EPFL

Thèse n° 9061

Improving the Training of Compact Neural Networks for Visual Recognition

Présentée le 9 décembre 2022

Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Shuxuan GUO

Acceptée sur proposition du jury

Prof. A. Bosselut, président du jury
Prof. P. Fua, Dr M. Salzmann, directeurs de thèse
Dr H. Jegou, rapporteur
Prof. D. Alistarh, rapporteur
Prof. M. Jaggi, rapporteur

École polytechnique fédérale de Lausanne

2022

Less is more.
– Robert Browning, 1855

To my family and all people I love.

Acknowledgements

It has been a fantastic and treasured journey of my PhD study at EPFL with tons of extraordinary memories and many great people, who have made me feel very grateful and happy along the way over the past five and a half years.

First and foremost, I would like to express my deepest gratitude to my supervisors, Pascal Fua, Mathieu Salzmann and Jose M. Alvarez. Thank you for giving me the opportunity to do my PhD work at CVLab. Mathieu has been an extremely exceptional advisor I have ever met. I feel so fortunate to have his professional guidance during my PhD study. Retrospectively, I have learned from Mathieu how to discover, define and solve a problem. Mathieu has given me very visionary advice, most generous support, and incredibly constructive feedback. Mathieu's research foresight and valuable mentoring to the students has influenced my doctoral study and will also affect me in my future work and life. I've worked with Jose since the beginning of my PhD journey, and we've held weekly meetings through my whole study. Thanks you Jose for teaching me how to make my first academic poster and connect with people in Vancouver. And also thank you for offering me an internship at NVIDIA, which was so unforgettable. I sincerely thank Jose for the insightful and constructive discussions, the ambitious encouragement along my PhD study and the generous GPU support during my internship at NVIDIA.

Furthermore, I would like to thank the members of my thesis committee, Hervé Jegou, Dan Alistarh, Martin Jaggi and the president of the jury, Antoine Bosselut, for their interests of my work, their valuable comments, and insightful questions.

The journey would not be undertaken without many great lab mates in CVLab. I would like to thank all the current and former members of CVLab for the warm and friendly atmosphere that has inspired me throughout my PhD life. To Isinsu and Weizhe, thank you for sharing your office with me, as well as your special delicious food and fun ideas. To Sena, Jan and Krishna, thank you for helping me out with the intensive course projects and assignments. To Leo, thank you for translating my French paperwork. To Jan and Helge, thank you for helping me achieve my first milestone of Via Ferrata. A huge thank you to Kaicheng, who has greatly inspired me both in research and in life. To Vidit, Yinlin, Krzysztof, Edoardo, Udaranga, Semih, Andrii, Agata, Bugra, Joachim, Mat, Chen, Wei and many others, thank you all for making the CVLab an unforgettable place. I would also like to give a thank you to Ariane, Patricia and Angela for their generous administrative support and kindness.

Many thanks also go to my dear friends in Lausanne and in China. In particular, I would like to thank Hui Chen, Yao Chen, Miao Hua, Ling Zhou, Shuhui who have created so many unforgettable memories in the cities, in the mountains and by the lakes all over Switzerland. I enjoyed every lunch and coffee break with you at nearly every location on the EPFL campus. We shared so many interesting stories and experiences together regardless of our different academic backgrounds. To Tao Lin, Zhaowen, Yugang, Yao Di, Yanfei, Jingyue, Lingjing, Ruofan, Chloee and all other folks, thank you for enriching and coloring my life outside the lab. Thanks also to Xinyue and Xiaoyu whom I have known since my childhood. They have supported me over ten years and they can always cheer me up even away from thousands of miles and hours of time difference.

Last but not least, I would like to give my special thanks to my loving family, to whom the thesis is dedicated. None of my endeavors would ever be possible without their infinite love and unconditional support. I love you.

Beijing, November 3, 2022 Shuxuan

Abstract

During the Artificial Intelligence (AI) revolution of the past decades, deep neural networks have been widely used and have achieved tremendous success in visual recognition. Unfortunately, deploying deep models is challenging because of their huge model size and computational complexity. Therefore, compact neural networks of small model size have been remarkably demanded for embedded/mobile/edge devices, which are omnipresent in our modern AI age.

The main goal of this thesis is to improve the training of *arbitrary, given* compact networks. To achieve this, we introduce several methods, including linear over-parameterization and two novel knowledge distillation, to facilitate the training of such compact models, and thus to improve their performance.

Over-parameterization was shown to be key to the success of conventional deep models, being essential to facilitate the optimization during training, even though not all the model weights are necessary at inference. Motivated by this observation, in this thesis, we firstly present a general optimization method, ExpandNets, leveraging linear over-parameterization to train a compact network from scratch. Specifically, we introduce two expansion strategies for convolutional layers and one for fully-connected layers by linearly expanding these linear operations into consecutive linear layers, without adding any nonlinearity. Our proposed linear expansion empirically improves the optimization behavior and generalization ability during network training. At test time, such an expanded network can be algebraically contracted back to the original compact network without any information loss, but yields better prediction performance. The effectiveness of ExpandNets is evidenced on several visual recognition tasks, including image classification, object detection, and semantic segmentation.

Our first knowledge distillation approach is for object detection, motivated by the fact that the recent knowledge distillation literature remains limited to the scenario where the student and the teacher tackle the same task, with similar network architectures. By contrast, we propose a classifier-to-detector knowledge distillation method for object detection, instead of the standard detector-to-detector distillation strategy. Our method improves the performance of the student detector on both classification and localization. In other words, our method successfully transfers the knowledge not only across architectures but also across tasks.

We then extend our knowledge distillation work to the task of 6D pose estimation, where

knowledge distillation has been completely unstudied. Specifically, we observe that, for this task, the keypoint-based models are less sensitive than the dense prediction ones to a decrease in the model size. We therefore introduce the first knowledge distillation method for 6D pose estimation by relying on optimal transport theory to align the keypoint distributions of student and teacher networks. Our experiments on several benchmarks show that our distillation method predicts better keypoints and yields state-of-the-art results with different compact student models.

To summarize, this thesis contributes to multiple investigations to improve the training phase of *arbitrary, given* compact networks for different visual recognition tasks. Our diverse strategies consistently improve the performance of the compact networks at inference time.

Keywords: compact neural networks, visual recognition, over-parameterization, knowledge distillation, classifier-to-detector, distribution alignment

Résumé

Au cours de la révolution de l'intelligence artificielle (IA) des dernières décennies, les réseaux de neurones profonds ont été largement utilisés et ont remporté un énorme succès pour la reconnaissance visuelle. Malheureusement, le déploiement de modèles profonds est difficile en raison de leur taille énorme et de leur complexité de calcul. Par conséquent, les réseaux de neurones compacts de petites tailles ont été remarquablement demandés pour les appareils embarqués/mobiles/de périphérie, qui sont omniprésents à notre ère moderne de l'IA.

L'objectif principal de cette thèse est d'améliorer l'apprentissage de réseaux compacts *arbitraire et donnés*. Pour y parvenir, nous introduisons plusieurs méthodes, dont la sur-paramétrisation linéaire et deux nouvelles distillations de connaissances, pour faciliter l'apprentissage de tels modèles compacts, et ainsi améliorer leurs performances lors de l'inférence.

La sur-paramétrisation s'est avérée être la clé du succès des modèles profonds conventionnels, étant essentielle pour faciliter l'optimisation leur de leur entraînement, même si tous les poids du modèle ne sont pas nécessaires lors de l'inférence. Motivés par cette observation, dans cette thèse, nous présentons d'abord une méthode d'optimisation générale, ExpandNets, tirant parti de la sur-paramétrisation linéaire pour former un réseau compact. Plus précisément, nous introduisons deux stratégies d'expansion pour les couches convolutionnelles et une pour les couches entièrement connectées en augmentant linéairement ces opérations en couches linéaires consécutives, sans ajouter de non-linéarité. Notre expansion linéaire améliore empiriquement le comportement de l'optimisation et la capacité de généralisation pendant l'entraînement du réseau. Lors de l'inférence, un tel réseau étendu peut être ramené algébriquement au réseau compact d'origine sans aucune perte d'informations, mais donne de meilleurs résultats. L'efficacité des ExpandNets est mise en évidence sur plusieurs tâches de reconnaissance visuelle, notamment la classification d'images, la détection d'objets et la segmentation sémantique.

Notre première approche de distillation de connaissances est pour la détection d'objets, motivée par le fait que la littérature récente sur la distillation des connaissances reste limitée au scénario où l'étudiant et l'enseignant s'attaquent à la même tâche, avec des architectures de réseau similaires. En revanche, nous proposons une méthode de distillation des connaissances d'un classificateur à un détecteur pour la détection d'objets, au lieu de s'appuyer sur la straté-

gie standard de distillation de détecteur à détecteur. Notre méthode améliore les performances du détecteur étudiant à la fois pour la classification et la localisation. En d'autres termes, notre méthode transfère avec succès les connaissances non seulement entre les architectures mais aussi entre les tâches.

Nous étendons ensuite notre travail de distillation des connaissances à la tâche d'estimation de pose 6D, où la distillation des connaissances n'a pas été étudiée. Plus précisément, nous observons que, pour cette tâche, les modèles basés sur les points clés sont moins sensibles que ceux de prédiction dense à une diminution de la taille du modèle. Nous introduisons donc la première méthode de distillation des connaissances pour l'estimation de pose 6D en nous appuyant sur la théorie du transport optimal pour aligner les distributions des points clés des réseaux étudiant et enseignant. Nos expériences sur plusieurs base de données montrent que notre méthode de distillation prédit mieux les points clés et donne des résultats de pointe avec différents modèles étudiants compacts.

Pour résumer, cette thèse contribue à de multiples aspects de l'amélioration de la phase d'apprentissage de réseaux compacts *arbitraire et donnés* pour différentes tâches de reconnaissance visuelle. Nos diverses stratégies améliorent constamment les performances des réseaux compacts lors de l'inférence.

Mots-clés : réseaux de neurones compacts, reconnaissance visuelle, sur-paramétrisation, distillation des connaissances, classificateur-à-détecteur, alignement de distribution

Contents

List of Figures

List of Tables

1 Introduction

Visual recognition, or computer vision, is a fundamental research topic in computer science and one of the key application domains for artificial intelligence (AI). It takes digital visual signals as input, such as images and videos, and derives meaningful high-level information, aiming to achieve automatic visual observation, perception and understanding for computers and machines as the human visual system does.

Before the deep learning era, researchers spent several decades in designing sophisticated handcrafted descriptors, e.g., SIFT (Lowe, 1999, 2004) or HOG (Dalal and Triggs, 2005), and classifiers, e.g., SVM (Cortes and Vapnik, 1995), for different visual recognition tasks. These descriptors can usually easily be interpreted by humans. However, they tend to be highly task-specific, and not to generalize well to other tasks. With the rapidly growing availability of large-scale datasets and advanced computational resources, the community has shifted its attention to deep learning models. The main advantage of the deep learning algorithms compared to the old-fashion handcrafted methods is their ability to learn general, highly-expressive feature descriptors from large amounts of data. Such descriptors can nicely generalize to different tasks by fine-tuning the model weights. The tremendous progress and remarkable performance of deep neural networks on several benchmark visual recognition tasks, including image classification (Krizhevsky et al., 2012a; Simonyan and Zisserman, 2015; He et al., 2016; Dosovitskiy et al., 2021), object detection (Ren et al., 2015; Redmon et al., 2016; Kehl et al., 2017; He et al., 2017a), semantic segmentation (Long et al., 2015; Chen et al., 2014; Papandreou et al., 2015), and 6D pose estimation (Kendall et al., 2015; Hu et al., 2019; Di et al., 2021), make them widely used in many real-world applications, such as autonomous driving, robotics, virtual reality.

Over the past decade, to maximize their performance on given tasks, the neural networks has evolved from shallow ones to significantly deeper and wider ones, with over hundreds of layers and tens of millions of parameters. Let us consider the task of image classification as an example because it has been widely studied to benchmark modern deep learning architectures. In this context, the use of convolutional neural networks (CNNs) began with LeNet-5, which was designed by LeCun et al. (1998) in 1998. LeNet-5 has 60K parameters,

with only 2 convolutional layers (ConvLayers) and 2 fully-connected layers (FCLayers), and was designed for hand-written digits recognition, where the input images are gray-scale with a resolution of 32 × 32. The massive growth of CNNs' sizes essentially started in 2012, with the release of the large-scale ImageNet dataset (Russakovsky et al., 2015)[1] for visual recognition. The dataset contains over 1 million labeled color images for 1000 object classes, with a much higher resolution of 256 × 256[2]. The first successful CNN applied to this data was AlexNet (Krizhevsky et al., 2012b), with 61 million parameters in 5 ConvLayers and 3 FCLayers. AlexNet significantly increased the top-1 classification accuracy to 57.2% compared with previous handcrafted approaches. Subsequently, VGGNets (Simonyan and Zisserman, 2015) were proposed, stacking more 3 × 3 kernels, and thus resulting deeper models, such as the famous VGG16 with 13 ConvLayers and 3 FCLayers, resulting in over 138 million parameters, increasing the top-1 accuracy on ImageNet to 68.5%. This was followed by the idea of residual blocks with skip connection layers proposed by He et al. (2016), making possible the training of networks with over hundreds layers. For example, ResNet50 has 25.5 million parameters with 49 ConvLayers and 1 FCLayer, reaching a top-1 accuracy on ImageNet of 76.1%. Nowadays, the ResNet family remains one of the most popular architecture for visual recognition because it balances computational complexity and prediction performance. Multiple variants of ResNet have been developed in the following years, such as ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017), and SENet (Hu et al., 2018), further boosting the accuracy on ImageNet.

Ultimately, "Wider and deeper are better" has become the rule of thumb to develop neural network architectures. However, with the ever-increasing size of the state-of-the-art (SOTA) neural networks, the performance improvement comes at the significant cost of resources such as hardware storage, software memory, and computational energy. These drawbacks reduce the suitability and deployment of these deep neural network models for resources-constrained devices and embedded platforms. Furthermore, despite the growth of computational resources, the deeper models suffer from latency. There is therefore a clear need for compact models performing fast inference for real-time applications.

As a consequence, AI on the edge with efficient deep learning models (Howard et al., 2017; Iandola et al., 2016; Ma et al., 2018; Sandler et al., 2018) and algorithms (Alvarez and Salzmann, 2016, 2017; Reed, 1993; Hinton et al., 2015) has drawn increasing attention in recent years, particularly with the growing presence of mobile devices and applications. This is the area to which this thesis contributes. Specifically, we investigate several novel approaches to improve the training process of compact neural networks. We introduce both general algorithms for multiple visual recognition tasks and methods for specific tasks. In particular, throughout the thesis, we consider the following four visual recognition tasks, illustrated in Figure 1.1:

[1] Referring to ImageNet-1K if not specified.

[2] 224, 384 and even larger image sizes are also used in recent research.

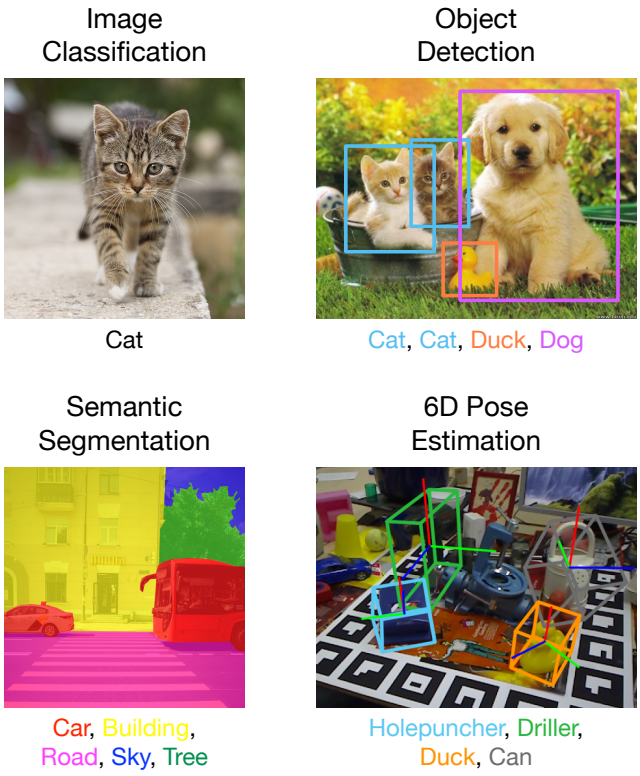


Figure 1.1: **Visual recognition tasks.** Image Classification takes an image depicting a single object as input and outputs the category of the object. Object Detection takes an image with multiple objects as input and outputs the categories and corresponding bounding boxes of all objects. Semantic Segmentation takes a single image as input and outputs the category of each pixel. 6D pose estimation takes an image with multiple objects as input and outputs their category and their 6D pose (3D translation and 3D rotation).

- Image classification, which recognizes the category of a single object;
- Object detection, which classifies and localizes multiple objects;
- Semantic segmentation, assigns a category to every image pixel;
- 6D pose estimation, which determines the category of multiple objects and estimates their 3D translation and 3D rotation w.r.t. the camera.

We have shown our work to improve the representations learned by compact neural networks, thus yielding better performance at inference time for these different visual recognition tasks.

1.1 Problem Definition

The compact and efficient deep learning models we study in this thesis are ones with fewer network layers, fewer parameters, fewer computational operations and faster inference speed.

Compared with conventional, gigantic deep learning models, they have the following technical and environmental benefits:

- **Hardware storage and software memory**. The over-parameterized and redundant nature of deep neural networks leads to huge model sizes, with millions or even billions of parameters, and high computational complexity, requiring considerable on-device hardware storage for the trained weights and powerful computational units for computing and processing the deep networks. In the resources-constrained scenario, these requirements are hard to satisfy. This makes storage- and memory-efficient compact models, with fewer parameters and computational operations, more suitable for edge platforms and mobile devices.
- **Inference speed**. The training and inference time of deep models are often long, which hinders their deployment for mobile applications. By contrast, compact models with fewer layers, fewer parameters and lower computational complexity are faster to train and make prediction with, thus making it possible to meet the real-time requirements of many tasks on edge devices.
- **Energy consumption**. Embedded platforms and mobile devices are usually also battery-constrained, and the applications should thus be energy saving. Compact models consume less energy compared to deeper ones, which enhances their suitability for deployment on such energy-constrained devices. Moreover, deeper models have an important carbon footprint on the environment (Hao, 2019), and compact models can therefore contribute to reducing this carbon impact.

Several lines of research have been proposed to produce efficient networks and improve the performance of given compact models, and the field is still evolving at an incredible pace. These approaches roughly fall into three categories: Efficient network architecture design, network compression, including network pruning, network quantization and matrix factorization, and knowledge distillation, aiming to train an arbitrary student model from a deeper teacher network. Below, we briefly discuss these three approaches.

Efficient network architecture design proposes new network architectures specifically optimized for mobile devices with some well-designed compact modules. This is illustrated in Figure 1.2 with methods such as MobileNet (Howard et al., 2017) with depthwise separable convolutions, MobileNetV2 (Sandler et al., 2018) with linear bottleneck and inverted residual block, ShuffleNet (Ma et al., 2018) with channel shuffling, and SqueezeNet (Iandola et al., 2016) with squeeze and expand fire module. Moreover, with the development of Neural Architecture Search (NAS), compact and efficient networks, such as MobileNetV3 (Howard et al., 2019), MnasNet (Tan et al., 2019), can be searched automatically. These compact and efficient networks can be trained from scratch while preserving the accuracy and speeding up inference with lightweight architectures.

Network compression usually starts from a pretrained deep neural network and reduces the

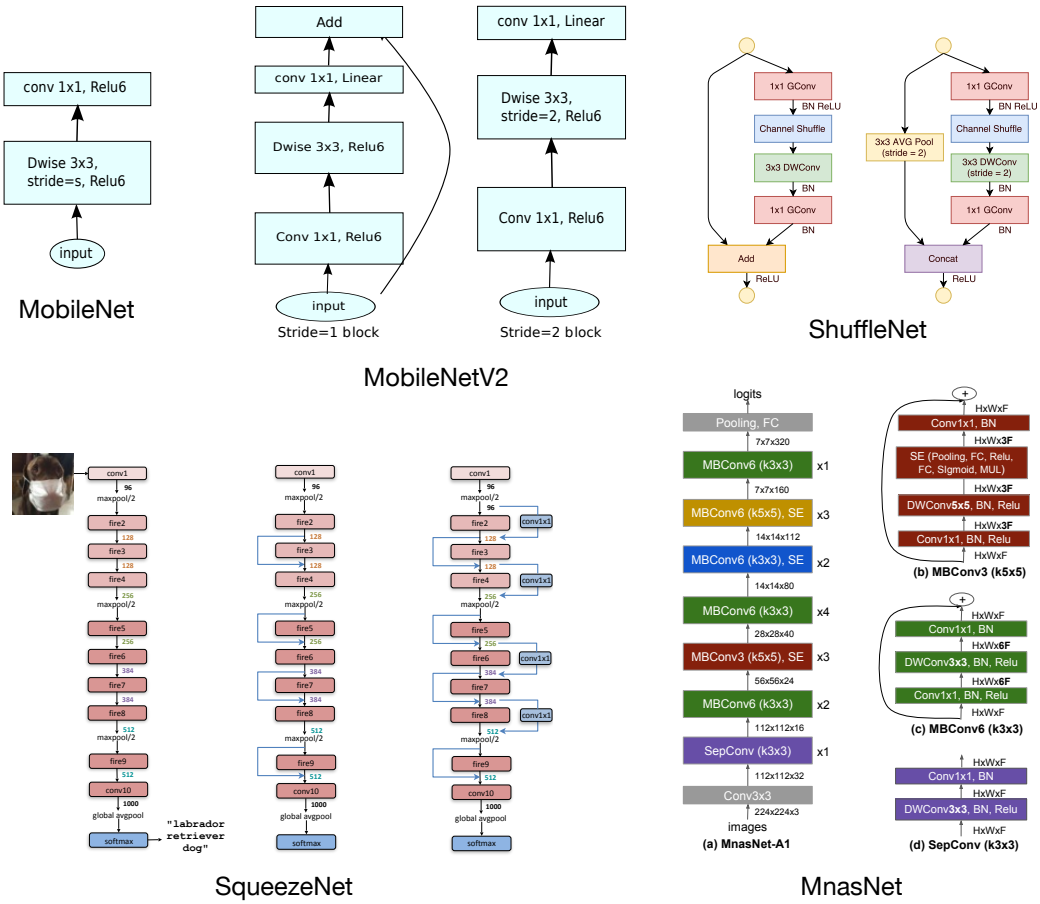


Figure 1.2: **Efficient network architectures** are designed for the edge mobile devices. E.g., MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018), ShuffleNet (Ma et al., 2018), SqueezeNet (Iandola et al., 2016) and MnasNet (Tan et al., 2019).

model size and computational complexity to produce a compact model, as illustrated in Figure 1.3. We discuss some existing strategies below.

Network pruning (Reed, 1993; Han et al., 2016; Molchanov et al., 2017; Liu et al., 2018) is one of the most popular techniques, which evaluates and cuts off the unimportant, redundant, or unnecessary parts of the deep neural network, such as connections or weights (Han et al., 2015; Guo et al., 2016; Lee et al., 2019), neurons (Hu et al., 2016), filters (He et al., 2019b), channels (He et al., 2017b; Zhuang et al., 2018) or layers (Wen et al., 2016; Lemaire et al., 2019). Connections or weights pruning leads to irregular kernels and unstructured architectures, which are hard to exploit in the current deep learning frameworks and thus hard to accelerate. By contrast, neurons, filters, channels or layers pruning removes entire units of the network, producing a structured pruned model with fewer computational units and a smaller model size. These pruning methods achieve real model size reduction and speed acceleration.

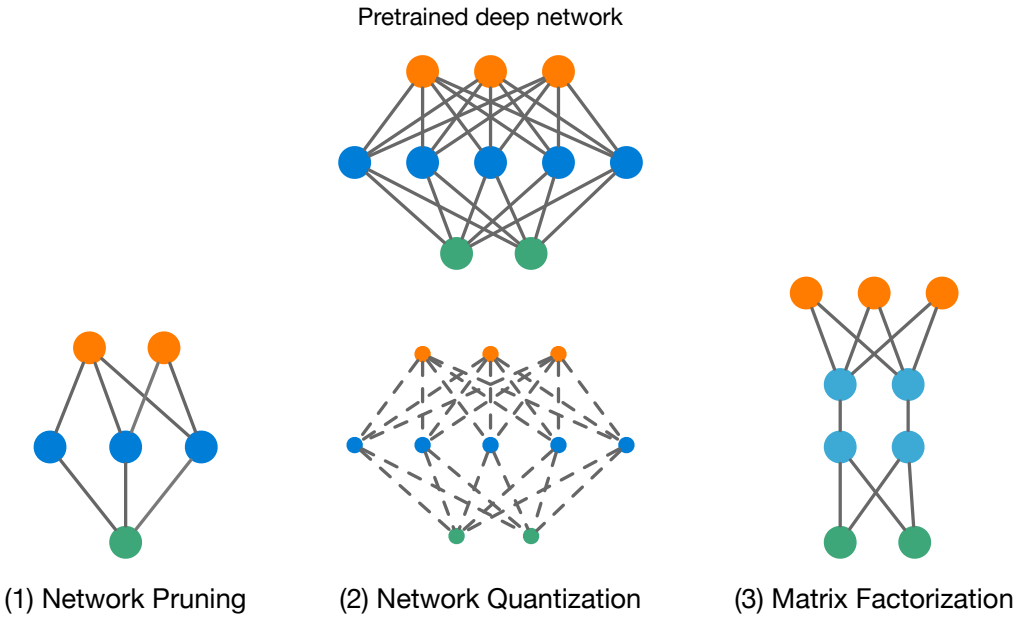


Figure 1.3: **Network compression.** Starting from a pretrained deep network, (1) Network pruning removes the unimportant weights, connections, neurons, channels and layers; (2) Network quantization represents the weights and outputs of the network with lower precision, shown with dotted connections and smaller nodes; (3) Matrix factorization decomposes the original layers into several (two in the figure) smaller matrices to achieve the approximation.

Network quantization (Cai et al., 2017; Li et al., 2017; Lin et al., 2016; Wu et al., 2018) aims to represent the parameters in a neural network with lower precision or even binary representations (Rastegari et al., 2016; Courbariaux et al., 2016). Typically, deep networks are trained and inferred with floating point (e.g. 32-bit) precision, which comes with a significant cost in computation, memory, and storage requirements. With quantization, the precision of floating point numbers is reduced from 32-bit to much lower bit width, and in the extreme case to binary and ternary states, or even a single bit with the vector quantization (Stock et al., 2020). Thus, network quantization not only reduces the model size but also accelerates the inference of the resulting neural network.

Matrix factorization (Rigamonti et al., 2013; Jaderberg et al., 2014; Novikov et al., 2015; Yu et al., 2017) approximates the original informative kernels in the deep networks by incorporating low-rank matrix decomposition for both fully connected layers and convolutional layers. In these methods, the heavy original kernel matrices is decomposed and replaced by low-rank matrices. As such, they reduce the number of parameter and speed up the inference process.

The compact models produced by network compression approaches are expected to be suitable for deployment to mobile devices for fast inference without significant drops in accuracy. Thus, the trade-off between accuracy and efficiency should be always considered when designing these algorithms. Moreover, while network compression indeed produces

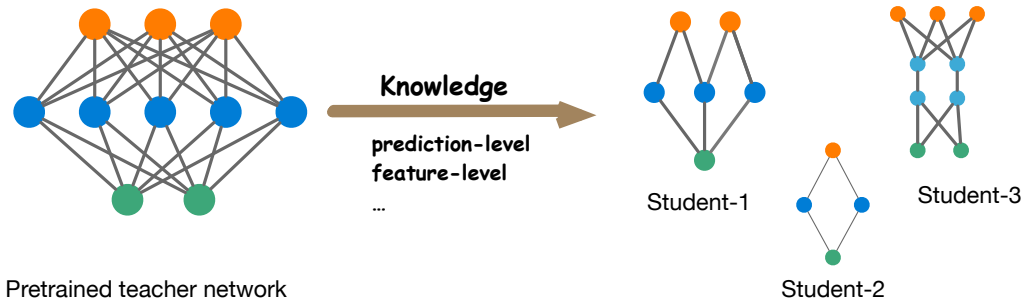


Figure 1.4: **Knowledge distillation** transfers knowledge at different levels from a deep teacher network to a lightweight student network with an arbitrary yet given architecture.

networks that are smaller than the original ones, some of the techniques, such as network pruning approaches do not provide one with the flexibility of designing a network with a specific architecture or of a specific size because the resulting compact models have an unpredictable network structure, and might be still too large to be deployed on some tiny mobile devices.

Knowledge distillation aims to facilitate the training, and thus boost the performance, of a *given* compact model with an *arbitrary* architecture. It is achieved by supervising the training of such a compact student model with knowledge encoded by the predictions (Hinton et al., 2015), the intermediate features (Romero et al., 2014; Zagoruyko and Komodakis, 2017; Heo et al., 2019a) or the relations between features (Yim et al., 2017; Tian et al., 2020) of a pretrained, more powerful teacher model. The key to knowledge distillation is to define, build and transfer the learned knowledge. Therefore, recent research has aimed to design task-specific knowledge distillation techniques for object detection (Chen et al., 2017a; Zhang and Ma, 2021; Guo et al., 2021b) and semantic segmentation (Liu et al., 2019). The advantage of knowledge distillation relies on its flexibility to be extended to other tasks with arbitrary student models and its natural complementary to other efficient deep learning methods as it can boost the performance of a compressed network or of a designed compact architectures.

Overall, the research in the efficient deep learning field has evidenced that deep neural networks are over-parameterized with significantly redundant weights or neurons. While this large number of parameters and computational complexity are required to achieve better performance during training, they are not necessary at inference time. However, the smaller number of parameters in compact and efficient neural networks make it difficult to learn good representations from the data.

The philosophy of this thesis is to study approaches to improve the representation learning ability of *given* compact and efficient neural networks for resource-constrained edge devices and achieve better performance at inference. We tackle this problem from three different perspectives: The design of linear over-parameterization strategies for training, which applies generally to multiple visual recognition tasks; the idea of cross-task and cross-architecture

knowledge distillation for object detection; and a task-driven knowledge distillation solution for 6D pose estimation, for which knowledge distillation has never been studied. Tackling the challenges of improving the representation learning ability and performance of *given* compact and efficient neural networks will increase their applicability to a broader range of applications.

1.2 Contribution

To boost the performance of *given* compact neural networks with *arbitrary* architectures, in this thesis, we present novel techniques based on linear over-parameterization and knowledge distillation to improve the representation learning ability of these compact models. We evidence the effectiveness of our work on a variety of visual recognition tasks. The main contributions of this thesis are:

ExpandNets: Improving the training of compact neural networks for multiple visual recognition tasks.

Conventional deep networks are known to be heavily over-paramenterized, which, while beneficial for training, is not required at inference. Motivated by this observation, we propose to introduce more parameters in a *given* compact neural network during training by linearly expanding its convolutional layers and fully-connected layers. At inference, the linearly expanded layers can be contracted algebraically without any information loss and performance drop. We analyze the optimization behavior and generalization ability of training such ExpandNets. Furthermore, we conduct experiments on multiple visual recognition tasks, including image classification, object detection and semantic segmentation, which evidence the effectiveness of our proposed expansion strategies. Moreover, we show that they are complementary to knowledge distillation methods. This work was accepted as a spotlight at the NeurIPS2020 conference (Guo et al., 2020):

- Guo, S., Alvarez, J. M., and Salzmann, M. (2020). *Expandnets: Linear over-parameterization to train compact convolutional networks.* Advances in Neural Information Processing Systems (NeurIPS2020 Spotlight).
- Code: https://github.com/GUOShuxuan/expandnets

Classifier-to-detector knowledge distillation: A cross-task and cross-architecture knowledge distillation approach for object detection.

Existing knowledge distillation approaches for visual recognition have only been demonstrated when the student and teacher networks exploit the same architecture family and tackle the same task, e.g., ResNet to ResNet, Wide ResNet to Wide ResNet; classifier to classifier, or detector to detector. As a second contribution of this thesis, we investigate the use of knowledge distillation across tasks and develop strategies to distill the knowledge of an image classification teacher to an object detection student. We observe that our proposed

classifier-to-detector knowledge distillation method decreases both the classification and localization errors of the detector student. Furthermore, the classifier teacher can be used to complement a detector teacher to jointly boost the performance of the student detector, outperforming the use of only one teacher. This contribution opens the door to the research direction of cross-task and cross-architecture knowledge distillation. This work was published at the NeurIPS2021 conference (Guo et al., 2021b):

- Guo, S., Alvarez, J. M., and Salzmann, M. (2021b). *Distilling image classifiers in object detectors*. Advances in Neural Information Processing Systems (NeurIPS2021).
- Code: https://github.com/NVlabs/DICOD

Keypoint distribution alignment: A task-driven knowledge distillation approach for 6D pose estimation.

Knowledge distillation has achieved great progress on multiple visual tasks, such as image classification, object detection, and semantic segmentation. However, it has never been studied for image-based 6D object pose estimation. In this part of the thesis, we observe that, for 6D pose estimation, the keypoint based models are less sensitive to the network size than the dense prediction based ones. Therefore, we propose a knowledge distillation method based on optimal transport to align the keypoint distribution from a teacher model to that of a student model. Specifically, we leverage both the predicted segmentation scores and locations of the keypoints during the distribution alignment process. We show that our proposed task-driven knowledge distillation method for 6D pose estimation improves the performance of lightweight student models. This work is currently under review (Guo et al., 2022):

- Guo, S., Hu, Y., Alvarez, J. M., and Salzmann, M. (2022). *Knowledge Distillation for 6D Pose Estimation by Keypoint Distribution Alignment*. arXiv Preprint.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces our proposed expansion algorithm based on linear over-parameterization to facilitate the training of *arbitrary, given* compact networks, improving both their optimization and generalization and yielding better performance at inference for multiple visual recognition tasks. Chapter 3 presents our classifier-to-detector knowledge transfer framework for training compact object detectors and demonstrates the effectiveness of this approach not only across architectures but also across tasks. Chapter 4 focuses on the 6D pose estimation task and introduces our keypoint distribution alignment knowledge distillation approach based on optimal transport theory. Finally, Chapter 5 concludes the thesis with a summary of our contributions and a brief discussion of future research directions.

2 ExpandNets

Over-parameterization is widely acknowledged to facilitate network training by improving both neural network optimization and generalization. In this chapter, we introduce an approach leveraging over-parameterization to train a *given* compact convolutional network with *arbitrary* architecture. Specifically, we propose to expand each linear layer of the compact network into multiple consecutive linear layers, without adding any nonlinearity. As such, the resulting expanded network, or ExpandNet, can be contracted back to the compact one algebraically at inference. In particular, we introduce two convolutional expansion strategies and demonstrate their benefits on several tasks, including image classification, object detection, and semantic segmentation. As evidenced by our experiments, our approach outperforms both training the compact network from scratch and performing knowledge distillation from a teacher. Furthermore, our linear over-parameterization empirically reduces gradient confusion during training and improves the network generalization.

The contents of this chapter are mainly from the following paper. I am the primary contributor.

- Guo, S., Alvarez, J. M., and Salzmann, M. (2020). *Expandnets: Linear over-parameterization to train compact convolutional networks.* Advances in Neural Information Processing Systems (NeurIPS2020 Spotlight).

2.1 Introduction

With the growing availability of large-scale datasets and advanced computational resources, convolutional neural networks have achieved tremendous success in a variety of tasks, such as image classification (He et al., 2016; Krizhevsky et al., 2012a), object detection (Redmon and Farhadi, 2016, 2018; Ren et al., 2015) and semantic segmentation (Long et al., 2015; Ronneberger et al., 2015). Over the past few years, "Wider and deeper are better" has become the rule of thumb to design network architectures (He et al., 2016; Huang et al., 2017; Simonyan and Zisserman, 2015; Szegedy et al., 2015). This trend, however, raises memory- and computation-related challenges, especially in the context of constrained environments, such as embedded platforms.

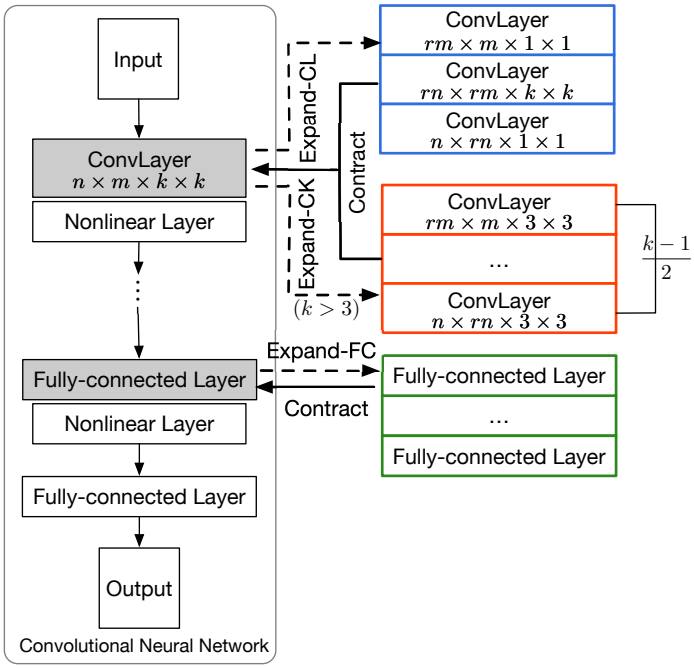


Figure 2.1: **ExpandNets.** We propose 3 strategies to linearly expand a compact network. An expanded network can then be contracted back to the compact one algebraically, and outperforms training the compact one, even with knowledge distillation.

Deep and wide networks are well-known to be heavily over-parameterized, and thus a compact network, both shallow and thin, should often be sufficient. Unfortunately, compact networks are notoriously hard to train from scratch. As a consequence, designing strategies to train a given compact network has drawn growing attention, the most popular approach consisting of transferring the knowledge of a deep teacher network to the compact one of interest (Heo et al., 2019b,a; Hinton et al., 2015; Passalis and Tefas, 2018; Romero et al., 2014; Tian et al., 2020; Yim et al., 2017; Zagoruyko and Komodakis, 2017).

In this chapter, we introduce an alternative approach to training compact neural networks, complementary to knowledge transfer. To this end, building upon the observation that network over-parameterization improves both optimization and generalization (Allen-Zhu et al., 2018a,b; Arora et al., 2018; Kawaguchi et al., 2018; Reed, 1993; Sankararaman et al., 2019; Zhang et al., 2017), we propose to increase the number of parameters of a given compact network by incorporating additional layers. However, instead of separating every two layers with a nonlinearity, we advocate introducing consecutive *linear* layers. In other words, we expand each linear layer of a compact network into a succession of multiple linear layers, without any nonlinearity in between. Since consecutive linear layers are equivalent to a single one (Saxe et al., 2014), such an expanded network, or ExpandNet, can be algebraically contracted back to the original compact one without any information loss.

While the use of successive linear layers appears in the literature, existing work (Arora et al., 2018; Baldi and Hornik, 1989; Kawaguchi, 2016; Laurent and von Brecht, 2018; Saxe et al., 2014; Zhou and Liang, 2018) has been mostly confined to *fully-connected networks without any nonlinearities* and to the theoretical study of their behavior under fairly unrealistic statistical assumptions. In particular, these studies aim to understand the learning dynamics and the loss landscapes of deep networks. Here, by contrast, we focus on *practical, nonlinear, compact convolutional* neural networks, and demonstrate the use of linear expansion as a means to introduce over-parameterization and facilitate training, so that a given compact network achieves better performance.

Specifically, as illustrated by Figure 2.1, we introduce three ways to expand a compact network: (i) replacing a $k \times k$ convolution by three convolutional layers with kernel size 1×1, $k \times k$ and 1×1, respectively; (ii) replacing a $k \times k$ convolution with $k > 3$ by multiple 3×3 convolutions; and (iii) replacing a fully-connected layer with multiple ones. Our experiments demonstrate that expanding convolutions is the key to obtaining more effective compact networks.

In short, our contributions in this chapter are (i) a novel approach to training a given compact, nonlinear convolutional network by expanding its linear layers; (ii) a strategy to expand convolutional layers with arbitrary kernels; and (iii) a strategy to expand convolutional layers with kernel size larger than 3. We demonstrate the benefits of our approach on several tasks, including image classification on ImageNet, object detection on PASCAL VOC and image segmentation on Cityscapes. Our ExpandNets outperform both training the corresponding compact networks from scratch and using knowledge distillation. We empirically show over-parameterization to be the key factor for such better performance. Furthermore, we analyze the benefits of linear over-parameterization during training via experiments studying generalization, gradient confusion and the loss landscape. Our code is available at https://github.com/GUOShuxuan/expandnets.

2.2 Related Work

Very deep convolutional neural networks currently constitute the state of the art for many tasks. These networks, however, are known to be heavily over-parameterized, and making them smaller would facilitate their use in resource-constrained environments, such as embedded platforms. As a consequence, much research has recently been devoted to developing more compact architectures.

Network compression constitutes one of the most popular trends in this area. In essence, it aims to reduce the size of a large network while losing as little accuracy as possible, or even none at all. In this context, existing approaches can be roughly grouped into two categories: (i) parameter pruning and sharing (Carreira-Perpinan and Idelbayev, 2018; Courbariaux et al., 2016; Figurnov et al., 2016; Han et al., 2016; LeCun et al., 1990; Lee et al., 2019; Molchanov et al., 2017; Ullrich et al., 2017), which aims to remove the least informative parameters, yielding an arbitrary compact network with information loss; and (ii) low-rank matrix factorization (Denil

et al., 2013; Jin et al., 2015; Lebedev et al., 2014; Liu et al., 2015; Sainath et al., 2013), which uses decomposition techniques to reduce the size of the parameter matrix/tensor in each layer. While compression is typically performed as a post-processing step, it has been shown that incorporating it during training could be beneficial (Alvarez and Salzmann, 2016, 2017; Wen et al., 2016, 2017). In any event, even though compression reduces a network's size, it neither provides one with the flexibility of designing a network with a specific architecture, nor incorporates over-parameterization to improve the performance of compact network training. Furthermore, it often produces networks that are much larger than the ones we consider here, e.g., compressed networks with $O(1\text{M})$ parameters vs $O(10\text{K})$ for the SmallNets used in our experiments.

In a parallel line of research, several works have proposed design strategies to reduce a network's number of parameters (Howard et al., 2017; Ma et al., 2018; Romera et al., 2018; Sandler et al., 2018; Szegedy et al., 2016; Wu et al., 2016). Again, while more compact networks can indeed be developed with these mechanisms, they impose constraints on the network architecture, and thus do not allow one to simply train a given compact network. Furthermore, as shown by our experiments, our approach is complementary to these works. For example, we can improve the results of MobileNets (Howard et al., 2017; Sandler et al., 2018) by training them using our expansion strategy.

Here, in contrast to the above-mentioned literature, we seek to train a *given* compact network with an arbitrary architecture. This is also the task addressed by knowledge transfer approaches (Heo et al., 2019a,b; Hinton et al., 2015; Passalis and Tefas, 2018; Romero et al., 2014; Tian et al., 2020; Yim et al., 2017; Zagoruyko and Komodakis, 2017). To achieve this, existing methods leverage the availability of a pre-trained very deep teacher network. In this chapter, we introduce an alternative strategy to train compact networks, complementary to knowledge transfer. Inspired by the theory showing that over-parameterization helps training (Allen-Zhu et al., 2018a,b; Arora et al., 2018; Kawaguchi et al., 2018; Reed, 1993; Sankararaman et al., 2019; Zhang et al., 2017), we expand each linear layer in a given compact network into a succession of multiple linear layers. Our experiments evidence that training such expanded networks, which can then be contracted back algebraically, yields better results than training the original compact networks, thus empirically confirming the benefits of over-parameterization. Our results also show that our approach outperforms knowledge distillation, even without using a teacher network.

Note that linearly over-parameterized neural networks have been investigated both in the early neural network days (Baldi and Hornik, 1989) and more recently (Arora et al., 2018; Gunasekar et al., 2018; Kawaguchi, 2016; Laurent and von Brecht, 2018; Saxe et al., 2014; Zhou and Liang, 2018). These methods, however, typically study purely linear networks, with a focus on the convergence behavior of training in this linear regime. For example, Gunasekar et al. (2018) demonstrated that a different parameterization of the same model dramatically affects the training behavior; Arora et al. (2018) showed that linear over-parameterization modifies the gradient updates in a unique way that speeds up convergence; Wu et al. (2019)

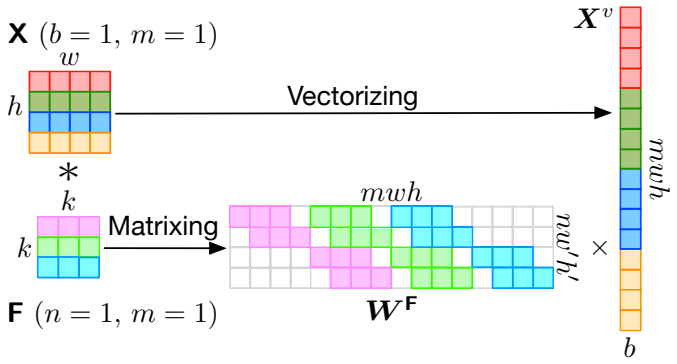


Figure 2.2: **Matrix representation of a convolutional layer** (best viewed in color).

collapsed multiple FC layers into a single one by removing the non-linearities from the *MLP* layers of a graph convolutional network. In contrast to these methods, which focus on fully-connected layers, we develop two strategies to expand *convolutional* layers, and empirically demonstrate the impact of our expansion strategies on prediction accuracy, training behavior and generalization ability.

The concurrent work ACNet of Ding et al. (2019) also advocates for expansion of convolutional layers. However, the two strategies we introduce differ from their use of 1D asymmetric convolutions, and our experiments show that our approach outperforms theirs. More importantly, this work constitutes further evidence of the benefits of linear expansion.

2.3 Methodology

Let us now introduce our approach to training compact networks by linearly expanding their layers. Below, we focus on our two strategies to expand convolutional layers, and then briefly discuss the case of fully-connected ones.

2.3.1 Expanding Convolutional Layers

We propose to linearly expand a convolutional layer by replacing it with a series of convolutional layers. To explain this, we will rely on the fact that a convolution operation can be expressed in matrix form. Specifically, let $\mathbf{X}_{b\times m\times w\times h}$ be the input tensor to a convolutional layer, with batch size b, m input channels, height h and width w, and $\mathbf{F}_{n\times m\times k\times k}$ be the tensor encoding the convolutional filters, with n output channels and kernel size k. Ignoring the bias, which can be taken into account by incorporating an additional channel with value 1 to $\mathbf{X}$, a convolution can be expressed as

$$\mathbf{Y}_{b\times n\times w'\times h'} = \mathbf{X}_{b\times m\times w\times h} * \mathbf{F}_{n\times m\times k\times k} = \text{reshape}(\boldsymbol{W}^{\mathbf{F}}_{nw'h'\times mwh} \times \boldsymbol{X}^{v}_{mwh\times b}) \,, \tag{2.1}$$

where $\mathbf{Y}_{b\times n\times w'\times h'}$ is the output tensor, $\boldsymbol{W}^{\mathbf{F}}_{nw'h'\times mwh}$ is a highly structured sparse matrix containing the convolutional filters, and $\boldsymbol{X}^{v}_{mwh\times b}$ is a matrix representation of $\mathbf{X}$. This process,

which is a bijection, is depicted by Figure 2.2 for $b = 1$, $m = 1$ and $n = 1$.

With this matrix representation, one can therefore expand a layer linearly by replacing the matrix $\boldsymbol{W}^{\mathsf{F}}$ with a product of an arbitrary number of matrices. However, using arbitrary matrices would ignore the convolution structure, and thus alter the original operation performed by the layer. Fortunately, multiplying several convolution matrices still yields a valid convolution, as can be confirmed by observing the pattern within the matrix in Figure 2.2. Nevertheless, one cannot simply expand a convolutional layer with kernel size $k \times k$ as a series of convolutions with arbitrary kernel sizes because, in general, the resulting receptive field size would differ from the original one. To overcome this, we propose the two expansion strategies discussed below.

Expanding general convolutions. For our first strategy, we note that 1×1 convolutions retain the computational benefits of convolutional layers while not modifying the receptive field size. As illustrated in Figure 2.1, we therefore propose to expand a $k \times k$ convolutional layer into 3 consecutive convolutional layers: a 1×1 convolution; a $k \times k$ one; and another 1×1 one. Importantly, this allows us to increase not only the number of layers, but also the number of channels by setting $p, q > n, m$. To this end, we rely on the notion of *expansion rate*. Specifically, for an original layer with m input channels and n output ones, given an expansion rate r, we define the number of output channels of the first 1×1 layer as $p = r\,m$ and the number of output channels of the intermediate $k \times k$ layer as $q = r\,n$. Note that other strategies are possible, e.g., $p = r^i\,m$, but ours has the advantage of preventing the number of parameters from exploding.

Once such an expanded convolutional layer has been trained, one can contract it back to the original one algebraically by considering the matrix form of Eq. 2.1. That is, given the filter tensors of the intermediate layers, $\mathsf{F}^1_{p\times m\times 1\times 1}$, $\mathsf{F}^2_{q\times p\times k\times k}$ and $\mathsf{F}^3_{n\times q\times 1\times 1}$, the matrix representation of the original layer can be recovered as

$$\boldsymbol{W}^{\mathsf{F}}_{nw'h'\times mwh} = \boldsymbol{W}^{\mathsf{F}^3}_{nw'h'\times qw'h'} \times \boldsymbol{W}^{\mathsf{F}^2}_{qw'h'\times pwh} \times \boldsymbol{W}^{\mathsf{F}^1}_{pwh\times mwh}\,, \tag{2.2}$$

which encodes a convolution tensor. At test time, we can thus use the original compact network. Because it applies to any size k, we will refer to this strategy as *expanding convolutional layers*.

Expanding $k \times k$ convolutions with $k > 3$. While 3×3 kernels are popular in very deep architectures (He et al., 2016), larger kernel sizes are often exploited in compact networks, so as to increase their expressiveness and their receptive fields. As illustrated in Figure 2.1, $k \times k$ kernels with $k > 3$ can be equivalently represented with a series of l 3×3 convolutions, where $l = (k-1)/2$. Note that k is typically odd in CNNs. We then have

$$\mathsf{Y} = \mathsf{X} * \mathsf{F}_{n\times m\times k\times k} = \mathsf{X} * \mathsf{F}^1_{p_1\times m\times 3\times 3} * \cdots * \mathsf{F}^{l-1}_{p_{l-1}\times p_{l-2}\times 3\times 3} * \mathsf{F}^{l}_{n\times p_{l-1}\times 3\times 3}\,. \tag{2.3}$$

As before, the number of channels in the intermediate layers can be larger than that in the original $k \times k$ one, thus allowing us to linearly over-parameterize the model. For an expansion rate r, we set the number of output channels of the first 3×3 layer to $p_1 = rm$ and that of the subsequent layers to $p_i = rn$. The same matrix-based strategy as before can be used to algebraically contract back the expanded unit into $\mathsf{F}_{n \times m \times k \times k}$. We will refer to this strategy as *expanding convolutional kernels*.

2.3.2 Expanding Convolutions in Practice

Padding and strides. In modern convolutional networks, padding and strides are widely used to retain information from the input feature map while controlling the size of the output one. To expand a convolutional layer with padding p, we propose to use padding p in the first layer of the expanded unit while not padding the remaining layers. Furthermore, to handle a stride s, when expanding convolutional layers, we set the stride of the middle layer to s and that of the others to 1. When expanding convolutional kernels, we use a stride 1 for all layers except for the last one whose stride is set to s. These two strategies guarantee that the resulting ExpandNet can be contracted back to the compact model without any information loss.

Depthwise convolutions. Depthwise convolutions are often used to design compact networks, such as MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018) and ShuffleNetV2 (Ma et al., 2018). To handle them, we make use of our general convolutional expansion strategy within each group. Specifically, we duplicate the input channels r times and employ cross-channel convolutions within each group. This makes the expanded layers equivalent to the original ones.

2.3.3 Expanding Fully-connected Layers

Beacuse the weights of a fully-connected layer can naturally be represented in matrix form, our approach directly extends to such layers. That is, we can expand a fully-connected layer with m input and n output dimensions into l layers as by noting that

$$\boldsymbol{W}_{n \times m} = \boldsymbol{W}_{n \times p_{l-1}} \times \boldsymbol{W}_{p_{l-1} \times p_{l-2}} \times \cdots \times \boldsymbol{W}_{p_1 \times m}\,, \tag{2.4}$$

where we typically define $p_1 = rm$ and $p_i = rn$, $\forall i \neq 1$. In practice, considering the computational complexity of fully-connected layers, we advocate expanding each layer into only two or three layers with a small expansion rate. Note that this expansion is similar to that used in (Arora et al., 2018), which we discuss in more detail in Section 2.7. However, as will be shown by our experiments, expanding *only* fully-connected layers, as in (Arora et al., 2018), does typically not yield a performance boost. By contrast, our two convolutional expansion strategies do.

Altogether, our strategies allow us to expand an arbitrary compact network into an equivalent

deeper and wider one, and can be used independently or together. Once trained, the resulting ExpandNet can be contracted back to the original compact architecture in an algebraic manner, i.e., at no loss of information.

2.4 Experiments

In this section, we demonstrate the benefits of our ExpandNets on image classification, object detection, and semantic segmentation.

We denote the expansion of convolutional layers by *CL*, of convolutional kernels by *CK*, and of fully-connected layers by *FC*. Specifically, we use *FC(Arora18)* to indicate that the expansion strategy is similar to the one used in (Arora et al., 2018). When combining convolutional expansions with fully-connected ones, we use *CL+FC* or *CK+FC*.

2.4.1 Image Classification

We first study the use of our approach with very small networks on CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009), and then turn to the more challenging ImageNet (Russakovsky et al., 2015) dataset, where we show that our method can improve the results of the compact MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018) and ShuffleNetV2 0.5× (Ma et al., 2018).

CIFAR-10 and CIFAR-100

Table 2.1: Top-1 accuracy (%) of SmallNet with 3×3 kernels vs ExpandNets with $r = 4$ on CIFAR-10 and CIFAR-100.

Model	Transfer	CIFAR-10	CIFAR-100
SmallNet	*w/o KD*	73.32 ± 0.20	40.40 ± 0.60
FC(Arora18) (Arora et al., 2018)	*w/o KD*	73.78 ± 0.83	40.52 ± 0.71
SmallNet	*w/ KD*	73.34 ± 0.31	40.46 ± 0.56
ExpandNet-CL	*w/o KD*	73.96 ± 0.30	40.91 ± 0.47
ExpandNet-CL+FC	*w/o KD*	$\mathbf{74.45 \pm 0.29}$	$\mathbf{41.12 \pm 0.49}$
ExpandNet-CL+FC	*w/ KD*	$\mathbf{74.52 \pm 0.37}$	$\mathbf{41.51 \pm 0.49}$

Experimental setup. For CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009), we use the same compact network as in (Passalis and Tefas, 2018). It is composed of 3 convolutional layers with 3×3 kernels and no padding. These 3 layers have 8, 16 and 32 output channels, respectively. Each of them is followed by a batch normalization layer, a ReLU layer and a 2×2 max pooling layer. The output of the last layer is passed through a fully-connected layer with 64 units, followed by a logit layer with either 10 or 100 units. All networks, including our ExpandNets, were trained for 150 epochs using a batch size of 128. We used standard

Table 2.2: Top-1 accuracy (%) of SmallNet with 7×7 kernels vs ExpandNets with $r = 4$ on CIFAR-10 and CIFAR-100.

Model	Transfer	CIFAR-10	CIFAR-100
SmallNet	*w/o* KD	78.63 ± 0.41	46.63 ± 0.27
FC(Arora18) (Arora et al., 2018)	*w/o* KD	78.64 ± 0.39	46.59 ± 0.45
ACNet (Ding et al., 2019)	*w/o* KD	79.37 ± 0.52	47.18 ± 0.57
SmallNet	*w/* KD	78.97 ± 0.37	47.04 ± 0.35
ExpandNet-CL	*w/o* KD	78.47 ± 0.20	46.90 ± 0.66
ExpandNet-CL+FC		79.11 ± 0.23	46.66 ± 0.43
ExpandNet-CK		80.27 ± 0.24	48.55 ± 0.51
ExpandNet-CK+FC		$\mathbf{80.31 \pm 0.27}$	$\mathbf{48.62 \pm 0.47}$
ExpandNet-CL+FC	*w/* KD	79.60 ± 0.25	47.41 ± 0.51
ExpandNet-CK+FC		$\mathbf{80.63 \pm 0.31}$	$\mathbf{49.13 \pm 0.45}$

stochastic gradient descent (SGD) with a momentum of 0.9 and a learning rate of 0.01, divided by 10 at epochs 50 and 100. With this strategy, all networks reached convergence. To evaluate our kernel expansion method CL and CK, we also report results obtained with a similar network where the 3×3 kernels were replaced by 7×7 ones with a padding of 3, because the CK one does not apply to 3×3 kernels. In this set of experiments, the expansion rate r is set to 4 to balance the accuracy-efficiency trade-off. Since our expansion strategy is complementary to knowledge transfer, i.e., an ExpandNet can act as student in knowledge transfer, we further demonstrate its benefits in this setting by conducting experiments using knowledge distillation (KD) (Hinton et al., 2015), hint-based transfer (Hint)(Romero et al., 2014) or probabilistic knowledge transfer (PKT) (Passalis and Tefas, 2018) from a ResNet18 teacher.

We then evaluate our expansion strategies on MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018), which we train for 350 epochs using a batch size of 128. We use stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 0.0005 and a learning rate of 0.1, divided by 10 at epochs 150 and 250. Note that training an ExpandNet takes slightly more time than training the compact network because of the extra parameters, as reported in Table 2.3. Therefore, to confirm that our better results are not just due to longer training, we also report the results of the baselines trained for the same amount of time as our ExpandNets.

Results. We first report the results over 5 runs of the model with 3×3 kernels in Table 2.1, expanding the convolutional layers by CL expansion strategy yields higher accuracy than the small network. This is further improved by also expanding the fully-connected layer. We then focus on the SmallNet with 7×7 kernels, for which we can evaluate all our expansion strategies, including the CK ones. Table 2.2 provides the results over 5 runs of all our networks with and without KD, which we have found to be the most effective knowledge transfer strategy, as evidenced by comparing these results with those obtained by Hint and PKT in Table 2.16 in

Table 2.3: Top-1 accuracy (%) of MobileNets vs ExpandNets with $r = 4$ on CIFAR-10 and CIFAR-100.

Model	Epoch Time	CIFAR-10	CIFAR-100
MobileNet	13.08s	89.61 (88.87†)	67.93 (68.18†)
ExpandNet-CL	22.78s	**91.79**	**69.75**
MobileNetV2	24.88s	91.64 (90.85†)	71.66 (71.41†)
ExpandNet-CL	49.22s	**92.58**	**72.33**

† Accuracy with the same training time as ExpandNet-CL.

‡ Epoch Time was evaluated on CIFAR-10 on 2 32G TITAN V100 GPUs.

Table 2.4: Top-1 accuracy (%) on the ILSVRC2012 validation set (ExpandNets with $r = 4$).

Model	MobileNet	MobileNetV2	ShuffleNetV2
original	66.48	63.75	56.89
ACNet (Ding et al., 2019)	67.61	64.29	52.43
original (*w/* KD)	69.01	65.40	57.59
ExpandNet-CL	**69.40**	**65.62**	**57.38**
ExpandNet-CL (*w/* KD)	**70.47**	**67.19**	**57.68**

Section 2.6. As shown in the top portion of the table, only expanding the fully-connected layer, as in (Arora et al., 2018), yields mild improvement. However, expanding the convolutional ones clearly outperforms the compact network, and is further boosted by expanding the fully-connected one. Overall, expanding the kernels yields the best results; it outperforms even the concurrent convolutional expansion ACNet of Ding et al. (2019). Note that even without KD, our ExpandNets outperform SmallNet with KD. The gap is further increased when we also use KD, as shown in the bottom portion of the table.

In Table 2.3, we provide the results for MobileNet and MobileNetV2, including the baselines trained for a longer time, denoted by a †. These results confirm that our expansion strategies also boost the performance of these MobileNet models, even when the baselines are trained longer.

ImageNet

Experimental setup. For ImageNet (Russakovsky et al., 2015), we use the compact MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018) and ShuffleNetV2 (Ma et al., 2018) models, which were designed to be compact and yet achieve good results. We rely on a pytorch implementation of these models. For our approach, we use our CL strategy to expand all convolutional layers with kernel size 3×3 in MobileNet and ShuffleNetV2, while only expanding the non-residual 3×3 convolutional layers in MobileNetV2. We trained the MobileNets using the short-term regime advocated in (He et al., 2016), i.e., 90 epochs with a weight decay of 0.0001 and an initial learning rate of 0.1, divided by 10 every 30 epochs. We

employed SGD with a momentum of 0.9 and a batch size of 256. For ShuffleNet, we used the small ShuffleNetV2 0.5×, trained as in (Ma et al., 2018). We also applied KD from a ResNet152 (with 78.32% top-1 accuracy), tuning the KD hyper-parameters to the best accuracy for each method.

Results. We compare the results of the original models with those of our expanded versions in Table 2.4. Our expansion strategy increases the top-1 accuracy of MobileNet, MobileNetV2 and ShuffleNetV2 0.5× by 2.92, 1.87 and 0.49 percentage points (pp). It also yields consistently higher accuracy than the concurrent ACNet of Ding et al. (2019). Furthermore, our ExpandNets without KD outperform the MobileNets with KD, even though we do not require a teacher.

2.4.2 Object Detection

Our approach is not restricted to image classification. We demonstrate its benefits for one-stage object detection.

Table 2.5: YOLO-LITE vs ExpandNet with $r = 4$ on the PASCAL VOC2007 test set.

Model	mAP (%)
YOLO-LITE	27.34
ExpandNet-CL	**30.97**

Experimental setup. YOLO-LITE (Huang et al., 2018), which was designed to work in constrained environments. The YOLO-LITE used here is very compact, consisting of a backbone with only 5 convolutional layers and of a head. We expanded the 5 backbone convolutional layers using our CL strategy with $r = 4$, and trained the resulting network on the PASCAL VOC2007 + 2012 (Everingham et al., 2007, 2012) training and validation sets in the standard YOLO fashion (Redmon and Farhadi, 2016, 2018). We report the mean average precision (mAP) on the PASCAL VOC2007 test set.

Results. The results are reported in Table 2.5. As for object detection, our expansion strategy boosts the performance of the compact network. Specifically, we outperform it by over 3.5pp.

2.4.3 Semantic Segmentation

We then demonstrate the benefits of our approach on semantic segmentation using the Cityscapes dataset (Cordts et al., 2016).

Table 2.6: U-Net vs ExpandNet with $r = 4$ on the Cityscapes validation set.

Model	mIOU	mRec	mPrec
U-Net	56.59	74.29	65.11
ExpandNet-CL	**57.85**	**76.53**	**65.94**

Experimental setup. For this experiment, we rely on the U-Net (Ronneberger et al., 2015), which is a relatively compact network consisting of a contracting and an expansive path. We apply our CL expansion strategy with $r = 4$ to all convolutions in the contracting path. We train the networks on 4 GPUs using the standard SGD optimizer with a momentum of 0.9 and a learning rate of $1e-8$. Following

the standard protocol, we report the mean Intersection over Union (mIOU), mean recall (mRec) and mean precision (mPrec).

Results. Our results on the Cityscapes validation set are shown in Table 2.6. Note that our ExpandNet outperforms the original compact U-Net.

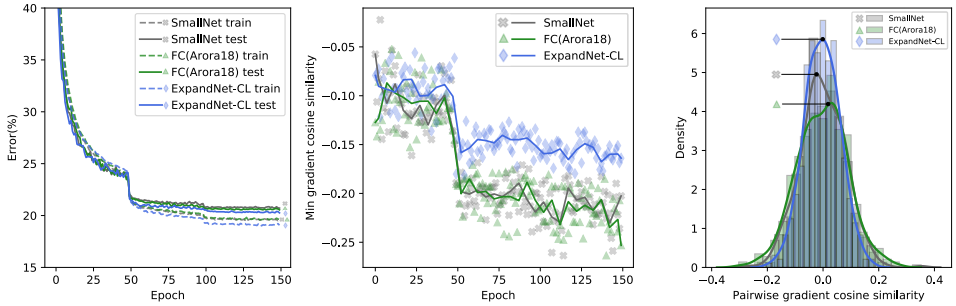
2.5 Analysis

To further analyze our approach, we first study its behavior during training and its generalization ability. For these experiments, we make use of the CIFAR-10 and CIFAR-100 datasets, and use the settings described in detail in our ablation study in Section 2.6. We then propose and analyze two hypotheses to empirically evidence that the better performance of our approach truly is a consequence of over-parameterization during training. We also evaluate the complexity of the models in terms of number of parameters, multiply-and-accumulate operations (MACs), and training and testing inference speed. Note that, since our ExpandNets can be contracted back to the original networks, *at test time, they have exactly the same number of parameters, MACs, and inference time as the original networks, but achieve better performance.*

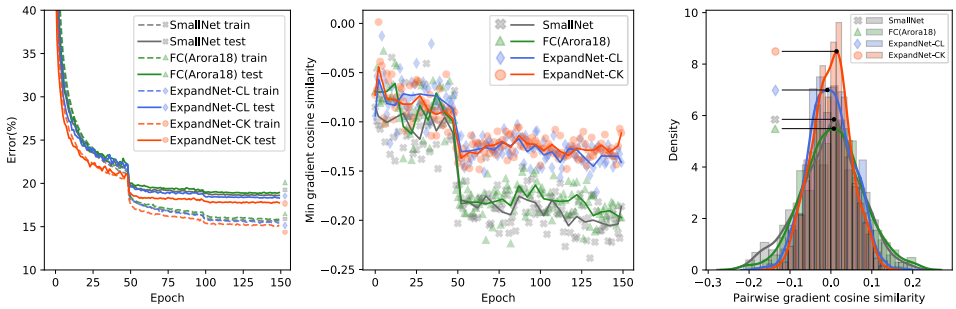
2.5.1 Training Behavior

To investigate the benefits of linear over-parameterization on training, we make use of the gradient confusion introduced by Sankararaman et al. (2019) to show that the gradients of non-linearly over-parameterized networks were more consistent across mini-batches. Specifically, following (Sankararaman et al., 2019), we measure gradient confusion (or rather consistency) as the minimum cosine similarity of gradients over 100 randomly-sampled pairs of mini-batches at the end of each training epoch. It measures the negative correlation between the gradients of different mini-batches, and thus indicates a disagreement on the parameter update. As in (Sankararaman et al., 2019), we also combine the gradient cosine similarity of 100 pairs of sampled mini-batches at the end of training from each independent run and perform Gaussian kernel density estimation on this data.

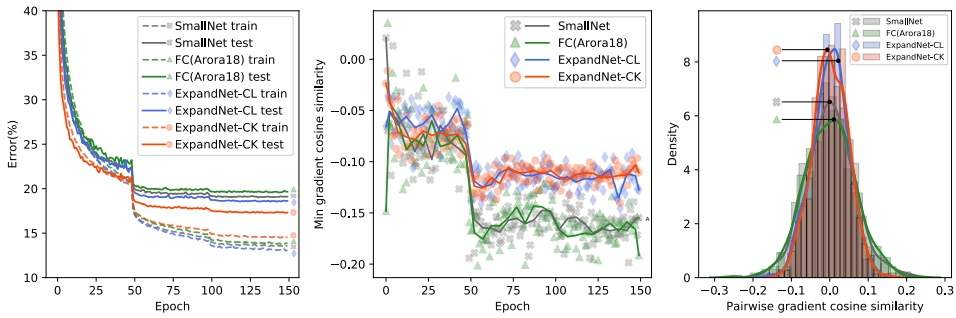
We run each experiment 5 times and show the average values across all runs on CIFAR-10 in Figure 2.3 and on CIFAR-100 in Figure 2.4 corresponding to networks with kernel sizes of 3, 5, 7, 9, respectively. The training and test curves in almost all cases show that our ExpandNets-CL/CK speed up convergence and yield a smaller generalization error. They also yield lower gradient confusion (higher minimum cosine similarity) and a more zero-peaked density of pairwise gradient cosine similarity. This indicates that our networks are easier to train than the compact model. By contrast, only expanding the FC layers, as in (Arora et al., 2018), does not facilitate training.



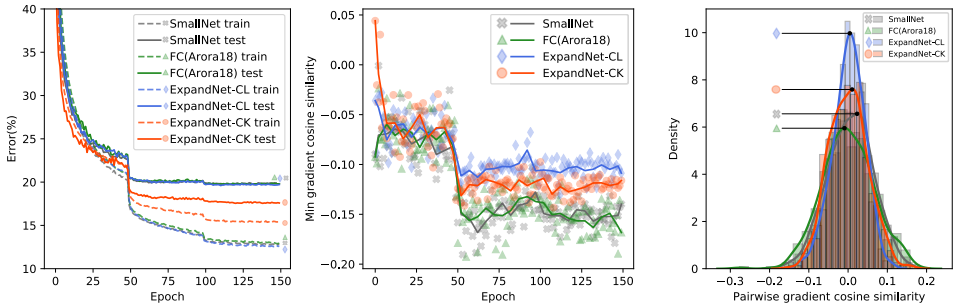
(a) kernel size: 3



(b) kernel size: 5

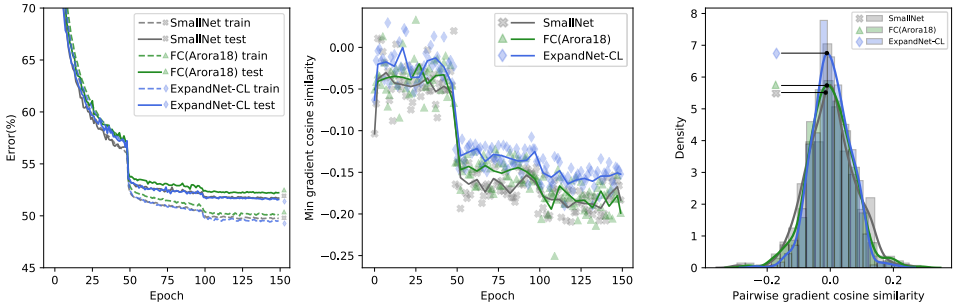


(c) kernel size: 7

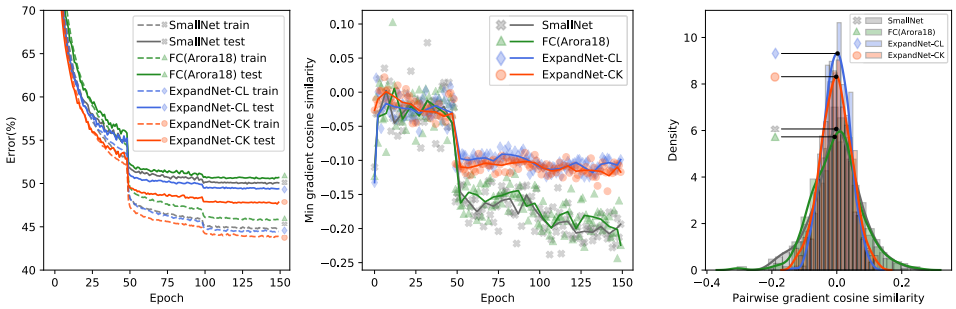


(d) kernel size: 9

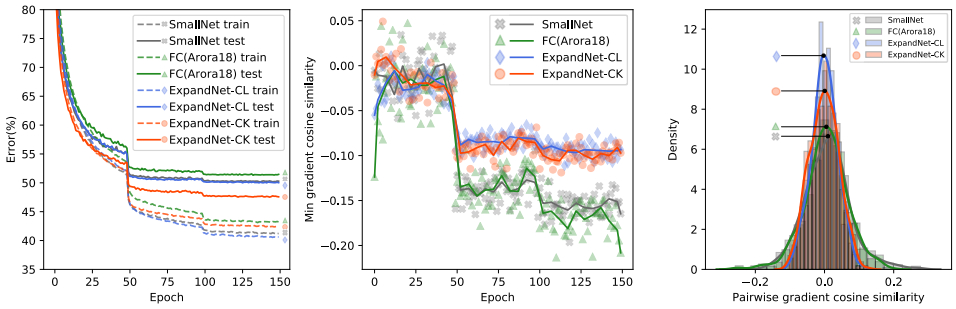
Figure 2.3: **Training behavior of networks on CIFAR-10** (best viewed in color). *Left:* Training and test curves over 150 epochs. *Middle:* Minimum pairwise gradient cosine similarity at the end of each training epoch (higher is better). *Right:* Kernel density estimation of pairwise gradient cosine similarity at the end of training (over 5 independent runs).



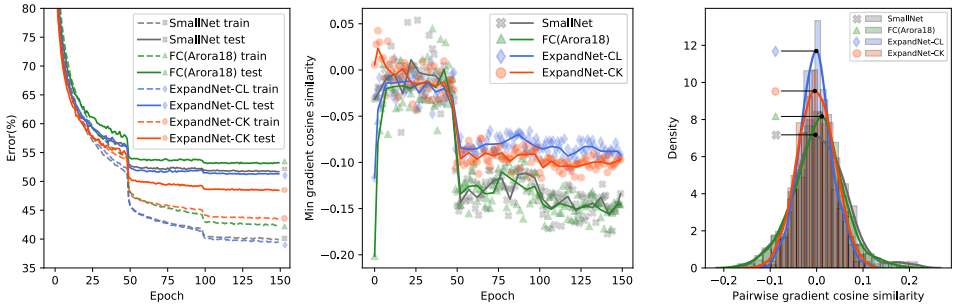
(a) kernel size: 3



(b) kernel size: 5



(c) kernel size: 7



(d) kernel size: 9

Figure 2.4: **Training behavior of networks on CIFAR-100** (best viewed in color). *Left:* Training and test curves over 150 epochs. *Middle:* Minimum pairwise gradient cosine similarity at the end of each training epoch (higher is better). *Right:* Kernel density estimation of pairwise gradient cosine similarity at the end of training (over 5 independent runs).

Table 2.7: **Complexity analysis on CIFAR-10 for different expansion rates** r. The baseline network is the SmallNet with kernel size 7 (#Params:150.35K, #MACs: 6.12M, Epoch Time: 4.05s). Note that, for a given training setting, the wall-clock time only moderately increases as r grows.

r		FC(Arora18)	ExpandNet-CL	ExpandNet-CK
2	#Params	339.40K	562.95K	237.72K
	#MACs	6.30M	25.16M	14.38M
	Epoch Time	4.09s	4.13s	4.10s
4	#Params	675.91K	2.17M	653.25K
	#MACs	6.64M	98.39M	42.64M
	Epoch Time	3.94s	4.61s	4.12s
8	#Params	1.74M	8.58M	2.07M
	#MACs	7.70M	389.35M	141.10M
	Epoch Time	4.02s	9.39s	5.50s

Table 2.8: **ExpandNet complexity analysis on CIFAR-10, ImageNet, PASCAL VOC and Cityscapes.** Note that, within each task, the metrics are the same for all networks, since we can compress our ExpandNets back to the small network.

Model	# Params(M)		# MACs		GPU Speed (imgs/sec)	
	Train	Test	Train	Test	Train	Test
SmallNet (7×7)	0.07	0.07	4.49M	4.49M	147822.51	154850.52
ExpandNet-CL	0.55		57.49M		64651.81	
ExpandNet-CL+FC	2.11		59.04M		61379.95	
ExpandNet-CK	0.19		23.95M		75065.09	
ExpandNet-CK+FC	1.75		25.5M		68679.89	
MobileNet	4.23	4.23	0.58G	0.58G	3797.21	3829.81
ExpandNet-CL	4.96		1.76G		729.78	
MobileNetV2	3.50	3.50	0.32G	0.32G	3417.20	3419.43
ExpandNet-CL	3.67		1.34G		1009.25	
ShuffleNetV2 0.5×	1.37	1.37	0.04G	0.04G	5404.06	5434.58
ExpandNet-CL	1.41		0.6G		4228.10	
YOLO-LITE	0.57	0.57	1.81G	1.81G	7.94	19.82
ExpandNet-CL	4.48		28.59G		6.07	
U-Net	7.76	7.76	389.26G	389.26G	8.25	8.25
ExpandNet-CL	82.97		2586.02G		2.98	

Computational overheads and complexity analysis.

To evaluate the influence of r on the complexity of training, we report the number of parameters, MACs and wall-clock training time of a SmallNet with kernel size 7 on CIFAR-10 on a single 12G TITAN V. As shown in Table 2.7, our expansion strategies better leverage GPU computation, thus leading to only moderate wall-clock time increases as r grows, particularly for our CK strategy. We also provide further comparisons of the complexity of our expanded networks and of the original ones in terms of number of parameters, MACs and GPU speed with full use of GPUs for both training and testing in Table 2.8. During training, because our

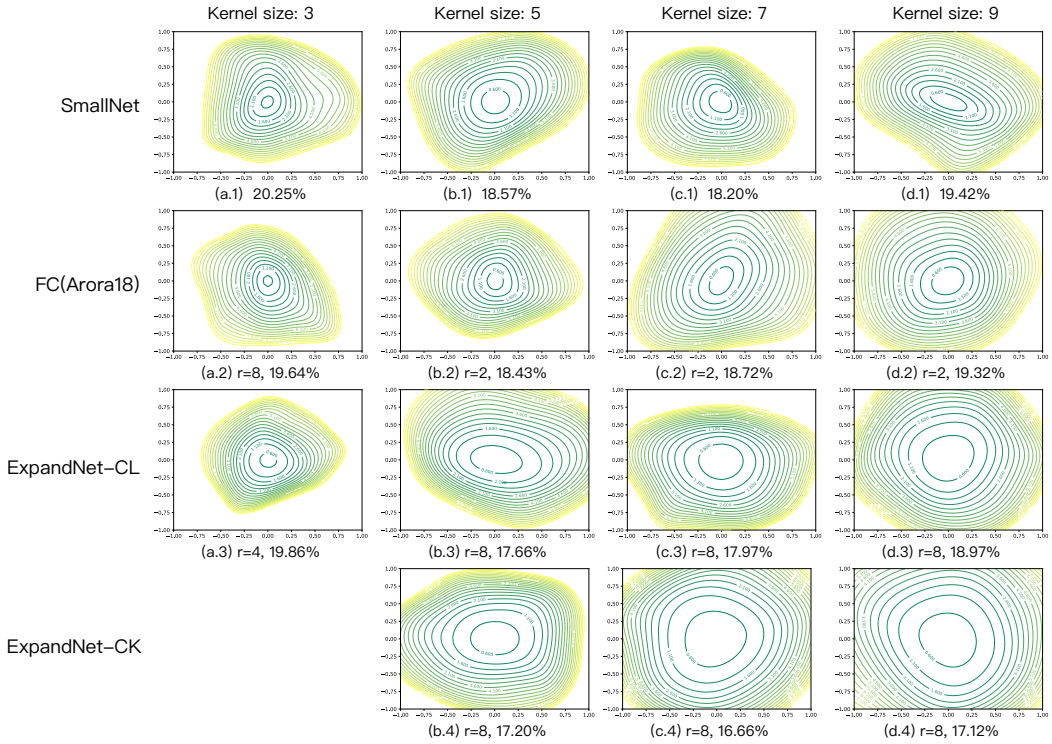


Figure 2.5: **Loss landscape** plots on CIFAR-10 (We report top-1 error (%)).

approach introduces more parameters, inference is 2 to 5 times slower than in the original network for an expansion rate of 4. Overall, as for very compact networks, our ExpandNets better exploit the GPU to make full use of its capacity, leading to similar training time to the original networks. For larger networks, such as MobileNets in Table 2.3, the GPU usage saturates, and thus the training time of ExpandNets increases. Nevertheless, since our ExpandNets can be contracted back to the original network, at test time, they have exactly the same number of parameters, MACs and inference time as the original networks, but our networks achieve better performance.

2.5.2 Generalization Ability

We then analyse the generalization ability of our approach. To this end, we first study the loss landscapes using the method in (Li et al., 2018). We plot the loss landscapes of SmallNets and corresponding ExpandNets on CIFAR-10 in Figure 2.5, showing that our ExpandNets with CL and CK expansion produce flatter minima, which, as discussed in (Li et al., 2018), indicates better generalization.

As a second study of generalization, we evaluate the memorization ability of our ExpandNets on corrupted datasets, as suggested by Zhang et al. (2017). To this end, we utilize the open-source implementation of (Zhang et al., 2017) to generate three CIFAR-10 and CIFAR-100 training sets, containing 20%, 50% and 80% of random labels, respectively, while the test set remains clean.

Table 2.9: **Generalization ability on Corrupted CIFAR-10.** We report the top-1 error (%). Note that our ExpandNets yield smaller generalization errors than the compact network in almost all cases involving convolutional expansion. By contrast expanding FC layers often does not help.

Dataset	Model	Kernel size k = 3			Kernel size k = 5		
		Best Test	Last Test	Train	Best Test	Last Test	Train
20%	SmallNet	22.20 ± 0.43	22.33 ± 0.40	34.85 ± 0.18	20.90 ± 0.16	21.09 ± 0.20	32.05 ± 0.31
	FC(Arora18)	22.40 ± 0.29	22.61 ± 0.27	35.12 ± 0.07	20.87 ± 0.29	21.06 ± 0.26	32.04 ± 0.12
	ExpandNet-CL	**21.55 ± 0.27**	**21.71 ± 0.30**	34.89 ± 0.26	20.47 ± 0.46	20.62 ± 0.43	31.80 ± 0.23
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**19.42 ± 0.20**	**19.63 ± 0.17**	31.55 ± 0.25
50%	SmallNet	25.74 ± 0.25	25.94 ± 0.15	56.48 ± 0.25	25.38 ± 0.45	25.68 ± 0.52	54.49 ± 0.41
	FC(Arora18)	25.54 ± 0.47	25.80 ± 0.41	56.37 ± 0.15	25.36 ± 0.63	25.71 ± 0.77	54.44 ± 0.08
	ExpandNet-CL	**25.48 ± 0.35**	**25.66 ± 0.43**	56.41 ± 0.33	24.27 ± 0.33	24.63 ± 0.44	54.29 ± 0.24
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**22.82 ± 0.27**	**23.00 ± 0.29**	53.93 ± 0.23
80%	SmallNet	37.49 ± 0.62	37.87 ± 0.63	77.46 ± 0.16	37.99 ± 0.64	39.33 ± 0.75	76.14 ± 0.15
	FC(Arora18)	37.26 ± 0.16	37.63 ± 0.14	77.54 ± 0.07	38.35 ± 0.59	39.61 ± 0.87	76.51 ± 0.15
	ExpandNet-CL	**35.86 ± 0.43**	**36.05 ± 0.44**	**77.56 ± 0.11**	36.75 ± 0.64	38.08 ± 0.50	76.09 ± 0.11
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**33.29 ± 1.04**	**34.24 ± 0.85**	75.77 ± 0.22

Dataset	Model	Kernel size k = 7			Kernel size k = 9		
		Best Test	Last Test	Train	Best Test	Last Test	Train
20%	SmallNet	21.64 ± 0.36	21.98 ± 0.42	30.42 ± 0.32	22.56 ± 0.39	22.93 ± 0.18	29.61 ± 0.36
	FC(Arora18)	21.92 ± 0.23	22.35 ± 0.43	30.39 ± 0.19	22.95 ± 0.39	23.48 ± 0.38	29.83 ± 0.34
	ExpandNet-CL	21.25 ± 0.41	21.54 ± 0.40	30.36 ± 0.24	22.13 ± 0.49	22.73 ± 0.53	29.76 ± 0.19
	ExpandNet-CK	**19.11 ± 0.33**	**19.30 ± 0.35**	**31.14 ± 0.11**	**19.32 ± 0.31**	**19.55 ± 0.30**	**31.65 ± 0.17**
50%	SmallNet	26.99 ± 0.69	27.87 ± 0.71	53.27 ± 0.21	28.64 ± 0.46	30.44 ± 0.57	52.67 ± 0.45
	FC(Arora18)	26.86 ± 0.46	28.23 ± 0.61	53.14 ± 0.20	28.46 ± 0.43	30.89 ± 0.38	52.51 ± 0.36
	ExpandNet-CL	26.05 ± 0.31	26.99 ± 0.15	53.21 ± 0.16	27.42 ± 0.35	29.28 ± 0.50	52.67 ± 0.27
	ExpandNet-CK	**22.43 ± 0.47**	**22.61 ± 0.49**	**53.74 ± 0.16**	**22.77 ± 0.14**	**22.99 ± 0.15**	**54.37 ± 0.12**
80%	SmallNet	39.08 ± 0.41	43.33 ± 0.77	74.69 ± 0.26	41.73 ± 0.58	47.96 ± 1.07	74.01 ± 0.32
	FC(Arora18)	40.51 ± 0.39	44.82 ± 0.62	75.38 ± 0.23	42.31 ± 0.46	49.36 ± 1.44	74.59 ± 0.35
	ExpandNet-CL	39.40 ± 0.93	42.77 ± 0.96	75.24 ± 0.22	41.44 ± 0.46	46.75 ± 0.49	74.46 ± 0.08
	ExpandNet-CK	**32.62 ± 0.28**	**33.86 ± 0.37**	**75.65 ± 0.16**	**33.29 ± 0.58**	**33.75 ± 0.49**	**76.27 ± 0.23**

In Tables 2.9 and 2.10 by using different networks with kernel sizes of 3, 5, 7, 9, respectively, we report the top-1 test errors of the best model and of the one after the last epoch, as well as the training errors of the last model. Our method consistently improves the generalization error gap across all kernel sizes and corruption rates (20%, 50%, 80%) and yields from around 1pp to over 6pp error drop in testing. These results evidence that CL and CK expansion typically yields lower test errors and higher training ones, which implies that our better results in the other experiments are not due to simply memorizing the datasets, but truly to better generalization ability.

Table 2.10: **Generalization ability on Corrupted CIFAR-100.** We report the top-1 error (%). Note that our ExpandNets yield smaller generalization errors than the compact network in almost all cases involving convolutional expansion. By contrast expanding FC layers often does not help.

Dataset	Model	Kernel size k					
		3			5		
		Best Test	Last Test	Train	Best Test	Last Test	Train
20%	SmallNet	55.30 ± 0.42	55.48 ± 0.41	62.20 ± 0.31	53.95 ± 0.33	54.16 ± 0.34	58.53 ± 0.30
	FC(Arora18)	56.15 ± 0.22	56.35 ± 0.23	62.60 ± 0.19	54.84 ± 0.71	55.05 ± 0.76	59.47 ± 0.32
	ExpandNet-CL	**54.85 ± 0.27**	**55.04 ± 0.33**	61.62 ± 0.43	53.50 ± 0.35	53.71 ± 0.38	58.09 ± 0.31
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**51.98 ± 0.28**	**52.10 ± 0.26**	57.67 ± 0.63
50%	SmallNet	62.54 ± 0.74	62.71 ± 0.75	78.81 ± 0.39	61.84 ± 0.29	62.16 ± 0.21	76.78 ± 0.28
	FC(Arora18)	63.65 ± 0.50	63.88 ± 0.47	79.40 ± 0.18	62.99 ± 0.69	63.21 ± 0.60	77.85 ± 0.31
	ExpandNet-CL	**61.95 ± 0.61**	**62.11 ± 0.59**	78.78 ± 0.48	61.49 ± 0.39	61.70 ± 0.43	76.73 ± 0.26
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**58.96 ± 0.32**	**59.14 ± 0.41**	76.24 ± 0.30
80%	SmallNet	78.35 ± 0.83	78.52 ± 0.86	93.78 ± 0.18	78.59 ± 0.27	78.81 ± 0.35	93.10 ± 0.12
	FC(Arora18)	80.36 ± 0.55	80.47 ± 0.55	94.38 ± 0.12	80.97 ± 0.51	81.15 ± 0.53	94.10 ± 0.16
	ExpandNet-CL	79.44 ± 0.72	79.66 ± 0.75	94.02 ± 0.16	79.87 ± 0.29	80.04 ± 0.29	93.59 ± 0.20
	ExpandNet-CK	*N/A*	*N/A*	*N/A*	**77.22 ± 0.47**	**77.38 ± 0.41**	93.15 ± 0.25

Dataset	Model	Kernel size k					
		7			9		
		Best Test	Last Test	Train	Best Test	Last Test	Train
20%	SmallNet	55.36 ± 0.44	55.66 ± 0.43	56.33 ± 0.51	56.59 ± 0.72	57.26 ± 0.64	55.16 ± 0.32
	FC(Arora18)	56.31 ± 0.78	56.58 ± 0.77	57.93 ± 0.29	57.82 ± 0.23	58.07 ± 0.22	57.09 ± 0.52
	ExpandNet-CL	54.87 ± 0.47	55.22 ± 0.55	55.52 ± 0.49	56.05 ± 0.75	56.51 ± 0.76	54.99 ± 0.48
	ExpandNet-CK	**51.24 ± 0.60**	**51.40 ± 0.66**	**56.40 ± 0.21**	**52.36 ± 0.54**	**52.55 ± 0.47**	**57.76 ± 0.50**
50%	SmallNet	63.76 ± 0.59	64.08 ± 0.58	75.45 ± 0.23	64.83 ± 0.41	65.63 ± 0.40	75.21 ± 0.31
	FC(Arora18)	64.54 ± 0.72	64.91 ± 0.55	76.75 ± 0.39	66.11 ± 0.45	66.73 ± 0.41	76.44 ± 0.40
	ExpandNet-CL	63.36 ± 0.49	63.73 ± 0.54	75.25 ± 0.45	64.36 ± 0.54	65.25 ± 0.37	74.84 ± 0.28
	ExpandNet-CK	**58.74 ± 0.25**	**58.98 ± 0.20**	75.24 ± 0.24	**60.42 ± 0.86**	**60.65 ± 0.86**	**76.73 ± 0.40**
80%	SmallNet	79.73 ± 0.47	79.95 ± 0.36	92.58 ± 0.20	81.02 ± 0.92	81.70 ± 0.97	92.54 ± 0.26
	FC(Arora18)	82.97 ± 0.83	83.20 ± 0.83	94.13 ± 0.34	83.42 ± 0.72	83.82 ± 0.71	93.94 ± 0.35
	ExpandNet-CL	80.79 ± 0.54	81.09 ± 0.62	93.22 ± 0.30	81.02 ± 0.44	81.58 ± 0.46	93.25 ± 0.56
	ExpandNet-CK	**78.51 ± 0.41**	**78.64 ± 0.36**	**93.24 ± 0.15**	**80.15 ± 0.50**	**80.32 ± 0.55**	**94.04 ± 0.21**

2.5.3 Is Over-parameterization the Key to the Success?

In the previous experiments, we have shown the good training behavior and generalization ability of our expansion strategies. Below, we explore and reject two hypotheses other than over-parameterization that could be thought to explain our better results.

Hypothesis 1: The improvement comes from the different initialization resulting from expansion.

The standard (e.g., Kaiming) initialization of our ExpandNets is in fact equivalent to a non-standard initialization of the compact network. In other words, an alternative would consist of initializing the compact network with an untrained algebraically-contracted ExpandNet. To investigate the influence of such different initialization schemes, we conduct several experiments on CIFAR-10, CIFAR-100 and ImageNet.

The results are provided in Table 2.11. On CIFAR-10, the compact networks initialized with FC(Arora18) and ExpandNet-CL yield slightly better results than training the corresponding ExpandNets. However, the same trend does not occur on CIFAR-100 and ImageNet, where, with

Table 2.11: Top-1 accuracy (%) of compact networks initialized with different ExpandNets on CIFAR-10, CIFAR-100 and ImageNet.

Model	Initialization	CIFAR-10	CIFAR-100
SmallNet	Standard	78.63 ± 0.41	46.63 ± 0.27
	FC(Arora18)	79.09 ± 0.56	46.52 ± 0.36
	ExpandNet-CL	78.65 ± 0.36	46.65 ± 0.47
	ExpandNet-CL+FC	78.81 ± 0.52	46.43 ± 0.72
	ExpandNet-CK	78.84 ± 0.30	46.56 ± 0.23
	ExpandNet-CK+FC	79.27 ± 0.29	46.62 ± 0.29
ExpandNet-CK+FC	Standard	$\mathbf{80.31 \pm 0.27}$	$\mathbf{48.62 \pm 0.47}$

Model	Initialization	ImageNet
MobileNet	Standard	66.48
MobileNet	ExpandNet-CL	66.44
ExpandNet-CL	Standard	**69.40**
MobileNetV2	Standard	63.75
MobileNetV2	ExpandNet-CL	63.07
ExpandNet-CL	Standard	**65.62**
ShuffleNetV2 0.5×	Standard	56.89
ShuffleNetV2 0.5×	ExpandNet-CL	56.91
ExpandNet-CL	Standard	**57.38**

Table 2.12: Top-1 accuracy (%) of SmallNet with 7×7 kernels vs ExpandNets with different rs on CIFAR-10 and CIFAR-100.

r	#params(K)†	CIFAR-10	CIFAR-100
0.25	37.91/43.76	72.32 ± 0.62	39.23 ± 0.84
0.50	42.81/48.66	76.77 ± 0.36	43.68 ± 0.51
0.75	48.43/54.28	78.70 ± 0.42	46.41 ± 0.52
1.00	54.77/60.62	79.22 ± 0.52	47.25 ± 0.40
SmallNet	66.19/72.04	78.63 ± 0.41	46.63 ± 0.27
2.0	87.32/93.17	79.97 ± 0.18	48.13 ± 0.42
4.0	187.0/192.8	$\mathbf{80.27 \pm 0.24}$	$\mathbf{48.55 \pm 0.51}$

† #params(K) denotes the number of parameters (CIFAR-10 / CIFAR-100).

ExpandNet initialization, MobileNet, MobileNetV2 and ShuffleNetV2 0.5× reach results similar to or worse than standard initialization, while training ExpandNet-CL always outperforms the baselines. Moreover, the compact networks initialized by ExpandNet-CK always yield worse results than training ExpandNets-CK from scratch. This confirms that our results are not due to a non-standard initialization.

Hypothesis 2: The improvement is due to an intrinsic property of the CK expansion.

The amount of over-parameterization is directly related to the expansion rate r. Therefore, if some property of the CK strategy was the sole reason for our better results, and not over-parameterization, setting $r \leq 1$ should be sufficient. To study this, we follow the same experimental setting as for Table 2.2 but set $r \in \{0.25, 0.50, 0.75, 1.0, 2.0, 4.0\}$. As shown in Table 2.12, for $r < 1$, the performance of ExpandNet-CK drops by 6.38pp on CIFAR-10 and by 7.09pp on CIFAR-100 as the number of parameters decreases. For $r > 1$, ExpandNet-CK consistently outperforms SmallNet. Interestingly, with $r = 1$, ExpandNet-CK still yields better performance. This shows that our method benefits from both ExpandNet-CK and over-parameterization.

2.6 Ablation Study

In this part, we conduct multiple ablation study to further evidence the effectiveness of our proposed ExpandNets. We provide an ablation study to analyze the influence of different expansion strategies and expansion rates.

2.6.1 Hyper-parameter Choices

In this section, we evaluate the influence of the hyper-parameters of our approach, i.e., the expansion rate r and the kernel size k. We study the behavior of our different expansion strategies, FC, CL and CK, separately, when varying the expansion rate $r \in \{2, 4, 8\}$ and the kernel size $k \in \{3, 5, 7, 9\}$. Compared to our previous CIFAR-10 and CIFAR-100 experiments, we use a deeper network with an extra convolutional layer with 64 channels, followed by a batch normalization layer, a ReLU layer and a 2×2 max pooling layer. We use SGD with a momentum of 0.9 and a weight decay of 0.0005 for 150 epochs. The initial learning rate was 0.01, divided by 10 at epoch 50 and 100. Furthermore, we used zero-padding to keep the size of the input and output feature maps of each convolutional layer unchanged.

The results of these experiments are provided in Table 2.13. We observe that our different strategies to expand convolutional layers outperform the compact network in almost all cases, while only expanding fully-connected layers doesn't work well. In particular, for kernel sizes $k > 3$, ExpandNet-CK yields consistently higher accuracy than the corresponding compact network, independently of the expansion rate. For $k = 3$, where ExpandNet-CK is not applicable, ExpandNet-CL comes as an effective alternative, also consistently outperforming the baseline. In almost all cases, the performance of convolutional expansions improves as the expansion rate increases.

2.6.2 Initializing ExpandNets

As demonstrated by our experiments in Section 2.4, training an ExpandNet from scratch yields consistently better results than training the original compact network. However, with deep

Table 2.13: **Small networks vs ExpandNets on CIFAR-10 (Top) and CIFAR-100 (Bottom).** We report the top-1 accuracy for the original compact networks and for different versions of our approach. Note that our ExpandNets yield higher accuracy than the compact network in almost all cases involving expanding convolutions. By contrast expanding FC layers does often not help.

Model	r	Kernel size k			
		3	5	7	9
SmallNet		79.34 ± 0.42	81.25 ± 0.14	81.44 ± 0.20	80.08 ± 0.48
FC(Arora18)	2	79.13 ± 0.47	81.26 ± 0.33	80.98 ± 0.25	80.43 ± 0.22
	4	78.92 ± 0.36	81.13 ± 0.46	80.85 ± 0.24	80.13 ± 0.29
	8	79.64 ± 0.41	81.21 ± 0.18	80.75 ± 0.45	80.16 ± 0.16
ExpandNet-CL	2	79.46 ± 0.21	81.50 ± 0.31	81.30 ± 0.30	80.26 ± 0.66
	4	$\mathbf{79.90 \pm 0.21}$	81.60 ± 0.15	81.15 ± 0.36	80.62 ± 0.32
	8	79.78 ± 0.20	$\mathbf{81.75 \pm 0.40}$	$\mathbf{81.53 \pm 0.33}$	$\mathbf{80.78 \pm 0.25}$
ExpandNet-CK	2	N/A	81.72 ± 0.31	82.19 ± 0.24	81.60 ± 0.11
	4	N/A	82.34 ± 0.43	82.34 ± 0.22	81.73 ± 0.33
	8	N/A	$\mathbf{82.37 \pm 0.25}$	$\mathbf{82.84 \pm 0.28}$	$\mathbf{82.53 \pm 0.30}$
SmallNet		48.14 ± 0.29	50.44 ± 0.07	49.62 ± 0.50	48.70 ± 0.38
FC(Arora18)	2	47.21 ± 0.46	48.39 ± 0.77	47.88 ± 0.41	46.36 ± 0.34
	4	47.44 ± 0.66	48.92 ± 0.47	48.43 ± 0.56	46.90 ± 0.34
	8	47.55 ± 0.25	49.44 ± 0.65	48.66 ± 0.49	47.15 ± 0.28
ExpandNet-CL	2	47.68 ± 0.85	50.39 ± 0.45	49.78 ± 0.33	48.68 ± 0.70
	4	48.25 ± 0.13	50.68 ± 0.27	49.81 ± 0.31	$\mathbf{48.87 \pm 0.65}$
	8	$\mathbf{48.93 \pm 0.13}$	$\mathbf{50.95 \pm 0.42}$	$\mathbf{49.95 \pm 0.37}$	48.85 ± 0.42
ExpandNet-CK	2	N/A	51.18 ± 0.44	51.09 ± 0.41	50.40 ± 0.35
	4	N/A	$\mathbf{52.13 \pm 0.36}$	51.82 ± 0.67	50.62 ± 0.65
	8	N/A	52.05 ± 0.59	$\mathbf{52.48 \pm 0.54}$	$\mathbf{51.57 \pm 0.15}$

networks, initialization can have an important effect on the final results. While designing an initialization strategy specifically for compact networks is an unexplored research direction, our ExpandNets can be initialized in a natural manner. To this end, we exploit the fact that an ExpandNet has a natural nonlinear counterpart, which can be obtained by incorporating a nonlinear activation function between each pair of linear layers. We therefore propose to initialize the parameters of an ExpandNet by simply training its nonlinear counterpart and transferring the resulting parameters to the ExpandNet. The initialized ExpandNet is then trained in the standard manner.

We applied this initialization scheme to the SmallNets with 7×7 and 3×3 kernels used in our CIFAR-10 and CIFAR-100 experiments, and report the results in Table 2.14, respectively, where *+Init* denotes the use of our initialization strategy. We also report the result of this initialization scheme on object detection in Table 2.15.

Note that this strategy yields an additional accuracy boost to our approach. In particular, since

Table 2.14: Top-1 accuracy (%) of SmallNet vs ExpandNets with the initialization with $r = 4$ on CIFAR-10 and CIFAR-100 with 3×3 kernels and with 7×7 kernels(bottom).

Model	Transfer	CIFAR-10	CIFAR-100
SmallNet (3×3)	*w/o KD*	73.32 ± 0.20	40.40 ± 0.60
SmallNet (3×3)	*w/ KD*	73.34 ± 0.31	40.46 ± 0.56
ExpandNet-CL	*w/o KD*	73.96 ± 0.30	40.91 ± 0.47
ExpandNet-CL+FC		74.45 ± 0.29	41.12 ± 0.49
ExpandNet-CL+FC+Init		$\mathbf{75.16 \pm 0.23}$	$\mathbf{42.41 \pm 0.21}$
ExpandNet-CL+FC	*w/ KD*	74.52 ± 0.37	41.51 ± 0.49
ExpandNet-CL+FC+Init		$\mathbf{75.17 \pm 0.51}$	$\mathbf{42.67 \pm 0.67}$
SmallNet (7×7)	*w/o* KD	78.63 ± 0.41	46.63 ± 0.27
SmallNet (7×7)	*w/* KD	78.97 ± 0.37	47.04 ± 0.35
ExpandNet-CL+FC	*w/o KD*	79.11 ± 0.23	46.66 ± 0.43
ExpandNet-CL+FC+Init		79.98 ± 0.28	47.98 ± 0.48
ExpandNet-CK+FC		80.31 ± 0.27	48.62 ± 0.47
ExpandNet-CK+FC+Init		$\mathbf{80.81 \pm 0.27}$	$\mathbf{49.82 \pm 0.25}$
ExpandNet-CL+FC	*w/* KD	79.60 ± 0.25	47.41 ± 0.51
ExpandNet-CL+FC+Init		80.29 ± 0.25	48.62 ± 0.34
ExpandNet-CK+FC		80.63 ± 0.31	49.13 ± 0.45
ExpandNet-CK+FC+Init		$\mathbf{81.21 \pm 0.17}$	$\mathbf{50.37 \pm 0.39}$

YOLO-LITE is very compact, this scheme boosts performance by more than 4pp.

Note that, on ImageNet and Cityscapes, the nonlinear counterparts of the ExpandNets did not outperform the ExpandNets, and thus we did not use our initialization strategy. As a general rule, when the nonlinear counterparts achieves better performance than the ExpandNets, we recommend using them for initialization. This suggests interesting directions for future research on the initialization of our ExpandNets and of compact networks in general.

Table 2.15: YOLO-LITE vs ExpandNets with $r = 4$ on the PASCAL VOC2007 test set.

Model	mAP(%)
YOLO-LITE	27.34
ExpandNet-CL	30.97
ExpandNet-CL+Init	**35.14**

2.6.3 Knowledge Transfer with ExpandNets

In Section 2.4, we claim that our ExpandNet strategy is complementary to knowledge transfer. Following (Passalis and Tefas, 2018), on CIFAR-10, we make use of the ResNet18 as teacher. Furthermore, we also use the same compact network with kernel size 3×3 and training setting

Table 2.16: **Knowledge transfer from the ResNet18 on CIFAR-10.** Using ExpandNets as student networks yields consistently better results than directly using SmallNet.

Network	Transfer	Top-1 Accuracy
SmallNet	Baseline	73.32 ± 0.20
SmallNet	KD	73.34 ± 0.31
	Hint	33.71 ± 4.35
	PKT	68.36 ± 0.35
ExpandNet (CL+FC)	KD	74.52 ± 0.37
	Hint	52.46 ± 2.43
	PKT	70.97 ± 0.70
ExpandNet (CL+FC+Init)	KD	$\mathbf{75.17 \pm 0.51}$
	Hint	58.27 ± 3.83
	PKT	71.65 ± 0.41

Table 2.17: **Top-1 accuracy (%) of AlexNet vs ExpandNets with $r = 4$ on the ILSVRC2012 validation set for different number of channels in the last convolutional layer.** Note that, while our expansion strategy always helps, its benefits decrease as the original model grows.

# Channels	128	256 (Original)	512
Baseline	46.72	54.08	58.35
ExpandNet-CK	**49.66**	**55.46**	**58.75**
↑	2.94	1.38	0.4

as in (Passalis and Tefas, 2018). In Table 2.16, we compare the results of different knowledge transfer strategies, including knowledge distillation (KD) (Hinton et al., 2015), hint-based transfer (Hint)(Romero et al., 2014) and probabilistic knowledge transfer (PKT) (Passalis and Tefas, 2018), applied to the compact network and to our ExpandNets, without and with our initialization scheme. Note that using knowledge transfer with our ExpandNets, with and without initialization, consistently outperforms using it with the compact network. Altogether, we therefore believe that, to train a given compact network, one should really use both knowledge transfer and our ExpandNets to obtain the best results.

2.6.4 Working with Larger Networks

We also evaluate the use of our approach with a larger network. To this end, we make use of AlexNet (Krizhevsky et al., 2012a) on ImageNet. AlexNet relies on kernels of size 11 and 5 in its first two convolutional layers, which makes our CK expansion strategy applicable.

We use a modified, more compact version of AlexNet, where we replace the first fully-connected layer with a global average pooling layer, followed by a 1000-class fully-connected layer with softmax. To evaluate the impact of the network size, we explore the use of different dimen-

sions, [128, 256, 512], for the final convolutional features. We trained the resulting AlexNets and corresponding ExpandNets using the same training regime as for our MobileNets experiments in Section 2.4.

As shown in Table 2.17, while our approach outperforms the baseline AlexNets for all feature dimensions, the benefits decrease as the feature dimension increases. This indicates that our approach is better suited for truly compact networks, and developing similar strategies for deeper ones will be the focus of our future research.

2.7 Discussion

In this section, we discuss in more detail the two works that are most closely related to ours. These two works also evidence the benefits of linear over-parameterization, thus strengthening our argument, but differ significantly from ours in terms of specific strategy. Note that, as shown by our experiments, our approach outperforms theirs. Furthermore, we also provide a concrete example of the matrix representation of a convolution operator and provide the source code in Appendix A.

2.7.1 Discussion of (Arora et al., 2018)

Arora et al. (2018) worked mostly with purely linear, fully-connected models, with only one example using a nonlinear model, where again only the fully-connected layer was expanded. By contrast, we focus on *practical, nonlinear, compact convolutional* networks, and we propose two ways to expand convolutional layers, which have not been studied before. As shown by our experiments and ablation study, our convolutional linear expansion strategies yield better solutions than vanilla training, with higher accuracy, more zero-centered gradient cosine similarity during training and minima that generalize better. This is in general not the case when expanding the fully-connected layers only, as proposed by Arora et al. (2018). Furthermore, in contrast with (Arora et al., 2018), who only argue that depth speeds up convergence, we empirically show, by using different expansion rates, that increasing width helps to reach better solutions. We now discuss in more detail the only experiment in (Arora et al., 2018) with a nonlinear network.

In their paper, Arora et al. (2018) performed a sanity test on MNIST with a CNN, but only expanding the fully-connected layer. According to our experiments, expanding fully-connected layers only (denoted as FC(Arora18) in our results) is typically insufficient to outperform vanilla training of the compact network. This was confirmed by using their code, with which we found that, in their setting, the over-parameterized model yields higher test error. We acknowledge that Arora et al. (2018) did not claim that expansion led to better results but sped up convergence. Nevertheless, this seemed to contradict our experiments, in which our FC expansion was achieving better results than that of (Arora et al., 2018).

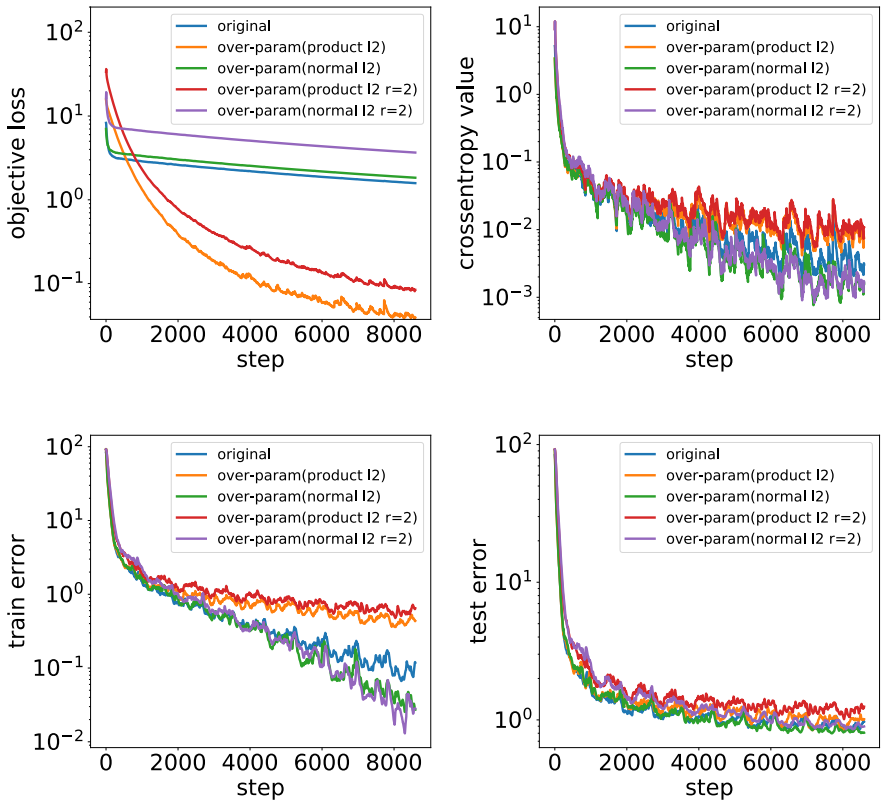


Figure 2.6: **Product L^2 vs Normal L^2** (best viewed in color). *Top Left:* Training curves of the overall loss function. *Top Right:* Training curves of the cross-entropy. *Bottom Left:* Curves of training errors. *Bottom Right:* Curves of test errors. (Note that the y-axis is in log scale.)

While analyzing the reasons for this, we found that Arora et al. (2018) used a different weight decay regularizer than us. Specifically, considering a single fully-connected layer expanded into two, this regularizer is defined as

$$L_r = \|\tilde{\boldsymbol{W}}_{fc}\|_2^2 = \|\boldsymbol{W}_{fc_1}\boldsymbol{W}_{fc_2}\|_2^2\,, \tag{2.5}$$

where $\boldsymbol{W}_{fc_1}$ and $\boldsymbol{W}_{fc_2}$ represent the parameter matrices of the two fully-connected layers after expansion. That is, the regularizer is defined over the *product* of these parameter matrices. While this corresponds to weight decay on the original parameter matrix, without expansion, it contrasts with usual weight decay, which sums over the different parameter matrices, yielding a regularizer of the form

$$L_r = \|\boldsymbol{W}_{fc_1}\|_2^2 + \|\boldsymbol{W}_{fc_2}\|_2^2\,. \tag{2.6}$$

The product L^2 norm regularizer used by Arora et al. (2018) imposes weaker constraints on the individual parameter matrices, and we observed their over-parameterized model to converge to a worse minimum and lead to worse test performance when used in a nonlinear CNN.

To evidence this, in Figure 2.6, we compare the original model with an over-parameterized one

relying on a product L^2 regularizer as in (Arora et al., 2018), and with an over-parameterized network with normal L^2 regularization, corresponding to our FC expansion strategy. Even though the overall loss of Arora et al. (2018)'s over-parameterized model decreases faster than that of the baseline, the cross-entropy loss term, the training error and the test error do not show the same trend. The test errors of the original model, Arora et al. (2018)'s over-parameterized model with product L^2 norm and our ExpandNet-FC with normal L^2 norm are 0.9%, 1.1% and 0.8%, respectively. Furthermore, we also compare Arora et al. (2018)'s over-parameterized model and our ExpandNet-FC with an expansion rate $r = 2$. We observe that Arora et al. (2018)'s over-parameterized model performs even worse with a larger expansion rate, while our ExpandNet-FC works well.

Note that, in the experiments and ablation study part above, the models denoted by FC(Arora18) rely on a normal L^2 regularizer, which we observed to yield better results and makes the comparison fair as all models then use the same regularization strategy.

2.7.2 Discussion of ACNet (Ding et al., 2019)

The work of Ding et al. (2019), concurrent to ours, also proposed a form of expansion of convolutions. Specifically, as shown in Figure 2.7, their approach consists of replacing a convolutional layer with $k \times k$ kernels with three parallel layers: One with the same square $k \times k$ kernel, and two with 1D asymmetric convolutions of size $1 \times k$ and $k \times 1$. These three different convolutions are then applied in parallel on the same input feature map, and their outputs are combined via addition.

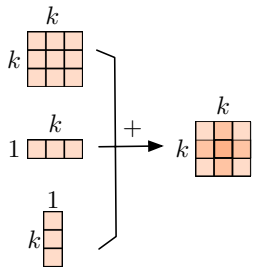


Figure 2.7: **One ACNet block** (best viewed in color).

As argued in (Ding et al., 2019), the goal of this operation is to increase the representation power of a standard square kernel by strengthening the kernel skeletons. While effective, the over-parameterization resulting from this approach remains limited; by using 1D convolutions in parallel to the original ones, it can only add $2kmn$ parameters for every $k \times k$ kernel with m input and n output channels. By contrast, by incorporating new convolutional layers in a serial manner, we can modify the number of channels of the intermediate layers so as to increase the number of parameters of the network much more drastically, and in a much more flexible way, thanks to our expansion rate. Specifically, with an expansion rate r, our CL expansion strategy yield $rm^2 + k^2r^2mn + rn^2$ parameters instead of k^2mn for the original convolution. Ultimately, while ACNet can indeed improve the image classification performance, as shown in (Ding et al., 2019) and confirmed by our experiments, the greater flexibility of our approach yields significantly better results, particularly for networks relying on depthwise convolutions, as evidenced by our ImageNet results, and networks with kernel sizes larger than 3, as evidenced by our results with a SmallNet with 7×7 kernels. Furthermore, note that, in contrast to Arora

et al. (2018) and Ding et al. (2019), we also demonstrate the effectiveness of our expansion strategy on object detection and semantic segmentation.

2.7.3 Matrix Representation of a Convolution Operator

We provide an example of the matrix representation of a convolutional layer, following Eq. 2.1 in the Section 2.3. Given an input $\mathbf{X}_{1\times1\times3\times3}$ and convolutional filters $\mathbf{F}_{1\times1\times2\times2}$, expressed as

$$\mathbf{X}_{1\times1\times3\times3} = \begin{bmatrix}\begin{bmatrix}\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}\end{bmatrix}\end{bmatrix}, \ \mathbf{F}_{1\times1\times2\times2} = \begin{bmatrix}\begin{bmatrix}\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}\end{bmatrix}\end{bmatrix}, \tag{2.7}$$

the matrix representation of a convolution can be obtained by vectorizing the input as

$$\boldsymbol{X}^{v}_{9\times1} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{bmatrix}^{T}, \tag{2.8}$$

and by defining a highly-structured matrix containing the filters as

$$\boldsymbol{W}^{\mathbf{F}}_{4\times9} = \begin{bmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 \\ 0 & 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} \end{bmatrix}. \tag{2.9}$$

Then, the convolution operation $(*)$ can be equivalently written as

$$\mathbf{Y}_{1\times1\times2\times2} = \mathbf{X}_{1\times1\times3\times3} * \mathbf{F}_{1\times1\times2\times2} = \text{reshape}(\boldsymbol{W}^{\mathbf{F}}_{4\times9} \times \boldsymbol{X}^{v}_{9\times1}), \tag{2.10}$$

where $\times$ denotes the standard matrix-vector product.

To contract an ExpandNet, one can then compute the matrix product of its expanded layers to obtain a single matrix representing these multiple operations. This matrix can then be transferred back to a standard convolution filter tensor representation following the reverse strategy to the one explained above. For the details of how we contract our ExpandNets in practice, we invite the reader to check our published code (https://github.com/GUOShuxuan/expandnets.). Note that, in our implementation, we take advantage of the Pytorch tensor operators.

2.8 Conclusion

We have introduced an approach to training a given compact network from scratch by exploiting linear over-parameterization in this chapter. Specifically, we have shown that over-parameterizing the network *linearly* facilitates the training of compact networks, particularly

when linearly expanding convolutional layers. Our analysis has further evidenced that over-parameterization is the key to the success of our approach, improving both the training behavior and generalization ability of the networks, and ultimately leading to better performance at inference without any increase in computational cost. Our technique is general and can also be used in conjunction with knowledge transfer approaches to further boost performance. Finally, as shown in our ablation study, initializing an ExpandNet with its trained nonlinear counterpart can further boost its results. This motivates us to investigate the design of other effective initialization schemes for compact networks in the future.

3 Distilling Image Classifiers in Object Detectors

We have shown the generality of our ExpandNets in the former chapter not only on multiple visual recognition tasks, but also used in conjunction with several knowledge distillation approaches. In this chapter, we dive into knowledge distillation itself, focusing on the task of object detection. The knowledge distillation literature remains limited to the scenario where the student and the teacher tackle the same task. In this chapter, we investigate the problem of transferring knowledge not only across architectures but also across tasks. To this end, we study the case of object detection and, instead of following the standard detector-to-detector distillation approach, introduce a classifier-to-detector knowledge transfer framework. In particular, we propose strategies to exploit the classification teacher to improve both the detector's recognition accuracy and localization performance. Our experiments on several detectors with different backbones demonstrate the effectiveness of our approach, allowing us to outperform the state-of-the-art detector-to-detector distillation methods.

The contents of this chapter are mainly from the following paper. I am the primary contributor.

- Guo, S., Alvarez, J. M., and Salzmann, M. (2021b). *Distilling image classifiers in object detectors*. Advances in Neural Information Processing Systems (NeurIPS2021).

3.1 Introduction

Object detection plays a critical role in many real-world applications, such as autonomous driving and video surveillance. While deep learning has achieved tremendous success in this task (Lin et al., 2017b; Liu et al., 2016; Redmon and Farhadi, 2018; Ren et al., 2015; Yang et al., 2019), the speed-accuracy trade-off of the resulting models remains a challenge. This is particularly important for real-time prediction on embedded platforms, whose limited memory and computation power impose strict constraints on the deep network architecture.

To address this, much progress has recently been made to obtain compact deep networks. Existing methods include pruning (Alvarez and Salzmann, 2016, 2017; Han et al., 2016; Lee et al., 2019; Ullrich et al., 2017) and quantization (Courbariaux et al., 2016; Rastegari et al.,

2016; Zhao et al., 2019), both of which aim to reduce the size of an initial deep architecture, as well as knowledge distillation, whose goal is to exploit a deep teacher network to improve the training of a given compact student one. In this chapter, we introduce a knowledge distillation approach for object detection.

While early knowledge distillation techniques (Hinton et al., 2015; Romero et al., 2014; Tian et al., 2020) focused on the task of image classification, several attempts have nonetheless been made for object detection. To this end, existing techniques (Chen et al., 2017b; Guo et al., 2021a; Wang et al., 2019) typically leverage the fact that object detection frameworks consist of three main stages depicted by Figure 3.1(a): A backbone to extract features; a neck to fuse the extracted features; and heads to predict classes and bounding boxes. Knowledge distillation is then achieved using a teacher with the same architecture as the student but a deeper and wider backbone, such as a Faster RCNN (Ren et al., 2015) with ResNet152 (He et al., 2016) teacher for a Faster RCNN with ResNet50 student, thus facilitating knowledge transfer at all three stages of the frameworks. To the best of our knowledge, Zhang and Ma (2021) constitutes the only exception to this strategy, demonstrating distillation across different detection frameworks, such as from a RetinaNet (Lin et al., 2017b) teacher to a RepPoints (Yang et al., 2019) student. This method, however, requires the teacher and the student to rely on a similar detection strategy, i.e., both must be either one-stage detectors or two-stage ones, and, more importantly, still follows a detector-to-detector approach to distillation. In other words, the study of knowledge distillation remains limited to transfer across architectures tackling the same task. Our classification teacher tackles a different task from the detection student and is trained in a different manner but on the same dataset. Therefore, the classification teacher is capable of providing a different knowledge to the student, for both classification and localization, than that extracted by a detection teacher.

In this chapter, we investigate the problem of transferring knowledge not only across architectures but also across tasks. In particular, we observed that the classification head of state-of-the-art object detectors still typically yields inferior performance compared to what can be expected from an image classifier. Thus, as depicted by Figure 3.1(b), we focus on the scenario where the teacher is an image classifier while the student is an object detector. We then develop distillation strategies to improve both the recognition accuracy and the localization ability of the student.

Our contributions in this chapter can thus be summarized as follows:

- We introduce the idea of classifier-to-detector knowledge distillation to improve the performance of a student detector using a classification teacher.
- We propose a distillation method to improve the student's classification accuracy, applicable when the student uses either a categorical cross-entropy loss or a binary cross-entropy one.
- We develop a distillation strategy to improve the localization performance of the student be exploiting the feature maps from the classification teacher.

We demonstrate the effectiveness of our approach on the COCO2017 benchmark (Lin et al., 2014) using diverse detectors, including the relatively large two-stage Faster RCNN and single-stage RetinaNet used in previous knowledge distillation works, as well as more compact detectors, such as SSD300, SSD512 (Liu et al., 2016) and Faster RCNNs(Ren et al., 2015) with lightweight backbones. Our classifier-to-detector distillation approach outperforms the detector-to-detector distillation ones in the presence of compact students, and helps to further boost the performance of detector-to-detector distillation techniques for larger ones, such as Faster RCNN and RetinaNet with a ResNet50 backbone. Our code is avlaible at: https://github.com/NVlabs/DICOD.

3.2 Knowledge Distillation in Object Detection

Object detection is one of the fundamental tasks in computer vision, aiming to localize the objects observed in an image and classify them. Recently, much progress has been made via the development of both one-stage (Duan et al., 2019; Law and Deng, 2018; Liu et al., 2016; Redmon and Farhadi, 2018; Tian et al., 2019) and two-stage (Cai and Vasconcelos, 2018; He et al., 2017a; Lin et al., 2017a; Ren et al., 2015) deep object detection frameworks, significantly improving the mean average precision (mAP) on standard benchmarks (Everingham et al., 2007, 2012; Lin et al., 2014). However, the performance of these models typically increases with their size, and so does their inference runtime. This conflicts with their deployment on embedded platforms, such as mobile phones, drones, and autonomous vehicles, which involve computation and memory constraints. While some efforts have been made to design smaller detectors, such as SSD (Liu et al., 2016), YOLO (Redmon and Farhadi, 2018) and detectors with lightweight backbones (Howard et al., 2017; Sandler et al., 2018), the performance of these methods does not match that of deeper ones.

Knowledge distillation offers the promise to boost the performance of such compact networks by exploiting deeper teacher architectures. Early work in this space focused on the task of image classification. In particular, Hinton et al. (2015) proposed to distill the teacher's class probability distribution into the student, and Romero et al. (2014) encouraged the student's intermediate feature maps to mimic the teacher's ones. These initial works were followed by a rapid growth in the number of knowledge distillation strategies, including methods based on attention maps (Zagoruyko and Komodakis, 2017), on transferring feature flows defined by the inner product of features (Yim et al., 2017), and on contrastive learning to structure the knowledge distilled from teacher to the student (Tian et al., 2020). Heo et al. (2019a) proposed a synergistic distillation strategy aiming to jointly leverage a teacher feature transform, a student feature transform, the distillation feature position and a better distance function.

Compared to image classification, object detection poses the challenge of involving both recognition and localization. As such, several works have introduced knowledge distillation methods specifically tailored to this task. This trend was initiated by Chen et al. (2017b), which proposed to distill knowledge from a teacher detector to a student detector in both

the backbone and head stages. Then, Wang et al. (2019) proposed to restrict the teacher-student feature imitation to regions around positive anchor boxes; Dai et al. (2021) produced general instances based on both the teacher's and student's outputs, and distilled feature-based, relation-based and response-based knowledge in these general instances; Guo et al. (2021a) proposed to decouple the intermediate features and classification predictions of the positive and negative regions during knowledge distillation. All the aforementioned knowledge distillation methods require the student and the teacher to follow the same kind of detection framework, and thus typically transfer knowledge between models that only differ in terms of backbone, such as from a RetinaNet-ResNet152 to a RetinaNet-ResNet50. In (Zhang and Ma, 2021), such a constraint was relaxed via a method able to transfer knowledge across the feature maps of different frameworks. This allowed the authors to leverage the best one-stage, resp. two-stage, teacher model to perform distillation to any one-stage, resp. two-stage, student. This method, however, still assumes that the teacher is a detector.

In short, existing knowledge distillation methods for object detection all follow a detector-to-detector transfer strategy. In fact, to the best of our knowledge, distillation has only been studied across two architectures that tackle the same task, may it be image classification, object detection, or even semantic segmentation (He et al., 2019a; Liu et al., 2019). In this chapter, by contrast, we investigate the use of knowledge distillation across tasks and develop strategies to distill the knowledge of an image classification teacher to an object detection student.

3.3 Classifier-to-detector Distillation

Our goal is to investigate the transfer of knowledge from an image classifier to an object detector. As illustrated in Figure 3.1, this contrasts with existing knowledge distillation techniques for object detection, which typically assume that the teacher and the student both follow a similar three-stage detection pipeline. For our classifier-to-detector knowledge distillation to be effective, we nonetheless need the student and teacher to process the same data and use the same loss for classification. To this end, given a detection dataset $\mathscr{D}_{det}$ depicting C foreground object categories, we construct a classification dataset $\mathscr{D}_{cls}$ by extracting all objects from $\mathscr{D}_{det}$ according to their ground-truth bounding boxes and labels. We then train our classification teacher $\mathscr{F}_t$, with parameters θ^t, on $\mathscr{D}_{cls}$ in a standard classification manner. In the remainder of this section, we introduce our strategies to exploit the resulting teacher to improve both the classification and localization accuracy of the student detector $\mathscr{F}_s$, with parameters θ^s.

3.3.1 KD$_{cls}$: Knowledge Distillation for Classification

Our first approach to classifier-to-detector distillation focuses on the classification accuracy of the student network. To this end, we make use of the class-wise probability distributions obtained by the teacher and the student, softened by making use of a temperature parameter T. Below, we first derive our general formulation for distillation for classification, and then

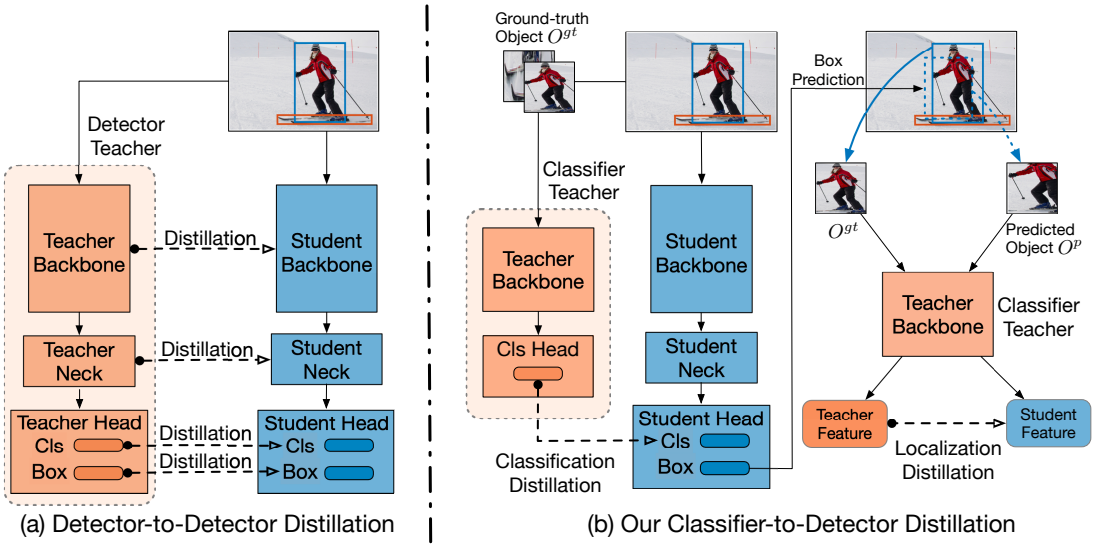


Figure 3.1: **Overview of our classifier-to-detector distillation framework.** (a) Existing methods perform distillation across corresponding stages in the teacher and student, which restricts their applicability to detector-to-detector distillation. (b) By contrast, we introduce strategies to transfer the knowledge from an image classification teacher to an object detection student, improving both its recognition and localization accuracy.

discuss in more detail how we obtain the teacher and student class distributions for the two types of classification losses commonly used by object detection frameworks.

Formally, given K positive anchor boxes or object proposals, which are assigned with one of the ground-truth labels and bounding boxes during training, let $p_k^{s,T}$ denote the vector of softened class probabilities for box k from the student network, obtained at temperature T, and let $p_k^{t,T}$ denote the corresponding softened probability vector from the teacher network. We express knowledge distillation for classification as a loss function measuring the Kullback-Leibler (KL) divergence between the teacher and student softened distributions. This can be written as

$$\mathcal{L}_{kd-cls} = \frac{1}{K}\sum_{k=1}^{K} KL(p_k^{t,T} \parallel p_k^{s,T}) \,. \tag{3.1}$$

The specific way we define the probability vectors $p_k^{s,T}$ and $p_k^{t,T}$ then depends on the loss function that the student detector uses for classification. Indeed, existing detectors follow two main trends: some, such as Faster RCNN and SSD, exploit the categorical cross-entropy loss with a softmax, accouting for the C foreground classes and 1 background one; others, such as RetinaNet, employ a form of binary cross-entropy loss with a sigmoid[1], focusing only on the C foreground classes. Let us now discuss these two cases in more detail.

[1] In essence, the RetinaNet focal loss follows a binary cross-entropy formulation.

Categorical cross-entropy. In this case, for each positive object bounding box k, the student detector outputs logits $z_k^s \in (C+1)$. We then compute the corresponding softened probability for class c with temperature T as

$$p_k^{s,T}(c|\theta^s) = \frac{e^{z_{k,c}^s/T}}{\sum_{j=1}^{C+1} e^{z_{k,j}^s/T}}\,, \tag{3.2}$$

where $z_{k,c}^s$ denote the logit corresponding to class c. By contrast, as our teacher is a C-way classifier, it produces logits $z_k^t \in C$. We thus compute its softened probability for class c as

$$\tilde{p}_k^{t,T}(c|\theta^t) = \frac{e^{z_{k,c}^t/T}}{\sum_{j=1}^{C} e^{z_{k,j}^t/T}}\,, \tag{3.3}$$

and, assuming that all true objects should be classified as background with 0 probability, augment the resulting distribution to account for the background class as $p^{t,T} = [\tilde{p}^{t,T}, 0]$.

The KL-divergence between the teacher and student softened distributions for object k can then be written as

$$KL(p_k^{t,T} \parallel p_k^{s,T}) = T^2 \sum_{c=1}^{C+1} p_{k,c}^{t,T} \log p_{k,c}^{t,T} - p_{k,c}^{t,T} \log p_{k,c}^{s,T}\,. \tag{3.4}$$

Binary cross-entropy. The detectors that rely on the binary cross-entropy output a score between 0 and 1 for each of the C foreground classes, but, together, these scores do not form a valid distribution over the C classes as they do not sum to 1. To nonetheless use them in a KL-divergence measure between the teacher and student, we rely on the following strategy. Given the student and teacher C-dimensional logit vectors for an object k, we compute softened probabilities as

$$\begin{aligned} \tilde{p}_k^{s,T}(c|\theta^s) &= (1 + e^{-z_{k,c}^s/T})^{-1}\,, \\ \tilde{p}_k^{t,T}(c|\theta^t) &= (1 + e^{-z_{k,c}^t/T})^{-1}\,. \end{aligned} \tag{3.5}$$

We then build a 2-class (False-True) probability distribution for each category according to the ground-truth label l of object k. Specifically, for each category c, we write

$$p_{k,c}^{s,T} = [1 - \tilde{p}_{k,c}^{s,T}, \tilde{p}_{k,c}^{s,T}], \tag{3.6}$$

for the student, and similarly for the teacher. This lets us express the KL-divergence for object k as

$$KL(p_k^{t,T} \parallel p_k^{s,T}) = \frac{T^2}{C} \sum_{c=1}^{C} \sum_{i=0}^{1} p_{k,c}^{t,T}(i) \log p_{k,c}^{t,T}(i) - p_{k,c}^{t,T}(i) \log p_{k,c}^{s,T}(i)\,, \tag{3.7}$$

where $p_{k,c}^{t,T}(i)$ indicates the i-th element of the 2-class distribution $p_{k,c}^{t,T}$.

3.3.2 KD$_{loc}$: Knowledge Distillation for Localization

While, as will be shown by our experiments, knowledge distillation for classification already helps the student detector, it does not aim to improve its localization performance. Nevertheless, localization, or bounding box regression, is critical for the success of a detector and is typically addressed by existing detector-to-detector distillation frameworks (Chen et al., 2017b; Dai et al., 2021). To also tackle this in our classifier-to-detector approach, we develop a feature-level distillation strategy, exploiting the intuition that the intermediate features extracted by the classification teacher from a bounding box produced by the student should match those of the ground-truth bounding box.

Formally, given an input image I of size $w \times h$, let us denote by $B_k = (x_1, y_1, x_2, y_2)$ the top-left and bottom-right corners of the k-th bounding box produced by the student network. Typically, this is achieved by regressing the offset of an anchor box or object proposal. We then make use of a Spatial Transformer (Jaderberg et al., 2015) unit to extract the image region corresponding to B_k. It is a non-parametric differentiable module that links the regressed bounding boxes with the classification teacher to yield an end-to-end model during training. Specifically, we compute the transformer matrix

$$A_k = \begin{bmatrix} (x_2 - x_1)/w & 0 & -1 + (x_1 + x_2)/w \\ 0 & (y_2 - y_1)/h & -1 + (y_1 + y_2)/h \end{bmatrix} , \tag{3.8}$$

which allows us to extract the predicted object region O_k^p with a grid sampling size s as

$$O_k^p = f_{ST}(A_k, I, s) , \tag{3.9}$$

where f_{ST} denotes the spatial transformer function. As illustrated in the right portion of Figure 3.1(b), we then perform distillation by comparing the teacher's intermediate features within the predicted object region O_k^p to those within its assigned ground-truth one O_k^{gt}.

Specifically, for a given layer ℓ, we seek to compare the features $\mathscr{F}_t^\ell(O_k^p)$ and $\mathscr{F}_t^\ell(O_k^{gt})$ of the positive box k. To relax the pixel-wise difference between the features, we make use of the adaptive pooling strategy of (McFee et al., 2018), which produces a feature map $AP(\mathscr{F}_t^\ell(O))$ of a fixed size $M \times W \times H$ from the features extracted within region O. We therefore write our localization distillation loss as

$$\mathscr{L}_{kd-loc}^L = \frac{1}{KLMHW} \sum_{k=1}^{K} \sum_{\ell=1}^{L} \mathbb{1}_\ell \| AP(\mathscr{F}_t^\ell(O_k^p)) - AP(\mathscr{F}_t^\ell(O_k^{gt})) \|_1 , \tag{3.10}$$

where K is the number of positive anchor boxes or proposals, L is the number of layers at which we perform distillation, $\mathbb{1}_l$ is the indicator function to denote whether the layer ℓ is used or not to distill knowledge, and $\|\cdot\|_1$ denotes the L_1 norm. As both the spatial transformer and the adaptive pooling operation are differentiable, this loss can be backpropagated through the student detector.

Note that, as a special case, our localization distillation strategy can be employed not only on intermediate feature maps but on the object region itself (ℓ_0), encouraging the student to produce bounding boxes whose underlying image pixels match those of the ground-truth box. This translates to a loss function that does not exploit the teacher and can be expressed as

$$\mathscr{L}^0_{kd-loc}(O^p, O^{gt}) = \frac{1}{KMHW} \sum_{k=1}^{K} \|AP(O^p_k) - AP(O^{gt}_k)\|_1 \,. \tag{3.11}$$

Depending on the output size of the adaptive pooling operation, this loss function encodes a more-or-less relaxed localization error. As will be shown by our experiments, it can serve as an attractive complement to the standard bounding box regression loss of existing object detectors, whether using distillation or not.

3.3.3 Overall Training Loss

To train the student detector given the image classification teacher, we then seek to minimize the overall loss

$$\mathscr{L} = \mathscr{L}_{det} + \lambda_{kc}\mathscr{L}_{kd-cls} + \lambda_{kl}\mathscr{L}_{kd-loc} \,, \tag{3.12}$$

where $\mathscr{L}_{det}$ encompasses the standard classification and localization losses used to train the student detector of interest. λ_{kc} and λ_{kl} are hyper-parameters setting the influence of each loss.

3.4 Experiments

In this section, we first introduce how to train the classification teacher used in our method, and then conduct a full study of our classification and localization distillation methods on several compact detectors. Furthermore, we compare our classifier-to-detector approach to the state-of-the-art detector-to-detector ones. Finally, we perform an extensive ablation study of our method and analyze how it improves the class recognition and localization in object detection. All models are trained and evaluated on MS COCO2017 (Lin et al., 2014). Our implementation is based on MMDetection (Chen et al., 2019) with Pytorch (Paszke et al., 2019). Together with MMDetection, we also use the MMCV library, which is a dependent library for MMDetection. Otherwise specified, we take the ResNet50 as the classification teacher. We will use the same teacher for all two-stage Faster RCNNs and one-stage RetinaNets in our classifier-to-detector distillation method. We consider this to be an advantage of our method, since it lets us use the same teacher for multiple detectors.

Codebase and Dataset

Here, we provide the details and licenses of the existing assets we used in our experiments, such as the MS COCO2017 (Lin et al., 2014) dataset and the MMDetection (Chen et al., 2019) codebase. Both of them are open source and available for non-commercial academic research.

MS COCO2017 (Lin et al., 2014)[2] is a large-scale object detection, segmentation, key-point detection and captioning dataset. We use its detection benchmark, which consists of 118k training images and 5k validation ones, depicting 80 foreground object classes. The annotations for object detection are bounding boxes and object labels. In this work, we respect the terms of use listed on the website. The annotations in this dataset, along with their website, belong to the COCO Consortium and are licensed under a Creative Commons Attribution 4.0 License.

MMDetection (Chen et al., 2019)[3] is an open source object detection toolbox based on Pytorch (Paszke et al., 2019), which is released under the Apache 2.0 license. Together with MMDetection, we also use the MMCV library[4], which is a dependent library for MMDetection. MMCV is mainly released under the Apache 2.0 license, while some specific operations in this library fall under other licenses. Please refer to LICENSES.md in their website.

3.4.1 Training Classification Teachers

At the beginning, we provide the details of our experimental classification setup and of training classification teachers.

Experimental setup. To train and validate our classification teachers, we use the MS COCO2017 (Lin et al., 2014) detection dataset and crop all the objects according to their ground-truth bounding boxes. The resulting classification dataset consists of 849,902 objects for training and 36,334 objects for validation. We then train the teacher models in an image-classification manner, using the same data augmentation strategy and loss function as the student detector. Specifically, Faster RCNNs and RetinaNets share the same data augmentation methods, denoted as "general", but use the categorical cross-entropy loss (CEL) and focal loss (FL) for their classification heads, respectively; SSDs have their own data augmentation strategy and use the categorical cross-entropy loss (CEL). To train this classification teacher, we use the losses from Faster RCNN and RetinaNet frameworks jointly. Since SSDs use different data augmentation, we train another ResNet50 classification teacher for them.

In our experiments, we take ResNet50 as the teacher model. In Section 3.5, we conduct an ablation study with different teachers. Furthermore, we investigate the influence of different input sizes to our classification teachers because the objects in object detection have different resolutions than they typically have in image classification. Therefore, we train the classification teacher with input sizes in $[56 \times 56, 112 \times 112, 224 \times 224]$. Because Faster RCNNs and RetinaNets share the same data augmentation, we train a teacher for both frameworks using the two losses jointly. All the teacher models are trained using ImageNet-pretrained weights for 90 epochs with an initial learning rate of 0.0001, divided by 10 at epoch 50.

[2] https://cocodataset.org
[3] https://github.com/open-mmlab/mmdetection
[4] https://github.com/open-mmlab/mmcv

Table 3.1: Top-1 accuracy of classification teacher ResNet50 on the COCO2017 classification validation dataset.

Data Aug. + Loss	Input resolution		
	56 × 56	112 × 112	224 × 224
SSD + CEL	76.26	80.30	80.41
general + CEL	76.92	80.81	81.42
general + FL	72.86	77.50	77.04
general + CEL + FL	**77.01**	**81.02**	**81.67**

Results. In Table 3.1, we report the top-1 accuracy of our ResNet50 classification teacher on the COCO2017 classification validation dataset. The teacher models trained with the categorical cross-entropy loss benefit from larger input sizes, as shown by the top-1 accuracy increasing by more than 4 points when the input size increases from 56 to 224. Surprisingly, with the focal loss, increasing the input size to 224 yields slightly worse results than with an input of size 112. Note that the teacher trained with the focal loss underperforms those trained with categorical cross-entropy loss by more than 3 points. Furthermore, training the classification teacher with both losses always yields better top-1 accuracy than training with a single loss. To this end, we will use the same classification teacher for all two-stage Faster RCNNs and one-stage RetinaNets in our classifier-to-detector distillation method. We consider this to be an advantage of our method, since it lets us use the same teacher for multiple detectors.

3.4.2 Classifier-to-Detector Distillation on Compact Students

We then demonstrate the effectiveness of our classifier-to-detector distillation method on compact detectors, namely, SSD300, SSD512 (Liu et al., 2016) and the two-stage Faster RCNN (Ren et al., 2015) detector with lightweight backbones, i.e., MobileNetV2 (Sandler et al., 2018) and Quartered-ResNet50 (QR50), obtained by dividing the number of channels by 4 in every layer of ResNet50, reaching a 66.33% top-1 accuracy on ImageNet (Russakovsky et al., 2015).

Experimental setting. All object detectors are trained in their default settings on Tesla V100 GPUs. The SSDs follows the basic training recipe in MMDetection (Chen et al., 2019). The lightweight Faster RCNNs are trained with a 1× training schedule for 12 epochs. Let us now specify the details for the training settings of the compact student models. All experiments in this chapter were performed on Tesla V100 GPUs.

SSD300 and SSD512. For data augmentation, we first apply photometric distortion transformations on the input image, then scale up the image by a factor chosen randomly between 1× and 4× by filling the blanks with the mean values of the dataset. We then sample a patch from the image so that the minimum IoU with the objects is in $[0.1, 0.3, 0.5, 0.7, 0.9]$, with the precise value chosen randomly. Afterwards, the sampled patch is resized to 300 × 300 or 512 × 512, normalized by subtracting the mean values of the dataset, and horizontally flipped with a

Table 3.2: **Analysis of our classifier-to-detector distillation method with compact students on the COCO2017 validation set.** R50 is ResNet50, MV2 is MobileNetV2, QR50 is quartered ResNet50.

Method	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	mAR	AR_s	AR_m	AR_l
SSD300-VGG16	25.6	43.8	26.3	6.8	27.8	42.2	37.6	12.5	41.7	58.6
+ KD_{cls}	26.3 (↑ 0.7)	45.2	27.2	7.3	28.5	43.6	38.4	12.8	42.6	59.1
+ KD^0_{loc}	27.1 (↑ 1.5)	43.2	28.4	7.5	29.4	43.3	40.0	13.4	44.4	60.6
+ KD_{loc}	27.2 (↑ 1.6)	43.3	28.5	7.5	29.5	43.5	40.2	13.2	44.7	61.5
+ KD_{cls} + KD_{loc}	**27.9** (↑ **2.3**)	**45.1**	**29.2**	**8.1**	**30.1**	**45.4**	**40.4**	**13.9**	**44.7**	**61.4**
SSD512-VGG16	29.4	49.3	31.0	11.7	34.1	44.9	42.7	17.6	48.7	60.6
+ KD_{cls}	30.3 (↑ 0.9)	51.1	31.7	12.7	34.6	45.5	43.3	19.4	49.0	60.4
+ KD^0_{loc}	30.8 (↑ 1.4)	48.8	32.9	12.8	35.8	46.2	44.7	18.8	51.1	63.4
+ KD_{loc}	31.0 (↑ 1.6)	49.1	32.8	12.6	35.8	46.2	45.0	18.9	51.6	63.2
+ KD_{cls} + KD_{loc}	**32.1** (↑ **2.7**)	**51.0**	**34.0**	**13.3**	**36.6**	**47.9**	**45.3**	**20.1**	**51.2**	**63.1**
Faster RCNN-QR50	23.3	40.7	23.9	13.1	25.0	30.7	40.2	22.7	42.8	51.8
+ KD_{cls}	25.9 (↑ 2.6)	45.5	26.2	15.3	27.9	34.0	42.8	25.5	46.0	54.9
+ KD^0_{loc}	24.2 (↑ 0.9)	41.1	25.0	13.7	25.8	32.1	41.7	23.8	44.3	54.8
+ KD_{loc}	24.3 (↑ 1.0)	41.0	25.1	13.0	25.9	32.5	41.6	22.7	44.6	54.7
+ KD_{cls} + KD_{loc}	**27.2** (↑ **3.9**)	**46.0**	**27.7**	**15.2**	**29.3**	**36.2**	**44.5**	**25.9**	**48.1**	**58.3**
Faster RCNN-MV2	31.9	52.0	34.0	18.5	34.4	41.0	47.5	29.7	50.9	60.4
+ KD_{cls}	32.6 (↑ 0.7)	53.3	34.6	18.9	34.8	42.3	48.1	29.7	51.2	61.5
+ KD^0_{loc}	32.2 (↑ 0.3)	51.9	34.2	18.3	34.4	41.8	47.9	29.0	50.8	61.5
+ KD_{loc}	32.3 (↑ 0.4)	52.0	34.7	18.1	34.8	41.6	48.0	28.7	51.3	61.6
+ KD_{cls} + KD_{loc}	**32.7** (↑ **0.8**)	**52.9**	**35.0**	**19.0**	**35.0**	**42.9**	**48.4**	**29.9**	**51.8**	**61.9**

probability of 0.5. We use SGD with an initial learning rate of 0.002 to train the SSDs for 24 epochs, where the dataset is repeated 5 times. The batch size is 64, and the learning rate decays by a factor of 0.1 at the 16th and 22nd epoch.

Faster RCNN with lightweight backbones. For data augmentation, the input image is first resized so that either the maximum of the longer side is 1333 pixels, or the maximum of the shorter side is 800 pixels. Then, the image is horizontally flipped with a probability of 0.5. Afterwards, it is normalized by subtracting the mean values and dividing by the standard deviation of the dataset. The Faster RCNN-MobileNetV2 is trained by SGD for 12 epochs with a batch size of 16, and an initial learning rate set to 0.02 and divided by 10 at the 8th and 11th epoch. Faster RCNN-QR50 is trained with a larger batch size of 32 and a larger initial learning rate of 0.04. Note that, in practice, increasing the batch size and the learning rate enables us to shorten the training time while keeping the same performance as with the default 1× training setting in MMDetetion (Chen et al., 2019).

We use a ResNet50 with input resolution 112×112 as classification teacher for all student detectors. We report the mean average precision (mAP) and mean average recall (mAR) for intersection over unions (IoUs) in [0.5:0.95], the APs at IoU=0.5 and 0.75, and the APs and ARs for small, medium and large objects.

Results. The results are shown in Table 3.2. Our classification distillation yields improvements of at least 0.7 mAP for all student detectors. It reaches a 2.6 mAP improvement for Faster RCNN-QR50, which indicates that the classification in this model is much weaker. The classification distillation improves AP_{50} more than AP_{75}, while the localization distillation improves AP_{75} more than AP_{50}. As increasing AP_{75} requires more precise localization, these results indicate that each of our distillation losses plays its expected role. Note that the SSDs benefit more from the localization method than the Faster RCNNs. We conjecture this to be due to the denser, more accurate proposals of the Faster RCNNs compared to the generic anchors of the SSDs. Note also that a Faster RCNNs with a smaller backbone benefits more from our distillation than a larger one.

3.4.3 Comparison with Detector-to-detector Distillation

We then compare our classifier-to-detector distillation approach with the state-of-the-art detector-to-detector ones, such as KD (Chen et al., 2017b), FGFI (Wang et al., 2019), GID (Dai et al., 2021) and FKD (Zhang and Ma, 2021). Here, in addition to the compact students used in Section 3.4.2, we also report results on the larger students that are commonly used in the literature, i.e., Faster RCNN and RetinaNet with deeper ResNet50 (R50) backbones.

Experimental setting. Following (Zhang and Ma, 2021), the Faster RCNN-R50 and RetinaNet-R50 are trained with a 2× schedule for 24 epochs. To illustrate the generality of our approach, we also report the results of our distillation strategy used in conjunction with FKD (Zhang and Ma, 2021), one of the current best detector-to-detector distillation methods. Note that, while preparing this work, we also noticed the concurrent work of (Guo et al., 2021a), whose DeFeat method also follows a detector-to-detector distillation approach, and thus could also be complemented with out strategy.

Results. We report the results in Table 3.3. For compact student detectors, such as Faster RCNN-QR50 and SSD512, our classifier-to-detector distillation surpasses the best detector-to-detector one by 1.1 and 0.9 mAP points, respectively. For student detectors with deeper backbones, our method improves the baseline by 0.8, 0.4 and 0.5 points. Furthermore, using it in conjunction with the FKD detector-to-detector distillation method boosts the performance to the state-of-the-art of 28.0, 32.6, 34.2, 41.9 and 40.7 mAP. Overall, these results evidence that our approach is orthogonal to the detector-to-detector distillation methods, allowing us to achieve state-of-the-art performance by itself or by combining it with a detector-to-detector distillation strategy.

3.5 Ablation Study

In this section, we investigate the influence of the hyper-parameters and of different classification teachers in our approach. To this end, we use the SSD300 student detector.

Table 3.3: Comparison to detector-to-detector distillation methods on the COCO2017 validation set.

Method	mAP	AP_s	AP_m	AP_l
Faster RCNN-QR50	23.3	13.1	25.0	30.7
+ FKD (Zhang and Ma, 2021)	26.1	14.6	27.3	35.0
+ Ours	27.2	15.2	29.3	36.2
+ Ours + FKD	**28.0**	**15.4**	**29.8**	**38.5**
SSD512-VGG16	29.4	11.7	34.1	44.9
+ FKD (Zhang and Ma, 2021)	31.2	12.6	37.4	46.2
+ Ours	32.1	13.3	36.6	47.9
+ Ours + FKD	**32.6**	**13.5**	**37.6**	**48.3**
Faster RCNN-MV2	31.9	18.5	34.4	41.0
+ FKD (Zhang and Ma, 2021)	33.9	18.3	36.3	45.4
+ Ours	32.7	**19.0**	35.0	42.9
+ Ours + FKD	**34.2**	18.5	**36.3**	**45.9**
Faster RCNN-R50	38.4	21.5	42.1	50.3
+ KD (Chen et al., 2017b)	38.7	22.0	41.9	51.0
+ FGFI (Wang et al., 2019)	39.1	22.2	42.9	51.1
+ GID (Dai et al., 2021)	40.2	22.7	44.0	53.2
+ FKD (Zhang and Ma, 2021)	41.5	23.5	45.0	55.3
+ Ours	38.8	22.5	42.5	50.8
+ Ours + FKD	**41.9**	**23.8**	**45.2**	**56.0**
RetinaNet-R50	37.4	20.0	40.7	49.7
+ FGFI (Wang et al., 2019)	38.6	21.4	42.5	51.5
+ GID (Dai et al., 2021)	39.1	22.8	43.1	52.3
+ FKD (Zhang and Ma, 2021)	39.6	22.7	43.3	52.5
+ Ours	37.9	20.5	41.3	50.5
+ Ours +FKD	**40.7**	**23.1**	**44.7**	**53.8**

3.5.1 Ablation Study of KD_{cls}

We first study the effect of the loss weight λ_{kc} and the temperature T for classification distillation. As shown in Table 3.4a, these two hyper-parameters have a mild impact on the results, and we obtain the best results with $\lambda_{kc} = 0.4$ and $T = 2$, which were used for all other experiments with SSDs.

We then investigate the impact of different classification teacher networks. To this end, we trained three teacher networks ranging from shallow to deep: ResNet18, ResNet50 and ResNext101-32×8d. We further study the impact of the input size to these teachers on classification distillation, using the three sizes [56 × 56, 112 × 112, 224 × 224]. As shown in Table 3.4b,

Table 3.4: **Ablation study of KD$_{cls}$.** We evaluate the impact of the hyper-parameters and of various classification teachers on our classification distillation.

(a) Varying λ_{kc} and T.

λ_{kc}	T	mAP	AP$_{50}$	AP$_{75}$
baseline	/	25.6	43.8	26.3
0.1	1	25.8	44.2	26.6
0.1	2	25.4	44.4	25.7
0.2	1	25.8	44.2	26.6
0.3	1	26.0	44.6	26.7
0.4	1	26.1	44.8	26.6
0.4	2	**26.3**	**45.2**	**27.2**
0.4	3	26.0	45.2	26.7

(b) Varying the teacher network.

Teacher	Top-1	mAP	AP$_{50}$	AP$_{75}$
ResNet18	75.78	25.9	44.4	26.4
ResNet50	80.30	**26.3**	**45.2**	**27.2**
ResNeXt101	83.35	25.3	43.3	25.8
Input size	**Top-1**	**mAP**	**AP$_{50}$**	**AP$_{75}$**
56×56	76.26	26.2	44.8	26.9
112×112	80.30	**26.3**	**45.2**	**27.2**
224×224	80.41	26.2	44.9	26.9

even the shallow ResNet18 classification teacher can improve the performance of the student detector by 0.3 points, and the improvement increases by another 0.4 points with the deeper ResNet50 teacher. However, the performance drops with the ResNeXt101 teacher, which is the teacher with the highest top-1 accuracy. This indicates that a deeper teacher is not always helpful, as it might be overconfident to bring much additional information compared to the ground-truth labels. As for the input size, we observe only small variations across the different sizes, and thus use a size of 112 in all other experiments.

3.5.2 Ablation Study of KD$_{loc}$

We then evaluate the influence of the two main hyper-parameters of localization distillation, i.e., the grid sampling size of the spatial transformer and the adaptive pooling size of the feature maps. To this end, we vary the sampling size in $[14, 28, 56, 112, 224]$ and the pooling size in $[2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16]$.

Table 3.5: **Ablation study of KD$_{loc}$**. We investigate the effect of the sampling size, the pooling size and the choice of distilled layers on our localization distillation.

(a) Varying the sampling size.

Sampling size	mAP	AP$_{50}$	AP$_{75}$
14 × 14	26.4	43.0	27.0
28 × 28	26.7	43.2	27.8
56 × 56	26.8	43.3	28.0
112 × 112	**27.0**	43.5	28.1
224 × 224	**27.0**	43.4	28.2

(b) Varying the pooling size.

Pooling size	mAP	AP$_{50}$	AP$_{75}$
2 × 2	26.6	43.5	27.5
4 × 4	27.0	43.5	28.1
8 × 8	**27.1**	43.2	28.4
16 × 16	26.9	42.8	28.1

(c) Varying distilled layers.

ℓ_0	ℓ_1	ℓ_2	mAP
✓			27.1
	✓		26.8
✓	✓		**27.2**
✓	✓	✓	26.9

As shown in Table 3.5a, our localization distillation method benefits from a larger sampling size, although the improvement saturates after a size of 112. This lets us use the same classification teacher, with input size 112, for both classification and localization distillation. The adaptive pooling size has a milder effect on the performance, as shown in Table 3.5b, with a size of 8 yielding the best mAP. In our experiments, we adopt either 4 or 8, according to the best performance on the validation set.

We further study the layers to be distilled in our localization distillation. To this end, we extract features from the first convolutional layer ℓ_1, and from the following bottleneck block ℓ_2 of the ResNet50 teacher. As shown in Table 3.5c, distilling the knowledge of only the object regions (ℓ_0) yields a better mAP than using the ℓ_1 features. However, combining the object regions (ℓ_0) with the feature maps from ℓ_1 improves the results. Adding more layers does not help, which we conjecture to be due to the fact that these layers extract higher-level features that are thus less localized.

Table 3.6: Results of our classifier-to-detector distillation method with Faster RCNN-QR50 on the COCO2017 validation set for 1×, 2× and 4× training schedulers.

Scheduler	1×	2×	4×
Faster RCNN-QR50	23.3	23.6	24.5
+ Ours	**27.2** (↑ **3.9**)	**27.7** (↑ **4.1**)	**28.4** (↑ **3.9**)

Table 3.7: APs for IoUs ranging from 0.5 to 0.95 on the COCO2017 validation set.

Method	mAP	AP_{50}	AP_{55}	AP_{60}	AP_{65}	AP_{70}	AP_{75}	AP_{80}	AP_{85}	AP_{90}	AP_{95}
SSD300	25.6	43.8	41.3	38.4	35.1	31.2	26.3	20.3	13.0	5.2	0.5
+ KD_{cls}	26.3	45.2	42.6	39.9	36.1	31.6	27.2	21.0	13.5	5.1	0.5
+ KD_{loc}	27.2	43.3	41.3	38.8	36.0	32.9	28.5	23.0	16.5	8.4	1.3
+ KD_{cls} + KD_{loc}	27.9	45.1	42.8	40.2	37.0	34.0	29.2	23.9	17.0	8.8	1.2

3.5.3 Training Longer with 1×, 2× and 4× Schedulers

To study the effects of longer training on our approach, we trained Faster RCNN-QR50 with 1×, 2× and 4× schedulers, and reported the mAP in the Table 3.6. Our distillation method yields consistent and significant improvements with all training schedulers. This indicates that our method can make the student model converge to a better solution, not just train faster.

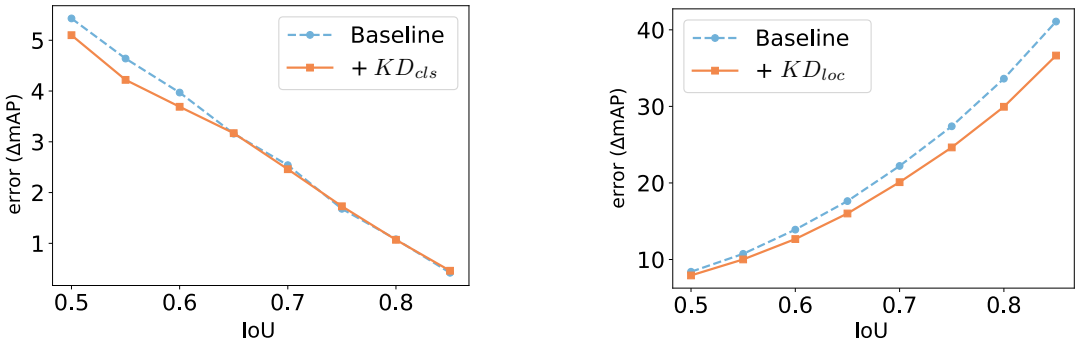
3.6 Analysis

To further understand how our classifier-to-detector distillation method affects the quality of the classification and localization, in Table 3.7, we report the APs obtained with IoUs in $[0.5, 0.95]$ with a step of 0.05. These results highlight that our classification and localization distillation strategies behave differently for different IoU thresholds. Specifically, KD_{cls} yields larger improvements for smaller IoUs, whereas KD_{loc} is more effective with IoUs larger than 0.75. This indicates that KD_{loc} indeed focuses on precise localization, while KD_{cls} distills category information. The complementarity of both terms is further evidenced by the fact that all APs increase when using both of them jointly.

3.6.1 Detection Error Analysis

We analyze the different types of detection errors using the tool proposed by Bolya et al. (2020) for the baseline SSD300 and the distilled models with our KD_{cls} and KD_{loc}. We focus on the classification and localization errors first, which are the main errors in object detection. As shown in Figure 3.2a, KD_{cls} decreases the classification error especially for IoUs smaller than 0.65. By contrast, as shown in Figure 3.2b, the effect of KD_{loc} increases with the IoU. This again shows the complementary nature of these terms.

We then discuss the details of all 6 types of detection errors discussed by Bolya et al. (2020),



(a) Classification error.

(b) Localization error.

Figure 3.2: Classification and localization error analysis in detection.

namely, classification (cls) error, localization (loc) error, both cls and loc error, duplicate detection error, background error, missed ground-truth error (missedGTerror).

In essence, as shown by Figure 3.3, localization error increases significantly as the foreground IoU increases, while all other errors decrease. The classification-related errors typically drop by using our classification distillation strategy. See, for example, the classification error for IoUs smaller than 0.65, and the error of both cls and loc for all IoUs. By contrast, our localization distillation decreases the localization-related errors, including localization error and duplicate detection errors. Specifically, with localization distillation, the localization error drops by more than 2 mAP points for IoUs larger than 0.7, albeit with a marginal increase in missedGTerror and background error. Overall, while there is a tradeoff between our classification and localization distillation strategies, they play complementary roles in improving the performance of the student detector.

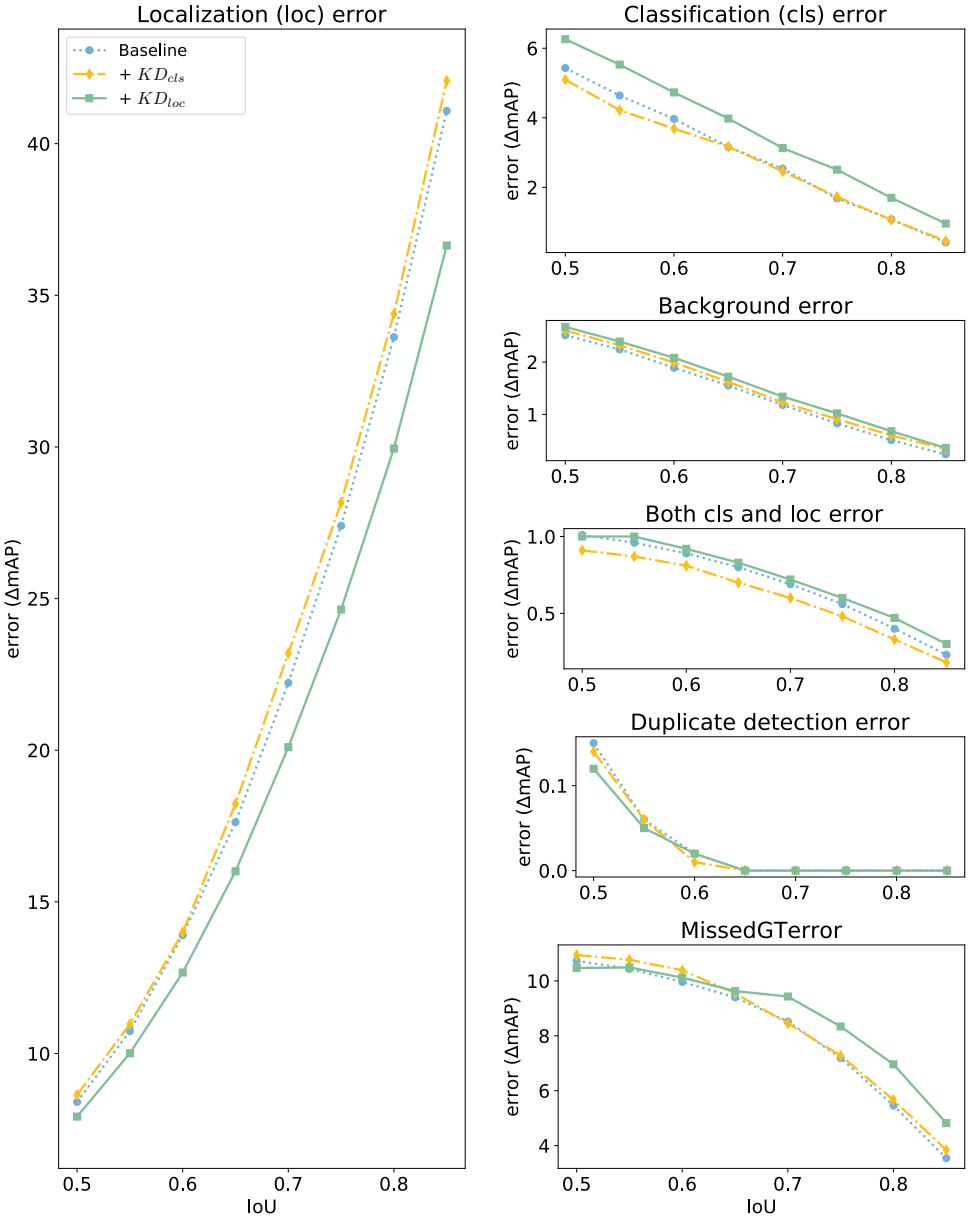


Figure 3.3: **Detection errors (better viewed in color).** We show 6 types of detection errors for the baseline SSD300, and with our classification and localization distillation methods. Note that we scaled the plots according to the magnitude of the errors they represent; the localization error, classification error and missedGTerror are the main sources of errors.

3.6.2 Qualitative Analysis

Figure 3.4 compares the detection results of the baseline model and of our distilled model on a few images. We observe that (i) the bounding box predictions of the distilled model are more precise than those of the baseline; (ii) the distilled model generates higher confidences for the

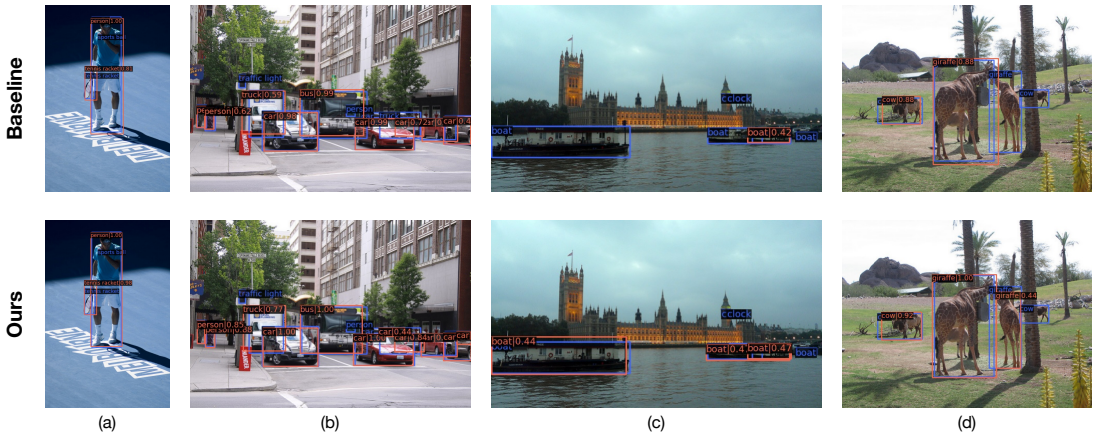


Figure 3.4: **Qualitative analysis (better viewed in color).** The ground-truth bounding boxes are in blue with their labels, and the predictions are in red with predicted labels and confidence.

correct predictions and is thus able to detect objects that were missed by the baseline, such as the boat in Figure 3.4c and the giraffe in Figure 3.4d.

3.7 Conclusion

In this chapter, we have introduced a novel approach to knowledge distillation for object detection, replacing the standard detector-to-detector strategy with a classifier-to-detector one. To this end, we have developed a classification distillation loss function and a localization distillation one, allowing us to exploit the classification teacher in two complementary manners. Our approach outperforms the state-of-the-art detector-to-detector ones on compact student detectors. While the improvement decreases for larger student networks, our approach can nonetheless boost the performance of detector-to-detector distillation. We have further shown that the same classification teacher could be used for all student detectors if they employ the same data augmentation strategy, thus reducing the burden of training a separate teacher for every student detector. Ultimately, we believe that the work in this chapter opens the door to a new approach to distillation beyond object detection: Knowledge should be transferred not only across architectures, but also across tasks.

4 Keypoint Distribution Alignment for 6D Pose Estimation

In the previous chapters, we have seen that knowledge distillation has achieved great success in many visual tasks. However, it remains completely unstudied for image-based 6D object pose estimation. In this chapter, we explore and introduce the first knowledge distillation method for 6D pose estimation. Specifically, we follow a standard approach to 6D pose estimation, consisting of predicting the 2D image locations of object keypoints. In this context, we observe the compact student network to struggle predicting precise 2D keypoint locations. Therefore, to address this, instead of training the student with keypoint-to-keypoint supervision, we introduce a strategy based the optimal transport theory that distills the teacher's keypoint *distribution* into the student network, facilitating its training. Our experiments on several benchmarks show that our distillation method yields state-of-the-art results with different compact student models.

The contents of this chapter are mainly from the following paper. I am the primary contributor.

- Guo, S., Hu, Y., Alvarez, J. M., and Salzmann, M. (2022). *Knowledge Distillation for 6D Pose Estimation by Keypoint Distribution Alignment.* arXiv Preprint.

4.1 Introduction

Estimating the 3D position and 3D orientation, a.k.a. 6D pose, of an object relative to the camera from a single image has a longstanding history in computer vision, with many real-world applications, such as robotics, autonomous navigation, and virtual and augmented reality. While modern methods (Xiang et al., 2017; Peng et al., 2019; Kendall et al., 2015; Hu et al., 2019, 2021; Li et al., 2019; Wang et al., 2021; Di et al., 2021) now all rely on deep networks to address this task, the most effective ones draw their inspiration from the traditional approach, which consists of establishing correspondences between the object's 3D model and the input image and compute the 6D pose from these correspondences using a Perspective-n-Point (PnP) algorithm. In particular, the state-of-the-art methods, GDR-Net (Wang et al., 2021) and SO-Pose (Di et al., 2021), achieve this by combining an object detection module, a large encoder-decoder to predict multiple intermediate dense representations, including 2D-3D

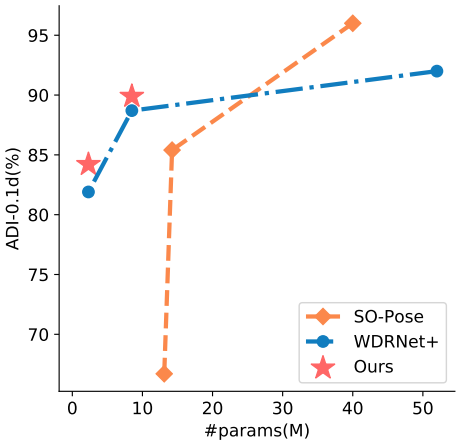


Figure 4.1: **Performance vs number of parameters.** We show the SOTA dense prediction-based SO-Pose (orange diamond) and the keypoint-based WDRNet+[1] (blue circles). Although SO-Pose outperforms WDRNet+ with a large backbone, its performance drops more rapidly than that of WDRNet+ as the number of parameters decreases. Our KD method (stars) further boosts the performance of WDRNet+ with lightweight backbones.

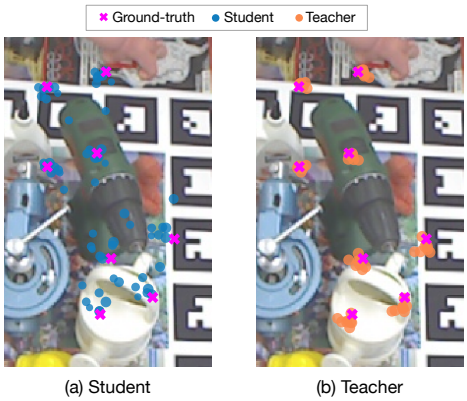


Figure 4.2: **Student vs teacher keypoint predictions.** The large backbone of the teacher allows it to produce accurate keypoints, indicated by tight clusters. By contrast, because of its more compact backbone, the student struggles to predict accurate keypoints when trained with keypoint-to-keypoint supervision. We therefore propose to align the student's and teacher's keypoint *distributions*.

correspondences, and a learnable PnP module to output the final pose. Unfortunately, the intermediate dense feature maps they extract incur many parameters, thus yielding huge models that are impractical for deployment on embedded platforms and edge devices. In principle, one could reduce the models' size by employing smaller backbones. However, as shown in Figure 4.1 for SO-Pose, the resulting accuracy drops quickly.

By contrast, we have observed keypoint-based methods (Hu et al., 2019; Peng et al., 2019; Hu et al., 2020, 2021) to yield better robustness to the use of lightweight backbones. This is illustrated in Figure 4.1 for WDRNet (Hu et al., 2021) whose performance undergoes a much smaller drop than that of SO-Pose as the number of parameter decreases. We believe this to be due to the fact that these methods aggregate multiple votes, originating from the individual feature-map locations containing the object, for the 2D image positions of a sparse set of 3D keypoints, as shown in Figure 4.2 where the keypoints are the 8 corners of the 3D object bounding box. Nevertheless, the resulting compact architectures still incur a significant drop in pose accuracy. In this chapter, we address this by introducing a knowledge distillation strategy for such keypoint-based 6D pose estimation networks.

[1]WDRNet+ is a modified version of WDRNet (Hu et al., 2021) that incorporates a detection step to be consistent with SO-Pose (Di et al., 2021).

Knowledge distillation aims to transfer some information from a deep teacher network to a compact student one. The research on this topic has tackled diverse tasks, such as image classification (Hinton et al., 2015; Zagoruyko and Komodakis, 2017; Romero et al., 2014), object detection (Zhang and Ma, 2021; Guo et al., 2021b,a) and semantic segmentation (Liu et al., 2019; He et al., 2019a). While some techniques, such as feature distillation (Romero et al., 2014; Zhang and Ma, 2021; Zagoruyko and Komodakis, 2017; Heo et al., 2019a), can in principle generalize to other tasks, no prior work has studied knowledge distillation in the context of 6D pose estimation.

In this chapter, we introduce a knowledge distillation method for 6D pose estimation motivated by the following observation. As shown in Figure 4.2, predicting accurate keypoint locations with keypoint-to-keypoint supervision is much harder for a compact student network than for a deep teacher one. We therefore argue that knowledge distillation for 6D pose estimation should not be performed by matching the individual student and teacher keypoints but instead by encouraging the student's keypoint *distribution* to become similar to the teacher one, which leaves more flexibility to the student and thus facilitates its training. To achieve this, we follow an Optimal Transport (OT) formalism (Villani, 2009), which lets us measure the distance between the two keypoint sets. We express this as a loss function that can be minimized using a weight-based variant of Sinkhorn's algorithm (Cuturi, 2013), which further allows us to exploit predicted object segmentation scores in the distillation process. Our strategy is invariant to the order and the number of predicted keypoints within a cluster, making it applicable to unbalanced teacher and student predictions that are not in one-to-one correspondence.

We validate the effectiveness of our approach by conducting extensive experiments on the popular LINEMOD (Hinterstoisser et al., 2012) and Occluded-LINEMOD (Brachmann et al., 2014) datasets. Our keypoint distribution alignment strategy consistently outperforms both a keypoint-to-keypoint distillation baseline and the state-of-the-art feature distillation method (Zhang and Ma, 2021) using diverse lightweight backbones and architecture variations. Interestingly, our approach is orthogonal to feature distillation, and we show that combining it with the state-of-the-art approach of (Zhang and Ma, 2021) further boosts the performance of student network.

Our main contributions can be summarized as follows. (i) We investigate for the first time knowledge distillation in the context of 6D pose estimation. (ii) We introduce an approach that aligns the teacher and student keypoint distributions together with their predicted object segmentation scores. (iii) Our approach can be used in conjunction with feature distillation to further boost the student's performance. We will make our code publicly available.

4.2 Related Work

6D pose estimation. With the great development and success of deep learning in computer vision (Krizhevsky et al., 2012b; He et al., 2016; Liu et al., 2016; He et al., 2017a; Romera et al.,

2018; Long et al., 2015), many works have explored its use for 6D pose estimation. The first attempts (Xiang et al., 2017; Kendall et al., 2015; Kehl et al., 2017) aimed to directly regress the 6D pose from the input RGB image. However, the representation gap between the 2D image and 3D rotation and translation made this task difficult, resulting in limited success. Therefore, most methods currently predict quantities that are closer to the input image space. In particular, the state-of-the-art approaches (Li et al., 2019; Wang et al., 2021; Di et al., 2021) output dense correspondences between the input image and the object 3D model, typically by predicting a 3D coordinate at every input location containing an object of interest. A consequence of the resulting accuracy boost, however, is a significant increase in the number of learnable parameters, arising from the use of larger encoder-decoder architectures to obtain dense intermediate representations in the two best-performing frameworks, namely GDRNet (Wang et al., 2021) and SO-Pose (Di et al., 2021). Although we have tried to reduce the size of these models, as shown in Figure 4.1 for SO-Pose, we have observed a rapid performance drop, even when retaining a relatively large number of parameters. By contrast, we have found that networks predicting a sparser set of 2D-to-3D correspondences, such as WDRNet+ (Hu et al., 2021) in Figure 4.1, offered a better robustness to the use of lightweight backbones. In essence, these methods jointly segment the object by classifying learned local features and predict either the image locations (Hu et al., 2019, 2020, 2021) or the 2D displacements from the cells' center (Peng et al., 2019) or 3D object keypoints, typically taken as the corners of the object bounding box. Nevertheless, the original backbones used by these methods remain cumbersome, and reducing them yields a performance drop. Here, we address this by introducing a knowledge distillation for keypoint-based 6D pose estimation.

Knowledge distillation. Knowledge distillation has been proven effective to transfer information from a deep teacher to a shallow student in several tasks. This trend was initiated in the context of image classification, where Hinton et al. (2015) guide the student's output using the teacher's class probability distributions, and Romero et al. (2014); Zagoruyko and Komodakis (2017); Tian et al. (2020) encourage the student's intermediate feature representations to mimic the teacher's ones. Recently, many works have investigated knowledge distillation for other tasks, evidencing the benefits of extracting task-driven knowledge. For example, in object detection, Zhang and Ma (2021) adapt the feature distillation strategy of (Romero et al., 2014) to object detectors; Wang et al. (2019) restrict the teacher-student feature imitation to regions around the positive anchors; Guo et al. (2021a) decouple the intermediate features and the classification predictions of the positive and negative regions; Guo et al. (2021b) distill detection-related knowledge from a classification teacher to a detection student. In semantic segmentation, Liu et al. (2019) construct pairwise and holistic segmentation-structured knowledge to transfer. All of these works evidence that task-driven knowledge distillation boosts the performance of compact student models. Note that the feature distillation strategy of (Zhang and Ma, 2021), although developed for object detection, is general and can be applied to 6D pose estimation. However, such a general distillation method does not leverage the specific properties of 6D pose estimation. Here, we introduce an approach that does so, and show that it outperforms the general FKD method (Zhang and Ma, 2021) and can be combined with it to

obtain a further performance boost.

Optimal transport (OT) has received a growing attention both from a theoretical perspective (Villani, 2009; Cuturi, 2013; Santambrogio, 2015) and for specific tasks, including shape matching (Su et al., 2015), generative modeling (Arjovsky et al., 2017), domain adaptation (Courty et al., 2017), and model fusion (Singh and Jaggi, 2020). In particular, OT has the advantage of providing a theoretically sound way of comparing multivariate probability distributions without approximating them with parametric models. Furthermore, it can capture more useful information about the nature of the problem by considering the geometric properties of the underlying space. Our work constitutes the first attempt at using OT to align student and teacher keypoint distributions for knowledge distillation in 6D pose estimation.

4.3 Methodology

Let us now introduce our method to knowledge distillation for 6D pose estimation. As discussed above, we follow a keypoint-based approach (Hu et al., 2019; Peng et al., 2019; Hu et al., 2020, 2021) to 6D pose estimation. Given a single RGB image captured by a calibrated camera, this approach aims to predict the 2D locations of 3D object keypoints in the image plane. This is illustrated in Figure 4.3 for keypoints taken as the 8 corners of the 3D object bounding box. More precisely, the deep network classifies each local cell, i.e., anchor, in the feature map to obtain a rough object segmentation, and predict the 2D keypoint locations for each active anchor, i.e., each anchor classified as belonging to the object. As such, each bounding box corner corresponds to a cluster of 2D locations in Figure 4.2. These locations then form 2D-to-3D correspondences that can act as input to a RANSAC-based PnP solver, e.g., (Lepetit et al., 2009) or to a shallow PnP network (Hu et al., 2020) to estimate the 6D pose.

In essence, the key to the success of such a correspondence-based approach is the prediction of accurate 2D keypoint locations. However, as shown in Figure 4.2, the predictions of a shallow student network differ significantly from those of a deep teacher one; they are less concentrated around the true keypoint locations, and thus yield less accurate 6D poses. Below, we first present a naive distillation strategy to addressing this, and then introduce our approach.

4.3.1 Naive Keypoint-to-keypoint Distillation

The most straightforward way of performing knowledge distillation is to encourage the student's predictions to match those of the teacher. In our context, one could therefore think of minimizing the distance between the locations predicted by the teacher and those predicted by the student. Formally, with N anchors in the feature map, this can be expressed as

$$\mathscr{L}_{naive-kd}(P^s, P^t) = \sum_{i=1}^{N} \lambda_i \sum_{k=1}^{8} \|P^s_{ik} - P^t_{ik}\|_p \,, \tag{4.1}$$

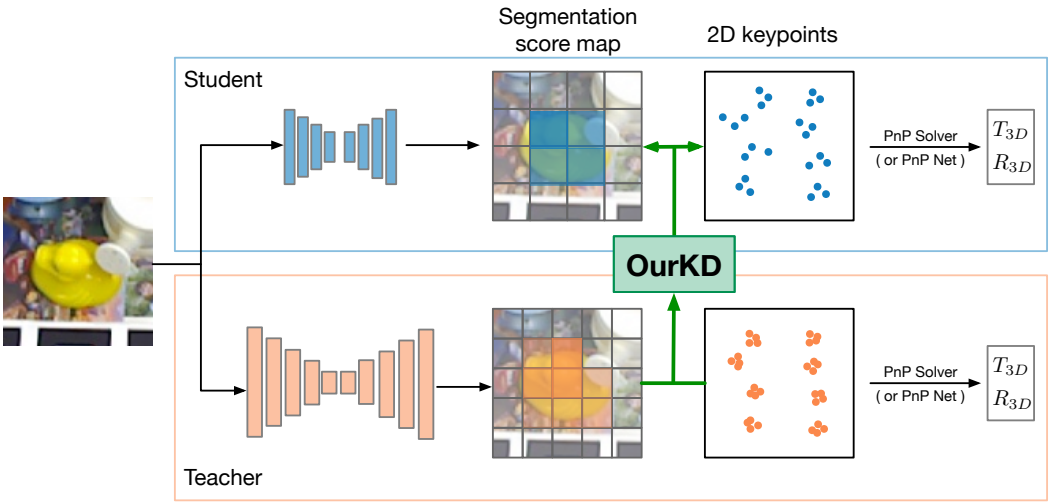


Figure 4.3: **Overview of our method** (better viewed in color). The teacher and student follow a segmentation-driven approach to 6D pose estimation, which, given an RGB input image, outputs both a segmentation score map by classifying the individual cells in the feature map and 2D keypoint locations, with each cell voting for the locations of the 8 corners of the 3D object bounding box. The keypoint locations with the bounding box corners form correspondences, which are passed to a RANSAC-based PnP solver (Lepetit et al., 2009) or a simple PnP network (Hu et al., 2020) to obtain the final 3D translation and 3D rotation. Instead of performing keypoint-to-keypoint distillation, we propose an optimal transport-based strategy that lets us jointly distill the teacher's keypoint distribution and segmentation score map into the student.

where P^s_{ik}, resp. P^t_{ik}, represent the student's, resp. teacher's, 2D prediction for keypoint k in anchor i, $\lambda_i \in \{0,1\}$ indicates whether the anchor is active or not, and $p \in \{1,2\}$.

One drawback of this strategy comes from the fact that the teacher and student network may disagree on which anchors are active and which are not, as illustrated in Figure 4.3. In other words, the λ_is may be taken from either the student, the teacher, or the intersection of their segmentation results. However, in any case, the distillation may be suboptimal, as some student's predictions would be either unsupervised by the teacher or supervised by potentially unconfident teacher predictions. Furthermore, and as argued above, a compact student tends to struggle when trained with keypoint-to-keypoint supervision, and such a naive KD formulation still follows this approach. Therefore, and as will be shown in our experiments, this naive strategy does not outperform the direct student training. Below, we therefore introduce a better-suited approach to keypoint-based knowledge distillation.

4.3.2 Keypoint Distribution Alignment

As discussed above and illustrated in Figure 4.3, the number of active student anchors N^s may differ from that of active teacher anchors N^t, making a direct match between the individual

teacher and student predictions ill-suited. To address this, and account for the observation that keypoint-to-keypoint supervision is ill-suited to train the student, we propose to align the *distributions* of the teacher and student keypoint predictions. Specifically, we achieve this using optimal transport, which can handle the case where $N^s \neq N^t$.

Formally, to allow the number of student and teacher anchors to differ, we leverage Kantorovich's relaxation (Kantorovitch, 1958) of the transportation problem. In our context, assuming that all the keypoints have the same probability mass, i.e., $\frac{1}{N^t}$ for the teacher predictions and $\frac{1}{N^s}$ for the student ones, we derive a distillation loss based on Kantorovich's optimal transport problem as

$$\begin{aligned} \tilde{\mathscr{L}}_{kd}(P^s, P^t) &= \sum_{k=1}^{8} \min_{\pi^k} \sum_{i=1}^{N^s} \sum_{j=1}^{N^t} \pi_{ij}^k \| P_{ik}^s - P_{jk}^t \|_p \\ \text{s.t.} \quad & \forall k, i, \ \sum_{j=1}^{N^t} \pi_{ij}^k = \frac{1}{N^s} \ , \ \ \forall k, j, \ \sum_{i=1}^{N^s} \pi_{ij}^k = \frac{1}{N^t} \ . \end{aligned} \tag{4.2}$$

Note that we consider separate costs for the individual keypoints, thus preventing a 2D location corresponding to one particular corner to be assigned to a different corner. In our experiments, we found $p = 2$ to be more effective than $p = 1$ and thus the ℓ_2 norm below.

The above formulation treats all keypoint predictions equally. However, different predictions coming from different anchors might not have the same degree of confidence. In particular, this can be reflected by how confident the network is that a particular anchor contains the object of interest, or, in other words, by the segmentation score predicted by the network. Let α_i^s denote such a score for anchor i in the student network, and α_j^t a similar score for anchor j in the teacher network.[2] We then re-write our distillation loss as

$$\begin{aligned} \tilde{\mathscr{L}}_{kd}(P^s, P^t) &= \sum_{k=1}^{8} \min_{\pi^k} \sum_{i=1}^{N^s} \sum_{j=1}^{N^t} \pi_{ij}^k \| P_{ik}^s - P_{jk}^t \|_2 \\ \text{s.t.} \quad & \forall k, i, \ \sum_{j=1}^{N^t} \pi_{ij}^k = \alpha_i^s \ , \ \ \forall k, j, \ \sum_{i=1}^{N^s} \pi_{ij}^k = \alpha_j^t \ . \end{aligned} \tag{4.3}$$

In essence, because this loss involves both the 2D keypoint locations and the anchor-wise classification scores, it distills jointly the correspondences and the segmentation results from the teacher to the student.

To solve this optimal transport problem, we rely on Sinkhorn's algorithm (Cuturi, 2013), which introduces a soft versions of the contraints via Kullback-Leibler divergence regularizers. This then yields the final distillation loss

[2] Note that these scores do not depend on the keypoint index k as a single score is predicted for each anchor.

$$\begin{aligned}\mathscr{L}_{kd}(P^s, P^t) = &\sum_{k=1}^{8} \min_{\pi^k} \sum_{i=1}^{N^s} \sum_{j=1}^{N^t} \pi_{ij}^k \|P_{ik}^s - P_{jk}^t\|_2 \\ &+ \sum_{k=1}^{8} \left(\varepsilon^2 \mathrm{KL}(\pi^k, \alpha^s \otimes \alpha^t) + \rho^2 \mathrm{KL}(\pi^k \mathbf{1}, \alpha^s) + \rho^2 \mathrm{KL}\left((\pi^k)^\top \mathbf{1}, \alpha^t\right)\right),\end{aligned} \tag{4.4}$$

where α^s and α^t concatenate the classification score values for the student and the teacher, respectively. This formulation was shown to be amenable to fast parallel optimization on GPU platforms, and thus well-suited for deep learning (Cuturi, 2013; Feydy et al., 2019). In our experiments, we normalize the predicted 2D keypoints by the image size to the $[0,1]^2$ space and set ε to 0.001, and ρ to 0.5 to handle outliers.

4.3.3 Network Architecture

Our approach can be applied to any network that predicts the 2D locations of 3D object keypoints. Without loss of generality, we use WDRNet (Hu et al., 2021) as our base network, which is a typical keypoint-based framework. WDRNet employs a feature pyramid strategy to predict the 2D keypoint locations at multiple stages of its decoder network. These multi-stage predictions are then fused by an ensemble-aware sampling strategy, ultimately still resulting in 8 clusters of 2D locations, i.e., one cluster per 3D bounding box corner. To make our baseline consistent with the state-of-the-art methods (Li et al., 2019; Wang et al., 2021; Di et al., 2021), we also use a detection pre-processing step that provides an image patch as input to WDRNet. We refer to this as WDRNet+. As will be shown in our results, the pre-processing detection step allows WDRNet to outperform the state-of-the-art SO-Pose (Di et al., 2021) with lightweight backbones. Nevertheless, as also evidenced by our experiments, the success of our distillation strategy does not depend on it.

In our experiments, the teacher and student networks follow the same general architecture, only differing in their backbones. Note that different backbones may also yield different number of stages in the feature pyramid, but our distribution matching approach to knowledge distillation is robust to such differences.

To train our WDRNet+ networks, we rely on the focal loss for the segmentation branch and the 3D regression loss proposed in (Hu et al., 2021) for the keypoint prediction one. When performing distillation to a student network, we complement these loss terms with our distillation loss of Eq. 4.4. To implement it, we rely on the GeomLoss library (Feydy et al., 2019)[3].

4.4 Experiments

In this section, we first discuss our experimental settings, and then demonstrate the effectiveness and generalization ability of our approach on two widely-adopted datasets, LINEMOD and Occluded-LINEMOD. Finally, we analyze different aspects of our method and evaluate it on variations of our architecture.

[3] https://github.com/jeanfeydy/geomloss

4.4.1 Experimental Settings

Datasets. We conduct experiments on the standard LINEMOD (Hinterstoisser et al., 2012) and Occluded-LINEMOD (Brachmann et al., 2014) 6D pose estimation benchmarks. The LINEMOD dataset contains around 16000 RGB images depicting 13 objects, with a single object per image. Following Brachmann et al. (2016), we split the data into a training set containing around 200 images per object and a test set containing around 1000 images per object. The Occluded-LINEMOD dataset was introduced as a more challenging version of LINEMOD, where multiple objects heavily occlude each other in each RGB image. It contains 1214 testing images. For training, following standard practice, we use the real images from LINEMOD together with the synthetic ones provided with the dataset and generated using physically-based rendering (Hodaň et al., 2020).

Teacher & student architectures. For our teacher models, we use DarkNet53 (Redmon and Farhadi, 2018) as backbone, as in the original WDRNet (Hu et al., 2021). For the compact students, we experiment with different lightweight backbones, including DarkNet-tiny (Redmon et al., 2016) and a further reduced model, DarkNet-tiny-H, containing half of the channels of DarkNet-tiny in each layer.

Baselines. We compare our method to the direct training of the student without any distillation (Student), the naive knowledge distillation strategy introduced in Section 4.3.1 (Naive-KD), and the state-of-the-art feature distillation method (FKD) (Zhang and Ma, 2021), which, although only demonstrated for object detection, is applicable to 6D pose estimation. For these baselines, we report the results obtained with the best hyper-parameter values. Specifically, for FKD, the best distillation loss weight on both datasets was 0.01; for Naive-KD, the best weight was 0.1, and the best norm was $p = 1$ for DarkNet-tiny and $p = 2$ for DarkNet-tiny-H, respectively. For our method, the distillation loss was set to 5 for LINEMOD and to 0.1 for Occluded-LINEMOD. We provide the results of the hyper-parameter search in Section 4.6.

Evaluation metric. We report our results using the standard ADD-0.1d metric. It is computed as the percentage of images for which the average 3D point-to-point distance between the object model in the ground-truth pose and in the predicted one is less than 10% of the object diameter. For symmetric objects, the point-to-point distances are computed between the nearest points. Note that, on LINEMOD, we report the results obtained using the ground-truth 2D bounding boxes to remove the effects of the pretrained detectors. On Occluded-LINEMOD, we report the results obtained with the same detector as in (Wang et al., 2021; Di et al., 2021) to evidence the effectiveness of our knowledge distillation method.

4.4.2 Comparison with the State of the Art

Results on LINEMOD. We report the results of our method and the baselines for all classes of the LINEMOD dataset in Table 4.1 and Table 4.2 for DarkNet-tiny and DarkNet-tiny-H, respectively. While Naive-KD slightly improves direct student training with the DarkNet-tiny

Table 4.1: **Results of DarkNet-tiny backbone on LINEMOD dataset.** We report the ADD-0.1d for the baseline model, Naive-KD, FKD (Zhang and Ma, 2021) and our KD method for each class. Our method not only outperforms Naive-KD and FKD, but can also be combined with FKD to obtain a further performance boost, yielding state-of-the-art results.

Class	Teacher	Student	Naive-KD	FKD	Ours	Ours+FKD
Ape	82.6	73.4	74.1	74.8	74.7	**76.2**
Bvise	95.5	95.2	95.4	94.2	95.5	**96.7**
Cam	93.8	91.2	89.7	91.3	91.3	**92.0**
Can	95.7	92.4	92.7	**94.4**	92.2	94.0
Cat	92.0	87.2	85.0	87.5	88.4	**88.6**
Driller	94.8	92.2	93.1	94.8	93.3	**94.8**
Duck	76.0	70.9	74.4	73.6	73.5	**74.7**
Eggbox*	99.1	99.3	98.7	98.9	99.1	**99.3**
Glue*	96.4	97.2	97.1	96.2	97.7	**97.7**
Holep	86.2	78.0	82.1	79.5	**82.4**	82.2
Iron	93.6	92.1	92.1	91.4	**93.5**	93.2
Lamp	97.7	96.6	95.3	96.9	**97.0**	96.8
Phone	91.2	87.5	88.4	89.4	88.2	**89.6**
AVG.	91.9	88.7	89.1	89.4	89.9 (↑ 1.2)	**90.4 (↑ 1.7)**

Table 4.2: **Results of DarkNet-tiny-H backbone on LINEMOD dataset.** We report the ADD-0.1d for the baseline model, Naive-KD, FKD (Zhang and Ma, 2021) and our KD method for each class. As in Table 4.1, we achieve the state-of-the-art results on the reduced tiny-H backbone.

Class	Teacher	Student	Naive-KD	FKD	Ours	Ours+FKD
Ape	82.6	65.4	64.1	68.4	69.4	**69.9**
Bvise	95.5	92.0	91.4	92.8	**93.8**	93.7
Cam	93.8	78.4	79.1	83.8	84.5	**84.5**
Can	95.7	82.2	81.0	83.3	83.9	**83.9**
Cat	92.0	81.5	78.7	80.7	**81.8**	81.6
Driller	94.8	85.5	87.4	**90.5**	90.0	90.3
Duck	76.0	64.3	63.6	66.8	66.5	**68.9**
Eggbox*	99.1	95.8	95.0	96.3	96.4	**96.4**
Glue*	96.4	90.7	91.2	91.0	91.9	**93.2**
Holep	86.2	73.2	72.3	**77.5**	74.1	76.3
Iron	93.6	86.3	86.3	87.6	88.7	**90.5**
Lamp	97.7	93.6	94.2	93.4	**94.8**	94.6
Phone	91.2	76.0	75.8	**80.6**	78.2	79.2
AVG.	91.9	81.9	81.6	84.1	84.2 (↑ 2.3)	**84.8 (↑ 2.9)**

Table 4.3: **Results on OCC-LINEMOD.** We report the ADD-0.1d for each class. Our method performs on par with FKD, combining it with FKD yields a further performance boost.

Class	SO-Pose	Teacher	Student	FKD	Ours	Ours + FKD
Ape	29.0	33.4	25.5	26.7	25.7	**26.9**
Can	51.8	70.9	46.6	53.9	53.5	**54.7**
Cat	19.1	45.1	31.4	31.1	32.2	**32.9**
Driller	46.2	70.9	51.2	52.1	52.9	**52.9**
Duck	29.7	27.0	22.5	25.3	25.7	**27.0**
Eggbox*	31.3	53.7	43.4	49.0	48.2	**50.0**
Glue*	46.8	70.7	54.5	55.6	55.8	**56.9**
Holep	45.4	59.7	49.3	52.2	52.1	**54.5**
AVG.	37.3	53.9	40.5	43.2	43.2 (↑ 2.7)	**44.5 (↑ 4.0)**

backbone, it degrades the performance with DarkNet-tiny-H. This matches our analysis in Section 4.3; the fact that the student's and teacher's active anchors differ make keypoint-to-keypoint distillation ill-suited.

Both FKD and our approach boost the student's results, with a slight advantage for our approach. In particular the accuracy improvement is larger, i.e., 2.3 points, for the smaller DarkNet-tiny-H backbone, for which the gap between the student and the teacher performance is also bigger. Note that the improvement of our approach over the student is consistent across the 13 objects. Interestingly, the types of distillation performed by FKD and by our approach are orthogonal; FKD distills the features while we distill the predictions. As such, the two methods can be used together. As can be seen from the table, this further boosts the results, reaching a margin over the student of 1.7 points and 2.9 points with DarkNet-tiny and DarkNet-tiny-H, respectively, and thus constitutes the state of the art on the LINEMOD dataset for such compact architectures.

Results on Occluded-LINEMOD. We now show the generality of our approach by evaluating it on the more challenging Occluded-LINEMOD. Here, we use only FKD (Zhang and Ma, 2021) as baseline and drop Naive-KD due to its inferior performance shown before. The results are provided in Table 4.3. Our keypoint-based knowledge distillation method yields results on par with the feature-based FKD on average. Note, however that FKD requires designing additional adaptive layers to match the misaligned feature maps, while our method does not incur additional parameters. More importantly, jointly using our method with FKD achieves the state-of-the-art results with 4.0 points improvements over the baseline student model. For some classes, such as *can*, *eggbox* and *holepuncher*, the improvements surpasses 5 points.

Table 4.4: **Ablation study on LINEMOD: With vs without segmentation scores.**

Model	#Param(M)	ADD-0.1d
SO-Pose†	14.2	85.4
WDRNet+(tiny)	8.5	88.7
Ours-NoScores		89.1
Ours		**89.9**
SO-Pose†	13.1	66.7
WDRNet+(tiny-H)	2.3	81.9
Ours-NoScores		83.1
Ours		**84.2**

† SO-Pose model with the corresponding backbone.

4.5 Analysis

4.5.1 With vs Without Segmentation Scores

We compare the results of our approach without and with the use of the segmentation scores in the optimal transport formulation, i.e., Eq. 4.3 vs Eq. 4.2. The comparison in Table 4.4 shows the benefits of jointly distilling the predicted 2D locations and the segmentation scores. In the table, we also report the results of the baseline SO-Pose (Di et al., 2021) and WDRNet+. The numbers show that, despite its larger number of weights, SO-Pose yields worse results than our student baseline for a comparable backbone. Moreover, its performance drops significantly from 85.4 to 66.7 as the number of parameters decreases, while that of WDRNet+ degrades more gracefully.

4.5.2 Without Detection Pre-processing

Note that we incorporated the pre-processing detection step in WDRNet only to show that keypoint-based methods could match the performance of the state-of-the-art dense prediction ones, such as SO-Pose, which also estimate the pose based on detected image patches. In other words, the success of our knowledge distillation strategy does not depend on the detection. To demonstrate this, in the left portion of Table 4.5, we report the results of our approach applied to the original WDRNet with a DarkNet-tiny backbone. As a matter of fact, the gap between direct student training and our approach is even larger (1.2 vs 2.1), showing the benefits of our approach on weaker networks.

4.5.3 With a Simple PnP Network

In the right portion of Table 4.5, we compare the results of our approach with those of the baselines on an architecture obtained by incorporating a simple PnP network at the end of

Table 4.5: **Evaluation under different network settings on LINEMOD.** We report the ADD-0.1d with the original WDRNet framework and with an additional simple PnP network. Our method improves the performance of the student network in both settings.

Class	WDRNet (Hu et al., 2021)			WDRNet + PnPNet (Hu et al., 2020)		
	Teacher	Student	Ours	Teacher	Student	Ours
Ape	70.3	41.2	43.0	50.6	29.4	35.1
Bvis	94.2	81.5	86.1	91.7	72.9	80.8
Cam	89.0	67.6	69.8	90.5	56.1	73.3
Can	90.6	72.1	73.8	88.3	57.5	75.9
Cat	87.1	54.3	61.5	62.5	61.8	48.5
Driller	93.6	78.3	79.3	87.1	68.6	71.9
Duck	64.5	35.9	39.6	38.1	32.0	39.6
Eggbox*	95.4	79.3	83.8	99.3	91.8	96.6
Glue*	93.4	83.4	82.7	92.8	87.3	92.2
Holep	77.1	44.2	46.9	70.9	46.4	49.9
Iron	90.9	75.8	75.1	93.3	76.1	80.3
Lamp	96.3	84.8	86.8	95.8	68.7	87.2
Phone	85.3	69.6	67.3	92.3	57.0	76.6
AVG.	86.7	66.8	**68.9 (↑ 2.1)**	81.0	62.0	**69.8 (↑ 7.8)**

WDRNet, following the strategy of (Hu et al., 2020). With such an architecture, the 2D keypoint locations only represent an intermediate output of the network, with the PnP module directly predicting the final 3D translation and 3D rotation from them. As can be seen from these results, our distillation strategy still effectively boosts the performance of the student with this modified architecture, further showing the generality of our approach, which can distill keypoint-based knowledge both for PnP solvers and PnP networks.

4.5.4 Qualitative Analysis

In this section, we provide additional comparisons of the predicted 2D keypoints distributions obtained with the baseline student model and with our distilled model on several examples from Occluded-LINEMOD. As shown in Figure 4.4, the predicted 2D keypoints clusters from our distilled models are closer to the ground-truth object corners than those of the baseline model. Furthermore, our distilled model mimics the teacher's keypoints distributions. As a result, our distilled model yields more accurate pose estimates than the baseline student one.

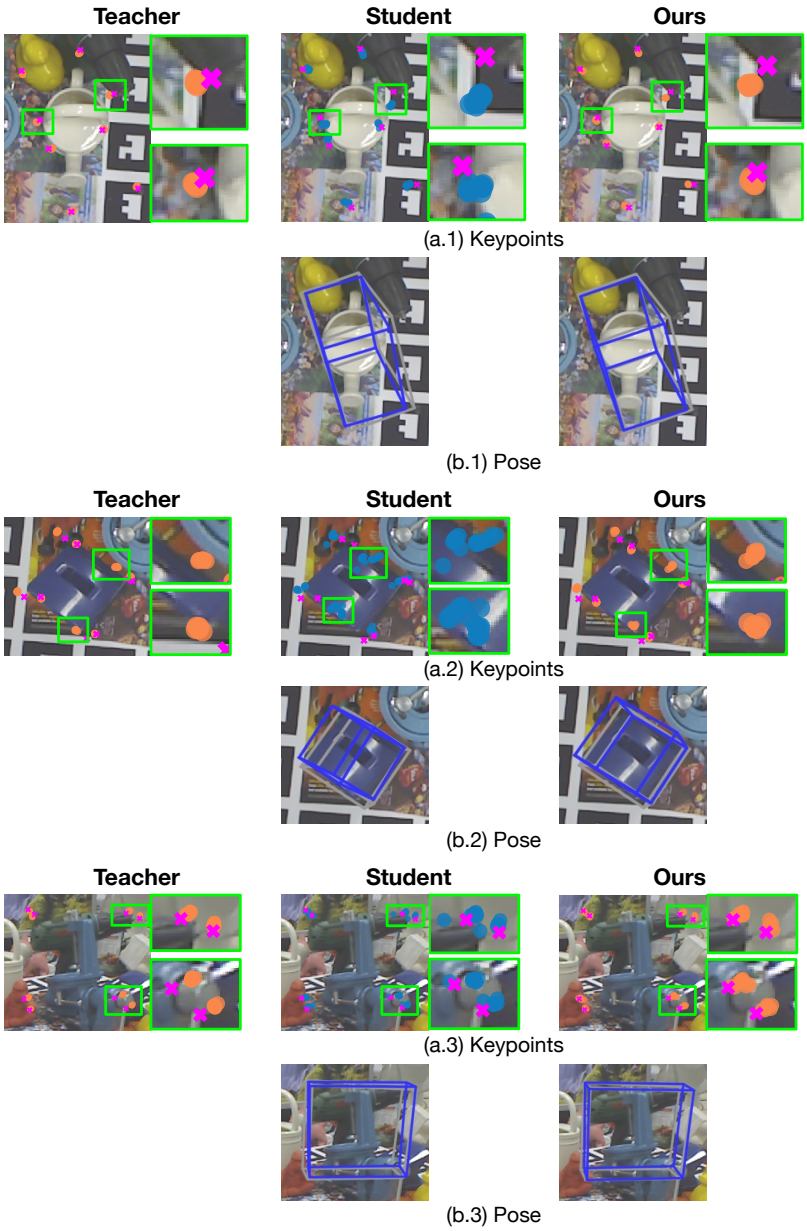


Figure 4.4: **Qualitative Analysis** (better viewed in color). In (a), we compare the 2D keypoints predictions from our distilled model (3rd column with orange dots) and the baseline student model (2nd column with blue dots). With our proposed keypoint distribution alignment distillation method, the model predicts tighter keypoint clusters, closer to the ground-truth corners (pink cross) than the baseline model. Furthermore, our distilled model is able to mimic the teacher's keypoint distributions (1st column with orange dots). Light-green boxes highlight some keypoints clusters, which are also zoomed in on the side of the image. In (b), we show the estimated poses. The blue boxes are the predicted 3D bounding boxes while the gray ones are the ground-truth bounding boxes. Our distilled model generates better pose estimates than the student model.

Table 4.6: **Results of Naive-KD with DarkNet-tiny-H backbone on Ape and Duck.** We report the ADD-0.1d for the Naive-KD with $p = 1$ and $p = 2$.

Class	Teacher	Student	$p = 1$			$p = 2$		
			0.01	0.1	1.0	0.01	0.1	1.0
Ape	82.6	65.4	63.2	64.4	65.7	63.8	64.1	64.8
Duck	76.0	64.3	59.4	63.3	60.3	59.0	63.6	62.2
AVG.	79.3	64.8	61.3	**63.9**	63.0	61.4	**63.9**	63.5

Table 4.7: **Results of Naive-KD on LINEMOD dataset.** We report the ADD-0.1d for the Naive-KD with DarkNet-tiny-H and DarkNet-tiny backbones with ps and weights searched from Table 4.6.

Class	Teacher	DarkNet-tiny-H			DarkNet-tiny		
		Student	$p = 1$ 0.1	$p = 2$ 0.1	Student	$p = 1$ 0.1	$p = 2$ 0.1
Ape	82.6	65.4	64.4	64.1	73.4	74.1	74.0
Bvise	95.5	92.0	90.6	91.4	95.2	95.4	96.6
Cam	93.8	78.4	77.8	79.1	91.2	89.7	90.0
Can	95.7	82.2	78.7	81.0	94.4	92.7	92.9
Cat	92.0	81.5	77.8	78.7	87.2	85.0	82.0
Driller	94.8	85.5	87.6	87.4	92.2	93.1	93.2
Duck	76.0	64.3	63.3	63.6	70.9	74.4	73.9
Eggbox*	99.1	95.8	95.3	95.0	99.3	98.7	99.4
Glue*	96.4	90.7	92.6	91.2	97.2	97.1	96.9
Holep	86.2	73.2	71.6	72.3	78.0	82.1	81.0
Iron	93.6	86.3	86.4	86.3	92.1	92.1	91.9
Lamp	97.7	93.6	93.3	94.2	96.6	95.3	96.5
Phone	91.2	76.0	75.7	75.8	87.5	88.4	87.4
AVG.	91.9	81.9	81.2	**81.6**	88.9	**89.1**	88.9

4.6 Ablation Study

4.6.1 Hyper-parameters for Naive-KD and FKD

In this section, as mentioned in Section 4.4, we provide the details of the hyper-parameter search for Naive-KD and FKD (Zhang and Ma, 2021). In both cases, this search was mostly focused on models with a DarkNet-tiny-H backbone and on 2 difficult LINEMOD classes, i.e., Ape and Duck.

Table 4.8: **Weight searching for FKD on LINEMOD dataset (Ape and Duck).** We report the ADD-0.1d for FKD (Zhang and Ma, 2021) with different weights.

Class	Teacher	Student	0.001	0.01	0.1	1.0
Ape	82.6	65.4	66.5	68.4	66.5	65.0
Duck	76.0	64.3	65.2	66.8	61.2	60.3
AVG.	79.3	64.8	65.9	**67.6**	63.8	62.7

Table 4.9: **Results of FKD on Occluded-LINEMOD dataset.** We report the ADD-0.1d for FKD (Zhang and Ma, 2021) with different weights. Note that due to the worse results on Ape and Duck with a weight of 0.1, we didn't extend this setting to other classes.

Class	Teacher	Student	0.001	0.01	0.1
Ape	33.4	25.5	26.8	26.7	22.6
Can	70.9	46.6	52.8	53.9	-
Cat	45.1	31.4	31.0	31.1	-
Driller	70.9	51.2	52.3	52.1	-
Duck	27.0	22.5	24.7	25.3	19.8
Eggbox*	53.7	43.4	47.9	49.0	-
Glue*	70.7	54.5	54.3	55.6	-
Holep	59.7	49.3	51.0	52.2	-
AVG.	53.9	40.5	42.6	**43.2**	-

Naive-KD. As shown in Table 4.6, the best results are obtained with a norm $p = 1$ and a distillation loss weight of 0.1, and with a norm $p = 2$ with a weight of 0.1. We therefore provide the corresponding results for all classes and for the DarkNet-tiny-H and DarkNet-tiny backbones in Table 4.7. Note that $p = 2$ with a weight of 0.1 yields the best results for DarkNet-tiny-H, and $p = 1$ with a weight of 0.1 gets the best performance for DarkNet-tiny. Therefore, we report the best results for each backbone in Section 4.4.

FKD (Zhang and Ma, 2021). We follow the same strategy as above, and report the results for Ape and Duck with FKD in Table 4.8. The best results are obtained with a distillation weight of 0.01. As the weight increases, the performance decreases significantly. We therefore adopted 0.01 as FKD weight for both the DarkNet-tiny-H and DarkNet-tiny backbones on the LINEMOD dataset. For FKD, we also conducted a hyper-parameter search on Occluded-LINEMOD. As shown in Table 4.9, a distillation weight of 0.01 also achieves the best results. Note that we did not test a weight of 0.1 on all classes because of the worse results it gave on Ape and Duck.

Table 4.10: **Results of our proposed KD with DarkNet-tiny-H backbone on LINEMOD dataset (Ape and Duck).** We report the ADD-0.1d for our proposed KD with different ps and weights.

Class	Teacher	Student	$p=1$		$p=2$		
			1.0	10.0	1.0	5.0	10.0
Ape	82.6	65.4	61.9	61.5	66.5	69.4	67.0
Duck	76.0	64.3	61.2	61.9	65.1	66.5	65.8
AVG.	79.3	64.8	61.6	61.7	65.8	**67.9**	66.4

Table 4.11: **Results of our proposed KD on Occluded-LINEMOD dataset.** We report the ADD-0.1d for our proposed KD with different weights.

Class	Teacher	Student	0.01	0.1
Ape	33.4	25.5	23.5	25.7
Can	70.9	46.6	51.2	53.5
Cat	45.1	31.4	31.3	32.2
Driller	70.9	51.2	51.5	52.9
Duck	27.0	22.5	20.0	25.7
Eggbox*	53.7	43.4	47.9	48.2
Glue*	70.7	54.5	54.3	55.8
Holep	59.7	49.3	51.0	52.1
AVG.	53.9	40.5	41.3	**43.2**

4.6.2 Hyper-parameters for our Approach

In this section, we include the hyper-parameter search for our proposed keypoint distribution alignment distillation method, including the norm ps and the weight of our distillation loss. As for the baseline, we focused this search on DarkNet-tiny-H for the Ape and Duck classes. As shown in Table 4.10, $p = 2$ yields much better results than $p = 1$, and we therefore use $p = 2$ in Section 4.4. As for the loss weight, on the LINEMOD dataset, 5 yields the best results, which we use to report the results on the 13 classes in Section 4.4. For Occluded-LINEMOD, as shown in Table 4.11, we obtain the best results with a weight of 0.1. Note that our preliminary experiments with a weight of 1 showed worse performance, and we thus did not compute full results with weights larger than 0.1.

4.7 Conclusion

We have introduced the first approach to knowledge distillation for 6D pose estimation. Our method is driven by matching the distributions of predicted 2D keypoint locations from a deep teacher network to a compact student one. We have formulated this as an optimal transport problem that lets us jointly distill the predicted 2D locations and classification scores that segment the object in the image. Our experiments have demonstrated the effectiveness of our approach and its benefits over a naive point-to-point distillation strategy. Furthermore, our formalism is complementary to feature distillation strategies and can further boost its performance. In essence, the work in this chapter confirms the importance of developing task-driven knowledge distillation methods, and we hope that it will motivate others to pursue research in this direction, may it be for 6D pose estimation or for other tasks.

5 Conclusion and Future work

In summary, this thesis has presented a variety of solutions for improving the representation learning ability and increasing the performance at inference of *arbitrary, given* compact neural networks. We have demonstrated the effectiveness of our solutions in multiple visual recognition tasks, namely, image classification, object detection, semantic segmentation and 6D pose estimation. We have started with a general approach based on over-parameterization, which generalizes to a wide range of visual recognition tasks, and then we have moved to designing a cross-architecture and cross-task knowledge distillation method for object detection. Finally, motivated by the inherent properties of 6D pose estimation, we have proposed to distill keypoint distribution knowledge for this task. Below, we first summarize the accomplishments and contributions presented in this thesis, and then discuss the remaining limitations of our approaches and identify some potential directions for future research in this field.

5.1 Summary

In Chapter 2, we have proposed an general algorithm to facilitate the training of any given compact model. Our algorithm is applicable to a broad range of visual recognition tasks, including those demonstrated in our experiments, i.e., image classification, object detection and semantic segmentation, but not limited to them. This work was motivated by the observation that over-parameterization is important for network training while it is not necessary at inference. We have therefore introduced additional parameters by linearly expanding the convolutional and fully connected layers in the compact networks to improve both the optimization behavior and generalization ability of the compact network.

In Chapter 3, we have presented our classifier-to-detector distillation approach. In contrast to detector-to-detector distillation methods, which rely on different teachers for one-stage and two-stage detectors, our approach allows us to train a single general classification teacher applicable to multiple detector types. Our classifier-to-detector strategy yields state-of-the-art results for compact detectors and is complementary to detector-to-detector distillation thus further boosting its performance on the detection benchmark. The proposed distillation is

capable of improving the representation ability of student detectors by reducing both the classification and localization errors. This chapter has illustrated the potential for cross-architecture and cross-task knowledge distillation.

In Chapter 4, we have explored knowledge distillation for image-based 6D pose estimation, for which knowledge distillation was unstudied. We have first studied both dense prediction based models and keypoint based ones with lightweight backbones and observed that the keypoint based models are more robust to compact backbones than the dense prediction ones, whose performance drops significantly as the size of the model decreases. The key to the success of keypoint based models lies in predicting tight keypoint distributions. Motivated by this, we have proposed to align the keypoint distribution from the student model to that of the teacher model. This was achieved by building upon optimal transport theory, and has allowed us to improve the keypoints prediction and the final performance of the compact 6D pose estimation models.

5.2 Limitations and Future Work

While bringing some major improvements in the field of efficient deep learning for multiple visual recognition tasks, much work remains to be done to obtain much better performance with compact neural networks at low cost. This section discusses the main limitations of our proposed methods and suggests potential directions for future work.

ExpandNets build on the theoretical research on over-parameterization, but provide practical and effective ways to facilitate the training of convolutional layers, with extensive experiments and empirical analysis of our expansion strategies and of their impact on training behavior and generalization ability. However, the expansion rate in the proposed method is manually set. Thus, one extension of this work is to automatically search for the appropriate expansion rate of the given compact model. Moreover, our results on AlexNet seem to indicate that expansion is not as effective on large networks as it is on compact ones. Another future research direction could involve studying the reason for this and propose solutions for this scenario. From an ecological standpoint, even though our expanded networks make better use of the GPU resources than the compact ones, they require more training resources, thus increasing their carbon footprint for training. Therefore, exploring more efficient expansion strategies is also an interesting future direction.

Classifier-to-Detector knowledge distillation introduces a simple yet general approach to knowledge distillation for object detection by transferring knowledge across architectures and tasks. Our approach enables distilling knowledge from a single classification teacher into different student detectors. As such, our work reduces the need for a separate deep teacher detector for each student network; therefore, we reduce training resources and memory footprint compared to the detector-to-detector knowledge distillation approaches. However, in our work, we only studied the classification and detection cross-task scenario. Investigating solutions for more tasks via the proposed cross-task and cross-architecture strategy would

strengthen our claim and open the door to a new research direction for knowledge distillation.

Keypoint distribution alignment successfully improves keypoint distribution prediction for compact 6D pose estimation models, thus achieving better performance. However, our results show that some object classes benefit more from our knowledge distillation approach than others. A future direction in this area would consist of analyzing this in more detail, with a view to designing a class-adaptive version of our method. Furthermore, our current framework is designed for keypoint-based 6D pose estimation methods. Although these methods are highly effective, extending our distribution-based approach to frameworks that perform dense prediction for 6D pose estimation would be of interest to the field.

A Appendix

Pytorch Code of Toy Example for Matrix Representation of a Convolution Operator

In the appendix, we provide the Pytorch code of the toy example for matrix representation in Chapter 2. Here, we list the toy code to expand a convolutional layer with either standard or depthwise convolutions and contract the expanded layers back. This code is based on our published code and can also be found in dummy_test.py.

Pytorch code to expand and contract back a standard convolutional layer:

```
import torch
import torch.nn as nn

m = 3  # input channels
n = 8  # output channels
k = 5  # kernel size
r = int(4)  # expansion rate
imgs = torch.randn((8, m, 7, 7))  # input images with batch size as 8

# original standard convolutional layer
F = nn.Conv2d(m, n, k)

# Expand-CL with r
F1 = nn.Conv2d(m, r*m, 1)
F2 = nn.Conv2d(r*m, r*n, k)
F3 = nn.Conv2d(r*n, n, 1)

# contracting
from exp_cifar.utils.compute_new_weights \
    import compute_cl, compute_cl_2
tmp = compute_cl(F1, F2)
tmp = compute_cl_2(tmp, F3)
F.weight.data, F.bias.data = tmp['weight'], tmp['bias']

# test
res_cl = F3(F2(F1(imgs)))
res_F = F(imgs)
print('Contract from Expand-CL: %.7f' % (res_cl-res_F).sum())# <10^-5

# Expand-CK
# k = 5, l=2
```

```
F1 = nn.Conv2d(m, r*m, 3)
F2 = nn.Conv2d(r*m, n, 3)

# contracting
from exp_cifar.utils.compute_new_weights import compute_ck
tmp = compute_ck(F1, F2)
F.weight.data = tmp['weight']
F.bias.data = tmp['bias']

# test
res_ck = F2(F1(imgs))
res_F = F(imgs)
print('Contract from Expand-CK: %.7f' % (res_ck-res_F).sum())# <10^-5
```

Listing A.1: Expansion and contraction of a standard convolutional layer

Pytorch code to expand and contract back a depthwise convolutional layer with a kernel size of 3:

```
# for depthwise conv, input channels=out channels
m = 4  # input channels
n = 4  # output channels
k = 3  # kernel size
r = int(4)  # expansion rate
imgs = torch.randn((8, m, 7, 7))

# original depthwise convolutional layer
F = nn.Conv2d(m, n, k, groups=m, bias=False)

# Expand-CL with r
F1 = nn.Conv2d(m, r*m, 1, groups=m, bias=False)
F2 = nn.Conv2d(r*m, r*m, k, groups=m, bias=False)
F3 = nn.Conv2d(r*m, n, 1, groups=m, bias=False)

# contracting
from exp_imagenet.utils.compute_new_weights \
    import compute_cl_dw_group, compute_cl_dw_group_2

tmp = compute_cl_dw_group(F1, F2)
tmp = compute_cl_dw_group_2(tmp, F3)

F.weight.data = tmp['weight']

# test
res_depthwise_cl = F3(F2(F1(imgs)))
res_F = F(imgs)
print('Contract from depthwise Expand-CL: %.7f' % (res_depthwise_cl-res_F).sum())
  # <10^-5
```

Listing A.2: Expansion and contraction of a depthwise convolutional layer

Bibliography

Allen-Zhu, Z., Li, Y., and Liang, Y. (2018a). Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv Preprint*.

Allen-Zhu, Z., Li, Y., and Song, Z. (2018b). A convergence theory for deep learning via over-parameterization. *arXiv Preprint*.

Alvarez, J. M. and Salzmann, M. (2016). Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*.

Alvarez, J. M. and Salzmann, M. (2017). Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*.

Arora, S., Cohen, N., and Hazan, E. (2018). On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *IEEE Transactions on Neural Networks*.

Bolya, D., Foley, S., Hays, J., and Hoffman, J. (2020). TIDE: A general toolbox for identifying object detection errors. In *European Conference on Computer Vision*.

Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., and Rother, C. (2014). Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision*.

Brachmann, E., Michel, F., Krull, A., Yang, M. Y., Gumhold, S., and Rother, C. (2016). Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Conference on Computer Vision and Pattern Recognition*.

Cai, Z., He, X., Sun, J., and Vasconcelos, N. (2017). Deep learning with low precision by half-wave gaussian quantization. In *Conference on Computer Vision and Pattern Recognition*.

Cai, Z. and Vasconcelos, N. (2018). Cascade R-CNN: Delving into high quality object detection. In *Conference on Computer Vision and Pattern Recognition*.

Carreira-Perpinan, M. A. and Idelbayev, Y. (2018). "learning-compression" algorithms for neural net pruning. In *Conference on Computer Vision and Pattern Recognition*.

Chen, G., Choi, W., Yu, X., Han, T., and Chandraker, M. (2017a). Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*.

Chen, G., Choi, W., Yu, X., Han, T., and Chandraker, M. (2017b). Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv Preprint*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv Preprint*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Conference on Computer Vision and Pattern Recognition*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv Preprint*.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017). Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*.

Dai, X., Jiang, Z., Wu, Z., Bao, Y., Wang, Z., Liu, S., and Zhou, E. (2021). General instance distillation for object detection. In *Conference on Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*.

Denil, M., Shakibi, B., Dinh, L., De Freitas, N., et al. (2013). Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*.

Di, Y., Manhardt, F., Wang, G., , Ji, X., Navab, N., and Tombari, F. (2021). SO-Pose: Exploiting self-occlusion for direct 6d pose estimation. In *International Conference on Computer Vision*.

Ding, X., Guo, Y., Ding, G., and Han, J. (2019). Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *International Conference on Computer Vision*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. (2019). CenterNet: Keypoint triplets for object detection. In *International Conference on Computer Vision*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*.

Figurnov, M., Ibraimova, A., Vetrov, D. P., and Kohli, P. (2016). PerforatedCNNs: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. (2018). Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*.

Guo, J., Han, K., Wang, Y., Wu, H., Chen, X., Xu, C., and Xu, C. (2021a). Distilling object detectors via decoupled features. *Conference on Computer Vision and Pattern Recognition*.

Guo, S., Alvarez, J. M., and Salzmann, M. (2020). Expandnets: Linear over-parameterization to train compact convolutional networks. In *Advances in Neural Information Processing Systems*.

Guo, S., Alvarez, J. M., and Salzmann, M. (2021b). Distilling image classifiers in object detectors. *Advances in Neural Information Processing Systems*.

Guo, S., Hu, Y., Alvarez, J. M., and Salzmann, M. (2022). Knowledge distillation for 6d pose estimation by keypoint distribution alignment. *arXiv Preprint*.

Guo, Y., Yao, A., and Chen, Y. (2016). Dynamic network surgery for efficient DNNs. *Advances in Neural Information Processing Systems*.

Han, S., Mao, H., and Dally, W. J. (2016). Deep Compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*.

Hao, K. (2019). Training a single ai model can emit as much carbon as five cars in their lifetimes. *MIT technology Review*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017a). Mask R-CNN. In *International Conference on Computer Vision*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*.

He, T., Shen, C., Tian, Z., Gong, D., Sun, C., and Yan, Y. (2019a). Knowledge adaptation for efficient semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*.

He, Y., Liu, P., Wang, Z., Hu, Z., and Yang, Y. (2019b). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Conference on Computer Vision and Pattern Recognition*.

He, Y., Zhang, X., and Sun, J. (2017b). Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision*.

Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., and Choi, J. Y. (2019a). A comprehensive overhaul of feature distillation. In *International Conference on Computer Vision*.

Heo, B., Lee, M., Yun, S., and Choi, J. Y. (2019b). Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *American Association for Artificial Intelligence Conference*.

Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv Preprint*.

Hodaň, T., Sundermeyer, M., Drost, B., Labbé, Y., Brachmann, E., Michel, F., Rother, C., and Matas, J. (2020). BOP challenge 2020 on 6D object localization. *European Conference on Computer Vision Workshops*.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *International Conference on Computer Vision*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv Preprint*.

Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K. (2016). Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv Preprint*.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Conference on Computer Vision and Pattern Recognition*.

Hu, Y., Fua, P., Wang, W., and Salzmann, M. (2020). Single-stage 6d object pose estimation. In *Conference on Computer Vision and Pattern Recognition*.

Hu, Y., Hugonot, J., Fua, P., and Salzmann, M. (2019). Segmentation-driven 6d object pose estimation. In *Conference on Computer Vision and Pattern Recognition*.

Hu, Y., Speierer, S., Jakob, W., Fua, P., and Salzmann, M. (2021). Wide-depth-range 6d object pose estimation in space. In *Conference on Computer Vision and Pattern Recognition*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*.

Huang, R., Pedoeem, J., and Chen, C. (2018). Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers. In *IEEE International Conference on Big Data*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv Preprint*.

Jaderberg, M., Simonyan, K., Zisserman, A., and kavukcuoglu, k. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*.

Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. *arXiv Preprint*.

Jin, J., Dundar, A., and Culurciello, E. (2015). Flattened convolutional neural networks for feedforward acceleration. In *International Conference on Learning Representations*.

Kantorovitch, L. (1958). On the translocation of masses. *Management science*.

Kawaguchi, K. (2016). Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2018). Generalization in deep learning. In *Mathematics of Deep Learning, Cambridge University Press. Prepint avaliable as: MIT-CSAIL-TR-2018-014, Massachusetts Institute of Technology*.

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., and Navab, N. (2017). SSD-6D: Making rgb-based 3d detection and 6d pose estimation great again. In *International Conference on Computer Vision*.

Kendall, A., Grimes, M., and Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *International Conference on Computer Vision*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Laurent, T. and von Brecht, J. (2018). Deep linear networks with arbitrary loss: All local minima are global. In *International Conference on Machine Learning*.

Law, H. and Deng, J. (2018). CornerNet: Detecting objects as paired keypoints. In *European Conference on Computer Vision*.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv Preprint*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in Neural Information Processing Systems*.

Lee, N., Ajanthan, T., and Torr, P. H. (2019). SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Lemaire, C., Achkar, A., and Jodoin, P.-M. (2019). Structured pruning of neural networks with budget-aware regularization. In *Conference on Computer Vision and Pattern Recognition*.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision*.

Li, H., De, S., Xu, Z., Studer, C., Samet, H., and Goldstein, T. (2017). Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*.

Li, Z., Wang, G., and Ji, X. (2019). CDPN: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *International Conference on Computer Vision*.

Lin, D., Talathi, S., and Annapureddy, S. (2016). Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Conference on Computer Vision and Pattern Recognition*.

Lin, T.-Y., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. *International Conference on Computer Vision*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., and Zitnick, L. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*.

Liu, B., Wang, M., Foroosh, H., Tappen, M., and Pensky, M. (2015). Sparse convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *European Conference on Computer Vision*.

Liu, Y., Chen, K., Liu, C., Qin, Z., Luo, Z., and Wang, J. (2019). Structured knowledge distillation for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. (2018). Rethinking the value of network pruning. In *International Conference on Learning Representations*.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *International Conference on Computer Vision*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). ShuffleNet V2: Practical guidelines for efficient cnn architecture design. In *European Conference on Computer Vision*.

McFee, B., Salamon, J., and Bello, J. (2018). Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Speech and Language Processing*.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. (2017). Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*.

Novikov, A., Podoprikhin, D., Osokin, A., and Vetrov, D. P. (2015). Tensorizing neural networks. In *Advances in Neural Information Processing Systems*.

Papandreou, G., Chen, L.-C., Murphy, K., and Yuille, A. L. (2015). Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv Preprint*.

Passalis, N. and Tefas, A. (2018). Learning deep representations with probabilistic knowledge transfer. In *European Conference on Computer Vision*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Peng, S., Liu, Y., Huang, Q., Zhou, X., and Bao, H. (2019). PVNet: Pixel-wise voting network for 6dof pose estimation. In *Conference on Computer Vision and Pattern Recognition*.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition*.

Redmon, J. and Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *arXiv Preprint*.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv Preprint*.

Reed, R. (1993). Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*.

Rigamonti, R., Sironi, A., Lepetit, V., and Fua, P. (2013). Learning separable filters. In *Conference on Computer Vision and Pattern Recognition*.

Romera, E., Álvarez, J. M., Bergasa, L. M., and Arroyo, R. (2018). ERFNet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). FitNets: Hints for thin deep nets. *arXiv Preprint*.

Ronneberger, O., P.Fischer, and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *International Conference on Acoustics, Speech, and Signal Processing*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Conference on Computer Vision and Pattern Recognition*.

Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., and Goldstein, T. (2019). The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. *arXiv Preprint*.

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Singh, S. P. and Jaggi, M. (2020). Model fusion via optimal transport. In *Advances in Neural Information Processing Systems*.

Stock, P., Joulin, A., Gribonval, R., Graham, B., and Jégou, H. (2020). And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations*.

Su, Z., Wang, Y., Shi, R., Zeng, W., Sun, J., Luo, F., and Gu, X. (2015). Optimal mass transport for shape matching and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Conference on Computer Vision and Pattern Recognition*.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive representation distillation. In *International Conference on Learning Representations*.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). FCOS: Fully convolutional one-stage object detection. In *International Conference on Computer Vision*.

Ullrich, K., Meeds, E., and Welling, M. (2017). Soft weight-sharing for neural network compression. In *International Conference on Learning Representations*.

Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer.

Wang, G., Manhardt, F., Tombari, F., and Ji, X. (2021). GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Conference on Computer Vision and Pattern Recognition*.

Wang, T., Yuan, L., Zhang, X., and Feng, J. (2019). Distilling object detectors with fine-grained feature imitation. In *Conference on Computer Vision and Pattern Recognition*.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*.

Wen, W., Xu, C., Wu, C., Wang, Y., Chen, Y., and Li, H. (2017). Coordinating filters for faster deep neural networks. In *International Conference on Computer Vision*.

Wu, B., Iandola, F., Jin, P. H., and Keutzer, K. (2016). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *arXiv Preprint*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International Conference on Machine Learning*.

Wu, S., Li, G., Chen, F., and Shi, L. (2018). Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv Preprint*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition*.

Yang, Z., Liu, S., Hu, H., Wang, L., and Lin, S. (2019). RepPoints: Point set representation for object detection. In *International Conference on Computer Vision*.

Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Conference on Computer Vision and Pattern Recognition*.

Yu, X., Liu, T., Wang, X., and Tao, D. (2017). On compressing deep models by low rank and sparse decomposition. In *Conference on Computer Vision and Pattern Recognition*.

Zagoruyko, S. and Komodakis, N. (2017). Paying More Attention to Attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*.

Zhang, L. and Ma, K. (2021). Improve object detection with feature-based knowledge distillation: Towards accurate and efficient detectors. In *International Conference on Learning Representations*.

Zhao, R., Hu, Y., Dotzel, J., De Sa, C., and Zhang, Z. (2019). Improving neural network quantization without retraining using outlier channel splitting. In *International Conference on Machine Learning*.

Zhou, Y. and Liang, Y. (2018). Critical points of linear neural networks: Analytical forms and landscape properties. In *International Conference on Learning Representations*.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., and Zhu, J. (2018). Discrimination-aware channel pruning for deep neural networks. *Advances in Neural Information Processing Systems*.

SHUXUAN GUO

+41 (0) 787320031
shuxuan.guo@epfl.ch / gsxuan.work@gmail.com
linkedin.com/in/shuxuan-guo
github.com/GUOShuxuan

RESEARCH INTERESTS

Machine Learning, Computer Vision, Efficient Deep Learning, Network Compression, Knowledge Transfer, Object Detection, 6D Pose Estimation

EDUCATION

Ph.D. Candidate | ***Computer Science*** Sep. 2017 - present
Computer Vision Laboratory (CVLab)
École Polytechnique Fédérale de Lausanne (EPFL) Lausanne, Switzerland

- Dissertation Topic: *Improving the Training of Compact Neural Networks for Visual Recognition*
- Thesis directors: Dr. Mathieu Salzmann, Dr. Jose M. Alvarez and Prof. Pascal Fua

M.Sc. | ***Control Science and Engineering*** Sep. 2014 - Jan. 2017
National University of Defense Technology (NUDT) Changsha, China

- GPA: 3.83/4.0
- Research topic: Computer Vision and Pattern Recognition
- Supervisors: Prof. Li Liu and Prof. Songyang Lao

B.Eng. | ***Information System Engineering*** Sep. 2010 - Jun. 2014
National University of Defense Technology (NUDT) Changsha, China

- GPA: 3.59/4.0
- Excellent Graduate Student of NUDT in 2014
- Excellent Undergraduate Thesis

PUBLICATIONS

- **Shuxuan Guo**, Yinlin Hu, Jose M. Alvarez and Mathieu Salzmann. *Knowledge Distillation for 6D Pose Estimation by Keypoint Distribution Alignment.* Under review.
- **Shuxuan Guo**, Jose M. Alvarez and Mathieu Salzmann. *Distilling Image Classifiers in Object Detectors.* NeurIPS2021.
- **Shuxuan Guo**, Jose M. Alvarez and Mathieu Salzmann. *ExpandNets: Linear Over-parameterization to Train Compact Convolutional Networks.* NeurIPS2020 **Spotlight**.

WORK EXPERIENCE

Deep Learning Research Intern June 2020 - June 2021
NVIDIA Zurich, Switzerland/Santa Clara, CA, USA

- Supervisor: Dr. Jose M. Alvarez
- Developed efficient compact object detection models in autonomous driving.

Exchange student November 2015 - July 2016
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation Chinese Academy of Sciences (CASIA) Beijing, China

- Supervisors: Dr. Wei Wang and Prof. Liang Wang
- Proposed a method based on attention model for scene classification and understanding.

PROJECTS AND RESEARCH

Knowledge distillation in 6D pose estimation June 2021 - present
CVLab, EPFL Switzerland

- Built up a state-of-the-art baseline for compact networks in 6D pose estimation task.
- Investigated knowledge distillation for the first time in the context of 6D pose estimation.
- Proposed a distillation approach aligning the teacher and student keypoint distributions to transfer 6D pose estimation task-driven knowledge.

Cross-task and Cross-architecture knowledge distillation in object detection June 2020 - June 2021
NVIDIA & EPFL Switzerland

- Research project during the internship at NVIDIA.
- Introduced the idea of cross-task and cross-architecture knowledge distillation, specifically classifier-to-detector, to improve the performance of a student detector using a classification teacher.
- Proposed a distillation approach to improve both the student detector's classification accuracy and localization performance.

ExpandNets improving compact models for multiple visual tasks September 2017 - June 2020
EPFL Switzerland

- Proposed strategies to linearly expand a compact network, which facilitate network training by improving both neural network optimization and generalization, achieving better performance at inference without information loss.
- Demonstrated the benefits of the ExpandNets on several visual recognition tasks, including image classification, object detection and semantic segmentation.

HONORS & AWARDS

Doctoral Fellowship 2017
EDIC, EPFL

Excellent Graduate Student of NUDT June 2014
Top 2% of all fresh graduates

Excellent Undergraduate Thesis June 2014

Meritorious Winner & Matlab Innovation Award September 2013
China Undergraduate Mathematical Contest in Modeling (CUMCM)

Meritorious Winner Winter 2012
Mathematical Contest in Modeling (MCM)

TEACHING & SERVICE EXPERIENCE

Teaching assistant: Introduction to machine learning (CS-233) 2019, 2020.
Review service: NeurIPS, ICLR, ECCV, AAAI, Workshop on Autonomous Driving (CVPR)

SKILLS

Programming: Python (NumPy, SciPy, Matplotlib, Pandas), C/C++, Matlab
Deep Learning: Pytorch, Tensorflow, Caffe
Languages: English(Fluent), Chinese(Native), French(beginner)
Document Creation: Microsoft Office Suite, LaTex, Markdown