversarial robustness requires revisiting misclassified examples. ICLR. 166, 174, 176

Wang, Z., Pang, T., Du, C., Lin, M., Liu, W., and Yan, S. (2023). Better diffusion models further improve adversarial training. arXiv preprint arXiv:2302.04638. 176

Wang, Z., Zheng, S., Boyd, S., and Ye, Y. (2006). Further relaxations of the sdp approach to sensor network localization. Tech. Rep. 19

Wei, C. and Ma, T. (2020). Improved sample complexities for deep neural networks and robust classification via an all-layer margin. In International Conference on Learning Representations. 157

Weisser, T., Lasserre, J. B., and Toh, K.-C. (2018). Sparse-bsos: a bounded degree sos hierarchy for large scale polynomial optimization with sparsity. Mathematical Programming Computation, 10(1):1-32. 12, 16, 19

Wen, F., Chu, L., Liu, P., and Qiu, R. C. (2018). A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning. IEEE Access, 6:69883-69906. 42

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. (2018). Towards fast computation of certified robustness for ReLU networks. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 5276-5285, Stockholmsmässan, Stockholm Sweden. PMLR. 12, 21

Wimalawarne, K., Tomioka, R., and Sugiyama, M. (2016). Theoretical and experimental analyses of tensor-based regression and classification. Neural Computation, 28(4):686-715. Publisher: MIT Press. 223

Wong, E. and Kolter, Z. (2018a). Provable defenses against adversarial examples via the convex outer adversarial polytope. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 52865295, Stockholmsmässan, Stockholm Sweden. PMLR. 14, 24, 169

Wong, E. and Kolter, Z. (2018b). Provable defenses against adversarial examples via the convex outer adversarial polytope. In Dy, J. and Krause, A., editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 5286-5295. PMLR. 165

Wong, E., Rice, L., and Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training. In International Conference on Learning Representations. 156, 177

Wong, E., Schmidt, F., and Kolter, Z. (2019). Wasserstein adversarial examples via projected Sinkhorn iterations. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 6808-6817, Long Beach, California, USA. PMLR. 14

Wu, D., tao Xia, S., and Wang, Y. (2020). Adversarial weight perturbation helps robust generalization. NeurIPS. 166

Wu, Y. and He, K. (2018). Group normalization. In Proceedings of the European conference on computer vision (ECCV), pages 3-19. 155, 159

Xiao, H., Rasul, K., and Vollgraf, R. (2017a). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747. 43, 69

Xiao, H., Rasul, K., and Vollgraf, R. (2017b). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747. 99, 131

Xiao, K., Engstrom, L., Ilyas, A., and Madry, A. (2021). Noise or signal: The role of image backgrounds in object recognition. International Conference on Machine Learning. 165

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. (2019). Feature denoising for improving adversarial robustness. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 141, 143

Xie, S., Kirillov, A., Girshick, R., and He, K. (2019). Exploring Randomly Wired Neural Networks for Image Recognition. International Conference on Computer Vision. 18

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv e-prints, page arXiv:1505.00853. 188

Xu, Y., Jin, R., and Yang, T. (2019a). Non-asymptotic analysis of stochastic methods for nonsmooth non-convex regularized problems. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc. 71

Xu, Y., Liu, M., Lin, Q., and Yang, T. (2017). Admm without a fixed penalty parameter: Faster convergence with new adaptive penalization. In Advances in Neural Information Processing Systems, pages 1267-1277. 186

Xu, Y., Qi, Q., Lin, Q., Jin, R., and Yang, T. (2019b). Stochastic optimization for DC functions and non-smooth non-convex regularizers with non-asymptotic convergence. In Chaudhuri, K. and Salakhutdinov, R., editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 6942-6951. PMLR. 71
$\mathrm{Xu}, \mathrm{Y}$. and Yin, W. (2013). A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM Journal on Imaging Sciences, 6(3):1758-1789. 75

Yan, Y., Yang, T., Li, Z., Lin, Q., and Yang, Y. (2018). A unified analysis of stochastic momentum methods for deep learning. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 2955-2961. International Joint Conferences on Artificial Intelligence Organization. 159

Yang, Y., Yuan, Y., Chatzimichailidis, A., van Sloun, R. J., Lei, L., and Chatzinotas, S. (2020). Proxsgd: Training structured neural networks under regularization and constraints. In International Conference on Learning Representations. 68

Yang, Z., Huang, C., Chen, X., Lin, W., and Liu, Z. (2016). A linear programming relaxation based approach for generating barrier certificates of hybrid systems. In Fitzgerald, J., Heitmeyer, C., Gnesi, S., and Philippou, A., editors, FM 2016: Formal Methods, pages 721-738, Cham. Springer International Publishing. 19

Ye, Y. (1999). Approximating quadratic programming with bound and quadratic constraints. Mathematical Programming, 84(2):219-226. 20

Yehudai, G. and Shamir, O. (2019). On the power and limitations of random features for understanding neural networks. 32. 222

Yoon, J. and Hwang, S. J. (2017). Combined group and exclusive sparsity for deep neural networks. In Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 3958-3966, International Convention Centre, Sydney, Australia. PMLR. 41

Yoshida, Y. and Miyato, T. (2017). Spectral Norm Regularization for Improving the Generalizability of Deep Learning. arXiv e-prints, page arXiv:1705.10941. 92

Yu, C., Han, B., Shen, L., Yu, J., Gong, C., Gong, M., and Liu, T. (2022). Understanding robust overfitting of adversarial training and beyond. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 25595-25610. PMLR. 176

Yu, H.-F., Jain, P., Kar, P., and Dhillon, I. (2014). Large-scale multi-label learning with missing labels. In Xing, E. P. and Jebara, T., editors, Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 593-601, Bejing, China. PMLR. 224

Yurtsever, A., Fercoq, O., and Cevher, V. (2019). A conditional-gradient-based augmented lagrangian framework. In Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 7272-7281, Long Beach, California, USA. PMLR. 243

Zech, J. R., Badgeley, M. A., Liu, M., Costa, A. B., Titano, J. J., and Oermann, E. K. (2018). Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. PLoS Medicine, 15. 2

Zhang, H., Wang, S., Xu, K., Wang, Y., Jana, S., Hsieh, C.-J., and Kolter, Z. (2022a). A branch and bound framework for stronger adversarial attacks of ReLU networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 26591-26604. PMLR. 152

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. In Advances in Neural Information Processing Systems 31, pages 4939-4948. Curran Associates, Inc. 5

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., and Jordan, M. (2019a). Theoretically principled trade-off between robustness and accuracy. In Proceedings of the 36th International Conference on Machine Learning. 132

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. (2019b). Theoretically principled trade-off between robustness and accuracy. In ICML. 166, 174

Zhang, Y., Yang, Z., Lin, W., Zhu, H., Chen, X., and Li, X. (2018). Safety verification of nonlinear hybrid systems based on bilinear programming. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(11):2768-2778. 19

Zhang, Y., Zhang, G., Khanduri, P., Hong, M., Chang, S., and Liu, S. (2022b). Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 26693-26712. PMLR. 177

Zhu, X. and Wu, X. (2004). Class Noise vs. Attribute Noise: A Quantitative Study. Artificial Intelligence Review, 22(3):177-210. 2

Zhu, Z., Latorre, E., Chrysos, G., and Cevher, V. (2022). Controlling the complexity and lipschitz constant improves polynomial nets. In International Conference on Learning Representations. iii, v, 6, 80

# FABIAN RICARDO LATORRE-GOMEZ 

+41 $764974435 \diamond$ Lausanne, Switzerland<br>fabian.latorre@epfl.ch $\diamond$ Google Scholar

## EDUCATION

## Ph.D. Computer and Communication Sciences.

EPFL, Lausanne, Switzerland.
(09/2017-08/2023 Expected)
M.Sc. Mathematics

Universidad de los Andes, Bogota, Colombia. GPA: 4.19/5.0
(08/2012-03/2015)

## B.Sc. Mathematics

Universidad de los Andes, Bogota, Colombia. GPA: 4.38/5.0 (01/2008-08/2012)

## AWARDS

1. EPFL School of Engineering Teaching Assistant Award (2022).
2. Swiss Data Science Center PhD Fellowship (2019).
3. EDIC Doctoral School Fellowship. EPFL (2017).
4. M.Sc. Scholarship. Mathematics Department, Universidad de los Andes (2012).
5. Quiero Estudiar Undergraduate Scholarship. Universidad de los Andes (2008).

## EXPERIENCE

Applied Scientist Intern, Amazon. Tuebingen, Germany. (07/2022-10/2022)
Lead a research project on Fairness in Generative Models (GANs). Obtained experimental evaluation using the AWS EC2 platform. Manuscript in preparation. Under supervision of Matthaeus Kleindessner, Ph.D.

Research Intern, Salesforce Research. Morges, Switzerland. (02/2022-04/2022) Lead a research project developing a new shape-sensitive metric for time series data based on the Optimal Transport (OT). It improved upon the baselines in 1-NN Classification, Hierarchical Clustering, and also as part of Deep Learning architectures. Experimental evaluation was obtained using Kubernetes and Google Cloud Engine. To appear at ICASSP 2023. Under supervision of Steven C.H. Hoi, Ph.D.

Doctoral Assistant, EPFL. Lausanne, Switzerland.
(08/2017 - present)
From 03/2018 I am part of the Laboratory for Information and Inference Systems (LIONS) Under supervision of Prof. Volkan Cevher, Ph.D. I have developed new algorithms, theory and empirical evaluation for the training of deep neural networks, and published my research at the top international Machine Learning Conferences (NeurIPS/ICML/ICLR). Additionaly, I supervise Bachelor/Master students theses, and I am a teaching assistant for the masters level course Mathematics of Data. From 08/2017 to 02/2019 I was part of the Machine Learning and Optimization Lab, EPFL, Lausanne Switzerland, where I did a Semester Project on Randomized Coordinate Descent Algorithms for minimization of convex objectives, under supervision of Professor Martin Jaggi, Ph.D.

Data Scientist, Quantil S.A.S. Bogota, Colombia.
(09/2016-07/2017)
I developed data science products for the following customers:

- Young \& Rubicam (www.yr.com): I implemented a predictive model for brand value, based on twitter data and sentiment analysis.
- Universidad de los Andes: Using tools from Natural Language Processing (NLP) I developed a topic model generator (backend + frontend) using corpora of research articles available from elsevier (www.elsevier.com/).


## Research Assistant, Independent Contractor. Bogota, Colombia. (05/2015 -

 08/2016)Joint work with Economics Ph.D. Candidates Santiago Saavedra (Stanford) and Mauricio Romero (UCSD), United States. I fitted and assessed the performance of a machine learning model for the prediction of illegal mining locations in the Colombian Territory, based on Landsat satellite images, among other datasets. Partially funded by University of California, San Diego (UCSD). See paper: https://bit.ly/3Qq9NOC

Teaching Assistant - Adjunct Instructor, Universidad de los Andes. Bogota,
Colombia. Courses taught: Vector Calculus, Calculus 3 for Economics and Business Administration, Integral Calculus \& Probability, Calculus for the Biological Sciences, Differential Calculus. Integral Calculus, Linear Algebra.

## RESEARCH ARTICLES

1. Latorre, Fabian; Liu, Chenghao; Sahoo, Doyen and Hoi, Steven C.H. OTW: Optimal Transport Warping for Time Series. To appear at ICASSP 2023.
2. Latorre, Fabian; Krawczuk, Igor; Dadi, Leello; Pethick, Thomas and Cevher, Volkan. Adversarial Training descends without descent: Finding actual descent directions based on Danskin's theorem . ICLR 2023 (Poster).
3. Zhu, Zhenyu; Latorre, Fabian; Chrysos, Grigorios and Cevher, Volkan. Controlling the Complexity and Lipschitz Constant improves Polynomial Nets. ICLR 2022 (Poster). code: https://bit.ly/3isHUco
4. Latorre, Fabian; Dadi, Leello; Roland, Paul and Cevher, Volkan. The Effect of the Intrinsic Dimension on the Generalization of Quadratic Classifiers. NeurIPS 2021 (Poster). code: https://bit.ly/3vL78pr
5. Sun, Zhaodong; Latorre, Fabian; Sanchez, Thomas and Cevher, Volkan. A Plug-andPlay Deep Image Prior. ICASSP 2021. code: http://bit.ly/3jS8Z92
6. Latorre, Fabian; Rolland, Paul; Hallak, Nadav and Cevher, Volkan. Efficient Proximal Mapping of the 1-path-norm regularizer of Shallow Networks. ICML 2020 (Poster). code: https://bit.ly/3GQmcYW
7. Latorre, Fabian; Rolland, Paul and Cevher, Volkan. Lipschitz constant estimation for Neural Networks via sparse polynomial optimization. ICLR 2020 (Poster). code: https: //bit.ly/3IwyYx1
8. Latorre, Fabian; Eftekhari, Armin and Cevher, Volkan. Fast and provable ADMM for learning with Generative priors. NeurIPS 2019 (Spotlight talk). code: https: //bit.ly/3IBJGma
9. Alacaoglu, Ahmet; Sahin, Fatih; Eftekhari, Armin; Latorre, Fabian and Cevher, Volkan. An Inexact Augmented Lagrangian Framework for Non-Convex Optimization with Nonlinear Constraints. NeurIPS 2019 (Poster).

## COMMUNITY SERVICE

Reviewer - Algorithms and Data Structures Symposium (WADS)
Reviewer - Transactions on Image Processing
Reviewer - ICLR
Reviewer - ICML
Reviewer - NeurIPS
(2019 - present)
Sponsor and Finance Chair - LatinX in AI workshop at ICML $(2020,2023)$
Sponsor and Finance Chair - LatinX in AI workshop at NeurIPS

## SOFTWARE

Python (2.7-3.8): PyTorch, Tensorflow, scikit-learn, numpy, scipy, pandas.
Others: Java, C++, HTML, CSS, JavaScript, Linux, Bash Scripting.
Open source projects (other than code from publications):

- TOPMINE python implementation (mining frequent phrases in text),
- Evaluation Metrics for Generative Models (Pytorch port)


[^0]:    ${ }^{1}$ https://artificialintelligenceact.eu

[^1]:    ${ }^{1}$ also known as Krivine's Positivstellensatz

[^2]:    ${ }^{2}$ For training we used the code from this reference. It is publicly available in https://github.com/

[^3]:    $\overline{\text { mightydeveloper/Deep-Compression-PyTorch }}$

[^4]:    ${ }^{1}$ We will call an optimal solution trivial if $v^{*}=0$.

[^5]:    ${ }^{1}$ Here, the 0 -th and $L$-th layer correspond to the input and output layer, respectively.

[^6]:    ${ }^{1}$ https://paperswithcode.com/sota/face-verification-on-megaface
    2 with non-polynomial activation functions.

[^7]:    ${ }^{3}$ The samples were found in https://www.tensorflow.org/datasets/catalog.

[^8]:    ${ }^{4}$ The single bound is mentioned as 'Our method' in the previous tables. In this experiment both 'single bound' and 'layer-wise bound' are proposed.

[^9]:    ${ }^{1}$ It is worth noting that the early stopping robust accuracy we achieve in ablations approximately matches that reported in Engstrom et al. (2019) on resnet50

[^10]:    ${ }^{2}$ There are whole lines of work studying the effects of BN (Bjorck et al., 2018; Santurkar et al., 2018; Kohler et al., 2019) as well as removing it altogether(Brock et al., 2021). It has also been found to interact with adversarial robustness in Wang et al. (2022) and Benz et al. (2021), the latter also finds GN to be a well performing alternative, justifying our choice.

[^11]:    ${ }^{1}$ To be precise, the optimal value $\eta^{\star}$ in (7.16) is a function of $(X, Y)$, i.e., $\eta^{\star}=\eta^{\star}(X, Y)$, and the constraint must hold for almost every $(X, Y) \sim \mathscr{D}$. We omit these details for ease of exposition.
    ${ }^{2}$ This result is similar in spirit to (Gowal et al., 2019, Theorem 3.1). However, (Gowal et al., 2019, Theorem 3.1) only holds for linear functions, whereas Proposition 7.1 holds for an abitrary function $f_{\theta}$.

[^12]:    ${ }^{3}$ In a strict sense, the formulation of Zhang et al. (2022b) is not a bilevel problem. In general, the most concise way to write a bilevel optimization problem is $\min _{\theta} f\left(\theta, \delta^{\star}(\theta)\right)$ subject to $\delta^{\star}(\theta) \in \operatorname{argmax} g(\theta, \delta)$. In such problems the value $\delta^{\star}(\theta)$ only depends on $\theta$, as the objective function $g(\theta, \cdot)$ is then uniquely determined. This is not the case in (Zhang et al., 2022b, eq. (7)), where an additional variable $z$ appears, corresponding to the random initialization of Fast-AT. Hence, in (Zhang et al., 2022b) the function $g(\theta, \cdot)$ is not uniquely defined by $\theta$, but is a random function realized at each iteration of the algorithm. Thus, it is not a true bilevel optimization problem in the sense of the textbook definition (Bard, 2013).

[^13]:    ${ }^{1}$ If necessary, the inclusion $\left\{w_{t}\right\}_{t \geq 0} \subset W$ might be enforced by adding the indicator function of the convex hull of $W$ to $R$ in (8.1), similar to Agarwal et al. (2010).

[^14]:    ${ }^{2}$ To be complete, $1_{W}(w)=0$ if $w \in W$ and $1_{W}(w)=\infty$ otherwise.

[^15]:    ${ }^{1}$ See remarks following the proof

[^16]:    ${ }^{2}$ for simplicity we ignore the logarithmic factor

