EPFL

Thèse n° 11 011

Machine Learning for Modeling Stock Returns

Présentée le 21 juin 2024

Collège du management de la technologie
Chaire du Prof. associé Malamud
Programme doctoral en management de la technologie

pour l'obtention du grade de Docteur ès Sciences

par

Teng Andrea XU

Acceptée sur proposition du jury

Prof. D. Kuhn, président du jury
Prof. S. Malamud, directeur de thèse
Prof. Y. Zhang, rapporteur
Dr E. Guidotti, rapporteur
Prof. D. Filipovic, rapporteur

École
polytechnique
fédérale
de Lausanne

2024

Sapere Aude!

Quintus Horatius Flaccus.
Immanuel Kant answering the question: What Is Enlightenment?

Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza.

Dante Alighieri. Inferno, Canto XXVI.

Abstract

Throughout history, the pace of knowledge and information sharing has evolved into an unthinkable speed and media. At the end of the XVII century, in Europe, the ideas that would shape the *"Age of Enlightenment"* were slowly being developed in coffeehouses, literary salons, and printed books. *L'âge des lumières* was characterized by shedding light on the darkness of human ignorance, and it was not a coincidence that it developed after Gutenberg invented the movable type printing machine. Today, at the end of the first quarter of the XXI century, *"The Age of AI has begun"*–Bill Gates. And again, it is not a coincidence that Artificial Intelligence is seeing its golden period after Tim Berners-Lee invented the World Wide Web. Books enlighten human brains. Similarly, massive datasets are now being fed to machine learning models and artificial brains, also known as neural networks. It is noteworthy that human brains are capable of creating knowledge; as for artificial brains, their capability in this regard is still uncertain. In this thesis, I will explore how data proliferation and artificial intelligence are affecting financial markets through the lens of a *statistician*. In the first chapter, I leverage extreme value theory to study time-varying idiosyncratic tail risk using option-implied information. In other words, does the information contained in the implied volatility surface explain extreme losses? The second chapter moves away from the distribution tails and focuses on the entire distribution of returns. It is known that stock and options traders systematically use the geometric shape of the option's implied volatility surface to infer market expectations and risk attitudes and make trading decisions. Thus, we leverage convolutional neural networks to capture spatial patterns and relationships between pixels in images and translate this information into a prediction forecast. Finally, it has been recently shown that a particular kernel, the Neural Tangent Kernel, represents an infinitely wide neural network in the lazy training regime. This discovery has revived interest in kernel methods and, thus, random features. In the final chapter, we provide an algorithm, which we call Fast Annihilating Batch Regression, which is capable of solving a regression with an *infinite* amount of random features, in theory.

Estratto

Durante la storia, il ritmo con cui condividiamo l'informazione e il *sapere* si sono evoluti completamente. I mezzi di comunicazione e la velocità di diffusione hanno raggiunto standard inimmaginabili. All fine del XVII secolo, in Europa, le idee che hanno poi formato l' *"Illuminismo"* si stavano lentamente sviluppando nei caffé, nei saloni letterari, e libri stampati. Illuminare l'oscurità dell'ignoranza umana è stato il tratto caratterizzante dell' *âge des lumières* e non è stato un caso che l'illuminismo si sia evoluto dopo che Gutenberg inventò la stampa mobile. Oggi, alla fine del primo quarto del XXI secolo, *"The Age of AI has begun"*–Bill Gates. Di nuovo, non è una coincidenza che l'Intelligenza Artificiale stia vivendo il periodo d'oro dopo che Tim Berners-Lee inventò il World Wide Web. Come i libri hanno illuminato le nostre teste, i dataset massivi sono il pane quotidiano di modelli di apprendimento automatico e cervelli artificiali, altrimenti conosciuti come reti neurali. È importante sottolineare il fatto che il cervello umano è capace di creare *sapere*, mentre per i cervelli artificiali non lo sappiamo ancora. In questa tesi, esplorerò come questa proliferatione di dati e l'uso dell'intelligenza artificiale stiano influenzando i mercati finanziari attraverso la lente di uno *statista.* Nel primo capitolo, sfrutto la teoria dei valori estremi per studiare il tempo-variante rischio idiosincratico di coda usando l'informazione derivanti le opzioni. In altre parole, può l'informazione contenuta nella superficie di volatilità implicita spiegare le perdite estreme? Il secondo capitolo lascia le distribuzioni di coda e si focalizza sull'intero distribuzione dei ritorni. È noto che i trader di stock e opzioni utilizzino sistematicamente la forma geometrica della superficie di volatilità implicita delle opzioni per dedurre conclusioni sul valore atteso del mercato, analisi dei rischi e prendere decisioni di trading. Allora, usiamo le reti neurali convoluzionali per catturare pattern spaziali e relazioni tra pixels nelle immagini e tradurre questa informazione in una previsione. Infine, recentemente è stato provate che un particolare kernel, il Neural Tangent Kernel, catturi l'essenza di una rete neural con infiniti nodi. Questa scoperta ha resuscitato interesse nei kenrel e quindi anche nei ranom features. Nel capitolo finale, sviluppiamo un algoritmo, che chiamiamo Fast Annihilating Batch Regression, capace di risolvere, teoricamente, una regressione con un *infinito* numero di random features.

Acknowledgements

I am extremely grateful to my advisor, Prof. Semyon Malamud, for giving me this invaluable experience and unconditional support. He is a brilliant mind who seeks truthful knowledge, and, at the same time, he is a *kind-hearted* man. I was truly inspired by him and enjoyed our brainstorming sessions, which spanned tons of fields, including asset pricing, derivatives, cryptocurrencies, kernel methods, statistical learning, deep learning, generative models, information theory, and computer science.

During my PhD, I also had the privilege of visiting the Yale School of Management and the University of San Diego's Halıcıoğlu Data Science Institute. Prof. Bryan Kelly and Prof. "Misha" Belkin are leaders in their respective fields. Thank you for your warm hospitality and for being professional in providing invaluable feedback and additional guidance. I'd like to thank Prof. Jiahua Xu for her co-supervision during my early career. I would like to express my gratitude to my dissertation committee members for their time and feedback: Prof. Damir Filipović, Prof. Daniel Kuhn, Dr. Emanuele Guidotti, and Prof. Yuan Zhang.

The voyage to become a *Philosophiae Doctor* could be a stormy cruise. Very stormy sometimes. Developing a vision, discovering unseen territories, and connecting with other pioneers will make the voyage brighter. This is our *"Quiet after the Storm."*

A journey needs companions. Sometimes, journeys may last a lifetime. Since kindergarten, I was lucky enough to meet, laugh, and share experiences with tons of people who I now call *friends*. Uncountable friends literally forged me, and I am who I am, thanks to them.

A journey can only be meaningful if we have a home to return to. And I was blessed with two of them. I am grateful for my entire family's unconditional love, particularly my parents, who did everything to ensure I never missed anything, and my siblings, who endured my craziness and still do.

A journey has an end. As a journey comes to its final destination, a new one has to start. I am excited to start a new one, this time together with Bahar. Thank you for being so inspirational and showing me that a stormy and dark day can also be joyful and peaceful. *"...e il naufragar m'è dolce in questo mare."*

Andrea

Introduction

Little more than half a century since Alan Turing proposed the imitation game, or the Turing test, to assess a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. In short, this experiment involves a human judge engaging in a conversation with both a human and a machine, unaware of which is which. The machine would pass the test if the judge were incapable of distinguishing the machine from the human. A recent Nature article now writes *"ChatGPT broke the Turing test, the race is on for new ways to assess AI"*–[Bie23]. Of course, the test was never designed to be a thorough experiment but rather to be food for thought. The race to strong AI, the race to a human-like digital brain, was never as intense as today. After the historic win of AlphaGo[Sil+16] over Lee Sedol winning the title of best Go player, AI re-shaped human-related tasks across a variety of fields, such as protein folding [Jum+21], mineral discovery [Mer+23], text [Tou+23; Tea+23] and image [Ram+21; Rom+22] generation, to name a few. Deepmind set the breaking point, the moment when the word "AI" was in the mouth of everyone. At the same time, a few years later, OpenAI published a generative model [Ach+23] which later led to *sparks of artificial general intelligence* [Bub+23]. It seems that given enough information and computational power, neural networks are capable of *understanding the nature* behind the given task [HSW89]. Equipped with these notions, the primary goal of this work is to understand the nature of return distributions using over-parametrized machine learning models. Let's introduce each concept. In capital markets, risky assets pay higher returns. The behavior of those markets is a challenge that intertwines modeling agents' utility function and pricing theory. This last sentence is well-explained by the consumption-based model: *"An investor must decide how much to save and how much to consume, and what portfolio of assets to hold.–[Coc09]."* Without going into details, one would consume less if and only if the utility of this action equals the future marginal utility gain of the assets payoff. The marginal utility rate, also known as the stochastic discount factor (SDF), is used to discount future payoff. The SDF is not observable. Capital market participants' consumption behavior leaves the breadcrumbs that could lead to the true SDF. Although the consumption-based model is an elegant model, it does not work well in practice: it shows high pricing errors [Coc09]. A different approach to explain the return distribution is to use a statistical model conditioned to some information set [WG08; RSZ10]. From the point of view of a *statistician*, future risk premia $R_{t+1} \in \mathbb{R}^N$, the payoff of holding a set of risky assets, can be modeled as follows

$$R_{t+1} \;=\; \mathbb{E}[R_{t+1}|\mathcal{F}_t] + \epsilon_{t+1}, \tag{1}$$

where the conditional expectation $\mathbb{E}[R_{t+1}|\mathcal{F}_t]$ represents the return forecast conditional to the information sets $\mathcal{F}_t$, and ϵ_{t+1} gathers the unpredictable variation in returns. The information sets $\mathcal{F}_t$ is the usual mathematical representation of all information known up to time t. Quoting [KX+23], these information sets are large and hard to bind, spanning from public information to more subtle, hard-to-get info. Then, by imposing the least statistical structure, we can rewrite (1): let $X_{i,t} \in \mathbb{R}^d$ to be a vector of predictors describing stock i at time

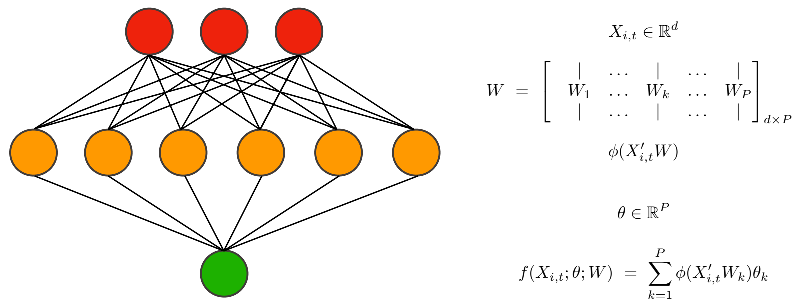

$X_{i,t} \in \mathbb{R}^d$
$W = \begin{bmatrix} | & \dots & | & \dots & | \\ W_1 & \dots & W_k & \dots & W_P \\ | & \dots & | & \dots & | \end{bmatrix}_{d\times P}$
$\phi(X'_{i,t}W)$
$\theta \in \mathbb{R}^P$
$f(X_{i,t};\theta;W) = \sum_{k=1}^{P} \phi(X'_{i,t}W_k)\theta_k$

Figure 1: A shallow neural network.

t then we get

$$\mathbb{E}[R_{i,t+1}|\mathcal{F}_t] = f(X_{i,t}), \tag{2}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a generic function. By plugging (2) into (1) we get

$$R_{i,t+1} = f(X_{i,t}) + \epsilon_{i,t+1}. \tag{3}$$

In the literature, by *machine learning portfolios*, it refers to using a highly parametrized statistical model f and to construct market timing strategies in the form of

$$\pi_{i,t} = f(X_{i,t}). \tag{4}$$

That is, the strategy takes positions proportional (or equal) to the asset's conditional expected return. Two natural questions are raised at this stage: first, which predictor should one pick, and then, without knowing the true data-generating process, how could she avoid model misspecification?

The virtue of complexity in return predictions

The series of work that gave birth to the *virtue of complexity* (VoC) [KMZ22; KMZ24] solves the two problems mentioned above. By leveraging the recent advances in statistical learning such as the *"double descent"* phenomenon [Bel+18], the notion of random features [RR07], and random matrix theory [Tao23], the authors are able to show that the portfolio out-of-sample performance, measured in Sharpe ratio, is strictly increasing in model complexity and number of predictors. The intuition is built from the work of [Has+19], where it is shown that when the number of parameters is much higher than the number of observations, ridgeless regression selects the solution with the smallest norm.

This acts as shrinkage, biasing the beta estimate toward zero and thus inducing the forecast variance to decrease. The choice of using random features should not be overlooked. On one hand, it allows the authors to move smoothly from a low-complexity model to a higher-complexity model. On the other hand, a least-squares involving random features is equivalent to a shallow neural network representation. Figure 1 depicts a neural network with a single hidden layer on the left, while the mathematical random features representation is shown on the right. When the elementwise non-linear activation function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ equals a cosine/sine, then a shallow neural network is equivalent to the least squares with a random Fourier features projection. The takeaway message, the bigger the information sets, the better. The *over-parametrized* regime, when the number of parameters is higher than the number of observations, is currently being studied by the most brilliant researchers. See, [COB19; dSB20; Bel21], to cite a few. In case it was not clear, the classic concept of *variance-bias* trade-off and the corresponding regularization of *early-stopping* are now obsolete. Past the *interpolation-threshold* (when the number of data points equals the number of features), the statistical model continues to improve its generalization performance. It even leads to curious phenomena such as *grokking* [Pow+22] and *neural-collapse* [PHD20].

This work

The VoC findings and the over-parametrized regime motivate the research of this work. Neural networks are interpolators [Has+19], little is known about their use in finance, and only a small subset of the current work in the literature show correct *parametrization*, connection to the most recent advances in statistical learning, and reproducibility of the results. Financial applications of neural nets usually get two major criticisms: (i) because of their non-convexity, we do not fully comprehend the reasons behind their exceptional out-of-sample performance, and (ii) they do not have any economic value. Let's first deep dive into the first problem. When talking about neural networks parametrized with w_t, it is impossible to avoid mentioning the gradient descent to minimize the in-sample loss $\mathcal{L}$

$$w_{t+1} = w_t - \eta \nabla \mathcal{L}, \tag{5}$$

where w_t represents the current state of parameters and η the learning rate. In this case, $\mathcal{L}$ is simply the mean-squared loss between the true realized return and the model forecast. For feasibility reasons, in practice, the stochastic version of the gradient descent is being used together with its variant, Adam [KB14]. At initialization, w_0 is usually drawn from a normal Gaussian distribution; then, the neural net will optimize those weights through multiple optimization steps. Because of the stochastic behavior, a correct parametrization is key to achieving high out-of-sample performance. A series of recent works called *Tensor Programs* provably show how to initialize and parameterize multi-layer perceptrons, convolutional neural networks, and attention matrices, among others, correctly [Yan19; Yan20a; YL21; Yan20b; YH20; YL23; Yan+21; Yan+23].

[Kel+23] show, surprisingly, that running neural networks over independent runs not only reduces uncertainty on convergence but also improves performance. Little by little, the scientific community is shedding light on neural networks by providing rigorous definitions and theorems. Finite neural nets are equivalent to Gaussian processes [Nea96; Lee+18]. When their width is pushed to the infinite limit, they converge to a kernel [JGH18; Aro+19b; Aro+19a]. The finite neural nets learn, through the *gradient-outer product*, which *feature* matters [Rad+22]. A recent work directly connects optimal portfolio construction for infinite wide neural nets [Kel+24]. Thus, neural net behavior under gradient descent steps is now almost *predictable.* About the lack of economic value criticism, I will quote [KX+23]:

> *Relatively unstructured prediction models make them no less economically important than the traditional econometrics of structural hypothesis testing; they just play a different scientific role. [...] Even if we could observe expected returns perfectly, we would still need theories to explain their behavior and empirical analysis to test those theories. [...] A critical benefit of expanding the set of known contours in the empirical landscape is that, even if details of the economic mechanisms remain shrouded, economic actors, financial market participants in particular, can always benefit from improved empirical maps. [...] The economics of the prediction model culture lies precisely in its ability to improve predictions. Armed with better predictions–i.e., more accurate assessments of the economic opportunity set—agents can better trade off costs and benefits when allocating scarce resources. This enhances welfare.*

We now close the long digression about consumption-based models, financial markets return forecasts, and (deep) statistical learning. By now, it should be clear that this work takes advantage of recent advances in deep learning to explore various methods for explaining risk premia. **The first chapter** uses extreme value theory to study time-varying idiosyncratic tail risk using option-implied information. In this case, a neural network is trained to maximize the conditional log-likelihood of log return exceedance using option-implied information. The key point is the known Pickands–Balkema–De Haan theorem [BDH74; Pic75]. In short, for a large class of distributions and given a sufficiently high threshold, the distribution of the returns exceeding this threshold is approximately given by a Generalized Pareto Distribution (GPD). The novel model outperforms the most common linear and non-linear counterparts in the literature in terms of out-of-sample R^2. **The second chapter** also leverages information from option implied volatility, but it focuses on the entire distribution of returns. Convolutional Neural Networks, introduced by Yann Lecun [LeC+98], are fundamental pieces of state-of-art image recognition models. The geometric shape of the option's implied volatility surfaces contains information about market expectations and market participants' risk attitudes. The experimental investigation tries to capture spatial patterns and relationships between pixels in images and translate this information into a return forecast. The returns of the portfolios have a high Sharpe ratio, and they are uncorrelated with most of the existing option-implied characteristics. **Finally**, as mentioned above, when the width of a neural network goes to infinity, its evolution during training can be captured by a special kernel, the Neural Tangent Kernel, which represents an infinitely wide neural network in the lazy training regime. This discovery has revived interest in kernel methods and, thus, random features. Random features are a cheap approximation of kernels [RR07] useful when the number of data points is massive [Sha+20]. In the final chapter, we provide an algorithm, which we call Fast Annihilating Batch Regression, which is capable of solving a theoretical regression with an *infinite* amount of random features. Furthermore, we demonstrate that our approach outperforms the state-of-the-art ridge regression implemented by *sklearn*, especially with a high number of ridge penalties and features.

Outline

In this section, we provide a concise overview of each chapter, the abstract. For a more comprehensive analysis of the findings and a discussion of related literature, please refer to the respective chapters. Furthermore, each chapter introduces its own notation and is designed to be read independently.

Part-I: Tail Risk

Chapter 1: Tail Recovery [Xu23]

We use extreme value theory to study time-varying idiosyncratic tail risk for a large panel of US stocks. We demonstrate a significant performance gain by using *forward-looking* information extracted from implied volatilities and non-linear models, compared to linear models that use only *backward-looking* information. Extreme value theory plays a key role in predicting the distribution of return realizations conditional on the occurrence of a tail event. We find that, surprisingly, out-the-money calls (respectively, puts) contain important information about lower (respectively, upper) tails. Furthermore, we find evidence that the asymmetric nature of the *negative* tail distribution in comparison to the *positive* tail is captured by non-linear models only.

Part-II: State Contingent Risk Premia

Chapter 2: Deep Learning from Implied Volatility Surfaces [Kel+23]

We develop a novel methodology for extracting information from option implied volatility (IV) surfaces for the cross-section of stock returns, using image recognition techniques from machine learning (ML). The predictive information we identify is essentially uncorrelated with most of the existing option-implied characteristics, delivers a higher Sharpe ratio, and has a significant alpha relative to a battery of standard and option-implied factors. We show the *virtue of ensemble complexity*: Best results are achieved with a large ensemble of ML models, with the out-of-sample performance increasing in the ensemble size, saturating when the number of model parameters significantly exceeds the number of observations. We introduce *principal linear features,* an analog of principal components for ML and use them to show *IV feature complexity:* A low-rank rotation of the IV surface cannot explain the model performance. Our results are robust to short-sale constraints and transaction costs.

Part-III: Random Features

Chapter 3: A Simple Algorithm For Scaling up Kernel Methods [XKM23]

The recent discovery of the equivalence between infinitely wide neural networks (NNs) in the lazy training regime and Neural Tangent Kernels (NTKs) [JGH18] has revived interest in kernel methods. However, conventional wisdom suggests kernel methods are unsuitable for large samples due to their computational complexity and memory requirements. We introduce a novel random feature regression algorithm that allows us (when necessary) to scale to virtually *infinite* numbers of random features. We illustrate the performance of our method on the CIFAR-10 dataset.

Contents

I Tail Risk 21

1 Tail Recovery 23
1.1 Introduction . . . 24
1.2 Literature Review . . . 25
1.3 Extreme Value Theory and Generalized Pareto Distribution . . . 27
1.4 Data and Feature Construction . . . 29
1.5 Estimation Procedures and Measures of OOS Predictability . . . 32
1.5.1 Do Stock Returns have Power Law Tails? . . . 33
1.5.2 Neural Network Models . . . 34
1.5.3 Log-Likelihood . . . 35
1.5.4 Benchmarks . . . 36
1.5.5 Measuring OOS Predictability . . . 36
1.6 Forecasting Tail Events using Backward-Looking Measures . . . 38
1.7 Forecasting Tail Risk Using Option-Implied Information . . . 39
1.7.1 Calls, Puts, Upper, and Lower Tails . . . 40
1.7.2 Non-linearity, Big Data, and Extreme Value Theory . . . 42
1.7.3 Tail risk during Earnings Announcements . . . 43
1.8 Which features matter? . . . 44
1.9 Conclusion . . . 48
1.10 Appendix - CBOE Rule filings that potentially enhance the informativeness of equity options . . . 49
1.10.1 Appendix - Features description . . . 51
1.11 Appendix - IV Bucket Correlation . . . 52

II State Contingent Risk Premia 55

2 Deep Learning from Options Implied Volatility 57
2.1 Introduction . . . 58
2.2 Related Literature . . . 60
2.3 Data and methodology . . . 62
2.3.1 Data . . . 62

2.3.2 Ensembles of Randomly Initialized Neural Nets 63
2.4 Convolutional Neural Networks 64
2.4.1 CNN Architecture 65
2.4.2 Training the CNN 66
2.5 CNN Portfolio Performance 66
2.5.1 Simpler Models 68
2.5.2 Long only 68
2.5.3 Transaction and Short-Sale Costs 69
2.6 Principal Linear Features 70
2.7 Conclusion 73
2.8 Appendix - Data Preprocessing 75
2.9 Appendix - More about Convolutional Neural Networks 76
2.9.1 The Convolution Function 76
2.9.2 The Activation Function 77
2.9.3 The Max-Pooling Function 77
2.9.4 The Batch-Normalization Function 78
2.9.5 The Global Average Pooling Function 78
2.10 Appendix - Results 79
2.10.1 Long-only Portfolio Performance 97
2.10.2 The Impact of Costs 99
2.11 Appendix - Additional Results 100
2.11.1 Ridge Regression Results 100
2.11.2 Comparison: Simple NN against CNN 106
2.12 Appendix - Additional Analysis for Different Size Groups of Stocks 108
2.13 Appendix - Proofs 108

III Kernel Methods 123

3 A Simple Algorithm For Scaling Up Kernel Methods 125
3.1 Introduction 126
3.2 Related Work 126
3.3 Random Features Ridge Regression and Classification 127
3.3.1 Dealing with High-Dimensional Features 128
3.3.2 Dealing with Massive Datasets 129
3.4 Numerical Results 130
3.4.1 A comparison with sklearn 130
3.4.2 Experiments on Real Datasets 132
3.5 Conclusion and Discussion 137
3.6 Appendix - Proofs 138
3.7 Appendix - Additional Experimental Results 138

4 Conclusion **141**
4.1 Working Experience . . . 145
4.2 Education . . . 146
4.3 Published Papers . . . 146
4.4 Teaching . . . 147
4.5 Talks . . . 147
4.6 Skills . . . 147

Part I

Tail Risk

1. Tail Recovery

1.1. Introduction

Option prices contain forward-looking information about risk preferences and beliefs of market participants. If market participants are sufficiently rational, these beliefs are informative about the probabilities of future states. Recovering these true probabilities from asset prices is one of the most important problems in financial economics. While most existing papers focus on the bulk of the return distribution, in this paper, we focus on "tail recovery": The idea that out-of-the-money (OTM) options prices should contain information about probabilities of tail risks; that is, the likelihood (and the distribution) of large, unexpected moves in the underlying.

The problem of forecasting tail risk can be decomposed into (1) predicting the likelihood of a tail event and (2) predicting the distribution of extreme return realizations conditional on the occurrence of a tail event. Our extensive experiments with the data suggest that (1) is not possible in our dataset. Neither backward-looking nor forward-looking (option-implied) information has any forecasting power out of sample for the probability of tail events. By contrast, we find strong evidence that the distribution of returns upon the arrival of a tail event can be efficiently predicted out of the sample, and the performance of our predictions, measured by the out-of-sample (OOS) r-squared, is comparable to that for forecasting realized volatility.

To estimate tail risks, we use extreme value theory that guarantees that return distribution in the tails (over a sufficiently high threshold) is always given by a power law. We set this threshold manually at two standard deviations of returns, ensuring that tail events occur approximately 5% of the time.[1] By the classic theorem of [BDH74] and [Pic75], the tail is approximately given by the generalized Pareto distribution (GPD) pinned down by two key characteristics: the shape parameter ξ (the reciprocal of the power law exponent) and the scale parameter σ that together determine the mean and dispersion of tail returns. We leverage the power of a large cross-sectional dataset and follow the approach of [GKX20b] and [Did+23], assuming that the potentially complex and nonlinear relationship between tail risk and stock characteristics is universal across stocks. We estimate the functional dependence of the GPD parameters on stock characteristics on a rolling window and test their performance in OOS prediction of the distribution of log returns above the two-sigma threshold. We find that OOS r-squared is high and stable over time (at weekly horizon) over the 15-year period in our data sample. Our key empirical findings can be summarized as follows[2]:

- OTM Calls (respectively, Puts) contain important information about lower (respectively, upper) tails. For example, a predictive model that only uses OTM Call implied volatilities (IVs) attains a high OOS r-squared for

[1] This approach is similar to that in [KJ14], who pick the threshold to be the 5th percentile of the cross-section distribution of returns.

[2] We provide full access to our Tail Recovery Github link, which contains the codebase for our research project.

predicting lower tail risk, similar to an analogous model that only uses OTM Puts IV. The same is true for upper tails.

- Predicting upper tails is easier than predicting lowering tails: Out-of-sample r-squared is significantly higher and more stable over time for the former than for the latter.

- Big data and non-linear models play an important role in efficient return estimation: a simple OLS trained using *backward-looking* (momentum) information underperforms both Lasso and a deep neural network (DNN) performance based on *forward-looking* (implied volatility) information. Finally, Lasso underperforms slightly a DNN model, but it is not "aware" of the *negative* and *positive* tails distribution asymmetry.

- *Ceteris paribus*,[3] a DNN trained to maximize the conditional log-likelihood of log return exceedances outperforms a DNN trained to minimize the mean-squared loss. This discovery paves the way for novel approaches to examining log return exceedances through the lens of extreme value theory.

- The magnitude of predicted tail risk builds up monotonically during the two weeks preceding earnings announcements and abruptly drops once the information is released. Importantly, this happens even if the model is trained on data that excludes earnings announcement dates. Several days before the earnings announcement, the model can successfully identify stocks that do indeed experience an abnormal move on the event. Furthermore, the model exhibits a remarkable ability to predict the distribution of tail returns on earnings announcement days.

- Following [Kel+23], we study option implied information relevance. We find that both Put and Call have predictive power on tail risks (independently if *negative* or *positive*). Moreover, coherent with our prediction horizon, short option maturities are preferred.

1.2. Literature Review

Our paper is related to the growing literature on the recovery of the probability distribution of future returns from option prices, partially motivated by the [Ros15b] recovery theorem. Several papers test different versions of the recovery theorem empirically. [JLP19] develop an extension of the recovery theorem that allows them to deal with a very broad set of models.[4] In their empirical tests, they focus on forecasting the first two moments (mean and variance) of the returns on a single asset, the S&P500 index, at monthly frequency. They find

[3]Same architecture and hyperparameters.

[4]Some papers extend Ross' analysis to a continuous time setting. See, for example, [BHS16a], [QL16], and [Wal17].

little evidence that the recovered moments contain predictive information beyond that already contained in VIX and SVIX measures (see, [Mar17]). [JM20] develop statistical tests for the recovery theorem and provide strong evidence rejecting the ability of recovered moments to forecast future realized moments of S&P500 index returns. We confirm the negative findings of [JLP19] and [JM20] by testing a plethora of models (both simple, linear regressions and complex, machine learning models such as deep neural networks) for their ability to predict the first two moments of single stock returns using a large panel of U.S. stocks. We find no evidence for OOS predictability. Similarly, [BCYG18] find no evidence of option-based predictability of moments of bond returns using options on the 30-year Treasury bond futures.[5]

A related strain of literature attempts to directly extract predictive information from option prices. [Bat00] investigates whether option implied volatility skew in S&P 500 future option prices contains predictive information about the likelihood (and the magnitude) of big downward jumps in the S&P500 index and finds no empirical evidence for this intuitive hypothesis. [BGZ11] and [BTX15b] estimate the volatility risk premium and show that this premium efficiently forecasts monthly expected returns on S&P500. [Mar17; MW19] provide evidence that option prices contain information about expected stock returns at long horizons (beyond one month).

[BT11] and [BTX15b] were among the first to investigate tail risk in market returns. They find that negative jumps forecast volatility increases and provide evidence for significant time variation in the tail behavior of the S&P 500. They argue that the huge variation in the macroeconomic tail risk is hard to reconcile with simple parametric models. They also derive model-free formulas for extracting upper (lower) tail power laws from deep OTM put (call) options. While these formulas are very intuitive, we do not find strong evidence that deep OTM options are key for predicting idiosyncratic tail risk distributions for single stocks in our data. As we mention above, OTM calls are equally (and, sometimes, even better) able to forecast lower tails than OTM puts, suggesting that information about tails is spread across option moneyness in a complex fashion. Even more surprisingly, the relationship between OTM puts (calls) and upper (lower) tail risk is often positive, which is hard to reconcile with model-free formulas of [BTX15b].

[HLT20b] show that a linear model[6] in option implied volatilities from the IvyDB dataset could predict the probability of a downward (but not upward) jump in stock prices at *monthly horizon*. In this paper, we look at shorter horizons (daily and weekly) and use a different dataset. We do not find any predictability of tail event probabilities. All (linear and non-linear) models trying to predict the probabilities of a tail event on our data systematically

[5][AHL19] develop a robust statistical methodology for implementing Ross Recovery but do not test the actual predictability of stock returns by the recovered moments. However, they show that a simple trading strategy that goes long S&P500 when the recovered first moment is higher than in the previous week yields significant positive returns.

[6]Partial least squares; see [KP13] and [KP15].

produce negative out-of-sample r-squared. [7] By contrast, we find strong out-of-sample predictability for the *distribution of tail return events* conditional on the occurrence of a tail event. We show that this distribution can be robustly forecasted, with an out-of-sample r-squared of about 10% at *daily and weekly* time windows. Furthermore, we find that the size of the upper tail is easier to predict than that of the lower tail. [HLT20b] also shows that a portfolio that is short stocks with high predicted downward jump probability and long stocks with low predicted jump probability produces significant abnormal returns. In contrast, [Neu+21] shows that this strategy has significant alpha relative to many factors. Understanding the links between our predicted tail risk and various stock characteristics is an important direction for future research.

The most closely related to ours is the paper by [KJ14]. Their aggregate tail risk measure, derived from pooled stock-level extreme downside returns (below the fifth percentile) every month, can predict market excess return from one month ahead up to the five-year horizon. [KJ14] show that aggregate tail risk is an important risk factor for the cross-section of stock returns: firms with high aggregate tail risk betas[8] outperform low-tail-risk-beta stocks. They also show that their tail risk measure is correlated with standard option-based tail risk measures, such as risk-neutral skewness and kurtosis for S&P 500 index options, put/call ratio for all stock options as implied volatility slope averaged across all stocks. In contrast to [KJ14], our focus in this paper is on predicting idiosyncratic tail risk. Several papers show how probabilities of different types of extreme events can be extracted from option prices. See, for example, [KLVN16], [KPV16], and [ISV21].[9] Investigating the link between our option-implied measure of tail risk and major macroeconomic events is an important direction for future research. This paper focuses on idiosyncratic events and investigates the connection between option-implied tail risk expectations and earnings announcements. We find that the magnitude of predicted tail risk builds up monotonically during the two weeks preceding earnings announcements and abruptly drops once the information is released.

1.3. Extreme Value Theory and Generalized Pareto Distribution

The starting point of our analysis is the celebrated Pickands–Balkema–De Haan theorem ([BDH74], [Pic75]) in extreme value theory. Roughly speaking, this

[7]Surprisingly, the model of [HLT20b] is estimated on a rolling window of just two months. The findings of our study indicate that the development of robust predictive models necessitates the utilization of substantially greater volumes of data. Furthermore, the relationship found by [HLT20b] between the actual probability of a jump and the predicted has a pronounced U-shape: When sorted in deciles according to the predicted probability of a downward jump, stocks in decile ten have a higher probability of a downward jump than those in decile one (by about 5-7%); however, stocks in deciles two to eight have a lower probability of a downward jump (by about 3-4%).

[8]See also [VOZ16].

[9]See also [Alm+17] and [GKP16] for studies of tail risk and the macroeconomy.

theorem states that the distribution of tail risk exhibits a power law beyond a sufficiently high threshold. Formally, for a large class of underlying distributions, there exists a sufficiently high threshold u such that the distribution $P(R-u|R > u)$ of returns R in excess of u is approximately given by a Generalized Pareto Distribution (GPD):

$$P(R-u \leq x|R>u) \;=\; 1-\left(1 \;+\; x\,\frac{\xi}{\sigma}\right)^{-1/\xi}$$

for some $\sigma > 0$ (the scale parameter) and $\xi \in \mathbb{R}$ (the shape parameter). As is common in the literature, we only consider the case with $\xi \geq 0$.[10] It will be convenient for us to work with the so-called log exceedances, $\log(R-u)$, that has an exponential GPD (exGPD) density

$$g_{(\xi,\sigma)}(y) = \begin{cases} \frac{e^y}{\sigma}\left(1+\frac{\xi e^y}{\sigma}\right)^{-1/\xi-1} & \text{for } \xi \neq 0 \\ \frac{1}{\sigma}e^{y-e^y/\sigma} & \text{for } \xi = 0 \end{cases} \tag{1.1}$$

with a support on the whole $\mathbb{R}$.[11] Log exceedances are easier to work with for statistical estimation. For example, the classic Hill estimator of the tail parameter ξ is defined as an empirical average of log exceedances. See, e.g., [KJ14]. The mean and variance of an exGPD random variable are given by

$$E[y|\xi,\sigma] = \begin{cases} \log\left(\frac{\sigma}{\xi}\right) + \psi(1) - \psi(1/\xi) & \text{for } \xi > 0 \\ \log\sigma + \psi(1) & \text{for } \xi = 0 \end{cases}, \tag{1.2}$$

and

$$\mathrm{Var}[y|\xi,\sigma] = \begin{cases} \psi'(1) + \psi'(1/\xi) & \text{for } \xi > 0 \\ \psi'(1) & \text{for } \xi = 0 \end{cases}, \tag{1.3}$$

The function $\psi(\cdot)$ is the so-called digamma function, defined as the log derivative of the gamma function.[12] One can see that the roles of the scale parameter σ and the shape parameter ξ under exGPD are separately interpretable since the variance is determined solely by the shape parameter ξ.

Following [KJ14], we assume that tail characteristics of stock returns (i.e., σ and ξ) move over time and are given by some (unknown but learnable) functions of observable stock characteristics. We will use $\mathcal{F}_t$ to denote the information set (consisting of the history of all stock characteristics) available at time t. Furthermore, we investigate both the right tail and the left tail of the stock return distribution. Formally, we assume that there exist $\mathcal{F}_t$-measurable variables

[10]When $\xi < 0$, return support has an unnatural upper bound of $-\sigma/\xi$.

[11]If returns are normally distributed, $R \sim N(0,\sigma^2)$, then the log exceedances $y = \log(R-u)$ have the density $g^G_{u,\sigma}(y) = c\frac{1}{\sqrt{2\pi}\sigma}e^{-(e^y+u)^2/(2\sigma^2)}e^y$ where $c = 1/(1-\Phi(u/\sigma))$ where Φ is the cumulative distribution function (c.d.f.) of the standard normal distribution. Thus, tails would be extremely thin under a Gaussian distribution.

[12]$\psi(x) = \frac{d}{dx}\ln(\Gamma(x)) = \frac{\Gamma'(x)}{\Gamma(x)}$. The function $\psi'(\cdot)$ is the so-called the trigamma function satisfying $\psi'(1) = \pi^2/6 \approx 1.645$.

$u_{i,t} > 0$ such that the returns $R_{i,t+5}$ on stock i at time $t+5$ satisfy

$$\begin{aligned} P(R_{i,t+5} - u_{i,t} \leq x | R_{i,t+5} > u_{i,t} \text{ and } \mathcal{F}_t) &= 1 - \left(1 \,+\, x\,\frac{\xi^+_{i,t}}{\sigma^+_{i,t}}\right)^{-1/\xi^+_{i,t}} \\ P(-R_{i,t+5} - u_{i,t} \leq x | - R_{i,t+5} > u_{i,t} \text{ and } \mathcal{F}_t) &= 1 - \left(1 \,+\, x\,\frac{\xi^-_{i,t}}{\sigma^-_{i,t}}\right)^{-1/\xi^-_{i,t}} \end{aligned} \tag{1.4}$$

where $\xi^+_{i,t}$, $\sigma^+_{i,t}$ and $\xi^-_{i,t}$, $\sigma^-_{i,t}$ are the $\mathcal{F}_t$-measurable parameters of the conditional distribution of tail returns for upper and lower tails, respectively. We follow a reduced form approach and assume that $u_{i,t} = \kappa\, v_{i,t}$ where $v_{i,t}$ is a measure of (past) realized volatility and $\kappa = 2$, so that a "tail return" corresponds roughly to a "two-sigma" event.[13]

Our estimation strategy exploits the power of a large cross-sectional dataset of stock returns. Specifically, we assume there exist universal functions F^+_ξ, F^-_ξ, F^+_σ, F^-_σ that map the vector $X_{i,t}$ of stock characteristics (to be defined below) to its corresponding parameters ($\xi^\pm_{i,t}$ and $\sigma^\pm_{i,t}$) that govern the distributions of upper and lower tails:

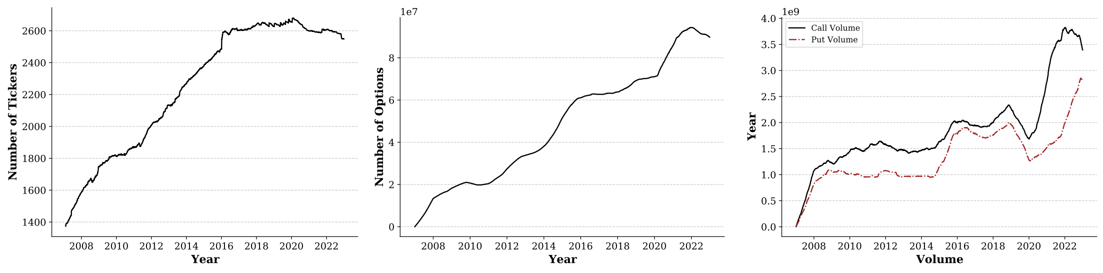
$$\begin{aligned} \xi^+_{i,t} &\equiv F^+_\xi(X_{i,t}) \\ \xi^-_{i,t} &\equiv F^-_\xi(X_{i,t}) \\ \sigma^+_{i,t} &\equiv F^+_\sigma(X_{i,t}) \\ \sigma^-_{i,t} &\equiv F^-_\sigma(X_{i,t})\,. \end{aligned} \tag{1.5}$$

To model the universal functions $F(\cdot)$ in a non-parametric fashion, we use the popular multi-layer perceptron (MLP) family (see, e.g., [GKX20b]). This fully non-parametric choice of F allows us to exploit the potentially highly nonlinear relationship between the features $X_{i,t}$ and the tail parameters $(\xi^\pm_{i,t}, \sigma^\pm_{i,t})$. In addition to DNN, we also use the least absolute shrinkage and selection operator (Lasso) to examine the redundancies of our input features as well as to investigate whether a linear (and sufficiently robust) model is sufficient to approximate the universal functions $F(\cdot)$. Once Lasso has selected the variables that "matter", we can use ordinary least squares to compute standard errors and evaluate variable significance. The details of estimation procedures are discussed in section 1.5.

1.4. Data and Feature Construction

We obtain *daily* option prices and Black-Scholes implied volatilities from Option Research and Technology Services (ORATS). ORATS covers and provides data for all US equity options quotes and implied volatilities.[14] The dataset spans the period of 2007-01-03 to 2022-12-31 and contains data for options bids, asks,

[13]We have tested other values of $\kappa \in [1.5, 3]$ and the results are similar.
[14]https://www.orats.com/.



(a) Firms (b) Options (c) Volume

Figure 1.1: The figures above describe the evolution of our dataset size and composition across time. Figure 1.1a shows the total number of different firms. Figure 1.1b shows the daily number of distinct options, while Figure 1.1c shows the daily volume of the number of contracts for put options and call options. We smooth all results with a 12-month rolling average.

volume, open interest, implied volatilities, as well as the price of the underlying asset for 5349 unique tickers traded at NYSE, AMEX, and NASDAQ .[15] We use all available stocks to avoid *survivorship bias.* Figure 1.1 shows the sample's size and composition evolution across time after our data preprocessing. Figure 1.1a shows the number of unique firms, Figure 1.1b the number of unique option contracts, and Figure 1.1c the traded volume (number of contracts) for the call- and put-options.[16]

To predict tail risk at *weekly* horizon, we define for each firm i at time t two sets of predicting variables, or features: *forward-looking* $X_{i,t}^{(f)}$ and *backward-looking* $X_{i,t}^{(b)}$. The former is constructed out of the cross-section of options, while the latter is estimated out of historical variables.

We define eleven distinct *backward-looking* features to construct $X_{i,t}^{(b)}$: a) three historical moments: standard deviation, skewness, and kurtosis, which we estimate over three different rolling time windows: one week, one month and one year for a total of nine different predictors, and b) a variable containing the numbers of days until the next earning announcement, c) a variable including the number of days until the next dividend.[17][18]

[15] Due to data limitations, the Covid period represents the only out-of-sample global financial crisis in our analysis. However, tail events may occur for individual stocks due to factors such as extremely poor earnings announcements, delisting, or sector-specific crashes, among others.

[16] Most of the existing academic literature uses the IvyDB database provided by *Option-Metrics*. We chose to use ORATS instead for two reasons: 1) unlike OptionMetrics, which is updated yearly, ORATS provides data in real-time, which allows us to fully include recent data up to and including the recent COVID crisis. 2) because the ORATS data is provided to us exactly as it is available to market participants in real-time, we are certain that our results do not suffer from look-ahead bias.

[17] The latter two variables are important because option markets may exhibit abnormal behavior around earnings announcements and ex-dividend dates.

[18] Please note that, as our predictor set requires ex-dividend dates and earnings announcements, we exclude any stocks lacking either of these elements. Consequently, ETFs and stocks

Table 1.1: Moneyness buckets.

Bucket Name	Threshold
Deep OTM	$m < -2$
OTM	$-2 \leq m < -1$

Table 1.2: Maturity buckets.

Bucket Name	Threshold
0	$0 < T \leq 5$[20]
5	$5 < T \leq 15$
15	$15 < T \leq 30$
30	$30 < T \leq 60$
60	$60 < T \leq 120$
120	$120 < T \leq 250$
250	$250 < T$

The construction of *forward-looking* predictors, $X_{i,t}^{(f)}$, is more subtle because the option data in our panel is highly imbalanced. First, working with option prices directly is inconvenient, and hence we need to use implied volatilities instead. Second, the sets of available times-to-maturity and option strike prices differ drastically across stocks. To circumvent these problems, we aggregate option implied volatilities into a small number of buckets according to their moneyness and time-to-maturity. These aggregated quantities are supposed to capture the whole shape of the implied volatility surface for any given stock.

We define option moneyness as

$$m = \frac{\ln\left(\frac{K}{S_t}\right)}{\sigma_t\sqrt{T}}, \tag{1.6}$$

where T is time-to-maturity (in calendar days), K is the strike price of the option, S_t is the underlying asset's price at time t and σ_t is a robust measure of historical standard deviation.[19]

We deliberately do not consider ATM, ITM, and deep ITM options from our analysis because they contain the least forward-looking information.[21] We also define seven different buckets grouping options by their time-to-maturity T. These moneyness-maturity buckets are shown in Table 1.1 and Table 1.2, respectively. Finally, we separately consider call and put options and use both bid and ask implied volatilities[22] for two reasons: (1) we find that bid-ask spreads do contain important information and (2) neural network performance

such as TSLA are not included in our study.

[19]We compute rolling 5-day standard deviation $\hat{\sigma}_{i,t} = \sqrt{\frac{1}{5}\sum_{\tau=1}^{5} r_{i,t-\tau}^2}$ and then define σ_t as the rolling 20-day median of $\hat{\sigma}_{i,t}$, scaled to get a daily standard deviation. Our measure is a bit non-standard and is roughly equal to the realized 20-day rolling standard deviation of stock returns, but the rolling median part makes it more robust to outliers. The results with a standard rolling 20-day standard deviation are similar and are available from the authors upon request.

[21]We have also investigated DNN models that use all options (OTM, ITM, and ATM) together, and the OOS performance deteriorates.

[22]The availability of implied volatilities for both bid and ask prices is another convenient feature of the ORATS database.

can actually increase upon inclusion of two highly correlated predictors because it helps build endogenous features inside the network. See [GBC16]. Therefore, on any given day and for each firm and each maturity-moneyness bin, we have 4 *forward-looking* features: bin-averaged implied volatilities of calls and puts for bids and asks.

When no options exist in a particular bin, we fill the feature's missing value with the closest non-empty feature along the moneyness axis. Appendix 1.10.1 provides some summary statistics for the *forward-looking* features and underlying bins.

This procedure produces 14 distinct moneyness-maturity bins, which translates into $14 \times 4 = 56$ unique features capturing the implied volatility surface for call/put and bid/ask-implied volatilities with the most forward-looking information.

This procedure clearly follows the recent findings in finance revisiting the *principle of parsimony* [Box+15] leading to the *virtue of complexity* theory: better models are found when both the number of observations and the number of features go to infinity. See the publication series [KMZ22], and [KMZ24]. Appendix 1.11 gives more insights about the implied volatility buckets correlation.[23]

1.5. Estimation Procedures and Measures of OOS Predictability

Following recent findings in empirical asset pricing (see, e.g., [GKX20b]), we rely on deep neural network models to estimate the potentially *non-linear* relationship between the input features and the predicted tail parameters. DNN models are known for their ability to approximate a wide class of functions. In fact, the universal approximation theorem[24] tells us that a neural network model with one hidden layer (i.e., a shallow network model) can potentially approximate any continuous function with arbitrary accuracy, provided that the model has a large number of neurons (the smallest unit in a neural network, which encapsulates the parameters of the model). In practice, we do not have the luxury of using an infinite number of neurons. Fortunately, with more hidden layers, the so-called deep neural network models can approximate reasonably well a wide class of functions with a finite number of parameters[25].

[23]While we do not observe any buckets 100% correlated, it is true that we might suffer from multicollinearity. Notice that these buckets are being used from our *forward-looking* models only, i.e., Lasso and DNN. Both are capable of learning to handle harmful dimensions through their penalization, the first, and learning weights, the latter. See, also [Bel21], [Sim+23].

[24]See for example, [Cyb89], [HSW89].

[25]See for example, [Pin99] [SCC18], and [AS20].

1.5.1. Do Stock Returns have Power Law Tails?

Before we proceed with modeling power laws for tail events, we would like to see some basic empirical evidence for the presence of these power laws. To this end, we first need to define the threshold for the exceedance returns. For each *day* t and each stock i, the historical volatility $v_{i,t}$ is estimated as

$$\hat{v}_{i,t} = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n} R_{i,t-j+1}^2}, \tag{1.7}$$

where we take $n = 252$ days. Then, we define our empirical tail threshold as $\hat{u}_{i,t} = 2\,\hat{v}_{i,t}$ for each stock i and for each day t. Next, we define our sample as the observations exceeding the threshold. In particular, for the right-tail estimation, we take only stock returns with $R_{i,t+5} > \hat{u}_{i,t}$; for the left-tail estimation, we take only stock returns with $-R_{i,t+5} > \hat{u}_{i,t}$.

After determining the sample of tail returns, we compute the log exceedance returns as the actual input data for our estimation. For those days and stocks with exceedance returns, we have our observations computed as

$$y_{i,t+5} \;\equiv\; \begin{cases} \log(R_{i,t+5} - \hat{u}_{i,t}) & \text{if } R_{i,t+5} > \hat{u}_{i,t} \\ \log(-R_{i,t+5} - \hat{u}_{i,t}) & \text{if } R_{i,t+5} < -\hat{u}_{i,t} \\ \text{not selected} & \text{if } |R_{i,t+5}| < \hat{u}_{i,t} \end{cases}, \tag{1.8}$$

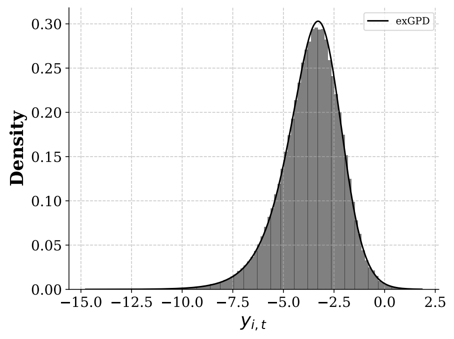
where the first row states the realized return for stock i at day $t+5$ is classified as the right-tail sample; the second row states the realized return is classified as the left-tail sample; the third row states the realized return is dropped from our estimation. Note that only one out of the three cases can happen. With daily returns in our sample, we never observe daily returns above 100% and hence $\log(R_{i,t+5} - \hat{u}_{i,t}) < \log(1) = 0$. Similarly, negative returns below -100% are simple impossible and hence $\log(-R_{i,t+5} - \hat{u}_{i,t}) < 0$. This can be seen directly from Figure 1.2, where the bulk of the support of the estimated exGPD is clearly concentrated on $\mathbb{R}_-$.

We define a set $\mathcal{S}_{+,\tau}$ as including all observations $y_{i,\tau}$ in the right-tail. Similarly, the set $\mathcal{S}_{-,\tau}$ is the collection of the left-tail observations. For ease of exposition, we also define the augmented set $\bar{\mathcal{S}}_{*,t}$ as including $\mathcal{S}_{*,t}$ and the associated features $X_{i,t-1}$ for observation $y_{i,t}$.[26]

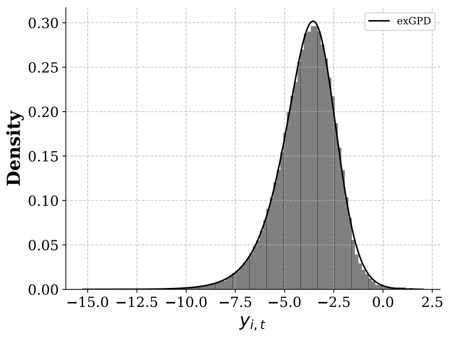
Figure 1.2 shows the empirical distributions of daily log exceedance return sets $\cup_\tau \mathcal{S}_{+,\tau}$ and $\cup_\tau \mathcal{S}_{-,\tau}$ as well as the theoretical fit of exGPD distributions.[27] As one can see, exGPD attains a remarkable fit.

[26]Thus, the augmented set $\bar{\mathcal{S}}_{\pm,\tau}$ contains pairs $(X_{i,t}, y_{i,t+5})$ for $y_{i,t+5}$.

[27]If instead we assume that $R \sim N(0,\sigma^2)$, then $y = \log(R-u)$ has the density $\frac{1}{\sqrt{2\pi}\sigma}\exp(-(e^y+u)^2/(2\sigma^2))\,e^y/c$ with $c = 1-\Phi(u/\sigma)$ with Φ being the c.d.f. of the standard normal. Clearly, this density has extremely thin tails.



(a)



(b)

Figure 1.2: The figures above show the distribution of the log exceedances $y^{\pm}_{i,t+5}$. Figure 1.2a shows the distribution of extreme negative events, while Figure 1.2b (b) shows the distribution of extreme positive events. In both figures, we show a histogram of the realized distribution and the PDF of an exGDP estimated through maximum likelihood on the whole sample. The parameters of the estimated distributions are given by $\xi_- = 0.442, \sigma_- = 0.037$ for the negative events, and $\xi_+ = 0.452, \sigma_+ = 0.029$ for positive events.

1.5.2. Neural Network Models

In this section, we discuss the architecture of our deep neural network (DNN) model. We use a DNN from the family of multi-layer perceptron (MLP),[28] which is formally defined as

Definition 1 (Multi-Layer Perceptron (MLP))**.** *Let $(n_1, \cdots, n_L)$ be the a neural network layer widths and $\theta = (W^0, b^0, \cdots, W^{(L)}, b^{(L)})$ be a collection of weights and biases. Then, we say that the neural network has L layers with $W^{(l)} \in \mathbb{R}^{n_{l+1} \times n_l}$, $b^{(l)} \in \mathbb{R}^{n_{l+1}}$, for each $l = 1, \ldots, L$. Thus, the total dimension of θ is $P = \sum_{l=1}^{(L-2)} (n_l + 1) n_{l+1} \; + \; n_L$. The MLP neural network $f(x;\theta)$ is defined as*

$$\begin{aligned} x &= input \;\in\; \mathbb{R}^d \\ y^{(l)}(x) &= \begin{cases} x & \text{if } l = 0, \\ \phi(z^{(l-1)}(x)) & \text{if } l > 0. \end{cases} \\ z^{(l)}(x) &= W^{(l)} y^{(l)}(x) + b^{(l)} \in \mathbb{R}^{n_{l+1}}, \end{aligned} \tag{1.9}$$

Where σ is an elementwise non-linear activation function. The output of the network is

$$f(x;\theta) \;=\; z^{(L)}(x) \;=\; \sum_{j=1}^{n_L} W_j^{(L)} y_j^{(L-1)}(x)\,. \tag{1.10}$$

[28]Also knows as dense network or feed-forward neural network.

In this work, the weights are initialized following [Yan+22], the biases are initialized as zeroes, and we use the hyperbolic tangent function as ϕ defined as follows

$$\tanh(x) \;=\; \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{1.11}$$

Finally, the output layer weights $W^L \in \mathbb{R}^{2\times(L-1)}$ produces the optimal parameters to approximate the optimal exGPD distribution

$$\begin{bmatrix} F_\xi(X_{i,t}|\theta) \\ F_\sigma(X_{i,t}|\theta) \end{bmatrix} = W^L y^{(L-1)} + b^{(L)} \in \mathbb{R}^2 \,.$$

We choose the DNN model with three hidden layers, with the following number of neurons in each hidden layer: $n_1 = 256$, $n_2 = 256$, and $n_3 = 64$. In order to train the model, we use a variation of the stochastic gradient descent (SGD) algorithm, which is called Adam[29], together with the most recent advances in transfer learning [Yan+22]. Every time we train a DNN model, we randomly remove 10% of the training sample to serve as a validation sample.[30] We train five different DNN models with learning rates $\in [2^{-3},\dots,2^{1}]$. We then select the optimal learning rate that produces the highest r-squared score on the validation sample.[31] We retrain our model using a one-year rolling window. Each time a new month begins, we retrain the neural network from scratch to prevent overfitting past historical market behavior. It is known that neural networks 'remember' the past through the gradient steps taken to reach the current weights, see [HS97]. Because the loss function is highly non-convex[32], the DNN results reported in this work follows [Kel+23], i.e., the DNN forecast are averaged across ten different and independent runs.

1.5.3. Log-Likelihood

We train the DNN model to maximize the conditional log-likelihood function generated by our DNN model for *weekly* log exceedances:

$$\mathcal{L}(F^*_\xi(\cdot), F^*_\sigma(\cdot)|\bar{\mathcal{S}}_{*,\tau}) = \sum_t \sum_{i:\; y_{i,t+5}\in\mathcal{S}_{*,t+5}} \log g_{(F^*_\xi, F^*_\sigma)}(y_{i,t+5}|X_{i,t}), \tag{1.12}$$

where the $F^*_\xi(\cdot)$ and $F^*_\sigma(\cdot)$ are defined[33] in (1.5), representing the universal function to produce $\xi^{\pm}_{i,t}$ and $\sigma^{\pm}_{i,t}$ for each firm i and each day t in our samples $\mathcal{S}_{\pm,\tau}$. The universal function will be modeled here as deep neural networks. The probability density function $g_{(\xi,\sigma)}(\cdot)$ is defined in (1.1), representing the exGPD density function.

[29] See [KB14].
[30] It is a random sample. We do not consider the time ordering.
[31] Batch size is set to 128 and number of epochs to 20.
[32] See, [LPB17].
[33] The superscript $*$ again denotes whether the universal function is for the right-tail or the left-tail.

Note that our log-likelihood function definition assumes that all observations (weekly log exceedance returns) are conditionally independent, conditional on the information at time t summarized by $X_{i,t}$.

1.5.4. Benchmarks

Lasso. In addition to the complex DNN model, we also consider a simple model that is linear in the input features $X_{i,t}$ and test its ability to forecast log exceedance returns $y_{i,t+5}$. To do so, we employ Lasso[34] which minimizes the following loss function

$$\mathcal{L}(\theta|\bar{\mathcal{S}}_{*,\tau}) = \sum_{y_{i,t+5}\in\mathcal{S}_{*,\tau}} \left(y_{i,t+5} - \hat{y}_{i,t+5}(\theta|X_{i,t})\right)^2 + \lambda\sum_{j=1}^{n_1}|\theta_j|\,, \tag{1.13}$$

where the set of observations $\bar{\mathcal{S}}_{*,\tau}$ include all tail events returns $y_{i,t+5}$ and associated features $X_{i,t}$ before time τ.

Similarly to the neural network model, we train five different lasso models with $\lambda = [1e^{-6}, 0.005, 0.01, 0.015, 0.2]$.[35] and selected the optimal λ that produces the highest r-squared score on the validation sample.

Autoregressive (AR) models. AR models are still widely used in forecasting time series. Therefore, it is natural to add an autoregressive model to our benchmark. We consider an AR model of the type

$$y_{i,t_{jump+1}} \;=\; \alpha + \beta y_{i,t_{jump}}, \tag{1.14}$$

where $y_{i,t_{jump+1}}$ and $y_{i,t_{jump}} \in \mathbb{R}$ are the future and past log return exceedances, respectively.

1.5.5. Measuring OOS Predictability

It is known (see [WG08]) that many predictive models of stock returns that work well in a sample fail to generate predictability out of a sample. Following [WG08], we measure the performance of our predictive models using the various versions of out-of-sample r-squared. To raise the bar for our (quite complex and hence subject to the risk of overfit) models, we define several measures of OOS predictability. These measures differ from each other in the benchmark against which the predictability is evaluated. Each benchmark uses future information to a different extent and is, therefore, hard to beat, implying that, in general, we might well expect to see a negative OOS r-squared.

Our measures will allow us to disentangle a model's (1) overall performance, (2) cross-sectional performance, and (3) time-series performance. To do this, we

[34]Lasso is efficient for data suffering from co-linearity, whereby the penalization sets coefficients of less relevant predictors to zero.

[35]The grid of potential λ has been selected through trials and errors. Increasing the range or refining the grid did not improve out-of-sample performance. We used 1e-6 instead of 0 for numerical stability.

first define three sets of parameters $[\xi, \sigma]$, which we will use to construct three different benchmarks.

Let $\bar{\xi}$ and $\bar{\sigma}$ be the parameters that maximize the log-likelihood of the exponential Pareto distribution over the whole sample:

$$\bar{\xi}, \bar{\sigma} = \arg\max_{\xi,\sigma} \sum_{i=1}^{N} \sum_{t=1}^{T} \log g_{(\xi,\sigma)}(y_{i,t}), \tag{1.15}$$

where $g_{(\xi,\sigma)}(\cdot)$ is the exGPD distribution defined in equation (1.1) and $y_{i,t}$ is the log of exceedance return defined in equation (1.8).

Next, we define $\bar{\xi}_i$ and $\bar{\sigma}_i$ as the parameters which maximize the log-likelihood *over the time period* on the subsample of firm i's jump events:

$$\bar{\xi}_i, \bar{\sigma}_i = \arg\max_{\xi,\sigma} \sum_{t=1}^{T} \log g_{(\xi,\sigma)}(y_{i,t}) \tag{1.16}$$

Finally, we define $\bar{\xi}_{year}$ and $\bar{\sigma}_{year}$ as the parameters that maximize the log-likelihood on the subsample of jump events that happen during a particular year:

$$\bar{\xi}_{year}, \bar{\sigma}_{year} = \arg\max_{\xi,\sigma} \sum_{i=1}^{N} \sum_{t\in \text{year}} \log g_{(\xi,\sigma)}(y_{i,t}) \tag{1.17}$$

We use these three sets of parameters to compute three sets of predictions using equation (1.2) to finally define:

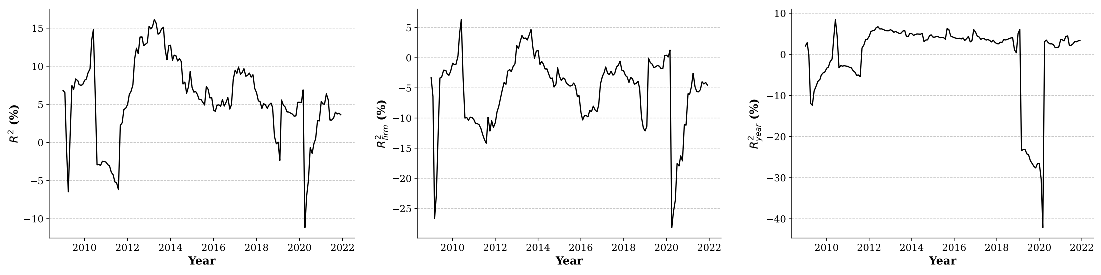
$$R^2 = 1 - \frac{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} (y_{t,i} - \hat{y}_{t,i})\right)^2}{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} \left(y_{t,i} - E[y_{t,i}|\bar{\xi}, \bar{\sigma}]\right)\right)^2}, \tag{1.18}$$

$$R^2_{firm} = 1 - \frac{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} (y_{t,i} - \hat{y}_{t,i})\right)^2}{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} \left(y_{t,i} - E[y_{t,i}|\bar{\xi}_i, \bar{\sigma}_i]\right)\right)^2}, \tag{1.19}$$

$$R^2_{year} = 1 - \frac{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} (y_{t,i} - \hat{y}_{t,i})\right)^2}{\left(\sum_{i=1}^{N} \sum_{t=1}^{T} \left(y_{t,i} - E[y_{t,i}|\bar{\xi}_{year}, \bar{\sigma}_{year}]\right)\right)^2}. \tag{1.20}$$

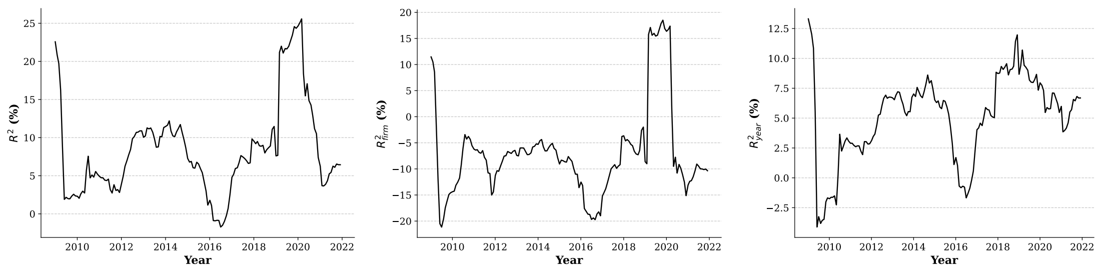
Intuitively, we expect to see $R^2 > \max\{R^2_{firm}, R^2_{year}\}$ because it uses a minimal amount of future information and hence is the easiest to beat OOS. Since there are thousands of firms in our sample, R^2_{firm} is the hardest to best because it uses future "average" information about all firms. We can view the R^2_{firm} as a version of R^2 which controls cross-sectional fixed effects, while R^2_{year} controls for yearly fixed effects.

1.6. Forecasting Tail Events using Backward-Looking Measures



(a) (b) (c)

Figure 1.3: The figures above show the out-of-sample performance of the *backward-looking* models on the subsample of *negative* tail event. Figure 1.3a shows the out-of-sample R^2 (OLS: 5.62 %) defined in equation (1.18), Figure 1.3b shows the out-of-sample R^2_{firm} (OLS: -4.01 %) controlling for cross-sectional fixed effects defined in equation (1.19), while Figure 1.3c shows the out-of-sample R^2_{year} (OLS: -1.44 %) controlling for yearly fixed effects defined in equation (1.20). In each panel, we compute the R^2 on a one-year rolling window.



(a) (b) (c)

Figure 1.4: The figures above show the performance of the *backward-looking* models as in figure 1.3 but on the subsample of positive jumps. R^2 values: R^2 (OLS: 9.78 %), R^2_{firm} (OLS: -4.64 %), R^2_{year} (OLS: 5.88 %)

In this section, we define and investigate a "backward-looking" model whereby the functions (1.5) do not depend on option-implied information. As we show below, such a model achieves low but positive r-squared OOS predictability. Our benchmark is simple OLS panel regression for log exceedances (1.8) in the sets $\mathcal{S}_{\pm,\tau}$ on the 11-dimensional vector $X^b_{i,t}$ of backward-looking features (see

Section 1.4). To get predictions out-of-sample, we estimate the OLS panel regression coefficients over a rolling window of one year.[36]

To evaluate this model performance, we estimate the out-of-sample R^2 as defined in section 1.5.5. Figure 1.3 shows the OOS r-squares computed on a yearly rolling window for the subsample of negative tail events, while Figure 1.4 shows the same performance measures on the subsample of positive tail events.

As is clear from both Figures 1.3 and 1.4, backward-looking models achieve barely positive R^2 most of the time. While the performance is slightly positive on most years when measured without any fixed effects (see (1.18)), this performance almost vanishes under the definition of performance controlling for cross-sectional or annual fixed effects (see (1.19) and (1.20)).

In the 2020 period, when the data includes observations from the COVID crisis, the OLS models achieve the best OOS performance for the *positive* subsample, whereas it stays below zero in the *negative* subsample. This suggests that historical moments can help capture the impact of systematic risk on idiosyncratic tail risks. The fact that almost no abnormal performance is measured with R^2_{year} further shows this performance comes from capturing effects that affect the whole cross-section of stocks as opposed to predicting idiosyncratic tail risk. Using the same set of information, the OLS out-of-sample performance on the positive tail (R^2: 9.8%) almost doubles the one on the negative tail (R^2: 5.6%). This is clear empirical evidence of the asymmetry in the two distributions. In the next sections, we will focus on the information set formed by option-implied information only. The predictive models, when trained with market participants' expectations, exhibit a dramatic increase in out-of-sample performance. We aim to interpret these results by analyzing the coefficients and, following [GKX20b], providing a feature heatmap.

1.7. Forecasting Tail Risk Using Option-Implied Information

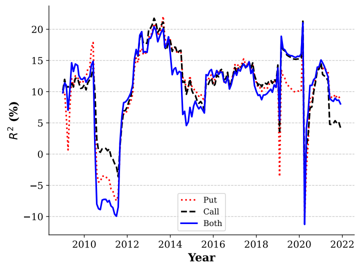
In this section, we report the results of our DNN model. We consider a DNN model that only uses out-of-the-money (OTM) option implied volatilities trained to maximize the conditional log-likelihood of log return exceedance defined in (1.12). The model is trained on a rolling window of one year. We show that our novel approach outperforms the most common predictive models for returns found in the literature, i.e., Lasso, AR, and a simple DNN trained to minimize the mean-squared loss. We conclude this section with several key takeaways: (i) non-linear models seem to better capture the asymmetry between the positive and negative tails; (ii) the model can successfully identify stocks that experience an abnormal move on earnings announcement days; (iii) it appears that most option-implied information predominantly pertains to the upper tail, while the lower tail heavily relies on put options with maturities of more than one year.

[36]The results obtained from expanding the window are slightly worse and more computationally expensive.

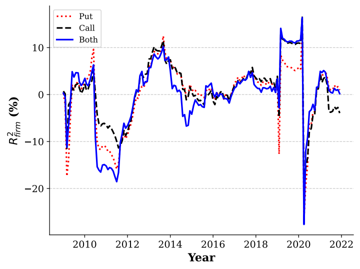
1.7.1. Calls, Puts, Upper, and Lower Tails

Figures 1.5 and 1.6 show the out-of-sample performance of our DNN model measured with the proposed definition of r-squared defined in (1.18). The predictability we find is significant, about 10% and 15% OOS, r-squared on average for R^2 when predicting negative and positive tails, respectively. Therefore, when predicting the negative tail, our nonlinear model, trained through maximum likelihood using *forward-looking* options data, doubles the performance compared to a linear model that minimizes the mean-squared error with *backward-looking* data. Figure 1.5 and Figure 1.6 show that the model achieves, on average, positive r-squared when controlling for cross-sectional fixed effects and yearly fixed effects. However, while the model has shown the ability to capture small shocks, we observe an abrupt drop of the r-squared in 2020, specifically in the negative tail.

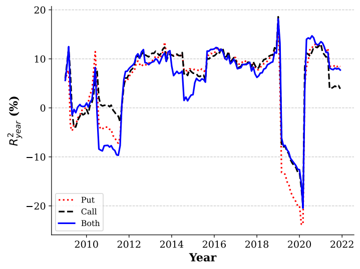
To further investigate the cross-sectional performance of the model, we report in Figures 1.9 and 1.11 the scatter plots of mean realized exceedances against predicted mean exceedances. The fit is striking, showing the model's remarkable ability to differentiate across firms with high and low tail risk. While our analysis of r-squared focuses on the mean size of log exceedances, the model also produces estimates for their standard deviations (see formulas (1.2) and (1.3)).



(a)



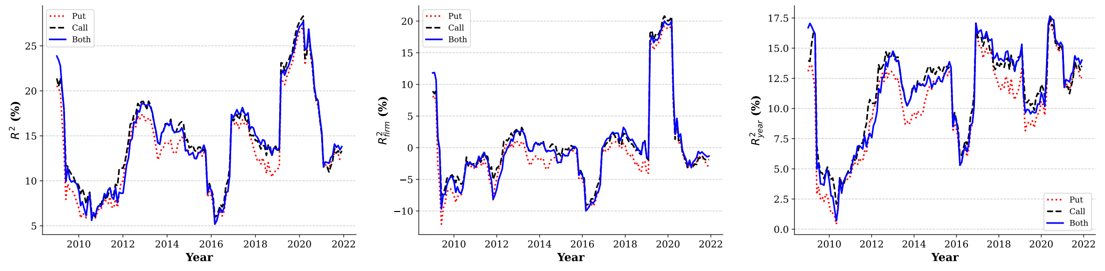
(b)



(c)

Figure 1.5: The figures above show the out-of-sample performance of the DNN on the subsample of *negative* tail event. Figure 1.5a shows the out-of-sample R^2 (Call only: 11.32%, Put only: 10.62 %, Both: 10.60 %)defined in equation (1.18), Figure 1.5b shows the out-of-sample R^2_{firm} (Call only: 2.34 %, Put only: 1.57 %, Both: 1.55 %) controlling for cross-sectional fixed effects defined in equation (1.19), while Figure 1.5c shows the out-of-sample R^2_{year} (Call only: 4.57 %, Put only: 3.82%, Both: 3.80 %) controlling for yearly fixed effects defined in equation (1.20).

Surprisingly, all three models (based on calls, puts, or both) produce comparable performance, even though OTM options of one type (call or put) completely miss the information for the strike subset corresponding to the "opposite" type. We find that a predictive model that only uses OTM call prices



(a) (b) (c)

Figure 1.6: The figures above show the performance of DNN as in figure 1.5 but on the subsample of upper tail events. R^2 values: R^2_{year} (Call only: 15.57 %, Put only: 14.40 %, Both: 15.44 %), R^2_{firm} (Call only: 2.27 %, Put only: 0.91 %, Both: 1.17 %), R^2_{year} (Call only: 11.85 %, Put only: 10.62 %, Both: 11.71 %)

attains comparable OOS r-squared for predicting lower tail risk than an analogous model that only uses OTM Put prices. The same is true for upper tails. DNN Using call prices, in particular, achieves constantly superior performance in predicting both lower and upper tail risks. This is a very surprising phenomenon. Indeed, rational models imply that deep OTM put prices contain all the relevant information about lower tails. This is formally proved in [BT11] and [BTX15b], which derive model-free formulas for extracting upper (lower) tail power laws from deep OTM put (call) options. The fact that OTM calls have similar lower tails predictive power to OTM puts suggests that the prices of these options might be hard to reconcile with standard, rational asset pricing models. To gain some insights into the nature of the dependence on tail risk on option prices, we estimate two simple, linear Lasso models: One with only OTM puts and another with only OTM calls. We then train an OLS with the statistically significant features selected by Lasso[37]. Figure 1.7 shows the dynamics of these coefficients. As one can see in Figure 1.7a and Figure 1.7b, the sum of the coefficients aligns coherently with the outcomes of the DNN models. Both tail risks nearly equally rely on information from both calls and puts, albeit with a slight preference for the former. Consistent with DNNs trained with call prices, it seems that call prices contain on average more information than put prices. However, the fact the OTM puts (calls) contain information related to upper (lower) tail expectations is surprising and is hard to reconcile with model-free formulas of [BT11] and [BTX15b] or any rational (arbitrage-free) asset pricing model.

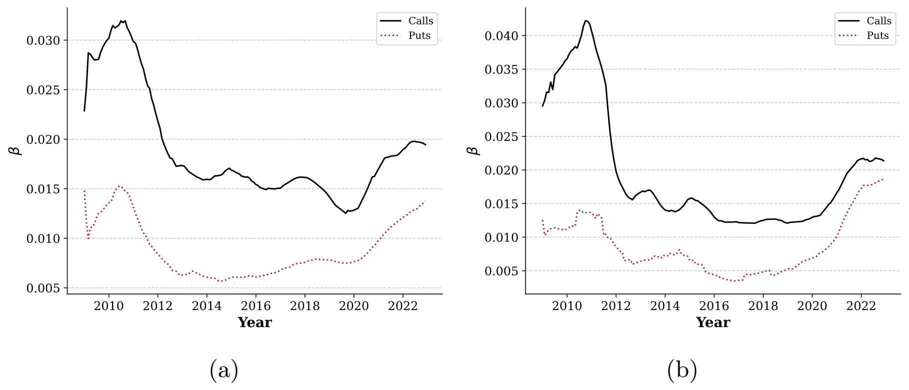


Figure 1.7: The figures above show the relative importance of the OTM puts and calls across time (Figure 1.7a is for the lower tail, Figure 1.7b is for the upper tail). Each month, we train a Lasso model with the OTM calls and puts only. We then select all the features selected by Lasso—that is, features with a non-zero coefficient—and we run an OLS regression and keep only the coefficient with p-values smaller than 5%. We finally sum the value of those coefficients by option types (put/call). In the figure above, we show the value of these sums smoothed with a moving average on a yearly rolling window. Linear models load on both call and put options. The figure above suggests that information about tails is spread across calls and puts option-implied information in a complex fashion.

1.7.2. Non-linearity, Big Data, and Extreme Value Theory

The *forward-looking* dataset that we have built and described in Section 1.4 creates big multi-dimensional option-implied features describing the cross-section. In high contrast with the classic *principle of parsimony* [Box+15] and following the *virtue of complexity* (VoC) discovered in [KMZ22], in this paragraph, we show that only complex, non-linear models can successfully utilize high-dimensional conditional information. Beginning with Figures 1.8a and 1.8b, we present scatter plots comparing the expected mean of a negative tail risk event, $E[y^-|\xi,\sigma]$ with the expected mean of a positive tail risk event, $E[y^+|\xi,\sigma]$, across the entire sample for both the Lasso and DNN models, respectively. The Lasso model exhibits a significant correlation between $E[y^-|\xi,\sigma]$ and $E[y^+|\xi,\sigma]$ of 91.69% demonstrating the inability to distinguish between negative and positive tails. Conversely, our DNN approach demonstrates a lower correlation of 80.88%, showing that it is somehow more "aware" of the differences. In terms of out-of-sample R^2 performance, Table 1.3 shows the predictive power of our main model (reported as DNN_exGPD) outperforming common models used in the literature. For the sake of brevity, we report the average performance measured in (1.18) of each model in forecasting both positive and negative tails.

[37]i.e. All features with non-zero coefficient and p-value smaller than 5%.

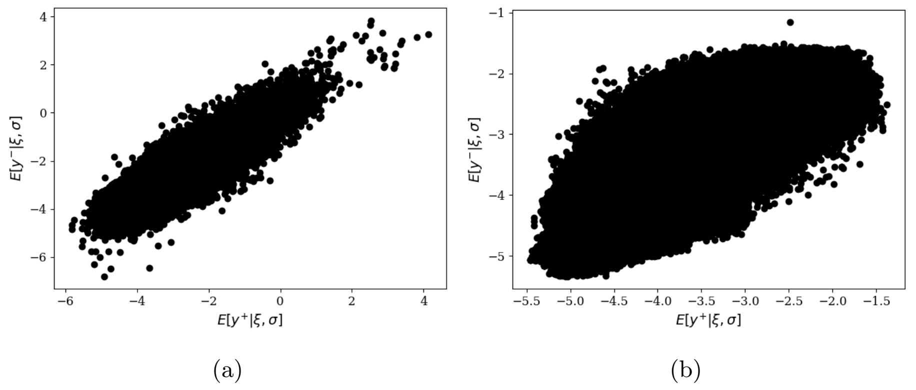


Figure 1.8: The figures above show scatter plots of the predicted mean of a negative tail risk event $E[y^-|\xi,\sigma]$ against the predicted mean of a positive tail risk event $E[y^+|\xi,\sigma]$. Figure 1.8a for the Lasso and Figure 1.8b for the DNN. Figure 1.8b shows that the DNN is able to distinguish more between negative and positive tail events than Lasso.

We can see that extracting option-implied information is not trivial. To start, a non-linear model (DNN_MSE) trained to minimize the mean-squared error slightly outperforms a linear model (OLS) trained with *backward-looking* information only, but it is outperformed by a linear model (LASSO) when trained with *forward-looking* data. When we plug our extreme value theory, and thus, the non-linear model (DNN_exGPD) is trained to maximize the conditional log-likelihood of an exponentiated generalized Pareto distribution, we achieve the best performance. At the bottom, an autoregressive model shows that the information of previous log return exceedances achieves a positive R^2 although the worst. Similar to the previous results, Table 1.3 shows the performance of each model when trained with the set of all information (Call & Put), with call-only option-implied information (Call), or with put-only option-implied information (Put). The last column, $y_{i,t_{jump}}$, corresponds to the information of a previous log return exceedance.

1.7.3. Tail risk during Earnings Announcements

We complete this section with a discussion of the DNN during the earning announcement dates. Many of the big moves in stock prices occur on earnings announcement dates when important fundamental news about companies is released. It is, therefore, important to see whether our model is able to learn this relationship from option prices. Figures 1.9 and 1.11 show scatter plots of realized returns log exceedances (1.8) and their predicted means (1.2) computed with the estimated exGPD-DNN model.[38] As one can see, the model achieves

[38]The model is trained on the full sample, including both regular and earnings announcement dates. Interestingly enough, a model trained only on regular dates produces exactly the

Table 1.3: The table below shows the out-of-sample performance measured as vanilla R^2 of our main model, DNN_exGPD) trained to maximize the conditional log-likelihood of log returns on the subsample of *negative* and *positive* tail events using option-implied information. We compare its performance with a simple DNN trained to minimize the classic mean-squared error, a Lasso model, and an autoregressive model. While DNN_exGPD, DNN_MSR, and Lasso use implied volatility information from either call, put, or both options, the autoregressive model only uses the previous log return exceedance to forecast the next one. Notice that our extreme value theory not only adds performance gain when compared to, *ceteris paribus*, other non-linear models but adds "awareness" about incoming earnings announcements and positive/negative tail asymmetry compared to the Lasso model.

	Call & Put	Call	Put	$y_{i,t_{jump}}$
DNN_exGPD	**12.94** %	**13.39 %**	**12.37** %	
DNN_MSE	6.09 %	9.15 %	7.92 %	
Lasso	12.93 %	12.47 %	12.29 %	
AR	-	-	-	4.29 %

a remarkable fit with a correlation of about 35% between the predicted and realized tail moves. The next interesting question is: Do option markets anticipate large moves on earnings announcement dates? To answer this question, we plot in Figure 1.10 the average predicted log exceedances over the earnings cycle, aggregating them as a function of the number of days to the nearest earnings announcement. Remarkably (and fully consistent with the conventional wisdom), the DNN finds that option prices imply a continuous, monotonic build-up of expectations of a large move over about two weeks preceding the announcement. These expectations immediately drop after the information is released.

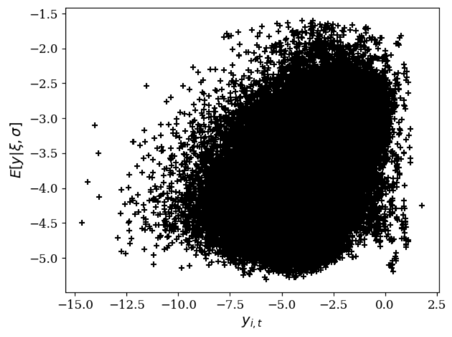
1.8. Which features matter?

For both negative and positive tails, we investigate the relative importance of each feature. Following [GKX20b], we compute the reduction in R^2 from setting all values of a given feature to zero within each test sample. Formally, we define the importance of feature j as,

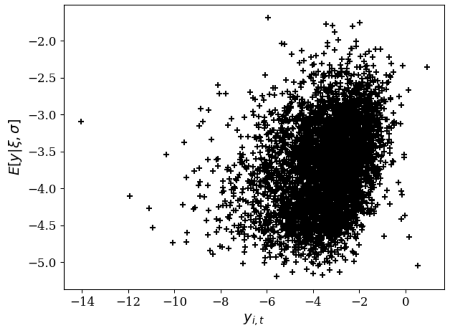
$$IMP(j) = |R^2(f(X)) - R^2(f(\tilde{X}_j))|, \tag{1.21}$$

Where $R^2(\cdot)$ is an out-of-sample measure of performance defined in equation (1.18). $f(\cdot)$ is a function mapping the inputs to a prediction In this case, $f(\cdot)$ is the mean log exceedance predicted with equation (1.2) and the DNN. X is the matrix of inputs, and $\tilde{X}_j$ is similar to X except for columns j which is set to 0.

same results on earnings announcement dates. This suggests that the relationship between option prices and tail events is universal.



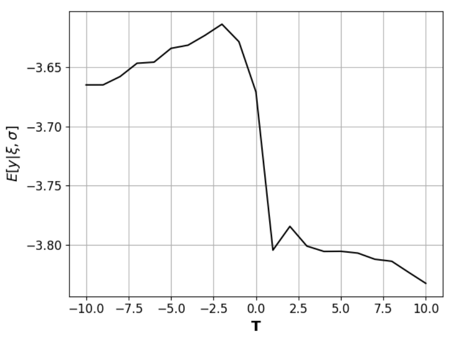
(a)

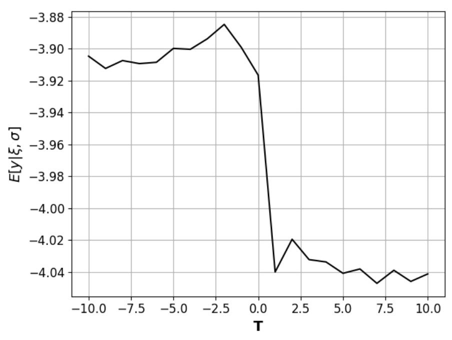


(b)

Figure 1.9: The figures above show scatter plots of the average predicted mean of a **negative** tail event $E[y|\xi, \sigma]$ against the realized exceedances on days with a negative tail event. Figure 1.9a shows this projection on the full sample (correlation: 34.5%), while Figure 1.9b shows this projection on the subsample of tail events occurring on an earning announcement day (correlation: 30.4%).

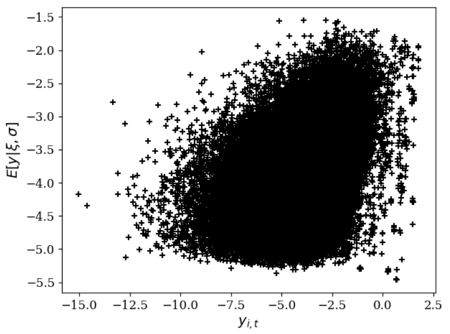
Figure 1.12 shows the relative importance of all 56 features when predicting a negative tail, shown in Figure 1.12a, and a positive tail, shown in Figure 1.12b. As in [GKX20b], we improved interpretation by normalizing the variables' importance to sum to one $\left(I\hat{M}P(j) = \frac{IMP(j)}{\sum_{k=1}^{2} 0IMP(k)}\right)$. Several important patterns emerge from these figures. First, no single feature can fully explain the predictive power of the DNN. Among all features, the highest normalized importance is as lower than 25% which shows that the DNNs do not just transform one single measure of implied volatility but instead use a large part of the cross-section. Second, surprisingly, the algorithms utilize a mixture of information from across the entire cross-section to predict the positive tail. Third, the model seems to favor implied volatility coming from out-the-money puts with expiration dates over one year. Finally, it appears that out-of-the-money puts with expiration dates exceeding one year are the most important features for predicting negative tails with a normalized $\hat{IMP}(j) > 50\%$ when combined (bid and ask). This is additional empirical evidence about the asymmetry of the two tails and an interesting direction for future research: not only are negative tails harder to forecast, but it seems that the cross-section (other than out-the-money puts) of features has little information about that.

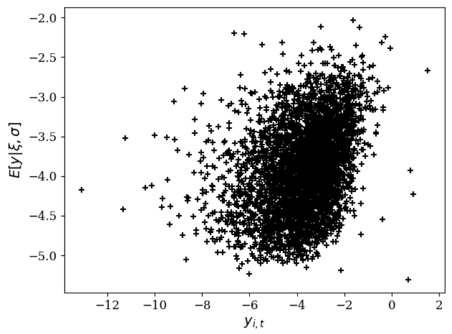




(a) (b)

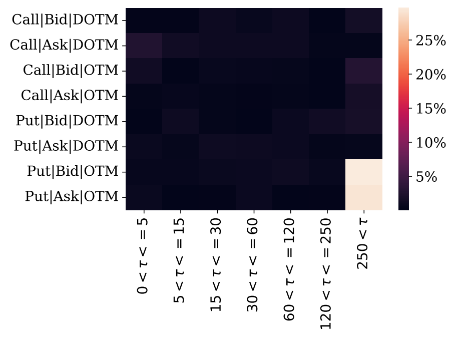
Figure 1.10: Figure 1.10a and Figure 1.10b show the average predicted mean $E[y|\xi,\sigma]$ for the **negative**and **positive** tail events, respectively. We define t=0 as the day on which earnings are announced. We compute the average of the DNN's forecasted moments across all assets and time periods. Consistent with the conventional wisdom, the DNN finds that option prices imply a continuous, monotonic expectation of a large move preceding the announcement.



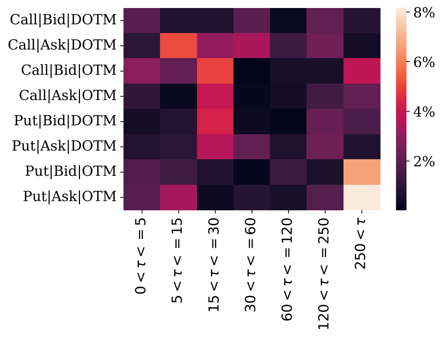


(a) (b)

Figure 1.11: The figures above show the average predicted mean of a **positive** event $E[y|\xi,\sigma]$ projected against the realized exceedances on days with a positive tail event. Figure 1.11a shows this projection on the full sample, while Figure 1.11b shows this projection on the subsample of tail events occurring on an earning announcement day. The correlation on the full sample is equal to 39.4%, while that on the earning announcement subsample is equal to 33.9%.



(a)



(b)

Figure 1.12: Figure 1.12a and Figure 1.12b show the normalized features importance heatmap for the negative and positive tails, respectively. Surprisingly, while the positive tail seems to extrapolate information across all maturities and option types, the negative tail is mostly influenced by the out-of-the-money puts with the largest time to maturity τ.

1.9. Conclusion

In this paper, we have investigated the ability of complex machine learning models to extract predictive information about tail risk, that is, the likelihood (and the distribution) of large, unexpected moves in the underlying asset prices. We find that the distribution of returns upon the arrival of a tail event can be efficiently predicted out of sample, and the performance of our predictions is comparable to that for forecasting realized volatility. Our dataset covers the period of 2007-01-03 to 2022-12-31 and contains data (Black-Scholes implied volatilities, underlying asset prices, volume, open interest, asks, bids) for 5349 different stocks traded at NYSE, AMEX, and NASDAQ at *daily* frequency. We use the classic theorem of [BDH74] to approximate the tail distribution to a generalized Pareto distribution, and we follow the approach of [GKX20b] to conduct our studies. First, options information about tails is spread across option moneyness in a complex fashion; in fact, we find that OTM calls (respectively, puts) contain important information about lower (respectively, upper) tails. Second, we find that predicting upper tails is easier than predicting lower tails. Third, upper- and lower-tail risks can load positively on puts information and call information, respectively. Then, we provide a new non-linear predictive model capable of outperforming the most common linear and non-linear counterparts in the literature through extreme value theory. Next, we find that the magnitude of predicted tail risk builds up monotonically during the two weeks preceding earnings announcements and abruptly drops once the information is released. Finally, we show how both put and call information are important for deep neural networks in predicting tail risks, and we find new evidence in equity premia's non-linear nature. Understanding the links between our predicted tail risk and various stock characteristics is an important direction for future research. Moreover, the importance of OTM call (puts) information about the lower (upper) tail is surprising, opening new challenges for both theoretical and empirical studies.

1.10. Appendix - CBOE Rule filings that potentially enhance the informativeness of equity options

Below is a timeline of CBOE rule filings that are closely related to the informativeness of equity options markets. We split them into several categories:

- Number of weekly classes (options on distinctive equities): regular option series expire on the third Friday of each month. However, that may not coincide with particular dates for corporate events (which are important risk sources for equity options). Hence, investors gain from being able to trade options that expire every week in the next five weeks.
  - CBOE proposed weekly option programs for index options back in 2005 (SEC pilot program)
  - CBOE then introduced weekly options for ETF and equities in 2010
  - On 12/12/2011, the filing SR-CBOE-2011-125 increases the number of classes from 25 to 30
  - On 03/01/2012, the filing SR-CBOE-2012-026 allows CBOE to open new classes that are chosen by the other competitive exchanges (NASDAQ, ISE by then)
  - On 12/11/2013, the filing SR-CBOE-2013-121 increase the number of classes to 50
- Maturities and strikes per weekly class
  - On 09/19/2011, SR-CBOE-2011-086 allows CBOE to increase the number of options per class from 20 to 30 for the weekly program
  - On 11/08/2012, SR-CBOE-2012-110 allows CBOE to expand the number of expirations available under the weekly program (i.e., freely delist unpopular options, and add others)
  - On 10/10/2012, SR-CBOE-2012-092 allows CBOE to initiate strike prices in more granular intervals for weekly options
  - On 07/02/2014, SR-CBOE-2014-052 allows CBOE to introduce finer strike price intervals for standard expiration (third Friday) contracts in option classes that also have weeklies
  - On 01/21/2015, SR-CBOE-2015-009 allows CBOE to extend current \$0.50 strike price intervals in non-index options to short-term options with strike prices less than \$100
- Margin

  - On 05/29/2012, SR-CBOE-2012-043 allows CBOE to implement a universal methodology for determining a spread margin requirement that would accommodate the many types of spread strategies in use today, and which will enable a wider variety of multi-leg option spreads, including numerous variations of butterfly and condor spreads, to be accommodated

- Mini options
  - On 01/04/2013, SR-CBOE-2013-001 allows CBOE to list and trade options contracts overlying 10 shares of a security ("mini-option contracts")

To sum up, all the rule changes allow CBOE customers to trade options that have finer strikes and expiration dates, under cheaper capital costs (due to portfolio margin for spread strategies). In the time series of equity options volumes, we shall expect to see these options slowly become popular over time since 2013. Mini options and options with shorter maturities allow retail investors to express their views or speculate on equity options. Today, there are 610 different equities have weekly options being traded on CBOE.

1.10.1. Appendix - Features description

Table 1.4: The table shows the daily average number of options in a bin bucket, the number of options standard deviation in the time series, and the time series's quantiles for the following values: 0.01, 0.25, 0.50, 0.75, 0.99. Please be advised that the raw option prices we receive from ORATS are indexed by strike. his indexing method results in the observed symmetry in the distribution of option counts across each bin.

right	moneyness	maturity	mean	std	q=0.01	q=0.25	q=0.50	q=0.75	q=0.99
put	atm	120.0	12973.49	7814.72	3645.19	7654.75	10997.5	15792.25	40390.36
call	atm	120.0	12973.49	7814.72	3645.19	7654.75	10997.5	15792.25	40390.36
put	deep_otm	15.0	12516.14	8268.44	10.00	5316.50	11446.5	19172.50	30814.10
call	deep_itm	15.0	12516.14	8268.44	10.00	5316.50	11446.5	19172.50	30814.10
put	deep_otm	5.0	12319.11	8844.63	5.00	5517.25	10194.0	19445.50	34101.36
call	deep_itm	5.0	12319.11	8844.63	5.00	5517.25	10194.0	19445.50	34101.36
put	atm	60.0	11029.50	5201.96	3202.29	7457.75	9866.5	13308.50	24969.44
call	atm	60.0	11029.50	5201.96	3202.29	7457.75	9866.5	13308.50	24969.44
put	deep_itm	5.0	10879.69	7702.11	9.00	5226.50	8720.0	17475.00	30075.10
call	deep_otm	5.0	10879.69	7702.11	9.00	5226.50	8720.0	17475.00	30075.10
put	atm	15.0	10847.75	7632.36	6.00	3897.00	10734.0	16208.00	31167.44
call	atm	15.0	10847.75	7632.36	6.00	3897.00	10734.0	16208.00	31167.44
put	deep_itm	15.0	10439.18	6933.60	17.00	4365.50	8606.0	16291.50	26691.78
call	deep_otm	15.0	10439.18	6933.60	17.00	4365.50	8606.0	16291.50	26691.78
call	deep_itm	60.0	9821.10	5808.45	452.56	5249.50	9998.0	13422.75	25551.82
put	deep_otm	60.0	9821.10	5808.45	452.56	5249.50	9998.0	13422.75	25551.82
call	deep_itm	30.0	9569.36	6972.71	447.68	3889.75	7693.0	14827.00	27508.77
put	deep_otm	30.0	9569.36	6972.71	447.68	3889.75	7693.0	14827.00	27508.77
put	atm	250.0	9026.53	6010.77	2205.21	4828.00	6819.5	11672.25	27466.25
call	atm	250.0	9026.53	6010.77	2205.21	4828.00	6819.5	11672.25	27466.25
put	deep_otm	0.0	8198.83	7417.50	15.46	3498.00	6288.0	9699.50	30228.08
call	deep_itm	0.0	8198.83	7417.50	15.46	3498.00	6288.0	9699.50	30228.08
call	atm	30.0	7510.54	4796.33	859.78	3545.25	7041.5	10124.25	21334.77
put	atm	30.0	7510.54	4796.33	859.78	3545.25	7041.5	10124.25	21334.77
call	deep_otm	0.0	7492.82	6581.73	27.00	3779.00	5908.0	8826.00	28021.16
put	deep_itm	0.0	7492.82	6581.73	27.00	3779.00	5908.0	8826.00	28021.16
put	deep_itm	30.0	7349.35	5440.12	513.29	3005.50	5753.5	11200.50	22526.07
call	deep_otm	30.0	7349.35	5440.12	513.29	3005.50	5753.5	11200.50	22526.07
put	deep_otm	120.0	6715.75	4369.24	114.39	3185.75	6437.0	9634.75	19136.56
call	deep_itm	120.0	6715.75	4369.24	114.39	3185.75	6437.0	9634.75	19136.56
put	deep_itm	60.0	6280.38	4000.68	704.39	3064.00	5875.5	8439.25	18144.40
call	deep_otm	60.0	6280.38	4000.68	704.39	3064.00	5875.5	8439.25	18144.40
put	atm	5.0	6058.02	4304.39	3.00	2571.00	5509.0	8712.50	18276.68
call	atm	5.0	6058.02	4304.39	3.00	2571.00	5509.0	8712.50	18276.68
put	itm	60.0	5169.78	2540.22	1169.95	3330.50	4730.0	6423.25	12216.24
call	otm	60.0	5169.78	2540.22	1169.95	3330.50	4730.0	6423.25	12216.24
call	otm	120.0	4954.89	2866.67	722.78	2941.25	4580.0	6448.50	14333.64
put	itm	120.0	4954.89	2866.67	722.78	2941.25	4580.0	6448.50	14333.64
call	otm	15.0	4636.69	2906.06	4.00	1926.25	4625.0	6769.25	11144.81
put	itm	15.0	4636.69	2906.06	4.00	1926.25	4625.0	6769.25	11144.81
call	itm	120.0	4069.12	2331.37	523.00	2306.00	3659.5	5412.00	10726.44
put	otm	120.0	4069.12	2331.37	523.00	2306.00	3659.5	5412.00	10726.44

Continued on next page

Table 1.4: The table shows the daily average number of options in a bin bucket, the number of options standard deviation in the time series, and the time series's quantiles for the following values: 0.01, 0.25, 0.50, 0.75, 0.99. Please be advised that the raw option prices we receive from ORATS are indexed by strike. his indexing method results in the observed symmetry in the distribution of option counts across each bin.

right	moneyness	maturity	mean	std	q=0.01	q=0.25	q=0.50	q=0.75	q=0.99
call	deep_itm	250.0	4008.94	5525.90	68.00	936.75	2207.0	4252.50	28713.27
put	deep_otm	250.0	4008.94	5525.90	68.00	936.75	2207.0	4252.50	28713.27
put	otm	15.0	3997.30	2522.69	2.00	1627.00	4077.0	5971.00	10052.64
call	itm	15.0	3997.30	2522.69	2.00	1627.00	4077.0	5971.00	10052.64
call	itm	60.0	3994.64	1831.06	1046.78	2604.75	3780.5	4921.25	8679.88
put	otm	60.0	3994.64	1831.06	1046.78	2604.75	3780.5	4921.25	8679.88
put	itm	30.0	3546.75	2274.92	402.17	1670.50	3239.0	4790.50	10275.27
call	otm	30.0	3546.75	2274.92	402.17	1670.50	3239.0	4790.50	10275.27
call	deep_otm	120.0	3475.57	2349.73	75.95	1708.75	3177.0	4799.00	9934.14
put	deep_itm	120.0	3475.57	2349.73	75.95	1708.75	3177.0	4799.00	9934.14
put	itm	5.0	2898.31	1948.49	2.00	1336.00	2663.0	4114.00	7962.52
call	otm	5.0	2898.31	1948.49	2.00	1336.00	2663.0	4114.00	7962.52
call	itm	30.0	2839.06	1744.38	335.39	1383.00	2689.0	3877.00	7640.54
put	otm	30.0	2839.06	1744.38	335.39	1383.00	2689.0	3877.00	7640.54
call	itm	5.0	2584.95	1729.72	2.00	1169.00	2397.0	3720.00	7422.00
put	otm	5.0	2584.95	1729.72	2.00	1169.00	2397.0	3720.00	7422.00
put	deep_itm	250.0	2343.47	4699.80	17.39	261.00	744.5	1562.50	24473.16
call	deep_otm	250.0	2343.47	4699.80	17.39	261.00	744.5	1562.50	24473.16
put	itm	250.0	2337.41	1530.74	302.39	1319.75	1915.5	2948.00	7298.00
call	otm	250.0	2337.41	1530.74	302.39	1319.75	1915.5	2948.00	7298.00
call	itm	250.0	2261.27	1531.31	372.00	1131.75	1722.0	2963.50	7272.83
put	otm	250.0	2261.27	1531.31	372.00	1131.75	1722.0	2963.50	7272.83
put	atm	0.0	1718.61	1377.57	4.00	735.75	1484.0	2388.25	6203.54
call	atm	0.0	1718.61	1377.57	4.00	735.75	1484.0	2388.25	6203.54
put	itm	0.0	864.72	689.90	3.00	371.25	752.0	1196.00	3178.75
call	otm	0.0	864.72	689.90	3.00	371.25	752.0	1196.00	3178.75
call	itm	0.0	811.05	635.41	2.00	360.00	712.0	1122.50	2856.20
put	otm	0.0	811.05	635.41	2.00	360.00	712.0	1122.50	2856.20

1.11. Appendix - IV Bucket Correlation

In this section, we show the correlation of each implied volatility bucket built following the pre-processing procedure described in Section 1.10.1.

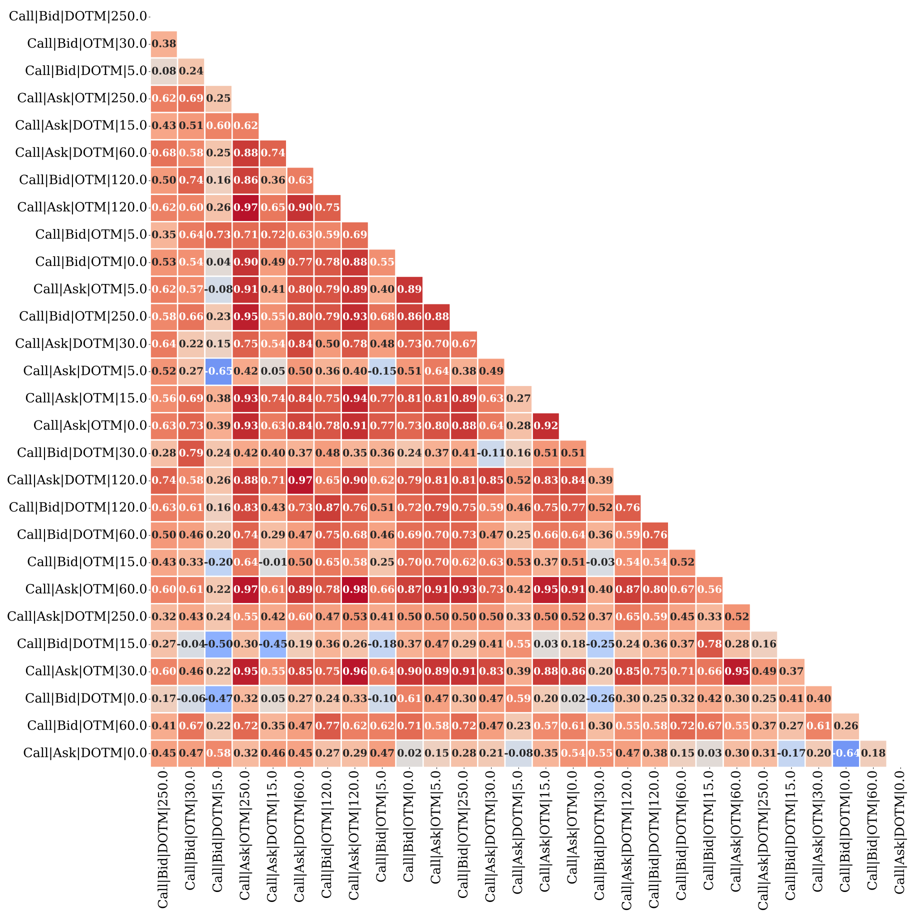


Figure 1.13: The figure above shows the correlation matrix for each maturity-moneyness bucket of call options only. We find that 59%, 27%, and 13% of the buckets exhibit correlations of 50%, 75%, and 90%, respectively. The implied volatility values are first averaged on a cross-sectional basis.

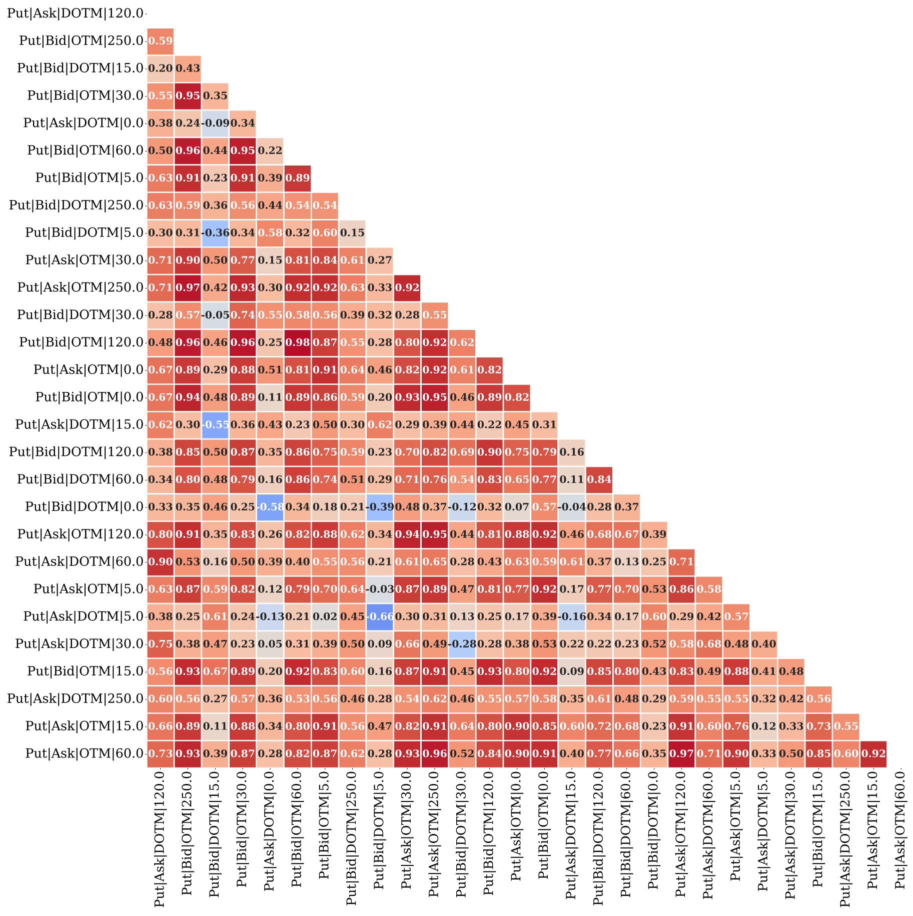


Figure 1.14: The figure above shows the correlation matrix for each maturity-moneyness bucket of call options only. We find that 59%, 34%, and 17% of the buckets exhibit correlations of 50%, 75%, and 90%, respectively. The implied volatility values are first averaged on a cross-sectional basis.

Part II

State Contingent Risk Premia

2. Deep Learning from Options Implied Volatility

2.1. Introduction

The Option Implied Volatility (IV) surface contains information about state-contingent risk premia and the probability distribution of returns over multiple horizons. Stock and option traders systematically use the shape of this surface, including its slope, skewness, and other geometric features, to infer market expectations and risk attitudes and make trading decisions. The option pricing theory provides a basis for such analysis: local properties of this surface indeed contain information about the underlying market dynamics. For example, the surface variation along the moneyness dimension can be used to uncover the Arrow-Debreu state prices [BL78a], while the surface slope along the maturity dimension contains information about volatility by the Dupire formula [Dav11]. In the language of machine learning, universal *local* features (non-linear transformations that depend on neighboring strikes and maturities) of the IV surface can be used to extract useful information about the stochastic structure of returns.

Despite the underlying theory's elegance, empirically computing the theory-driven features is associated with often unsurmountable econometric difficulties: theoretical formulas rely heavily on the continuity of strike and maturity dimensions, while, in reality, both strikes and maturities are discrete, living on a sparse grid. Furthermore, high bid-ask spreads due to often extreme illiquidity of options markets[1] introduce large amounts of noise into the estimation of equity implied volatilities.

In this paper, we leverage the progress in deep learning for computer vision and image recognition to construct powerful, non-linear, local features of IV surfaces. To this end, we exploit a particular neural network architecture called convolutional neural networks (CNNs). Thanks to their use of convolutional layers, CNNs are designed to capture spatial patterns and relationships between pixels in images. These layers apply filters to local regions of the input image, allowing the network to learn and identify patterns at various levels of abstraction (e.g., edges, shapes, and more complex features). The behavior of CNNs closely resembles the way in which humans interpret images. As a result, CNNs have the potential to detect patterns on the IV surface that are similar to those recognized by professional traders during visual inspection. Furthermore, the locality of CNNs makes them perfect instruments for learning non-linear filters, extracting the volatility and risk premia information from the noisy IV surface.[2]

We train several CNN architectures of increasing depth (complexity) on the standard OptionMetrics dataset of IV surfaces for several thousand stocks to predict stock returns over a one-month horizon. Since CNNs (like any neural networks) are trained by gradient descent, they are sensitive to (random) weight initialization: Depending on where the gradient descent starts, it may converge to a weight vector corresponding to a different local minimum.[3] A classic approach for dealing with this is to create an *ensemble* of neural networks corresponding to different weight initializations and then combine them. See, [LPB17]. We find that the benefits from this ensembling are huge because predictive models generated by different initializations produce portfolios with low pairwise correlations. As a result, for the most complex (4- and 5-layer) deep learning architectures, increasing the ensemble size from one to a hundred leads to an increase in the out-of-sample Sharpe ratio from 0.9 to 2.7 for the full stock universe. By contrast, for the lower complexity (1 hidden layer) CNN, the Sharpe ratio only increases from 0.80 to 1.6. As [LPB17] explain, the sensitivity of predictions generated by NN to initialization captures the degree of uncertainty around these predictions. In low signal-to-noise ratio environments of financial markets, this uncertainty is very high, implying large gains from ensembling.

Since each model in the ensemble is represented by a (completely) different set of parameters, this finding implies a very large *virtue of complexity* (in the language of [KMZ24]

[1] See [GMT23].

[2] Formally, locality means that two close points on the IV surface have similar informational content.

[3] The reason is that the dependence of CNN mean squared error on the CNN coefficients is highly non-convex. As a result, the problem is many local minima, and it is impossible to predict to which local minimum the gradient descent will converge.

and [Did+23]): bigger, more complex, non-linear models generate significant gains out-of-sample.[4] However, the nature of the complexity of the ensemble is different: While [KMZ24] and [Did+23] establish the virtue of complexity for one very big model with all parameters jointly trained, in this paper, we document *the virtue of ensemble complexity.* Namely, we train a large number of randomly initialized CNNs with identical architecture and then average their predictions, and show that the out-of-sample Sharpe ratios are approximately monotone, increasing in the ensemble size. A similar pattern is observed if we measure the out-of-sample performance of the model in terms of its alpha with respect to a large set of standard factors, including many standard stock characteristics as well as a large set of option-based characteristics from [Neu+22].

The complexity of our CNN-based model ensemble seems to contradict the conventional principle of parsimony in economics, suggesting that predictive information (originating, e.g., from risk premia or behavioral anomalies) can be encoded into a small number of factors and stock characteristics. When applied to the IV surface, the principle of parsimony suggests that only a few key surface features might contain useful information about future stock returns. Most of these features used in the existing literature are constructed as simple, linear combinations of implied volatilities, for example, the IV level, the slope of the term structure, and the smile. To test whether such a low-dimensional structure is indeed present in our highly non-linear predictive model, we use the methodology of [CDW14] to identify key linear combinations of IVs with the most explanatory power. These combinations are natural analogs of principal components for our highly non-linear models, and we refer to them as *principal linear features.* Contrary to conventional wisdom, we find no evidence for a low-dimensional structure. While broadly consistent with the complexity principle of [KMZ24] and [Did+23], the nature of this phenomenon is different. Indeed, the results in [KMZ24] and [Did+23] mostly concern complexity due to non-linearities, whereas our last finding is about *feature complexity,* expressing the fact that a very large number of linear features is necessary to extract the predictive information contained in the IV surface.

As [KMZ24; Did+23] explains, this *virtue of complexity* means that, even for an already complex model, we can often find a new, nonlinear transformation of the IV surface that boosts the out-of-sample performance. This complexity is not a puzzle to be solved or evidence of data mining. Instead, it is the theoretically expected outcome of learning a non-linear, high-dimensional relationship with limited data. The only constraint we impose on the model is the principle of locality, formalized by choice of a convolutional NN instead of a generic NN architecture utilized, say, in [GKX20b] and [CRW21].

Recently, several papers (see, e.g., [DeM+20], [DNMV23], and [MPP22]) have argued that the performance of modern multi-factor models needs to be evaluated by accounting for transaction costs and short-sale constraints. We perform a detailed analysis of trading costs for our strategies and find that, perhaps surprisingly, the market-neutral portfolio constructed by sorting stocks based on the CNN-predicted return makes most of the money with the long leg, implying that our findings are not sensitive to short-selling costs and constraints. Transaction costs incurred by our strategy are significant: As with many machine learning models, it exhibits a high turnover (about 80% per month). Following [JKPrt] and [Did+23], we study the model performance separately for several groups of stocks, created based on their market capitalization (mega, large, small, micro, and the non-micro group constructed as the set of all stocks excluding the micro-cap group). While the micro-group dominates the performance, the non-micro group also delivers a very strong Sharpe Ratio and a significant alpha. Both micro- and non-micro groups retain their alpha significance after accounting for realistic costs. It is also important to note that, in any case, the optionable stocks (i.e., stocks with options traded on them) are typically large and have a significant trading volume, implying that even the predictability we identify for micro stocks can be exploited with some meaningful arbitrage capital.[5]

The rest of this paper is organized as follows. Section 2.2 reviews the related literature.

[4] Table 2.9 shows that these models are indeed extremely complex, with the number of parameters significantly exceeding the number of observations in our panel dataset.

[5] Recent results of [Jen+22] suggest that, by properly smoothing positions, it is possible to reduce turnover and preserve the bulk of the performance of machine learning models such as ours. We leave this important direction for future research.

Section 2.3 describes data and methodology. Section 2.4 provides the necessary background on CNNs. Our main empirical results are reported in Section 2.5. Section 2.6 investigates feature importance, introduces *principal linear features*, and documents *the virtue of feature complexity*. Section 2.7 concludes.

2.2. Related Literature

Understanding why average returns differ across assets is a central question in finance. Over the last few decades, the search for stock characteristics that predict returns has led to the emergence of the (constantly growing) "factor zoo": A huge number of characteristics that contain information about the cross-section. See, for example, [Coc11], [HLZ16], [MP16], [HXZ20], [FGX20], [JKPrt], and [GKX22] for a recent overview.

Many papers in this literature focus on predicting asset returns using complex, non-linear models; see [MZ16], [CCJY19], [Han+19], [CPZ19], [BPZ20], [LZZ20], [GKX20b], [KNS20], [FNW20], [ACM21], [GOPZ21], [LWZ22], [KMZ24], and [Did+23]. Given the ever-growing complexity of these models (both in terms of the number of characteristics and the degree of non-linearity of the predictive relationships), several papers develop techniques to "shrink" the cross-section and find a sparse representation of the expected returns, either through a form of dimensionality reduction (e.g., by exploiting principal components, as in [KPS20], [KNS18], [KNS20], [LP20], and [GX21]), or by imposing sparsity directly in the space of characteristics (see, e.g., [GKX20b], [FNW20], and [BHJ23]). While the evidence is mixed, recent findings of [Did+23] suggest that complexity is there to stay, and there might be no feasible way to find a sparse representation of expected returns. Namely, as [Did+23] show, the factor zoo is simply a statistical phenomenon originating from the small data problem: We do not have enough data to find the right low-dimensional representation of expected returns, even if it exists; hence, our best bet is to build the most complex model without imposing a sparse prior.[6]

Most of the above-mentioned papers focus on stock characteristics that are either purely price-based (such as momentum; see, [CCM97]) or depend on company fundamentals such as the book-to-market ratio. By contrast, our paper focuses exclusively on the predictive information contained in the implied volatility surface, motivated by the idea that derivative prices provide an interesting lens to uncover rich information about the underlying assets and associated risks. The early contributions by [BL78b], [BM78] show how Arrow-Debreu state prices can be recovered from the option prices. State prices contain information about risk premia and (subjective) physical probabilities as market participants anticipate. Under technical conditions, some information about these physical probabilities can be recovered; see, e.g., [Ros15a], [BHS16b], and [JLP19]. The idea of extracting useful forward-looking information from both individual equity and index options has been exploited in many papers. For example, [BH09] show that the difference between implied (i.e., risk-neutral) and physical volatility, as well as the difference between the implied volatilities of near-the-money call and put options are both strong predictors of stock returns; [CW10] use deviations from put-call parity to predict stock returns; [JS12] show that the ratio of the volume between options and stocks predicts stock returns at the one-week horizon; [Cho+20] reach a similar conclusion when examining the (signed) order flow in equity index options; [An+14] find that stocks with large increases in call (put) implied volatilities over the previous month tend to have high (low) future returns; [AFT15], [BT14; BTX15a], [LT19], [BDG20], and [HLT20a] show how the jump (tail) risk extracted from options prices predicts future stock returns; [BBG18] show that the volatility of implied volatility has predictive power for future stock returns, and [DBG20] show how to estimate cross-sectional uncertainty from option prices.

Many option-implied predictors build on a potential risk-return relationship between the risk-neutral variance, variance risk-premium (VRP), and stock returns; see, e.g., [BTZ09], [FJPO18], [MW19], [KS19], [Feu+19], [Tan19], [KT20], [Ped20], and [DJW22].

[6]More generally, several recent papers have questioned the principle of sparsity in economic modeling. See, e.g., [GLP21].

Several other papers propose characteristics of the IV surface that are related to the shape of the implied volatility smirk or corresponding risk-neutral skewness portfolios; see, e.g., [XZZ10], [Yan11], [CDG13], [SKP17], [JMW18] and [BM19], and [Sch+20]. While many of these papers find that IV-surface-based skewness measures contain predictive information, the nature of this information seems extremely sensitive to precise details of factor construction. For example, [CDG13] document a negative relation between ex-ante risk-neutral skewness and asset returns, while [SKP17] finds a positive relation.[7] These findings suggest that the nature of predictive information might be extremely complex, making it difficult to pin down the precise underlying economic mechanism.

The paragraphs above suggest that, even in the smaller world of option-implied characteristics, we are clearly facing the problem of an ever-growing factor zoo, with new predictive relationships constantly discovered by academic researchers. Two recent papers attempt to bring order into this option-implied factor zoo by combining the informational contents of the multiple factors identified in the existing literature. [BCYM22] use five of those factors (based on the findings of [BH09], [CW10], [XZZ10], and [An+14]). They provide evidence that a linear tangency portfolio of these factors is not spanned by standard stock characteristics-based managed portfolios. However, in a more recent paper, [Neu+22] provide evidence that the factors in [BCYM22] are spanned by fundamental stock characteristics when more of those characteristics are included in the model. [Neu+22] then argue that machine learning methods can deal with the IV-surface-based factor zoo and extract useful information unspanned by standard factors. They consider 17 options-based characteristics, including five of [BCYM22]. Then, they apply the adaptive group LASSO methodology of [FNW20] to build non-linear predictive models based on these 17 characteristics. They provide strong evidence that only 4 out of 17 option characteristics contain information about future stock returns not spanned by a large set of more than 60 stock characteristics. These four economically important characteristics are all related to the shape of the IV smirk.

While [Neu+22] do develop a complex, non-linear, machine learning model, they utilize relatively small (only 17) ready, pre-built characteristics of the IV surface, motivated by sparsity considerations and the idea that a few linear features of the IV surface summarize its predictive information. In this paper, we follow a different approach. First, we completely abstract from the existing set of characteristics such as smirk, skew, slope, etc.; instead, we take an agnostic approach and let the machine learning model decide which features of the IV surface are useful for predicting stock returns. Second, we do not try to impose any form of sparsity on the model. Instead, we build a model with an exorbitant number of features and parameters following the principle of the virtue of complexity ([KMZ24; Did+23]) combined with the *virtue of ensemble complexity* introduced in this paper. As explained above, we find evidence for both virtues: There is a large amount of predictive information in the IV surface, and we need a large ensemble of highly complex models to leverage this information efficiently. Furthermore, our model exhibits a very high *feature complexity:* A small number of linear IV features cannot span our model's predictive content.

We complete this literature review by noting that implied volatility is closely related to the physical volatility (in fact, in the idealized Black-Scholes continuous time setting, the two should be identical) and, hence, any signal about the level of the IV surface is closely related to the physical idiosyncratic stock volatility. The latter contains predictive information about stock returns and belongs to the large family of "low-risk anomalies." See, [ACX06], [HS00], [BMV10], [FP14], [Ama+15] and [Sch+20]. In particular, [Sch+20] argue that these low-risk anomalies reflect compensation for co-skewness risk. It is possible that the more complex predictive signals extracted from the IV surface by our model might also reflect a form of risk premium related to some jump, tail, or high-order moment risk. Understanding the connection of our non-linear signals with risk premia is an important direction for future research.

[7]See, also, [Jia+20] for similar results based on physical measures of skewness; and [KNS13], [ST19], and [OST20] for a studies of aggregate measures of skewness.

2.3. Data and methodology

2.3.1. Data

Options data are from OptionMetrics IvyDB. The historical IvyDB Volatility_Surface dataset contains interpolated option implied volatility surfaces for a large set of firms for each trading day from January 1996 to December 2021. For interpolation, a proprietary kernel smoothing algorithm is applied by OptionMetrics across both moneynesses (defined in terms of option δ) and expiry (defined in terms of days-to-maturity τ) grids, allowing us to abstract from option cycles and varying strikes. Thus, for each trading day, δ-moneyness grid goes from -1 to +1 in equidistant 0.05 steps (puts have negative δ, calls have positive δ), and expiry grid ranges from 10 to 730 days-to-maturity.

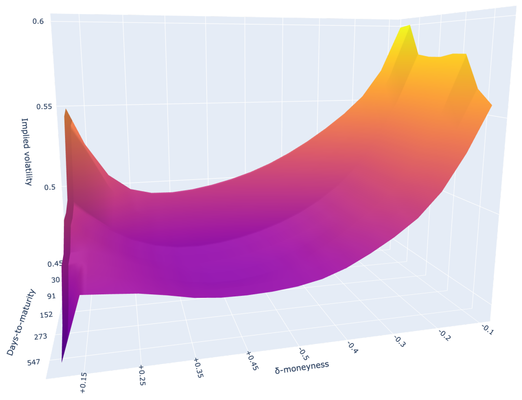


Figure 2.1: Transformed implied volatility surface of TSLA stock, 01/07/2016.

We hypothesize that in-the-money calls contain the least forward-looking information and only keep $\delta \in [-0.5, 0.5]$; then, we drop all 10-days-to-maturity values due to lack of data before 2010[8]; lastly, we re-stack volatility surfaces at-the-money (so that δ goes from 0.1 to 0.5, then goes from -0.5 to -0.1) – this way, the combination of put and call IV components results in a more seamless transition. Thus, we end up with implied volatility surface images of size 10×18. Figure 2.1 shows a typical example of an implied volatility surface from Option Metrics IvyDB that we work with. We denote $IV_{i,t} = (IV_{i,t}(\delta, \tau))_{\delta \in \Delta, \tau \in \S}, \in \mathbb{R}^{|\Delta| \times |\S|}$ the interpolated implied volatility surface of a stock $i \in N$ on the last day of the month $t \in T$. Here, $\delta \in \Delta$, where Δ is the grid of available δ-levels, and $\tau \in \S$, where $\S$ is the grid of available times to maturity.

We use daily stock data from CRSP to construct monthly total stock returns for all NYSE, AMEX, and NASDAQ firms. CRSP sample is aligned to our OptionsMetrics sample and lasts from January 1996 to December 2021. The number of stocks with available stock and options data is shown in Figure 2.2. The total number of stocks in our sample is close to 25 thousand, with the average number of stocks at any point in time exceeding three thousand.[9] We denote $R_{i,t+1}$ the total return on stock i from the last business day of the month t to the last business day of the month $t + 1$.

Monthly returns are constructed as follows. First, for each stock i and business day d, daily total gross returns are computed as $(1 + r_{i,d})$ or as $(1 + r_{i,d})(1 + r_{i,d}^{\text{delist}})$ if the stock

[8]This is likely because so-called ”weeklies” (options with weekly expiration cycles) were only introduced in 2005 and gained popularity after the Global Financial Crisis.

[9]See the section 2.8 of the Appendix for a detailed description of CRSP and OM datasets, the dataset linking procedure, and some additional statistics.

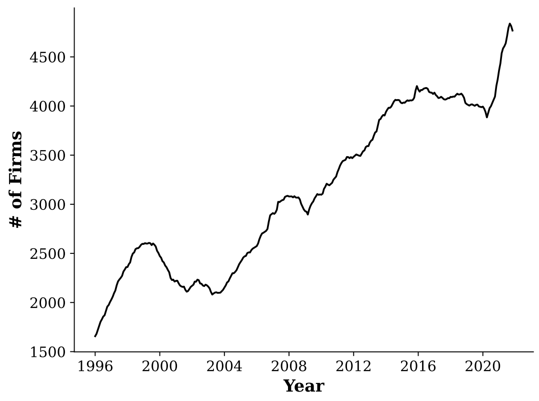


Figure 2.2: Evolution of our post-processed dataset size regarding the number of unique firms.

i delists on day d. Here, $r_{i,d}$ is the total daily return from the CRSP Daily Stock file, and $r_{i,d=delist}$ is the delisting return from the Delisting Information CRSP file. Missing daily returns[10] are replaced with zeros. Monthly total gross returns for a month t, denoted by $R_{i,t}$, are cumulative products of daily gross returns,

$$R_{i,t} \;=\; \prod_{d\in t}(1+r_{i,d})(1+r_{i,d}^{\text{delist}}\mathbf{1}_{d=delist}) \;-\; 1\,. \tag{2.1}$$

Everywhere in this paper, *we focus on predicting monthly returns* $R_{i,t+1}$ using only implied volatility surfaces *on the last business day of the preceding month* t, which we denote by $IV_{i,t}$.[11]

2.3.2. Ensembles of Randomly Initialized Neural Nets

Given a family of non-linear functions, $f(x;w)$, indexed by a (high-dimensional) vector of parameters w (e.g., neural weights for the case of neural networks), we can optimize w with the objective of predicting returns $R_{i,t+1}$ by plugging the month-end $IV_{i,t}$ into f and minimizing an error on the training data: Given a look-back horizon T, we can try selecting a parameter vector w that minimizes the in-sample prediction error based on the last T periods of the data:

$$w_{*,t} \;=\; \arg\min_w \ell(w),\;\; \ell(w) \;=\; \sum_{\theta=t-T}^{t}\sum_{i=1}^{N_\theta}(R_{i,\theta+1}-f(IV_{i,\theta};w))^2\,, \tag{2.2}$$

where N_θ is the number of stocks in our data set at time θ. The standard way of finding w_* is by using gradient descent: Given a learning rate η, one can randomly initialize w_0 and then implement the algorithm of gradient descent by iteratively computing

$$\hat{w}_j(w_0) \;=\; \hat{w}_{j-1} \;-\; \eta\,\nabla_w\ell(\hat{w}_{j-1}) \tag{2.3}$$

[10] 1.34% of the total number of return observations.

[11] One may wonder whether information about the daily dynamics of implied volatility surfaces in the preceding month t contains predictive information about $R_{i,t+1}$ and whether we are neglecting this information by keeping only month-end $IV_{i,t}$. We have tried to incorporate additional information about daily IV dynamics in the preceding month in our predictive models, and these experiments suggested that lagged IV information during the month does not add predictive power.

for $j \geq 1$, where $\nabla_w \ell$ is the gradient of the loss $\ell(w)$ with respect to w.

By definition, the path of $\hat{w}_j(w_0)$ during the gradient descent (as j increases from one to ∞) depends on the initialization w_0. However, standard results imply that, as the number j of gradient steps increases while the learning rate η converges to zero, $\hat{w}_j(w_0)$ converges to a *local extremum* $\hat{w}_\infty(w_0) = \lim_{j\to\infty,\eta\to 0} \hat{w}_j(w_0)$, satisfying $\nabla_w \ell(\hat{w}_\infty(w_0)) = 0$. In fact, as recent research shows, these local extrema are typically global minima for neural nets. See, e.g., [SC16]. In simple, linear regression problems, the loss function $\ell(w)$ is convex in w and, hence, has a unique extremum which is also a global minimum; hence, for convex problems, $\hat{w}_\infty$ is independent of w_0. By contrast, for realistic, non-linear machine-learning models (such that CNNs in our paper), $\ell(w)$ is not convex and has a tremendous number of (global or local) minima. See, e.g., [LPB17; Li+18; Yao+20; FHL19]. As a result, $\hat{w}_\infty(w_0)$ depends on the initial weights, w_0, in a highly complex fashion. In particular, for K different initializations $w_0(k), k = 1, \cdots, K$, we will typically end up with K different weight vectors $\hat{w}_\infty(w_0(k))$. This leads to a *randomly initialized ensemble* of K models, $\{f(x; \hat{w}_\infty(w_0(k)))\}$, indexed by $k = 1, \cdots, K$. We then build the ensemble prediction by taking a simple average across k :

$$\widehat{R}_{i,t}^{\text{ens}} = \frac{1}{K}\sum_{k=1}^{K} f(IV_{i,t}; \hat{w}_\infty(w_0(k)))\,. \tag{2.4}$$

Our findings present results for CNN ensembles of different sizes, with K ranging from 1 to 100. We observe a significant enhancement in portfolio performance as the ensemble size increases up to around 50 models, after which it saturates.

Given these predictions at each month-end, all stocks are sorted into decile, equal-weighted portfolios to construct a long-short spread portfolio, "H-L", that is long the upper decile and short the lower decile. Denote $N_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}$ the number of stocks in the upper decile at time t, and $N_{\widehat{R}_{i,t}^{\text{ens}} < q_t}$ the number of stocks in the lower decile at time t, the returns on "H-L" are given by:

$$R_{t+1}^{H-L} \;=\; \sum_{i=1}^{N_t} R_{i,t+1}\, w_{i,t}\;,\; w_{i,t} \;=\; \frac{\mathbf{1}_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}}{N_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}} - \frac{\mathbf{1}_{\widehat{R}_{i,t}^{\text{ens}} < q_t}}{N_{\widehat{R}_{i,t}^{\text{ens}} < q_t}}, \tag{2.5}$$

where Q_t and q_t are the 90% and 10% percentiles of the distribution of $\widehat{R}_{i,t}$ at time t.

The algorithm described above depends on two key objects: The look-back window T in (2.2) used for training the model and the family of non-linear functions $f(x; w)$. For the lookback window, we follow the approach of [GKX20b] and use an expanding window (allowing us to use all available data at time t). [12]

We consider three different families of functions $f_\ell(x; w),\ \ell \in \{CNN1, CNN4, CNN5\}$ in our analysis. Each family is a convolutional neural network (CNN, see Section 2.4 for details), with $CNNi$ equipped with i hidden convolutional layers, $i = 1, 4, 5$. The number of layers of the network defines its complexity, expressive power, and ability to approximate non-linear functions. The complexity of a CNN model (defined as the number of parameters divided by the sample size, see [KMZ24]) increases exponentially with depth. See Table 2.9 for details. Intuitively, we expect shallow CNNs to learn fewer and simpler features, whereas more complex CNNs will detect a greater variety of features. While such complex models are tremendously over-parametrized and severely overfit the data in-sample, their performance out-of-sample might be better due to the virtue of complexity. See [KMZ24], [Did+23], and [Bel21].

2.4. Convolutional Neural Networks

This section provides a concise overview of the rationale behind Convolutional Neural Networks (CNNs), the architecture of the proposed models, and the training methodology em-

[12] We also report that using a rolling window of five years, which allows us to account for potential non-stationarity, leads to a performance drop.

ployed.[13] Additionally, we briefly discuss the discrete convolution operation and model optimization via stochastic gradient descent. Please refer to Appendix 2.9 for rigorous mathematical definitions.

2.4.1. CNN Architecture

Convolutional Neural Networks (CNNs) continue to be the state-of-the-art approach for image classification (see [Sha+20]). We outline some of the key advantages of utilizing CNNs.

1. Spatial Hierarchy and Local Connectivity: CNNs are designed to capture spatial patterns and relationships between pixels in images, thanks to their use of convolutional layers. These layers apply filters to local regions of the input image, allowing the network to learn and identify patterns at various levels of abstraction (e.g., edges, shapes, and more complex features). In contrast, Deep Neural Networks (DNNs) treat all input features independently, losing the spatial information and relationships between neighboring pixels. This makes it difficult for DNNs to learn and recognize complex image patterns effectively.
2. Translational and Rotational Invariance: CNNs inherently possess translation and rotational invariance, meaning that they can recognize patterns and features regardless of their position in the image. Conversely, as mentioned earlier, DNNs treat each input independently, resulting in the loss of information.

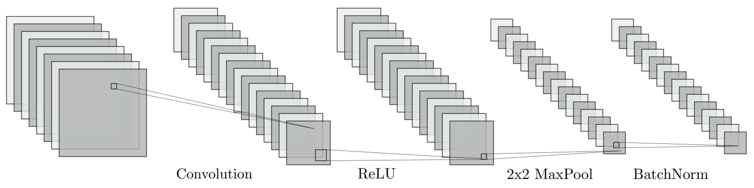


Figure 2.3: The figure above shows a building block of the CNN model consisting of a convolutional layer with a 3×3 filter, a ReLU layer, 2×2 max-pooling, and batch normalization layers. Note the max-pooling layer shrinks the height and width of the input by half and keeps the same depth.

We now describe a Convolutional Neural Network (CNN) as a sequence of operations to transform raw images, implied volatility surfaces in this case, into a prediction. See Appendix 2.9 for details.

A CNN core building block consists of three operations: convolution, activation, and pooling. In addition, we also use a batch normalization layer at the end of each block. A

[13]See, also, [JKDX22].

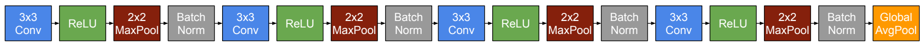


Figure 2.4: The figure above describes our CNN4 architecture.

batch normalization layer normalizes the building core's output reducing the so-called *internal covariance shift* (acting as a regularizer) from one building core to the next [IS15b]. The core CNN building block is shown in Fig. 2.3. A convolution layer applies filters to the input data, capturing local spatial patterns and generating feature maps as output. An activation function is a nonlinear function applied elementwise to the output of a layer. Max-pooling is a downsampling operation in a CNN that reduces the spatial dimensions of a feature map by selecting the maximum value within a defined neighborhood, helping to retain the most prominent features while reducing computational complexity and achieving translation invariance. Global Average Pooling is a pooling operation that computes the average value of each feature map across its entire spatial extent, effectively "vectorizing" the feature map. The final CNN layer is a single fully connected node that targets the next month's stock gross return. The CNNi architectures described above are constructed by stacking i such building blocks. For example, Figure 2.4 shows the CNN4 architecture.[14] See Appendix 2.9 for details and precise mathematical definitions of convolutions, MaxPool, and Global Average Pooling layers.

2.4.2. Training the CNN

Our initial training uses the first seven years of observations (1996-2002) for the expanding window case. We call this the *warm-up period*. In this period, the CNN is trained for ten epochs. Then, we apply the trained model to the features of the subsequent month (unseen to the model) to make predictions. Afterward, we include this new month in the training set to retrain the model using the same procedure. We call this the *transfer-learning period*. In this period, the CNN is trained for five epochs only, as the CNN has mostly learned the market dynamics during the *warm-up period*, and only requires *fine-tuning* to adjust to an additional one month of observations.

We employ the same regularization methods as in [GKX20a] to mitigate overfitting and facilitate efficient computation. We utilize the Xavier initializer for weight assignment in each layer, as proposed by [GB10], which accelerates convergence by producing initial weight values that align the prediction variance with the label scale. For loss function optimization, we combine stochastic gradient descent with the Adam algorithm [KB14], setting the initial learning rate at 1×10^{-3} and using a batch size of 512.

2.5. CNN Portfolio Performance

In this section, we investigate the out-of-sample performance of CNN-based return forecasts. Given a CNN architecture, we randomly initialize its weights w_0 using 100 different random

[14]In particular, we use an increasing number of convolution filters in each block, i.e., 16, 32, 64, and 128.

seeds and train each of these NNs using gradient descent, as described in Section 2.4, thus creating an ensemble of CNNs. As described in Section 2.3.2, we construct a long-short spread portfolio, "H-L", that is long the upper decile and short the lower decile of predicted stock returns, with a monthly holding period, see (2.5). Then, we compute the predictions (2.4) gradually increasing the ensemble size K from 1 to 100 and report the corresponding out-of-sample Sharpe ratios in Figure 2.13, and summarize their distribution in Table 2.2. Following [JKPrt] and [Did+23], we perform the analysis separately for different size groups of stocks: mega (largest 20% of stocks based on NYSE monthly breakpoints), large (between 80% and 50% percentile of NYSE breakpoints), small (between 50% and 20% percentile of NYSE breakpoints), and micro (smallest 20% of stocks). Additionally, we report the results for the non-micro group that comprises all stocks with a market cap larger than the 20% NYSE percentile.

Figure 2.13 clearly illustrates the *virtue of ensemble complexity*: Sharpe ratios are monotone increasing in the ensemble size and, for more complex models (CNN4 and CNN5), saturate only around $K = 40$. While a single run of CNN1, CNN4, and CNN5 achieves an annualized Sharpe Ratio of 0.80, 1.48, and 0.88, respectively, the portfolios built using ensembles of $K = 100$ randomly initialized CNN1, CNN4, and CNN5 (see (2.4) for the definition), achieve out-of-sample Sharpe ratios of 1.64, 2.72, and 2.50, respectively. Figure 2.14 shows the cumulative performance of our strategy over time and compares it with the three strongest option characteristics-based factors from [Neu+22]. As one can see, the complex model clearly outperforms these factors. Sharpe ratios for 100-ensemble CNN models per market cap segment are reported in Table 2.17.

We now investigate to what extent standard and option-based characteristics span the performance of CNN-based factors. As Table 2.3 shows, ensemble CNN portfolios deliver a highly significant alpha against a variety of option-based factor portfolios from [Neu+22] as well as the standard Fama-French factors. Complexity also has a very strong impact on the regression R^2 : While for the lower-complexity CNN1, the R^2 is about 66%, it drops all the way to 36% for CNN4 and CNN5 models, suggesting that more complex models are able to pick up highly non-linear predictive patterns in the IV surface that are not accessible to simpler models.

Perhaps surprisingly, our most complex CNN4 and CNN5 models have only minimal and marginally significant exposure to option characteristics-based factors. It has some exposure to long-term reversal while being significantly *negatively* exposed to short-term reversal, suggesting that the predictability we identify is not driven by standard short-term statistical arbitrage. Instead, exposure to long-term reversal suggests some link between our signals and fundamental stock valuations. The low complexity CNN1 model does pick up significant exposure to the Skew factor, but as complexity increases, this exposure vanishes, confirming our intuition that complex models identify highly non-linear patterns that are not spanned by standard, linear characteristics of the IV surface.

The H-L strategy (2.5) is non-linear and exploits the power of the relationship between CNN-based predictions and future returns in the tails of the prediction distribution. We test the robustness of the above results by running simple, cross-sectional (Fama-MacBeth) regressions of returns on CNN-based predicted returns, using multiple characteristics as controls. Table 2.10 strongly supports our findings: Stock returns are indeed highly significantly related to CNN-based predictions, even after controlling for a large number of standard stock characteristics, including reversal, momentum, idiosyncratic volatility, as well as standard option-based characteristics.

We find that smaller-capitalization stocks largely drive the model performance, see Tables 2.4-2.8, nonetheless, CNN models still generate significant alpha for the not-micro group. We also find that it exhibits the same *virtue of the ensemble complexity*[15]. Notably, the results from the Fama-Macbeth regressions per-segment reveal that the statistical power of CNN forecasts is mainly concentrated in the small-cap stock group. This is highlighted in Tables 2.11 to 2.15. Value-weighted[16] portfolios built using CNN4 forecasts generate alphas that are statistically significant at a 10% level, see Table 2.38.

[15]These results are not included and are available from the authors upon request.

[16]Weights are set in proportion to stock market capitalization, with market caps winsorized at 80% NYSE percentile as described in [JKPrt].

2.5.1. Simpler Models

The above findings clearly indicate that CNN-based models are able to extract important predictive information from the IV surface. But do we really need CNNs? Is it possible to extract the same information using simpler models?

We start by investigating whether it is possible to use all IVs in a linear fashion to predict returns. To this end, we generate predictions based on the simple ridge *panel* regression model with an expanding window by building the prediction $\hat{R}_t^{ridge}$:

$$\hat{R}_t^{ridge} = IV_t(IV_{0:t-1}^\top IV_{0:t-1} + zI)^{-1} IV_{0:t-1}^\top R_{0:t-1}, \tag{2.6}$$

for each period in time $t = 0, \dots, T$ with a ridge penalty z. Here, $IV_t \in \mathbb{R}^{N_t \times |\Delta| \times |\S|}$ represents the panel of implied volatility surfaces for N_t stocks at time t; $IV_{0:t-1} \in \mathbb{R}^{N_{0:t-1} \times |\Delta| \times |\S|}$ and $R_{0:t-1} \in \mathbb{R}^{N_{0:t-1}}$ denote the expanding window of implied volatility and true gross returns, respectively, covering the time period $t = 0, \dots, t-1$. This is a "kitchen sink" approach, whereby we are completely agnostic about the nature of predictors that the model generates. Table 2.18 reports the corresponding H-L strategy. As one can see, even such a simple strategy generates significant alpha relative to existing option-based factors. However, a per-group analysis, see Tables 2.19-2.23, indicates that, in fact, only the micro-group delivers significant alpha, in stark contrast to the case of CNN, see Tables 2.4-2.8.

We now turn to a simpler non-linear model, referred to as NN1 in the sequel: A single-hidden layer perceptron (a fully connected neural network). We use a relatively wide hidden layer of 128 neurons, implying a modest complexity (see Table 2.9). As for the CNNs, we build an ensemble of 100 randomly initialized NN1 models and find evidence for the same virtue of ensemble complexity as for the CNN models. Thus, we build our final NN1 model by averaging across 100 seeds and then build the corresponding H-L portfolios. Table 2.24 studies whether the NN1-based factor absorbs any significant fraction of the alpha generated by the CNN models. Comparing with Table 2.3, we see that both alphas significance and magnitudes are preserved: While NN1 does absorb about 50% of the CNN1 alpha, it absorbs only 25% of the alpha of more complex CNN4 and CNN5 models. At the same time, Table 2.25 shows that the NN1 model does not exhibit any alpha relative to the simple CNN1 model. These findings have two important implications for us. First, they imply a very strong form of the virtue of complexity, manifesting itself in the ability of CNN4 and CNN5 models to identify non-linear features that the simpler models cannot capture. Second, ignoring the geometry of the IV surface and the key *locality principle* used by CNNs for feature construction leads to highly inferior performance of the simple NN1 model. Just naively building bigger models suggested by the complexity principle of [KMZ24] and [Did+23] is not enough: One needs to exploit the economic structure of the data and build neural architectures that optimally exploit this structure.

2.5.2. Long only

Short selling might be extremely costly for some stocks, especially less liquid ones. Many anomalies have been criticized for being difficult to implement for this reason. See [MPP22]. In this section, we investigate the long leg of our strategy, defined as

$$R_{t+1}^H = \sum_{i=1}^{N_t} R_{i,t+1}\, w_{i,t} \ , \ w_{i,t} = \frac{\mathbf{1}_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}}{N_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}} \, , \tag{2.7}$$

where Q_t is the 90% percentile of the predicted returns, and $N_{\widehat{R}_{i,t}^{\text{ens}} > Q_t}$ is the number of stocks in the upper decile at time t. The performance of this long-only strategy is reported in Figure 2.15 (cumulative returns plot) and Table 2.16 (regressions statistics) for all stocks. To emphasize the power of our results, we set an extremely high bar for our long-only strategy and compute the alphas with respect to the benchmark factors (including all option-based factors) that are long-short. As one can see, the performance is robust, and alphas are highly significant. Compared to Table 2.3, complex models lose about half of their alpha and gain (not surprisingly) a huge exposure to the market portfolio.

2.5.3. Transaction and Short-Sale Costs

Many stock market anomalies and factors have been criticized for generating very high turnover and hence, their performance being extremely sensitive to transaction costs. Many characteristics-based portfolios generate negative performance after costs. See, for example, [DNMV23]. As [Jen+22] show, efficiently exploiting machine-learning-based strategies requires optimizing portfolio positions to reduce turnover optimally. While it is possible to use the methodology of [Jen+22] and incorporate costs and turnover directly into our optimization algorithm, in this paper, we purposely follow a simpler approach and evaluate the performance of (2.5) and its long-only version (2.7) after costs, without applying position smoothing techniques.

We apply linear transaction fees [17] and short-sale costs to all our portfolios (CNN1, CNN4, CNN5, NN1, and Ridge) and, for a fair comparison, to all option-based factors from [Neu+22] against which we benchmark our models. We do not apply fees to the factors that are not option-based (e.g., momentum, reversal, Fama-French factors), which makes our analysis even more conservative. Following [CSKS23], we apply the method that accounts for transaction costs at a portfolio level, which is described below. At a given time t, denote $w_t \in R^{N_t}$ the weight vector with coordinates

$$w_{i,t} \;=\; \frac{\mathbf{1}_{\widehat{R}^{\text{ens}}_{i,t} > Q_t}}{N_{\widehat{R}^{\text{ens}}_{i,t} > Q_t}} \;-\; \frac{\mathbf{1}_{\widehat{R}^{\text{ens}}_{i,t} < q_t}}{N_{\widehat{R}^{\text{ens}}_{i,t} < q_t}}\,, \tag{2.8}$$

i.e., our portfolios (CNN1, CNN4, CNN5, NN1, and Ridge) are equal-weighted, and, at a given t, all positive (negative) weights sum up to 1 (-1). We abuse the notation and use $N_t \cup N_{t+1}$ to denote the set of stocks available for trading at times t and $t+1$. Then, the unit cost (per dollar of investment) of rebalancing and short-selling at time $t+1$ is given by

$$\begin{aligned} \bar{f}_{t+1} \;&=\; \sum_{i \in N_t \cup N_{t+1}} \left(f_i \left|w_{i,t+1} - \frac{1+r_{i,t+1}}{1+r_{t+1}} w_{i,t}\right| + \;(-w_{i,t})^+ \theta_i\right) \\ &=\; \theta_i \;+\; \sum_{i \in N_t \cup N_{t+1}} f_i \left|w_{i,t+1} - \frac{1+r_{i,t+1}}{1+r_{t+1}} w_{i,t}\right|, \end{aligned} \tag{2.9}$$

where r_{t+1} stands for the total return on the long-short portfolio with weights defined in (2.8) and $x^+ = \max(x, 0)$. Notice that we used that weights set at time t change from t to $t+1$ relative to the total portfolio value by $\frac{1+r_{i,t+1}}{1+r_{t+1}}$. This reflects the fact that the security value in the portfolio compounds from t to $t+1$, and the portfolio value compounds from t to $t+1$ as a whole. For simplicity, we assume that short-sale costs are paid out at the beginning of each holding period (this is why we use $w_{i,t}$ in the $(-w_{i,t})^+\theta_i$ term). We also used that $\sum_{i \in N_t \cup N_{t+1}} (-w_{i,t})^+ = 1$ by definition (2.8). Now, let I_t denote the total investment value of the portfolio at time t. The investment value evolves from $t+1$ to $t+2$ as follows:

$$I_{t+2} = I_{t+1}(1 - \bar{f}_{t+1}) r_{t+2}, \tag{2.10}$$

where r_{t+2} is a pre-fee gross return on the portfolio of stocks between $t+1$ and $t+2$, and I_{t+1} is pre-$(t+1)$-rebalancing investment value. Regrouping, we get the net-of-fee net return on the portfolio

$$\frac{I_{t+2}}{I_{t+1}} - 1 = (1 - \bar{f}_{t+1}) r_{t+2} - 1. \tag{2.11}$$

This simple identity allows us to deduct linear transaction costs without keeping track of the investment value but by discounting net-of-fee returns only.

We set transaction costs f_i for a stock i in the micro-cap segment to be two times the cost level for other segments so that

$$f_i = (1 + \mathbf{1}_{i \in microcap}) f_{base} \tag{2.12}$$

[17]Restricting the analysis to linear transaction costs abstracts from important price impact considerations, see [Jen+22].

and we investigate two settings: $f_{base} = 0.001$ (ten basis points (bps)) and $f_{base} = 0.002$ (twenty bps). To account for costly short-selling, we assume a fixed monthly cost of

$$\theta_i = f_i. \tag{2.13}$$

For $f_{base} = 0.002$, this corresponds to 40 bps (= 4.8% per annum) short-sale cost for micro-stocks and 20 bps (=2.5% per annum) short-sale cost for non-micro stocks.[18]

Again, to ensure a fair comparison, we apply the same fee structure to all option-based factor portfolios from [Neu+22], and we do not apply fees to other factors, which adds to the challenge for CNN-based portfolios to maintain their superiority.

Tables 2.26 to 2.37 present the performance of long-short portfolios net of linear transaction costs and short-sale fixed monthly costs, at two different cost levels: 10 bps (and 20 bps for the micro-cap segment) and 20 bps (and 40 bps for the micro-cap segment). In both cases, the micro-cap segment shows high and significant net-of-fee excess returns for CNNs, and, notably, CNN4 also exhibits consistent excess performance in the not-micro-cap segment at 5% (10%) significance level for the 10 (20) bps cost level. On the other hand, the performance of NN1 and the ridge model diminishes, indicating that it's not solely the micro-cap segment that drives excess returns for CNNs. Overall, CNN4 withstands tests with conservative linear transaction and short-sale costs, even outside the micro-cap segment.

2.6. Principal Linear Features

Despite the abundance of data and the recent emergence of the "factor zoo" with numerous characteristics identified as return predictors, predictive relationships in economic and financial data are commonly believed to be sparse. While characteristic-based sparsity (the hypothesis that only a few characteristics matter for economic relationships) is likely an illusion [GLP21], many papers argue that there exists some form of *"linear feature sparsity"*, usually formulated in terms of the principal components [KNS20]. Namely, only a few top principle components are believed to be responsible for most of the predictable variation in returns.

Literature on the predictive content of the IV surface largely relies on the idea of linear feature sparsity and studies only a small number of signals represented as linear transformations of the IV surface, for instance, CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, and related option characteristics described above are all examples of linear transformations of the IV surface. It is thus natural to ask a more general question of whether it is possible to use some non-parametric statistical techniques to extract relevant, linear combinations of the IV surface with the most predictive content.[19] Under the linear feature sparsity hypothesis, we expect that the true dependence

$$f(IV_{i,t}) \;=\; E[R_{i,t+1}|IV_{i,t}] \tag{2.14}$$

of expected returns on implied volatilities is given by a *low-rank function*, as formalized in the following definition.

Definition 2. *Let $\mathcal{S} \subseteq \mathbb{R}^d$. A function $f : \mathcal{S} \to \mathbb{R}$ is said to have rank r on $\mathcal{S}$ if there exists a low-rank matrix $M \in \mathbb{R}^{r\times d}$, $r < d$, and a function $g : M\mathcal{S} \to \mathbb{R}$ such that $f(x) = g(Mx)$ $\forall x \in \mathcal{S}$.*

[18] These estimates are very conservative. According to [Da02], most stocks have short-selling costs below 1% per annum.

[19] One common way of detecting sparsity in a non-linear model is by measuring feature importance. If most features get a low importance score, then one can conclude that the model is effectively sparse. The standard way of measuring feature importance is based on the Shapley value. See, e.g., [LL17]. This approach is designed to measure the importance of a given feature that is expected to contain predictive information. This is the case for standard stock characteristics such as momentum, value, reversal, etc. (see, e.g., [Jen+22]). However, in the case of the IV surface, the features are individual IVs. Hence, it is unlikely that a given IV for a specific (moneyness, maturity) combination contains distinct predictive information, and removing just one point on the IV surface will likely not impact model performance.

In the context of our machine learning task, $\hat{x} = Mx \in \mathbb{R}^r$ is the r-dimensional vector of *linear features*, i.e., those linear transformations of x that matter for the function f. To determine whether a given function is low-rank, we will use the following observation from [CDW14] and [Rad+22]:

Lemma 2.6.1. *Let $\mathcal{S} \subset \mathbb{R}^d$ be an open subset and $f(x) : \mathcal{S} \to \mathbb{R}$ be a real analytic function such that $\nabla f(x)$ is bounded on $\mathcal{S}$. Let also X be a random vector taking values in $\mathcal{S}$, such that its density $p(x) : \mathcal{S} \to \mathbb{R}_+$ is Lebesgue-almost surely positive on $\mathcal{S}$. Define*

$$M_* \;=\; E[\nabla f(X) \nabla f(X)'] \;\in\; \mathbb{R}^{d\times d}\,. \tag{2.15}$$

Suppose that $\operatorname{rank} M_* = r$, *and let $M_* = UDU'$ be its eigenvalue decomposition, where $D = \operatorname{diag}(\lambda_1, \cdots, \lambda_r)$ is the vector of non-zero eigenvalues of M_* and $U \in \mathbb{R}^{d\times r}$ is the matrix of corresponding eigenvectors. Then, for Lebesgue almost every x_0, $g(y) = f((I - UU')x_0 + Uy)$ satisfies $f(x) = g(U'x)\ \forall x \in \mathcal{S}$. Conversely, any function $f : \mathcal{S} \to \mathbb{R}$ of rank r has* $\operatorname{rank} E[\nabla f(X)\nabla f(X)'] \le r$. *Furthermore, if $f(x) = g(Mx)$ where $M \in \mathbb{R}^{r\times d}$ has rank exactly r and $g(y)$ is real analytic on $M\mathcal{S}$ and the functions $\nabla_{y_i} g(y)$, $i = 1, \cdots, r$, are linearly independent, then,* $\operatorname{rank} E[\nabla f(X)\nabla f(X)'] = r$.

This lemma motivates the following definition:

Definition 3. *We refer to the eigenvectors of the gradient outer product matrix* (2.15) *as the principal linear features.*

By definition, principal linear features of f are the directions along which the function varies the most.

We now show how principal linear features can be used to capture a large fraction of variation of the f function. To state the result, we will make use of the Poincare inequality [Leo17]: We say that a probability measure $p(x)dx$ on $\Omega \subset \mathbb{R}^d$ satisfies the Poincare inequality on Ω if there exists a constant C such that

$$\operatorname{Var}[f(x)] \;\le\; C\, E[\|\nabla f(x)\|^2] \tag{2.16}$$

for any continuously differentiable function f. Here, we use this inequality for Gaussian measure [BL76]: If $p(x)$ is a Gaussian measure with the covariance matrix Σ on $\Omega = \mathbb{R}^n$, then $C = \lambda_1(\Sigma)$, where λ_1 is the largest eigenvalue of Σ. It is possible to prove the following result using subtle properties of the Gaussian measures.

Lemma 2.6.2. *Let $M_* = UDU'$ be the eigenvalue decomposition of M_* and U_p the matrix whose columns are the top p eigenvectors of M_*. Suppose that the data $x \sim N(\mu, \Sigma)$ is normally distributed. Let $f_p(y) : \mathbb{R}^p \to \mathbb{R}$ be defined via $f_p(y) \;=\; E[f(x)|U_p'x = y]$. Then,*

$$E[(f(x) - f_p(U_p'x))^2] \;\le\; \lambda_1(\Sigma)\, \Lambda(M_*)\,, \tag{2.17}$$

where

$$\Lambda_{-p}(M_*) \;=\; \sum_{i=p+1}^{d} \lambda_i(M_*)\,. \tag{2.18}$$

Lemma 2.6.2 shows that $f_p(U_p'x)$ represents a good approximation to $f(x)$ as long as the residual variation $\Lambda_{-p}(M_*)$ is sufficiently small. The linear feature sparsity hypothesis is then simply a claim about the quality of the approximation of Lemma 2.6.2. The smaller the residual variation $\Lambda_{-p}(M_*)$, the more accurate the approximation is. To validate our analysis, we verify that $\Lambda_{-p}(M_*)$ is reasonably small for each CNNi model and each p of interest.

Algorithm 1 describes the procedure of building the low-rank counterparts of the ensemble function $f^{ens}(x) = \frac{1}{K}\sum_{k=1}^{K} f(x; \hat{w}_\infty(w_o(k)))$. Using p-predictions obtained with $f_p^{ens}(x)$ counterparts, we build long-short portfolios and compare them by the Sharpe ratio.

Figure 2.5 shows the out-of-sample Sharpe ratios of this strategy as a function of p. We find a striking *virtue of feature complexity:* The out-of-sample performance is monotone increasing for $p > 50$, and the increase is slow. Even for $p = 100$, the Sharpe ratio is still significantly below the full model (corresponding to $p = 180$ linear features). Even for the low complexity CNN1 model, $p = 100$ recovers only about half of the full model Sharpe ratio, and the effect is even stronger for more complex models. It means that we need more than 100 linear features of the IV surface to capture the predictive relationships identified by the CNN models.

Algorithm 1 Linear Feature Sparsity

```
Require: Full sample of $IV_{i,t} \in \mathbb{R}^{18\times 10}$ for each stock $i \in N_t$ available at time $t$, and
    ensembles of CNNs $f_t^{ens}(\cdot)$ trained until $t$ for all $t \in T$.
 1: Compute the total number of $IV_{i,t}$ observations in the sample $N = \sum_t^T N_t$,
 2: Compute average outer gradient product $M = \frac{1}{N}\sum_{i,t} \nabla f_T^{ens}(IV_{i,t}) \nabla f_T^{ens}(IV_{i,t})'$,
 3: Perform eigenvalue decomposition of $M = UDU'$,
 4: for $p \in [1,\dots,180]$ do
 5:     Pick eigenvectors $U_p$ corresponding to $p$ largest eigenvalues, and denote $U_{-p}$ the
        remainder,
 6:     Let $\text{vec}(x)$ be the vertical stack of flattened $IV_{i,t}$ for each $i,t$, i.e. $\text{vec}(x) \in \mathbb{R}^{N\times 180\times 1}$,
 7:     Compute $\mu_y = \frac{1}{N}\sum_{i,t} U_p \text{vec}(x)$,
 8:     Compute $\mu_z = \frac{1}{N}\sum_{i,t} U_{-p} \text{vec}(x)$,
 9:     Compute $\Sigma_x = \text{Cov}(\text{vec}(x), \text{vec}(x)) \in \mathbb{R}^{180\times 180}$,
10:     Compute $\Sigma_z = U_{-p}' \Sigma_x U_{-p}$,
11:     Compute $\Sigma_{zy} = U_{-p}' \Sigma_x U_p$,
12:     Compute $\Sigma_y = U_p' \Sigma_x U_p$,
13:     Compute $\hat{\Sigma}_z = \Sigma_z - \Sigma_{zy} \Sigma_y^{-1} \Sigma_{yz}$,
14:     Sample $\epsilon \sim \mathcal{N}(0,1) \in \mathbb{R}^{n\times(180-p)}$,
15:     for $i,t$ do
16:         Compute $\text{vec}(y_{i,t}) = U_p' \text{vec}(x_{i,t})$, where
17:         Sample $N_\varepsilon$ realisations of $\text{vec}(z|y_{i,t,k}) = \mu_z + \Sigma_{zy}\Sigma_y^{-1}(\text{vec}(y_{i,t}) - \mu_y) + \hat{\Sigma}_z^{1/2}\varepsilon_k$,
            $k = 1, ..., N_\varepsilon$,
18:         Estimate $f_{p,t}^{ens}(y_{i,t}) = \mathbb{E}\big[f_t^{ens}(z + U_p y_{i,t})|y_{i,t}\big] = \frac{1}{N_\varepsilon}\sum_{k=1}^{N_\varepsilon} f_t^{ens}(z|y_{i,t,k} +
            U_p y_{i,t})$.
19:     end for
20: end for
21: We end up with $N$ new $f_{p,t}^{ens}(y_{i,t})$ for $p$-predictions (for each $p$ of interest) for each
    $i,t$ by which new $p$-portfolios are sorted.
```

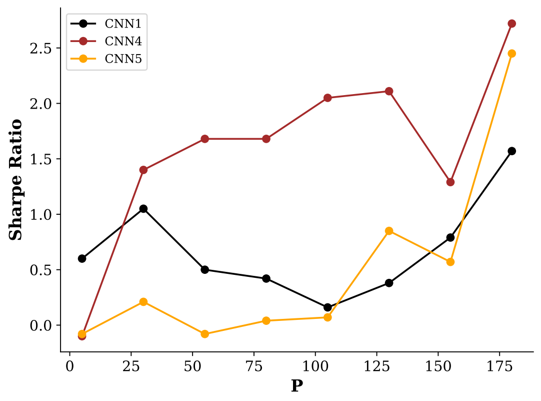


Figure 2.5: The figures above show the Sharpe Ratio of our H-L strategy (2.5) as a function of P, the number of principal features, based on the function $f_P(x)$ constructed using Algorithm 1. The experiment is run separately for each of the CNN1, CNN4, and CNN5 models.

2.7. Conclusion

The remarkable growth of the factor zoo [FGX20], [BHJ23] over the last few years has been accompanied by the development of machine learning methods for asset pricing [GKX20b]. As [KMZ24] and [Did+23] explain, this is no coincidence: Factor zoo is a natural consequence of *complexity*: A highly non-linear predictive relationship between returns and characteristics. The most naive and direct way of exploiting this complexity is to build large, unstructured non-linear models such as simple, fully connected neural networks of [GKX20b] or the random feature models of [KMZ24] and [Did+23]. While this approach works well with structured stock characteristics, it is unsuited for unstructured data, such as the IV surface. To deal with such data, one needs to develop tools and ML algorithms that exploit the data structure optimally. In this paper, we take a step in this direction and propose Convolutional Neural Networks (CNN) architecture designed specifically to extract features of the IV surface that respect locality, as economic theory requires. We show that CNNs can successfully identify highly complex non-linear relationships that cannot be learned with naive, fully-connected networks. Importantly, we find that consistent with the existing evidence for image data [LPB17], the loss landscape of the CNN is extremely non-convex and is characterized by a very large number of local minima. All those minima contain information about returns. Exploiting them requires using an ensemble of CNNs, and we document a very large *virtue of ensemble complexity*. Gaining insights into the incremental information offered by the model as it converges to different local minima for other return prediction problems (including even simpler ones, with the fully connected networks of [GKX20b]) is an important direction for future research.

Conventional wisdom based on the numerous manually constructed option characteristics suggests that a few linear features of the IV surface (e.g., level, slope, skew, and convexity) should fully summarize its predictive content. To test this “linear feature sparsity hypothesis,” we introduce a novel object in financial machine learning, the *gradient outer product*, whose eigenvectors, the *principal linear features*, are natural analogs of principal components for machine learning [Rad+22]. We find no evidence for linear feature sparsity and show that a very large number (more than 100) of linear features are necessary to explain the predictive content of IV, manifesting a very high *feature complexity*. Investigating principal linear features for other ML models and datasets might bring interesting novel insights into the different notions of sparsity in return prediction.

2.8. Appendix - Data Preprocessing

We get option implied volatility surfaces from the Option Metrics IvyDB. In IvyDB, each option chain is already normalized across expiration dates and deltas[20] (11 and 34 different values, respectively). We remove all rows with 10-days-to-expiration implied volatility values due to the high number of missing values at the beginning of the time frame. For each option implied volatility surface, we end up with a 2D matrix of size 10×34.

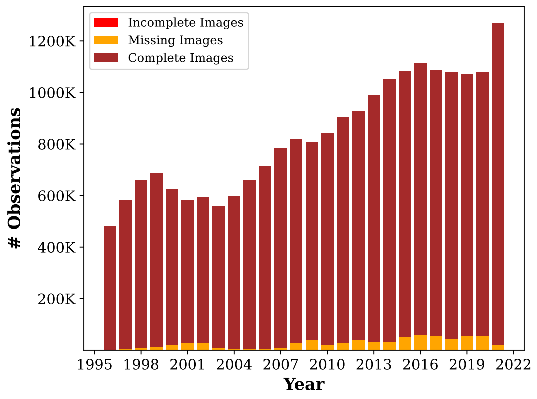


Figure 2.6: The figure above shows our dataset sample size. On the y-axis, we show the number of observations, while on the x-axis, we show the corresponding year. One observation is a single image obtained by stacking implied volatility surfaces for a given pair of stock and date. Incomplete Images are those observations (stock and date pair) where some implied volatility value is missing, given a particular expiration date and delta pair. Missing Images are the sample where the stock and date pair appear in the dataset but have all empty values. Complete Images are the correctly built images where the implied volatility surfaces are complete.

We link the CRSP dataset together with the CRSP delisting dataset, both available in WRDS, to take into account the delisting return as explained in subsection 2.3.1. As the Option Metrics database uses its own security identifier and CRSP uses the PERMNO to identify an asset uniquely, we merge these datasets thanks to a linking dataset provided by WRDS. From the linking dataset, we remove "bad" entries:

1. If an entry has a *score* lower than 1 is removed from our sample. A *score* of 1 means a 100% of mapping confidence.
2. Each entry has a starting date (*sdate*), ending date (*edate*), Option Metrics ID (*secid*), and the PERMNO. We double-sort the dataset using *sdate* and PERMNO. If two consecutive entries have the same PERMNO, but the first (preceding) row has *edate* higher than the second (following) row *sdate*, then the PERMNO is removed from our sample as well.

[20]In other words, each option implied volatility surface in IvyDB is interpolated across moneyness and maturities.

Table 2.1: The table below shows our preprocessed Option Metrics statistics. Namely, Incomplete Images are those observations (stock and date pair) where some implied volatility value is missing, given a particular expiration date and delta pair. Missing Images are the sample where the stock and date pair appear in the dataset but have all empty values. Complete Images are the correctly built images where the implied volatility surfaces are complete.

Year	Missing (%)	Incomplete (%)	Complete (%)	Total
1996	3289 (0.68%)	0 (0.0%)	477674 (99.32%)	480963
1997	5975 (1.03%)	0 (0.0%)	574945 (98.97%)	580920
1998	8789 (1.33%)	2 (0.0%)	651291 (98.67%)	660082
1999	11665 (1.7%)	10 (0.0%)	673334 (98.3%)	685009
2000	19943 (3.18%)	15 (0.0%)	606628 (96.81%)	626586
2001	28265 (4.84%)	6 (0.0%)	555407 (95.16%)	583678
2002	27345 (4.59%)	2 (0.0%)	568126 (95.41%)	595473
2003	10304 (1.84%)	5 (0.0%)	548341 (98.15%)	558650
2004	6137 (1.02%)	6 (0.0%)	593240 (98.98%)	599383
2005	6203 (0.94%)	3 (0.0%)	655095 (99.06%)	661301
2006	5991 (0.84%)	4 (0.0%)	706582 (99.16%)	712577
2007	8783 (1.12%)	5 (0.0%)	776060 (98.88%)	784848
2008	29436 (3.6%)	8 (0.0%)	788981 (96.4%)	818425
2009	40659 (5.04%)	7 (0.0%)	766591 (94.96%)	807257
2010	20822 (2.47%)	7 (0.0%)	821281 (97.53%)	842110
2011	26609 (2.94%)	9 (0.0%)	877955 (97.06%)	904573
2012	38756 (4.19%)	6 (0.0%)	886891 (95.81%)	925653
2013	31810 (3.22%)	3 (0.0%)	956879 (96.78%)	988692
2014	32621 (3.1%)	0 (0.0%)	1020567 (96.9%)	1053188
2015	51663 (4.78%)	0 (0.0%)	1030061 (95.22%)	1081724
2016	60616 (5.45%)	3 (0.0%)	1052121 (94.55%)	1112740
2017	53694 (4.95%)	1 (0.0%)	1031601 (95.05%)	1085296
2018	44705 (4.14%)	3 (0.0%)	1034656 (95.86%)	1079364
2019	54570 (5.1%)	1 (0.0%)	1016092 (94.9%)	1070663
2020	57409 (5.33%)	2 (0.0%)	1020423 (94.67%)	1077834
2021	20653 (1.63%)	0 (0.0%)	1248461 (98.37%)	1269114

2.9. Appendix - More about Convolutional Neural Networks

2.9.1. The Convolution Function

In this section, we present the convolution operation in continuous time and subsequently expound upon its counterpart in the discrete domain. Convolution is a fundamental concept in mathematics and is used in many areas of science and engineering to analyze and manipulate signals and images. The operation involves taking a smaller function, called a kernel or filter, and sliding it over a larger function, called the input, to compute the area of overlap at each point. This process can be thought of as extracting local features from the input function and creating a third function that captures the interactions between the two functions. Formally, if we define the kernel function with $g(t)$ and the signal as $f(t)$, then the convolution function is defined as

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau. \tag{2.19}$$

One can imagine this operation as "sliding" (to the right) the kernel function while the signal function stays still. At some point, the kernel function $g(t)$ will start overlapping with the signal function $f(t)$. The area of overlap will be less or bigger depending on the "shape" of the two functions. Equation 2.19 defines exactly the area where the two functions intersect. Figure 2.19 shows an example of convolution in continuous time.

We now introduce the definition of convolution in case the signal is discrete:

$$(f * g)(t) := \sum_{\tau=-\infty}^{\infty} f(a)g(t-a). \tag{2.20}$$

While in signal processing, the convolution function can be used to filter, smooth, or extract features from signals, in image processing and machine learning, an edge-detection kernel can be convolved with an image to highlight edges and contours. However, in this case, we will have a two-dimensional signal (the image) I and a two-dimensional kernel K

$$(I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n). \tag{2.21}$$

Starting from the top-left corner of the image matrix and moving clockwise, the 2D kernel extracts features from the matrix. Figure 2.8 illustrates an example of 2D convolution applied to a 6×6 matrix using a 3×3 kernel. In this case, the kernel captures the first 3×3 block of the image matrix (highlighted in yellow) and generates a *feature mapping* output for the first cell by computing $23+255+34-66-67-89 = 90$. The kernel then moves by 2 cells to capture the number -154. The distance by which the kernel moves is referred to as the *stride*. Finally, the last column is dropped as the kernel wouldn't fit, and no padding was specified. Padding refers to adding additional pixels or values around the edges of the input image to increase its spatial dimensions and prevent the output feature maps from becoming too small. By adding padding, we can control the spatial resolution of the output feature maps and ensure that the features extracted by the convolutional layers are more representative of the original input. Normally, in the machine learning literature, the following forms of padding are used:

- Zero-padding, where additional zero values are added to the edges of the input data.
- Reflective padding, where the values at the edges of the input are reflected to create the additional padding.

We show in Figure 2.9 a more concrete example using a popular edge detection algorithm in computer vision (see [Dav75]). Like the Sobel kernel, there are many others, like the Gaussian kernel and the Laplacian kernel. probably expand on the conclusion

2.9.2. The Activation Function

Now that we have a basic understanding of image processing and discrete convolution, we can discuss using activation functions in deep learning. These functions are non-linear and are applied element-wise to the tensor, determining how a neuron should "fire." In our work, we exclusively use the Rectified Linear Unit (ReLU) activation function, defined as

$$z = \max\{0, x\} \tag{2.22}$$

The ReLU activation function is popular in the literature due to its simplicity and ability to create sparsity by setting negative values to zero. This can help reduce overfitting and improve generalization. Additionally, the ReLU function is inspired by real biological neuron models, such as the leaky integrate-and-fire model (see, e.g., [AL01], [DA05], and [GB10]).

Figure 2.10 shows an example: By applying the ReLU function, the negative values are set to zero, leaving only the positive values. This helps highlight the important features in the input image and can improve the neural network's performance.

2.9.3. The Max-Pooling Function

Empirical data is often noisy, and images are no exception to this rule. When working with image data, noise can come from various sources, such as imperfect sensors, compression algorithms, or environmental factors. Moreover, the dimension of these images can grow exponentially when applying CNNs. To this end, [LeC+98] has first introduced the max-pooling function.

The max-pooling function selects the maximum value within each pooling region, making the pooling operation less sensitive to small variations in the input and more resistant to noise. By reducing the dimensionality of the data, the function also helps lower the network's computational cost while improving performance.

Figure 2.11 shows an example of this function. At the top, a max-pool with a kernel size of 2×2 is applied to an input matrix of 2×2, giving out a single number; then, at the bottom, the same kernel is applied to a 4×2 matrix, giving in output a 2×1 vector.

2.9.4. The Batch-Normalization Function

When training CNNs, finding a local minimum convergence might be slow due to the distribution change for each layer in the deep architecture. While data normalization is a common practice in machine learning, the hidden layers' input is not normalized due to random parameter initialization and non-linearities. Because these inputs are not normalized by they are rather shifted, [IS15b] call this phenomenon the *internal covariate shift.* The Batch-Normalization function accelerates convergence dramatically and acts as a regularizer, replacing the need for further heuristics techniques, e.g., Dropout. The Batch-Normalization function is applied at each mini-batch, and Algorithm 2 sums up the procedure.

Algorithm 2 Batch Normalization Algorithm

```
Require: Mini-batch of data $\{x_1, x_2, ..., x_m\}$, trainable parameters $\gamma$ and $\beta$, and a
    small constant $\epsilon$
1: $\mu = \frac{1}{m}\sum_{i=1}^{m} S_t$
2: $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(S_t - \mu)^2$
3: $\hat{x}_i = \frac{S_t - \mu}{\sqrt{\sigma^2 + \epsilon}}$
4: $R_{t+1} = \gamma \hat{x}_i + \beta$
```

2.9.5. The Global Average Pooling Function

As we approach the end of the CNN architecture, it is crucial to transform the input to the final layer into a vector right before producing a regression or classification output.

This transformation process, often referred to as "flattening" or "vectorization", involves taking the multi-dimensional output from the previous layer (usually a feature map or activation map) and converting it into a one-dimensional vector. This is an essential step because the final layer, which is typically a fully connected layer (also known as a dense layer), expects its input data to be in the form of a vector. This flattened vector is then fed into the final layer to produce the desired output, such as class probabilities for classification tasks or continuous values for regression tasks.

Flattening the image into a one-dimensional vector can lead to several issues when using images as input to a neural network. One problem is that flattening the image discards the spatial structure of the image, which can be important for capturing meaningful patterns in the data. In addition, flattening the image can result in a very high-dimensional input, which can increase the number of parameters in the model and make it more difficult to train.

In this work, we use the Global Average Pooling (GAP) [LCY13], a technique that addresses these issues by summarizing the feature maps produced by a convolutional layer using an average pooling operation. Unlike flattening, GAP preserves the feature maps' spatial structure by computing each feature map's average value over its entire spatial extent. This reduces the dimensionality of the data, which can improve the model's efficiency while reducing the risk of overfitting.

Table 2.2: Annualized Sharpe Ratio Minimum & Maximum Values of the curves shown in Figure 2.13 for CNN1, CNN4 and CNN5 long-short portfolios from 2003 to 2022.

		SR_{full}	SR_{mega}	SR_{large}	SR_{small}	SR_{micro}	SR_{not_micro}
CNN1							
	Min	0.756	-0.181	-0.038	0.371	1.506	0.175
	Max	1.641	0.201	0.534	0.962	2.163	1.059
CNN4							
	Min	1.477	-0.156	0.140	0.647	1.300	0.757
	Max	2.724	0.204	0.613	1.502	2.365	1.470
CNN5							
	Min	0.881	-0.204	-0.340	0.096	1.111	0.001
	Max	2.501	0.174	0.511	1.383	2.184	1.377

Furthermore, GAP has been shown to have additional benefits, such as better resistance to adversarial attacks and better generalization performance than flattening. This is because GAP encourages the model to learn features that are robust to spatial transformations of the input, which can help the model generalize better to new data.

Figure 2.12 illustrates a shallow CNN consisting of a sequence of operations: a single convolutional layer with a ReLU activation function, followed by a max-pooling layer, and finally, a GAP layer before the input is fed into the dense network.

2.10. Appendix - Results

Table 2.3: **All stocks, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	0.008***	0.012***	0.011***
	(0.002)	(0.002)	(0.002)
$r_M - r_f$	-0.047	0.016	-0.048
	(0.061)	(0.056)	(0.055)
SMB	-0.183*	-0.040	0.021
	(0.100)	(0.090)	(0.089)
HML	-0.085	-0.082	-0.132*
	(0.085)	(0.077)	(0.076)
2-12 Momentum	-0.097**	-0.081*	-0.107**
	(0.046)	(0.042)	(0.041)
ST Reversal	-0.074	-0.137**	-0.175***
	(0.067)	(0.060)	(0.060)
LT Reversal	0.153*	0.173**	0.262***
	(0.087)	(0.078)	(0.077)
CIV	0.519**	0.120	0.053
	(0.231)	(0.209)	(0.206)
PIV	-0.304	-0.025	-0.014
	(0.236)	(0.213)	(0.210)
IVS$_{atm}$	-0.356***	0.088	0.245**
	(0.118)	(0.106)	(0.105)
IVS$_{otm}$	0.679***	0.181	0.170
	(0.133)	(0.121)	(0.119)
Skew	0.600***	0.244**	0.135
	(0.125)	(0.113)	(0.111)
VOV	0.159*	0.139*	0.144*
	(0.084)	(0.076)	(0.075)
Δ CIV	0.055	0.288***	0.182*
	(0.119)	(0.107)	(0.106)
Δ PIV	0.135	0.170*	0.086
	(0.108)	(0.098)	(0.096)
Observations	227	227	227
R^2	0.657	0.367	0.344
Adjusted R^2	0.634	0.325	0.301
Residual Std. Error	0.023	0.020	0.020
F Statistic	29.003***	8.766***	7.949***

Note: *p<0.1; **p<0.05; ***p<0.01

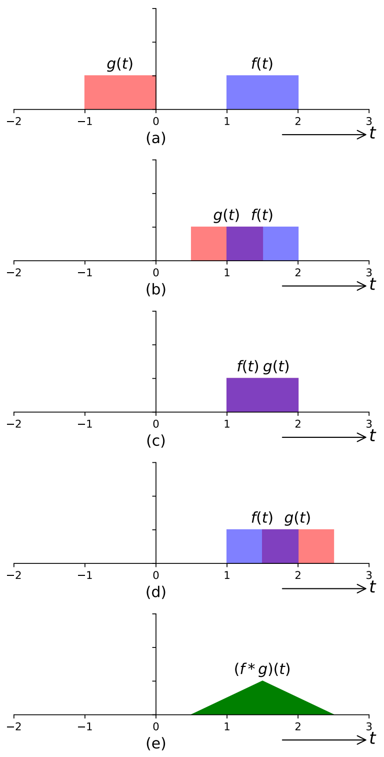


Figure 2.7: The figures above show the convolution operation $(f*g)(t)$ in continuous time. In this particular example, the kernel function, in red, $g(t)$, is equal to the input signal, in blue, $f(t)$, and they share the same area. In information theory, this is a known example where the convolution of two "rectangular" functions gives an output a "triangle." The more you slide the kernel function to the right, the kernel $g(t)$ will start overlapping with the signal $f(t)$. The overlapping area will start increasing (b), reaching the maximum at (c), and it will smoothly decay (d). The final figure (e) shows the convolution result.

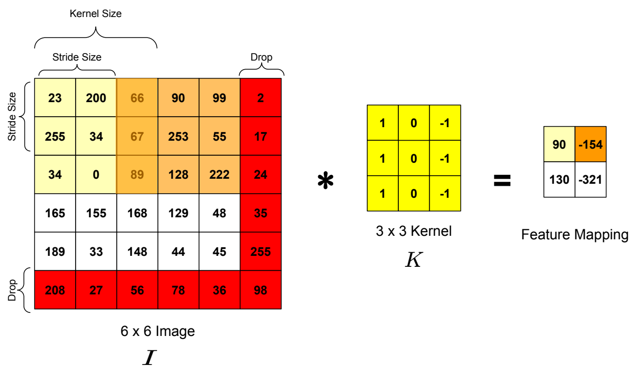


Figure 2.8: The figure above shows the convolution operation. A kernel filter with size 3x3 and stride equal to two is convolved to the 2-dimensional image matrix. The kernel filter moves clockwise and projects the output value to the Feature Mapping matrix. A padding is necessary to obtain the convolution in the figure. In this case, we show the "same" padding. Another popular option in literature is not to use padding at all. In case of no padding, the last column would have been dropped, and the feature mapping matrix would have been a 2x2 matrix.

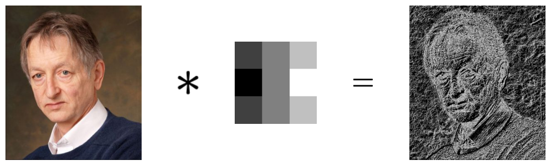


Figure 2.9: The figure above shows an example of an edge detection algorithm using the Sobel operator.

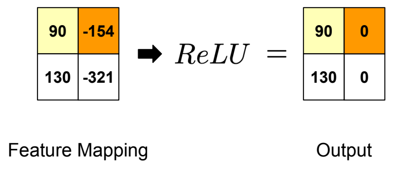


Figure 2.10: The figure above shows the output of a ReLU activation function applied to a 2×2 matrix.

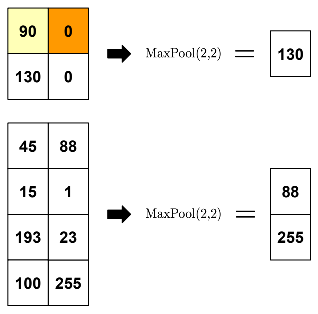


Figure 2.11: The figure above shows the max-pooling function. In this particular example, we show a max-pooling function with a kernel 2×2. That means the function extracts the maximum value in a pool as big as 2×2. Hence, the output of a 2×2 input has size 1×1, and similarly, the output of a 4×2 input has size 2×1.

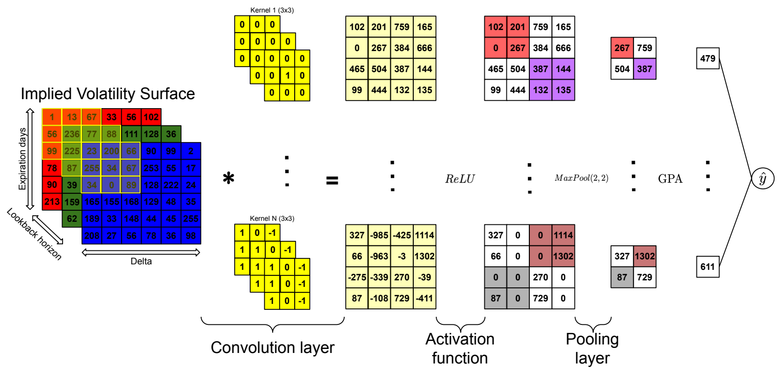


Figure 2.12: The figure above shows a building block of a CNN model. From left to right, we have the input image decomposed in an RGB tensor. To this tensor, a series of convolutions are applied using N different kernels, in this case, with size 3x3. Next, a ReLU activation function is applied to the previous step result. Finally, to reduce the image's complexity even more, a Pooling Layer is applied. The figure shows a classic MaxPool with kernel and stride size equal to two.

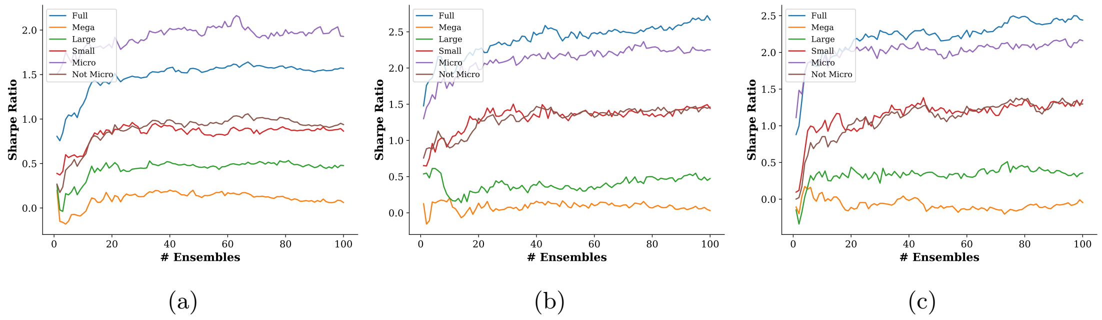


Figure 2.13: Sharpe ratios of ensemble-based returns (2.5) as a function of ensemble size. In particular, Figures 2.13a, 2.13b, and 2.13c show the performance of CNN1, CNN4, and CNN5, respectively.

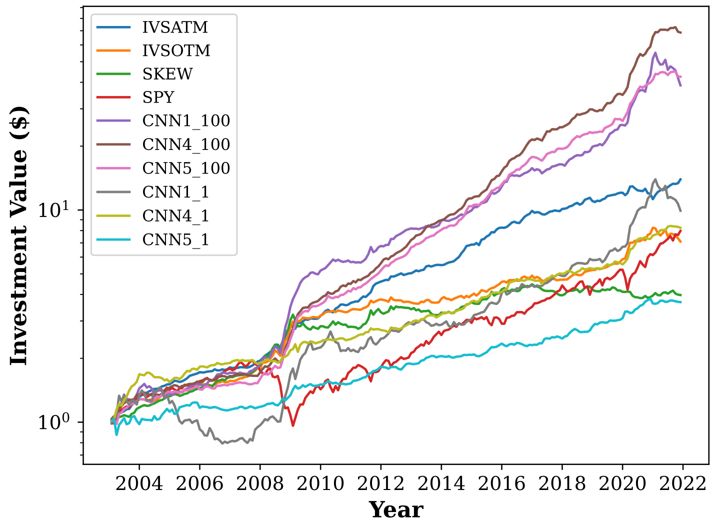


Figure 2.14: Cumulative Returns of the market neutral (long-short) strategy (2.5) for a full universe of stocks. CNNi_K refers to the ensemble of K CNNi models. IVSATM, IVSOTM, and SKEW are the top three option characteristics-based factors from [Neu+22] in terms of Sharpe ratio, and SPY is the S&P500 ETF.

Table 2.4: **Mega-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	-0.001	-0.001	-0.003
	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.065	0.107	0.062
	(0.073)	(0.073)	(0.071)
SMB	0.070	0.117	0.015
	(0.119)	(0.119)	(0.115)
HML	-0.061	-0.009	0.093
	(0.102)	(0.101)	(0.098)
2-12 Momentum	0.036	-0.029	-0.016
	(0.055)	(0.055)	(0.053)
ST Reversal	-0.238***	-0.154*	-0.202***
	(0.080)	(0.079)	(0.077)
LT Reversal	0.159	0.198*	0.124
	(0.103)	(0.103)	(0.100)
CIV	-0.443	-0.331	-0.444*
	(0.276)	(0.274)	(0.267)
PIV	0.206	0.242	0.376
	(0.282)	(0.280)	(0.272)
IVS$_{atm}$	-0.326**	0.031	0.195
	(0.141)	(0.140)	(0.136)
IVS$_{otm}$	0.232	-0.165	0.046
	(0.159)	(0.158)	(0.154)
Skew	0.666***	0.309**	0.147
	(0.149)	(0.148)	(0.144)
VOV	0.127	0.165*	0.209**
	(0.100)	(0.099)	(0.097)
Δ CIV	0.280**	0.246*	0.137
	(0.142)	(0.141)	(0.137)
Δ PIV	0.311**	0.106	-0.081
	(0.129)	(0.128)	(0.124)
Observations	227	227	227
R^2	0.523	0.100	0.098
Adjusted R^2	0.491	0.040	0.038
Residual Std. Error	0.027	0.027	0.026
F Statistic	16.591***	1.676*	1.636*
Note:			*p<0.1; **p<0.05; ***p<0.01

Table 2.5: **Large-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	-0.000	-0.001	-0.001
	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.153**	0.085	0.016
	(0.064)	(0.058)	(0.061)
SMB	0.118	-0.080	0.013
	(0.103)	(0.095)	(0.100)
HML	-0.022	-0.118	-0.076
	(0.088)	(0.081)	(0.085)
2-12 Momentum	-0.025	-0.052	-0.066
	(0.048)	(0.044)	(0.046)
ST Reversal	-0.031	-0.088	-0.159**
	(0.069)	(0.063)	(0.067)
LT Reversal	0.029	0.195**	0.160*
	(0.090)	(0.082)	(0.086)
CIV	-0.099	-0.180	-0.379
	(0.239)	(0.219)	(0.231)
PIV	-0.077	0.176	0.367
	(0.244)	(0.223)	(0.236)
IVS$_{atm}$	-0.451***	0.226**	0.368***
	(0.122)	(0.111)	(0.117)
IVS$_{otm}$	0.684***	-0.094	-0.154
	(0.138)	(0.126)	(0.133)
Skew	0.577***	0.419***	0.375***
	(0.129)	(0.118)	(0.125)
VOV	-0.060	0.200**	0.207**
	(0.087)	(0.079)	(0.084)
Δ CIV	-0.050	0.114	0.126
	(0.123)	(0.112)	(0.118)
Δ PIV	0.166	-0.150	-0.199*
	(0.112)	(0.102)	(0.108)
Observations	227	227	227
R^2	0.534	0.199	0.194
Adjusted R^2	0.503	0.146	0.141
Residual Std. Error	0.023	0.021	0.023
F Statistic	17.321***	3.752***	3.641***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.6: **Small-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	0.001	0.005**	0.005**
	(0.002)	(0.002)	(0.002)
$r_M - r_f$	-0.013	0.125	0.005
	(0.077)	(0.079)	(0.072)
SMB	-0.333***	-0.022	0.054
	(0.125)	(0.129)	(0.117)
HML	0.176*	0.065	0.006
	(0.107)	(0.110)	(0.099)
2-12 Momentum	0.022	0.007	-0.074
	(0.058)	(0.060)	(0.054)
ST Reversal	0.022	-0.050	-0.109
	(0.084)	(0.086)	(0.078)
LT Reversal	-0.022	-0.034	0.152
	(0.108)	(0.111)	(0.101)
CIV	0.347	0.374	0.207
	(0.290)	(0.298)	(0.270)
PIV	-0.068	-0.318	-0.181
	(0.295)	(0.304)	(0.275)
IVS$_{atm}$	-0.198	0.059	0.414***
	(0.147)	(0.152)	(0.137)
IVS$_{otm}$	0.511***	0.190	0.081
	(0.167)	(0.172)	(0.155)
Skew	0.540***	0.191	0.121
	(0.156)	(0.161)	(0.146)
VOV	-0.007	0.106	0.105
	(0.105)	(0.108)	(0.098)
Δ CIV	-0.039	0.011	-0.139
	(0.148)	(0.153)	(0.138)
Δ PIV	0.232*	0.124	-0.053
	(0.135)	(0.139)	(0.126)
Observations	227	227	227
R^2	0.457	0.132	0.190
Adjusted R^2	0.421	0.074	0.136
Residual Std. Error	0.028	0.029	0.026
F Statistic	12.723***	2.297***	3.547***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.7: **Micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	0.034***	0.041***	0.038***
	(0.006)	(0.006)	(0.006)
$r_M - r_f$	-0.512***	-0.389*	-0.245
	(0.187)	(0.204)	(0.204)
SMB	-0.228	0.061	0.291
	(0.305)	(0.333)	(0.332)
HML	-0.417	-0.341	-0.456
	(0.260)	(0.283)	(0.283)
2-12 Momentum	-0.253*	-0.376**	-0.378**
	(0.141)	(0.154)	(0.154)
ST Reversal	-0.265	-0.639***	-0.670***
	(0.204)	(0.222)	(0.222)
LT Reversal	0.397	0.434	0.568**
	(0.264)	(0.288)	(0.288)
CIV	1.769**	0.543	0.459
	(0.706)	(0.770)	(0.769)
PIV	-1.351*	-0.478	-0.666
	(0.720)	(0.785)	(0.785)
IVS$_{atm}$	-0.304	0.553	0.440
	(0.359)	(0.391)	(0.391)
IVS$_{otm}$	0.858**	0.385	0.741*
	(0.406)	(0.443)	(0.443)
Skew	0.372	-0.031	-0.280
	(0.381)	(0.416)	(0.416)
VOV	0.279	1.043***	0.833***
	(0.255)	(0.279)	(0.278)
Δ CIV	0.301	0.691*	0.774*
	(0.362)	(0.394)	(0.394)
Δ PIV	0.212	0.370	0.293
	(0.329)	(0.359)	(0.359)
Observations	227	227	227
R^2	0.354	0.240	0.219
Adjusted R^2	0.312	0.190	0.167
Residual Std. Error	0.069	0.075	0.075
F Statistic	8.305***	4.778***	4.235***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.8: **Not-micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, and CNN5 portfolios on the factor model that includes CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	CNN1	CNN4	CNN5
Intercept	0.002	0.004**	0.003*
	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.061	0.047	0.023
	(0.057)	(0.051)	(0.048)
SMB	-0.067	0.017	0.032
	(0.093)	(0.083)	(0.079)
HML	0.018	0.029	-0.012
	(0.079)	(0.070)	(0.067)
2-12 Momentum	-0.042	-0.002	-0.026
	(0.043)	(0.038)	(0.036)
ST Reversal	-0.059	-0.001	-0.086
	(0.062)	(0.055)	(0.053)
LT Reversal	0.063	0.074	0.147**
	(0.080)	(0.071)	(0.068)
CIV	0.150	-0.076	-0.107
	(0.215)	(0.191)	(0.182)
PIV	-0.081	0.130	0.132
	(0.219)	(0.195)	(0.186)
IVS$_{atm}$	-0.303***	0.089	0.286***
	(0.109)	(0.097)	(0.093)
IVS$_{otm}$	0.490***	0.029	0.005
	(0.124)	(0.110)	(0.105)
Skew	0.582***	0.329***	0.256***
	(0.116)	(0.103)	(0.098)
VOV	0.018	0.095	0.111*
	(0.078)	(0.069)	(0.066)
Δ CIV	-0.065	0.087	-0.043
	(0.110)	(0.098)	(0.093)
Δ PIV	0.120	0.066	-0.081
	(0.100)	(0.089)	(0.085)
Observations	227	227	227
R^2	0.512	0.205	0.216
Adjusted R^2	0.479	0.153	0.164
Residual Std. Error	0.021	0.019	0.018
F Statistic	15.865***	3.908***	4.169***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.9: The number of parameters for a single, 40-ensemble, and 100-ensemble for our set of CNN models. Similarly, Single c, 40-ensemble c, and 100-ensemble c show the total model complexity as defined in [KMZ24]: $c := P/T$, where $T = 973947$ is the total number of observations at the end of the sample (all models are trained using an expanding window).

Model	Single	40-Ensemble	100-Ensemble	Single c	40-Ensemble c	100-Ensemble c
CNN1	1921	76840	192100	0.002	0.079	0.197
CNN4	98241	3929640	9824100	0.101	4.035	10.087
CNN5	394561	15782440	39456100	0.405	16.205	40.512
NN1	23809	952360	2380900	0.024	0.978	2.445

Table 2.10: **All stocks, long-short portfolio.** Cross-sectional (Fama-MacBeth) regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. $ret\_$ are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; $ivol\_$ are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}^{ens}_{t,CNN}$	16.28***	8.30***	3.85**
	(4.38)	(4.42)	(2.44)
CIV	0.54**	0.52*	0.57**
	(2.02)	(1.91)	(2.14)
PIV	-0.70***	-0.64**	-0.68**
	(-2.58)	(-2.32)	(-2.46)
IVS$_{atm}$	-1.24***	-1.15***	-1.25***
	(-4.15)	(-3.80)	(-4.16)
IVS$_{otm}$	0.10	-0.13	-0.16
	(0.31)	(-0.40)	(-0.51)
Skew	-0.56*	-0.73**	-0.75**
	(-1.68)	(-2.19)	(-2.24)
VOV	-0.88**	-0.91**	-0.93**
	(-2.44)	(-2.50)	(-2.54)
ΔCIV	-0.22	-0.28	-0.33
	(-0.64)	(-0.85)	(-0.98)
ΔPIV	0.22	0.18	0.21
	(0.62)	(0.50)	(0.59)
ret_1_0	-0.48	-0.50	-0.54
	(-0.86)	(-0.89)	(-0.97)
ret_6_1	0.06	0.05	0.05
	(0.24)	(0.20)	(0.19)
ret_12_1	0.30**	0.28*	0.28*
	(2.01)	(1.90)	(1.91)
ret_18_1	-0.26**	-0.26**	-0.27**
	(-2.21)	(-2.16)	(-2.21)
rvol_21d	-8.68	-9.64	-9.34
	(-0.43)	(-0.48)	(-0.46)
rvol_252d	-125.12**	-116.52**	-113.90**
	(-2.15)	(-2.01)	(-1.96)
beta_21d	-0.08	-0.07	-0.08
	(-0.43)	(-0.42)	(-0.46)
beta_252d	0.70**	0.63*	0.63*
	(2.11)	(1.90)	(1.91)
ivol_capm_21d	6.02	6.74	6.67
	(0.31)	(0.35)	(0.35)
ivol_capm_252d	105.79*	98.22*	95.50*
	(1.89)	(1.76)	(1.71)
ami_126d	1.05	1.22	1.24
	(1.01)	(1.18)	(1.21)
Note:			*t>1.645; **t<1.960; ***t>2.576

Table 2.11: **Mega-cap segment, long-short portfolio.** Cross-sectional (Fama-MacBeth) regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. *ret_* are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; *ivol_* are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}_{t+1,CNN}$	-3.56	1.30	-6.29
	(-0.25)	(0.17)	(-0.98)
CIV	0.54	0.65	0.60
	(0.75)	(0.89)	(0.83)
PIV	0.11	0.32	0.25
	(0.14)	(0.44)	(0.34)
ΔCIV	0.45	0.45	0.43
	(0.45)	(0.43)	(0.41)
ΔPIV	-0.68	-0.75	-0.63
	(-0.65)	(-0.71)	(-0.60)
IVS$_{atm}$	-0.43	-0.33	-0.35
	(-0.43)	(-0.34)	(-0.36)
IVS$_{otm}$	-1.26	-1.14	-1.23
	(-1.54)	(-1.47)	(-1.58)
Skew	-0.53	-1.31	-1.29
	(-0.54)	(-1.31)	(-1.25)
VOV	-0.70	-0.64	-0.71
	(-1.04)	(-0.95)	(-1.06)
ret_1_0	-0.26	-0.32	-0.29
	(-0.34)	(-0.41)	(-0.38)
ret_6_1	0.16	0.17	0.16
	(0.42)	(0.44)	(0.42)
ret_12_1	0.35	0.35	0.35
	(1.11)	(1.09)	(1.11)
ret_18_1	0.05	0.03	0.03
	(0.23)	(0.16)	(0.14)
rvol_21d	-91.26**	-102.38**	-97.26**
	(-2.19)	(-2.45)	(-2.34)
rvol_252d	-31.30	-15.06	-22.78
	(-0.29)	(-0.14)	(-0.21)
beta_21d	0.58	0.68*	0.67*
	(1.57)	(1.81)	(1.79)
beta_252d	-0.48	-0.63	-0.56
	(-0.54)	(-0.71)	(-0.64)
ivol_capm_21d	80.89**	89.25**	84.49**
	(2.13)	(2.36)	(2.24)
ivol_capm_252d	17.16	1.28	8.71
	(0.18)	(0.01)	(0.09)
ami_126d	59.06	63.12	72.27
	(0.62)	(0.65)	(0.75)

Note: *t>1.645; **t<1.960; ***t>2.576

Table 2.12: **Large-cap segment, long-short portfolio.** Cross-sectional (Fama-MacBeth) regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. $ret\_$ are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; $ivol\_$ are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}_{t+1,CNN}$	5.82	-1.05	-0.50
	(0.67)	(-0.28)	(-0.15)
CIV	0.43	0.33	0.37
	(1.02)	(0.79)	(0.88)
PIV	-0.28	-0.38	-0.33
	(-0.67)	(-0.91)	(-0.79)
ΔCIV	0.02	-0.06	-0.08
	(0.03)	(-0.11)	(-0.14)
ΔPIV	0.25	0.28	0.27
	(0.46)	(0.50)	(0.50)
IVS$_{atm}$	-0.70	-0.71	-0.69
	(-1.27)	(-1.25)	(-1.23)
IVS$_{otm}$	0.30	0.20	0.15
	(0.64)	(0.42)	(0.33)
Skew	-0.95**	-1.03**	-1.03**
	(-2.01)	(-2.08)	(-2.08)
VOV	-0.97**	-0.93*	-0.93*
	(-2.00)	(-1.92)	(-1.92)
ret_1_0	-0.69	-0.65	-0.63
	(-0.95)	(-0.91)	(-0.87)
ret_6_1	0.25	0.24	0.25
	(0.81)	(0.77)	(0.79)
ret_12_1	0.25	0.25	0.25
	(1.26)	(1.24)	(1.23)
ret_18_1	-0.24	-0.24	-0.24
	(-1.53)	(-1.50)	(-1.48)
rvol_21d	2.96	3.39	4.67
	(0.10)	(0.11)	(0.16)
rvol_252d	38.71	34.80	36.87
	(0.37)	(0.33)	(0.35)
beta_21d	-0.26	-0.26	-0.26
	(-1.06)	(-1.04)	(-1.08)
beta_252d	0.13	0.16	0.15
	(0.21)	(0.25)	(0.24)
ivol_capm_21d	2.28	1.63	0.29
	(0.08)	(0.06)	(0.01)
ivol_capm_252d	-59.77	-53.94	-56.82
	(-0.62)	(-0.57)	(-0.60)
ami_126d	4.24	0.60	0.05
	(0.25)	(0.04)	(0.00)

Note: *t>1.645; **t<1.960; ***t>2.576

Table 2.13: **Small-cap segment, long-short portfolio.** Cross-sectional (Fama-MacBeth) regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. *ret_* are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; *ivol_* are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}_{t+1,CNN}$	19.85***	10.43***	6.88**
	(2.86)	(3.71)	(2.47)
CIV	0.85**	0.74*	0.77*
	(2.02)	(1.78)	(1.86)
PIV	-0.78**	-0.74*	-0.79**
	(-1.98)	(-1.92)	(-2.03)
ΔCIV	-0.45	-0.61	-0.65
	(-0.86)	(-1.18)	(-1.26)
ΔPIV	0.29	0.35	0.38
	(0.56)	(0.69)	(0.74)
IVS$_{atm}$	-1.62***	-1.48***	-1.57***
	(-3.70)	(-3.36)	(-3.53)
IVS$_{otm}$	-0.12	-0.38	-0.39
	(-0.29)	(-0.97)	(-0.97)
Skew	0.30	0.11	0.09
	(0.59)	(0.22)	(0.18)
VOV	-0.98**	-0.94**	-0.94**
	(-2.09)	(-2.10)	(-2.08)
ret_1_0	-0.61	-0.61	-0.65
	(-0.98)	(-0.98)	(-1.04)
ret_6_1	0.14	0.12	0.12
	(0.50)	(0.43)	(0.43)
ret_12_1	0.17	0.17	0.18
	(1.01)	(1.01)	(1.07)
ret_18_1	-0.23*	-0.22*	-0.23*
	(-1.75)	(-1.65)	(-1.75)
rvol_21d	-2.11	-2.74	-2.36
	(-0.07)	(-0.10)	(-0.08)
rvol_252d	-143.25	-133.55	-140.45
	(-1.22)	(-1.15)	(-1.20)
beta_21d	-0.27	-0.26	-0.26
	(-0.97)	(-0.93)	(-0.94)
beta_252d	0.63	0.55	0.58
	(1.22)	(1.06)	(1.13)
ivol_capm_21d	4.33	4.12	4.33
	(0.16)	(0.15)	(0.16)
ivol_capm_252d	121.52	113.08	119.74
	(1.10)	(1.03)	(1.08)
ami_126d	1.10	0.64	1.02
	(0.25)	(0.15)	(0.23)
Note:			*t>1.645; **t<1.960; ***t>2.576

Table 2.14: **Micro-cap segment, long-short portfolio.** Cross-sectional (Fama-MacBeth) regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. *ret_* are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; *ivol_* are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}_{t+1,CNN}$	16.11*	6.71	2.11
	(1.94)	(1.26)	(0.37)
CIV	-0.49	-0.48	-0.35
	(-0.83)	(-0.83)	(-0.56)
PIV	-0.79	-0.65	-0.70
	(-1.42)	(-1.14)	(-1.20)
ΔCIV	0.36	0.35	0.17
	(0.46)	(0.44)	(0.22)
ΔPIV	0.23	0.12	0.24
	(0.29)	(0.15)	(0.30)
IVS$_{atm}$	-0.29	-0.17	-0.35
	(-0.32)	(-0.19)	(-0.37)
IVS$_{otm}$	-0.58	-1.05	-1.02
	(-0.58)	(-1.13)	(-1.13)
Skew	-1.32	-1.63**	-1.68**
	(-1.58)	(-2.12)	(-2.24)
VOV	-3.94***	-4.43***	-4.50***
	(-2.77)	(-3.07)	(-3.07)
ret_1_0	-0.25	-0.26	-0.38
	(-0.32)	(-0.34)	(-0.49)
ret_6_1	-0.31	-0.30	-0.33
	(-0.71)	(-0.69)	(-0.76)
ret_12_1	0.33	0.24	0.25
	(0.94)	(0.70)	(0.71)
ret_18_1	-0.21	-0.16	-0.15
	(-0.76)	(-0.55)	(-0.51)
rvol_21d	-98.64	-91.46	-93.97
	(-1.60)	(-1.53)	(-1.59)
rvol_252d	197.02	241.10	261.30
	(0.60)	(0.74)	(0.80)
beta_21d	0.08	0.09	0.08
	(0.30)	(0.34)	(0.29)
beta_252d	-0.40	-0.63	-0.61
	(-0.40)	(-0.63)	(-0.62)
ivol_capm_21d	83.45	75.97	78.99
	(1.40)	(1.30)	(1.37)
ivol_capm_252d	-193.20	-236.00	-256.20
	(-0.60)	(-0.74)	(-0.80)
ami_126d	0.57	0.34	0.36
	(0.42)	(0.26)	(0.27)

Note: *t>1.645; **t<1.960; ***t>2.576

Table 2.15: **Not-micro-cap segment, long-short portfolio.** Cross-sectional regression of next-month security returns $r_{i,t+1}$ for the full stock universe on a set of predictive characteristics over the time period February 2003 to December 2021. $\hat{R}^{ens}_{t,CNN}$ is the prediction in (2.4). CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV and ΔPIV are option characteristics from [Neu+22]. $ret\_$ are momentum- and short-term-reversal-based characteristics; β are market beta characteristics; $ivol\_$ are idiosyncratic volatility characteristics. All these are taken from [JKP22]. T-statistics are reported in parentheses.

	CNN1	CNN4	CNN5
$\hat{R}_{t+1,CNN}$	18.18***	7.50***	4.60**
	(3.38)	(3.15)	(2.17)
CIV	0.61*	0.53*	0.56*
	(1.93)	(1.69)	(1.77)
PIV	-0.50*	-0.49	-0.53*
	(-1.66)	(-1.59)	(-1.71)
ΔCIV	-0.23	-0.33	-0.35
	(-0.58)	(-0.83)	(-0.89)
ΔPIV	0.19	0.18	0.21
	(0.46)	(0.45)	(0.52)
IVS$_{atm}$	-1.12***	-1.02***	-1.08***
	(-3.58)	(-3.23)	(-3.43)
IVS$_{otm}$	0.15	-0.11	-0.13
	(0.44)	(-0.34)	(-0.39)
Skew	-0.36	-0.53	-0.55
	(-0.99)	(-1.48)	(-1.53)
VOV	-0.83**	-0.84**	-0.85**
	(-2.20)	(-2.24)	(-2.25)
ret_1_0	-0.66	-0.70	-0.70
	(-1.08)	(-1.14)	(-1.15)
ret_6_1	0.14	0.12	0.13
	(0.54)	(0.46)	(0.48)
ret_12_1	0.27*	0.27*	0.27*
	(1.73)	(1.71)	(1.74)
ret_18_1	-0.23*	-0.23*	-0.23*
	(-1.89)	(-1.85)	(-1.88)
rvol_21d	-4.91	-5.17	-4.94
	(-0.24)	(-0.25)	(-0.24)
rvol_252d	-101.99	-91.69	-93.77
	(-1.44)	(-1.30)	(-1.32)
beta_21d	-0.08	-0.09	-0.09
	(-0.40)	(-0.42)	(-0.41)
beta_252d	0.48	0.42	0.43
	(1.10)	(0.97)	(1.00)
ivol_capm_21d	7.84	7.85	7.73
	(0.40)	(0.40)	(0.39)
ivol_capm_252d	79.41	70.26	71.99
	(1.20)	(1.07)	(1.09)
ami_126d	2.58	2.79	3.06
	(0.70)	(0.76)	(0.83)

Note: *t>1.645; **t<1.960; ***t>2.576

2.10.1. Long-only Portfolio Performance

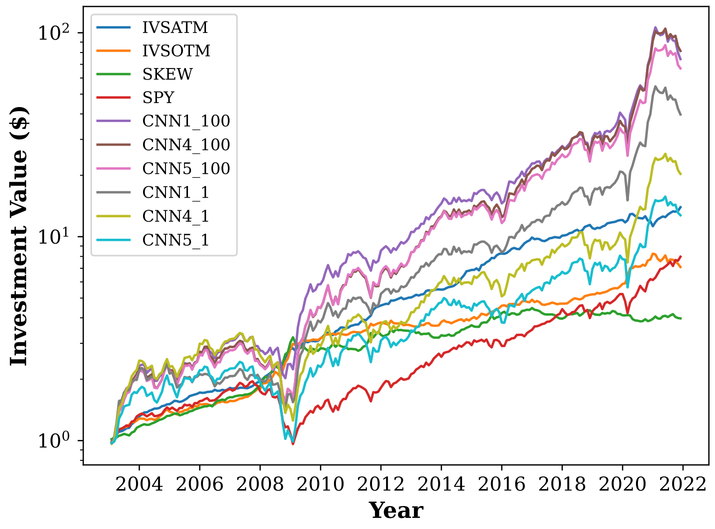


Figure 2.15: **Long-only strategy** (2.7)**.** Cumulative Returns of (2.7) for the full universe of stocks. CNNi_K refers to the ensemble of K CNNi models. IVSATM, IVSOTM, and SKEW are the top three option characteristics-based factors from [Neu+22], and SPY is the S&P500 ETF.

Table 2.16: **All stocks, long-only portfolio** (2.7)**.** Monthly OLS regression of the CNN1, CNN4, and CNN5 long-only portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$
Intercept	0.004***	0.005***	0.005***
	(0.002)	(0.001)	(0.001)
$r_M - r_f$	0.835***	0.944***	0.930***
	(0.049)	(0.041)	(0.042)
SMB	0.248***	0.422***	0.473***
	(0.079)	(0.066)	(0.068)
HML	0.048	0.016	-0.014
	(0.067)	(0.056)	(0.058)
2-12 Momentum	-0.167***	-0.127***	-0.145***
	(0.037)	(0.031)	(0.032)
ST Reversal	0.048	0.015	0.008
	(0.053)	(0.044)	(0.046)
LT Reversal	0.099	0.130**	0.176***
	(0.068)	(0.057)	(0.059)
CIV	0.655***	0.568***	0.467***
	(0.183)	(0.153)	(0.158)
PIV	-0.105	-0.026	0.016
	(0.187)	(0.157)	(0.161)
IVS$_{atm}$	-0.299***	-0.157**	-0.050
	(0.093)	(0.078)	(0.080)
IVS$_{otm}$	0.401***	0.171*	0.142
	(0.105)	(0.088)	(0.091)
Skew	0.458***	0.279***	0.221**
	(0.099)	(0.083)	(0.085)
VOV	0.115*	0.125**	0.141**
	(0.066)	(0.056)	(0.057)
Δ CIV	0.035	0.112	0.029
	(0.094)	(0.079)	(0.081)
Δ PIV	-0.007	0.036	-0.022
	(0.085)	(0.072)	(0.074)
Observations	227	227	227
R^2	0.946	0.967	0.963
Adjusted R^2	0.943	0.965	0.961
Residual Std. Error	0.018	0.015	0.015
F Statistic	267.669***	444.960***	395.497***
Note:			*p<0.1; **p<0.05; ***p<0.01

2.10.2. The Impact of Costs

Table 2.17: Annualized Sharpe Ratio calculated for equal-weighted portfolios constructed using our 100-Ensemble models (CNNi and NN1) and Ridge Regression predictions. We consider both transaction and short-sale fees of 0, 20, and 40 bps for the micro-cap group, and fees of 0, 10, and 20 bps for all other groups.

		SR_{full}	SR_{mega}	SR_{large}	SR_{small}	SR_{micro}	SR_{not_micro}
Base Fee: 0bps							
	Model						
	CNN1	1.57	0.06	0.47	0.86	1.92	0.94
	CNN4	2.66	0.03	0.47	1.45	2.22	1.45
	CNN5	2.44	-0.05	0.36	1.34	2.14	1.29
	Ridge$_{z=0.1}$	1.70	0.43	0.59	1.09	1.93	0.98
	NN1	2.04	0.31	0.63	1.30	2.53	1.22
Base Fee: 10bps							
	Model						
	CNN1	1.20	-0.26	0.11	0.54	1.67	0.54
	CNN4	2.05	-0.42	-0.08	1.03	1.95	0.84
	CNN5	1.80	-0.51	-0.16	0.89	1.86	0.65
	Ridge$_{z=0.1}$	1.30	0.09	0.23	0.74	1.56	0.61
	NN1	1.62	-0.03	0.23	0.94	2.21	0.79
Base Fee: 20bps							
	Model						
	CNN1	0.83	-0.59	-0.26	0.21	1.41	0.13
	CNN4	1.43	-0.87	-0.62	0.61	1.67	0.23
	CNN5	1.16	-0.98	-0.69	0.43	1.58	0.00
	Ridge$_{z=0.1}$	0.90	-0.24	-0.13	0.39	1.19	0.23
	NN1	1.19	-0.38	-0.16	0.58	1.89	0.36

2.11. Appendix - Additional Results

2.11.1. Ridge Regression Results

Table 2.18: **All stocks, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Full Sample: 1.70.

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	0.006***	0.006***	0.006***	0.006***	0.006***
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.066	0.066	0.065	0.059	0.059
	(0.051)	(0.051)	(0.051)	(0.051)	(0.050)
SMB	-0.028	-0.028	-0.028	-0.026	-0.042
	(0.083)	(0.083)	(0.083)	(0.082)	(0.081)
HML	-0.139*	-0.139**	-0.141**	-0.140**	-0.158**
	(0.071)	(0.071)	(0.070)	(0.070)	(0.069)
2-12 Momentum	0.109***	0.109***	0.108***	0.107***	0.117***
	(0.038)	(0.038)	(0.038)	(0.038)	(0.038)
ST Reversal	0.013	0.013	0.016	0.022	0.029
	(0.055)	(0.055)	(0.055)	(0.055)	(0.054)
LT Reversal	0.223***	0.223***	0.226***	0.223***	0.233***
	(0.072)	(0.072)	(0.072)	(0.071)	(0.070)
CIV	0.473**	0.473**	0.475**	0.500***	0.518***
	(0.192)	(0.192)	(0.191)	(0.190)	(0.188)
PIV	-0.169	-0.169	-0.171	-0.193	-0.190
	(0.195)	(0.195)	(0.195)	(0.194)	(0.192)
IVS$_{atm}$	-0.010	-0.010	-0.009	-0.014	-0.012
	(0.097)	(0.097)	(0.097)	(0.097)	(0.096)
IVS$_{otm}$	0.412***	0.412***	0.408***	0.405***	0.353***
	(0.110)	(0.110)	(0.110)	(0.110)	(0.108)
Skew	0.549***	0.549***	0.554***	0.555***	0.584***
	(0.104)	(0.104)	(0.103)	(0.103)	(0.102)
VOV	0.152**	0.152**	0.153**	0.155**	0.137**
	(0.069)	(0.069)	(0.069)	(0.069)	(0.068)
Δ CIV	-0.055	-0.055	-0.060	-0.066	-0.070
	(0.098)	(0.098)	(0.098)	(0.098)	(0.096)
Δ PIV	0.145	0.145	0.142	0.149*	0.158*
	(0.089)	(0.089)	(0.089)	(0.089)	(0.088)
Observations	227	227	227	227	227
R^2	0.734	0.734	0.735	0.738	0.743
Adjusted R^2	0.717	0.717	0.718	0.721	0.726
Residual Std. Error	0.019	0.019	0.019	0.019	0.018
F Statistic	41.861***	41.874***	42.093***	42.653***	43.801***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.19: **Mega-cap segment, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS_{atm}, IVS_{otm}, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Mega Sample: 0.42.

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	-0.001	-0.001	-0.001	-0.001	-0.001
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.173**	0.173**	0.173**	0.178**	0.180**
	(0.073)	(0.073)	(0.073)	(0.073)	(0.073)
SMB	0.055	0.055	0.057	0.060	0.048
	(0.118)	(0.118)	(0.118)	(0.119)	(0.119)
HML	-0.165	-0.165	-0.165	-0.153	-0.178*
	(0.101)	(0.101)	(0.101)	(0.101)	(0.101)
2-12 Momentum	0.190***	0.190***	0.190***	0.188***	0.199***
	(0.055)	(0.055)	(0.055)	(0.055)	(0.055)
ST Reversal	0.004	0.004	0.000	-0.000	-0.003
	(0.079)	(0.079)	(0.079)	(0.079)	(0.079)
LT Reversal	0.214**	0.214**	0.215**	0.214**	0.232**
	(0.102)	(0.102)	(0.102)	(0.103)	(0.103)
CIV	-0.050	-0.050	-0.057	-0.056	-0.025
	(0.273)	(0.273)	(0.273)	(0.275)	(0.275)
PIV	0.070	0.070	0.075	0.067	0.034
	(0.279)	(0.279)	(0.279)	(0.281)	(0.280)
IVS_{atm}	-0.262*	-0.262*	-0.261*	-0.272*	-0.311**
	(0.139)	(0.139)	(0.139)	(0.140)	(0.140)
IVS_{otm}	0.215	0.215	0.214	0.235	0.226
	(0.157)	(0.157)	(0.157)	(0.158)	(0.158)
Skew	0.597***	0.597***	0.593***	0.579***	0.571***
	(0.148)	(0.148)	(0.148)	(0.149)	(0.149)
VOV	0.551***	0.551***	0.550***	0.547***	0.540***
	(0.099)	(0.099)	(0.099)	(0.100)	(0.099)
Δ CIV	-0.077	-0.077	-0.075	-0.071	-0.014
	(0.140)	(0.140)	(0.140)	(0.141)	(0.141)
Δ PIV	0.174	0.174	0.173	0.181	0.229*
	(0.127)	(0.127)	(0.127)	(0.128)	(0.128)
Observations	227	227	227	227	227
R^2	0.444	0.444	0.443	0.439	0.447
Adjusted R^2	0.408	0.408	0.406	0.402	0.410
Residual Std. Error	0.027	0.027	0.027	0.027	0.027
F Statistic	12.106***	12.106***	12.051***	11.847***	12.240***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.20: **Large-cap segment, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Large Sample: 0.60.

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	-0.003	-0.003	-0.003	-0.003	-0.002
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.214***	0.214***	0.214***	0.204***	0.188***
	(0.062)	(0.062)	(0.062)	(0.062)	(0.061)
SMB	0.321***	0.321***	0.322***	0.304***	0.327***
	(0.101)	(0.101)	(0.101)	(0.101)	(0.099)
HML	-0.239***	-0.239***	-0.237***	-0.243***	-0.283***
	(0.086)	(0.086)	(0.086)	(0.086)	(0.085)
2-12 Momentum	0.185***	0.185***	0.184***	0.181***	0.174***
	(0.047)	(0.047)	(0.047)	(0.047)	(0.046)
ST Reversal	0.183***	0.183***	0.183***	0.174**	0.191***
	(0.068)	(0.068)	(0.068)	(0.067)	(0.066)
LT Reversal	0.194**	0.194**	0.193**	0.197**	0.247***
	(0.088)	(0.088)	(0.088)	(0.087)	(0.086)
CIV	-0.028	-0.028	-0.035	-0.035	-0.030
	(0.234)	(0.234)	(0.234)	(0.233)	(0.230)
PIV	0.046	0.046	0.052	0.064	0.046
	(0.239)	(0.239)	(0.239)	(0.238)	(0.234)
IVS$_{atm}$	-0.072	-0.072	-0.074	-0.052	-0.062
	(0.119)	(0.119)	(0.119)	(0.118)	(0.117)
IVS$_{otm}$	0.281**	0.281**	0.279**	0.261*	0.265**
	(0.135)	(0.135)	(0.135)	(0.134)	(0.132)
Skew	0.587***	0.587***	0.590***	0.595***	0.568***
	(0.127)	(0.127)	(0.126)	(0.126)	(0.124)
VOV	0.262***	0.262***	0.260***	0.275***	0.277***
	(0.085)	(0.085)	(0.085)	(0.084)	(0.083)
Δ CIV	-0.156	-0.156	-0.154	-0.170	-0.194
	(0.120)	(0.120)	(0.120)	(0.119)	(0.118)
Δ PIV	0.258**	0.258**	0.258**	0.234**	0.256**
	(0.109)	(0.109)	(0.109)	(0.109)	(0.107)
Observations	227	227	227	227	227
R^2	0.557	0.557	0.557	0.556	0.570
Adjusted R^2	0.528	0.528	0.527	0.527	0.542
Residual Std. Error	0.023	0.023	0.023	0.023	0.022
F Statistic	19.063***	19.062***	19.005***	18.971***	20.088***
Note:					*p<0.1; **p<0.05; ***p<0.01

Table 2.21: **Small-cap segment, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Small Sample: 1.09.

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	0.000	0.000	0.000	0.000	0.000
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.038	0.038	0.035	0.032	0.045
	(0.067)	(0.067)	(0.067)	(0.067)	(0.067)
SMB	-0.022	-0.022	-0.026	-0.031	-0.036
	(0.109)	(0.109)	(0.109)	(0.109)	(0.109)
HML	0.026	0.026	0.020	0.020	0.032
	(0.093)	(0.093)	(0.093)	(0.093)	(0.092)
2-12 Momentum	0.174***	0.173***	0.173***	0.173***	0.162***
	(0.051)	(0.051)	(0.051)	(0.050)	(0.050)
ST Reversal	-0.015	-0.015	-0.015	-0.016	-0.001
	(0.073)	(0.073)	(0.073)	(0.073)	(0.073)
LT Reversal	0.111	0.111	0.116	0.121	0.099
	(0.095)	(0.095)	(0.095)	(0.094)	(0.094)
CIV	0.121	0.121	0.110	0.107	0.050
	(0.253)	(0.253)	(0.253)	(0.252)	(0.251)
PIV	0.153	0.153	0.166	0.172	0.228
	(0.258)	(0.258)	(0.258)	(0.257)	(0.256)
IVS$_{atm}$	0.194	0.194	0.199	0.201	0.173
	(0.129)	(0.129)	(0.129)	(0.128)	(0.128)
IVS$_{otm}$	0.386***	0.385***	0.380***	0.365**	0.376**
	(0.146)	(0.146)	(0.146)	(0.145)	(0.145)
Skew	0.370***	0.370***	0.375***	0.374***	0.408***
	(0.137)	(0.137)	(0.137)	(0.136)	(0.136)
VOV	0.041	0.041	0.044	0.044	0.040
	(0.091)	(0.091)	(0.092)	(0.091)	(0.091)
Δ CIV	-0.027	-0.026	-0.029	-0.021	0.015
	(0.130)	(0.130)	(0.130)	(0.129)	(0.129)
Δ PIV	0.125	0.126	0.123	0.141	0.169
	(0.118)	(0.118)	(0.118)	(0.118)	(0.117)
Observations	227	227	227	227	227
R^2	0.526	0.526	0.526	0.526	0.532
Adjusted R^2	0.494	0.494	0.495	0.495	0.501
Residual Std. Error	0.025	0.025	0.025	0.025	0.024
F Statistic	16.786***	16.785***	16.805***	16.793***	17.197***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.22: **Micro-cap segment, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Micro Sample: 1.90.

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	0.015***	0.015***	0.015***	0.015***	0.018***
	(0.004)	(0.004)	(0.004)	(0.004)	(0.005)
$r_M - r_f$	0.242*	0.242*	0.246*	0.237*	-0.011
	(0.139)	(0.139)	(0.138)	(0.139)	(0.147)
SMB	-0.030	-0.030	-0.031	-0.030	-0.193
	(0.226)	(0.226)	(0.225)	(0.226)	(0.240)
HML	-0.021	-0.021	-0.028	-0.013	-0.001
	(0.192)	(0.192)	(0.192)	(0.192)	(0.205)
2-12 Momentum	-0.149	-0.149	-0.147	-0.139	-0.125
	(0.104)	(0.104)	(0.104)	(0.104)	(0.111)
ST Reversal	0.172	0.172	0.169	0.197	0.107
	(0.151)	(0.151)	(0.151)	(0.151)	(0.161)
LT Reversal	-0.116	-0.116	-0.112	-0.141	0.039
	(0.195)	(0.195)	(0.195)	(0.196)	(0.208)
CIV	1.241**	1.241**	1.252**	1.270**	1.729***
	(0.522)	(0.522)	(0.521)	(0.523)	(0.556)
PIV	-0.972*	-0.972*	-0.979*	-0.991*	-1.390**
	(0.533)	(0.533)	(0.532)	(0.533)	(0.567)
IVS$_{atm}$	0.284	0.284	0.278	0.268	0.415
	(0.266)	(0.266)	(0.265)	(0.266)	(0.283)
IVS$_{otm}$	0.320	0.320	0.320	0.303	0.285
	(0.301)	(0.301)	(0.300)	(0.301)	(0.320)
Skew	0.947***	0.947***	0.949***	0.958***	0.815***
	(0.282)	(0.282)	(0.282)	(0.282)	(0.300)
VOV	0.230	0.230	0.222	0.214	0.456**
	(0.189)	(0.189)	(0.189)	(0.189)	(0.201)
Δ CIV	-0.535**	-0.535**	-0.523*	-0.532**	-0.595**
	(0.268)	(0.268)	(0.267)	(0.268)	(0.285)
Δ PIV	-0.422*	-0.422*	-0.418*	-0.417*	-0.295
	(0.244)	(0.244)	(0.243)	(0.244)	(0.259)
Observations	227	227	227	227	227
R^2	0.345	0.345	0.348	0.345	0.333
Adjusted R^2	0.302	0.302	0.304	0.302	0.289
Residual Std. Error	0.051	0.051	0.051	0.051	0.054
F Statistic	7.982***	7.982***	8.067***	7.974***	7.556***
Note:					*p<0.1; **p<0.05; ***p<0.01

Table 2.23: **Not-micro-cap segment, ridge long-short portfolio** Monthly OLS regression of the Ridge portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. z is the penalty term. Sharpe Ratio Non-Micro Sample: 0.99

	$z = 10^{-5}$	$z = 10^{-3}$	$z = 10^{-1}$	$z = 10^{0}$	$z = 10^{1}$
Intercept	-0.001	-0.001	-0.001	-0.001	-0.001
	(0.002)	(0.002)	(0.002)	(0.002)	(0.001)
$r_M - r_f$	0.118**	0.118**	0.118**	0.120**	0.121**
	(0.049)	(0.049)	(0.049)	(0.049)	(0.048)
SMB	0.151*	0.151*	0.152*	0.152*	0.160**
	(0.080)	(0.080)	(0.080)	(0.080)	(0.078)
HML	-0.137**	-0.137**	-0.139**	-0.140**	-0.150**
	(0.068)	(0.068)	(0.068)	(0.068)	(0.067)
2-12 Momentum	0.146***	0.146***	0.146***	0.145***	0.144***
	(0.037)	(0.037)	(0.037)	(0.037)	(0.036)
ST Reversal	0.053	0.053	0.054	0.055	0.051
	(0.054)	(0.054)	(0.054)	(0.053)	(0.052)
LT Reversal	0.173**	0.173**	0.172**	0.169**	0.179***
	(0.069)	(0.069)	(0.069)	(0.069)	(0.068)
CIV	-0.002	-0.002	-0.001	-0.021	-0.018
	(0.185)	(0.185)	(0.185)	(0.185)	(0.181)
PIV	0.203	0.203	0.203	0.221	0.228
	(0.189)	(0.189)	(0.189)	(0.189)	(0.184)
IVS$_{atm}$	0.061	0.061	0.060	0.061	0.057
	(0.094)	(0.094)	(0.094)	(0.094)	(0.092)
IVS$_{otm}$	0.301***	0.301***	0.299***	0.303***	0.285***
	(0.107)	(0.107)	(0.107)	(0.106)	(0.104)
Skew	0.519***	0.519***	0.522***	0.517***	0.533***
	(0.100)	(0.100)	(0.100)	(0.100)	(0.098)
VOV	0.148**	0.148**	0.150**	0.149**	0.142**
	(0.067)	(0.067)	(0.067)	(0.067)	(0.065)
Δ CIV	-0.055	-0.055	-0.057	-0.051	-0.063
	(0.095)	(0.095)	(0.095)	(0.095)	(0.093)
Δ PIV	0.209**	0.209**	0.211**	0.216**	0.212**
	(0.086)	(0.086)	(0.086)	(0.086)	(0.084)
Observations	227	227	227	227	227
R^2	0.681	0.681	0.683	0.683	0.698
Adjusted R^2	0.660	0.660	0.662	0.662	0.678
Residual Std. Error	0.018	0.018	0.018	0.018	0.018
F Statistic	32.393***	32.393***	32.555***	32.660***	34.945***

Note: *p<0.1; **p<0.05; ***p<0.01

2.11.2. Comparison: Simple NN against CNN

Table 2.24: **All stocks, long-short.** Monthly OLS regression of the CNNs portfolios on the factor model that includes the NN, CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. *As we can see, the alphas survive for any CNNi even if the NN factor portfolio is included on the right-hand side.*

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$
Intercept	0.004**	0.009***	0.009***
	(0.001)	(0.002)	(0.002)
$r_M - r_f$	-0.134***	-0.045	-0.096*
	(0.047)	(0.048)	(0.050)
SMB	-0.170**	-0.030	0.028
	(0.076)	(0.078)	(0.081)
HML	-0.035	-0.047	-0.104
	(0.065)	(0.066)	(0.069)
2-12 Momentum	-0.142***	-0.112***	-0.132***
	(0.035)	(0.036)	(0.038)
ST Reversal	-0.065	-0.131**	-0.170***
	(0.051)	(0.052)	(0.054)
LT Reversal	0.053	0.102	0.205***
	(0.066)	(0.068)	(0.071)
CIV	0.242	-0.076	-0.104
	(0.177)	(0.182)	(0.190)
PIV	-0.153	0.082	0.071
	(0.180)	(0.184)	(0.192)
IVS$_{atm}$	-0.394***	0.061	0.224**
	(0.089)	(0.092)	(0.096)
IVS$_{otm}$	0.427***	0.002	0.028
	(0.103)	(0.106)	(0.110)
Skew	0.222**	-0.024	-0.079
	(0.100)	(0.102)	(0.107)
VOV	0.089	0.089	0.104
	(0.064)	(0.066)	(0.068)
Δ CIV	0.049	0.283***	0.179*
	(0.090)	(0.092)	(0.096)
Δ PIV	0.039	0.102	0.032
	(0.082)	(0.084)	(0.088)
NN	0.619***	0.438***	0.350***
	(0.049)	(0.051)	(0.053)
Observations	227	227	227
R^2	0.803	0.532	0.456
Adjusted R^2	0.789	0.499	0.418
Residual Std. Error	0.017	0.018	0.018
F Statistic	57.399***	15.985***	11.814***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.25: **All stocks, long-short.** Monthly OLS regression of the NN portfolio on the factor model that includes the CNN, CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. *As we can see, the NN's alpha does not survive when the CNN1 factor portfolio is included in the right-hand side.*

	$NN1_{ew}$
Intercept	0.001
	(0.002)
$r_M - r_f$	0.172***
	(0.049)
SMB	0.104
	(0.081)
HML	-0.022
	(0.068)
2-12 Momentum	0.139***
	(0.037)
ST Reversal	0.037
	(0.054)
LT Reversal	0.057
	(0.070)
CIV	0.090
	(0.187)
PIV	-0.034
	(0.190)
IVS$_{atm}$	0.306***
	(0.096)
IVS$_{otm}$	-0.060
	(0.113)
Skew	0.198*
	(0.105)
VOV	0.004
	(0.068)
Δ CIV	-0.027
	(0.095)
Δ PIV	0.063
	(0.087)
CNN1	0.688***
	(0.055)
Observations	227
R^2	0.755
Adjusted R^2	0.738
Residual Std. Error	0.018
F Statistic	43.409***
Note:	*p<0.1; **p<0.05; ***p<0.01

2.12. Appendix - Additional Analysis for Different Size Groups of Stocks

2.13. Appendix - Proofs

Proof of Lemma 2.6.1. For any x_0, define

$$g(y) \;=\; f((I-UU')x_0+Uy)\,, \tag{2.23}$$

and note that g is also real analytic. Then, defining $\tilde{x}=(I-UU')x_0+UU'x$

$$g(U'x)-f(x) \;=\; f(\tilde{x}) \;-\; f(x) \;=\; \int_0^1 \nabla f(t(\tilde{x}-x)+x)'(\tilde{x}-x)dt. \tag{2.24}$$

We have

$$(I-UU')M_*(I-UU') \;=\; (I-UU')UDU'(I-UU') \;=\; 0 \tag{2.25}$$

since $(I-UU')U=0$. Thus, for any vector z, the function $\nabla f(x)'(I-UU')z$ is zero for Lebesgue almost every x because $E[(\nabla f(x)'(I-UU')z)^2]=z'M_*z=0$ and, hence, the real analytic function

$$G(x_1,x_2) \;=\; \nabla f(x_1)'(I-UU')x_2 \tag{2.26}$$

is zero for Lebesgue almost every (x_1,x_2). Therefore, for any fixed t, the real analytic function

$$\hat{G}(t,x_0,x) \;=\; \nabla f(t(\tilde{x}-x)+x)'(\tilde{x}-x) \;=\; \nabla f(t(I-UU')(x-x_0)+x)'(I-UU')(x-x_0) \tag{2.27}$$

is zero for Lebesgue almost every (x_0,x). Hence,

$$g(U'x)-f(x) \;=\; 0 \tag{2.28}$$

for Lebesgue almost every (x_0,x). Thus, for Lebesgue almost every x_0, we have that $g(U'x)=f(x)$. To prove the last statement, note that

$$E[\nabla f(X)\nabla f(X)'] \;=\; E[M'\nabla g(MX)\nabla g(MX)'M] \;=\; M'E[\nabla g(MX)\nabla g(MX)]M. \tag{2.29}$$

Since the components of $\nabla g(MX)$ are linearly independent, we have that the term

$$E[\nabla g(MX)\nabla g(MX)']\in\mathbb{R}^{r\times r}$$

is strictly positive definite, and hence, $\mathrm{rank}(M'E[\nabla g(MX)\nabla g(MX)]M)=r$. Indeed,

$$a'E[\nabla g(MX)\nabla g(MX)']a=E[(\nabla g(MX)'a)^2]>0 \tag{2.30}$$

because $\nabla g(MX)'a$ is not identically zero (linear independence) and hence is almost surely non-zero (by real analyticity). Then, for any y in the image of M (which has dimension r), we have that $y'E[\nabla g(MX)\nabla g(MX)]y>0$, this concludes the proof. □

Proof of Lemma 2.6.2. Let $U_p\in\mathbb{R}^{d\times p}$ be the matrix with first p eigenvectors and $U_{-p}\in\mathbb{R}^{d\times(d-p)}$ the matrix with the last $d-p$ eigenvectors. Then, defining $y=U_p'x\in\mathbb{R}^p$ and $z=U_{-p}'x\in\mathbb{R}^{d-p}$, we get $x=U_py+U_{-p}z$. Then, the conditional multivariate Gaussian $z|y\sim N(\mu(y),\hat{\Sigma}_z)$, where $\hat{\Sigma}_z\in\mathbb{R}^{d\times d}$ satisfies

$$\hat{\Sigma}_z \;=\; \Sigma_z-\Sigma_{zy}\Sigma_y^{-1}\Sigma_{yz} \;\le\; \lambda_1(\Sigma)I, \tag{2.31}$$

where we have defined the blocks

$$\Sigma_z \;=\; U_{-p}'\Sigma U_{-p}\in\mathbb{R}^{(d-p)\times(d-p)},\ \Sigma_{zy}=U_{-p}'\Sigma U_p\in\mathbb{R}^{(d-p)\times d},\ \Sigma_y=U_p'\Sigma U_p\in\mathbb{R}^{p\times p}. \tag{2.32}$$

Let $f_p(y) \;=\; E[f(x)|U_p'x=y]$. Let us fix y and define $F(z)=f(U_{-p}z+U_py)-f_p(y):\mathbb{R}^{d-p}\to\mathbb{R}$ (with a fixed y). Then, its gradient $\nabla_zF(z)\in\mathbb{R}^{d-p}$ satisfies (by the chain rule)

$$\nabla_zF(z) \;=\; \nabla_z(f(U_{-p}z+U_py)-f_p(y) \;=\; U_{-p}'\nabla_xf(x) \tag{2.33}$$

Table 2.26: **All-stocks, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **10 bps** and a short-sale monthly cost of **10 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.010***	0.013***	0.011***	0.010***	0.007***
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	-0.047	0.016	-0.047	0.099*	0.065
	(0.061)	(0.055)	(0.054)	(0.058)	(0.050)
SMB	-0.179*	-0.035	0.024	-0.024	-0.025
	(0.100)	(0.090)	(0.089)	(0.095)	(0.082)
HML	-0.085	-0.080	-0.130*	-0.105	-0.141**
	(0.085)	(0.077)	(0.075)	(0.081)	(0.070)
2-12 Momentum	-0.098**	-0.081*	-0.107***	0.064	0.107***
	(0.046)	(0.042)	(0.041)	(0.044)	(0.038)
ST Reversal	-0.074	-0.135**	-0.173***	-0.039	0.017
	(0.067)	(0.060)	(0.059)	(0.063)	(0.055)
LT Reversal	0.156*	0.173**	0.261***	0.239***	0.227***
	(0.086)	(0.078)	(0.077)	(0.082)	(0.071)
CIV	0.525**	0.120	0.048	0.463**	0.480**
	(0.231)	(0.209)	(0.205)	(0.220)	(0.191)
PIV	-0.314	-0.029	-0.011	-0.266	-0.179
	(0.236)	(0.213)	(0.209)	(0.224)	(0.195)
IVS$_{atm}$	-0.357***	0.089	0.248**	0.018	-0.010
	(0.118)	(0.106)	(0.105)	(0.112)	(0.097)
IVS$_{otm}$	0.686***	0.190	0.176	0.432***	0.415***
	(0.133)	(0.120)	(0.118)	(0.127)	(0.110)
Skew	0.594***	0.237**	0.129	0.542***	0.548***
	(0.125)	(0.113)	(0.111)	(0.119)	(0.103)
VOV	0.159*	0.139*	0.144*	0.115	0.153**
	(0.084)	(0.076)	(0.074)	(0.080)	(0.069)
Δ CIV	0.050	0.279**	0.176*	0.067	-0.069
	(0.119)	(0.108)	(0.106)	(0.113)	(0.098)
Δ PIV	0.130	0.162*	0.081	0.160	0.135
	(0.108)	(0.098)	(0.096)	(0.103)	(0.089)
Observations	227	227	227	227	227
R^2	0.657	0.366	0.345	0.620	0.735
Adjusted R^2	0.634	0.324	0.302	0.595	0.718
Residual Std. Error	0.022	0.020	0.020	0.021	0.019
F Statistic	28.965***	8.752***	7.972***	24.756***	42.095***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.27: **Mega-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **10 bps** and a short-sale monthly cost of **10 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.000	-0.002	-0.004*	0.001	0.001
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.065	0.108	0.064	0.149*	0.172**
	(0.073)	(0.073)	(0.071)	(0.076)	(0.072)
SMB	0.069	0.119	0.017	0.022	0.056
	(0.119)	(0.118)	(0.115)	(0.124)	(0.118)
HML	-0.063	-0.007	0.095	-0.016	-0.166*
	(0.101)	(0.101)	(0.098)	(0.106)	(0.100)
2-12 Momentum	0.034	-0.028	-0.015	0.211***	0.190***
	(0.055)	(0.055)	(0.053)	(0.058)	(0.054)
ST Reversal	-0.237***	-0.153*	-0.202***	-0.129	-0.000
	(0.080)	(0.079)	(0.077)	(0.083)	(0.079)
LT Reversal	0.161	0.194*	0.121	0.187*	0.218**
	(0.103)	(0.102)	(0.099)	(0.108)	(0.102)
CIV	-0.444	-0.330	-0.448*	-0.240	-0.050
	(0.276)	(0.274)	(0.267)	(0.289)	(0.273)
PIV	0.206	0.240	0.377	0.206	0.067
	(0.282)	(0.279)	(0.272)	(0.294)	(0.279)
IVS$_{atm}$	-0.327**	0.026	0.192	-0.214	-0.265*
	(0.141)	(0.140)	(0.136)	(0.147)	(0.139)
IVS$_{otm}$	0.234	-0.160	0.051	0.041	0.217
	(0.159)	(0.158)	(0.154)	(0.167)	(0.158)
Skew	0.664***	0.305**	0.144	0.768***	0.589***
	(0.149)	(0.148)	(0.144)	(0.156)	(0.148)
VOV	0.125	0.162	0.206**	0.390***	0.548***
	(0.100)	(0.099)	(0.097)	(0.105)	(0.099)
Δ CIV	0.280*	0.245*	0.138	0.133	-0.076
	(0.143)	(0.141)	(0.138)	(0.149)	(0.141)
Δ PIV	0.308**	0.104	-0.082	0.233*	0.170
	(0.129)	(0.128)	(0.125)	(0.135)	(0.128)
Observations	227	227	227	227	227
R^2	0.522	0.099	0.097	0.385	0.442
Adjusted R^2	0.491	0.039	0.037	0.345	0.405
Residual Std. Error	0.027	0.027	0.026	0.028	0.027
F Statistic	16.561***	1.661*	1.628*	9.485***	11.992***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.28: **Large-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **10 bps** and a short-sale monthly cost of **10 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	-0.001	-0.002	-0.002	-0.002	-0.002
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.152**	0.086	0.017	0.202***	0.215***
	(0.063)	(0.058)	(0.061)	(0.064)	(0.062)
SMB	0.117	-0.084	0.008	0.208**	0.324***
	(0.103)	(0.094)	(0.099)	(0.105)	(0.101)
HML	-0.027	-0.119	-0.072	-0.150*	-0.239***
	(0.088)	(0.080)	(0.084)	(0.089)	(0.086)
2-12 Momentum	-0.025	-0.048	-0.062	0.148***	0.182***
	(0.048)	(0.043)	(0.046)	(0.049)	(0.047)
ST Reversal	-0.029	-0.088	-0.157**	0.123*	0.184***
	(0.069)	(0.063)	(0.066)	(0.070)	(0.067)
LT Reversal	0.032	0.197**	0.157*	0.191**	0.192**
	(0.089)	(0.081)	(0.086)	(0.091)	(0.087)
CIV	-0.098	-0.179	-0.375	0.038	-0.028
	(0.239)	(0.218)	(0.230)	(0.243)	(0.234)
PIV	-0.080	0.179	0.368	-0.105	0.041
	(0.244)	(0.222)	(0.235)	(0.248)	(0.238)
IVS$_{atm}$	-0.449***	0.220**	0.362***	-0.007	-0.081
	(0.122)	(0.111)	(0.117)	(0.124)	(0.119)
IVS$_{otm}$	0.682***	-0.100	-0.157	0.220	0.284**
	(0.138)	(0.126)	(0.133)	(0.140)	(0.135)
Skew	0.577***	0.428***	0.381***	0.636***	0.585***
	(0.129)	(0.118)	(0.124)	(0.131)	(0.126)
VOV	-0.061	0.195**	0.204**	0.119	0.259***
	(0.087)	(0.079)	(0.083)	(0.088)	(0.085)
Δ CIV	-0.055	0.113	0.123	-0.018	-0.164
	(0.124)	(0.113)	(0.119)	(0.126)	(0.121)
Δ PIV	0.157	-0.149	-0.200*	0.247**	0.253**
	(0.112)	(0.102)	(0.108)	(0.114)	(0.109)
Observations	227	227	227	227	227
R^2	0.533	0.199	0.192	0.452	0.556
Adjusted R^2	0.502	0.146	0.139	0.416	0.527
Residual Std. Error	0.023	0.021	0.022	0.024	0.023
F Statistic	17.267***	3.754***	3.600***	12.503***	18.951***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.29: **Small-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **10 bps** and a short-sale monthly cost of **10 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.002	0.005**	0.003	0.002	0.002
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	-0.003	0.130*	0.013	0.154**	0.038
	(0.077)	(0.079)	(0.071)	(0.072)	(0.068)
SMB	-0.334***	-0.017	0.072	-0.087	-0.034
	(0.125)	(0.128)	(0.116)	(0.118)	(0.110)
HML	0.183*	0.065	-0.008	0.112	0.010
	(0.106)	(0.109)	(0.099)	(0.100)	(0.094)
2-12 Momentum	0.020	0.010	-0.080	0.096*	0.174***
	(0.058)	(0.059)	(0.054)	(0.054)	(0.051)
ST Reversal	0.006	-0.048	-0.106	-0.027	-0.010
	(0.083)	(0.086)	(0.078)	(0.079)	(0.074)
LT Reversal	-0.014	-0.019	0.162	0.015	0.135
	(0.108)	(0.111)	(0.101)	(0.102)	(0.095)
CIV	0.371	0.325	0.155	-0.012	0.146
	(0.289)	(0.297)	(0.270)	(0.273)	(0.255)
PIV	-0.100	-0.274	-0.145	0.181	0.130
	(0.295)	(0.303)	(0.275)	(0.279)	(0.260)
IVS$_{atm}$	-0.216	0.074	0.422***	0.503***	0.182
	(0.147)	(0.152)	(0.138)	(0.139)	(0.130)
IVS$_{otm}$	0.518***	0.194	0.086	0.257	0.353**
	(0.167)	(0.171)	(0.156)	(0.158)	(0.147)
Skew	0.537***	0.187	0.128	0.499***	0.393***
	(0.156)	(0.161)	(0.146)	(0.148)	(0.138)
VOV	0.006	0.105	0.103	0.005	0.045
	(0.105)	(0.108)	(0.098)	(0.099)	(0.092)
Δ CIV	-0.042	-0.001	-0.153	-0.057	-0.062
	(0.149)	(0.153)	(0.139)	(0.141)	(0.132)
Δ PIV	0.228*	0.116	-0.067	0.049	0.119
	(0.135)	(0.139)	(0.126)	(0.128)	(0.119)
Observations	227	227	227	227	227
R^2	0.454	0.131	0.185	0.416	0.514
Adjusted R^2	0.418	0.073	0.131	0.377	0.482
Residual Std. Error	0.028	0.029	0.026	0.027	0.025
F Statistic	12.580***	2.280***	3.443***	10.783***	16.034***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.30: **Micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.036***	0.046***	0.042***	0.036***	0.013***
	(0.005)	(0.006)	(0.006)	(0.005)	(0.004)
$r_M - r_f$	-0.517***	-0.401*	-0.254	-0.160	0.245*
	(0.185)	(0.203)	(0.203)	(0.160)	(0.136)
SMB	-0.246	0.049	0.257	-0.325	-0.017
	(0.301)	(0.331)	(0.331)	(0.261)	(0.221)
HML	-0.427*	-0.310	-0.410	-0.079	-0.007
	(0.256)	(0.282)	(0.282)	(0.222)	(0.188)
2-12 Momentum	-0.259*	-0.366**	-0.359**	-0.057	-0.137
	(0.139)	(0.153)	(0.153)	(0.121)	(0.102)
ST Reversal	-0.242	-0.625***	-0.664***	-0.027	0.132
	(0.201)	(0.221)	(0.221)	(0.174)	(0.148)
LT Reversal	0.412	0.402	0.544*	0.063	-0.105
	(0.260)	(0.286)	(0.286)	(0.225)	(0.191)
CIV	1.858***	0.732	0.524	1.615***	1.269**
	(0.697)	(0.769)	(0.767)	(0.605)	(0.513)
PIV	-1.457**	-0.663	-0.713	-1.189*	-0.984*
	(0.711)	(0.783)	(0.782)	(0.617)	(0.523)
IVS$_{atm}$	-0.321	0.525	0.460	-0.020	0.227
	(0.356)	(0.392)	(0.391)	(0.308)	(0.261)
IVS$_{otm}$	0.883**	0.415	0.748*	0.563	0.302
	(0.402)	(0.443)	(0.443)	(0.349)	(0.296)
Skew	0.341	-0.072	-0.300	0.467	0.945***
	(0.377)	(0.415)	(0.414)	(0.327)	(0.277)
VOV	0.289	1.020***	0.819***	0.503**	0.152
	(0.253)	(0.278)	(0.278)	(0.219)	(0.186)
Δ CIV	0.267	0.691*	0.749*	-0.138	-0.481*
	(0.360)	(0.397)	(0.396)	(0.312)	(0.265)
Δ PIV	0.155	0.367	0.253	0.068	-0.367
	(0.326)	(0.359)	(0.359)	(0.283)	(0.240)
Observations	227	227	227	227	227
R^2	0.356	0.239	0.215	0.305	0.347
Adjusted R^2	0.314	0.188	0.163	0.259	0.304
Residual Std. Error	0.068	0.075	0.075	0.059	0.050
F Statistic	8.378***	4.748***	4.139***	6.635***	8.057***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.31: **Not-micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **10 bps** and a short-sale monthly cost of **10 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.002	0.004**	0.002	0.001	0.001
	(0.002)	(0.002)	(0.001)	(0.002)	(0.001)
$r_M - r_f$	0.066	0.049	0.028	0.151***	0.123**
	(0.057)	(0.051)	(0.048)	(0.053)	(0.049)
SMB	-0.069	0.022	0.036	0.071	0.151*
	(0.093)	(0.082)	(0.078)	(0.087)	(0.080)
HML	0.018	0.031	-0.010	-0.026	-0.144**
	(0.079)	(0.070)	(0.066)	(0.074)	(0.068)
2-12 Momentum	-0.040	-0.000	-0.025	0.123***	0.145***
	(0.043)	(0.038)	(0.036)	(0.040)	(0.037)
ST Reversal	-0.064	-0.002	-0.085	0.015	0.056
	(0.062)	(0.055)	(0.052)	(0.058)	(0.053)
LT Reversal	0.066	0.073	0.149**	0.139*	0.176**
	(0.080)	(0.071)	(0.068)	(0.075)	(0.069)
CIV	0.160	-0.099	-0.132	-0.020	0.014
	(0.215)	(0.191)	(0.181)	(0.201)	(0.186)
PIV	-0.090	0.150	0.151	0.105	0.185
	(0.219)	(0.195)	(0.185)	(0.205)	(0.189)
IVS$_{atm}$	-0.313***	0.096	0.284***	0.211**	0.056
	(0.110)	(0.097)	(0.092)	(0.102)	(0.095)
IVS$_{otm}$	0.485***	0.029	0.009	0.291**	0.299***
	(0.124)	(0.110)	(0.104)	(0.116)	(0.107)
Skew	0.587***	0.329***	0.260***	0.486***	0.519***
	(0.116)	(0.103)	(0.098)	(0.108)	(0.100)
VOV	0.014	0.097	0.110*	0.045	0.146**
	(0.078)	(0.069)	(0.066)	(0.073)	(0.067)
Δ CIV	-0.059	0.082	-0.041	0.031	-0.070
	(0.111)	(0.099)	(0.094)	(0.104)	(0.096)
Δ PIV	0.126	0.058	-0.082	0.165*	0.209**
	(0.100)	(0.089)	(0.085)	(0.094)	(0.087)
Observations	227	227	227	227	227
R^2	0.511	0.202	0.216	0.528	0.679
Adjusted R^2	0.479	0.150	0.164	0.497	0.658
Residual Std. Error	0.021	0.019	0.018	0.020	0.018
F Statistic	15.812***	3.838***	4.170***	16.950***	32.048***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.32: **All-stocks, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.012***	0.013***	0.011***	0.012***	0.009***
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	-0.046	0.016	-0.047	0.098*	0.065
	(0.061)	(0.055)	(0.054)	(0.058)	(0.050)
SMB	-0.174*	-0.031	0.027	-0.020	-0.021
	(0.099)	(0.090)	(0.088)	(0.094)	(0.082)
HML	-0.086	-0.079	-0.127*	-0.105	-0.141**
	(0.084)	(0.076)	(0.075)	(0.080)	(0.070)
2-12 Momentum	-0.098**	-0.082**	-0.107***	0.063	0.106***
	(0.046)	(0.041)	(0.041)	(0.044)	(0.038)
ST Reversal	-0.074	-0.133**	-0.171***	-0.037	0.019
	(0.066)	(0.060)	(0.059)	(0.063)	(0.055)
LT Reversal	0.158*	0.173**	0.260***	0.241***	0.228***
	(0.086)	(0.077)	(0.076)	(0.081)	(0.071)
CIV	0.531**	0.120	0.043	0.471**	0.485**
	(0.231)	(0.208)	(0.205)	(0.219)	(0.190)
PIV	-0.324	-0.033	-0.008	-0.278	-0.187
	(0.235)	(0.212)	(0.208)	(0.223)	(0.194)
IVS$_{atm}$	-0.359***	0.090	0.250**	0.018	-0.011
	(0.118)	(0.106)	(0.105)	(0.112)	(0.097)
IVS$_{otm}$	0.693***	0.199*	0.182	0.441***	0.421***
	(0.133)	(0.120)	(0.118)	(0.127)	(0.110)
Skew	0.588***	0.230**	0.125	0.533***	0.542***
	(0.125)	(0.112)	(0.110)	(0.118)	(0.103)
VOV	0.159*	0.140*	0.144*	0.116	0.152**
	(0.084)	(0.075)	(0.074)	(0.079)	(0.069)
Δ CIV	0.045	0.271**	0.171	0.054	-0.078
	(0.120)	(0.108)	(0.106)	(0.114)	(0.099)
Δ PIV	0.124	0.154	0.076	0.147	0.127
	(0.108)	(0.097)	(0.096)	(0.103)	(0.089)
Observations	227	227	227	227	227
R^2	0.656	0.366	0.346	0.621	0.735
Adjusted R^2	0.634	0.324	0.303	0.596	0.718
Residual Std. Error	0.022	0.020	0.020	0.021	0.018
F Statistic	28.926***	8.747***	8.003***	24.782***	42.107***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.33: **Mega-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.001	-0.003	-0.005*	0.003	0.002
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
$r_M - r_f$	0.065	0.110	0.066	0.149*	0.172**
	(0.073)	(0.072)	(0.070)	(0.076)	(0.072)
SMB	0.070	0.120	0.018	0.023	0.056
	(0.119)	(0.118)	(0.115)	(0.124)	(0.118)
HML	-0.064	-0.004	0.096	-0.017	-0.166*
	(0.101)	(0.100)	(0.097)	(0.106)	(0.100)
2-12 Momentum	0.034	-0.028	-0.015	0.210***	0.189***
	(0.055)	(0.054)	(0.053)	(0.057)	(0.054)
ST Reversal	-0.237***	-0.152*	-0.202***	-0.129	-0.001
	(0.079)	(0.079)	(0.077)	(0.083)	(0.078)
LT Reversal	0.163	0.191*	0.120	0.189*	0.219**
	(0.103)	(0.102)	(0.099)	(0.107)	(0.102)
CIV	-0.443	-0.330	-0.450*	-0.232	-0.043
	(0.276)	(0.274)	(0.267)	(0.289)	(0.273)
PIV	0.203	0.238	0.378	0.196	0.058
	(0.282)	(0.279)	(0.272)	(0.294)	(0.278)
IVS$_{atm}$	-0.327**	0.022	0.189	-0.219	-0.270*
	(0.141)	(0.140)	(0.136)	(0.148)	(0.140)
IVS$_{otm}$	0.238	-0.155	0.054	0.046	0.222
	(0.160)	(0.158)	(0.154)	(0.167)	(0.158)
Skew	0.660***	0.300**	0.142	0.762***	0.584***
	(0.149)	(0.148)	(0.144)	(0.156)	(0.148)
VOV	0.122	0.160	0.203**	0.387***	0.546***
	(0.100)	(0.099)	(0.097)	(0.105)	(0.099)
Δ CIV	0.280*	0.243*	0.139	0.132	-0.079
	(0.144)	(0.142)	(0.139)	(0.150)	(0.142)
Δ PIV	0.305**	0.101	-0.082	0.229*	0.167
	(0.129)	(0.128)	(0.125)	(0.135)	(0.128)
Observations	227	227	227	227	227
R^2	0.522	0.098	0.097	0.384	0.441
Adjusted R^2	0.490	0.039	0.037	0.343	0.404
Residual Std. Error	0.027	0.027	0.026	0.028	0.026
F Statistic	16.527***	1.648*	1.620*	9.425***	11.930***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.34: **Large-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	-0.001	-0.003	-0.003	-0.000	-0.000
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.151**	0.086	0.017	0.201***	0.214***
	(0.063)	(0.057)	(0.061)	(0.064)	(0.062)
SMB	0.118	-0.082	0.008	0.209**	0.323***
	(0.103)	(0.094)	(0.099)	(0.105)	(0.101)
HML	-0.028	-0.119	-0.072	-0.151*	-0.240***
	(0.087)	(0.080)	(0.084)	(0.089)	(0.085)
2-12 Momentum	-0.025	-0.048	-0.061	0.148***	0.182***
	(0.048)	(0.043)	(0.046)	(0.048)	(0.046)
ST Reversal	-0.029	-0.087	-0.155**	0.123*	0.184***
	(0.069)	(0.063)	(0.066)	(0.070)	(0.067)
LT Reversal	0.033	0.196**	0.157*	0.193**	0.194**
	(0.089)	(0.081)	(0.085)	(0.090)	(0.087)
CIV	-0.093	-0.174	-0.371	0.046	-0.023
	(0.239)	(0.218)	(0.230)	(0.243)	(0.234)
PIV	-0.086	0.173	0.363	-0.115	0.035
	(0.244)	(0.222)	(0.234)	(0.248)	(0.238)
IVS$_{atm}$	-0.450***	0.216*	0.359***	-0.009	-0.082
	(0.122)	(0.111)	(0.118)	(0.124)	(0.120)
IVS$_{otm}$	0.685***	-0.099	-0.157	0.223	0.286**
	(0.138)	(0.126)	(0.133)	(0.141)	(0.135)
Skew	0.573***	0.427***	0.380***	0.631***	0.580***
	(0.129)	(0.118)	(0.124)	(0.131)	(0.126)
VOV	-0.062	0.194**	0.202**	0.118	0.259***
	(0.087)	(0.079)	(0.083)	(0.088)	(0.085)
Δ CIV	-0.059	0.111	0.121	-0.024	-0.170
	(0.124)	(0.113)	(0.119)	(0.126)	(0.121)
Δ PIV	0.153	-0.151	-0.202*	0.240**	0.248**
	(0.112)	(0.102)	(0.108)	(0.114)	(0.109)
Observations	227	227	227	227	227
R^2	0.532	0.197	0.190	0.451	0.555
Adjusted R^2	0.501	0.144	0.137	0.414	0.525
Residual Std. Error	0.023	0.021	0.022	0.024	0.023
F Statistic	17.195***	3.716***	3.562***	12.419***	18.862***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.35: **Small-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.004	0.004	0.002	0.004	0.004*
	(0.003)	(0.003)	(0.003)	(0.003)	(0.002)
$r_M - r_f$	-0.002	0.130*	0.013	0.153**	0.037
	(0.076)	(0.078)	(0.071)	(0.072)	(0.067)
SMB	-0.331***	-0.014	0.072	-0.085	-0.033
	(0.124)	(0.128)	(0.116)	(0.118)	(0.110)
HML	0.180*	0.066	-0.008	0.109	0.008
	(0.106)	(0.109)	(0.099)	(0.100)	(0.093)
2-12 Momentum	0.019	0.010	-0.079	0.096*	0.174***
	(0.057)	(0.059)	(0.054)	(0.054)	(0.051)
ST Reversal	0.005	-0.048	-0.105	-0.025	-0.009
	(0.083)	(0.085)	(0.077)	(0.079)	(0.073)
LT Reversal	-0.012	-0.019	0.161	0.018	0.136
	(0.107)	(0.110)	(0.100)	(0.102)	(0.095)
CIV	0.380	0.325	0.155	-0.001	0.153
	(0.289)	(0.297)	(0.270)	(0.274)	(0.255)
PIV	-0.111	-0.276	-0.145	0.167	0.121
	(0.294)	(0.302)	(0.275)	(0.279)	(0.260)
IVS$_{atm}$	-0.221	0.073	0.422***	0.499***	0.179
	(0.148)	(0.152)	(0.138)	(0.140)	(0.131)
IVS$_{otm}$	0.520***	0.197	0.085	0.260	0.355**
	(0.167)	(0.171)	(0.156)	(0.158)	(0.148)
Skew	0.534***	0.182	0.126	0.492***	0.388***
	(0.156)	(0.160)	(0.146)	(0.148)	(0.138)
VOV	0.006	0.105	0.102	0.003	0.043
	(0.105)	(0.108)	(0.098)	(0.099)	(0.093)
Δ CIV	-0.041	-0.004	-0.159	-0.063	-0.067
	(0.150)	(0.154)	(0.140)	(0.142)	(0.133)
Δ PIV	0.228*	0.114	-0.070	0.043	0.114
	(0.135)	(0.139)	(0.126)	(0.128)	(0.120)
Observations	227	227	227	227	227
R^2	0.453	0.130	0.184	0.413	0.512
Adjusted R^2	0.417	0.072	0.130	0.374	0.480
Residual Std. Error	0.028	0.029	0.026	0.027	0.025
F Statistic	12.540***	2.257***	3.410***	10.656***	15.890***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.36: **Micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **40 bps** and a short-sale monthly cost of **40 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.037^{***}	0.052^{***}	0.047^{***}	0.036^{***}	0.010^{**}
	(0.007)	(0.007)	(0.007)	(0.006)	(0.005)
$r_M - r_f$	-0.514^{***}	-0.399^{**}	-0.253	-0.160	0.244^{*}
	(0.184)	(0.202)	(0.202)	(0.159)	(0.135)
SMB	-0.241	0.053	0.259	-0.321	-0.017
	(0.299)	(0.330)	(0.329)	(0.260)	(0.220)
HML	-0.427^{*}	-0.308	-0.407	-0.081	-0.008
	(0.254)	(0.280)	(0.280)	(0.221)	(0.187)
2-12 Momentum	-0.256^{*}	-0.362^{**}	-0.355^{**}	-0.055	-0.135
	(0.138)	(0.152)	(0.152)	(0.120)	(0.102)
ST Reversal	-0.237	-0.617^{***}	-0.656^{***}	-0.023	0.132
	(0.200)	(0.220)	(0.220)	(0.173)	(0.147)
LT Reversal	0.414	0.403	0.543^{*}	0.067	-0.102
	(0.259)	(0.285)	(0.284)	(0.224)	(0.190)
CIV	1.873^{***}	0.733	0.527	1.629^{***}	1.273^{**}
	(0.696)	(0.766)	(0.765)	(0.603)	(0.511)
PIV	-1.480^{**}	-0.671	-0.721	-1.209^{*}	-0.990^{*}
	(0.709)	(0.781)	(0.779)	(0.615)	(0.521)
IVS$_{atm}$	-0.321	0.533	0.467	-0.021	0.222
	(0.356)	(0.392)	(0.391)	(0.309)	(0.261)
IVS$_{otm}$	0.897^{**}	0.435	0.765^{*}	0.575	0.305
	(0.402)	(0.442)	(0.442)	(0.348)	(0.295)
Skew	0.318	-0.096	-0.322	0.448	0.936^{***}
	(0.376)	(0.414)	(0.413)	(0.326)	(0.276)
VOV	0.285	1.017^{***}	0.815^{***}	0.499^{**}	0.148
	(0.252)	(0.278)	(0.277)	(0.219)	(0.185)
Δ CIV	0.239	0.665^{*}	0.721^{*}	-0.165	-0.489^{*}
	(0.362)	(0.398)	(0.397)	(0.314)	(0.266)
Δ PIV	0.129	0.339	0.226	0.043	-0.374
	(0.326)	(0.359)	(0.358)	(0.282)	(0.239)
Observations	227	227	227	227	227
R^2	0.354	0.237	0.213	0.302	0.346
Adjusted R^2	0.311	0.187	0.161	0.256	0.303
Residual Std. Error	0.067	0.074	0.074	0.058	0.050
F Statistic	8.295^{***}	4.708^{***}	4.101^{***}	6.566^{***}	8.020^{***}

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.37: **Not-micro-cap segment, long-short portfolio.** Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses. We apply a linear fee of **20 bps** and a short-sale monthly cost of **20 bps** to the returns of CNN1, CNN4, CNN5, NN1, Ridge Regression and all option-based portfolios from [Neu+22].

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.003	0.003*	0.000	0.003*	0.003*
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.067	0.049	0.028	0.151***	0.122**
	(0.057)	(0.050)	(0.048)	(0.053)	(0.049)
SMB	-0.068	0.024	0.037	0.072	0.152*
	(0.092)	(0.082)	(0.078)	(0.086)	(0.080)
HML	0.016	0.031	-0.010	-0.028	-0.145**
	(0.079)	(0.070)	(0.066)	(0.073)	(0.068)
2-12 Momentum	-0.040	0.000	-0.024	0.123***	0.145***
	(0.043)	(0.038)	(0.036)	(0.040)	(0.037)
ST Reversal	-0.065	-0.002	-0.084	0.015	0.057
	(0.062)	(0.055)	(0.052)	(0.058)	(0.053)
LT Reversal	0.068	0.072	0.149**	0.142*	0.177**
	(0.080)	(0.071)	(0.067)	(0.075)	(0.069)
CIV	0.166	-0.095	-0.132	-0.010	0.024
	(0.215)	(0.191)	(0.181)	(0.201)	(0.186)
PIV	-0.098	0.145	0.151	0.092	0.173
	(0.219)	(0.194)	(0.184)	(0.205)	(0.189)
IVS$_{atm}$	-0.315***	0.093	0.283***	0.209**	0.052
	(0.110)	(0.098)	(0.093)	(0.103)	(0.095)
IVS$_{otm}$	0.487***	0.031	0.009	0.295**	0.302***
	(0.124)	(0.110)	(0.104)	(0.116)	(0.107)
Skew	0.585***	0.326***	0.259***	0.480***	0.513***
	(0.116)	(0.103)	(0.098)	(0.108)	(0.100)
VOV	0.013	0.096	0.109*	0.044	0.145**
	(0.078)	(0.069)	(0.066)	(0.073)	(0.067)
Δ CIV	-0.058	0.079	-0.043	0.025	-0.075
	(0.112)	(0.099)	(0.094)	(0.104)	(0.097)
Δ PIV	0.125	0.056	-0.084	0.158*	0.204**
	(0.101)	(0.089)	(0.085)	(0.094)	(0.087)
Observations	227	227	227	227	227
R^2	0.510	0.200	0.214	0.526	0.677
Adjusted R^2	0.478	0.147	0.162	0.494	0.656
Residual Std. Error	0.021	0.018	0.018	0.019	0.018
F Statistic	15.762***	3.786***	4.131***	16.788***	31.776***

Note: *p<0.1; **p<0.05; ***p<0.01

Table 2.38: **Capped-value-weighted long-short portfolio (weights are set in proportion to stock market capitalisation, with market caps winsorized at 80% NYSE percentile as described in** [**JKPrt**]). Monthly OLS regression of the CNN1, CNN4, CNN5, NN1, and Ridge Regression **long-short** portfolios on the factor model that includes the CIV, PIV, IVS$_{atm}$, IVS$_{otm}$, Skew, VOV, ΔCIV, and ΔPIV factor **long-short** portfolios from [Neu+22], along with the standard Fama-French factors. The intercept coefficient is reported in monthly return terms, with corresponding standard errors in parentheses.

	$CNN1_{ew}$	$CNN4_{ew}$	$CNN5_{ew}$	$NN1_{ew}$	$z = 0.1$
Intercept	0.000	0.003*	0.002	0.000	0.001
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$r_M - r_f$	0.079	0.081	0.053	0.112*	0.044
	(0.073)	(0.059)	(0.059)	(0.063)	(0.053)
SMB	-0.110	0.128	0.104	0.061	0.126
	(0.109)	(0.088)	(0.087)	(0.093)	(0.078)
HML	-0.048	-0.012	0.039	-0.003	-0.207***
	(0.101)	(0.082)	(0.081)	(0.087)	(0.073)
2-12 Momentum	-0.117**	-0.119***	-0.076*	0.092*	0.024
	(0.056)	(0.045)	(0.044)	(0.047)	(0.040)
ST Reversal	-0.040	0.044	-0.043	0.016	0.025
	(0.080)	(0.065)	(0.064)	(0.069)	(0.058)
LT Reversal	0.012	0.012	0.045	0.091	0.109
	(0.107)	(0.087)	(0.085)	(0.091)	(0.077)
CIV	0.144	0.039	-0.045	0.109	0.109
	(0.160)	(0.130)	(0.128)	(0.136)	(0.115)
PIV	-0.005	-0.022	-0.001	0.030	0.138
	(0.165)	(0.134)	(0.132)	(0.141)	(0.119)
IVS$_{atm}$	-0.141	0.165*	0.190*	0.412***	0.200**
	(0.122)	(0.099)	(0.098)	(0.104)	(0.088)
IVS$_{otm}$	0.771***	0.066	0.091	0.371***	0.410***
	(0.140)	(0.114)	(0.112)	(0.120)	(0.101)
Skew	0.818***	0.364***	0.234**	0.621***	0.596***
	(0.122)	(0.099)	(0.097)	(0.104)	(0.088)
VOV	-0.049	-0.036	0.101	0.010	0.160**
	(0.090)	(0.073)	(0.072)	(0.077)	(0.065)
Δ CIV	-0.096	0.221*	0.137	-0.107	0.027
	(0.147)	(0.120)	(0.118)	(0.126)	(0.106)
Δ PIV	0.019	0.149	0.055	0.056	0.168*
	(0.134)	(0.109)	(0.108)	(0.115)	(0.097)
Observations	227	227	227	227	227
R^2	0.597	0.280	0.174	0.589	0.728
Adjusted R^2	0.571	0.233	0.119	0.562	0.710
Residual Std. Error	0.027	0.022	0.022	0.023	0.020
F Statistic	22.476***	5.897***	3.183***	21.684***	40.452***

Note: *p<0.1; **p<0.05; ***p<0.01

for $x = U_{-p}z + U_p y$. Then, by the [BL76] inequality,

$$\begin{aligned}
&\mathrm{Var}[F(z)|y] \;=\; E[F(z)^2|y] \;\leq\; E[\nabla_z F(z)'\hat{\Sigma}_z \nabla_z F(z)|y] \\
&\leq\; \Lambda_1(\Sigma)E[\|\nabla_z F(z)\|^2|y] \;=\; \Lambda_1(\Sigma)E[\|\nabla_z F(z)\|^2|y] \\
&=\; \lambda_1(\Sigma)E[\|U'_{-p}\nabla_x f(x)\|^2|U'_p x = y] \\
&=\; \lambda_1(\Sigma)E[(U'_{-p}\nabla_x f(x))'U'_{-p}\nabla_x f(x)|U'_p x = y] \\
&=\; \lambda_1(\Sigma)E[\nabla_x f(x)'U_{-p}U'_{-p}\nabla_x f(x)|U'_p x = y] \\
&=\; \lambda_1(\Sigma)\,\mathsf{T}\,E[U_{-p}U'_{-p}\nabla_x f(x)\nabla_x f(x)'|U'_p x = y] \\
&=\; \lambda_1(\Sigma)\,\mathsf{T}\,(U_{-p}U'_{-p}E[\nabla_x f(x)\nabla_x f(x)'|U'_p x = y])
\end{aligned} \tag{2.34}$$

and, hence,

$$\begin{aligned}
&E[(f(x) - f_p(U'_p x))^2] \;=\; E[F(z)^2] \;=\; E[E[F(z)^2|y]] \\
&\leq\; \lambda_1(\Sigma)E[\lambda_1(\Sigma)\,\mathsf{T}\,(U_{-p}U'_{-p}E[\nabla_x f(x)\nabla_x f(x)'|U'_p x = y])] \\
&=\; \lambda_1(\Sigma)\,\mathsf{T}\,(U_{-p}U'_{-p}E[\nabla_x f(x)\nabla_x f(x)']) \\
&=\; \lambda_1(\Sigma)\,\mathsf{T}\,(U_{-p}U'_{-p}\bar{M}_*) \;=\; \lambda_1(\Sigma)\Lambda_{-p}(M_*).
\end{aligned} \tag{2.35}$$

$\square$

Part III

Kernel Methods

3. A Simple Algorithm For Scaling Up Kernel Methods

3.1. Introduction

Modern neural networks operate in the over-parametrized regime, which sometimes requires orders of magnitude more parameters than training data points. Effectively, they are *interpolators* (see, [Bel21]) and overfit the data in the training sample, with no consequences for the out-of-sample performance. This seemingly counterintuitive phenomenon is sometimes called "benign overfit" [Bar+20; TB20].

In the so-called lazy training regime [COB19], wide neural networks (many nodes in each layer) are effectively kernel regressions, and "early stopping" commonly used in neural network training is closely related to ridge regularization [AKT19]. See, [JGH18; Has+19; Du+18; Du+19a; AZLS19]. Recent research also emphasizes the "double descent," in which expected forecast error drops in the high-complexity regime. See, for example, [Zha+16; Bel+18; BRT19; Spi+19; BHX20].

These discoveries made many researchers argue that we need to gain a deeper understanding of kernel methods (and, hence, random feature regressions) and their link to deep learning. See, e.g., [BMM18]. Several recent papers have developed numerical algorithms for scaling kernel-type methods to large datasets and large numbers of random features. See, e.g., [Zan+21; MB17; Aro+19a; Sha+20]. In particular, [Aro+19b] show how NTK combined with the support vector machines (SVM) (see also [FD+14]) perform well on small data tasks relative to many competitors, including the highly over-parametrized ResNet-34. In particular, while modern deep neural networks do generalize on small datasets (see, e.g., [OWB18]), [Aro+19b] show that kernel-based methods achieve superior performance in such small data environments. Similarly, [Du+19b] find that the graph neural tangent kernel (GNTK) dominates graph neural networks on datasets with up to 5000 samples. [Sha+20] show that, while NTK is a powerful kernel, it is possible to build other classes of kernels (they call Neural Kernels) that are even more powerful and are often at par with extremely complex deep neural networks.

In this paper, we develop a novel form of kernel ridge regression that can be applied to any kernel and any way of generating random features. We use a doubly stochastic method similar to that in [Dai+14], with an important caveat: We generate (potentially large, defined by the RAM constraints) batches of random features and then use linear algebraic properties of covariance matrices to recursively update the eigenvalue decomposition of the feature covariance matrix, allowing us to perform the optimization in one shot across a large grid of ridge parameters.

The paper is organized as follows. Section 3.2 discusses related work. In Section 3.3, we provide a novel random feature regression mathematical formulation and algorithm. Then, Section 3.4 and Section 3.5 present numerical results and conclusions, respectively.

3.2. Related Work

Before the formal introduction of the NTK in [JGH18], numerous papers discussed the intriguing connections between infinitely wide neural networks and kernel methods. See, e.g., [Nea96]; [Wil97]; [LRB07]; [HJ15]; [Lee+18]; [Mat+18]; [Nov+18]; [GARA18]; [CS09]; [DFS16]; [Dan17]. As in the standard random feature approximation of the kernel ridge regression (see [RR07]), only the network's last layer is trained in the standard kernel ridge regression. A surprising discovery of [JGH18] is that (infinitely) wide neural networks in the lazy training regime converge to a kernel even though all network layers are trained. The corresponding kernel, the NTK, has a complex structure dependent on the neural network's architecture. See also [Lee+19], [Aro+19a] for more results about the link between NTK and the underlying neural network, and [Nov+19] for an efficient algorithm for implementing the NTK. In a recent paper, [Sha+20] introduce a new class of kernels and show that they perform remarkably well on even very large datasets, achieving a 90% accuracy on the CIFAR-10 dataset. While this performance is striking, it comes at a huge computational cost. [Sha+20] write:

"CIFAR-10/CIFAR-100 consist of $60,000$ $32 \times 32 \times 3$ *images and MNIST consists of* $70,000$ 28×28 *images. Even with this constraint, the largest compositional kernel matrices*

we study took approximately 1000 GPU hours to compute. Thus, we believe an imperative direction of future work is reducing the complexity of each kernel evaluation. Random feature methods or other compression schemes could play a significant role here.

In this paper, we offer one such highly scalable scheme based on random features. However, computing the random features underlying the Neural Kernels of [Sha+20] would require developing non-trivial numerical algorithms based on the recursive iteration of non-linear functions. We leave this as an important direction for future research.

As in standard kernel ridge regressions, we train our random feature regression on the full sample. This is a key computational limitation for large datasets. After all, one of the reasons for the success of modern deep learning is the possibility of training them using stochastic gradient descent on mini-batches of data. [MB17] shows how mini-batch training can be applied to kernel ridge regression. A key technical difficulty arises because kernel matrices (equivalently, covariance matrices of random features) have eigenvalues that decay very quickly. Yet, these low eigenvalues contain essential information and cannot be neglected. Our regression method can be easily modified to allow for mini-batches. Furthermore, it is known that mini-batch linear regression can even lead to performance gains in the high-complexity regime. As [LJB20] show, one can run regression on mini-batches and then treat the obtained predictions as an ensemble. [LJB20] prove that, under technical conditions, the average of these predictions attains a lower generalization error than the full-train-sample-based regression. We test this mini-batch ensemble approach using our method and show that, indeed, with moderately-sized mini-batches, the method's performance matches that of the full sample regression.

Moreover, there is an intriguing connection between mini-batch regressions and spectral dimensionality reduction. By construction, the feature covariance matrix with a mini-batch of size B has at most B non-zero eigenvalues. Thus, a mini-batch effectively performs a dimensionality reduction on the covariance matrix. Intuitively, we expect that the two methods (using a mini-batch of size B or using the full sample but only keeping B largest eigenvalues) should achieve comparable performance. We show that this is indeed the case for small sample sizes. However, the spectral method for larger-sized samples ($N \geq 10000$) is superior to the mini-batch method unless we use very large mini-batches. For example, on the full CIFAR-10 dataset, the spectral method outperforms the mini-batch approach by 3% (see Section 3.4 for details).

3.3. Random Features Ridge Regression and Classification

Suppose that we have a train sample $(X, y) = (x_i, y_i)_{i=1}^N,\ x_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}$, so that $X \in \mathbb{R}^{N\times d},\ y \in \mathbb{R}^{N\times 1}$. Following [RR07] we construct a large number of random features $f(x; \theta_p),\ p = 1, \ldots, P$, where f is a non-linear function and θ_p are sampled from some distribution, and P is a large number. We denote $S = f(X; \theta) \in \mathbb{R}^{N\times P}$ as the train sample realizations of random features. Following [RR07], we consider the random features ridge regression,

$$\beta(z) \ = \ (S^\top S/N + zI)^{-1} S^\top y/N\,, \tag{3.1}$$

as an approximation for kernel ridge regression when $P \to \infty$. For classification problems, it is common to use categorical cross-entropy as the objective. However, as [Bel21] explains, minimizing the mean-squared error with one-hot encoding often achieves superior generalization performance. Here, we follow this approach. Given the K labels, $k = 1, \ldots, K$, we build the one-hot encoding matrix $Q = (q_{i,k})$ where $q_{i,k} = \mathbf{1}_{y_i = k}$. Then, we get

$$\beta(z) \ = \ (S^\top S/N + zI)^{-1} S^\top Q/N \ \in \ \mathbb{R}^{P\times K}\,. \tag{3.2}$$

Then, for each test feature vector $\mathbf{s} = f(\mathbf{x}; \theta) \in \mathbb{R}^P$, we get a vector $\beta(z)^\top \mathbf{s} \in \mathbb{R}^K$. Next, define the actual classifier as

$$k(\mathbf{x}; z) \ = \ \arg\max\{\beta(z)^\top \mathbf{s}\} \ \in \ \{1, \cdots, K\}\,. \tag{3.3}$$

3.3.1. Dealing with High-Dimensional Features

A key computational (hardware) limitation of kernel methods comes from the fact that, when P is large, computing the matrix $S^\top S \in \mathbb{R}^{P\times P}$ becomes prohibitively expensive, in particular, because S cannot even be stored in RAM. We start with a simple observation that the following identity implies that storing all these features is not necessary:[1]

$$(S^\top S/N + zI)^{-1} S^\top \;=\; S^\top (SS^\top/N + zI)^{-1}\,, \tag{3.4}$$

and therefore we can compute $\beta(z)$ as

$$\beta(z) \;=\; S^\top (SS^\top/N + zI)^{-1} y/N\,. \tag{3.5}$$

Suppose now we split S into multiple blocks, $S_1, \dots, S_K$, where $S_k \in \mathbb{R}^{N\times P_1}$ for all $k = 1, \dots, K$, for some small P_1, with $KP_1 = P$. Then,

$$\Psi \;=\; SS^\top \;=\; \sum_{k=1}^{K} S_k S_k^\top \tag{3.6}$$

can be computed by generating the blocks S_k, one at a time, and recursively adding $S_k S_k^\top$ up. Once Ψ has been computed, one can calculate its eigenvalue decomposition, $\Psi = VDV^\top$, and then evaluate $Q(z) = (\Psi/N + zI)^{-1} y/N \;=\; V(D+zI)^{-1}V^\top y/N \in \mathbb{R}^N$ in one go for a grid of z. Then, using the same seeds, we can again generate the random features S_k and compute $\beta_k(z) = S_k^\top Q(z) \in \mathbb{R}^{P_1}$. Then, $\beta(z) = (\beta_k(z))_{k=1}^K \in \mathbb{R}^P$. The logic described above is formalized in Algorithm 3.

Algorithm 3 FABReg

```
Require: $P_1$, $P$, $X \in \mathbb{R}^{N\times d}$, $y \in \mathbb{R}^N$, $z$, $voc\_curve$
  $blocks \leftarrow P//P_1$
  $k \leftarrow 0$
  $\Psi \leftarrow 0_{N\times N}$
  while $k < blocks$ do
    Generate $S_k \in \mathbb{R}^{N\times P_1}$ Use $k$ as seed
    $\Psi \leftarrow \Psi + S_k S_k \top$
    if $k$ in $voc\_curve$ then
      $DV \leftarrow eigen(\frac{\Psi}{N})$
      $Q_k(z) \leftarrow V(D+zI)^{-1}V^\top \frac{y}{N}$ {Store $Q_k(z)$}
    end if
    $k = k + 1$
  end while
  $DV \leftarrow eigen(\frac{\Psi}{N})$
  $Q(z) \leftarrow V(D+zI)^{-1}V^\top \frac{y}{N}$
  $k \leftarrow 0$
  while $k < blocks$ do
    (re-)Generate $S_k \in \mathbb{R}^{N\times P_1}$ {Use $k$ as seed}
    $\beta_k(z) \leftarrow S_k^\top Q(z)$
    $\hat{y}$ += $S_k \beta_k$
  end while
```

[1]This identity follows directly from $(S^\top S/N + zI)S^\top = S^\top(SS^\top/N + zI)$.

3.3.2. Dealing with Massive Datasets

The above algorithm relies crucially on the assumption that N is small. Suppose now that the sample size N is so large that storing and eigen-decomposing the matrix $SS^\top \in \mathbb{R}^{N\times N}$ becomes prohibitively expensive. In this case, we proceed as follows.

Define for all $k = 1, \dots, K$

$$\Psi_k \;=\; \sum_{\kappa=1}^{k} S_k S_k^\top \in \mathbb{R}^{N\times N},\;\; \Psi_0 = 0_{N\times N}\,, \tag{3.7}$$

and let $\lambda_1(A) \geq \cdots \geq \lambda_N(A)$ be the eigenvalues of a symmetric matrix $A \in \mathbb{R}^{N\times N}$. Our goal is to design an approximation to $(\Psi_K + zI)^{-1}$, based on a simple observation that the eigenvalues of the empirically observed Ψ_k matrices tend to decay very quickly, with only a few hundreds of largest eigenvalues being significantly different from zero. In this case, we can fix a $\nu \in \mathbb{N}$ and design a simple, rank$-\nu$ approximation to Ψ_K by annihilating all eigenvalues below $\lambda_\nu(\Psi_K)$. As we now show, it is possible to design a recursive algorithm for constructing such an approximation to Ψ_K, dealing with small subsets of random features simultaneously. To this end, we proceed as follows.

Suppose we have constructed an approximation $\hat\Psi_k \in \mathbb{R}^{N\times N}$ to Ψ_k with rank ν, and let $V_k \in \mathbb{R}^{N\times\nu}$ be the corresponding matrix of orthogonal eigenvectors for the non-zero eigenvalues, and $D_k \in \mathbb{R}^{\nu\times\nu}$ the diagonal matrix of eigenvalues so that $\hat\Psi_k = V_k D_k V_k^\top$ and $V_k^\top V_k = I_{\nu\times\nu}$. Instead of storing the full $\hat\Psi_k$ matrix, we only need to store the pair (V_k, D_k). For all $k = 1, \dots, K$, we now define

$$\tilde\Psi_{k+1} \;=\; \hat\Psi_k \;+\; S_{k+1}S_{k+1}^\top\,. \tag{3.8}$$

This $N\times N$ matrix is a theoretical construct. We never actually compute it (see Algorithm 4). Let $\Theta_k = I - V_kV_k^\top$ be the orthogonal projection on the kernel of $\hat\Psi_k$, and

$$\tilde S_{k+1} \;=\; \Theta_k S_{k+1} \;=\; S_{k+1} - \underbrace{V_k}_{N\times\nu}\underbrace{(V_k^\top S_{k+1})}_{\nu\times P_1} \tag{3.9}$$

be $\tilde S_{k+1}$ orthogonalized with respect to the columns of V_k. Then, we define $\tilde W_{k+1}$ to be a matrix with orthogonalized columns of $\tilde S_{k+1}$, and $\hat V_{k+1} = [V_k, \tilde W_{k+1}]$. We can compute $\tilde W_{k+1}$ using the following lemma.

Lemma 3.3.1. *Let* $\underbrace{\tilde S_{k+1}^\top \tilde S_{k+1}}_{P_1\times P_1} = W\delta W^\top$ *be the eigenvalue decomposition of* $\tilde S_{k+1}^\top \tilde S_{k+1}$. *Then,* $\tilde W_{k+1} = \tilde S_{k+1} W \delta^{-1/2}$ *is the matrix of eigenvectors of* $\tilde S_{k+1}\tilde S_{k+1}^\top$ *for the non-zero eigenvalues.*

By construction, the columns of $\hat V_{k+1}$ form an orthogonal basis of the span of the columns of V_k, S_{k+1}, and hence

$$\Psi_{k+1,*} \;=\; \hat V_{k+1}^\top \tilde\Psi_{k+1} \hat V_{k+1} \;\in\; \mathbb{R}^{(P_1+\nu)\times(P_1+\nu)} \tag{3.10}$$

has the same non-zero eigenvalues as $\tilde\Psi_{k+1}$. We then define $\tilde V_{k+1} \in \mathbb{R}^{(P_1+\nu)\times\nu}$ to be the matrix with eigenvectors of $\Psi_{k+1,*}$ for the largest ν eigenvalues, and we denote the diagonal matrix of these eigenvalues by $D_{k+1} \in \mathbb{R}^{\nu\times\nu}$, and then we define $V_{k+1} \;=\; \hat V_{k+1}\tilde V_{k+1}$. Then, $\hat\Psi_{k+1} \;=\; V_{k+1}D_{k+1}V_{k+1} \;=\; \Pi_{k+1}\tilde\Psi_{k+1}\Pi_{k+1}^\top$, where $\Pi_{k+1} \;=\; \hat V_{k+1}\tilde V_{k+1}\tilde V_{k+1}^\top \hat V_{k+1}^\top$ is the orthogonal projection onto the eigen-subspace of $\tilde\Psi_{k+1}$ for the largest ν eigenvalues.

Lemma 3.3.2. *We have* $\hat\Psi_k \leq \tilde\Psi_k \leq \Psi_K$ *and*

$$\|\Psi_k - \hat\Psi_k\| \;\leq\; \sum_{i=1}^{k} \lambda_{\nu+1}(\Psi_i) \;\leq\; k\,\lambda_{\nu+1}(\Psi_K)\,, \tag{3.11}$$

and

$$\|(\Psi_{k+1} + zI)^{-1} - (\hat\Psi_{k+1} + zI)^{-1}\| \;\leq\; z^{-2}\sum_{i=1}^{k} \lambda_{\nu+1}(\Psi_i)\,. \tag{3.12}$$

Proof. We have

$$\begin{aligned}\Psi_{k+1} &= \Psi_k + S_{k+1}S_{k+1}^\top \\ \tilde{\Psi}_{k+1} &= \hat{\Psi}_k + S_{k+1}S_{k+1}^\top \\ \hat{\Psi}_{k+1} &= P_{k+1}\tilde{\Psi}_{k+1}P_{k+1}^\top .\end{aligned} \tag{3.13}$$

By the definition of the spectral projection, we have

$$\|\tilde{\Psi}_{k+1} - \hat{\Psi}_{k+1}\| \le \lambda_{\nu+1}(\tilde{\Psi}_{k+1}) \le \lambda_{\nu+1}(\Psi_{k+1}), \tag{3.14}$$

and hence

$$\begin{aligned}&\|\Psi_{k+1} - \hat{\Psi}_{k+1}\| \\ &\le \|\Psi_{k+1} - \tilde{\Psi}_{k+1}\| + \|\tilde{\Psi}_{k+1} - \hat{\Psi}_{k+1}\| \\ &= \|\Psi_k - \hat{\Psi}_k\| + \|\tilde{\Psi}_{k+1} - \hat{\Psi}_{k+1}\| \\ &\le \|\Psi_k - \hat{\Psi}_k\| + \lambda_{\nu+1}(\Psi_{k+1}),\end{aligned} \tag{3.15}$$

and the claim follows by induction. The last claim follows from the simple inequality

$$\|(\Psi_{k+1} + zI)^{-1} - (\hat{\Psi}_{k+1} + zI)^{-1}\| \le z^{-2}\|\Psi_{k+1} - \hat{\Psi}_{k+1}\|. \tag{3.16}$$

□

There is another important aspect of our algorithm: It allows us to directly compute the performance of models with an expanding level of complexity. Indeed, since we load random features in batches of size P_1, we generate predictions for $P \in [P_1, 2P_1, \cdots, KP_1]$. This is useful because we might use it to calibrate the optimal degree of complexity and because we can directly study the double descent-like phenomena, see, e.g., [Bel+18] and [Nak+21]. That is the effect of complexity on the generalization error. In the next section, we do this. As we show, consistent with recent theoretical results [KMZ22], with sufficient shrinkage, the double descent curve disappears, and the performance becomes almost monotonic in complexity. Following [KMZ22], we name this phenomenon *the virtue of complexity (VoC)* and the corresponding performance plots *the VoC curves.* See, Figure 3.6 below.

We call this algorithm Fast Annihilating Batch Regression (FABReg) as it annihilates all eigenvalues below $\lambda_\nu(\Psi_K)$ and allows to solve the random features ridge regression in one go for a grid of z. Algorithm 4 formalizes the logic described above.

3.4. Numerical Results

This section presents several experimental results on different datasets to evaluate FABReg's performance and applications. In contrast to the most recent computational power demand in kernel methods, e.g., [Sha+20], we ran all experiments on a laptop, a MacBook Pro model A2485, equipped with an M1 Max with a 10-core CPU and 32 GB RAM.

3.4.1. A comparison with sklearn

We now aim to show FABReg's training and prediction time with respect to the number of features d. To this end, we do not use any random feature projection or the rank-ν matrix approximation described in Section 3.3.1. We draw $N = 5000$ i.i.d. samples from $\otimes_{j=1}^d \mathcal{N}(0,1)$ and let

$$y_i = x_i\beta + \epsilon_i \quad \forall i = 1, \ldots, N,$$

where $\beta \sim \otimes_{j=1}^d \mathcal{N}(0,1)$, and $\epsilon_i \sim \mathcal{N}(0,1)$ for all $i = 1, \ldots, N$. Then, we define

$$y_i = \begin{cases} 1 & \text{if } y_i > \text{median}(\boldsymbol{y}), \\ 0 & \text{otherwise} \end{cases} \quad \forall i = 1, \ldots, N.$$

Next, we create a set of datasets for classification with varying complexity d and keep the first 4000 samples as the training set and the remaining 1000 as the test set. We show in Figure 3.1

Algorithm 4 FABReg-ν

```
Require: $\nu$, $P_1$, $P$, $X \in \mathbb{R}^{N\times d}$, $y \in \mathbb{R}^N$, $z$, $voc\_curve$
  $blocks \leftarrow P//P_1$
  $k \leftarrow 0$
  while $k < blocks$ do
    Generate $S_k \in \mathbb{R}^{N\times P_1}$ {Use $k$ as seed to generate the random features}
    if $k = 0$ then
      $\tilde{d}, \tilde{V} \leftarrow eigen(S_k^\top S_k)$
      $V \leftarrow S_k \tilde{V} diag(\tilde{d})^{-\frac{1}{2}}$
      $V_0 \leftarrow V_{:,min(\nu,P_1)}$ {Save $V_0$}
      $d_0 \leftarrow \tilde{d}_{:min(\nu,P_1)}$ {Save $d_0$}
      if k in $voc\_curve$ then
        $Q_0(z) \leftarrow V_0(diag(d_0) + zI)^{-1} V_0^\top y$ {Save $Q_0(z)$}
      end if
    else if $k > 0$ then
      $\tilde{S_k} \leftarrow (I - V_{k-1}V_{k-1}^\top)S_k$
      $\Gamma_k \leftarrow \tilde{S_k}^\top \tilde{S_k}$
      $\delta_k, W_k \leftarrow eigen(\Gamma_k)$
      Keep top min$(\nu, P1)$ eigenvalues and eigenvectors from $\delta_k, W_k$
      $\tilde{W_k} \leftarrow \tilde{S_k} W_k diag(\delta_k)^{-\frac{1}{2}}$
      $\hat{V_k} \leftarrow [V_{k-1}, \tilde{W_k}]$
      $\bar{V_k} \leftarrow \hat{V_k}^\top V_{k-1}$
      $\bar{W_k} \leftarrow \bar{V_k} diag(d_{k-1}) \bar{V_k}^\top$
      $\bar{S_k} \leftarrow \hat{V_k}^\top S_k$
      $\bar{Z_k} \leftarrow \bar{S_k} S_k^\top$
      $\Psi_* \leftarrow \bar{W_k} \bar{Z_k}$
      $d_k, V_k \leftarrow eigen(\Psi_*)$
      Keep top min$(\nu, P1)$ eigenvalues and eigenvectors from $d_k, V_k$
      $V_k \leftarrow \hat{V_k} V_k$ {Save $d_k, V_k$}
      if k in $voc\_curve$ then
        $Q_k(z) \leftarrow V_k(diag(d_k) + zI)^{-1} V_k^\top y$ {Save $Q_k(z)$}
      end if
    end if
    $k = k + 1$
  end while
  $k \leftarrow 0$
  while $k < blocks$ do
    (re-)Generate $S_k \in \mathbb{R}^{N\times P_1}$ {Use $k$ as seed to generate the random features}
    $\beta_k(z) \leftarrow S_k^\top Q_k(z)$
    $\hat{y}$ += $S_k \beta_k$
  end while
```

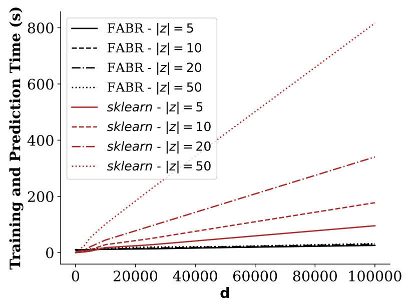


Figure 3.1: The figure above compares FABReg training and prediction time, shown on the y-axis, in black, against *sklearn*'s RidgeClassifier, in red, for an increasing amount of features, shown on the x-axis, and the number of shrinkages z. Here, $|z|$ denotes the number of different values of z for which we perform the training.

the average training and prediction time (in seconds) of FABReg with a different number of regularizers (we denote this number by $|z|$) and *sklearn* RidgeClassifier with an increasing number of features d. The training and prediction time is averaged over five independent runs. As one can see, our method is drastically faster when $d > 10000$. E.g., for $d = 100000$ we outperform *sklearn* by approximately 5 and 25 times for $|z| = 5$ and $|z| = 50$, respectively. Moreover, one can notice that the number of different shrinkages $|z|$ does not affect FABReg. We report a more detailed table with average training and prediction time and standard deviation in Appendix 3.7.

3.4.2. Experiments on Real Datasets

We assess FABReg's performance on both small and big datasets regimes for further evaluation. For all experiments, we perform a random features kernel ridge regression for demeaned one-hot labels and solve the optimization problem using FABReg as described in Section 3.3.

Data Representation

Table 3.1: The table below shows the average test accuracy and standard deviation of ResNet-34, CNTK, and FABR on the subsampled CIFAR-10 datasets. The test accuracy is average over twenty independent runs.

n	ResNet-34	14-layer CNTK	z=1	z=100	z=10000	z=100000
10	14.59% ± 1.99%	15.33% ± 2.43%	18.50% ± 2.18%	**18.50%** ± **2.18%**	18.42% ± 2.13%	18.13% ± 2.01%
20	17.50% ± 2.47%	18.79% ± 2.13%	20.84% ± 2.38%	**20.85%** ± **2.38%**	20.78% ± 2.35%	20.13% ± 2.34%
40	19.52% ± 1.39%	21.34% ± 1.91%	25.09% ± 1.76%	25.10% ± 1.76%	**25.14%** ± **1.75%**	24.41% ± 1.88%
80	23.32% ± 1.61%	25.48% ± 1.91%	29.61% ± 1.35%	29.60% ± 1.35%	**29.62%** ± **1.39%**	28.63% ± 1.66%
160	28.30% ± 1.38%	30.48% ± 1.17%	34.86% ± 1.12%	34.87% ± 1.12%	**35.02%** ± **1.11%**	33.54% ± 1.24%
320	33.15% ± 1.20%	36.57% ± 0.88%	40.46% ± 0.73%	40.47% ± 0.73%	**40.66%** ± **0.72%**	39.34% ± 0.72%
640	41.66% ± 1.09%	42.63% ± 0.68%	45.68% ± 0.71%	45.68% ± 0.72%	**46.17%** ± **0.68%**	44.91% ± 0.72%
1280	49.14% ± 1.31%	48.86% ± 0.68%	50.30% ± 0.57%	50.32% ± 0.56%	**51.05%** ± **0.54%**	49.74% ± 0.42%

FABReg requires, like any standard kernel methods or randomized-feature techniques, a good data representation. Usually, we don't know such a representation *a-priori*, and learning a good kernel is outside the scope of this paper. Therefore, we build a simple Convolutional

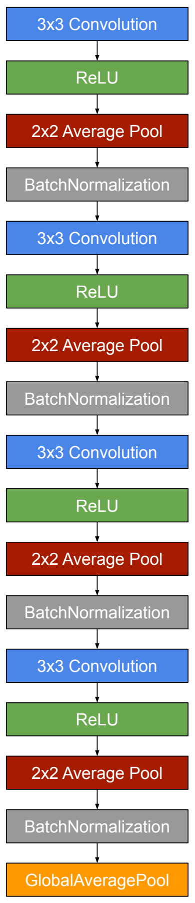


Figure 3.2: CNN architecture used to extract image features.

Neural Network (CNN) mapping $h : \mathbb{R}^d \to \mathbb{R}^D$; that extracts image features $\tilde{x} \in \mathbb{R}^D$ for some sample $x \in \mathbb{R}^d$. The CNN is not optimized; we use it as a simple random feature mapping. The CNN architecture, shown in Fig. 3.2, alternates a 3×3 convolution layer with a *ReLU* activation function, a 2×2 Average Pool, and a BatchNormalization layer [IS15a]. Convolutional layers weights are initialized using He Uniform [He+15]. To vectorize images, we use a global average pooling layer that has proven to enforce correspondences between feature maps and to be more robust to spatial translations of the input [LCY13]. We finally obtain the train and test random features realizations $s = f(\tilde{x}, \theta)$. Specifically, we use the following random features mapping

$$s_i = \sigma(W\tilde{x}), \tag{3.17}$$

where $W \in \mathbb{R}^{P \times D}$ with $w_{i,j} \sim \mathcal{N}(0, 1)$ and σ is some elementwise activation function. This can be described as a one-layer neural network with random weights W. To show the importance of over-parametrized models, throughout the results, we report the *complexity*, c, of the model as $c = P/N$, that is, the ratio between the parameters (dimensions) and the number of observations. See [Bel+18; Has+19; KMZ22].

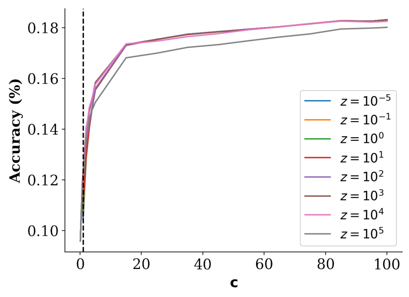


Figure 3.3: The figures above show FABReg's test accuracy increases with the model's complexity c on the subsampled CIFAR-10 dataset for $n = 10$. The test accuracy is averaged over five independent runs.

Small Datasets

We now study the performance of FABReg on the subsampled CIFAR-10 dataset [KH+09]. To this end, we reproduce the same experiment described in [Aro+19b]. In particular, we obtain random subsampled training set $(y; X) = (y_i; x_i)_{i=1}^n$ where $n \in \{10, 20, 40, 80, 160, 320, 640, 1280\}$ and test on the *whole* test set of size 10000. We make sure that exactly $n/10$ sample from each image class is in the training sample. We train FABReg using random features projection of the subsampled training set

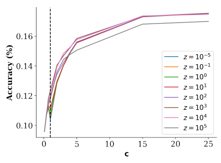
$$S = \sigma(Wg(X)) \in \mathbb{R}^{n \times P},$$

where g is an untrained CNN from Figure 3.2, randomly initialized using He Uniform distribution. In this experiment, we push the model complexity c to 100; in other words, FABReg's number of parameters equals a hundred times the number of observations in the subsample. As n is small, we deliberately do not perform any low-rank covariance matrix approximation. Finally, we run our model twenty times and report the mean out-of-sample performance and the standard deviation. We report in Table 3.1 FABReg's performance for different shrinkages (z) together with ResNet-34 and the 14-layers CNTK. Without any complicated random feature projection, FABReg can outperform both ResNet-34 and CNTK. FABReg's test accuracy increases with the model's complexity c on different (n) subsampled CIFAR-10 datasets. We show Figure 3.3 as an example for $n = 10$. Additionally, we show, to better observe the double descent phenomena, truncated curves at $c = 25$ for all CIFAR-10 subsamples in Figure 3.4. The full curves are shown in Appendix 3.7. To sum up this section findings:

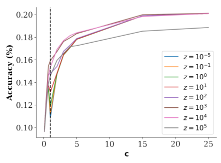
- FABReg, with enough complexity together and a simple random feature projection, is able to outperform deep neural networks (ResNet-34) and CNTKs.
- FABReg always reaches the maximum accuracy beyond the interpolation threshold.
- Moreover, if the random feature ridge regression shrinkage z is sufficiently high, the double descent phenomenon disappears, and the accuracy does not drop at the interpolation threshold point, i.e., when $c = 1$ or $n = P$. Following [KMZ22], we call this phenomenon *virtue of complexity* (VoC).

Big Datasets

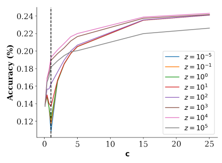
In this section, we repeat the same experiments described in Section 3.4.2, but we extend the training set size n up to the full CIFAR-10 dataset. For each n, we train FABReg, FABReg-ν with a rank-ν approximation as described in Algorithm 4, and the min-batch FABReg. We use $\nu = 2000$ and batch size $= 2000$ in the last two algorithms. Following [Aro+19b], we train ResNet-34 as the benchmark for 160 epochs, with an initial learning rate of 0.001 and



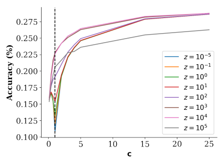
(a) $n = 10$



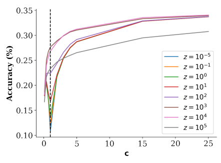
(b) $n = 20$



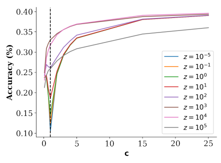
(c) $n = 40$



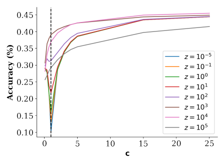
(d) $n = 80$



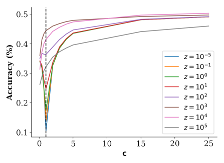
(e) $n = 160$



(f) $n = 320$



(g) $n = 640$



(h) $n = 1280$

Figure 3.4: The figures above show FABReg's test accuracy increases with the model's complexity c on different (n) subsampled CIFAR-10 datasets. The expanded dataset follows similar patterns. We truncate the curve for $c > 25$ to better show the double descent phenomena. The full curves are shown in Appendix 3.7. Notice that when the shrinkage is sufficiently high, the double descent disappears, and the accuracy monotonically increases in complexity. Following [KMZ22], we name this phenomenon *the virtue of complexity* (VoC). The test accuracy is averaged over 20 independent runs.

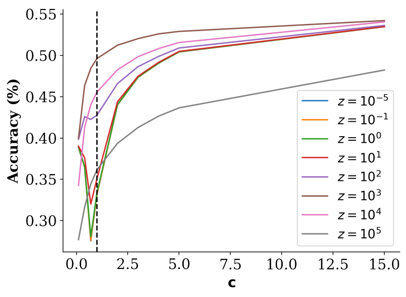
a batch size of 32. We decrease the learning rate by ten at epochs 80 and 120. ResNet-34 always reaches close to perfect accuracy on the training set, i.e., above 99%. We run each training five times and report mean out-of-sample performance and its standard deviation. As the training sample is sufficiently large already, we set the model complexity to *only* $c = 15$, meaning that for the full sample, FABReg performs a random feature ridge regression with $P = 7.5 \times 10^5$. We report the results in Tables 3.4.2 and 3.3.

Table 3.2: The table below shows the average test accuracy and standard deviation of ResNet-34 and FABR on the subsampled and full CIFAR-10 dataset. The test accuracy is average over five independent runs.

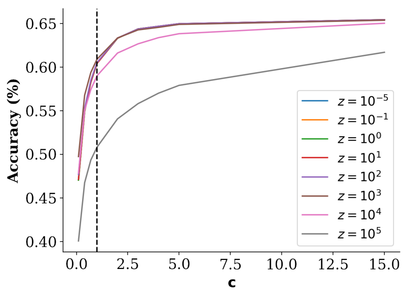
n	ResNet-34	z=1	z=100	z=10000	z=100000
2560	48.12% ± 0.69%	52.24% ± 0.29%	52.45% ± 0.21%	**54.29%** ± **0.44%**	48.28% ± 0.37%
5120	56.03% ± 0.82%	55.34% ± 0.32%	55.74% ± 0.34%	**58.29%** ± **0.20%**	52.06% ± 0.08%
10240	**63.21%** ± **0.26%**	58.36% ± 0.45%	58.86% ± 0.54%	62.17% ± 0.35%	55.75% ± 0.18%
20480	**69.24%** ± **0.47%**	61.08% ± 0.17%	61.65% ± 0.27%	65.12% ± 0.19%	59.34% ± 0.14%
50000	**75.34%** ± **0.21%**	66.38% ± 0.00%	66.98% ± 0.00%	68.62% ± 0.00%	63.25% ± 0.00%

The experiment delivers a number of additional conclusions:

- First, we observe that, while for small train sample sizes of $n \leq 10000$, simple kernel methods achieve performance comparable with that of DNNs, this is not the case for $n > 20000$. Beating DNNs on big datasets with shallow methods requires more complex kernels, such as those in [Sha+20; Li+19].
- Second, we confirm the findings of [MB17; Lee+20] suggesting that the role of small eigenvalues is important. For example, FABReg-ν with $\nu = 2000$ loses several percent of accuracy on larger datasets.



(a) $n = 2560$



(b) $n = 50000$

Figure 3.5: The figures above show FABReg's test accuracy increases with the model's complexity c on the subsampled CIFAR-10 dataset 3.5a and the full CIFAR-10 dataset 3.5b. FABReg is trained using a $\nu = 2000$ low-rank covariance matrix approximation. Notice that we still observe a (shifted) double descent when $\nu \approx n$. The same phenomenon disappears when $\nu \ll n$. The test accuracy is averaged over five independent runs.

Table 3.3: The table below shows the average test accuracy and standard deviation of FABReg-ν and mini-batch FABReg on the subsampled and full CIFAR-10 dataset. The test accuracy is average over five independent runs.

	$z = 1$		$z = 100$		$z = 10000$		$z = 100000$	
FABReg n	batch = 2000	$\nu = 2000$	batch = 2000	$\nu = 2000$	batch = 2000	$\nu = 2000$	batch = 2000	$\nu = 2000$
2560	53.13% ± 0.38%	53.48% ± 0.22%	53.15% ± 0.42%	53.63% ± 0.24%	52.01% ± 0.51%	54.05% ± 0.44%	46.78% ± 0.52%	48.23% ± 0.34%
5120	57.68% ± 0.18%	57.63% ± 0.19%	57.70% ± 0.16%	57.63% ± 0.18%	56.83% ± 0.27%	57.53% ± 0.11%	51.42% ± 0.22%	51.75% ± 0.14%
10240	59.79% ± 0.35%	61.20% ± 0.39%	59.79% ± 0.35%	61.20% ± 0.38%	58.63% ± 0.28%	60.63% ± 0.21%	53.73% ± 0.37%	55.16% ± 0.34%
20480	61.56% ± 0.35%	63.50% ± 0.12%	61.55% ± 0.37%	63.50% ± 0.13%	60.90% ± 0.20%	62.92% ± 0.12%	57.10% ± 0.19%	58.40% ± 0.21%
50000	62.74% ± 0.10%	65.45% ± 0.18%	62.74% ± 0.10%	65.44% ± 0.18%	62.35% ± 0.05%	65.04% ± 0.19%	59.99% ± 0.02%	61.71% ± 0.09%

- Third, surprisingly, both the mini-batch FABReg and FABReg-ν sometimes achieve higher accuracy than the full sample regression on moderately-sized datasets. See Tables 3.2 and 3.3. Understanding these phenomena is an interesting direction for future research.
- Fourth, the double descent phenomenon naturally appears for both FABReg-ν and the mini-batch FABReg but only when $\nu \approx n$ or batch size $\approx n$. However, the double descent phenomenon disappears when $\nu \ll n$. This intriguing finding is shown in Figure 3.5 for FABReg-ν, and in Appendix 3.7 for the mini-batch FABReg.
- Fifth, on average, FABReg-ν outperforms mini-batch FABReg on larger datasets.

3.5. Conclusion and Discussion

The recent discovery of the equivalence between infinitely wide neural networks (NNs) in the lazy training regime and neural tangent kernels (NTKs) [JGH18] has revived interest in kernel methods. However, these kernels are extremely complex and usually require running on big and expensive computing clusters [ACW17; Sha+20] due to memory (RAM) requirements. This paper proposes a highly scalable random features ridge regression that can run on a simple laptop. We name it Fast Annihilating Batch Regression (FABReg). Thanks to the linear algebraic properties of covariance matrices, this tool can be applied to any kernel and any way of generating random features. Moreover, we provide several experimental results to assess its performance. We show how FABReg can outperform (in training and prediction speed) the current state-of-the-art ridge classifier's implementation. Then, we show how a simple data representation strategy combined with a random features ridge regression can outperform complicated kernels (CNTKs) and over-parametrized Deep Neural Networks (ResNet-34) in the few-shot learning setting. The experiments section concludes by showing additional results on big datasets. In this paper, we focus on very simple classes of random features. Recent findings (see, e.g., [Sha+20]) suggest that highly complex kernel architectures are necessary to achieve competitive performance on large datasets. Since each kernel regression can be approximated with random features, our method is potentially applicable to these kernels as well. However, directly computing the random feature representation of such complex kernels is non-trivial and we leave it for future research.

3.6. Appendix - Proofs

3.7. Appendix - Additional Experimental Results

This section provides additional experiments and findings that may xhelp the community with future research.

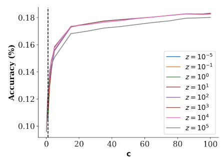
First, we dive into more details about our comparison with *sklearn*. Table 3.4 shows a more detailed training and prediction time comparison between FABReg and *sklearn*. In particular, we average training and prediction time over five independent runs. The experiment settings are explained in Section 3.4.1. We show how one, depending on the number shrinkages $|z|$, would start considering using FABReg when the number of observations in the dataset $n \approx 5000$. In this case, we have used the *numpy* linear algebra library to decompose FABReg's covariance matrix, which appears to be faster than the *scipy* counterpart. We share our code in the following repository: `https://github.com/tengandreaxu/fabr`.

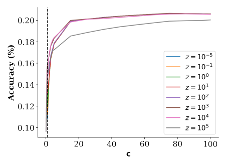
Second, while Figure 3.4 shows FABReg's test accuracy on increasing complexity c truncated curves, we present here the whole picture; i.e., Figure 3.6 shows full FABReg's test accuracy increases with the model's complexity c on different (n) subsampled CIFAR-10 datasets averaged over twenty independent runs. The expanded dataset follows similar patterns. Similar to Figure 3.4, one can notice that when the shrinkage is sufficiently high, the double descent disappears, and the accuracy monotonically increases in complexity.

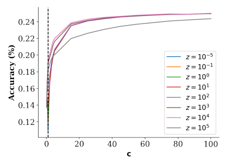
Third, the double descent phenomenon naturally appears for both FABReg-ν and the mini-batch FABReg but only when $\nu \approx n$ or batch size $\approx n$. However, the double descent phenomenon disappears when $\nu \ll n$. This intriguing finding is shown in Figure 3.5 for FABReg-ν, and here, in Figure 3.7, we report the same curves for mini-batch FABReg.

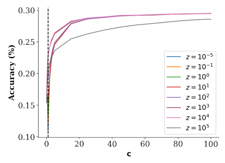
Table 3.4: The table below shows FABReg and *sklearn*'s training and prediction time (in seconds) on a synthetic dataset. We vary the dataset number of features d and the number of shrinkages $(|z|)$. We report the average running time and the standard deviation over five independent runs.

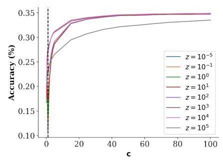
	$\|z\| = 5$		$\|z\| = 10$		$\|z\| = 20$		$\|z\| = 50$	
	FABReg	*sklearn*	FABReg	*sklearn*	FABReg	*sklearn*	FABReg	*sklearn*
d								
10	7.72s ± 0.36s	**0.01s ± 0.00s**	6.90s ± 0.77s	**0.02s ± 0.00s**	7.04s ± 0.67s	**0.03s ± 0.00s**	7.44s ± 0.57s	**0.07s ± 0.01s**
100	7.35s ± 0.36s	**0.06s ± 0.02s**	6.58s ± 0.34s	**0.11s ± 0.01s**	7.61s ± 1.14s	**0.24s ± 0.04s**	7.3s ± 0.49s	**0.53s ± 0.06s**
500	7.37s ± 0.44s	**0.33s ± 0.16s**	6.81s ± 0.25s	**0.54s ± 0.03s**	7.02s ± 0.35s	**1.01s ± 0.07s**	7.44s ± 0.48s	**2.41s ± 0.21s**
1000	7.62s ± 0.31s	**0.58s ± 0.21s**	7.38s ± 0.23s	**1.06s ± 0.04s**	7.51s ± 0.24s	**2.04s ± 0.04s**	7.69s ± 0.08s	**4.79s ± 0.36s**
2000	8.33s ± 0.42s	**1.21s ± 0.03s**	8.09s ± 0.73s	**2.44s ± 0.05s**	8.33s ± 0.24s	**4.87s ± 0.07s**	**8.29s ± 0.47s**	12.21s ± 0.15s
3000	9.24s ± 0.25s	**2.49s ± 0.05s**	9.18s ± 0.41s	**5.08s ± 0.03s**	**9.51s ± 0.20s**	10.06s ± 0.02s	**9.67s ± 0.41s**	25.67s ± 0.23s
5000	10.64s ± 0.86s	**5.36s ± 0.05s**	11.01s ± 0.7s	**10.74s ± 0.06s**	**11.57s ± 0.81s**	21.31s ± 0.12s	**11.54s ± 0.41s**	54.18s ± 0.73s
10000	**11.49s ± 0.66s**	17.87s ± 8.58s	**11.81s ± 0.47s**	28.32s ± 10.53s	**11.61s ± 0.49s**	44.72s ± 9.99s	**12.55s ± 0.3s**	101.58s ± 15.66s
25000	**13.89s ± 0.21s**	27.79s ± 8.75s	**14.50s ± 0.45s**	49.84s ± 9.68s	**14.46s ± 0.96s**	94.08s ± 10.94s	**15.68s ± 0.74s**	224.31s ± 11.75s
50000	**17.99s ± 0.22s**	50.51s ± 8.99s	**18.27s ± 0.37s**	92.88s ± 10.45s	**19.10s ± 0.37s**	176.24s ± 10.07s	**19.68s ± 0.85s**	422.95s ± 13.22s
100000	**25.30s ± 0.39s**	95.57s ± 0.25s	**26.16s ± 0.46s**	177.54s ± 3.77s	**27.93s ± 0.35s**	340.32s ± 3.74s	**29.48s ± 1.38s**	816.25s ± 4.35s

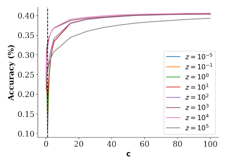

(a) $n = 10$

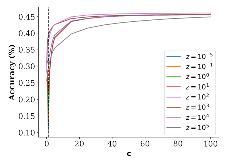

(b) $n = 20$


(c) $n = 40$


(d) $n = 80$


(e) $n = 160$


(f) $n = 320$


(g) $n = 640$

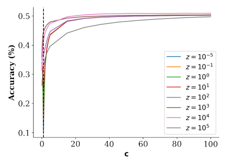
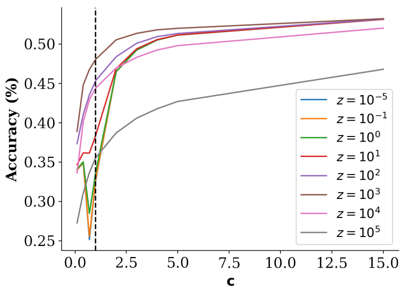
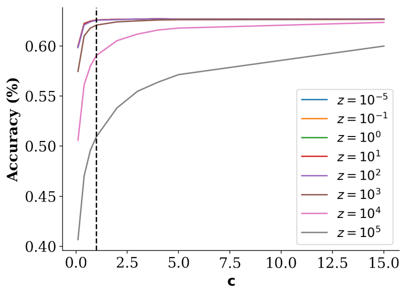

(h) $n = 1280$

Figure 3.6: The figure above shows the full FABReg's accuracy increase with the model's complexity c in the small dataset regime. The expanded dataset follows similar patterns.



(a) $n = 2560$



(b) $n = 50000$

Figure 3.7: Similar to Figure 3.5, the figures above show FABReg's test accuracy increases with the model's complexity c on the subsampled CIFAR-10 dataset 3.7a and the full CIFAR-10 dataset 3.7b. FABReg trains using mini-batches with batch size=2000 in both cases. Notice that we still observe a (shifted) double descent when batch size $\approx n$, while the same phenomenon disappears when batch size $\ll n$. The test accuracy is averaged over 5 independent runs.

4. Conclusion

This work was inspired by recent breakthroughs in combining deep learning with empirical asset pricing. Regarding neural nets, the "black-box" criticism—the fact that we do not yet fully understand their incredible out-of-sample performance and their feature learning process—is slowly fading away. These models are interpolators acting like Gaussian processes [Nea96; Lee+18] in the finite case, while they are equivalent to kernels [JGH18; Aro+19b; Aro+19a] when the width is pushed to the infinite limit. Finite neural networks are provable to learn features via *Dynamic Dichotomy Theorem* [Yan+23]. Optimal parametrization, guaranteeing converge settings, are explained via the *Tensor Programs* work series [Yan19; Yan20b; YH20; Yan+23], to cite a few, and under the *dynamic isometry* technical condition [SMG13]. In particular, under [Yan+21] parametrization, wider neural networks are *always* better in terms of out-of-sample performance. When the network satisfies the dynamic isometry condition, the deeper neural networks will not collapse during training [PSG17]. Deeper neural networks are proven–via Riemann geometry and mean-field theory [Poo+16]–to show higher (exponential) expressivity growth, *i.e.*, higher generalization, than their shallow counterparties. Highly parametrized statistical models are being used in return predictions as well. The economic gain from more accurate forecasts comes from better portfolio construction and, thus, utility for the agent. [GKX20b] set the breaking point by showing that machine learning models were outperforming the class least-square models in terms of R^2 and Sharpe ratio. Then, the series of work of [KMZ22] and [KMZ24] has provably shown how highly parametrized models (when the number of parameters is higher than the number of data points) higher out-of-sample performance. This is directly connected to the *double descent* phenomenon in the machine learning literature [Bel+18; Has+19]. These discoveries together with other phenomena, such as *grokking* [Pow+22] and *neural collapse* [PHD20], have put in discussion the *principle of parsimony* [Box+15], the variance-bias trade-off [Bel+18], and different type of regularization such as dropout and early-stopping. With these in mind, we have explored different topics in empirical asset pricing using neural networks and machine learning models. The conclusion of each chapter, and thus, their contribution, is briefly reported below.

Tail Recovery In this paper, we have investigated the ability of complex machine learning models to extract predictive information about tail risk, that is, the likelihood (and the distribution) of large, unexpected moves in the underlying asset prices. We find that the distribution of returns upon the arrival of a tail event can be efficiently predicted out of sample, and the performance of our predictions is comparable to that for forecasting realized volatility. Our dataset covers the period of 2007-01-03 to 2022-12-31 and contains data (Black-Scholes implied volatilities, underlying asset prices, volume, open interest, asks, bids) for 5349 different stocks traded at NYSE, AMEX, and NASDAQ at *daily* frequency. We use the classic theorem of [BDH74] to approximate the tail distribution to a generalized Pareto distribution, and we follow the approach of [GKX20b] to conduct our studies. First, options information about tails is spread across option moneyness in a complex fashion; in fact, we find that OTM calls (respectively, puts) contain important information about lower (respectively, upper) tails. Second, we find that predicting upper tails is easier than predicting lower tails. Third, upper- and lower-tail risks can load positively on puts information and call information, respectively. Then, we provide a new non-linear predictive model capable of outperforming the most common linear and non-linear counterparts in the literature through extreme value theory. Next, we find that the magnitude of predicted tail risk builds up monotonically during the two weeks preceding earnings announcements and abruptly drops once the information is released. Finally, we show how both put and call information are important for deep neural networks in predicting tail risks, and we find new evidence in equity premia's non-linear nature. Understanding the links between our predicted tail risk and various stock characteristics is an important direction for future research. Moreover, the importance of OTM call (puts) information about the lower (upper) tail is surprising, opening new challenges for both theoretical and empirical studies.
Deep Learning from Options Implied Volatility Surfaces The remarkable growth of the factor zoo [FGX20], [BHJ23] over the last few years has been accompanied by the development of machine learning methods for asset pricing [GKX20b]. As [KMZ24] and [Did+23] explain, this is no coincidence: Factor zoo is a natural consequence of *complexity*: A highly non-linear predictive relationship between returns and characteristics. The most naive and

direct way of exploiting this complexity is to build large, unstructured non-linear models such as simple, fully connected neural networks of [GKX20b] or the random feature models of [KMZ24] and [Did+23]. While this approach works well with structured stock characteristics, it is unsuited for unstructured data, such as the IV surface. To deal with such data, one needs to develop tools and ML algorithms that exploit the data structure optimally. In this paper, we take a step in this direction and propose Convolutional Neural Networks (CNN) architecture designed specifically to extract features of the IV surface that respect locality, as economic theory requires. We show that CNNs can successfully identify highly complex non-linear relationships that cannot be learned with naive, fully-connected networks. Importantly, we find that consistent with the existing evidence for image data [LPB17], the loss landscape of the CNN is extremely non-convex and is characterized by a very large number of local minima. All those minima contain information about returns. Exploiting them requires using an ensemble of CNNs, and we document a very large *virtue of ensemble complexity.* Gaining insights into the incremental information offered by the model as it converges to different local minima for other return prediction problems (including even simpler ones, with the fully connected networks of [GKX20b]) is an important direction for future research. Conventional wisdom based on the numerous manually constructed option characteristics suggests that a few linear features of the IV surface (e.g., level, slope, skew, and convexity) should fully summarize its predictive content. To test this "linear feature sparsity hypothesis," we introduce a novel object in financial machine learning, the *gradient outer product*, whose eigenvectors, the *principal linear features*, are natural analogs of principal components for machine learning [Rad+22]. We find no evidence for linear feature sparsity and show that a very large number (more than 100) of linear features are necessary to explain the predictive content of IV, manifesting a very high *feature complexity.* Investigating principal linear features for other ML models and datasets might bring interesting novel insights into the different notions of sparsity in return prediction.

A Simple Algorithm for Scaling Up Kernel Methods The recent discovery of the equivalence between infinitely wide neural networks (NNs) in the lazy training regime and neural tangent kernels (NTKs) [JGH18] has revived interest in kernel methods. However, these kernels are extremely complex and usually require running on big and expensive computing clusters [ACW17; Sha+20] due to memory (RAM) requirements. This paper proposes a highly scalable random features ridge regression that can run on a simple laptop. We name it Fast Annihilating Batch Regression (FABReg). Thanks to the linear algebraic properties of covariance matrices, this tool can be applied to any kernel and any way of generating random features. Moreover, we provide several experimental results to assess its performance. We show how FABReg can outperform (in training and prediction speed) the current state-of-the-art ridge classifier's implementation. Then, we show how a simple data representation strategy combined with a random features ridge regression can outperform complicated kernels (CNTKs) and over-parametrized Deep Neural Networks (ResNet-34) in the few-shot learning setting. The experiments section concludes by showing additional results on big datasets. In this paper, we focus on very simple classes of random features. Recent findings (see, e.g., [Sha+20]) suggest that highly complex kernel architectures are necessary to achieve competitive performance on large datasets. Since each kernel regression can be approximated with random features, our method is potentially applicable to these kernels as well. However, directly computing the random feature representation of such complex kernels is non-trivial and we leave it for future research.

Curriculum

Teng Andrea Xu

Email : andrea.xu@epfl.ch
Mobile : +410787011864

Working Experience

- **AQR Capital Management** — Greenwich, CT, USA.
  Contractor - Full Remote — *December 2023 - Present*
  - Research.
- **École Polytechnique Fédérale de Lausanne (EPFL)** — Lausanne, Switzerland
  Ph.D. Swiss Finance Institute (SFI-SM) — *Sep 2020 - Present*
  - MTE 2020-2022 Best Teaching Assistant Award.
  - Granted 36,000 GPU hours on "Piz Daint" by the Swiss National Supercomputing Centre.
  - EPFL-CDMT Research Day 2023 Runner-up.
- **Schlossberg&Co.** — Zug, Switzerland
  Quant Engineer (Remote) — *June 2021 - November 2023*
  - Implemented a fully automated portfolio management strategy with Interactive Brokers (IB) TWS.
  - Built the distributed system in Google Cloud (GC) exploiting Python, Docker, PostgreSQL, and Linux.
  - Restructured the company's private dashboard using Google App Engine, Next.js, React, and Typescript.
- **Hercle** — Milan, Italy
  Software Developer (Remote) — *April 2022 - June 2022*
  - Integrated FTX and Bitpanda APIs to Hercle's high-frequency market-making algorithm in C-Sharp.
  - Managed user account information via REST endpoints.
  - Collected full-depth order book data and placed orders via Websocket and FIX API, respectively.
- **Science Quant Investments LLC** — Lausanne, Switzerland
  IT Consultant - Software Engineer — *Nov. 2020 - April 2022*
  - Created a (private) fully automated GC live options mid-frequency trading strategy.
  - Used software: Python, Docker, Firebase Cloud Firestore (CF), Redis, Angular8, and Linux.

- Analyzed and customized ad-hoc broker IB API using asynchronous socket message communications.

- **École Polytechnique Fédérale de Lausanne (EPFL)** Lausanne, Switzerland
  Software Engineer *June 2019 - Sep 2020*
  - Developed a GC production web application for retail option trading.
  - Built private mid-frequency options trading algorithm.
  - Led a group of three master students and coordinated their programming tasks.
- **University of Luxembourg** Luxembourg, Luxembourg
  Research Assistant *May 2018 - Nov. 2018*
  - Implemented the protocol IEEE 802.1 Audio Video Bridging (AVB) & Time-Sensitive Networking (TSN).
  - Built a working test bed using an open-source C++ project compliant with AVB/TSN standards.

Education

- **University of Yale** New Haven, Connecticut, USA
  Visiting PhD with Prof. Kelly *December 2023*
- **University of San Diego** San Diego, California, USA
  Visiting PhD with Prof. Belkin *October 2023*
- **University of Rome "La Sapienza".** Rome, Italy
  M.Sc. Eng. in Computer Sc.; Summa cum laude. *Sep. 2016 – Jan. 2019*
- **Rjksuniversiteit Groningen.** Groningen, Netherlands
  M.Sc. Computer Science (Exchange, 6 months); 8.5/10. Sep. 2017 – Jan. 2018
- **University of Rome 3.** Rome, Italy
  B.Sc. Information Technology Engineering; 103/110. *Sep. 2013 – Jul. 2016*

Published Papers

- ***Deep Learning from Implied Volatility Surfaces.*** *2023*
  *Bryan Kelly, Boris Kuznetsov, Semyon Malamud, and **TA Xu***
- ***A Simple Algorithm For Scaling Up Kernel Methods.*** *2023*
  ***TA Xu**, Bryan Kelly, and Semyon Malamud*
- ***Tail Recovery.*** *2023*
  TA Xu
- ***Benign Autoencoders.*** 2022
  ***TA Xu**, Semyon Malamud, and Antoine Didisheim*
- ***A Framework for DAO Token Valuation.*** *2022*
  ***TA Xu**, Jiahua Xu, and Kristof Lommers*
- ***DeFi vs TradFi: Valuation using multiples and discounted cash flows.*** *2022*
  ***TA Xu**, Jiahua Xu, and Kristof Lommers*
- ***A Short Survey on Business Models of DeFi Protocols.*** *2022*
  ***TA Xu** and Jiahua Xu*
- ***Poster: Performance Evaluation of an Open-Source Audio-Video Bridging/Time-Sensitive Networking Testbed for Automotive Ethernet*** *2018*
  ***TA Xu**, F. Adamsky, I. Turcanu, R. Soua, C. Köbel, T. Engel, and A. Baiocchi*

Teaching

- **Financial applications of blockchains and distributed ledgers.** *Fall 2022*
  FIN-413. Lecturer: Jiahua Xu.
- **Corporate strategy.** *2020-2022*
  MGT-400. Lecturer: Katherine Tatarinov.
- **Data driven business analytics.** *Spring 2021*
  MGT-302. Lecturer: S. Malamud, N. Kiyavash, and J. Etesami.

Talks

- **EPFL - CDMT Research Day.** *2023*
  Deep Learning from Implied Volatility Surfaces.
- **University College London - Financial Computing.** *2022*
  Benign Autoencoders.
- **Complexity Science Hub Vienna.** *2022*
  A Short Survey on Business Models of DeFi Protocols.
- **EPFL - CDMT Research Day.** *2022*
  A Simple Algorithm For Scaling Up Kernel Methods.
- **Financial Cryptography and Data Security - DeFi Workshop** *2022*
  A Short Survey on Business Models of DeFi Protocols.
- **University of Luxembourg - SECAN Lab.** *2018*
  Poster: AVB/TSN Testbed for Automotive Ethernet.

Skills

- **Programming Languages**
  Python, Java, Javascript, C++, C-Sharp, Typescript.
- **Softwares–Part 1**
  PostgreSQL, Docker, Redis, systemd, ufw, Nginx, gUnicorn
- **Softwares–Part 2**
  MongoDB, tmux, VSCode, Git, SLURM, Jenkins, ELK.
- **Libraries/Frameworks**
  Pytorch, Tensorflow, React, Angular, Next.js.
- **Languages–Part 1**
  English (Fluent). Italian (Native).
- **Languages–Part 2**
  Spanish, French, and Wenzhounese(Conversational). Turkish (Currently Learning).

Bibliography

[Ach+23] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. "Gpt-4 technical report". *arXiv preprint arXiv:2303.08774* (2023).

[ACM21] D. Avramov, S. Cheng, and L. Metzker. "Machine learning versus economic restrictions: Evidence from stock return predictability". *Available at SSRN 3450322* (2021).

[ACW17] H. Avron, K. L. Clarkson, and D. P. Woodruff. "Faster kernel ridge regression using sketching and preconditioning". *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1116–1138.

[ACX06] A. Ang, J. Chen, and Y. Xing. "Downside risk'". en. *Review of Financial Studies* 19 (2006), 1191–1239.

[AFT15] T. Andersen, N. Fusari, and V. Todorov. "The risk premia embedded in index options'". en. *Journal of Financial Economics* 117 (2015), 558–584.

[AHL19] F. Audrino, R. Huitema, and M. Ludwig. "An empirical implementation of the Ross recovery theorem as a prediction device". *Journal of Financial Econometrics* (2019).

[AKT19] A. Ali, J. Z. Kolter, and R. J. Tibshirani. "A continuous-time view of early stopping for least squares regression". *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1370–1378.

[AL01] D. Attwell and S. B. Laughlin. "An energy budget for signaling in the grey matter of the brain". *Journal of Cerebral Blood Flow & Metabolism* 21.10 (2001), pp. 1133–1145.

[Alm+17] C. Almeida, K. Ardison, R. Garcia, and J. Vicente. "Nonparametric tail risk, stock returns, and the macroeconomy". *Journal of Financial Econometrics* 15.3 (2017), pp. 333–376.

[Ama+15] D. Amaya, P. Christoffersen, K. Jacobs, and A. Vasquez. "Does realized skewness predict the cross-section of equity returns?'" en. *Journal of Financial Economics* 118.1 (2015), 135–167.

[An+14] B.-J. An, A. Ang, T. Bali, and N. Cakici. "The joint cross section of stocks and options'". en. *The Journal of Finance* 69.5 (2014), 2279–2337.

[Aro+19a] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. "On exact computation with an infinitely wide neural net". *Advances in Neural Information Processing Systems* 32 (2019).

[Aro+19b] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu. "Harnessing the power of infinitely wide deep nets on small-data tasks". *arXiv preprint arXiv:1910.01663* (2019).

[AS20] E. Abbe and C. Sandon. "Poly-time universality and limitations of deep learning". *arXiv preprint arXiv:2001.02992* (2020).

[AZLS19] Z. Allen-Zhu, Y. Li, and Z. Song. "A convergence theory for deep learning via over-parameterization". *International Conference on Machine Learning*. PMLR. 2019, pp. 242–252.

[Bar+20] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. "Benign overfitting in linear regression". *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070.

[Bat00] D. S. Bates. "Post-'87 crash fears in the S&P 500 futures option market". *Journal of econometrics* 94.1-2 (2000), pp. 181–238.

[BBG18] G. Baltussen, S. Bekkum, and B. Grient. "Unknown unknowns: uncertainty about risk and stock returns'". io. *Journal of Financial and Quantitative Analysis* 53.4 (2018), 1615–1651.

[BCYG18] G. Bakshi, F. Chabi-Yo, and X. Gao. “A recovery that we can trust? deducing and testing the restrictions of the recovery theorem”. *The Review of Financial Studies* 31.2 (2018), pp. 532–555.

[BCYM22] T. G. Bali, F. Chabi-Yo, and S. Murray. “A factor model for stock returns based on option prices”. *Available at SSRN 4071995* (2022).

[BDG20] J.-F. Begin, C. Dorion, and G. Gauthier. “Idiosyncratic jump risk matters: Evidence from equity returns and options’”. en. *Review of Financial Studies* 33 (2020), 155–211.

[BDH74] A. A. Balkema and L. De Haan. “Residual life time at great age”. *The Annals of probability* (1974), pp. 792–804.

[Bel+18] M Belkin, D Hsu, S Ma, and S Mandal. *Reconciling modern machine learning and the bias-variance trade-off. arXiv e-prints.* 2018.

[Bel21] M. Belkin. “Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation”. *Acta Numerica* 30 (2021), pp. 203–248.

[BGZ11] T. Bollerslev, M. Gibson, and H. Zhou. “Dynamic estimation of volatility risk premia and investor risk aversion from option-implied and realized volatilities”. *Journal of econometrics* 160.1 (2011), pp. 235–245.

[BH09] T. Bali and A. Hovakimian. “Volatility spreads and expected stock returns’”. en. *Management Science* 55.11 (2009), 1797–1812.

[BHJ23] S. Bryzgalova, J. Huang, and C. Julliard. “Bayesian solutions for the factor zoo: We just ran two quadrillion models”. *The Journal of Finance* 78.1 (2023), pp. 487–557.

[BHS16a] J. Borovička, L. P. Hansen, and J. A. Scheinkman. “Misspecified recovery”. *The Journal of Finance* 71.6 (2016), pp. 2493–2544.

[BHS16b] J. Borovička, L. Hansen, and J. Scheinkman. “Misspecified recovery’”. en. *The Journal of Finance* 71.6 (2016), 2493–2544.

[BHX20] M. Belkin, D. Hsu, and J. Xu. “Two models of double descent for weak features”. *SIAM Journal on Mathematics of Data Science* 2.4 (2020), pp. 1167–1180.

[Bie23] C. Biever. “ChatGPT broke the Turing test-the race is on for new ways to assess AI”. *Nature* 619.7971 (2023), pp. 686–689.

[BL76] H. J. Brascamp and E. H. Lieb. “On extensions of the Brunn-Minkowski and Prékopa-Leindler theorems, including inequalities for log concave functions, and with an application to the diffusion equation”. *Journal of functional analysis* 22.4 (1976), pp. 366–389.

[BL78a] D. T. Breeden and R. H. Litzenberger. “Prices of state-contingent claims implicit in option prices”. *Journal of business* (1978), pp. 621–651.

[BL78b] D. Breeden and R. Litzenberger. “Prices of state-contingent claims implicit in option prices’”. en. *The Journal of Business* 51 (1978), 621–651.

[BM19] T. Bali and S. Murray. “In search for a factor model for optionable stocks”. en. Unpublished manuscript, 2019.

[BM78] R. Banz and M. Miller. “Prices for state-contingent claims: Some estimates and applications’”. en. *Journal of Business* (1978), 653–672.

[BMM18] M. Belkin, S. Ma, and S. Mandal. “To understand deep learning we need to understand kernel learning”. *International Conference on Machine Learning.* PMLR. 2018, pp. 541–549.

[BMV10] B. Boyer, T. Mitton, and K. Vorking. “Expected idiosyncratic skewness’”. en. *Review of Financial Studies* 23 (2010), 169–202.

[Box+15] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[BPZ20] S. Bryzgalova, M. Pelger, and J. Zhu. “Forest through the trees: Building cross-sections of stock returns”. *Available at SSRN 3493458* (2020).

[BRT19] M. Belkin, A. Rakhlin, and A. B. Tsybakov. “Does data interpolation contradict statistical optimality?” *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 1611–1619.

[BT11] T. Bollerslev and V. Todorov. “Tails, fears, and risk premia”. *The Journal of Finance* 66.6 (2011), pp. 2165–2211.

[BT14] T. Bollerslev and V. Todorov. “Time-varying jump tails’”. en. *Journal of Econometrics* 183 (2014), 168–180.

[BTX15a] T. Bollerslev, V. Todorov, and L. Xu. “Tail risk premia and return predictability’”. es. *Journal of Financial Economics* 118 (2015), 113–134.

[BTX15b] T. Bollerslev, V. Todorov, and L. Xu. “Tail risk premia and return predictability”. *Journal of Financial Economics* 118.1 (2015), pp. 113–134.

[BTZ09] T. Bollerslev, G. Tauchen, and H. Zhou. “Expected stock returns and variance risk premia”. *The Review of Financial Studies* 22.11 (2009), pp. 4463–4492.

[Bub+23] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. “Sparks of artificial general intelligence: Early experiments with gpt-4”. *arXiv preprint arXiv:2303.12712* (2023).

[CCJY19] A. Chinco, A. D. Clark-Joseph, and M. Ye. “Sparse signals in the cross-section of returns”. *The Journal of Finance* 74.1 (2019), pp. 449–492.

[CCM97] M. Carhart, P. Carr, and D. Madan. “On persistence in mutual fund performance’”. en. *The Journal of Finance* 52.1 (1997). Towards the theory of volatility trading, in R. Jarrow, ed., ‘Risk book on, 57–82.

[CDG13] J. Conrad, R. Dittmar, and E. Ghysels. “Ex ante skewness and expected stock returns’”. en. *The Journal of Finance* 68.1 (2013), 85–124.

[CDW14] P. G. Constantine, E. Dow, and Q. Wang. “Active subspace methods in theory and practice: applications to kriging surfaces”. *SIAM Journal on Scientific Computing* 36.4 (2014), A1500–A1524.

[Cho+20] T. Chordia, A. Kurov, D. Muravyev, and A. Subrahmanyam. *Index option trading activity and market returns’, Management Science forthcoming*. en. 2020.

[COB19] L. Chizat, E. Oyallon, and F. Bach. “On lazy training in differentiable programming”. *Advances in Neural Information Processing Systems* 32 (2019).

[Coc09] J. H. Cochrane. *Asset pricing: Revised edition*. Princeton university press, 2009.

[Coc11] J. H. Cochrane. “Presidential address: Discount rates”. *The Journal of Finance* 66.4 (2011), pp. 1047–1108.

[CPZ19] L. Chen, M. Pelger, and J. Zhu. “Deep learning in asset pricing”. *arXiv preprint arXiv:1904.00745* (2019).

[CRW21] Q. Chen, N. Roussanov, and X. Wang. “Semiparametric Conditional Factor Models: Estimation and Inference”. *arXiv preprint arXiv:2112.07121* (2021).

[CS09] Y. Cho and L. Saul. “Kernel methods for deep learning”. *Advances in neural information processing systems* 22 (2009).

[CSKS23] J. Crego, J. Soerlie Kvaerner, and M. Stam. “Machine Learning and Expected Returns”. *Available at SSRN 4345646* (2023).

[CW10] M. Cremers and D. Weinbaum. “Deviations from Put-Call Parity and Stock Return Predictability”. *The Journal of Financial and Quantitative Analysis* 45.2 (2010), pp. 335–367. ISSN: 00221090, 17566916. URL: http://www.jstor.org/stable/27801488 (visited on 04/30/2023).

[Cyb89] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[Da02] G. D' avolio. "The market for borrowing stock". *Journal of financial economics* 66.2-3 (2002), pp. 271–306.

[DA05] P. Dayan and L. F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems.* MIT press, 2005.

[Dai+14] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. "Scalable kernel methods via doubly stochastic gradients". *Advances in neural information processing systems* 27 (2014).

[Dan17] A. Daniely. "SGD learns the conjugate kernel class of the network". *Advances in Neural Information Processing Systems* 30 (2017).

[Dav11] M. H. Davis. "The Dupire formula". *Imperial College London, Finite Difference Methods Course material* (2011).

[Dav75] L. S. Davis. "A survey of edge detection techniques". *Computer graphics and image processing* 4.3 (1975), pp. 248–270.

[DBG20] I. Dew-Becker and S. Giglio. en. Cross-sectional uncertainty and the business cycle: evidence from 40 years. 2020.

[DeM+20] V. DeMiguel, A. Martin-Utrera, F. J. Nogales, and R. Uppal. "A transaction-cost perspective on the multitude of firm characteristics". *The Review of Financial Studies* 33.5 (2020), pp. 2180–2222.

[DFS16] A. Daniely, R. Frostig, and Y. Singer. "Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity". *Advances in neural information processing systems* 29 (2016).

[Did+23] A. Didisheim, S. B. Ke, B. T. Kelly, and S. Malamud. *Complexity in factor pricing models.* Tech. rep. National Bureau of Economic Research, 2023.

[DJW22] J. Duarte, C. Jones, and H. Wang. "Very noisy option prices and inference regarding the volatility risk premium". en. Unpublished manuscript, 2022.

[DNMV23] A. Detzel, R. Novy-Marx, and M. Velikov. "Model comparison with transaction costs". *The Journal of Finance* (2023).

[dSB20] S. d'Ascoli, L. Sagun, and G. Biroli. "Triple descent and the two kinds of overfitting: Where & why do they appear?" *Advances in Neural Information Processing Systems* 33 (2020), pp. 3058–3069.

[Du+18] S. S. Du, X. Zhai, B. Poczos, and A. Singh. "Gradient descent provably optimizes over-parameterized neural networks". *arXiv preprint arXiv:1810.02054* (2018).

[Du+19a] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. "Gradient descent finds global minima of deep neural networks". *International conference on machine learning.* PMLR. 2019, pp. 1675–1685.

[Du+19b] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu. "Graph neural tangent kernel: Fusing graph neural networks with graph kernels". *Advances in neural information processing systems* 32 (2019).

[FD+14] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research* 15.1 (2014), pp. 3133–3181.

[Feu+19] B. Feunou, R. Aliouchkin, R. Tedongap, and L. Xu. "Loss uncertainty, gain uncertainty, and expected stock returns". es. Unpublished manuscript, 2019.

[FGX20] G. Feng, S. Giglio, and D. Xiu. "Taming the factor zoo: A test of new factors". *The Journal of Finance* 75.3 (2020), pp. 1327–1370.

[FHL19] S. Fort, H. Hu, and B. Lakshminarayanan. "Deep ensembles: A loss landscape perspective". *arXiv preprint arXiv:1912.02757* (2019).

[FJPO18] B. Feunou, M. Jahan-Parvar, and C. Okou. "Downside variance risk premium'". es. *Journal of Financial Econometrics* 16 (2018), 341–383.

[FNW20] J. Freyberger, A. Neuhierl, and M. Weber. “Dissecting characteristics nonparametrically”. *The Review of Financial Studies* 33.5 (2020), pp. 2326–2377.

[FP14] A. Frazzini and L. H. Pedersen. “Betting against beta”. *Journal of Financial Economics* 111.1 (2014), pp. 1 –25. ISSN: 0304-405X. DOI: https://doi.org/10.1016/j.jfineco.2013.10.005. URL: http://www.sciencedirect.com/science/article/pii/S0304405X13002675.

[GARA18] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. “Deep Convolutional Networks as shallow Gaussian Processes”. *International Conference on Learning Representations*. 2018.

[GB10] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. en. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, 249–256.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GKP16] S. Giglio, B. Kelly, and S. Pruitt. “Systemic risk and the macroeconomy: An empirical evaluation”. *Journal of Financial Economics* 119.3 (2016), pp. 457–471.

[GKX20a] S. Gu, B. Kelly, and D. Xiu. “Empirical asset pricing via machine learning’”. en. *The Review of Financial Studies* 33.5 (2020), 2223–2273.

[GKX20b] S. Gu, B. Kelly, and D. Xiu. “Empirical asset pricing via machine learning”. *The Review of Financial Studies* 33.5 (2020), pp. 2223–2273.

[GKX22] S. Giglio, B. Kelly, and D. Xiu. “Factor models, machine learning, and asset pricing”. *Annual Review of Financial Economics* 14 (2022).

[GLP21] D. Giannone, M. Lenza, and G. E. Primiceri. “Economic predictions with big data: The illusion of sparsity”. *Econometrica* 89.5 (2021), pp. 2409–2437.

[GMT23] S. Glebkin, S. Malamud, and A. Teguia. “Illiquidity and Higher Cumulants”. *The Review of Financial Studies* 36.5 (2023), pp. 2131–2173.

[GOPZ21] J. Guijarro-Ordonez, M. Pelger, and G. Zanotti. “Deep learning statistical arbitrage”. *arXiv preprint arXiv:2106.04028* (2021).

[GX21] S. Giglio and D. Xiu. “Asset pricing with omitted factors”. *Journal of Political Economy* 129.7 (2021), pp. 1947–1990.

[Han+19] Y. Han, A. He, D. Rapach, and G. Zhou. “Expected stock returns and firm characteristics: E-LASSO, assessment, and implications”. *SSRN* (2019).

[Has+19] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. “Surprises in high-dimensional ridgeless least squares interpolation”. *arXiv preprint arXiv:1903.08560* (2019).

[He+15] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[HJ15] T. Hazan and T. Jaakkola. “Steps toward deep kernel methods from infinite neural networks”. *arXiv preprint arXiv:1508.05133* (2015).

[HLT20a] Y. Han, F. Liu, and X. Tang. *The information content of the implied volatility surface: Can option prices predict jumps?’* en. Working Paper, 2020.

[HLT20b] Y. Han, F. Liu, and X. Tang. “The Information Content of The Implied Volatility Surface: Can Option Prices Predict Jumps?” *Available at SSRN 3454330* (2020).

[HLZ16] C. R. Harvey, Y. Liu, and H. Zhu. “... and the cross-section of expected returns”. *The Review of Financial Studies* 29.1 (2016), pp. 5–68.

[HS00] C. Harvey and A. Siddique. “Conditional skewness in asset pricing tests”. en. *Journal of Finance* 55 (2000), 1263– 1295.

[HS97] S. Hochreiter and J. Schmidhuber. "Long short-term memory". *Neural computation* 9.8 (1997), pp. 1735–1780.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". *Neural networks* 2.5 (1989), pp. 359–366.

[HXZ20] K. Hou, C. Xue, and L. Zhang. "Replicating anomalies". *The Review of Financial Studies* 33.5 (2020), pp. 2019–2133.

[IS15a] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[IS15b] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *32nd International Conference on Machine Learning, ICML 2015* 1 (Feb. 2015), pp. 448–456. DOI: 10.48550/arxiv.1502.03167. URL: https://arxiv.org/abs/1502.03167v3.

[ISV21] E. Ilhan, Z. Sautner, and G. Vilkov. "Carbon tail risk". *The Review of Financial Studies* 34.3 (2021), pp. 1540–1571.

[Jen+22] T. I. Jensen, B. T. Kelly, S. Malamud, and L. H. Pedersen. "Machine learning and the implementable efficient frontier". *Available at SSRN 4187217* (2022).

[JGH18] A. Jacot, F. Gabriel, and C. Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". *Advances in neural information processing systems* 31 (2018).

[Jia+20] L. Jiang, K. Wu, G. Zhou, and Y. Zhu. "Stock return asymmetry: Beyond skewness". io. *Journal of Financial and Quantitative Analysis* 55.2 (2020), 357–386.

[JKDX22] J. Jiang, B. Kelly, and N. Dacheng Xiu. *(Re-)Imag(in)ing Price Trends*. Tech. rep. 2022. DOI: http://dx.doi.org/10.2139/ssrn.3756587. URL: https://ssrn.com/abstract=3756587.

[JKP22] T. I. Jensen, B. T. Kelly, and L. H. Pedersen. "Is there a replication crisis in finance?" en. *Journal of Finance* (2022).

[JKPrt] T. I. Jensen, B. T. Kelly, and L. H. Pedersen. *Is there a replication crisis in finance?* Tech. rep. Journal of Finance, Forthcoming.

[JLP19] C. S. Jensen, D. Lando, and L. H. Pedersen. "Generalized recovery". *Journal of Financial Economics* 133.1 (2019), pp. 154–174.

[JM20] J. C. Jackwerth and M. Menner. "Does the Ross recovery theorem work empirically?" *Journal of Financial Economics* 137.3 (2020), pp. 723–739.

[JMW18] C. Jones, H. Mo, and H. Wang. "Do option prices forecast aggregate stock returns?" en. Unpublished manuscript, 2018.

[JS12] T. Johnson and E. So. "The option to stock volume ratio and future returns". en. *Journal of Financial Economics* 106.2 (2012), 262–286.

[Jum+21] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. "Highly accurate protein structure prediction with AlphaFold". *Nature* 596.7873 (2021), pp. 583–589.

[KB14] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014).

[Kel+23] B. T. Kelly, B. Kuznetsov, S. Malamud, and T. A. Xu. "Deep Learning from Implied Volatility Surfaces". *Swiss Finance Institute Research Paper* 23-60 (2023).

[Kel+24] B. Kelly, B. Kuznetsov, S. Malamud, and T. A. Xu. "Large (and Deep) Factor Models". *arXiv preprint arXiv:2402.06635* (2024).

[KH+09] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images" (2009).

[KJ14] B. Kelly and H. Jiang. "Tail risk and asset prices". *The Review of Financial Studies* 27.10 (2014), pp. 2841–2871.

[KLVN16] B. Kelly, H. Lustig, and S. Van Nieuwerburgh. "Too-systemic-to-fail: What option markets imply about sector-wide government guarantees". *American Economic Review* 106.6 (2016), pp. 1278–1319.

[KMZ22] B. T. Kelly, S. Malamud, and K. Zhou. "The virtue of complexity everywhere". *Available at SSRN 4166368* (2022).

[KMZ24] B. Kelly, S. Malamud, and K. Zhou. "The virtue of complexity in return prediction". *The Journal of Finance* 79.1 (2024), pp. 459–503.

[KNS13] R. Kozhan, A. Neuberger, and P. Schneider. "The skew risk premium in the equity index option market'". en. *Review of Financial Studies* 26 (2013), 2174–2203.

[KNS18] S. Kozak, S. Nagel, and S. Santosh. "Interpreting Factor Models". *The Journal of Finance* 73.3 (2018), pp. 1183–1223. DOI: 10.1111/jofi.12612. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jofi.12612. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/jofi.12612.

[KNS20] S. Kozak, S. Nagel, and S. Santosh. "Shrinking the cross-section". *Journal of Financial Economics* 135.2 (2020), pp. 271–292. ISSN: 0304-405X. DOI: https://doi.org/10.1016/j.jfineco.2019.06.008. URL: http://www.sciencedirect.com/science/article/pii/S0304405X19301655.

[KP13] B. Kelly and S. Pruitt. "Market expectations in the cross-section of present values". *The Journal of Finance* 68.5 (2013), pp. 1721–1756.

[KP15] B. Kelly and S. Pruitt. "The three-pass regression filter: A new approach to forecasting using many predictors". *Journal of Econometrics* 186.2 (2015), pp. 294–316.

[KPS20] B. Kelly, S. Pruitt, and Y. Su. "Characteristics are Covariances: A Unified Model of Risk and Return". *Journal of Financial Economics* (2020).

[KPV16] B. Kelly, L. Pástor, and P. Veronesi. "The price of political uncertainty: Theory and evidence from the option market". *The Journal of Finance* 71.5 (2016), pp. 2417–2480.

[KS19] M. Kilic and I. Shaliastovich. "Good and bad variance premia and expected returns". fr. *Management Science* 65 (2019), 2522–2544.

[KT20] O. Kadan and X. Tang. "A bound on expected stock returns". en. *Review of Financial Studies* 33 (2020), 1565–1617.

[KX+23] B. Kelly, D. Xiu, et al. "Financial machine learning". *Foundations and Trends® in Finance* 13.3-4 (2023), pp. 205–363.

[LCY13] M. Lin, Q. Chen, and S. Yan. "Network in network". *arXiv preprint arXiv:1312.4400* (2013).

[LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lee+18] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. "Deep Neural Networks as Gaussian Processes". *International Conference on Learning Representations*. 2018.

[Lee+19] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. "Wide neural networks of any depth evolve as linear models under gradient descent". *Advances in neural information processing systems* 32 (2019), pp. 8572–8583.

[Lee+20] J. Lee, S. Schoenholz, J. Pennington, B. Adlam, L. Xiao, R. Novak, and J. Sohl-Dickstein. "Finite versus infinite neural networks: an empirical study". *Advances in Neural Information Processing Systems* 33 (2020), pp. 15156–15172.

[Leo17] G. Leoni. *A first course in Sobolev spaces.* American Mathematical Soc., 2017.

[Li+18] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. "Visualizing the loss landscape of neural nets". *Advances in neural information processing systems* 31 (2018).

[Li+19] Z. Li, R. Wang, D. Yu, S. S. Du, W. Hu, R. Salakhutdinov, and S. Arora. "Enhanced convolutional neural tangent kernels". *arXiv preprint arXiv:1911.00809* (2019).

[LJB20] D. LeJeune, H. Javadi, and R. Baraniuk. "The implicit regularization of ordinary least squares ensembles". *International Conference on Artificial Intelligence and Statistics.* PMLR. 2020, pp. 3525–3535.

[LL17] S. M. Lundberg and S.-I. Lee. "A unified approach to interpreting model predictions". *Advances in neural information processing systems* 30 (2017).

[LP20] M. Lettau and M. Pelger. "Factors that fit the time series and cross-section of stock returns". *The Review of Financial Studies* 33.5 (2020), pp. 2274–2325.

[LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". *Advances in neural information processing systems* 30 (2017).

[LRB07] N. Le Roux and Y. Bengio. "Continuous neural networks". *Artificial Intelligence and Statistics.* PMLR. 2007, pp. 404–411.

[LT19] H. Lin and V. Todorov. "Aggregate asymmetry in idiosyncratic jump risk". en. Unpublished manuscript, 2019.

[LWZ22] M. Leippold, Q. Wang, and W. Zhou. "Machine learning in the Chinese stock market". *Journal of Financial Economics* 145.2 (2022), pp. 64–82.

[LZZ20] Y. Liu, G. Zhou, and Y. Zhu. "Maximizing the Sharpe ratio: A genetic programming approach". *Available at SSRN 3726609* (2020).

[Mar17] I. Martin. "What is the Expected Return on the Market?" *The Quarterly Journal of Economics* 132.1 (2017), pp. 367–433.

[Mat+18] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. "Gaussian process behaviour in wide deep neural networks". *arXiv preprint arXiv:1804.11271* (2018).

[MB17] S. Ma and M. Belkin. "Diving into the shallows: a computational perspective on large-scale shallow learning". *Advances in neural information processing systems* 30 (2017).

[Mer+23] A. Merchant, S. Batzner, S. S. Schoenholz, M. Aykol, G. Cheon, and E. D. Cubuk. "Scaling deep learning for materials discovery". *Nature* 624.7990 (2023), pp. 80–85.

[MP16] R. D. McLean and J. Pontiff. "Does academic research destroy stock return predictability?" *The Journal of Finance* 71.1 (2016), pp. 5–32.

[MPP22] D. Muravyev, N. D. Pearson, and J. M. Pollet. "Anomalies and Their Short-Sale Costs". *Available at SSRN 4266059* (2022).

[MW19] I. W. Martin and C. Wagner. "What is the Expected Return on a Stock?" *The Journal of Finance* 74.4 (2019), pp. 1887–1929.

[MZ16] B. Moritz and T. Zimmermann. "Tree-based conditional portfolio sorts: The relation between past and future stock returns". *Available at SSRN 2740751* (2016).

[Nak+21] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. "Deep double descent: Where bigger models and more data hurt". *Journal of Statistical Mechanics: Theory and Experiment* 2021.12 (2021), p. 124003.

[Nea96] R. M. Neal. "Priors for infinite networks". *Bayesian Learning for Neural Networks.* Springer, 1996, pp. 29–53.

[Neu+21] A. Neuhierl, X. Tang, R. T. Varneskov, and G. Zhou. "Stock Option Predictability for the Cross-Section". *Available at SSRN 3795486* (2021).

[Neu+22] A. Neuhierl, X. Tang, R. T. Varneskov, and G. Zhou. "Option characteristics as cross-sectional predictors". *SSRN Electronic Journal* June 9, 2022 (Mar. 2022). DOI: 10.2139/SSRN.3795486. URL: http://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/65244.

[Nov+18] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. "Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes". *International Conference on Learning Representations*. 2018.

[Nov+19] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz. "Neural tangents: Fast and easy infinite neural networks in python". *arXiv preprint arXiv:1912.02803* (2019).

[OST20] P. Orlowski, P. Schneider, and F. Trojani. "On the nature of jump risk premia". fr. Unpublished manuscript, Swiss finance institute. 2020.

[OWB18] M. Olson, A. Wyner, and R. Berk. "Modern neural networks generalize on small data sets". *Advances in Neural Information Processing Systems* 31 (2018).

[Ped20] P. Pederzoli. "Skewness swaps on individual stocks". en. Unpublished manuscript, 2020.

[PHD20] V. Papyan, X. Han, and D. L. Donoho. "Prevalence of neural collapse during the terminal phase of deep learning training". *Proceedings of the National Academy of Sciences* 117.40 (2020), pp. 24652–24663.

[Pic75] J. Pickands. "Statistical inference using extreme order statistics". *the Annals of Statistics* 3.1 (1975), pp. 119–131.

[Pin99] A. Pinkus. "Approximation theory of the MLP model". *Acta Numerica 1999: Volume 8* 8 (1999), pp. 143–195.

[Poo+16] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. "Exponential expressivity in deep neural networks through transient chaos". *Advances in neural information processing systems* 29 (2016).

[Pow+22] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. "Grokking: Generalization beyond overfitting on small algorithmic datasets". *arXiv preprint arXiv:2201.02177* (2022).

[PSG17] J. Pennington, S. Schoenholz, and S. Ganguli. "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice". *Advances in neural information processing systems* 30 (2017).

[QL16] L. Qin and V. Linetsky. "Positive eigenfunctions of markovian pricing operators: Hansen-scheinkman factorization, ross recovery, and long-term pricing". *Operations Research* 64.1 (2016), pp. 99–117.

[Rad+22] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin. "Feature learning in neural networks and kernel machines that recursively learn features". *arXiv preprint arXiv:2212.13881* (2022).

[Ram+21] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. "Zero-shot text-to-image generation". *International conference on machine learning*. Pmlr. 2021, pp. 8821–8831.

[Rom+22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. "High-resolution image synthesis with latent diffusion models". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[Ros15a] S. Ross. "The recovery theorem'". en. *The Journal of Finance* 70.2 (2015), 615–648.

[Ros15b] S. Ross. "The recovery theorem". *The Journal of Finance* 70.2 (2015), pp. 615–648.

[RR07] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines." *NIPS*. Vol. 3. 4. Citeseer. 2007, p. 5.

[RSZ10] D. E. Rapach, J. K. Strauss, and G. Zhou. "Out-of-sample equity premium prediction: Combination forecasts and links to the real economy". *The Review of Financial Studies* 23.2 (2010), pp. 821–862.

[SC16] D. Soudry and Y. Carmon. "No bad local minima: Data independent training error guarantees for multilayer neural networks". *arXiv preprint arXiv:1605.08361* (2016).

[SCC18] U. Shaham, A. Cloninger, and R. R. Coifman. "Provable approximation properties for deep neural networks". *Applied and Computational Harmonic Analysis* 44.3 (2018), pp. 537–557.

[Sch+20] P. Schneider, C. Wagner, J. Zechner, and W. Sharpe. "Low-risk anomalies?"" en. *The Journal of Finance* 75.5 (2020), 2673–2718.

[Sha+20] V. Shankar, A. Fang, W. Guo, S. Fridovich-Keil, J. Ragan-Kelley, L. Schmidt, and B. Recht. "Neural kernels without tangents". *International Conference on Machine Learning*. PMLR. 2020, pp. 8614–8623.

[Sil+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". *nature* 529.7587 (2016), pp. 484–489.

[Sim+23] J. B. Simon, D. Karkada, N. Ghosh, and M. Belkin. "More is better in modern machine learning: when infinite overparameterization is optimal and overfitting is obligatory". *arXiv preprint arXiv:2311.14646* (2023).

[SKP17] P. Stilger, A. Kostakis, and S.-H. Poon. "What does risk-neutral skewness tell us about future stock returns?"" fr. *Management Science* 63 (2017), 1814–1834.

[SMG13] A. M. Saxe, J. L. McClelland, and S. Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". *arXiv preprint arXiv:1312.6120* (2013).

[Spi+19] S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart. "A jamming transition from under-to over-parametrization affects generalization in deep learning". *Journal of Physics A: Mathematical and Theoretical* 52.47 (2019), p. 474001.

[ST19] P. Schneider and F. Trojani. "Divergence and the price of uncertainty"". en. *Journal of Financial Econometrics* 17 (2019), 341–396.

[Tan19] X. Tang. "Variance asymmetry managed portfolios". en. Unpublished manuscript, 2019.

[Tao23] T. Tao. *Topics in random matrix theory.* Vol. 132. American Mathematical Society, 2023.

[TB20] A. Tsigler and P. L. Bartlett. *Benign overfitting in ridge regression.* 2020. arXiv: 2009.14286 [math.ST].

[Tea+23] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. "Gemini: a family of highly capable multimodal models". *arXiv preprint arXiv:2312.11805* (2023).

[Tou+23] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. "Llama: Open and efficient foundation language models". *arXiv preprint arXiv:2302.13971* (2023).

[VOZ16] M. R. Van Oordt and C. Zhou. "Systematic tail risk". *Journal of Financial and Quantitative Analysis* (2016), pp. 685–705.

[Wal17] J. Walden. "Recovery with unbounded diffusion processes". *Review of Finance* 21.4 (2017), pp. 1403–1444.

[WG08] I. Welch and A. Goyal. "A comprehensive look at the empirical performance of equity premium prediction". *The Review of Financial Studies* 21.4 (2008), pp. 1455–1508.

[Wil97] C. K. Williams. "Computing with infinite networks". *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*. Vol. 9. MIT Press. 1997, p. 295.

[XKM23] T. A. Xu, B. Kelly, and S. Malamud. "A Simple Algorithm For Scaling Up Kernel Methods". *arXiv preprint arXiv:2301.11414* (2023).

[Xu23] T. A. Xu. "Tail Recovery". *Available at SSRN 4380818* (2023).

[XZZ10] Y. Xing, X. Zhang, and R. Zhao. "What does the individual option volatility smirk tell us about future equity returns?'" en. *Journal of Financial and Quantitative Analysis* (2010), 641–662.

[Yan11] Y. Yan. "Jump risk, stock returns and slope of implied volatility smile'". en. *Journal of Financial Economics* 99 (2011), 216–223.

[Yan19] G. Yang. "Wide feedforward or recurrent neural networks of any architecture are gaussian processes". *Advances in Neural Information Processing Systems* 32 (2019).

[Yan20a] G. Yang. "Tensor programs ii: Neural tangent kernel for any architecture". *arXiv preprint arXiv:2006.14548* (2020).

[Yan20b] G. Yang. "Tensor programs iii: Neural matrix laws". *arXiv preprint arXiv:2009.10685* (2020).

[Yan+21] G. Yang, E. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. "Tuning large neural networks via zero-shot hyperparameter transfer". *Advances in Neural Information Processing Systems* 34 (2021), pp. 17084–17097.

[Yan+22] G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. "Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer". *arXiv preprint arXiv:2203.03466* (2022).

[Yan+23] G. Yang, D. Yu, C. Zhu, and S. Hayou. "Feature Learning in Infinite Depth Neural Networks". *The Twelfth International Conference on Learning Representations*. 2023.

[Yao+20] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney. "Pyhessian: Neural networks through the lens of the hessian". *2020 IEEE international conference on big data (Big data)*. IEEE. 2020, pp. 581–590.

[YH20] G. Yang and E. J. Hu. "Feature learning in infinite-width neural networks". *arXiv preprint arXiv:2011.14522* (2020).

[YL21] G. Yang and E. Littwin. "Tensor programs iib: Architectural universality of neural tangent kernel training dynamics". *International Conference on Machine Learning*. PMLR. 2021, pp. 11762–11772.

[YL23] G. Yang and E. Littwin. "Tensor programs ivb: Adaptive optimization in the infinite-width limit". *arXiv preprint arXiv:2308.01814* (2023).

[Zan+21] A. Zandieh, I. Han, H. Avron, N. Shoham, C. Kim, and J. Shin. "Scaling Neural Tangent Kernels via Sketching and Random Features". *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 1062–1073. URL: https://proceedings.neurips.cc/paper/2021/file/08ae6a26b7cb089ea588e94aed36bd15-Paper.pdf.

[Zha+16] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". *arXiv preprint arXiv:1611.03530* (2016).