Imperial College of Science, Technology and Medicine
Department of Electrical and Electronic Engineering

Towards Resource Efficient Vision Models

Roy Miles

Submitted in part fulfillment of the requirements for the degree of PhD in Electrical and Electronic Engineering and the Diploma of Imperial College London

March 2024

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution 4.0 International Licence (CC BY).

Under this licence, you may copy and redistribute the material in any medium or format for both commercial and non-commercial purposes. You may also create and distribute modified versions of the work. This on the condition that you credit the author.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Statement of Originality

I certify that this thesis entitled "Towards Resource Efficient Vision Models using Knowledge Distillation and Pruning" is my original work. I assert that, except where acknowledged, the material contained herein is the result of my own efforts and has not been submitted in whole or in part for any other degree at any university or institution.

Contents

Abstract **1**

Acknowledgements **3**

1 Introduction **4**

1.1 Network Pruning 6

1.1.1 Unstructured pruning 6

1.1.2 Structured pruning 8

1.2 Weight Quantisation 9

1.3 Low-rank factorization 9

1.4 Knowledge Distillation 11

1.5 Thesis Outline 13

2 Compressing Local Descriptor Models **16**

2.1 Introduction 16

2.2 Related Work 18

2.3 Compressed Descriptor Model 19

2.4 Experimental results 22

2.5 Conclusions 25

2.6 Supplementary 26

2.6.1 Compression ratio 26

2.6.2 Computational cost 27

3 Reconstructing Pruned Filters using Cheap Spatial Transformations 29

3.1 Introduction 29

3.1.1 Related work 31

3.2 Method 32

3.2.1 Constructing diverse convolutional filters. 33

3.2.2 Using pruning to select the filter templates. 34

3.2.3 Performance and efficient implementations. 36

3.2.4 Channel connectivity and feature compression. 36

3.2.5 Computational cost and parameter efficiency. 38

3.3 Experiments 40

3.3.1 Experimental results on CIFAR-10 40

3.3.2 Experimental results on ImageNet 41

3.3.3 Ablation experiments 42

3.4 Conclusion and Future work 44

4 Cascaded Channel Pruning using Hierarchical Self-Distillation **45**

4.1 Introduction 45

4.2 Related Work 47

4.3 Method 49

4.4 Experiments 54

4.5 Ablation studies 58

4.6 Conclusion 59

4.7 Supplementary 59

5 Information Theoretic Representation Distillation **63**

5.1 Introduction 63

5.2 Related Work 65

5.3 Preliminaries 67

5.4 Information Theoretic Loss Functions 70

5.5 Experiments 74

5.6 Discussion 81

5.7 Conclusion 82

5.8 Supplementary 82

5.8.1 Mutual Information Loss 82

5.8.2 Correlation Loss 83

6 Understanding the Role of the Projector in Knowledge Distillation **89**

6.1 Introduction 90

6.2 Related Work 91

6.3 Understanding the Role of the Projector 93

6.4 Benchmark Evaluation 100

6.4.1 Classification on CIFAR100 and ImageNet 100

6.4.2 Data efficient training for transformers 101

6.4.3 Object Detection on COCO 104

6.5 Conclusion 105

6.6 Supplementary Material 106

6.6.1 Measure of translational equivariance 106

6.6.2 Few-shot distillation experiments 107

6.6.3 Model Architectures 108

7 Conclusion **110**

7.1 Summary of Contributions 110

7.2 Limitations 111

7.3 Future Work 111

7.3.1 Data-Efficient Training 111

7.3.2 Multi-Modality Models 112

Bibliography **112**

List of Figures

1.1 The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge from 2010 to 2015 [131]. These state-of-the-art models are becoming increasingly more computationally and memory intensive, which has limited their deployment on resource constrained devices. 5

1.2 Various schemes have been proposed for pruning the convolutional weights in a CNN. Generally speaking these schemes can be categorised into unstructured and structured pruning, whereby structured pruning preserves the underlying architecture and enables the layer to remain efficient on standard consumer hardware. 9

1.3 Tensor decomposition can be graphically expressed using tensor networks. Above shows two common decompositions of a 4-way convolutional weight tensor $\mathcal{W}$ with a receptive field size of $K_h \times K_w$, input depth C and output depth N. . . . 10

1.4 Knowledge distillation can be broadly categorised into logit, representation, and ensemble methods. In the output space, the metric is already defined from the downstream task objective, however, for intermediate representations heuristics measures must be adopted. Constructing Gram G, or correlation C matrices has been shown to effectively extract the structural information needed to distill these intermediate representations. 11

2.1 The proposed Convolutional-Depthwise-Pointwise (CDP) layer partitions the input tensor across the depth. Most of the computational resources are then reserved for only a subset of the input features, while the rest use an efficient depth-wise convolution. The resulting features are then concatenated and aggregated using a pointwise convolution. 17

2.2 Depthwise separable convolution (left) and the Tucker decomposition applied to a convolutional layer (right) [88]. 19

2.3 Example of the convolutional weight slices $\mathcal{W}[:,:,i,j]$ from the pre-trained HardNet++ model for layers 6 and 7, Where i, j is a given spatial coordinate in the receptive field. Note that the columns correspond to a given input channel index and the weights are scaled to be in the range $[0, 255]$. 20

3.1 Constructing more convolutional filters using cheap spatial transformations. (a) Original convolutional layer. (b) Depth-wise separable layers, which fully decouple the spatial and depthwise aggregation of features. (c) Proposed layer expressed new filters as spatial transformations of a smaller set of templates. . . 30

3.2 Too few templates can overly compress the input features. We propose to introduce a group parameter to naturally balance the expressiveness of the both the template and transformation stages. 34

3.3 Visualising learned transformations for a given layer. (left) original learned filters. (middle) expressive filters using affine template transformations. (right) pruning filters. Both the vanilla pruned layer and the decomposed layers use a pruning rate of 0.7. Each column represents a filter for a given output channel and black pixels represent zero entries. 34

3.4 Comparison with a pruned baseline and the importance of which pruning heuristic. (left) Performance of a VGG16 network trained using affine (with) and scalar (without) spatial transformations with the same training methodology. (right) show the performance trade-off at different pruning rates using both a magnitude-based and a gradient-based saliency. 35

3.5 Importance of channel connectivity and the group extension. (left) Using more groups enables a smaller bottleneck ratio, which improves the top-end accuracy. (right) shows that increasing the minimum number of templates also reduces feature compression but at a much larger overall cost. 37

3.6 Comparing the pruning rates of each layer using different saliency measures. We see that a magnitude based (MAG) criterion exhibits a more uniform pruning rate than gradient based measures (FO). 40

3.7 Latency of the proposed decomposition with an efficient implementation. (left) Compute time of an original VGG16 network with and without the decomposition at various pruning rates. (right) Cumulative contribution of each operation on the overall on-device compute time. 44

4.1 Proposed hierarchical self-distillation (HSD) strategy for channel pruning. Each of the models are jointly trained with shared convolutional weights but with independent binary masks, batch normalisation layers, and classification layers. The lesser constrained models provide knowledge distillation and importance score gradients down the hierarchy. The frozen teacher for model T_N has been omitted for clarity. 47

4.2 Left: Top-1 accuracy's of each model across the joint training step and the fine-tuning step. The training consists of one student T_0 and 3 TA's that are trained on the CIFAR10 dataset. Right: Accuracy and performance comparisons on the CIFAR100 dataset using the MobileNetV1 model. 54

4.3 Evaluation of the computational complexity (left) and the number of parameters (right) for a student model with a varying number of teaching assistants. Each data point on the graph is ordered according to their pre-defined filter-pruning ratio using the modified VGG16 architecture on the CIFAR10 dataset. 58

4.4 The layer-wise pruning for a student and two TA models trained using cascaded pruning. From left to right are models T_0, T_1, and T_2 respectively. Each TA uses the VGG16 architecture and is jointly trained on the CIFAR10 dataset. 60

5.1 Information theoretic representation distillation (ITRD) involves two distinct losses, namely a correlation loss and a mutual information loss. The former loss maximises the correlation between the student and teacher, while the latter maximises a quantity resembling the mutual information that aims to transfer the intra-batch sample similarity. 64

5.2 Top-1 Accuracy on ImageNet v.s. training efficiency with a ResNet-18 as the student and a pre-trained ResNet-34 as the teacher. For CRCD, the training efficiency was evaluated using the authors unofficial implementation, while this accuracy is reported in their paper. 77

5.3 Accuracy (%) when varying both the correlation loss (left) and mutual information loss (right) weightings. 78

5.4 Accuracy (%) when varying both the correlation loss (left) and mutual information loss (right) weightings. 85

5.5 Training epochs *vs* validation accuracy for VGG13→VGG8 CIFAR 100 distillation. Zoomed-in regions show that our method converges faster to a higher accuracy. 87

6.1 Proposed feature distillation pipeline using three distinct components: linear projection (a), batch norm (b), and a *LogSum* distance (c). We provide an interpretable explanation for each each of these three components, which results in a very cheap and effective recipe for distillation. 92

6.2 Evolution of singular values of the projection weights $\mathbf{W}_p$ under three different representation normalisation schemes. The student is a Resnet-18, while the teacher is a ResNet-50. The three curves shows the evolution of singular values for the projector weights when the representations undergo no normalisation, L2 normalisation, and batch norm respectively. 93

6.3 Correlation between input-output features using different projector architectures. All projector architectures considered will gradually decorrelate the input-output features. Although this decorrelation is attributed to the layer removing irrelevant information, it can degrade the efficacy of distilling through to the student backbone. 95

List of Tables

2.1 The CDP SuperPoint variants are able to achieve significant model compression with minimal degradation against the related detector and descriptor performance metrics. The computational complexity Ops. is measured in GFLOPs, and it is the contribution of the VGG16 backbone and the separate detector/descriptor heads. 23

2.2 Comparison of using depthwise-separable layers, Tucker decomposition, and a proposed scheme for combining the two (TDW). The number(s) in the braces indicates the layers replaced. The image matching and patch retrieval accuracy is evaluated with mean Average Precision (mAP). 23

2.3 Applying different depthwise offsets for the HardNet model with CDP layers. The results are evaluated on the HPatches benchmark and averaged over the Easy, Hard, and Tough distributions. 24

3.1 Comparison to other pruning methods on CIFAR10. Each model is trained using a magnitude based measure for selecting the filter templates, number of groups = 2, and with a minimum of 8 template filters per layer to avoid catastrophic pruning. 41

3.2 Comparison to other pruning methods on ImageNet-1K. Our model is trained from random initialisation and with a simple magnitude based criterion. Using number of groups = 2 and with a minimum of 32 templates per layer. 42

3.3 Ablating the family of transformations. Increasing the expressivity of transformations has a small improvement in performance, suggesting that most of the network capacity is reserved for depthwise feature aggregation. 43

4.1 Comparison to other filter-level pruning methods on the CIFAR10 benchmark and with the VGG16 architecture. For each model, the drop in accuracy is with reference to their own baseline. 56

4.2 Top-1 Accuracy and pruning ratios on the ImageNet2012 validation split using the ResNet50 model. The accuracy drops are reported in comparison to their corresponding baseline. The calculations for these baseline performance metrics are covered in the supplementary materials. 57

4.3 Performance statistics for the ResNet50 architecture on the ImageNet2012 dataset. 60

4.4 Pruning % in each layer as a result of cascaded pruning on the CIFAR10 dataset and with the VGG16 architecture at varying filter-pruning ratios. The last two layers (13 & 14) are the two dense classification layers which are not masked. . . 61

4.5 Accuracy and performance metrics for two efficient VGG16 variants trained from random initialisation on the CIFAR10 dataset. Group-g indicates the use of group convolutions with g groups, while Standard-s uses s% width scaling for all the convolutional layers. 61

5.1 CIFAR-100 test *accuracy* (%) of student networks trained with a number of distillation methods. The best results are highlighted in **bold**, while the second best results are <u>underlined</u>. The mean and standard deviation was estimated over 3 runs. Same-architecture transfer experiments are highlighted in blue, whereas cross-architectural transfer is shown in red. 75

5.2 Relative performance improvement (averaged over all architecture pairs in table 5.1) of the correlation and mutual information based losses against ReviewKD, WCoRD and $\mathcal{L}_{corr}$ only. 76

5.3 Accuracy (%) when varying α in the correlation loss for CIFAR-100 ResNet50→ MobileNetV2 distillation. 78

5.4 Transferability of the representations from CIFAR-100 to STL-10 and TinyImageNet. Only the linear classifier heads of each model are fine-tuned on the target datasets. The top-1 classification accuracies are reported (%). 79

5.5 Performance comparison with the state-of-the-art on CIFAR-10 for binary networks. All models, except full precision, use a bit length of 2 for the weights and activations. 80

5.6 Question Answering on SQuAD 1.1. The teacher architecture, BERT, contains 12 layers, whereas the students, *T6* and *T3*, follow the same architecture as BERT but with 6 and 3 layers respectively. 81

5.7 Accuracy (%) with and without the $\log_2$ data transformation. The experiments were performed for CIFAR-100 ResNet50→ MobileNetV2 distillation. 83

5.8 Accuracy (%) with the RBF kernel for the $\mathcal{L}_{gram}$ with different kernel sizes σ. The experiments were performed for CIFAR-100 ResNet50→ MobileNetV2 distillation. 83

5.9 Accuracy (%) when varying α in the correlation loss for CIFAR-100 ResNet50→ MobileNetV2 distillation. 85

5.10 Relative overhead in terms of memory and training time against main competing distillation methods on ImageNet. Training time used a variable batch size to fit a pre-defined memory limit, while the memory experiments were using a fixed batch size. 87

6.1 Normalisation ablation for distillation across a range of architecture pairs on an ImageNet-1K 20% subset. Although distillation improves performance with a variety of normalisation schemes, we find batch normalisation is consistently the most effective. 98

6.2 Ablating the importance of α. Distillation is generally robust for various values of α, but consistently optimal in range 4-5 across various architecture pairs. . . . 98

6.3 LogSum ablation across various architecture pairs. Left: 20% subset. Right: Full ImageNet. The soft maximum function provides consistent improvement across both the CNN→CNN and ViT→CNN distillation settings. 99

6.4 KD between Similar and Different Architectures. Top-1 accuracy (%) on CIFAR100. Bold is used to denote the best results. All reported models are trained using pairs of augmented images. Those reported in the top box use RandAugment [44] strategy, while those in the bottom box use pre-defined rotations, as used in SSKD. † denotes reproduced results in a new augmentation setting using the authors provided code. 99

6.5 Data-efficient training of transformers and CNNs on the ImageNet-1K dataset. Unless specified, all student models are trained for 300 epochs. 103

6.6 Top-1 and Top-5 error rates (%) on ImageNet. ResNet18 as student, ResNet34 as teacher. 103

6.7 Measure of translational equivariance of a DeiT-S transformer model trained with and without distillation. These results confirm that distillation can transfer explicit inductive biases from the teacher. 104

6.8 Object detection on COCO. (top) We report the standard COCO metric of mAP averaged over IOU thresholds in [0.5 : 0.05 : 0.95] along with the standard PASCAL VOC's metric, which is the average mAP@0.5. (bottom) For the R-CNN results, we report the mAP and AP50 metrics to enable a consistent comparison with ReviewKD. 105

6.9 Ablating the importance in the choice of metric function using a ResNet34 and a ResNet18 student on the ImageNet dataset. The loss modifications are highlighted in red. 106

Abstract

This PhD thesis focuses on improving the efficiency of deep neural networks for computer vision tasks by employing two key techniques: distillation and pruning. Distillation involves training a smaller network to mimic the behavior of a larger, more complex network, thereby reducing the number of parameters required for accurate inference, reduce the computational complexity, and, in some cases, improve the data-efficiency. Pruning, on the other hand, is typically a post-processing technique that involves removing redundant parameters from the trained network to further reduce its size and computational requirements. This thesis explores various approaches in combining these techniques for enhancing the efficiency of vision models by incorporating domain knowledge and designing novel distillation and pruning techniques. Some of the work presented here also touches upon an orthogonal direction, known as tensor decomposition, which parameterise the weights in a more compact and efficient manner. Overall, this thesis contributes to the development of more efficient and practical deep learning models for computer vision applications. Some examples of these applications may be autonomous driving, surveillance, and augmented virtual reality, but the main emphasis being deploying these models on resource constrained devices, such as mobile phones. The results presented also show various insights into how these efficient models can be designed and trained, thus incorporating all the components of a standard machine learning pipeline, from the architecture design through to the deployment on device.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Krystian Mikolajczyk, for their guidance and encouragement throughout the course of this research project. Their invaluable insights and expertise have been instrumental in shaping my work and helping me to grow as a researcher.

I am deeply grateful to all of my colleagues in the MatchLab group for creating a supportive and intellectually stimulating environment. In particular, I would like to thank them for their helpful discussions and for their friendship.

Finally, I would like to express my heartfelt thanks to my family and friends for their unwavering support and encouragement. This achievement would not have been possible without their love and encouragement.

Chapter 1

Introduction

Over the past few decades, the field of machine learning has witnessed remarkable progress and growth, with numerous algorithms and models developed to tackle complex problems in various domains. One area where machine learning has made significant strides is computer vision, which aims to enable computers to interpret and understand the visual world. This thesis focuses on deep learning, a subfield of machine learning, and its immense success in solving complex visual tasks that were once considered unattainable for computers. Deep learning, inspired by the structure and function of the human brain, refers to a class of artificial neural networks (ANNs) with multiple hidden layers. This architecture allows deep learning models to automatically learn hierarchical features from raw input data, making them particularly adept at handling high-dimensional data such as images. In recent years, deep learning has consistently outperformed traditional machine learning algorithms in various computer vision tasks, including image classification [69], object detection [146], semantic segmentation [150], and scene understanding [42]. The introduction of Convolutional Neural Networks (CNNs) has been a key factor in the success of deep learning in computer vision. CNNs exploit the spatial relationships within images by employing convolutional layers, which apply filters to local regions of the input data. This unique structure allows CNNs to learn robust and discriminative features that can generalize well to unseen data. In addition, advances in hardware, particularly Graphics Processing Units (GPUs), have facilitated the training of large-scale deep learning models on

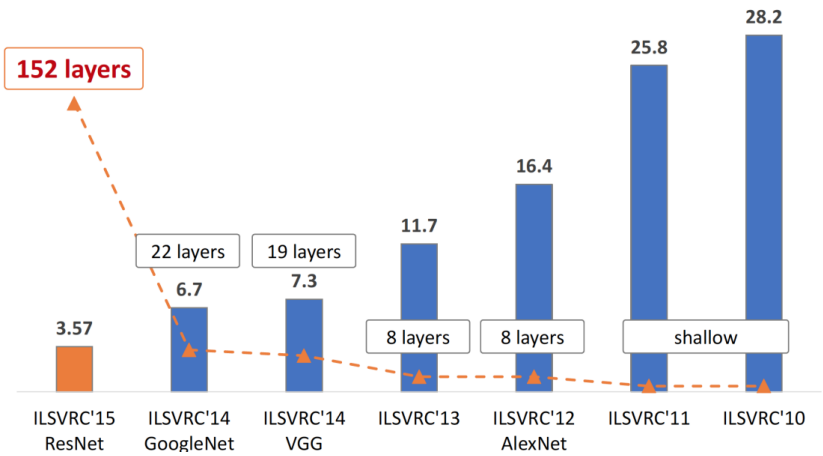


Figure 1.1: The evolution of the winning entries on the ImageNet Large Scale Visual Recognition Challenge from 2010 to 2015 [131]. These state-of-the-art models are becoming increasingly more computationally and memory intensive, which has limited their deployment on resource constrained devices.

large datasets [153, 130, 89, 192], further propelling their performance. Deep learning's success in computer vision has led to numerous practical applications in various industries, including healthcare, automotive, entertainment, and security. For instance, deep learning models are now capable of detecting diseases in medical images with high accuracy and aiding doctors in diagnosing patients more efficiently. Similarly, the automotive industry has benefited from the advancements in object detection and scene understanding for the development of autonomous vehicles.

Figure 1.1 shows the early trend of model complexity over the years on the popular ImageNet image classification task. In very recent years, we have now seen models exceed a billion, or even a trillion, parameters and this rate of growth has not been matched by the compute available on mobile phones. Fortunately, compressing deep learning models is an active research area with the goal of making it possible to deploy these powerful machine learning models on devices with limited computational and storage resources. Specific examples of these devices can be mobile phones or embedded devices for Internet of Things (IoT) applications. However, the limitations of these models is not only restricted to the latency and memory, but also the power consumption. More specifically, a smaller model will typically incur a much lower average power consumption and thus prolong the devices battery life. Another motivation for this research is with regards to data privacy. In some cases, it may not be practical or desirable to send

sensitive data to a remote server for processing. By compressing the model and deploying it on the device itself, it is possible to perform the computation locally, which can help protect the privacy of the data. In this chapter, we introduce the current trend of deep learning research and its difficulty in aligning with the aforementioned objectives and their constraints. Following this we outline some common orthogonal research areas for approaching this task through either improving the training procedure or addressing and removing the emerging redundancy of the trained models. This chapter provides a general overview of the field of pruning, low-rank decomposition, quantisation, and knowledge distillation. The more concise and contextually relevant discussions are provided at the start of each chapter.

1.1 Network Pruning

Most deep learning models benefit from training in the highly over-parameterised regime [91, 69]. Through a dense sampling of the data, these models can then learn to smoothly interpolate over the low-dimensional latent manifold for which the data inherently lies [53]. Unfortunately, this regime leads to very large models that are incompatible with embedded devices, however, it does suggest significant parameter redundancy. This redundancy can be explicitly exploited as a post processing technique, described as pruning.

Pruning involves removing unnecessary connections or parameters from the model. Setting the weights to zero is an equivalent action to pruning said weight and is what is commonly adopted in practice through learning a binary mask over the weights. A highly pruned network is described as having a high level of weight sparsity and these networks should then be small enough to fit on a embedded device, such as a mobile phone.

1.1.1 Unstructured pruning

Unstructured pruning aims to remove individual connections or neurons based on their importance. It is often described as a fine-grain pruning strategy that can achieve very high levels of

sparsity. However, this sparsity does not often translate to the same reduction in latency due to the need for sparse hardware or software libraries. Unstructured pruning can be used alone or in combination with other techniques, such as weight sharing or quantization, to further compress and optimize the model.

Sparsity inducing regularization and iterative pruning. A natural approach to induce this sparsity among weights is to restrict their magnitude using some sparsity inducing regularization term during training. One example is the l_0 norm, which restricts the number of non-zero weights in the network [110]. However, it is not commonly used as it lacks convexity and is non-differentiable. The l_1 norm also induces sparsity in the weights but has the benefit of being convex and being much more computationally efficient. In practise, it is very useful to target specific levels of sparsity to meet pre-defined resource constraints, however, this is difficult to enforce using regularisation losses alone. This has led to the development of iterative pruning/fine-tuning pipelines coupled with some heuristic measures for the saliency of weights [92]. At each pruning step, those weights with a low saliency are stripped from the network and the drop in accuracy is recovered through the subsequent fine-tuning. This line of research is very tightly coupled with the lottery ticket hypothesis, which is discussed in the next section.

Lottery ticket hypothesis. The lottery ticket hypothesis (LTH) suggests that neural networks contain small, sub-networks that can be trained in isolation to achieve high accuracy on a given task [170]. When a neural network is randomly initialized, there are some sub-networks, or "winning tickets", that are capable of achieving high accuracy on a given task. These winning tickets can be identified and trained in isolation using iterative unstructured pruning techniques, such as magnitude pruning.

Several studies have confirmed the existence of winning tickets and the effectiveness of the lottery ticket hypothesis. Unfortunately, its extension to larger models and larger datasets is difficult as a result of its connection with the stability of linear modes and SGD noise [55]. More concretely, this work has suggested that an earlier checkpoint can be used in favour of the original initialisation to ensure stability in the LTH procedure. Other works have provided

additional insights into the underlying pruning pipeline [213], transferring tickets between models [33], or even learning random weight subnetworks that can generalise well. Although the LTH is theoretically interesting, its motivation in deploying small models on mobile devices is limited due to the inherent sparse operations needed for inference as a result of the subnetworks being derived from unstructured pruning techniques. However, the LTH is still an active area of research, with many open questions and opportunities for further investigation.

1.1.2 Structured pruning

Structured pruning attempts to alleviate the practical shortcoming of unstructured pruning. By sacrificing some degree of sparsity, we can ensure that the pruned model can still be run efficiently on standard GPU hardware. This works in practise because GPUs are optimised for dense matrix multiplications and, as an example, removing entire filters is equivalent to simply reducing the number of rows/columns for this operation. However, in the unstructured setting, the model would need to to keep track of all the zeros and perform some form of selective computation, which requires a lot of expensive memory movement operations. Although sparse inference is an active field of research both in the hardware and software domain, it has yet to make a strong impact in the wider or commercial setting.

Filter pruning involves the removal of entire convolutional filters (see figure 1.2). This pruning strategy preserves the underlying architecture, thus enabling the use of standard convolution operations to be maintained. Similarly for unstructured pruning, the filters can be ranked based on some heuristic criteria such as the magnitude [94], geometric median [70], or the batch statistics [133]. Liu *et al.* [108] observe some limitations in the iterative pruning and fine-tuning procedure, especially when choosing to prune filters. They find that re-training the pruned network from scratch can yield comparable or superior performance than doing any iterative pruning and fine-tuning. Although it is difficult to find this subnetwork without adopting any iterative pruning and fine-tuning, it has opened lots of new research directions, such as zero-shot and few-shot pruning [31]. We direct the reader to [103, 97] for a thorough survey on the state of pruning and its various extensions.

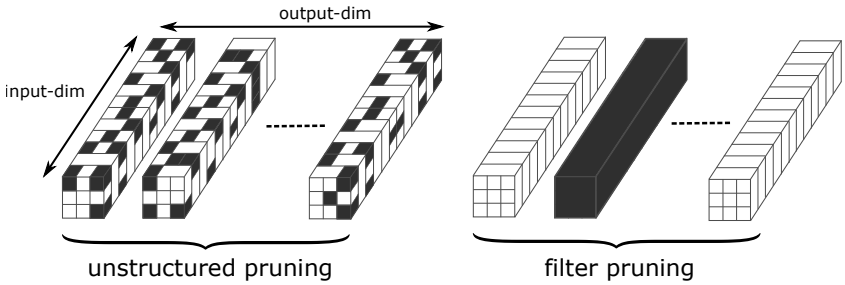


Figure 1.2: Various schemes have been proposed for pruning the convolutional weights in a CNN. Generally speaking these schemes can be categorised into unstructured and structured pruning, whereby structured pruning preserves the underlying architecture and enables the layer to remain efficient on standard consumer hardware.

1.2 Weight Quantisation

Quantization involves reducing the number of bits used to represent the model weights and activations. This can be done by mapping the original values to a smaller set of discrete values, such as 8-bit integers or binary values. Quantization can significantly reduce the model size and the amount of memory required to store the model, but it can also degrade the model's performance when stretched to very low bit-widths.

Binary quantisation is the most extreme quantisation scheme, whereby the weights are reduced to two values (typically 0 or 1). In this setting the convolutional layers can be implemented through simple logic and binary operations, such as XNOR and pop-count [145, 23, 181] which can be very efficiently executed on CPUs and FPGAs. Similarly, AdderNet [27] propose a surrogate template matching operation using additions only [27]. In our presented work on information-theoretic representation distillation [122] we provide an additional experiment on a standard binary quantisation benchmark. We find that our proposed distillation loss couples most favourably with SoTA binary quantisation than other distillation method, and in doing so sheds light to a new SoTA for a binary ResNet-18 on CIFAR10.

1.3 Low-rank factorization

Low-rank factorization involves approximating the model weights with a composition of low-rank tensors. This factorisation can reduce the total number of parameters and can improve

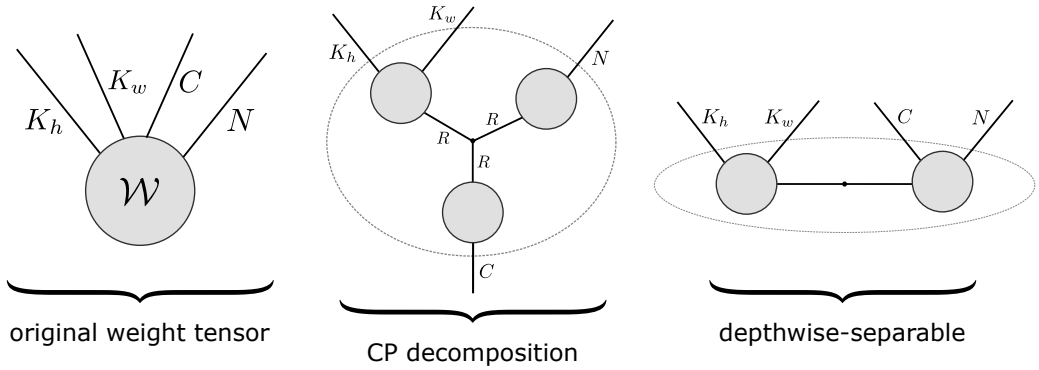


Figure 1.3: Tensor decomposition can be graphically expressed using tensor networks. Above shows two common decompositions of a 4-way convolutional weight tensor $\mathcal{W}$ with a receptive field size of $K_h \times K_w$, input depth C and output depth N.

the computational complexity for inference. Historically, this problem can be approached by flattening dimensions and applying techniques such as singular value decomposition (SVD) or matrix factorization [128]. Unfortunately, this flattening operation destroys a lot of the structural information and complex interactions between dimensions [159]. This has motivated the use of multilinear algebra to provide more structurally rich decompositions of the weights. This also gives rise to the use of tensor networks, which provide a graphical framework for multilinear decomposition, and has shown promising results [184, 132]. However, finding the low-rank decomposition of a tensor (order ≥ 3) is inherently an NP-hard problem and even finding an optimal low-rank approximation is ill-posed [161]. This is why iterative methods are commonly used that minimize the reconstruction error under an L_p norm. EinConv [65] proposed to explore the space of convolutional weight decomposition using neural architecture search techniques. They observed that the heuristically derived depthwise-separable convolutions [28] performed the best. These depthwise-separable convolutions have since been extended by addressing the increased cost of the pointwise convolution [114, 164]. Our work on the CDP layer also addresses this problem on the task of descriptor learning. We graphically show a select few tensor decompositions of convolutional weights in figure 1.3, but direct the reader to [12] for a more comprehensive study on this topic and its application in deep learning in general. More recently, concepts from this field of tensor decomposition have been modified and applied to vision transformers with a lot of success [39, 188]. This is currently a very open and active field of research.

On a related thread, geometric deep learning [21, 22] describes a general perspective of deep learning through the lens of data symmetries. We have described how tensor decomposition provides a framework for parameterising layer weights, and this parameterising can naturally enforce these symmetries into the architecture. Later in this thesis we present a low-rank decomposition that couples the construction of convolutional filters with learned group actions in this way. By combining this decomposition with pruning, we are then able to design very small and efficient networks.

1.4 Knowledge Distillation

Knowledge distillation involves training a smaller model to mimic the behavior of a larger, pre-trained model. Historically, the smaller model is trained to produce outputs that are similar to the larger model when given the same input. However, recent works have shown that distilled models are not always in agreement with the teacher, thus the confounding factors contributing to the improved generalisation of the student is an active area of research.

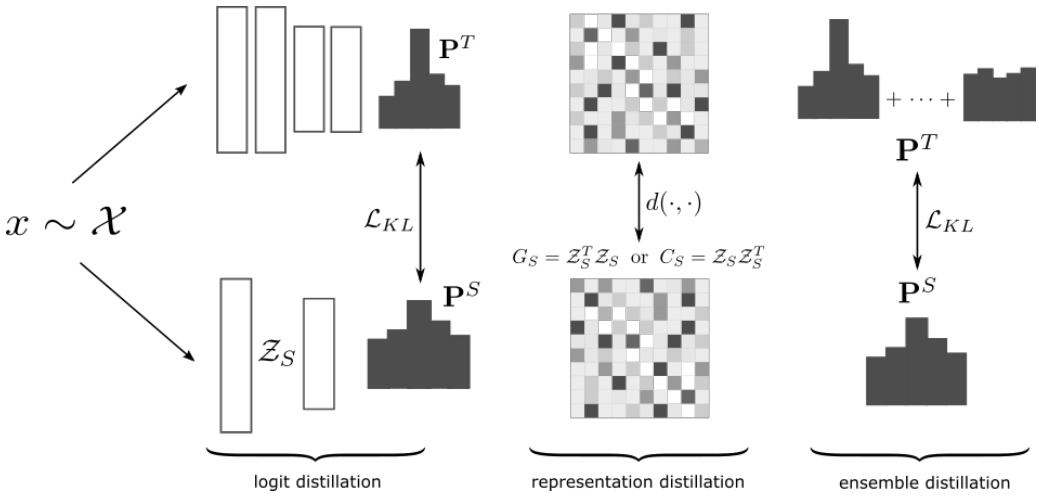


Figure 1.4: Knowledge distillation can be broadly categorised into logit, representation, and ensemble methods. In the output space, the metric is already defined from the downstream task objective, however, for intermediate representations heuristics measures must be adopted. Constructing Gram G, or correlation C matrices has been shown to effectively extract the structural information needed to distill these intermediate representations.

Logit Distillation initially opened up the field of knowledge distillation using an image classification task. The teacher's predictions are used as pseudo-ground truth labels for the student [73]

in addition to a temperature term τ for softening the probabilities. The intuition is that the student can then learn the correlation between classes, which is not available in the ground truth one-hot encoded labels. This additional knowledge is described as "dark knowledge". Decoupled Knowledge Distillation [211] is an example of work that has improved this general framework through decoupling and re-normalising the incorrect and correct probabilities.

Representation Distillation is a natural extension of logit distillation on the latent space immediately before the classifier. Its adoption enables the application to other tasks, such as regression and segmentation, while also preserving a lot of information about the input that may be useful for the distillation loss itself. Representation distillation was originally proposed in CRD [172], but has since been extended through a plethora of works by providing more structural information [7], better distance functions [122, 216, 29], or improved projectors [117, 121, 9]. Our later work exploring the training dynamics of distillation, presented in this thesis, provides a cheaper and more theoretically motivated methodology than those previously mentioned.

Ensemble Distillation Distilling from an ensemble has shown to be very effective [178] and its effectiveness can be explained through a multi-view perspective [5]. [142] proposed a multi-stage approach to leverage both labelled and unlabelled data using the ensemble prediction formed from multiple-views. Similarly, [171] used kmeans to divide the data and train experts on each cluster. These experts then formed an ensemble for which a student model can learn from. Unfortunately, in general, both training the multiple diverse teachers and/or adopting them in the distillation pipeline, is very expensive. Figure 1.4 provides a general overview of the three discussed domains in distillation. We note that there are many other specific applications of distillation, from zero-shot distillation through to data distillation [154]. We omit any discussion of these applications as they are much less related to the presented work in this thesis.

1.5 Thesis Outline

Chapter 2. Compressing Descriptor Models. We focus on the task of evaluating local descriptors. This task is an essential stage in most image matching pipelines, which has its applications in robotics, 3D reconstructions, and SLAM. This work was motivated by the observation of a distinct emergent structure in the convolutional weights for the state-of-the-art HardNet++ model. This structure could then be naturally exploited through a low-rank decomposition to provide minimal reconstruction error. We proposed an alternative to the standard convolution that enforced this decomposition and allowed us to re-train the HardNet model with far fewer parameters and a significantly lower computational overhead. This method was further extended to the SuperPoint model with similar results and has been published in ICASSP 2021 [119].

Chapter 3. Reconstructing Pruned Filters with Cheap Transformations. The work provides an efficient alternative to the standard convolutional layer. We proposed to express the convolutional filters as spatial transformations of a core set of basis filters. Unlike in chapter 2, this work was not motivated by an observation of any specific task. This work was in fact originally motivated by group convolutions [41], whereby equivariance to pre-defined group actions can be achieved through applying these transformations to the filters. Explicitly enforcing these properties was shown to improve the generalisation across a few small datasets. In contrast, we proposed to relax these constraints and use this methodology to improve the performance of smaller networks. This work is published in the ICCV 2023 workshop on resource efficient deep learning for computer vision [120].

Chapter 4. Cascaded Channel Pruning. Here we provide a natural extension of slimmable networks [199]. These networks provide a scheme for training a single model that can instantly switch to different widths, thus providing adaptive performance v.s. accuracy trade-offs. They proposed to jointly train all the sub-networks at given widths with independent batch normalisation layers. One significant limitation of this work was that these sub-networks were

restricted to a fixed width % for each layer. From the pruning literature it has been observed that the redundancy in weights is not uniform across all layers, and in fact a large proportion of this redundancy resides in the later layers. In light of this, we proposed to jointly train the sub-networks through a standard pruning methodology, whereby each network is trained with a unique hard pruning mask which is updated using a straight through estimator for its gradients. However, this hard mask was shown to provide poor gradient flow, and thus we proposed to replace the gradients with a surrogate from the lesser pruned sub-network. This led to a hierarchy of networks, each of which would provide explicit knowledge distillation through the surrogate mask gradients. Once all the sub-networks converged, the smallest network was then individually trained using the larger sub-networks as the teachers. This work was accepted to BMVC 2020 [118].

Chapter 5. Information Theoretic Representation Distillation. This chapter takes a slight departure from both pruning and the low-rank decomposition of weights. Here we instead look at knowledge distillation as a way for improving the performance of a small *student* model. We propose to approach the problem of distillation from an information theoretic perspective using a cheap surrogate estimator for entropy. In doing so we introduce two distinct loss terms which maximise the correlation and batch-wise similarity between the student and teacher features. These two losses were used to construct a cheap and interpretable method for distillation that led to improved performance across all the standard image classification benchmarks. Furthermore, to show the generality of this framework, we also applied it to both natural language understanding and training binary networks. Not only did our framework improve the performance over language specific distillation methods, but it took the first step in bridging the gap in performance between binary and full precision networks. This work has been published at BMVC 2022 [122].

Chapter 6. Understanding the Role of the Projector in Knowledge Distillation. This chapter follows on from the previous chapter by proposing a simple 3-step distillation recipe using just a projector, batch normalisation, and a log-sum metric. This simple and

cheap recipe leads to improvements across a wide range of large-scale distillation benchmarks. Most notable of which is with the data-efficient training of transformers, whereby a significant improvement in student performance is achieved. Not only do we provide the empirical results confirming the effectiveness of this approach, but we also provide a more concrete understanding into the inner mechanisms of distillation through understanding the evolvement of weights in the projection layer. Furthermore, we also show that effective cross-modal distillation can be attributed to the soft distillation of inductive biases. This work has been published at AAAI 2024 [121].

Chapter 7. Summary and Future Work. This final chapter provides a more exhaustive overview of the contributions and ongoing work following on from this thesis. We also discuss limitations and the broader impact in the field of deep learning. In summary our contributions and key take-aways from this thesis are given as follows:

1. Task-aware architecture designs can be very effective in providing strong performance v.s. accuracy trade-offs.

2. Knowledge distillation (KD) is complimentary to many other efficient and scaleable vision approaches.

3. KD demands a more concrete and interpretable explanation for its success. Carefully exploring the training dynamics is critical for constructing effective distillation pipelines.

Chapter 2

Compressing Local Descriptor Models

Feature-based image matching has been significantly improved through the use of deep learning and new large datasets. However, there has been little work addressing the computational cost, model size, and matching accuracy tradeoffs for the state of the art models. In this chapter we consider these practical aspects and improve the state-of-the-art HardNet model through the use of depthwise separable layers and an efficient tensor decomposition. We propose the Convolution-Depthwise-Pointwise (CDP) layer, which partitions the weights into a low and full rank decomposition to exploit the naturally emergent structure in the convolutional weights. We can achieve an 8× reduction in the number of parameters on the HardNet model, 13× reduction in the computational complexity, while sacrificing less than 1% on the overall accuracy across the *HPatches* benchmarks. To further demonstrate the generalisation of this approach, we apply it to other state-of-the-art descriptor models, where we are able to a significant performance improvement.

2.1 Introduction

Local features have a wide range of applications in robotics, tracking, and 3D reconstruction, where the algorithms are often required to operate in real time on resource constrained devices. However, this is generally not possible for most CNN based models due to the memory and

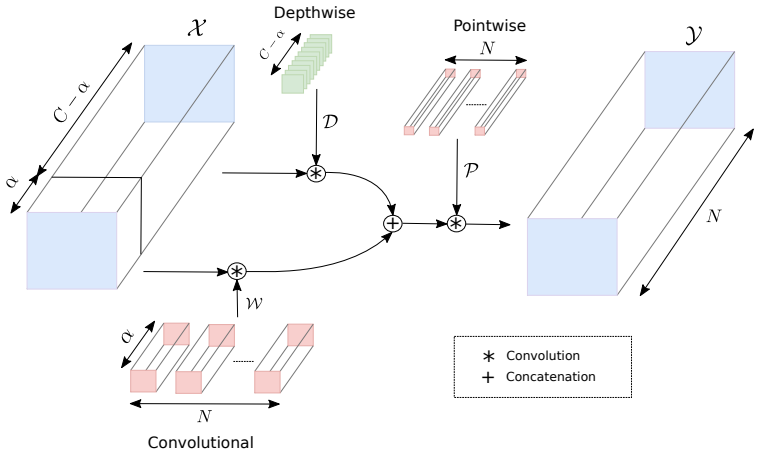


Figure 2.1: The proposed Convolutional-Depthwise-Pointwise (CDP) layer partitions the input tensor across the depth. Most of the computational resources are then reserved for only a subset of the input features, while the rest use an efficient depth-wise convolution. The resulting features are then concatenated and aggregated using a pointwise convolution.

computational cost far exceeding the resources. Feature extraction and matching is a critical component in any virtual/augmented reality pipeline.

Although the computational cost of handcrafted descriptors has been extensively researched [4, 17, 151, 24], there have been few methods for improving the efficiency of deep-learning based descriptors. In contrast, CNN models for image classification or object detection have been successfully compressed and deployed on mobile platforms through MobileNet [77] or ShuffleNet [209]. This is not possible on the large models such as ResNet[69], VGG [163] or GoogLeNet [166], which are commonly used as backbones for other tasks.

The number of parameters and the computational cost can act as a reasonable set of indirect metrics for the practical performance on-device, such as the inference latency. Our proposed method uses standard dense primitives, which have been efficiently implemented in most GPU-accelerated libraries, such as CuDNN.

We explore the use of popular low-rank decomposition methods [179, 75] on two state-of-the-art descriptor models, namely HardNet [126] and SuperPoint [45]. We provide an extensive evaluation of these descriptors with efficient operations and provide a practical scheme for combining them. Unfortunately, both depthwise-separable convolutions [160] and Tucker decomposition [179] sacrifice the top-end accuracy of the models. To address this issue we propose a new layer, convolution-depthwise-pointwise (CDP), which partitions the input features to utilise both the

standard convolution and the efficient depthwise convolution. The output features are then concatenated and aggregated using a pointwise convolution to maintain the original output dimensions (see figure 2.1). Using this proposed decomposition we can significantly compress the models with minimal degradation on the task accuracy.

2.2 Related Work

The related work is divided into the recent advances of descriptor models, followed by the successful methods for compressing convolutional neural networks.

Descriptors There has been a lot of research on developing handcrafted descriptors that trade-off robustness for computational efficiency [151, 17, 24]. However, machine learning approaches have been able to achieve significant accuracy improvements. A number of recently proposed top performing descriptors have used the L2Net [174] architecture with different training methodologies such as HardNet [126], GeoDesc [113], SOSNet [175], LF-Net [134], etc. Despite the improved results of these descriptor models, they all leverage a large CNN backbone, which makes their deployment on mobile devices very difficult. To this end, we propose a drop-in replacement for the standard convolutional layer that proves to be very effective for models trained on image matching related tasks. To verify this claim, we consider the state-of-the-art HardNet [126] and SuperPoint [45] models, whereby we are able to achieve significant model compression with minimal degradation in accuracy.

CNN compression This section presents several methods to improve the efficiency of the convolutional layers that have been successfully utilised in the context of object recognition, but not yet applied to local descriptors.

Pruning is an active removal of individual weights, kernels, or even entire layers from a network based on a saliency measure or a regularization term. Optimal Brain Damage [92] proposed to evaluate the saliency of individual weight entries using an approximation of the Hessian. This idea was further developed in Optimal Brain Surgery [64] through iteratively computing

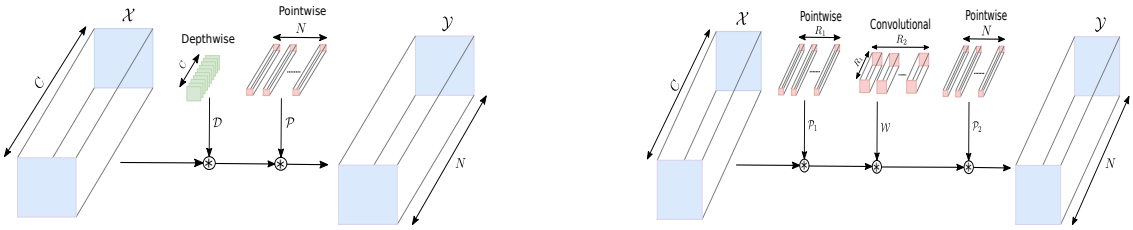


Figure 2.2: Depthwise separable convolution (left) and the Tucker decomposition applied to a convolutional layer (right) [88].

the Hessian to obtain a more exact approximation. [186, 94] consider the pruning of weights in a group-wise fashion, while NISP [200] prune kernels through a propagated importance score, and [212] use Bayesian inference with sparsity-inducing priors. Pruning based methods for compression are orthogonal to low-rank decomposition, which is what we explore in our approach.

Low-rank tensor decomposition is a method for approximating a higher order tensor using simple components, which are then combined using elementary operations. This can lead to a significant reduction in the number of parameters used to represent the original tensor, improve computational efficiency, and result in a more compact and interpretable model. Depthwise-separable convolutions (see figure 2.2 left) are the most common and have been used in all the MobileNet variants [77, 54]. Xception [38] has also successfully used these layers as replacements in the Inception modules [166], however, they did not explore partitioning the input tensor across the depth, which is the focus of our proposed CDP layer. Tucker [179] and its special case Canonical Polyadic (CP) [75] decomposition factor an N-dimensional tensor into lower dimensional components. The original convolution operation can then be replaced by a series of convolutions with smaller tensors [82, 88]. Tensor networks further provide a theoretical framework for such decomposition and have shown promising results [184].

2.3 Compressed Descriptor Model

In this section we introduce our new CDP layer and a scheme that combines the Tucker decomposition and depthwise-separable layers.

Convolution-Depthwise-Pointwise (CDP) Our proposed approach is motivated by the observation of two distinct partitions of the weights across the input channel depth. Figure 2.3 shows some of the convolutional weight slices for the pre-trained HardNet++ model, where a significantly higher variance in the weight entries is present across only a subset of the input channels. In fact, we observe a clear and consistent cut-off point (offset) across each layer. These low-variance columns (input channel slices) can be approximated using a low-rank decomposition with a lower reconstruction error. In light of this, we enforce a partition on the convolutional weights that matches this observation before training.

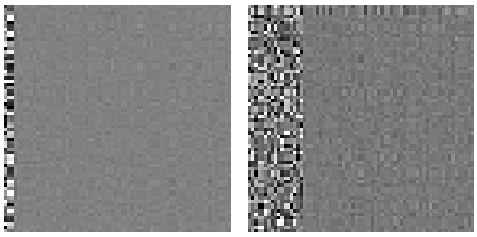


Figure 2.3: Example of the convolutional weight slices $\mathcal{W}[:,:,i,j]$ from the pre-trained HardNet++ model for layers 6 and 7, Where i, j is a given spatial coordinate in the receptive field. Note that the columns correspond to a given input channel index and the weights are scaled to be in the range $[0, 255]$.

Based on these observations, we propose an approach that combines a depthwise separable layer with a standard convolutional layer to provide a certain degree of full dense spatial and channel-wise connectivity for a subset of the input channels. This is shown in figure 2.1, where the first few channels are reserved for a standard convolution, while depthwise kernels are used for the rest. These output features maps are then concatenated and aggregated using a pointwise convolution. We argue that, although the CDP layer will have more parameters than the typical depthwise separable layers, the dense connectivity will ensure that the high-end accuracy is maintained. We define the number of input channels for the standard convolution to be the offset parameter α. This parameter provides a smooth transition between a normal convolution layer (albeit with a redundant pointwise convolution), where α is equal to the number of input channels, and a fully depthwise separable layer, when α is equal to zero.

The input feature maps can be represented as a 3-way tensor, $\mathcal{X} \in \mathbb{R}^{W \times H \times C}$, where C indicates the number of feature maps and W, H are the spatial dimensions. We use $\mathcal{X}_{i:j}$ to indicate feature maps indexed from i (inclusive) through to j (non-inclusive). This notation is used to define

the operation of the CDP layer proposed. Let $\alpha \in [0, C]$ indicate the depthwise offset for compressing the convolutional layer and $\mathcal{W}, \mathcal{D}, \mathcal{P}$ correspond to the convolution, depthwise, and pointwise weights respectively.

$$\mathcal{Z}_n = \begin{cases} \mathcal{X}_{0:\alpha} * \mathcal{W} & 0 \leq n < N \\ \mathcal{X}_{\alpha:C} * \mathcal{D} & N \leq n < N + C \end{cases} \tag{2.1}$$

$$\mathcal{Y} = \mathcal{Z} * \mathcal{P} \tag{2.2}$$

The total number of weights for stage (1) is given by $K^2 \cdot \alpha \cdot N + K^2 \cdot (C - \alpha)$, where K^2 is the receptive field size and N is the number of kernels used for the standard convolution block. The output from both these blocks are then concatenated along the depth-axis and followed by a pointwise convolution with O kernels. Both α and N can be adjusted to control the overall compression however, for simplicity, we use $N = O$ throughout. On this basis, compression and acceleration of the overall layer is achieved if $\alpha < C - \frac{N}{K^2-1}$ (see Supplementary).

Pointwise linear bottleneck We also explore an approach that attempts to combine both Tucker decomposition and the depthwise separable layers. This method was motivated by the fact that the pointwise kernels contribute far more significantly to the total computation and the number of parameters than the depthwise kernels. We choose to replace the pointwise convolution in the depthwise separable layer with a bottleneck, where the size of this bottleneck is determined by Tucker decomposition with Variational Bayesian Matrix Factorization (VBMF) [128]. The core tensor is not used since the pointwise kernel has unit spatial dimensions and the maximum rank is chosen to ensure restorability of the models accuracy. In this case, model compression and acceleration is only achieved if the depth of the intermediate feature map R is sufficiently small s.t. $CN > CR + RN$. This methodology differs from the Tucker decomposition [88] by the fact that the core tensor is not used. This proposed scheme is inspired by the linear bottlenecks used in MobileNetV2 [54], except that our spatial aggregation is not

performed in this lower-dimensional subspace and the low-rank approximation is pre-computed using VBMF of the pre-trained network weights. Hand-crafted linear decomposition schemes, such as SVD or PCA, could also be used, but may incur additional computational overheads.

2.4 Experimental results

The task performance of the descriptor models are evaluated using the HPatches benchmark [13]. This dataset is composed of local patches at varying levels of illumination and viewpoints changes from 116 different scenes. We focus on the matching and retrieval tasks as these were found to be more challenging and useful for practical applications [13] unlike the validation task. The HardNet model variants were all implemented in PyTorch [137] using the same training procedure[1] and the HardNet++ weights were trained on the *Liberty*, *Yosemite*, and *Notredame* datasets, while all the models proposed in this chapter are trained solely on the *Liberty* dataset from random initialisation.

We perform an evaluation of the standard depthwise separable layers and Tucker decomposition on the HardNet model along with a comparison to our proposed pointwise linear bottleneck scheme and the use of CDP layers.

Performance metrics For our evaluation we consider both the computational cost, measured in the number of floating-point operations (FLOPs), and the total number of parameters across all the convolutional layers. The cost of the element-wise non-linearity is negligible and the batch normalisation can be fused with the previous layer, thus they have been omitted from calculations. This computational complexity puts a theoretical bound on the minimum attainable latency with an efficient GPU implementation. The model size is described through a compression ratio i.e., the ratio of the total number of parameters in the original network against the compressed network.

[1] https://github.com/DagnyT/hardnet

Layer offsets									Homography Estimation			Detector metrics		Descriptor metrics		Performance	
#2	#3	#4	#5	#6	#7	#8	#9	#10	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	Rep.	MLE	NN mAP	M. Score	Compr.	Ops.
Original									.440	**.770**	**.830**	**.606**	**1.14**	.810	**.550**	1×	6.55
2	4	4	8	8	16	16	16	16	.445	.762	.824	.601	1.06	.835	.519	2.58×	2.10
5	5	5	5	5	5	5	5	5	**.453**	.752	.822	.596	1.06	.839	.517	2.97×	2.15
2	2	2	2	2	2	2	2	2	.407	.741	.826	.594	1.06	**.840**	.511	**3.21×**	**1.91**

Table 2.1: The CDP SuperPoint variants are able to achieve significant model compression with minimal degradation against the related detector and descriptor performance metrics. The computational complexity Ops. is measured in GFLOPs, and it is the contribution of the VGG16 backbone and the separate detector/descriptor heads.

Compressed HardNet performance We first report the baseline results for state of the art descriptors and then compare the proposed accelerations in terms of network compression ratio, mAP and computational cost (FLOPs). Table 2.2 (top) compares the number of parameters and HPatches results for three descriptors frequently used in the literature. L2Net [174] and TFeat-M* [14] are CNN architectures and SIFT [111] is a handcrafted descriptor with square root normalisation [6]. The SIFT consists of two convolutions to obtain image gradients which is equivalent to two 5x5 hardcoded kernels, thus 50 parameters

Alternative methods. To form a baseline for the evaluation of our proposed CDP layers, we consider the use of both depthwise separable layers, and an efficient Tucker decomposition implementation [88]. We further consider the use of a pointwise linear bottleneck to combine the benefits of both these methods. The results can be seen in table 2.2.

Model	# Parameters	Image Matching	Patch Retrieval
L2Net	1,334,560	**38.8**	**59.0**
SIFT	50	25.7	42.7
TFeat-M*	599,808	28.7	52.0

Model	Compression ratio	Image Matching	Patch Retrieval	Operations (MFLOPs)
HardNet	1×	**51.1**	**70.5**	35.7
DepthSep{7}	4.3×	50.1	69.4	34.6
DepthSep{6-7}	7.39×	47.1	67.5	26.3
DepthSep{5-7}	10.72×	46.4	67.0	25.4
DepthSep{2-7}	18.91×	44.5	66.0	**5.6**
Tucker{2-6}	1.21×	50.7	70.1	10.3
Tucker{7}	3.11×	29.7	50.1	34.8
Tucker{2-7}	6.81×	21.8	42.2	9.4
DepthSep{7} + Tucker{2-6}	**20.56×**	27.4	47.3	**8.5**
DepthSep{7} + TDW{5-6}	12.01×	47.0	67.2	25.0

Table 2.2: Comparison of using depthwise-separable layers, Tucker decomposition, and a proposed scheme for combining the two (TDW). The number(s) in the braces indicates the layers replaced. The image matching and patch retrieval accuracy is evaluated with mean Average Precision (mAP).

Layer offsets						Image Matching	Patch Retrieval	Compression ratio	Operations
#2	#3	#4	#5	#6	#7	mAP	mAP		MFLOPs
Original						**51.1**	**70.5**	1×	32.35
2	2	2	2	2	2	48.6	68.6	**9.50×**	**10.65**
5	5	5	5	5	5	50.3	70.4	7.66×	12.43
10	10	10	10	10	10	50.0	70.0	5.79×	15.43
15	15	15	15	15	15	50.4	70.3	4.65×	18.42
2	4	4	8	8	16	50.0	70.0	5.01×	11.76
4	8	8	16	16	32	50.1	70.1	3.21×	14.07
4	8	8	16	16	2	49.9	69.9	7.61×	13.83

Table 2.3: Applying different depthwise offsets for the HardNet model with CDP layers. The results are evaluated on the HPatches benchmark and averaged over the Easy, Hard, and Tough distributions.

CDP Layers. The results from table 2.3 demonstrate how enabling a subset of the channels to utilise full dense connectivity allows for the model to reach the high-end accuracy, while still achieving the favourable compression and acceleration from depthwise separable layers. The first group of rows considers a fixed offset for each layer, the second group takes into account the expansion of every odd layer by doubling the offset on each of these layers, and finally the bottom row uses the offsets as defined from the pre-trained HardNet++ weights (see figure 2.3). Specifically, for the last row, we define the offset as the channel index where the variance drops below the average across all channels. We observe that, as long as there is at least some dense channel connectivity (i.e. $\alpha \geq 5$), the architecture is able to achieve the top-end accuracy, which was not attainable with just depthwise separable layers, Tucker decomposition, or even the proposed combined approach. The CDP variants are able to achieve the best balance between the number of parameters and computation, while also demonstrating very little drop in accuracy ($< 1\%$). Not only is the top-end accuracy maintained, but the model is pushed into a suitable range for real-time performance on low-compute mobile phones (10-150MFLOPs).

SuperPoint We explore how the CDP layer can be applied to other models for a significant performance improvement. For this we consider the SuperPoint [45] model, which leverages a VGG [163] backbone to jointly predict the interest points and descriptors for matching.

To ensure consistency, we follow the original training methodology for all the evaluations, which includes the same homographic adaptions of training images. The model is jointly trained using the pseudo ground truth labels on the MS-COCO [102] dataset, while the evaluation

is performed using *HPatches* [13] benchmark. The results for the complete image matching pipeline can be seen in table 2.1 and show that the CDP layers are able to achieve a significant $\sim 3.2\times$ reduction in parameters and a $\sim 1.9\times$ reduction in FLOPs with minimal loss in the attainable matching score.

2.5 Conclusions

In this chapter we demonstrate the accuracy/performance trade-offs of applying various factorisation and networks compression methods on CNN models used for local feature extraction. We have proposed a novel Convolution-Depthwise-Pointwise (CDP) layer that consists of a partitioned low and full rank decomposition of the weights that matches the naturally emergent structure of the pre-trained weights. The allocated dense connectivity for a subset of the input features helps maintain the top-end descriptor accuracy. We further demonstrate the generalisability of this idea onto large architectures, namely the SuperPoint model. In both cases, we are able to compress the models significantly, with minimal to no accuracy degradation. This enables these models to be meet the resource constraints imposed by mobile devices for a wide host of applications, such as augmented/virtual reality.

2.6 Supplementary

In this section we provide derivations for the compression ratio, which relates to the reduction in parameters, and the computational cost, which can be used to infer the inference latency, of our proposed CDP layer.

2.6.1 Compression ratio

The CDP layer replaces the standard convolution operation with a smaller convolution that acts on a subset of the input channel dimensions. A depthwise convolution is then used for the remaining feature maps and a pointwise convolution is used for fusing features from both. With a depthwise offset given by α, the total number of weights is given by the sum of each of the contributing blocks respectively.

$$PARAMS = K^2 \times \alpha \times N + K^2 \times (C - \alpha) + 1 \times 1 \times (N + (C - \alpha)) \times N \tag{2.3}$$

Note that we make the assumptions that there are N convolutional kernels and N pointwise kernels, however, further compression could be achieved by reducing the number of standard convolutional kernels. The overall compression is achieved if α satisfies the following inequality:

$$K^2 \cdot \alpha \cdot N + K^2 \cdot (C - \alpha) + (N + (C - \alpha)) \cdot N < K^2 \cdot C \cdot N \tag{2.4}$$

Solving for α, this can then be reduced to:

$$\alpha < \frac{K^2 \cdot C \cdot N - K^2 \cdot C - N^2 - C \cdot N}{K^2 \cdot N - K^2 - N} \tag{2.5}$$

$$= \frac{C \cdot (K^2 \cdot N - K^2 - N) - N^2}{K^2 \cdot N - K^2 - N} \tag{2.6}$$

$$= C - \frac{N^2}{K^2 \cdot N - K^2 - N} \tag{2.7}$$

$$= C - \frac{N^2}{K^2 \cdot (N-1) - N} \tag{2.8}$$

By assuming $N \gg 1$, the bound on α is further restricted but significantly simplified.

$$\boxed{\alpha < C - \frac{N}{K^2 - 1}} \tag{2.9}$$

2.6.2 Computational cost

The cost of applying the element-wise ReLU operation to the intermediate feature map of both the depthwise and standard convolution is given by:

$$W \times H \times (N + (C - \alpha)) \tag{2.10}$$

Thus, the overall computational cost of the CDP layer is as follows:

$$FLOPS = WH \cdot (K^2 \alpha N + K^2(C - \alpha) + (N + (C - \alpha)) + (N + (C - \alpha))N) \tag{2.11}$$

The component parts are for the standard convolution, depthwise convolution, element-wise ReLU and pointwise convolution respectively. From this, the computational speedup can be derived:

$$\frac{C \cdot N}{\alpha N + C - \alpha} + \frac{K^2 \cdot C}{1 + N + C - \alpha} + \frac{K^2 \cdot C \cdot N}{C - \alpha} \tag{2.12}$$

Similarly for the compression, we can also derive an inequality for the requirement on α for achieving a reduction in the computational cost.

$$\alpha < \frac{K^2CN - K^2C - N - C - N^2 - NC}{K^2N - K^2 - 1 - N} \tag{2.13}$$

$$= C + \frac{N \cdot (N+1)}{K^2 \cdot (1-N) + N + 1} \tag{2.14}$$

Using the same assumption that $N \gg 1$ further simplifies this inequality:

$$\alpha < C + \frac{N^2}{N - K^2N} \tag{2.15}$$

$$= \boxed{C - \frac{N}{K^2 - 1}} \tag{2.16}$$

Which is the same condition imposed on the compression of parameters. This is a result of the imposed assumption that $N \gg 1$, which effectively ignores the cost of the ReLU operation. This is consistent in practise, where the cost of the ReLU operation is often not significant.

Chapter 3

Reconstructing Pruned Filters using Cheap Spatial Transformations

In this chapter we present another efficient alternative to the convolutional layer using cheap spatial transformations. This decomposition is not derived from any specific task properties but instead exploits an inherent spatial redundancy of the learned convolutional filters. This type of decomposition enables a much greater parameter efficiency, while maintaining the top-end accuracy of their dense counter-parts. Training these networks is modelled as a generalised pruning problem, whereby the pruned filters are replaced with cheap transformations from the set of non-pruned filters. We provide an efficient implementation of the proposed layer, followed by two natural extensions to avoid excessive feature compression and to improve the expressivity of the transformed features. We show that these networks can achieve comparable or improved performance to state-of-the-art pruning models across both the CIFAR-10 and ImageNet-1K datasets.

3.1 Introduction

Recent work on compressing CNNs [94, 219, 185, 99, 48, 141, 70, 72], have exploited the inherent weight redundancy using structured pruning. This approach provides a way to reduce

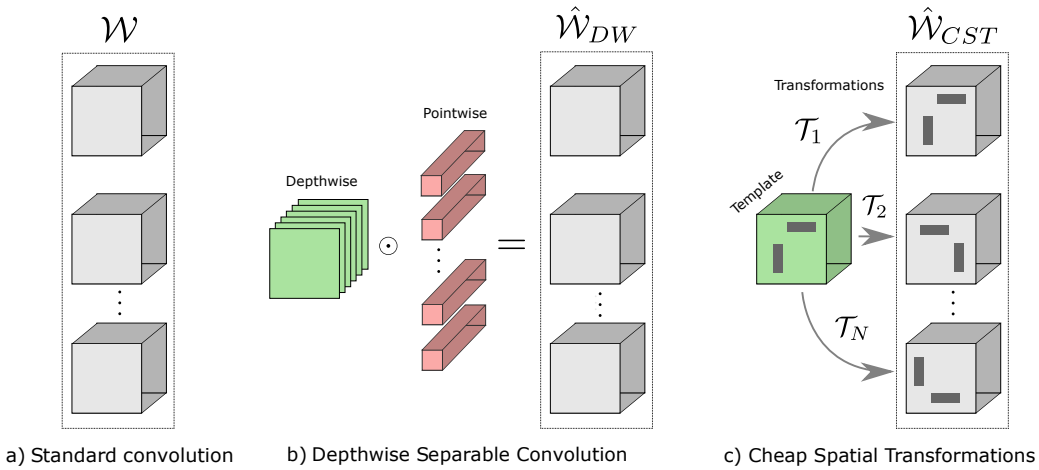


Figure 3.1: Constructing more convolutional filters using cheap spatial transformations. (a) Original convolutional layer. (b) Depth-wise separable layers, which fully decouple the spatial and depthwise aggregation of features. (c) Proposed layer expressed new filters as spatial transformations of a smaller set of templates.

the size of a network without relying on sparse software libraries or hardware accelerators. Most of these methods involve ranking the importance of filters and then removing those that fall below a specific threshold. However, it is important to note that when the pruning rates are high, some of these pruned filters can still contribute in retaining the top-end accuracy. To address this limitation, we propose a cheap decomposition of the convolutional layer where the pruned filters are reconstructed using cheap spatial transformations of the non-pruned filters, which we call templates. We propose an approach to transfer an existing CNN to this efficient architecture through a generalised pruning pipeline. This methodology can be considered a natural extension of pruning, but instead of zeroing out the pruned filters, we are replacing them with the cheap template transformation. This work can be related to group equivariant convolutional networks [41], which consider the hand-crafted construction of filters using a pre-defined group to learn equivariant features. In contrast to this work, we jointly learn both the transformations and templates with the alternative objective of training small and efficient CNNs. Our contributions can be summarised as follows:

- We propose a novel approach to construct expressive convolutional filters from cheap spatial transformations using a set of filter templates.
- We model the training as a generalised pruning problem with a simple magnitude based saliency measure.

- We introduce a grouped extension to mitigates excessive feature compression.
- Our results show competitive performance over state-of-the-art pruning methods on both the CIFAR-10 and ImageNet-1K datasets.

3.1.1 Related work

Pruning explicitly exploits the inherent parameter redundancy by removing individual weight entries or entire filters that have the least contribution to the performance on a given task. This was first introduced in [92, 64] using the Hessian of the loss to derive a saliency measure for the individual weights. SNIP [93] proposed to prune weights using the connection sensitivity between individual neurons. Subsequent work propose a sparse neuron skip layer [165] to achieve fast training convergence and a high connectivity between layers. Cheap heuristic measures have also been used, such as the magnitude [63, 94], geometric median [70], or average percentage of zeros [79]. Although some of these unstructured pruning methods are able to achieve significant model size compression, the theoretical reduction in floating-point operations (FLOPs) does not translate to the same practical improvements without the use of dedicated sparse hardware and software libraries. This has led to the more widespread adoption of structured pruning approaches, which focus on removing entire filters. [219] introduced additional loss terms to select the channels with the highest discriminative power, while [200] proposed to prune in accordance with a neural importance score. DMCP [61] models the pruning operation as a differentiable markov chain, where compression is achieved through a sparsity inducing prior. Similarly, [11, 212] model pruning in the probabilistic setting using both hierarchical and sparsity inducing priors. Unlike these prior works, we propose to reconstruct the pruned filters using cheap transformations. These reconstructed filters are shown to be expressive and learn diverse features, thus mitigating the need for any sophisticated pruning strategies.

Low-rank decomposition is concerned with compactly representing a high-dimensional tensor, such as the convolutional weights, as linear compositions of much smaller, lower-dimensional, tensors, called factors. Any linear operations that are then parameterised by these weights can be expressed using cheaper operations with these factors, which can lead to a reduction in the

computational complexity. Depthwise separable convolutions split the standard convolution into two stages, the first extracts the local spatial features in the input, while the second aggregates these features across channels. They were originally proposed in Xception [38] but have since been adopted in the design of a range of efficient models [28, 54, 76, 168]. This has led to the development of optimized GPU kernels that bridge the gap between the theoretical FLOP improvements and the practical on-device latency. Both CP-decomposition [75] and Tucker decomposition [179] have also been used to construct or compress pre-trained models [82, 88, 119]. Another line of work has explored the use of tensor networks as a mathematical framework for generalising tensor decomposition in the context of deep learning [65, 184]. Ghost modules [62] use depthwise convolutions to construct more features, leading to improved capacity at a much smaller overhead. Our proposed layer can be seen as an alternative parameterisation of the convolutional weights which can be naturally pruned using a generalised pruning pipeline.

Knowledge distillation attempts to transfer the knowledge of a large pre-trained model (teacher) to a much smaller compressed model (student). This was originally introduced in the context of image classification [73], whereby the soft predictions of the teacher can act as pseudo ground truth labels for the student. This methodology enables the student model to more easily learn the correlations between classes which are not available through the one-hot encoded ground truth labels. Hinted losses further provide knowledge distillation for the intermediate representation [149] and can be modelled as reconstruction L2 loss terms in the same space or in a projected feature space [196, 139, 123, 37, 122]. Weight sharing and jointly training models at different widths/pruning-rates has also been shown to provide implicit knowledge distillation to the smallest models [199, 198]. In general, our proposed method is orthogonal to knowledge distillation - its adoption can be employed in addition to further improve performance.

3.2 Method

In this section we propose a novel decomposition of the convolutional layer. We do this be expressing the convolutional filters as spatial transformations of a compact set of template filters. These templates are obtained through a well-established pruning procedure, ensuring

discriminative features. Subsequently, we present an algorithmically equivalent derivation of this layer that has much fewer floating-point operations (FLOPs). Moreover, we extend our approach naturally by introducing a group extension, which enhances the connectivity between layers. This extension enables an improved channel connectivity, fostering more robust and informative feature propagation.

3.2.1 Constructing diverse convolutional filters.

Let $\mathcal{W} = \{\mathcal{W}_n \in \mathbb{R}^{K\times K\times C}\}_{n=1}^{N}$ describe the set of filters for a given convolutional layer with an input depth C, output depth N, and a receptive field size of $K \times K$. Our method is based on an assumption that a large subset of these filters can be faithfully approximated as spatial transformations of a much smaller set of filters, which we call templates $\mathcal{B}$. For simplicity, and without loss in generality, consider the scalar transformations, which can be implemented as cheap element-wise products between the spatial entries of the templates. Thus, for a $K\times K\times C$ template, each transformation can be parameterised using $K \times K$ learnable weights. Consider the case with N templates and N output feature maps. The proposed decomposition is given as follows:

$$\mathcal{Y}_{h,w,n} = \sum_{k_w,k_h}^{K} \sum_{i}^{C} \mathcal{X}_{h',w',i} \cdot \mathcal{W}_{k_h,k_w,i,n} \tag{3.1}$$

$$\approx \sum_{k_w,k_h}^{K} \sum_{i}^{C} \mathcal{X}_{h',w',i} \cdot \mathcal{B}_{k_h,k_w,i,n} \cdot \mathcal{T}_{k_h,k_w,n} \tag{3.2}$$

$$h' = (h-1)s + k_h - p, \quad w' = (w-1)s + k_w - p$$

where s is the stride and p is zero-padding. The general formulation using affine transformations is illustrated in figure 3.1. Model compression can be achieved when the number of basis filters M is less than the number of output feature maps N. This is realised through pruning, which is discussed in second 3.2.2. In this case, the templates are then re-used to compute more filters using different transformations.

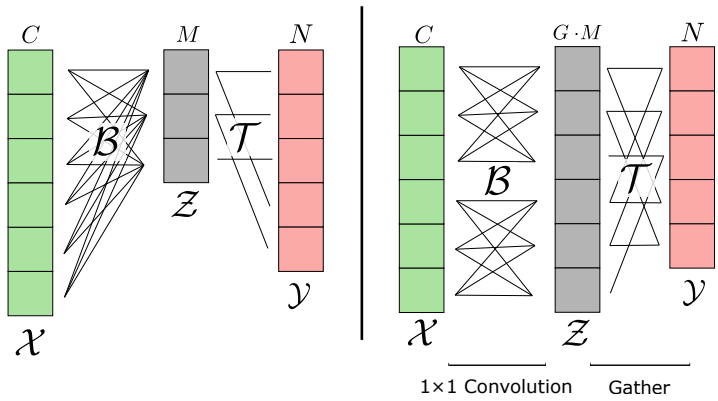


Figure 3.2: Too few templates can overly compress the input features. We propose to introduce a group parameter to naturally balance the expressiveness of the both the template and transformation stages.

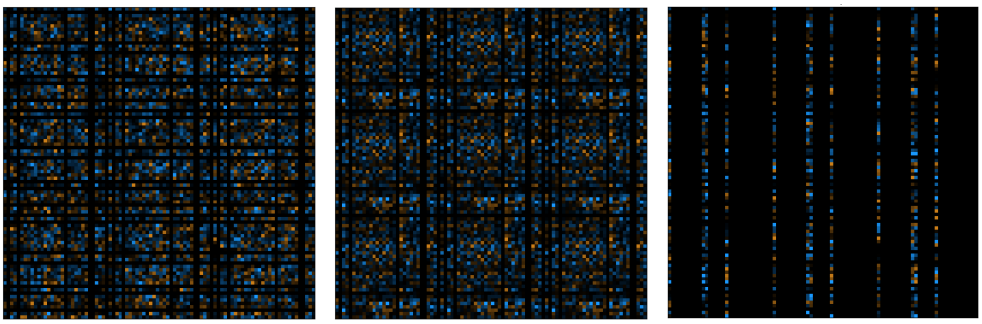


Figure 3.3: Visualising learned transformations for a given layer. (left) original learned filters. (middle) expressive filters using affine template transformations. (right) pruning filters. Both the vanilla pruned layer and the decomposed layers use a pruning rate of 0.7. Each column represents a filter for a given output channel and black pixels represent zero entries.

The choice of mapping from which template to which output feature map is not critical, as long as it is fixed after the pruning stage to enable fine-tuning. For our experiments, we set this mapping to be $i = j \ mod \ M$, where the ith output feature map is allocated the jth template, and where M is the total number of templates. This choice of mapping ensures that all templates are uniformly used, thus enabling a diverse set of transformed filters. The spatial transformations are then jointly learned alongside the templates.

3.2.2 Using pruning to select the filter templates.

The pruning literature has proposed increasingly sophisticated pruning heuristics and training pipelines. Examples of such including layer-wise pruning strategies [50] and gradient-based saliency measures [127], which incur additional hyperparameters and increased computational

costs. In favour of simplicity, and to demonstrate the generalisability of our decomposition, we propose to use a very simple magnitude-based criterion to rank the importance of filters for selecting the set of templates. We observe that this choice of saliency measure naturally leads to a uniform pruning strategy across all layers in the network (figure 3.6), which reduces excessive feature compression for any given layer (see section 3.2.4).

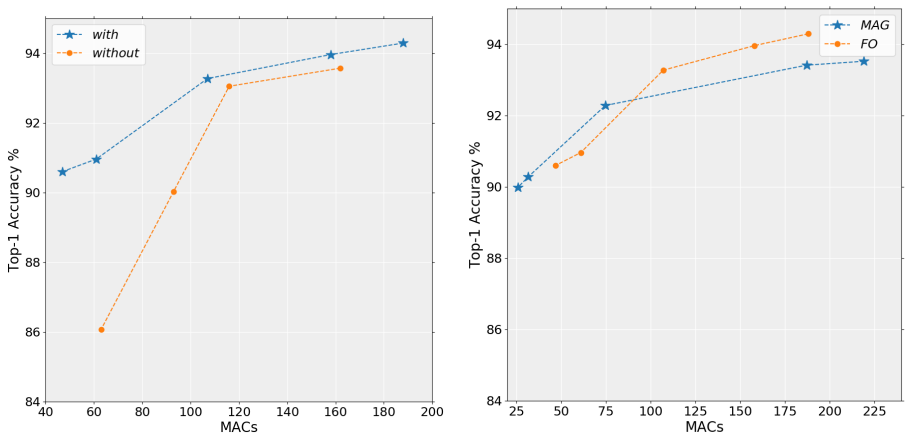


Figure 3.4: Comparison with a pruned baseline and the importance of which pruning heuristic. (left) Performance of a VGG16 network trained using affine (with) and scalar (without) spatial transformations with the same training methodology. (right) show the performance trade-off at different pruning rates using both a magnitude-based and a gradient-based saliency.

We provide an ablation on the importance on the choice of saliency measure in figure 3.4 (right). In this ablation we compare the performance using two different measures, namely magnitude based and gradient based [127]. Although in some cases the gradient based measure does lead to better performance, which is attributed to a more discriminative selection of templates, it does come at an increased computational overhead. In favour of simplicity, and to demonstrate the robustness to the choice of templates, we use a magnitude based measure throughout. In fact, for the ImageNet-1K experiments we extend this hypothesis and use a randomly initialised network to begin with, rather than from a pre-trained network - as is more commonly used in the pruning literature.

3.2.3 Performance and efficient implementations.

Computing the output features using the transformed filters and then using a standard convolution would not lead to any reduction in FLOPs. To address this, we propose to decompose the convolutional layer into two stages. The first computes the template *features* $\mathcal{Z}$ using the template filters $\mathcal{B}$, while the second projects these features to a different space using the spatial transformations $\mathcal{T}$. The output features are then the union of the original template features (identity transformations) and the transformed features. This two stage implementation is algorithmically equivalent to first constructing the filters and then performing a convolution, and its derivation is given as follows:

$$\mathcal{Y}_{h,w,n} \approx \sum_{k_w,k_h}^{K} \sum_{i}^{C} \mathcal{X}_{h',w',i} \cdot \mathcal{B}_{k_h,k_w,i,n} \cdot \mathcal{T}_{k_h,k_w,n} \tag{3.3}$$

$$= \sum_{k_w,k_h}^{K} \mathcal{T}_{k_h,k_w,n} \underbrace{\left(\sum_{i}^{C} \mathcal{X}_{h',w',i} \cdot \mathcal{B}_{k_h,k_w,i,n} \right)}_{\mathcal{Z}} \tag{3.4}$$

Computing $\mathcal{Z}$ can be achieved using a pointwise convolution, which translates to an optimised general matrix multiply primitive. The second stage, which consists of projecting $\mathcal{Z}$ to the output space, reduces to a series of gather operations and multiplications, which will implement the spatial transformations. Both of these operations can be trivially implemented in most deep learning frameworks.

3.2.4 Channel connectivity and feature compression.

By design, the proposed decomposition preserves the same number of input and output channels as the original convolution. This means that all the pruned filters are being reconstructed using some cheap and learned template transformation. The consequence of this design is that at high pruning rates there will be a significant bottleneck in the latent space $\mathcal{Z}$ (see figure 3.2). This bottleneck can result in significant feature compression that can degrade the downstream performance and discriminative power of representations. We could address this problem by

simply increasing the number of templates per layer, but this would incur a significant overhead in terms of both parameters and FLOPs. Instead, we propose to introduce a grouped extension that can naturally scale the dimensionality of the latent space $\mathcal{Z}$ with a minimal computational and parameter overhead. To do this we replace the *pointwise convolution* in the two stage processing with a *grouped pointwise convolution* [91], which has an efficient implementation in most deep learning frameworks. This transformation then translates to the sum of G transformations applied to feature maps from the G distinct groups. Doing so in this way enables cross-group information flow without the need for any channel shuffles [209]. Figure 3.2 graphically demonstrates this grouped extension. On the left is the original case, whereby $G = 1$. At this pruning rate, there is a very large compression of features. Increasing the groups to 2 (as shown on the right) provides a natural scheme for increasing the depth of $\mathcal{Z}$ without incurring any significant computational overhead. The results of the G different transformations across groups are then added to form each of the N output channels.

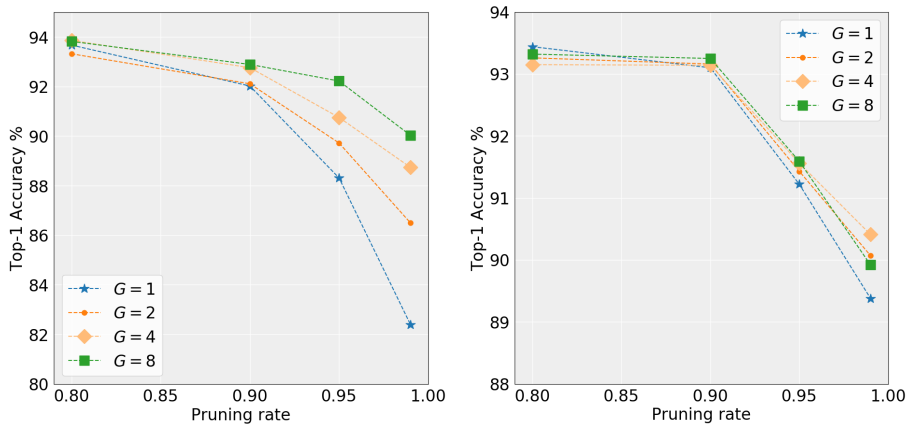


Figure 3.5: Importance of channel connectivity and the group extension. (left) Using more groups enables a smaller bottleneck ratio, which improves the top-end accuracy. (right) shows that increasing the minimum number of templates also reduces feature compression but at a much larger overall cost.

In the limiting case where $G = M$, the channel connectivity pattern is very similar to that of depthwise-separable convolutions since there will be a one-to-one mapping between channels in the first stage, while the second stage will be fully connected. However, there is still a significant distinction between the two - our proposed decomposition enables spatial aggregation of features in both stages.

Figure 3.2 demonstrates the importance of this grouped extension at high pruning rates. We find that although simply increasing the minimum number of templates in each layer does implicitly address this feature compression problem, it comes at a much larger overall cost. In general, G can be tuned depending on the target pruning rate.

3.2.5 Computational cost and parameter efficiency.

The standard convolutional layer has the computational cost of the order of $H \cdot W \cdot K^2 \cdot C \cdot N$, whereas the cost of the proposed decomposition is given by:

$$\begin{aligned} FLOPS = H \cdot W \cdot K^2 \cdot \frac{C}{\cancel{G}} \cdot M \cdot \cancel{G} + \\ H \cdot W \cdot K^2 \cdot G \cdot (N - M) \end{aligned} \tag{3.5}$$

Where M indicates the number of templates and G is the number of groups. The reduction in computation ($FLOPS \downarrow$) is subsequently given by:

$$FLOPS \downarrow = \frac{HWK^2 \cdot C \cdot M + HWK^2 \cdot G \cdot (N - M)}{HWK^2 \cdot C \cdot N} \tag{3.6}$$

$$= \frac{M}{N} + \frac{G}{C} - \frac{GM}{CN} \tag{3.7}$$

We prune the set of templates such that $M \ll N$ and we set $G \ll C$ to yield a reduction in FLOPs. We further improve the bottleneck problem by using $G \cdot M$ templates applied to C/G channels that are efficiently implemented with grouped convolutions. Finally, we ensure cross-group information flow by increasing the number of cheap spatial transformations that are then applied cross group.

From a similar view, we can also derive the reduction in parameters, where the number of parameters for a convolutional layer is given by:

$$PARAMS = K^2 \cdot C \cdot N \tag{3.8}$$

and our proposed decomposition has a parameter count given by:

$$PARAMS = K^2 \cdot \frac{C}{G} \cdot M + K^2 \cdot G \cdot (N - M) \tag{3.9}$$

Not the subtraction is because we use M identity transformation, while the rest of the output features are computed using cheap spatial transformations. Using both 3.8 and 3.9, we can derive the reduction in parameters ($PARAMS \downarrow$), which ends up being identical to equation 3.7.

$$PARAMS \downarrow = \frac{K^2 \cdot \frac{C}{G} \cdot M + K^2 \cdot G \cdot (N - M)}{K^2 \cdot C \cdot N} \tag{3.10}$$

$$= \frac{M}{GN} + \frac{G}{C} - \frac{GM}{CN} \tag{3.11}$$

When using more general and expressive spatial transformations, such as $GL(3)$ or $SO(3)$, the second stage can instead be implemented using a bilinear sampling of neighbouring spatial pixels in $\mathcal{Z}$. These transformations will be parameterised using a 2×3 matrix and result in the number of floating point operations being increased to 4 per spatial location. This increase in FLOPs and the number of parameters is often small, but it enables a significant increase in the expressiveness of transformations. However, in general, we observe that a learned scalar transformations can still yield a strong accuracy v.s. performance trade-off (see figure 3.2 left and table 3.3). We wish to highlight that in the case of these more general spatial transformations, the parameter reduction equation 3.11 and flop reduction equation 3.7 will differ.

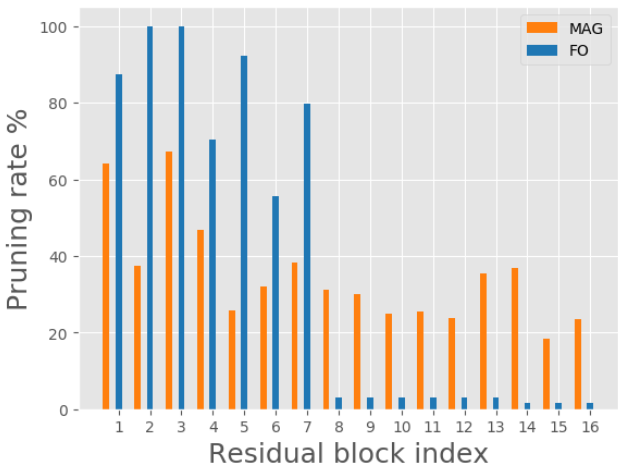


Figure 3.6: Comparing the pruning rates of each layer using different saliency measures. We see that a magnitude based (MAG) criterion exhibits a more uniform pruning rate than gradient based measures (FO).

3.3 Experiments

In this section we evaluate our approach on the CIFAR and ImageNet datasets. The models are compared through the number of parameters, the number of floating point operations, followed by the top-1 classification accuracy. All of these models are trained on a single NVIDIA RTX 2080Ti GPU using either stochastic gradient descent (CIFAR10) or AdamW (ImageNet-1K). We set the minimum number of templates for each layer to be 8 for CIFAR10 and 32 for ImageNet-1K. Finally, we use the number of groups in each layer to be 2 for all the main benchmark experiments

3.3.1 Experimental results on CIFAR-10

The CIFAR10 dataset [90] consist of 60K 32×32 RGB images across 10 classes and with a 5:1 training/testing split. The chosen VGG16 [163] architecture is modified for this dataset with batch normalisation layers after each convolution block and by reducing the number of classification layers to one. During training, we augment the datasets using random horizontal flips, random 32×32 crops, and random rotations. The baseline architectures are trained for 300 epochs with a step learning rate decay and we use a simple magnitude based criterion for ordering and selecting the most important filters to form the set of templates. This selection is

Model	Method	Baseline Acc. (%)	Acc. (%)	Acc. Drop (%)	FLOPs ↓ (%)	Parameters ↓ (%)
VGG16	Hinge [96]	93.59	94.02	-0.43	39.07	19.95
	Ours	**93.26**	**93.92**	**-0.66**	**45.56**	**56.77**
	NSPPR [218]	93.88	93.92	-0.04	54.00	-
	AOFP [48]	93.38	93.84	-0.46	60.17	-
	DLRFC [72]	93.25	93.93	-0.68	61.23	92.86
	DPFPS [152]	93.85	93.67	0.18	70.85	93.92
	Ours	**93.26**	**93.62**	**-0.36**	**61.38**	**81.15**
	ABC [98]	93.02	93.08	-0.06	73.68	88.68
	HRank [100]	93.96	91.23	2.73	76.50	92.00
	AOFP [48]	93.38	93.28	0.10	75.27	-
	Ours	**93.26**	**92.74**	**0.52**	**80.89**	**95.26**
ResNet56	NISP [200]	93.04	93.01	0.03	43.60	42.60
	Ours	**93.60**	**94.18**	**-0.58**	**35.91**	**51.64**
	FPGM [70]	93.59	93.49	0.10	53.00	-
	NSPPR [218]	93.83	93.84	-0.03	47.00	-
	ABC [98]	93.26	93.23	0.03	54.13	54.20
	SRR-GR [185]	93.38	93.75	-0.37	53.80	-
	Ours	**93.60**	**93.35**	**0.25**	**48.27**	**71.24**
	DPFPS [152]	93.81	93.20	0.61	52.86	46.84
	Ours	**93.60**	**92.81**	**0.79**	**56.36**	**80.05**

Table 3.1: Comparison to other pruning methods on CIFAR10. Each model is trained using a magnitude based measure for selecting the filter templates, number of groups = 2, and with a minimum of 8 template filters per layer to avoid catastrophic pruning.

in conjunction with a simple linear pruning schedule that spans the first 40 epochs of training. We highlight that this choice of pruning schedule is in contrast to most of the other pruning methods [219, 112], which can adopt much longer pruning stages and introduce additional layer-by-layer stopping conditions. The results are shown in table 3.1 and show comparable or improved performance to the much more sophisticated pruning strategies across a wide range of compression ratios.

3.3.2 Experimental results on ImageNet

For experiments on ImageNet-1K we use the ResNet-50 architecture and the same magnitude based saliency measure described in section 3.3.1. In general, magnitude pruning was empirically shown to provide a more uniform pruning rate across all of the residual blocks, which is important to avoid overly compressing intermediate features. We set the minimum number of templates for each layer to be 8 and the number of groups in each layer to be 2. The model is trained for a 300 epochs with a linear learning rate decay every 25 epochs. Finally, we use MixUp and CutMix augmentations with α set to 0.1 and 1.0 respectively.

Model	Method	Baseline Acc. (%)	Acc. (%)	FLOPs ↓ (%)	Parameters ↓ (%)
ResNet50	G-SD-B [106]	76.15	75.85	44	23
	MetaPruning [107]	76.60	75.40	50	-
	NSPPR [218]	76.15	75.63	54	-
	DPFPS [152]	76.15	75.55	46	-
	S-COP [169]	76.15	75.26	54	52
	LRF-60 [84]	76.15	75.71	56	53
	DLRFC [72]	76.13	75.84	54	40
	Ours	**76.20**	**75.59**	**47**	**40**

Table 3.2: Comparison to other pruning methods on ImageNet-1K. Our model is trained from random initialisation and with a simple magnitude based criterion. Using number of groups = 2 and with a minimum of 32 templates per layer.

The ImageNet results are shown in table 3.2 and show comparable performance to state-of-the-art pruning methods without the need for extensive pruning and fine-tuning pipelines. To further demonstrate the robustness of this decomposition to the choice of templates and the effectiveness of joint template/transformation training, we propose to begin training from a randomly initialised network. In doing so, we attain comparable performance other pruning methodologies, without the need for any sophisticated pruning pipeline and stopping conditions.

3.3.3 Ablation experiments

Group extension. To demonstrate the benefit of our proposed group extension, we train a VGG16 network at different pruning rates and with a varying number of groups. The results in figure 3.5 (left) show that at high pruning rates, whereby the layer will incur a large compression of features, increasing the number of groups will help. Although increasing the minimum number of templates per layer can also partially address this problem as shown in figure 3.5 (right), it would come with a much more significant computational overhead. In practice, we find that carefully selecting both the minimum number of templates and the number of groups can lead to the best performance trade-off.

Transformation family. We explore the importance of choosing a suitable parametric family of transformations for the template filters. To do this, we first consider simple scalar multiplications of the templates, and then we consider learnable rotations. Finally, we consider the more expressive affine transformations. The results are shown in table 3.3. We find that in-

Transformations	Top-1 Accuracy (%)	Pruning Rate
Scalar	90.62	0.9
SO(3)	92.32	0.9
GL(3)	**92.33**	**0.9**
Scalar	92.48	0.7
SO(3)	93.48	0.7
GL(3)	**93.57**	**0.7**

Table 3.3: Ablating the family of transformations. Increasing the expressivity of transformations has a small improvement in performance, suggesting that most of the network capacity is reserved for depthwise feature aggregation.

troducing more expressive transformations does improve the attainable performance, which is more significant at the higher pruning rates.

We explore the importance of selecting an appropriate parametric family of transformations for the template filters. To do this, we first consider a simple scalar multiplications applied to the templates. Subsequently, we extend our analysis to encompass learnable rotations, further expanding the range of potential transformations. Finally, to unlock the full expressive transformations, we consider the general linear group, which provide a richer and more versatile set of manipulations.

Visualising learned transformations. Figure 3.3 provides a comparison between the original filters, the reconstructed filters, and the pruned filters. We can discern that the the reconstructed filters are significantly distinct, thus enabling highly discriminative features for the downstream task. This result is in stark contrast with conventional pruning, which simply zeroes out these pruned filters. This visualisation highlights the significance of our novel approach which not only prunes but also actively reconstructs the filters, resulting in more informative representation of the data.

Efficient Implementation. To demonstrate that the theoretical reduction in FLOPs can translate to a real reduction in latency, we implement a simple CUDA kernel for the decomposed layer. The results are shown in figure 3.7, where we can see that at even moderate pruning rates there is a noticeable reduction in latency in comparison to the standard convolutional

layer. We can also see that a large proportion of the latency is being spent on computing the template features, while a much smaller proportion comes from the scalar transformation of these features, which is implemented through a parallelized gather operation.

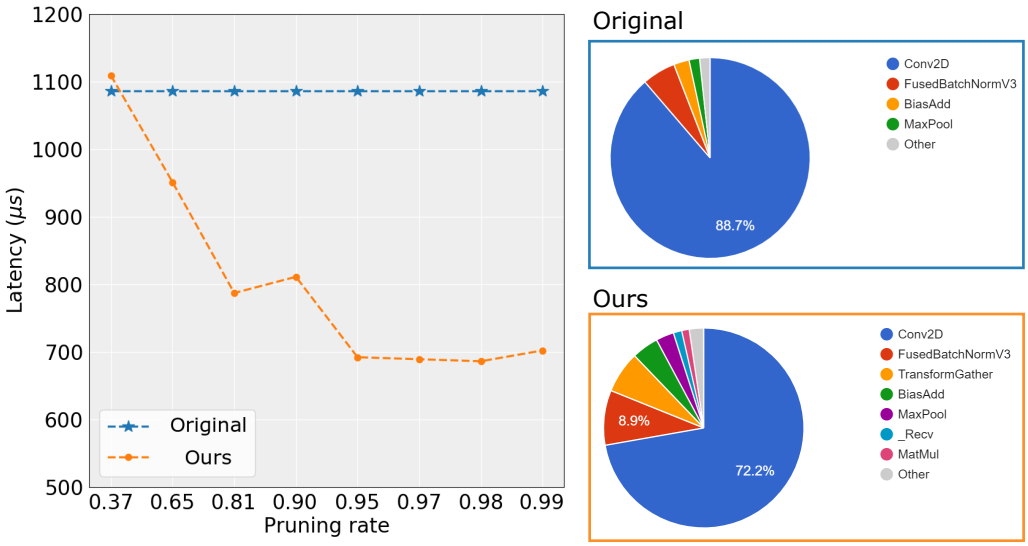


Figure 3.7: Latency of the proposed decomposition with an efficient implementation. (left) Compute time of an original VGG16 network with and without the decomposition at various pruning rates. (right) Cumulative contribution of each operation on the overall on-device compute time.

3.4 Conclusion and Future work

In this paper, we proposed the use of cheap transformations to reconstruct pruned filters. Instead of zeroing out the pruned filters, they are replaced with spatial transformations from the remaining set of non-pruned filters. These trained networks are able to achieve comparable or improved results on the image classification task across a range of datasets and architectures, despite using a simple magnitude based pruning criterion. We also introduce a grouped extension that can mitigate excessive feature compression at a minimal computational cost. Our approach applied to VGG16, ResNet34 and ResNet50 is able to significantly reduce the models size and computational cost while retaining the top recognition accuracy on CIFAR-10 and ImageNet-1K datasets.

Future research may explore potential applications in localization tasks that rely on equivariant features. Additionally, another promising direction is in data-efficient training. By incorporating hand-crafted transformations and leveraging prior knowledge of the data, it becomes possible to eliminate the necessity for the network to learn this information.

Chapter 4

Cascaded Channel Pruning using Hierarchical Self-Distillation

We propose an approach for filter-level pruning with hierarchical knowledge distillation based on the teacher, teaching-assistant, and student framework. Our method makes use of teaching assistants at intermediate pruning levels that share the same architecture and weights as the target student. We propose to prune each model independently using the gradient information from its corresponding teacher. By considering the relative sizes of each student-teacher pair, this formulation provides a natural trade-off between the capacity gap for knowledge distillation and the bias of the filter saliency updates. Our results show improvements in the attainable accuracy and model compression across the CIFAR10 and ImageNet classification tasks using the VGG16 and ResNet50 architectures. We provide an extensive evaluation that demonstrates the benefits of using a varying number of teaching assistant models at different sizes.

4.1 Introduction

It has been shown that CNNs exhibit significant redundancy, which has led to the development of various pruning techniques. Such methods attempt to identify and remove these redundant weights, which leads to improved memory and computational efficiency with minimal degrada-

tion in task accuracy. However, although pruning individual weights [92, 93] can achieve very high levels of sparsity i.e., parameter reduction, the irregular pruning is ill-suited for standard hardware accelerators. In contrast, channel pruning naturally addresses this issue by removing entire convolutional filters.

Most pruning pipelines use rule-based annealing schedules, with intermediate pruning and fine-tuning cycles. We instead jointly train both the set of pruning masks and weights in the same phase. To do this, we first propose a surrogate gradient for the importance scores of each filter, which are updated using standard back-propagation. The binary filter mask is then computed using a global threshold on these scores to achieve a given target compression. We propose a formulation for the updates of this importance score by extending the idea of "teaching-assistants" (TA) for knowledge distillation [125]. To enable the contribution of previously pruned filters to be re-considered into the student network, we use the gradients from a lesser-pruned TA, thus providing gradients from a model with a higher capacity. We describe the use of passing down surrogate gradients from the TAs to update the pruning masks as *cascaded pruning*, since the pruning is performed in a sequential fashion starting from the largest model in the hierarchy.

Each TA must also share the same set of weights as the student and have the same architecture to reduce the inherent bias in this surrogate gradient term. Using this formulation, we are able to build a hierarchy of student-teacher pairs from the same network (see figure 4.1) and by considering the relative sizes of each pair, provide a natural trade-off between the bias of the filter saliency gradients and the capacity gap for knowledge distillation. The additional benefit for sharing weights between the student model and all the TA's is a significant reduction in the memory overhead. Having disjoint TA's does not scale well and requires a set of pre-trained models, at appropriate relative sizes, to be readily available.

We extensively evaluate our approach for widely used state of the art networks and datasets. For the VGG16 [163] architecture trained on the CIFAR10 dataset, we are able to achieve a 1.9× reduction in parameters and a 2.3× reduction in FLOPs, while improving upon its top-1 % accuracy (see table 4.1). We also consider ResNet50 [69] on the ImageNet2012 classification

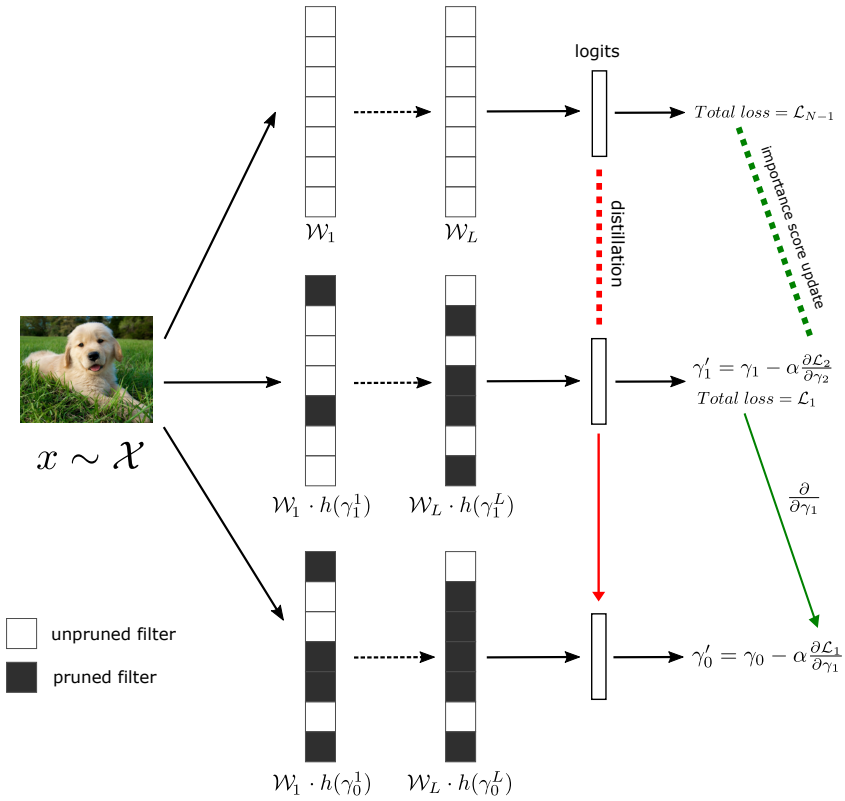


Figure 4.1: Proposed hierarchical self-distillation (HSD) strategy for channel pruning. Each of the models are jointly trained with shared convolutional weights but with independent binary masks, batch normalisation layers, and classification layers. The lesser constrained models provide knowledge distillation and importance score gradients down the hierarchy. The frozen teacher for model T_N has been omitted for clarity.

task, in which we are able to achieve a 3.6× reduction in parameters and a 3.7× reduction in FLOPs for a 2.5% drop in accuracy, which is very significant at this high level of compression (see table 4.2).

4.2 Related Work

Filter pruning methods attempt to remove both the feature maps/channels and corresponding filters that have the least positive contribution to the network accuracy. These techniques lead to a structured sparsity in the weights that can directly reduce the number of dense matrix multiplications needed and result in improved on-device performance with standard consumer hardware.

Pruning filters based on their absolute response magnitude was proposed in [94], while [71] performed the pruning with channel selection based on a LASSO-regression. In [112] the

pruning of a given layer is guided by subsequent layer statistics. Similarly, NISP [200] formulates the pruning problem as a binary integer program by which the error-propagation across layers is considered. Discrimination-aware losses were proposed by [219] for selecting channels based on their discriminative power. Probabilistic methods have also been explored for measuring the importance of filters through Bayesian inference and sparsity-inducing priors [215, 212]. Network pruning can also be modeled as a Neural Architecture Search (NAS) problem, whereby the depth of each layer is incorporated into the design space. AutoSlim [197] proposed the training of a single slimmable network that is iteratively slimmed and evaluated to ensure minimal accuracy drop, while MorphNet [58] optimizes the model using shrinking and cycle phases. The inefficient filters are then removed using sparsifying regularizers.

In a different approach [144] demonstrated the existence of sparse subnetworks within the large model, with randomly-initialised weights that can achieve high accuracy without any training. They identified this "super mask" using a straight-through estimator for the importance scores of each weight entry. We extend their method in evaluating this mask for the case of pruning entire filters.

Knowledge distillation was originally proposed by [73] to allow a smaller network to learn the correlations between classes from the output of a larger pre-trained teacher model. This work was extended in [149] by using intermediate representations as hints to the student. They approached this by minimising the $L2$ distance between the student and teacher's feature maps. It was shown in [125] that the student's performance can degrade if the gap between the student and the teacher is too large. They proposed to use intermediate teaching-assistants to distill knowledge between the teacher and the student. Each of these models used a different architecture and had an independent set of weights. Slimmable neural networks [199] defined a network that is executable at different widths through jointly training subsets of uniformly slimmed models. The smaller models benefited from the shared weights and the implicit knowledge distillation provided. We further demonstrate the effectiveness of this knowledge distillation between shared models, but instead independently prune each model using learned pruning masks.

Efficient architectures incorporate the efficiency and accuracy metrics into the initial architectural design choices. MobileNetV1 [77] proposed to use depthwise separable layers, which decomposes the 2D convolution operation into two subsequent operations for local spatial and channel aggregation. These layers have been efficiently integrated into all commonly used deep learning frameworks including Inceptions models [166], and all the MobileNet variants [77, 54, 76]. MobileNetV2 [54] proposed a linear bottleneck and an inverted residual connection to enforce feature re-use. ShuffleNet [209] built upon this idea by using group pointwise convolutions followed by a shuffle operation for enabling cross-group information flow. EfficientNets [167] use compound scaling for uniformly scaling a network's depth, resolution, and width to yield very efficient network architectures. All of these low-rank decomposition methods are complimentary to channel pruning and can be combined with our proposed method for further improvements.

4.3 Method

In this section, we first provide the formulation for a typical pruning problem and then describe our cascade approach for pruning entire filters through incorporating knowledge distillation, based on the student, teaching-assistant, and teacher paradigm.

Formulation The pruning objective can be described through the use of a binary mask $\mathcal{M} \in \{0,1\}^{|\mathcal{W}|}$ that is applied to the weights. Although this mask can span all the weights in the network, we restrict our attention to the convolutional layers as they contribute most significantly to the overall computational cost. The objective of pruning is then to learn a small subset of weights that can achieve comparable performance to the original model. These conditions can be described as follows:

$$\mathcal{L}(f(\mathcal{X}, \mathcal{W} \cdot \mathcal{M})) \approx \mathcal{L}(f(\mathcal{X}, \mathcal{W})), \quad \frac{\|\mathcal{M}\|_0}{|\mathcal{W}|} = p \tag{4.1}$$

Where $p \in [0,1]$ is a pre-defined pruning ratio that controls the trade-off between the number of used weights, the computational complexity, and the expressiveness of the model.

Finding the optimal mask The binary mask $\mathcal{M}$ disables the least "important" weights. To identify these weights we introduce an importance score $\gamma \in \mathbb{R}^{|\mathcal{W}|}$. This score can be evaluated using a set of static criteria [92, 70] or integrated directly into the learning procedure. The benefit of the latter approach is that the network can capture the complex mutual activations and dependencies of the weights. For example, some of the weights may only be important if another set of weights are enabled or vice-versa.

We model the importance score γ as a differentiable weight for which we can compute the binary mask $\mathcal{M}$. A pruning threshold is then defined as the smallest top-$p\%$ of the importance scores across all the convolutional layers. Corresponding weight entries are then conditionally masked if they are below this threshold. This operation of mapping the importance scores to the binary mask can be described through the function $h : \mathbb{R}^{|\mathcal{W}|} \rightarrow \{0, 1\}^{|\mathcal{W}|}$. Since this function is not differentiable with respect to γ, we adopt the straight through estimator [18, 83] of its gradients. We also use the derivation from [144] for the γ update rule, which we describe below. Consider the following masked convolutional layer:

$$\mathcal{Y}_{h,w,n} = \mathcal{X} * (\mathcal{W} \cdot \mathcal{M}) = \sum_{k_h}^{K_h} \sum_{k_w}^{K_w} \sum_{i}^{C} \mathcal{W}_{k_h,k_w,i,n} \cdot \mathcal{X}_{h',w',i} \cdot h(\gamma_{k_h,k_w,i,n}), \tag{4.2}$$

where $h' = h + K_h - \lceil \frac{K_h}{2} \rceil$ and $w' = w + K_w - \lceil \frac{K_w}{2} \rceil$ with K_h and K_w being the kernels height and width respectively. We also use $*$ to denote the discrete convolution operator and $\cdot$ is used for the element-wise product. The weights in this equation are represented as a 4-dimensional tensor $\mathcal{W} \in \mathbb{R}^{K \times K \times C \times N}$, where $K \times K$ is the filter size and C, N are the number of input and output channels respectively. The γ update can then be computed using the chain rule $\partial \mathcal{L} / \partial \gamma = \partial \mathcal{L} / \partial \mathcal{Y} \cdot \partial \mathcal{Y} / \partial \gamma$.

Due to the previously described practical performance constraint of hardware accelerators, we depart from this general formulation and instead consider pruning entire filters rather than individual weight entries. This results in the binary mask and importance scores for each layer being reduced to N-dimensional vectors, where N is the number of filters in the layer. The

original $\partial\mathcal{Y}/\partial\gamma$ term is then also reduced by summing over the spatial and input-channel axes as shown in equation 4.3. In light of this modification, we use k to refer to the ratio of pruned filters, as opposed to p which was used for the ratio of individually pruned weights.

$$\frac{\partial\mathcal{Y}}{\partial\gamma}_n = \sum_{w}^{W}\sum_{h}^{H}\sum_{k_h,k_w}^{K}\sum_{i}^{C}\mathcal{W}_{k_h,k_w,i,n}\cdot\mathcal{X}_{h',w',i} = \sum_{w}^{W}\sum_{h}^{H}\mathcal{W}*\mathcal{X}, \tag{4.3}$$

with $K = K_w = K_h$ to simplify notation but without any loss in generality. The final update[1] for γ is given as follows:

$$\gamma' = \gamma - \alpha\sum_{w=1}^{W}\sum_{h=1}^{H}\frac{\partial\mathcal{L}}{\partial\mathcal{Y}}\cdot(\mathcal{X}*\mathcal{W}) \tag{4.4}$$

Each filter is assigned an importance score that is related to their weighted contribution in decreasing the task loss $\mathcal{L}$. We expect that incorporating the practical performance metrics into this loss could improve the results, however, we show that using the task loss only is sufficient in providing an excellent accuracy vs model size reduction trade-off.

Shared teaching assistants An effective use of knowledge distillation requires a set of models with different capacities, strong task accuracy, and with high levels of diversity between them. Each model can then provide supervision to the smaller models through knowledge distillation. In this section, we describe a method for generating this diverse set from the same pre-trained model.

We can uniquely define a model from the same set of weights and architecture through the use of a binary mask $\mathcal{M}$ and its filter pruning ratio k. This allows us to define a set of N models $\{T_i \in (\mathcal{M}_i, k_i) \mid 0 \leq i < N\}$ where $k_i > k_{i+1}$ and $k_{N-1} = 0$. We expect that the model T_{i+1} is able to distill knowledge to T_i since this model has a larger expressive power. We define the T_0 model to be the student and $\{T_i \mid 1 \leq i < N-1\}$ to be the set of teaching-assistants (TAs).

[1]Tensorflow [1] internally computes all of the gradients and implements these update rules.

Model T_{i+1} acts as the teacher for the more constrained model T_i, while the teacher for model T_{N-1} is derived from the original pre-trained model with frozen weights.

Each of these models has an independent set of importance scores for each filter, batch normalisation statistics, and classification layers. Independent batch normalisation layers were originally proposed in Yu *et al.* [199] for networks that are executable at different widths, while the independent classifications layers are needed to enable sufficient diversity between the models. Figure 4.1 shows this proposed cascaded pruning method using a hierarchy of shared TA models.

Sharing the convolutional weights between the teaching-assistants and the student significantly reduces the training memory overhead while providing implicit knowledge distillation [199]. We further conjecture that the set of important filters for model T_{i+1} should also be important for model T_i if $k_i \approx k_{i+1}$. Thus, we propose that for each student-teacher pair, we can use the importance score gradients from the teacher to update the student. The proposed modification can be reflected in the γ update rule:

$$\gamma_i' = \gamma_i - \alpha \sum_{w=1}^{W} \sum_{h=1}^{H} \frac{\partial \mathcal{L}_{i+1}}{\partial \mathcal{Y}_{i+1}} \cdot (\mathcal{X}_{i+1} * \mathcal{W}) \tag{4.5}$$

Where the subscript i is used to indicate the ith model in the hierarchy. By updating γ_i using its corresponding teacher, we are making an assumption that $\partial \mathcal{L}_i / \partial \gamma_i \approx \partial \mathcal{L}_{i+1} / \partial \gamma_{i+1}$ or to the very least their sign is the same; which would indicate that the update is moving the importance score in the right direction for T_i. The benefit of this proposed formulation is twofold:

- Since both the models share the same weights, the teacher's gradient should be a reasonable estimator for the student.
- The teacher has a lower pruning ratio k and can thus can provide knowledge distillation to the student.

Consider the case where a filter has been pruned away by model T_i, but not by model T_{i+1}. In the original formulation, this will result in the corresponding channels being zeroed out

in $\mathcal{X}_i$ for the next layer and thus not considered in any of its γ updates. In contrast, using the proposed modified update rule from equation 4.5 will enable the student to consider the weighted contribution of this filter to the loss even if it is currently below the pruning threshold.

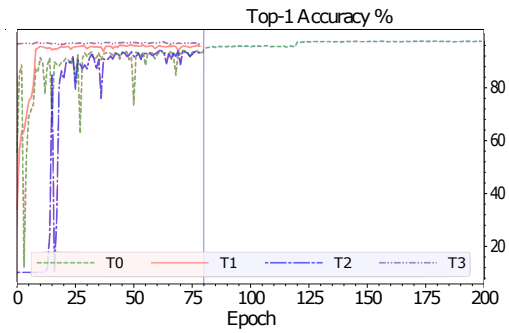
When the TA is much larger than the student, the bias of the gradient estimate will be too large and the updates will span a large set of importance scores, which makes the convergence of a suitable mask for the student very difficult. By ensuring each teacher is sufficiently large to provide useful knowledge distillation, while being sufficiently small such that this gradient update is stable, very effective training can emerge. This result is most noticeable at large pruning rates and evaluated further in section 4.5.

Knowledge distillation Supervised classification uses the cross-entropy loss $H(\cdot, \cdot)$ between the softmax logits of the network y_s and the one-hot encoded ground truth labels y_{GT}. On the other hand, knowledge distillation uses the KL divergence between the output logits of a teacher model y_t and the student y_s [73]. The student can then learn the correlations between classes from the teacher's predictions. A temperature term τ can also be used to soften the output probabilities to compensate for the different network capacities. The loss for the student is then formulated as the weighted combination of these two terms. Hinted losses [149] provide teacher-student supervision for the intermediate representations. For this, we consider the simple reconstruction error term [219] between feature maps.

Although sharing of convolutional weights does provide some level of implicit knowledge distillation between models, we find that providing additional explicit knowledge distillation and hints from T_N down to T_0 improves the performance of the student. We use the hint losses on the last few layers of the network, while for the knowledge distillation we use the KL divergence between the softened output probabilities of each teacher-student pair. The hyper-parameters λ_{KD} and λ_H are used to scale the KD and hint losses, respectively.

4.4 Experiments

In this section, we empirically validate the cascaded pruning approach on CIFAR10, CIFAR100 and ImageNet 2012 classification tasks, in which we consider the VGG16 [163]. MobileNetV1 [77] and ResNet50 [69] architectures respectively. The hint loss is placed in the last 3 layers of VGG16 network and the last 3 residual blocks of ResNet50. For the first convolutional layer in each network, we do not use any masking and keep an independent set of weights for each model.



Method	Params	FLOPs	Top-1 Accuracy (%)
Original	3.89M	495M	74.10%
HSD-0.2	2.95M	376M	↓ 0.29%
HSD-0.5	1.77M	156M	↓ 1.42%
HSD-0.7	0.38M	75M	↓ 3.69%

Figure 4.2: Left: Top-1 accuracy's of each model across the joint training step and the fine-tuning step. The training consists of one student T_0 and 3 TA's that are trained on the CIFAR10 dataset. Right: Accuracy and performance comparisons on the CIFAR100 dataset using the MobileNetV1 model.

The training process is decomposed into two steps; namely, the joint model training and the fine-tuning. The first step consists of jointly training all of the models on the same task, while updating their pruning masks using the proposed formulation in equation 4.5. The second step is where we only train the student model with the final frozen binary mask. In this latter stage, the KD and hint loss between the corresponding teacher and the student are still used, but the teacher is replaced with the next subsequent teaching-assistant in the hierarchy when the student starts to outperforms it on the target task. Figure 4.2 (left) shows these two training steps and we observe that this incremental fine-tuning step is vital to recover the student's accuracy after the previous joint training. The smaller models also experience much higher variance in their validation accuracy during the first stage as the enabled filters are constantly changing. After a suitable number of epochs, these student model naturally converge on a strong mask/weight initialisation for the subsequent fine-tuning step.

For calculating the number of parameters and the computational cost in terms of floating-point operations (FLOPs), we consider just the convolutional layers and the dense layers, without the biases. The batch normalisation layers are not considered as they are typically fused with the previous convolutional weights, while the residual addition and bias terms have a negligible contribution to the total FLOPs. Since we adopt filter pruning, these theoretical FLOP metrics should naturally translate to reduced inference time.

Implementation details All our experiments are implemented in Tensorflow [1] with an NVIDIA 2080Ti GPU. We use SGD as the optimizer with a weight decay of 0.0004 and momentum of 0.9. We use a cosine learning rate schedule with an initial learning rate of 0.008, 5 epochs per cycle, and an exponential decay. For a given model, we fix its filter pruning ratio k and use $\lambda_{KD} = 0.4$, $\lambda_H = 0.001$, and $\tau = 15.0$. Effective knowledge distillation is not only dependant on these loss weights, but also on the relative sizes of each student-teacher pair and the number of intermediate TA's, which is the focus of our attention.

Comparisons on CIFAR10 and CIFAR100 The CIFAR10 dataset [90] consist of 60K 32×32 RGB images across 10 classes and with a 5:1 training/testing split. The chosen VGG16 [163] architecture is modified for this dataset by adding independent batch normalisation layers [199] after each convolution block and by reducing the number of classification layers to two; of depth 512 and 10 respectively. The pre-processing step involves random horizontal flips and center crops of size 32×32. We jointly train the models for 80 epochs and then fine-tune the student for additional 80 epochs with a batch-size of 128. Each model uses a single student and 3 TAs with the filter pruning ratios evenly spaced between from k_0 (full pruning) down to 0 (no pruning). We observe that any reasonably uniform allocation between $k0$ and 1.0 is sufficient to balance the gradient bias and the capacity gaps.

Table 4.1 shows the accuracy and performance metrics for the cascaded pruning in comparison to other channel pruning methods with the same VGG16 architecture. Our results demonstrate the inherent redundancy in this choice of model for the CIFAR10 task as we are able to achieve significant compression while improving upon its top-1 % accuracy.

Method	Top-1 Baseline Accuracy (%)	Params	FLOPs	Top-1 Accuracy (%)
Original		14.98M	313M	93.26%
Variational pruning [212]	93.25%	3.92M	190M	↓ 0.07%
Geometric median [70]	93.53%	–	237M	↓ 0.34%
Try-and-learn [80]	92.77%	2.59M	140M	↓ 1.10%
Magnitude pruning [94]	93.25%	5.40M	206M	↓ 0.15%
Discrimination-aware [219]	93.99%	7.80M	157M	↓ 0.17%
Bayesian Pruning [215]	91.60%	**0.38M**	89M	↓ 0.60%
Hierarchical Self-Distillation	93.25%	7.76M	134M	↑ **0.28**%
student pruning rate	93.25%	2.50M	83M	↑ 0.07%
k = [0.3, 0.6, 0.8]	93.25%	0.97M	**52M**	↓ 0.28%

Table 4.1: Comparison to other filter-level pruning methods on the CIFAR10 benchmark and with the VGG16 architecture. For each model, the drop in accuracy is with reference to their own baseline.

The CIFAR100 dataset is very similar to CIFAR10, except that it instead contains 100 classes and with 600 images per class. For this evaluation we consider the efficient MobileNetV1 architecture, whereby the learned pruning masks are only applied on the 1×1 convolutional layers. We use the same number of TA's and corresponding filter pruning ratios as the CIFAR10 experiments, however, we only fine-tune for the student model for 40 epochs. The results are shown in table 4.2 (right) and demonstrate the validity of this proposed training methodology on an already parameter efficient architecture.

Method	Top-1 Baseline Accuracy (%)	Params	FLOPs	Top-1 Accuracy (%)	Top-5 Accuracy (%)
Original		25.5M	3.86B	75.03%	92.11%
Discrimination-aware [219]	76.01	12.38M	1.72B	↓ 1.06%	↓ 0.61%
Bayesian Pruning [215]	76.10	-	1.68B	↓ 3.10%	↓ 2.90%
NISP [200]	-	18.58M	2.81B	↓ 0.21%	-
Filter Sketch [99]	76.13	14.53M	2.23B	↓ 1.45%	↓ 0.69%
ThiNet [112]	72.88	12.28M	1.71B	↓ 1.87%	↓ 1.12%
		25.5M	4.1B	↑ **0.10**%	-
S-ResNet-50 [199]	76.10	14.7M	2.3B	↓ 1.20%	-
$[0.25, 0.5, 0.75, 1.0]\times$		6.9M	1.1B	↓ 4.00%	-
		2.0M	278M	↓ 11.10%	-
Hierarchical Self-Distillation	75.03	7.12M	1.04B	↓ 2.50%	↓ 1.29%

Table 4.2: Top-1 Accuracy and pruning ratios on the ImageNet2012 validation split using the ResNet50 model. The accuracy drops are reported in comparison to their corresponding baseline. The calculations for these baseline performance metrics are covered in the supplementary materials.

Comparisons on ImageNet We evaluate cascaded pruning on ResNet-50 [69] for the ImageNet 2012 classification task [153]. Unlike most other pruning strategies [112, 215], we also prune the projection layers and the last convolutional layer in each residual block. We train the models for 20 epochs and follow this by 20 epochs of student fine-tuning with a batch-size of 30. We use one student and 2 TAs with the filter pruning ratios of 0.3, 0.5, and 1.0, respectively. We follow the same pre-processing step as for the CIFAR10 experiments but with a central crop of size 224×224. The results can be seen in table 4.2 and demonstrate strong performance at high-levels of compression.

We observed that using the original SGD optimizer for the γ updates led to the majority of the pruning taking place in the last convolutional layer of each residual block, which severely limited the performance improvements. This outcome was a consequence of the fixed learning rates across all of the layers and their corresponding importance scores. We replaced SGD with an adaptive learning rate schedule, namely using RMSProp, whereby we were able to achieve a more uniform pruning strategy along with faster convergence. To further reduce training time, we also used an additional intermediate fine-tuning step that lasted 10 epochs and consisted of jointly training all the models with fixed pruning masks.

4.5 Ablation studies

In this section we evaluate the benefit of using multiple teaching-assistants. In the supplementary material we further provide a comparison against uniformly pruned baselines and evaluate the impact of using the additional explicit KD loss terms.

Increasing the number of teaching-assistants To evaluate the importance of using the shared teaching-assistant models, we consider training a student with a varying number of teaching-assistants. Figure 4.3 shows the task accuracy vs performance trade-offs for training a student with no TAs, with one TA, and with two TAs. In all of these cases, we use the same set of pruning ratios k and when no TA is used, the student only receives knowledge distillation from the fixed pre-trained model. We observe that the student model significantly benefits from having a TA to distill knowledge from and this is especially significant at the higher pruning ratios, whereby the difference in capacities between the student and the teacher is large. These results further demonstrate why a uniform allocation of pruning ratios between $k0$ and 1.0 is most suited for training these models; since it minimises the capacity gap between any of the student-teacher pairs.

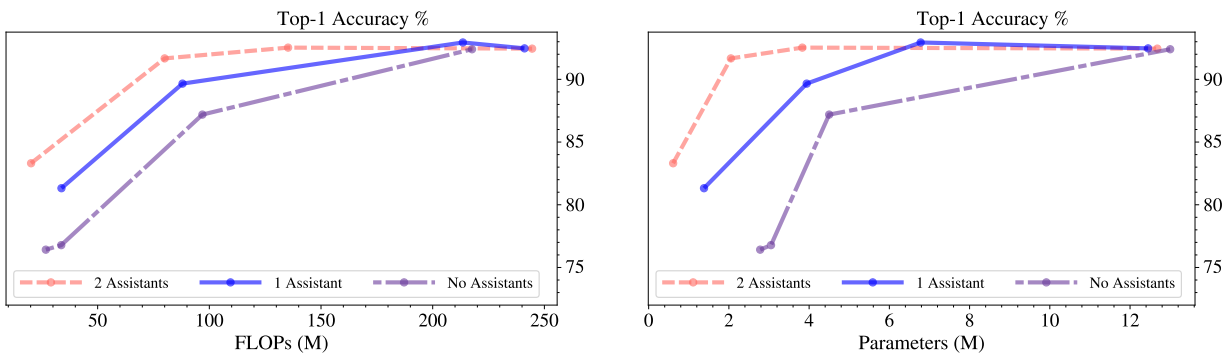


Figure 4.3: Evaluation of the computational complexity (left) and the number of parameters (right) for a student model with a varying number of teaching assistants. Each data point on the graph is ordered according to their pre-defined filter-pruning ratio using the modified VGG16 architecture on the CIFAR10 dataset.

4.6 Conclusion

We proposed cascaded-pruning, that is a channel pruning based method using a set of jointly trained and shared models. Each model provides both pruning guidance and knowledge distillation to its corresponding student. Besides the advantages in terms of scalability, cascaded pruning can achieve strong results without any hand-crafted annealing schedules, or iterative training and fine-tuning cycles. We demonstrate these results using a simple straight-through estimator for the pruning mask update, while providing a thorough set of evaluations of their performance with a varying number of teaching-assistants at different sizes. The results are especially significant at high-pruning rates, whereby the student benefits from these intermediate teaching assistants in the fine-tuning stage. We are able to achieve a $\sim 15\times$ compression and $\sim 6\times$ reduction in FLOPs with negligible accuracy degradation using VGG16 on the CIFAR10 dataset. We also consider the much larger ImageNet dataset with ResNet-50, in which comparable or better accuracy v.s. performance is demonstrated against other state-of-the-art filter pruning methods.

4.7 Supplementary

In this supplementary chapter, we provide more details on the underlying architectures used and the pruning rates attained for each layer. We also provide ablation experiments for the distillation losses.

ResNet50 architecture and performance metrics Table 4.3 shows the complexity and parameter break-down for each layer in the ResNet-50 model with an input size of 224×224.

Layer-wise pruning Figure 4.4 shows the percentage of pruned filters in each layer for a student model and two TAs. The student uniformly prunes the layers, while the larger TA models focus on the last layers. This is in contrast to how most other pruning methodologies work, which tend to result in significant pruning for the last few layers of the network. Table 4.4 shows how this distribution of pruning levels changes with the filter pruning ratio.

Block #	$w \times h$	#Filters	FLOPs	Params
	224×224	64	118M	0.01M
0	56×56	$[64, 64, 256]$	231M	0.07M
1	56×56	$[64, 64, 256]$	218M	0.07M
2	56×56	$[64, 64, 256]$	218M	0.07M
3	56×56	$[128, 128, 512]$	295M	0.38M
4	28×28	$[128, 128, 512]$	218M	0.28M
5	28×28	$[128, 128, 512]$	218M	0.28M
6	28×28	$[128, 128, 512]$	218M	0.28M
7	28×28	$[256, 256, 1024]$	295M	1.51M
8	14×14	$[256, 256, 1024]$	218M	1.11M
9	14×14	$[256, 256, 1024]$	218M	1.11M
10	14×14	$[256, 256, 1024]$	218M	1.11M
11	14×14	$[256, 256, 1024]$	218M	1.11M
12	14×14	$[256, 256, 1024]$	218M	1.11M
13	14×14	$[512, 512, 2048]$	295M	6.03M
14	7×7	$[512, 512, 2048]$	218M	4.46M
15	7×7	$[512, 512, 2048]$	218M	4.46M
	1×1	1000	2.05M	2.05M
Total:			3.85B	25.5M

Table 4.3: Performance statistics for the ResNet50 architecture on the ImageNet2012 dataset.

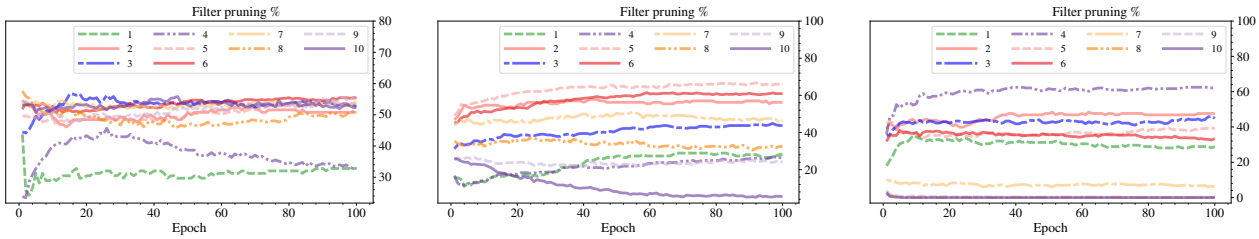


Figure 4.4: The layer-wise pruning for a student and two TA models trained using cascaded pruning. From left to right are models T_0, T_1, and T_2 respectively. Each TA uses the VGG16 architecture and is jointly trained on the CIFAR10 dataset.

Uniformly pruned baselines and KD loss terms The empirical results demonstrated by [108] showed that most channel pruning pipelines achieve comparable or worse performance to training the equivalent smaller model from scratch. Therefore, to confirm the performance benefits of our proposed method, we compare our results against individually training two smaller VGG16 variants from random initialisation. Specifically, we consider using both width scaling and shuffle units [209]. Width scaling reduces the depth of each layer by a given %, while a shuffle unit replaces the convolutional layers with group convolutions and channel shuffles. We use the same training methodology as the original baseline for all these models, which lasts 150 epochs with a cosine learning rate schedule. Liu *et al.* [108] considered two training schemes

Layer #	$w \times h$	#Filters	FLOPs	Params	Filter pruning			
					$k_0 = 0.1$	$k_0 = 0.5$	$k_0 = 0.6$	$k_0 = 0.8$
0	32×32	64	1.77M	1.73K	0%	0%	0%	0%
1	32×32	64	37.75M	36.86K	8.5%	20.1%	50.8%	89.4%
2	16×16	128	18.87M	73.73K	1.4%	11.7%	29.6%	83.2%
3	16×16	128	37.75M	0.15M	40.0%	45.4%	51.1%	67.7%
4	8×8	256	18.87M	0.29M	17.8%	33.3%	45.3%	63.1%
5	8×8	256	37.75M	0.59M	11.1%	36.0%	46.1%	77.2%
6	8×8	256	37.75M	0.59M	37.3%	59.0%	61.2%	86.6%
7	4×4	512	18.87M	1.18M	30.1%	60.8%	61.5%	73.7%
8	4×4	512	37.75M	2.36M	0%	29.7%	65.9%	67.3%
9	4×4	512	37.75M	2.36M	0%	17.0%	56.5%	92.3%
10	2×2	512	9.44M	2.36M	6.0%	69.0%	75.9%	88.9%
11	2×2	512	9.44M	2.36M	0%	85.5%	72.4%	81.7%
12	2×2	512	9.44M	2.36M	0%	63.5%	63.7%	84.1%
13	1×1	512	0.26M	0.26M	0%	31.8%	31.9%	42.1%
14	1×1	512	5.12K	5.12K	0%	0%	0%	0%

Table 4.4: Pruning % in each layer as a result of cascaded pruning on the CIFAR10 dataset and with the VGG16 architecture at varying filter-pruning ratios. The last two layers (13 & 14) are the two dense classification layers which are not masked.

for these uniformly pruned baseline: training for the same number of epochs as the baseline and training for the same computational budget. In both cases, the reported accuracy's were similar, and in our evaluation we found that further training any of these uniformly pruned baseline results had little effect on the accuracy.

To provide a thorough evaluation of cascaded pruning, we also consider the impact of using the explicit KD and hint loss terms between each student-teacher pair. We use only a single TA

Method	Params	FLOPs	Top-1 Accuracy (%)
Baseline	14.98M	313M	93.26%
Standard-0.75	8.48M	184M	↓ 5.69%
Standard-0.5	3.82M	89M	↓ 6.82%
Standard-0.25	**1.00M**	**28M**	↓ 11.08%
Group-2	7.62M	158M	↓ 6.46%
Group-4	3.95M	80M	↓ 7.69%
HSD-0.5 None	4.13M	102M	↓ 2.40%
HSD-0.5 w/ KD	3.62M	97M	↓ **0.79**%
HSD-0.5 w/ Hints	3.61M	96M	↓ 2.42%
HSD-0.5 w/ KD & Hints	3.69M	99M	↓ 1.27%

Table 4.5: Accuracy and performance metrics for two efficient VGG16 variants trained from random initialisation on the CIFAR10 dataset. Group-g indicates the use of group convolutions with g groups, while Standard-s uses s% width scaling for all the convolutional layers.

with a filter pruning ratio of 0.5 and set $\lambda_H = 0.001$ and $\lambda_{KD} = 0.4$ throughout. These complete sets of results can be seen in table 4.5. In the case where no KD or hinted losses are used, only implicit knowledge is distilled between the models, as attributed to the sharing of weights and joint training of all the models. The KD loss term between each teacher-student pair significantly increases the student's performance, while the hinted losses damage the student's performance. The hinted losses perform poorly in this framework since the enabled filters are constantly changing through the importance score updates. However, the student models trained using cascaded pruning still significantly outperform the equivalent smaller models when trained from scratch. These results demonstrate how the learnt mask structure is an integral part and contributing factor to the performance of these cascaded pruned networks.

Chapter 5

Information Theoretic Representation Distillation

Despite the empirical success of knowledge distillation, current state-of-the-art methods are computationally expensive to train, which makes them difficult to adopt in practice. To address this problem, we introduce two distinct complementary losses inspired by a cheap entropy-like estimator. These losses aim to maximise the correlation and mutual information between the student and teacher representations. Our method incurs significantly less training overheads than other approaches and achieves competitive performance to state-of-the-art on the knowledge distillation and cross-model transfer tasks. We further demonstrate the effectiveness of our method on a binary distillation task, whereby it leads to a new state-of-the-art for binary quantisation and approaches the performance of a full precision model. The code, evaluation protocols, and trained models are made publicly available on GitHub[1].

5.1 Introduction

Knowledge distillation proposes an alternative approach whereby a much larger pre-trained model can provide additional supervision for a smaller model during training. This paradigm

[1]https://github.com/roymiles/ITRD

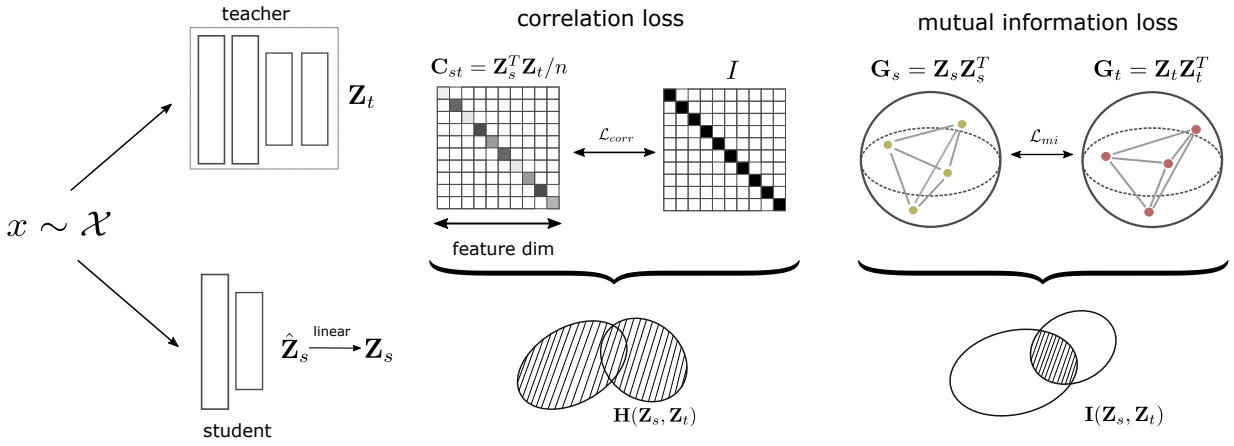


Figure 5.1: Information theoretic representation distillation (ITRD) involves two distinct losses, namely a correlation loss and a mutual information loss. The former loss maximises the correlation between the student and teacher, while the latter maximises a quantity resembling the mutual information that aims to transfer the intra-batch sample similarity.

removes the restriction of the two models to share the same underlying architecture, thus enabling hand-crafted designs of the target architecture to meet the imposed resource constraints. However, some of the recent state-of-the-art distillation methods, *e.g.* including the recent union of self-supervision and knowledge distillation [193, 189], have made it increasingly expensive to train these student models. To this end, we develop a distillation method with a low computational overhead.

Information theory provides a natural lens for quantifying the statistical relationship between these models, and so is a common framework for deriving distillation losses [29, 172]. Hence, we propose **I**nformation **T**heoretic **R**epresentation **D**istillation (ITRD) as a unified and computationally efficient framework that directly connects information theory with representation distillation. Specifically, this framework is inspired by the generalised Rényi's entropy and makes the training for specific applications more effective. Rényi's entropy is a generalisation of Shannon's entropy and has lead to improvements in other areas [124, 201, 158]. As figure 5.1 shows, we propose to model the distillation task with two distinct loss functions that correspond to maximising the correlation and mutual information between the student and teacher representations. The correlation loss aims to increase the similarity between teacher and student representations across the feature dimension. Conversely, the mutual information loss aims to match the intra-batch sample similarity between the teacher and the student. Our results show a strong accuracy v.s. training cost trade-off in comparison to state-of-the-art across two

standard benchmarks, CIFAR100 and ImageNet, for a range of architecture pairings where we achieve up to 24.4% relative improvement. Our loss directly addresses the training efficiency problem, which we believe will encourage its adoption amongst machine learning researchers and practitioners. We further demonstrate the effectiveness of this framework on representation transfer and binary network transfer, whereby we are able to improve upon the state-of-the-art for both.

5.2 Related Work

Knowledge Distillation (KD) attempts to transfer the knowledge from a large pre-trained model (teacher) to a much smaller compressed model (student). This was originally introduced in the context of image classification [73], whereby the soft predictions of the teacher can act as pseudo ground truth labels for the student. The soft predictions then provide the student with supervision on the correlations between classes which are not explicitly available from one-hot encoded ground truth labels. Spherical knowledge distillation [60] proposes to re-scale the logits before KD to address the capacity gap problem, while Prime-Aware Adaptive Distillation [210] introduces an adaptive sample weighting. Hinted losses provide a natural extension of KD using an L_2 distance between the student and teacher's intermediate representation [149]. Attention transfer [205] proposed to re-weight the spatial entries before the matching losses, while neuron selectivity transfer [81], similarity-preserving KD [180], and relational KD [135] attempt to transfer the structural similarity. Similarly, FSP matrices [196] attempt to capture the flow of information and Review KD [30] propose the use of attention based and hierarchical context modules. KD can also be modelled directly within a probabilistic framework [3, 136] through estimating and maximising the mutual information between the student and the teacher. ICKD [104] propose to transfer the correlation between channels of intermediate representations. A natural extension of supervised contrastive learning in the context of knowledge distillation was proposed in CRD [172]. WCoRD [29] also use a contrastive learning objective but through leveraging the dual and primal forms of the Wasserstein distance. CRCD [216] further develop this contrastive framework through the use of both feature and gradient infor-

mation. Unfortunately, all of these contrastive methods require a large set of negative samples, which are sampled from a memory bank. The use of these memory banks incurs additional memory and computational costs, which we avoid altogether.

Additional self-supervision tasks have shown strong performance when coupled with representation distillation. Both SSKD [189] and HSAKD [193] introduce auxiliary tasks for classifying the rotation of images. However, these approaches incur a high training cost due to the added self-supervision task, which augments the training batches and adds additional classifiers. Weight sharing through jointly training sub-networks has also been shown to provide implicit knowledge distillation [199, 118, 198] and promising results. In this work, we propose two distinct distillation losses applied to the features before the final fully-connected layer. Similarly to CRD [172], we posit that the logit representations lack relevant structural information that is necessary for effective distillation through the low dimensional embedding, while using the earlier intermediate representations can hinder the downstream task performance.

Information Theory (IT) provides a natural lens for interpreting and modelling the statistical relationships between intermediate representations of a neural network. This intersection of information theory and deep learning has subsequently led to a rigorous foundation in understanding the dynamics of training [2, 176], while also offering fruitful insights into other application domains, such as network pruning and knowledge distillation. In the context of representation distillation, most losses can be modelled as maximising some lower bound on the mutual information between the student and the teacher [172, 29]. In this work, we propose to forge an alternative connection between knowledge distillation and information theory using infinitely divisible kernels [20]. Specifically, we show that maximising both the correlation and mutual information yields two complimentary loss functions that can be related to these entropy-like quantities. We achieve this using a matrix-based function that closely resembles Rényi's α-entropy [155, 156, 187], which is in turn a natural extension of the well-known Shannon's entropy used in IT. More recently, this work has been applied in the context of representation learning [203] for parameterising the information bottleneck principle.

5.3 Preliminaries

Representation Distillation describes the family of distillation methods which use the representation space that is given as the input to the final fully connected layer of a model. The generalised loss used for representation distillation can be concisely expressed in the following form:

$$\mathcal{L} = L_{XE}(\mathbf{y}, \text{softmax}(\mathbf{y}_s)) + \beta \cdot d(\mathbf{z}_s, \mathbf{z}_t), \tag{5.1}$$

where $\mathbf{z}_s \in \mathbb{R}^{d_s}$ and $\mathbf{z}_t \in \mathbb{R}^{d_t}$ are the student and teacher representations, β is a loss weighting, and d is the distillation loss function. The cross entropy between labels $\mathbf{y}$ and student logits $\mathbf{y}_s$, i.e., L_{XE} above, can be defined as the sum of an entropy and KL divergence term. Furthermore, standard KD [74] uses an additional KL divergence as the distillation loss between the student and teacher logits, with a temperature term that can soften or sharpen the two distributions.

Following [172], the motivation for using the feature representation space, as opposed to logits or any of the intermediate feature maps is two-fold. Firstly, this space preserves the structural information about the input, which may be lost through the low-dimensional embedding of the final layer. Secondly, intermediate feature matching losses may negatively impact the students' downstream performance in the cross-architecture tasks due to differing inductive biases [172], while also incurring significant computational and memory overheads due to the high-dimensionality of these feature maps.

In our work, to maximize the information transfer, we propose to express the distillation loss $d(.,.)$ as the weighted sum of a correlation and mutual information term. Below we link these two terms to a general formulation of entropy [156].

Information Theory Rényi's α-entropy [148] provides a natural extension of Shannon's entropy, which has been successfully applied in the context of differential privacy [124], understanding autoencoders [201], and face recognition [158]. For a random variable X with

probability density function (PDF) $f(x)$ in a finite set $\mathcal{X}$, the α-entropy $\mathbf{H}_\alpha(X)$ is defined as:

$$\mathbf{H}_\alpha(f) = \frac{1}{1-\alpha} \log_2 \int_{\mathcal{X}} f^\alpha(x) dx, \tag{5.2}$$

where the limit as $\alpha \to 1$ is the well-known Shannon entropy. We restrict our attention to the discrete relaxation of this definition which more closely resembles the conventional definition of entropy in information theory. More concretely, for a discrete random variable X, Rényi's α-entropy $\mathbf{H}_\alpha(X)$ and Shannon's entropy $\mathbf{H}(X)$ are given as follows:

$$\mathbf{H}_\alpha(X) = \frac{1}{1-\alpha} \log_2 \sum_{x \in X} p(x)^\alpha \tag{5.3}$$

$$\mathbf{H}(X) = \sum_{x \in X} p(x) \log_2 p(x) \tag{5.4}$$

Since we wish to avoid the need for evaluating the underlying probability distributions, we propose to use a set of entropy-like quantities that closely resemble Renyi's entropy[156, 187] and are estimated directly from the data. These quantities depart from any conventional perspective in information theory, but are intimately connected through Rényi's axioms. They are based on the theory of infinitely divisible matrices and leverage the representational power of reproducing kernel Hilbert spaces (RKHS), which have been widely studied and adopted in classical machine learning. Since its fruition, this framework has been applied in understanding convolutional neural networks (CNNs) [202], whereby they verify the important data processing inequality in information theory and further demonstrate a redundancy-synergy trade-off in layer representations. We propose to apply these estimators in the context of representation distillation, although they can also be applied in the context of network pruning.

In the following section, we provide definitions of the entropy based quantities and their connections with positive semidefinite matrices. This idea then naturally leads to a multi-variate extension using Hadamard products, from which conditional and mutual information can be defined. For brevity, we omit the proofs and connections with Rényi's axioms, which can be found in [156] and [187].

Definition 1: Let $X = \{x^{(1)}, \dots x^{(n)}\}$ be a set of n data points of dimension d and $\kappa : X \times X \to \mathbb{R}$ be a real valued positive definite kernel. The Gram matrix $\mathbf{K}$ is obtained from evaluating κ on all pairs of examples, that is $K_{ij} = \kappa(x^i, x^j)$. The matrix-based analogue to Rényi's α-entropy for a normalized positive definite (NPD) matrix $\mathbf{A}$ of size $n \times n$, such that $tr(\mathbf{A}) = 1$, can be given by the following functional:

$$\begin{aligned} \mathbf{S}_\alpha(\mathbf{A}) &= \frac{1}{1-\alpha} \log_2(tr(\mathbf{A}^\alpha)) \\ &= \frac{1}{1-\alpha} \log_2 \left[\sum_{i=1}^{n} \lambda_i(\mathbf{A}^\alpha) \right] \end{aligned} \tag{5.5}$$

where $\mathbf{A}$ is the kernel matrix $\mathbf{K}$ normalised to have a trace of 1 and $\lambda_i(\mathbf{A})$ denotes its i-th eigenvalue. This estimator can be seen as a statistic on the space computed by the kernel κ, while also satisfying useful properties attributed to entropy. In practice, the choice of both κ and α can be governed by domain specific knowledge, which we exploit for the task of knowledge distillation. The *log* in these definitions, which is conventionally taken as base 2, can be interpreted as a data-dependant transformation, and its argument is called the *information potential* [155]. In the context of optimisation, the information potential and entropy definitions can be used interchangeably since they are related by a strictly monotonic function.

We are interested in the statistical relationship between two sets of variables, namely the student and teacher representations. To measure this relationship, we introduce the notion of joint entropy, which naturally arises using the product kernel.

Definition 2: Let X and Y be two sets of data points. After computing the corresponding Gram matrices $\mathbf{A}$ and $\mathbf{B}$, the joint entropy is then given by:

$$\mathbf{S}_\alpha(\mathbf{A}, \mathbf{B}) = \mathbf{S}_\alpha \left(\frac{\mathbf{A} \circ \mathbf{B}}{tr(\mathbf{A} \circ \mathbf{B})} \right) \tag{5.6}$$

where $\circ$ denotes the Hadamard product between two matrices. Using these two definitions, the notion of conditional entropy and mutual information can be derived. We focus on the mutual

information, which is given by:

$$\tilde{\mathbf{I}}_\alpha(\mathbf{A};\mathbf{B}) = \mathbf{S}_\alpha(\mathbf{A}) + \mathbf{S}_\alpha(\mathbf{B}) - \mathbf{S}_\alpha(\mathbf{A},\mathbf{B}), \tag{5.7}$$

where $\tilde{\mathbf{I}}$ is used to denote this notion of mutual information. Both equation 5.6 and 5.7 form a foundation for the correlation and mutual information losses respectively, which are proposed in the following section.

5.4 Information Theoretic Loss Functions

In this section we introduce two distillation losses that use two distinct and complimentary similarity measures between the student and teacher representations. The first loss uses a correlation measure which captures the similarity across the feature dimension, while the second loss is derived from a measure of mutual information and captures the similarity between examples within the mini-batch.

Maximising correlation This first loss attempts to correlate the student and teacher representations. The intuition is that if the two sets of representations are perfectly correlated then the student is at least as discriminative as the teacher. Let $\mathbf{Z}_s \in \mathbb{R}^{n\times d}$ and $\mathbf{Z}_t \in \mathbb{R}^{n\times d}$ [2] denote a batch of representations from the student and teacher respectively. These matrices are computed before the final fully-connected layer to preserve the structural information of the data, thus enabling a strong distillation signal for the student. We first normalise these representations to zero mean and unit variance across the batch dimension and then propose to construct a cross-correlation matrix, $\mathbf{C}_{st} = \mathbf{Z}_s^T\mathbf{Z}_t/n \in \mathbb{R}^{d\times d}$. Perfect correlation between the two sets of representations is achieved if all of the diagonal entries $v_i = (\mathbf{C}_{st})_{ii}$ are equal to one.

[2]For clarity, we omit a linear embedding layer used on the student representations to match its dimensionality with the teacher.

To formulate this as a minimization problem, we propose the following loss:

$$\mathcal{L}_{corr} = \log_2 \sum_{i=1}^{d} |v_i - 1|^{2\alpha} \tag{5.8}$$

This general objective is motivated by the recent work on Barlow Twins [207] for self-supervised learning, however, there are several distinct differences. Firstly, we drop the redundancy reduction term, which minimizes the off-diagonal entries in the cross correlation matrix, since we are not jointly learning both representations, *i.e.*, the teacher is fixed. In fact we observed that this objective significantly hurts the performance of the student. This performance degradation was similarly observed when decorrelating the off-diagonal entries in the self-correlation matrix $\mathbf{C}_{ss}$, and is likely a consequence of the limited model capacity. Secondly, we introduce an α parameter, which provides a natural generalisation to emphasise low or highly correlated features. Finally, the $\log_2$ transformation was empirically shown to improve the performance by reducing spurious variations within a batch. These modifications were not only empirically justified, but also provide a closer relationship with the matrix-based entropy function in equation 5.5, which is discussed next.

Relationship to joint entropy. The objective from equation 5.8 closely resembles maximising $\log_2 \sum_i v_i$. However, although these two objectives share the same optimum solution, the flexibility in tuning the sharpness of the loss with α proved very effective. If we consider the self correlation matrices $\mathbf{C}_{ss}$ and $\mathbf{C}_{tt}$, the diagonal entries in $(\mathbf{C}_{ss} \circ \mathbf{C}_{tt})^2$ [3] will be populated with products of pairs of cross-correlation terms between $\mathbf{Z}_s$ and $\mathbf{Z}_t$. This matrix construction can then be used in equation 5.6 to compute the joint α-order entropy between the student and the teacher, where $\alpha = 2$. In the case where the features are strictly independent, *i.e.*, $(\mathbf{C}_{ss})_{ij} = (\mathbf{C}_{tt})_{ij} = 0 \quad \forall\, i \neq j$, the objective of the proposed loss in equation 5.8 and maximising this joint entropy are equivalent. In the more general setting, the joint entropy formulation maximises the correlation between all pairs of exemplars, while our proposed loss only maximises the correlation along the leading diagonal of $\mathbf{C}_{st}$.

[3] This exponent denotes the square of a matrix, rather than an element-wise operation.

Correlation v.s. Gram matrices. The connection to joint entropy is limited in that the matrices used are correlation matrices as opposed to Gram matrices in equation 5.6. This is an important distinction since in this loss we wish to capture the similarity across the feature-dimension as opposed to the batch-dimension. However, despite this distinction, there is still an intimate connection between these two matrices. As discussed in the recent work on cross-covariance attention [51], the non-zero part of the eigenspectrum of the Gram and covariance matrices are equivalent. Since the entropy-like formulation described in equation 5.5 is a spectral function of $\mathbf{A}$, the two resulting quantities are in turn closely related.

Maximising batch-wise similarity The correlation loss aims to match the information present in each feature dimension between the teacher and student representations. The gram loss provides an additional complimentary objective whereby we transfer the intra-batch similarity (*i.e.*, the relationship between samples) from the teacher representations to the student representations. The natural choice for achieving this through the lens of information theory is to maximise the mutual information between the two representations. Maximising the mutual information has been successfully applied in past distillation methods [3], following the idea that a high mutual information indicates a high dependence between the two models and thus resulting in a strong student representation. Most others work relate their distillation losses to some lower bound on mutual information [172], however, using an alternative cheap entropy-like estimator, we propose to maximise this quantity directly.

$$\begin{aligned}\mathcal{L}_{gram} &= -\tilde{\mathbf{I}}_{\alpha}(\mathbf{G}_s;\mathbf{G}_t) \\ &= \mathbf{S}_{\alpha}(\mathbf{G}_s,\mathbf{G}_t) - \mathbf{S}_{\alpha}(\mathbf{G}_s) - \cancel{\mathbf{S}_{\alpha}(\mathbf{G}_t)}\end{aligned} \tag{5.9}$$

where $\mathbf{G}_s \in \mathbb{R}^{n\times n}$ and $\mathbf{G}_t \in \mathbb{R}^{n\times n}$ are the student and teacher Gram matrices (*i.e.*, $\mathbf{A}$ and $\mathbf{B}$ in equation 5.7). These matrices are constructed using a batch of normalised features $\mathbf{Z}_s$ and $\mathbf{Z}_t$ with a polynomial kernel of degree 1. The resulting matrix is subsequently normalised to have a trace of one. The teacher entropy term in this loss is omitted since the teacher weights are fixed during training. Substituting the marginal and joint entropy definitions from equations 5.5 and

5.6, with $\mathbf{G}_{st} = \mathbf{G}_s \circ \mathbf{G}_t$ (normalised to have a trace of one), leads to

$$\mathcal{L}_{gram} = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{n} \lambda_i \left(\mathbf{G}_{st}^{\alpha}\right) - \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{n} \lambda_i \left(\mathbf{G}_{s}^{\alpha}\right) \tag{5.10}$$

Where $\mathbf{G}_{st}$ is also normalised to have unit trace. Since computing the eigenvalues for lots of large matrices can be computationally expensive during training [86], we restrict our attention to $\alpha = 2$. This allows us to use the Frobenius norm as a proxy objective and one of which has a connection with the eigenspectrum - $\|\mathbf{A}\|_F^2 \geq \sum_{i=1}^{n} \lambda_i^2(\mathbf{A})$.

$$\mathcal{L}_{gram} = \log_2 \|\mathbf{G}_s\|_F^2 - \log_2 \|\mathbf{G}_{st}\|_F^2 \tag{5.11}$$

In practice, we observed that removing the *log* transformations improved the performance, thus resulting in a slight departure from the connection to the surrogate mutual information quantity defined in equation 5.7. Specifically, this loss instead minimises the distance between the marginal and joint *information potential*, rather than the mutual information.

Combining the two losses Both the proposed losses provide two different learning objectives. Maximising the correlation is applied across the feature dimension, thus ensuring that the students average representation across the batch is perfectly correlated with the teacher. On the other hand, maximising the mutual information encourages the same similarity between samples as from the teacher. These two losses effectively operate distinctly over the two dimensions of the representations, namely the *feature*-dim and the *batch*-dim. The final loss for which we aim to minimise is given as follows:

$$\mathcal{L}_{ITRD} = \mathcal{L}_{CE} + \beta_{corr}\mathcal{L}_{corr} + \beta_{gram}\mathcal{L}_{gram} \tag{5.12}$$

where $\mathcal{L}_{CE}$ is a standard cross-entropy loss, while β_{corr} and β_{gram} are hyperparameters to weight the losses. To demonstrate the simplicity of our proposed method, and similarly to past works [207], we provide the PyTorch-based pseudocode in algorithm 1.

Listing 5.1: PyTorch-style pseudocode for ITRD

```
# f_s: Student network
# f_t: Teacher network
# y: Ground-truth labels
# y_s, y_t: Student and teacher logits
# z_s, z_t: Student and teacher representations (n x d)
for x in loader:
  # Forward pass
  z_s, y_s = f_s(x)
  z_t, y_t = f_t(x)
  z_s = embed(z_s)
  # Cross entropy loss
  loss = cross_entropy(y_s, y)

  # Normalise representations
  z_s_norm = (z_s - z_s.mean(0)) / z_s.std(0)
  z_t_norm = (z_t - z_t.mean(0)) / z_t.std(0)
  # Compute cross-correlation vector
  v = einsum('bx,bx→x', z_s, z_t) / n
  # Compute correlation loss
  dist = torch.pow(v - torch.ones_like(v), 2)
  h_st = torch.log2(torch.pow(dist, alpha).sum())
  loss += h_st.mul(beta_corr)

  # Compute Gram matrices
  z_s_norm = normalize(z_s, p=2)
  z_t_norm = normalize(z_t, p=2)
  g_s = einsum('bx,dx→bd', z_s_norm, z_s_norm)
  g_t = einsum('bx,dx→bd', z_t_norm, z_t_norm)
  g_st = g_s * g_t
  # Normalize Gram matrices
  g_s = g_s / torch.trace(g_s)
  g_st = g_st / torch.trace(g_st)
  # Compute the gram loss
  p = g_s.pow(2) - g_st.pow(2)
  loss += p.sum().mul(beta_gram)

  # Optimisation step
  loss.backward()
  optimizer.step()
```

5.5 Experiments

We evaluate our proposed distillation across two standard benchmarks, namely the CIFAR-100 and ImageNet datasets. To further demonstrate the effectiveness of our loss, we perform additional experiments on the transferability of the students representations and on distilling from a full-precision model to a binary network. For all of these experiments, we jointly train the student model with an additional linear embedding for the student representation. This embedding is used for the correlation loss and is shared by the mutual information loss when there is a mismatch in dimensions between the student and the teacher.

Experiments on CIFAR-100 classification [90] consist of 60K 32×32 RGB images across 100 classes and with a 5:1 training/testing split. The results are shown in table 5.1 for a range

Teacher Student	W40-2 W16-2	W40-2 W40-1	R56 R20	R110 R20	R110 R32	R32x4 R8x4	V13 V8	V13 MN2	R50 MN2	R50 V8	R32x4 SN1	R32x4 SN2	W40-2 SN1
Teacher	75.61	75.61	72.32	74.31	74.31	79.42	74.64	74.64	79.34	79.34	79.42	79.42	75.61
Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36	64.60	64.60	70.36	70.50	71.82	70.50
KD [73]	74.92	73.54	70.66	70.67	73.08	73.33	72.98	67.37	67.35	73.81	74.07	74.45	74.83
FitNet [149]	73.58	72.24	69.21	68.99	71.06	73.50	71.02	64.14	63.16	70.69	73.59	73.54	73.73
AT [205]	74.08	72.77	70.55	70.22	72.31	73.44	71.43	59.40	58.58	71.84	71.73	72.73	73.32
SP [180]	73.83	72.43	69.67	70.04	72.69	72.94	72.68	66.30	68.08	73.34	73.48	74.56	74.52
CC [138]	73.56	72.21	69.63	69.48	71.48	72.97	70.71	64.86	65.43	70.25	71.14	71.29	71.38
RKD [135]	73.35	72.22	69.61	69.25	71.82	71.90	71.48	64.52	64.43	71.50	72.28	73.21	72.21
PKT [136]	74.54	73.45	70.34	70.25	72.61	73.64	72.88	67.13	66.52	73.01	74.10	74.69	73.89
FT [87]	73.25	71.59	69.84	70.22	72.37	72.86	70.58	61.78	60.99	70.29	71.75	72.50	72.03
NST [81]	73.68	72.24	69.60	69.53	71.96	73.30	71.53	58.16	64.96	71.28	74.12	74.68	74.89
CRD [172]	75.64	74.38	71.63	71.56	73.75	75.46	74.29	69.94	69.54	74.58	75.12	76.05	76.27
WCoRD [29]	<u>76.11</u>	74.72	**71.92**	<u>71.88</u>	<u>74.20</u>	<u>76.15</u>	74.72	70.02	70.12	74.68	75.77	76.48	76.68
ReviewKD [30]	**76.12**	<u>75.09</u>	<u>71.89</u>	-	73.89	75.63	<u>74.84</u>	<u>70.37</u>	69.89	-	**77.45**	**77.78**	<u>77.14</u>
$\mathcal{L}_{corr}$	75.85 ±0.12	74.90 ±0.29	71.45 ±0.21	71.77 ±0.34	74.02 ±0.27	75.63 ±0.09	74.70 ±0.27	69.97 ±0.33	**71.41** ±0.41	**75.71** ±0.02	76.80 ±0.28	77.27 ±0.25	**77.35** ±0.25
$\mathcal{L}_{corr} + \mathcal{L}_{gram}$	**76.12** ±0.04	**75.18** ±0.22	71.47 ±0.07	**71.99** ±0.46	**74.26** ±0.05	**76.19** ±0.22	**74.93** ±0.12	**70.39** ±0.39	<u>71.34</u> ±0.33	<u>75.49</u> ±0.32	<u>76.91</u> ±0.19	<u>77.40</u> ±0.06	77.09 ±0.08

Table 5.1: CIFAR-100 test *accuracy* (%) of student networks trained with a number of distillation methods. The best results are highlighted in **bold**, while the second best results are <u>underlined</u>. The mean and standard deviation was estimated over 3 runs. Same-architecture transfer experiments are highlighted in blue, whereas cross-architectural transfer is shown in red.

of student-teacher architecture pairs, where all of the reported methods use the same teacher weights. For a fair comparison, we only compare our results to methods that use the standard CRD [172] teacher weights.

The model abbreviations in the results table are given as follows: Wide residual networks (WRNd-w) [206], MobileNetV2 [54] (MN2), ShuffleNetV1 [209] / ShuffleNetV2 [168] (SN1 / SN2), and VGG13 / VGG8 [163] (V13 / V8). R32x4, R8x4, R110, R56 and R20 denote **CIFAR**-style residual networks, while R50 denotes an **ImageNet**-style ResNet50 [69].

CRCD [216] is not shown in this table since it uses different teacher weights, which are not released. Additionally, using the unofficial code that is released by the authors, we were unable to replicate their reported results. Although both SSKD and HSAKD do provide official implementations and corresponding teacher weights, their use of self-supervision and additional auxiliary tasks is much more computationally expensive and orthogonal to our work. However, we do include these methods in the experiment on ImageNet since the same teacher weights are used.

For all experiments in table 5.1, we set $\beta_{corr} = 2.0$ and $\beta_{gram} = 1.0$ (or $\beta_{gram} = 0.0$ when only using $\mathcal{L}_{corr}$). For the correlation loss α, we use a value of 1.01 for the same architectures and

v.s.	ReviewKD	WCoRD	$\mathcal{L}_{corr}$
$\mathcal{L}_{corr}$	-3.7%	+16.2%	-
$\mathcal{L}_{corr} + \mathcal{L}_{gram}$	+6.8%	+24.4%	+10.5%

Table 5.2: Relative performance improvement (averaged over all architecture pairs in table 5.1) of the correlation and mutual information based losses against ReviewKD, WCoRD and $\mathcal{L}_{corr}$ only.

1.50 for the cross-architectures. ITRD achieves the best performance for 10 out of 13 of the architecture pairs, with a 6.8% and 24.4% relative improvement[4] over ReviewKD and WCoRD respectively. The addition of $\mathcal{L}_{gram}$ is also shown to complement the $\mathcal{L}_{corr}$ loss through a 10.5% average relative improvement over all pairs, as shown in table 5.10.

Experiments on ImageNet classification [153] involve 1.3 million images from 1000 different classes. In this experiment, we set the input size to 224×224, and follow a standard augmentation pipeline of cropping, random aspect ratio and horizontal flipping. We use the *torchdistill* library with standard settings, *i.e.*, 100 epochs of training using SGD with an initial learning rate of 0.1 that is divided by 10 at epochs 30, 60 and 90. The results are shown against the total training efficiency in figure 5.2. The training efficiency is measured in *img/s*, which is inversely proportional to the total training time. For evaluating this metric, we used the official *torchdistill* implementations where possible. In the case of HSAKD, we used their official implementation and for CRCD we used the unofficial implementation provided by the authors. For a fair comparison, the batch sizes were scaled to ensure the training would fit within a pre-determined memory constraint of 8GB, and we used for training an RTX 2080Ti GPU.

[4]For clarity, we use the same definition for relative improvement as provided in WCoRD [29]. This is given by $\frac{X-Y}{X-KD}$, where the X method is compared to Y relative to standard KD with KL divergence.

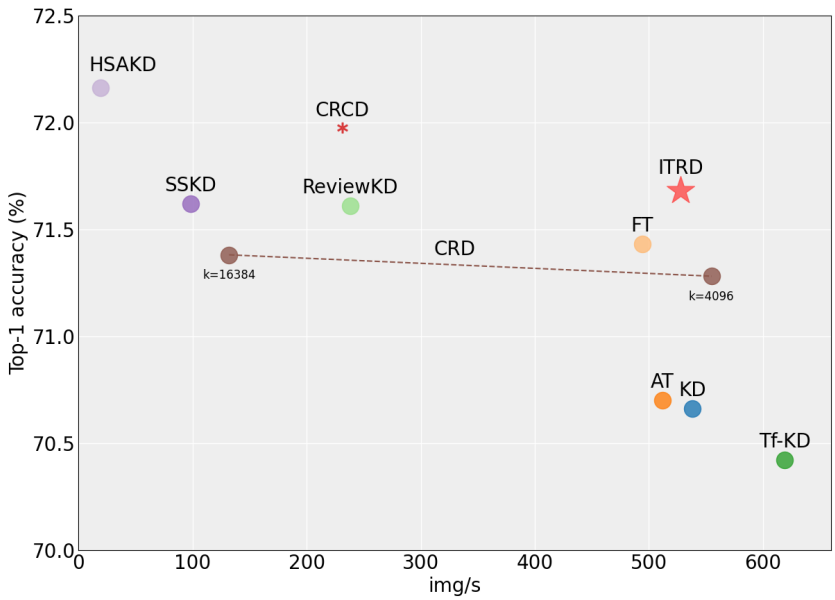


Figure 5.2: Top-1 Accuracy on ImageNet v.s. training efficiency with a ResNet-18 as the student and a pre-trained ResNet-34 as the teacher. For CRCD, the training efficiency was evaluated using the authors unofficial implementation, while this accuracy is reported in their paper.

In terms of accuracy, ITRD achieves an error of 28.32%, being only behind CRCD and HSAKD, which are much more computationally intensive through the use of either negative contrastive sampling and a gradient-based loss, or additional augmented training data. Conversely, our method is computationally efficient. We only introduce a small complexity overhead that comes from a single linear layer that embeds the student and teacher representations in the same space, and also from computing the gram and cross-correlation matrices. The results show the applicability of our method to large-scale datasets, while also being significantly more efficient and simple to adopt.

Ablation study is performed for the impact of the weightings in the loss, namely β_{corr} and β_{gram}. The experiments were performed on CIFAR100 with a ResNet50 for the teacher and a MobileNetV2 for the student. The results are given in figure 5.4 and show that the student's performance is relatively robust to a wide range of values. For the β_{mi} weighting, the average loss maintains within 0.5% and a similar level of variation is achieved for $\beta_{corr} \in [1.5, 2.5]$. We further provide some insight into the choice of α for the correlation loss. Specifically, we evaluate the students performance when trained using a range of values for α, of which the

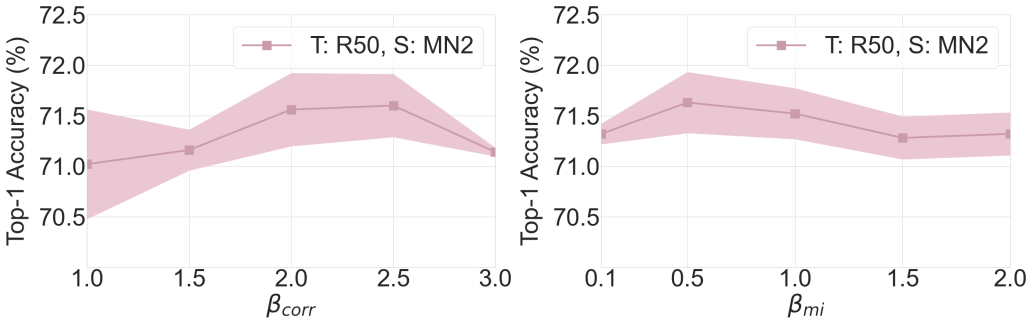


Figure 5.3: Accuracy (%) when varying both the correlation loss (left) and mutual information loss (right) weightings.

α	1.01	1.5	2.0	3.0	4.0	5.0	10.0
Mean	71.15	71.34	71.42	71.32	71.22	70.41	62.91
Std	0.21	0.33	0.39	0.16	0.06	0.43	1.21

Table 5.3: Accuracy (%) when varying α in the correlation loss for CIFAR-100 ResNet50→ MobileNetV2 distillation.

results can be seen in table 5.9. The same dataset and student-teacher architecture are used from the previous ablation experiments. The best results are achieved with $\alpha = 2.0$, which demonstrates the benefit of incorporating Rényi's generalisation for entropy into the proposed losses.

Transferability of representations The main task of representation distillation is to train a smaller model to learn general and discriminative representations of the data. To confirm this result, we explore the task of transferring these models to two different datasets, namely Tiny ImageNet [192], and STL-10 [40]. Tiny ImageNet is a subset of ImageNet that contains 200 classes, with 500 training and 50 validation images per class each of size 64×64. On the other hand, STL-10 contains 10 classes, with 500 training and 800 testing images per class each of size 32×32. A WRN-16-2 student is first trained using ITRD from a WRN-40-2 teacher on the CIFAR100 dataset, after which the representation extractor is frozen and a new linear classifier is fine-tuned on the target data. The results are shown in table 5.4 and show that ITRD outperforms CRD+KD.

	Student	KD	AT	FitNet	CRD	ITRD	Teacher
CIFAR100 → STL-10	69.7	70.9	70.7	70.3	71.6	**72.7**	68.6
CIFAR100 → TinyImageNet	33.7	33.9	34.2	33.5	35.6	**36.0**	31.5

Table 5.4: Transferability of the representations from CIFAR-100 to STL-10 and TinyImageNet. Only the linear classifier heads of each model are fine-tuned on the target datasets. The top-1 classification accuracies are reported (%).

Binary distillation Quantisation is often described as an orthogonal approach for network compression against other methods such knowledge distillation, pruning, and low-rank decomposition. Binary neural networks (BNNs) [47, 140, 101, 191] are an extreme case of quantisation, where the weights can only represent two values. BNNs can obtain a steep increase of inference speed on CPUs [145] and FPGAs [181], while achieving significant model size reduction at the cost of only a small drop in accuracy compared to their full-precision counterparts.

In this section, we show that combining ITRD with binary quantisation can begin to bridge the gap between the binary and full-precision networks. For our experiments we use the state-of-the-art method ReCU [191] as our base model, and we distill the information from a full precision teacher to our binary student. In this experiment, both the full precision teacher and the binary student share the same architecture, the only difference being the quantisation modules in the student. Table 5.5 shows the full set of results, and for all distillation methods we employed the same hyperparameters used in the previous experiments. Both CRD and ReviewKD were shown to degrade the students performance. In contrast, ITRD improves upon the baseline accuracy by 1.3%, which is only 0.7% shy of the full-precision model. Since both the student and teacher networks adopt different non-linearities, the two networks interpolate and extrapolate between data very differently [190], subsequently making effective distillation very difficult. Although we expect a thorough search of hyperparameters for CRD and ReviewKD could improve their performance, the results demonstrate the robustness of ITRD to the training parameters as they were not modified for this experiment.

NLP Question Answering. To show the wide applicability of our method, Table 5.5 shows the results of ITRD in a distillation task on the SQuAD 1.1 [143] reading comprehension task, using the transformer-based [182] BERT [46] as a teacher and modified versions of BERT with

Network	Method	Top-1 (%)
ResNet-18	Full Precision	94.8
	RAD [47]	90.5
	IR-Net [140]	91.5
	RBNN [101]	92.2
	ReCU [191]	92.8
	ReCU + CRD	92.1
	ReCU + ReviewKD	92.6
	ReCU + KD	93.3
	ReCU $+\mathcal{L}_{corr}$	93.9
	ReCU $+\mathcal{L}_{corr} + \mathcal{L}_{gram}$	**94.1**
VGG-small	Full Precision	94.1
	XNOR-Net [145]	89.8
	BNN [43]	89.9
	DoReFa [214]	90.2
	IR-Net [140]	90.4
	RBNN [101]	91.3
	DSQ [57]	91.7
	SLB [194]	92.0
	ReCU [191]	92.2
	ReCU+KD	92.5
	ReCU + CRD	91.3
	ReCU $+\mathcal{L}_{corr}$	**93.4**
	ReCU $+\mathcal{L}_{corr} + \mathcal{L}_{gram}$	93.3

Table 5.5: Performance comparison with the state-of-the-art on CIFAR-10 for binary networks. All models, except full precision, use a bit length of 2 for the weights and activations.

	Model	EM	F1
	Teacher (BERT)	81.5	88.6
T6	DistilBERT	79.1	86.9
	TextBrewer	80.8	88.1
	ITRD	**81.5**	**88.5**
T3	TextBrewer	76.3	84.8
	ITRD	**77.7**	**85.8**

Table 5.6: Question Answering on SQuAD 1.1. The teacher architecture, BERT, contains 12 layers, whereas the students, *T6* and *T3*, follow the same architecture as BERT but with 6 and 3 layers respectively.

fewer layers as the students. For this experiment, we use the same hyperparameters used in the previous experiments, and following TextBrewer we apply ITRD to the output of each of the student transformer layers, and also use a standard KD [74] loss between the teacher and students logits. Table 5.6 shows that we outperform both NLP-specific distillation methods TextBrewer [195] and DistilBert [157] in both the Exact Match (EM) metrics and in F1 score.

5.6 Discussion

Reproducibility To aid the reproducibility of this work, we implemented ITRD in both the CRD evaluation framework [172] and the *torchdistill* [116] KD reproducibility framework, which will both be released. Furthermore, the pseudo-code in algorithm 2.1 encapsulates both losses, showing the simplicity of using the proposed losses in current KD settings. We hope that the release of the code, along with the computational simplicity of our approach will encourage further development of this work.

Limitations and future work. A large amount of the spatial information is lost by the final representation, which is the space in which the distillation losses are defined. This loss of spatial information may be regarded as a limitation for more structured tasks (*e.g.*, semantic segmentation or object detection), however, we expect that research in this direction would be a natural extension of this work. Another promising direction for this work is in the context of deep mutual learning. Jointly training both the teacher and student models can provide effective collaboration at the cost of increased training time.

5.7 Conclusion

In this work, we proposed an information-theoretic setting for representation distillation. Using this framework, we introduce novel distillation losses that are very simple and computationally inexpensive to adopt into most deep learning pipelines. Each of the proposed losses aim to extract complementary information from the teacher network. The correlation loss aims to guide the student to match the teacher representation on a feature-level. Conversely, the mutual information loss aims to transfer the intra-batch similarity between samples from the teacher to the student. We have shown the superiority of our approach compared to methods of similar computational costs on standard classification benchmarks. Furthermore, we have shown the applicability of our method to binary networks, whereby we begin to bridge the performance gap between full-precision and binary networks.

5.8 Supplementary

In this section we provide some supplementary experiments demonstrating the robustness of our loss to varying values of α, β_{corr}, and β_{gram}. Additionally, we explore the impact of the choice of kernel function for the mutual information loss and a comparison of the of the training costs and convergence against other state-of-the-art distillation methods.

5.8.1 Mutual Information Loss

Information potential The *log* transformation must be used in both the marginal and joint entropy terms to be a valid measure of mutual information. Without it, the loss resembles a distance between these quantities, rather than a ratio. However, empirically, we observed a small degradation in accuracy when using the log in both these terms, which can be seen in table 5.7.

$\mathcal{L}_{mi}$	$G_s - G_{st}$	$log_2(G_s) - log_2(G_{st})$
Mean	71.52	71.36
Std	0.25	0.35

Table 5.7: Accuracy (%) with and without the $\log_2$ data transformation. The experiments were performed for CIFAR-100 ResNet50→ MobileNetV2 distillation.

Exploring different kernels κ The kernel used for the $\mathcal{L}_{gram}$ loss was a polynomial kernel of degree 2, however, we also considered the use of a radial basis function (RBF) kernel $\kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$. To select the values of σ, we then used Silverman's rule of thumb [162] $\sigma = h \times n^{-1/(4+d)}$, where n is the size of the mini-batch, d is the dimensionality of the representations, while h is an empirical value. The results can be seen in table 5.8 for both $h = 1.0$, $h = 5.0$, and the polynomial kernel. Although the RBF kernel did show promising results, the value of h is very dependant on both the dataset and the architectures used. To promote reproducibility of our results, we thus chose to use the polynomial kernel throughout.

κ	Polynomial	RBF ($\sigma \approx 1.0$)	RBF ($\sigma \approx 5.0$)
Mean	71.52	71.14	70.99
Std	0.25	0.18	0.08

Table 5.8: Accuracy (%) with the RBF kernel for the $\mathcal{L}_{gram}$ with different kernel sizes σ. The experiments were performed for CIFAR-100 ResNet50→ MobileNetV2 distillation.

5.8.2 Correlation Loss

Relationship to joint entropy. The objective from equation 6 closely resembles maximising $\log_2 \sum_i v_i$. However, although these two objectives share the same optimum solution, the flexibility in tuning the sharpness of the loss with α proved very effective. If we consider the self correlation matrices $\mathbf{C}_{ss}$ and $\mathbf{C}_{tt}$, the diagonal entries in $(\mathbf{C}_{ss} \circ \mathbf{C}_{tt})^2$ [5] will be populated with products of pairs of cross-correlation terms between $\mathbf{Z}_s$ and $\mathbf{Z}_t$. This matrix construction can then be used in equation 4 to compute the joint α-order entropy between the student and the teacher, where $\alpha = 2$. In the case where the features are strictly independent, *i.e.*,

[5] This exponent denotes the square of a matrix, rather than an element-wise operation.

$(\mathbf{C}_{ss})_{ij} = (\mathbf{C}_{tt})_{ij} = 0 \quad \forall\, i \neq j$, the objective of the proposed loss in equation 6 and maximising this joint entropy are equivalent. In the more general setting, the joint entropy formulation maximises the correlation between all pairs of exemplars, while our proposed loss only maximises the correlation along the leading diagonal of $\mathbf{C}_{st}$.

Correlation v.s. Gram matrices. The connection to joint entropy is limited in that the matrices used are correlation matrices as opposed to Gram matrices in equation 4. This is an important distinction since in this loss we wish to capture the similarity across the feature-dimension as opposed to the batch-dimension. However, despite this distinction, there is still an intimate connection between these two matrices. As discussed in the recent work on cross-covariance attention [51], the non-zero part of the eigenspectrum of the Gram and covariance matrices are equivalent. Since the entropy-like formulation described in equation 3 is a spectral function of $\mathbf{A}$, the two resulting quantities are in turn closely related.

Ablation study is performed for the impact of the weightings in the loss, namely β_{corr} and β_{gram}. The experiments were performed on CIFAR100 with a ResNet50 for the teacher and a MobileNetV2 for the student. The results are given in figure 5.4 and show that the student's performance is relatively robust to a wide range of values. For the β_{gram} weighting, the average loss maintains within 0.5% and a similar level of variation is achieved for $\beta_{corr} \in [1.5, 2.5]$. We further provide some insight into the choice of α for the correlation loss. Specifically, we evaluate the students performance when trained using a range of values for α, of which the results can be seen in table 5.9. The same dataset and student-teacher architecture are used from the previous ablation experiments. The best results are achieved with $\alpha = 2.0$, which demonstrates the benefit of incorporating Rényi's generalisation for entropy into the proposed losses.

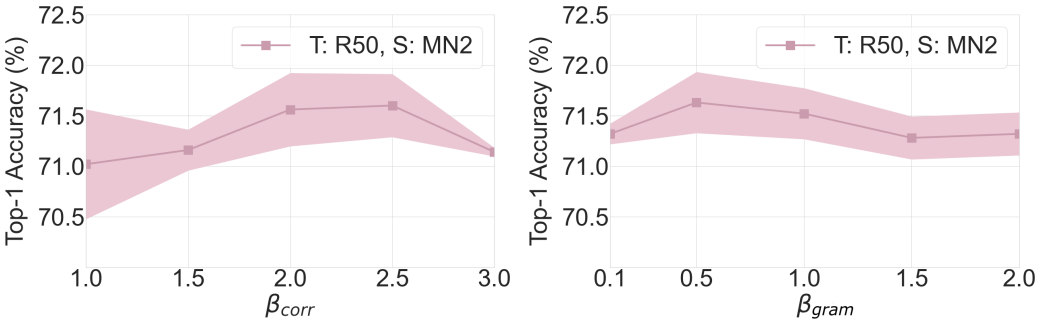


Figure 5.4: Accuracy (%) when varying both the correlation loss (left) and mutual information loss (right) weightings.

α	1.01	1.5	2.0	3.0	4.0	5.0	10.0
Mean	71.15	71.34	71.42	71.32	71.22	70.41	62.91
Std	0.21	0.33	0.39	0.16	0.06	0.43	1.21

Table 5.9: Accuracy (%) when varying α in the correlation loss for CIFAR-100 ResNet50→ MobileNetV2 distillation.

Reproducibility To aid the reproducibility of this work, we implemented ITRD in both the CRD evaluation framework [172] and the *torchdistill* [116] KD reproducibility framework, which will both be released. Furthermore, the pseudo-code in algorithm 5.1 encapsulates both losses, showing the simplicity of using the proposed losses in current KD settings. We hope that the release of the code, along with the computational simplicity of our approach will encourage further development of this work.

Model Architectures In experiments, we utilize the following model architectures.

- Wide Residual Network (WRN) [206]: WRN-d-w represents wide ResNet with depth d and width factor w.
- resnet [69]: We use ResNet-d to represent CIFAR-style resnet with 3 groups of basic blocks, each with 16, 32, and 64 channels, respectively. In our experiments, resnet8x4 and resnet32x4 indicate a 4 times wider net- work (namely, with 64, 128, and 256 channels for each of the blocks).

- ResNet [69]: ResNet-d represents ImageNet-style ResNet with bottleneck blocks and more channels.

- MobileNetV2 [54]: In our experiments, we use a width multiplier of 0.5.

- VGG [163]: The VGG networks used in our experiments are adapted from their original ImageNet counterpart.

- ShuffleNetV1 [209], ShuffleNetV2 [114]: ShuffleNets are proposed for efficient training and we adapt them to input of size 32x32.

Implementation Details The CIFAR100 experimental evaluation and architectures used for comparisons are provided by Tian *et al.*in their work on contrastive representation distillation [172]. For the ImageNet experiments, we use the *torchdistill* [116] reproducibility framework, and for the binary distillation experiments we use the code provided by ReCU [191]. For completeness, we include the detail of the CRD provided architectures and training schedules here:

All methods evaluated in our experiments use SGD. For CIFAR-100, we initialize the learning rate as 0.05, and decay it by 0.1 every 30 epochs after the first 150 epochs until the last 240 epoch. For MobileNetV2, ShuffleNetV1 and ShuffleNetV2, we use a learning rate of 0.01 as this learning rate is optimal for these models in a grid search, while 0.05 is optimal for other models.

Training costs Figure 5.2 provides the ImageNet training costs with the same hardware and a fixed memory constraint, where ITRD is relatively on par with the cheap standard KD. Table 5.10 shows additional results on the memory overhead for a fixed batch-size. In addition to these metrics, for the R34 $\rightarrow$ R18 ImageNet comparison, ITRD adds only **0.26M** trainable parameters by using a linear embedding layer, whereas e.g., ReviewKD introduces **1.8M** and CRD/WCoRD use a memory bank that stores tens of millions of parameters.

v.s.	ReviewKD	CRCD	HSAKD
Memory	**4.08**×	over GPU (24GB) limit	**5.61**×
Train time	**2.22**×	**4.00**×	**31.06**×

Table 5.10: Relative overhead in terms of memory and training time against main competing distillation methods on ImageNet. Training time used a variable batch size to fit a pre-defined memory limit, while the memory experiments were using a fixed batch size.

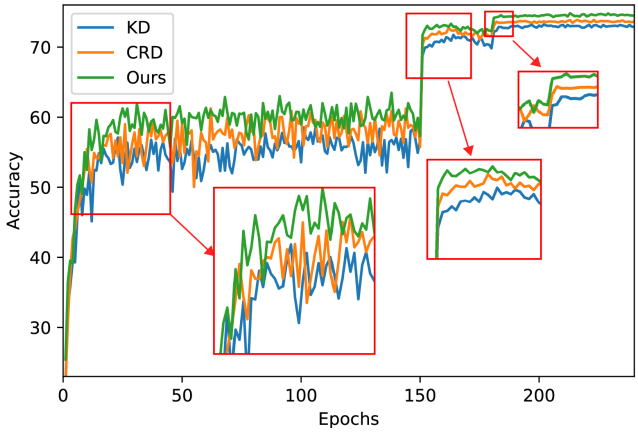


Figure 5.5: Training epochs *vs* validation accuracy for VGG13→VGG8 CIFAR 100 distillation. Zoomed-in regions show that our method converges faster to a higher accuracy.

Training convergence. Our method achieves a fast convergence, thus training with fewer epochs may lead to similar results to using the full training schedule. As shown in figure 5.5, ITRD achieves a higher final accuracy and also converges much faster after each learning rate drop than two of the main competitors (*i.e.*, CRD and KD).

Hyper-parameters fine-tuning The proposed ITRD framework does not use any KL divergence for the logits. This would need additional tuning of α and τ, where τ is often very dataset dependent. Lots of other methods report their results in conjunction with the KL loss [172, 29]. In addition, these methods also contain extra hyperparameters that need to be carefully tuned. For example, WCoRD[29] shows a robustness over a very small range $\mathbf{0} < \lambda_{\mathbf{2}} < \mathbf{0.2}$, while we provide a much larger range of values tested for our hyperparameters (β_{corr}, β_{gram}, α) whilst providing a comparable, or strictly lower, variation of performance. Similarly, CRD performance is also highly dependent on the number of negative samples used or the temper-

ature chosen. Furthermore, we also use the same hyper-parameters used in the paper for the NLP experiments, thus demonstrating the robustness to the choice of hyperparameters.

Chapter 6

Understanding the Role of the Projector in Knowledge Distillation

In this chapter, we revisit the efficacy of knowledge distillation as a function matching and metric learning problem. In doing so we verify three important design decisions, namely the normalisation, soft maximum function, and projection layers as key ingredients. We theoretically show that the projector implicitly encodes information on past examples, enabling relational gradients for the student. We then show that the normalisation of representations is tightly coupled with the training dynamics of this projector, which can have a large impact on the students performance. Finally, we show that a simple soft maximum function can be used to address any significant capacity gap problems. Experimental results on various benchmark datasets demonstrate that using these insights can lead to superior or comparable performance to state-of-the-art knowledge distillation techniques, despite being much more computationally efficient. In particular, we obtain these results across image classification (CIFAR100 and ImageNet), object detection (COCO2017), and on more difficult distillation objectives, such as training data efficient transformers, whereby we attain a 77.2% top-1 accuracy with DeiT-Ti on ImageNet. All code and model checkpoints for this chapter are available on GitHub [1].

[1] https://github.com/roymiles/Simple-Recipe-Distillation

6.1 Introduction

Although knowledge distillation has shown to be very effective, there are still some limitations related to the computational and memory overheads in constructing and evaluating the losses, as well as an insufficient theoretical explanation for the underlying core principles. To overcome these limitations, we revisit knowledge distillation from both a function matching and metric learning perspective. We perform an extensive ablation of three important components of knowledge distillation, namely the distance metric, normalisation, and projector network. Alongside this ablation we provide a theoretical perspective and unification of these design principles through exploring the underlying training dynamics. Finally, we extend these principles to a few large scale vision tasks, whereby we achieve comparable or improved performance over state-of-the-art. The most significant result of which pertains to the data-efficient training of transformers, whereby a performance gain of 2.2% is achieved over the best-performing distillation methods that are designed explicitly for this task. Our main contributions can be summarised as follows.

- We explore three distinct design principles from knowledge distillation, namely the projection, normalisation, and distance function. In doing so we demonstrate their unification and coupling with each other, both through analytical means and by observing the training dynamics.

- We show that a projection layer implicitly encodes relational information from previous samples. Using this knowledge we can remove the need to explicitly construct correlation matrices or memory banks that will inevitably incur a significant memory overhead.

- We propose a simple recipe for knowledge distillation using a linear projection, batch normalisation, and a $LogSum$ function. These three design choices can attain competitive or improved performance to state-of-the-art for image classification, object detection, and the data efficient training of transformers.

6.2 Related Work

Knowledge Distillation Knowledge distillation is the process of transferring the knowledge from a large, complex model to a smaller, simpler model. Its usage was originally proposed in the context of image classification [73] whereby the soft teacher predictions would encode relational information between classes. Spherical KD [60] extended this idea by re-scaling the logits, prime aware adaptive distillation [210] introduced an adaptive weighting strategy, while DKD [211] proposed to decouple the original formulation into target class and non-target class probabilities.

Hinted losses [149] were a natural extension of the logit-based approach whereby the intermediate feature maps are used as hints for the student. Attention transfer [205] then proposed to re-weight this loss using spatial attention maps. ReviewKD [30] addressed the problem relating to the arbitrary selection of layers by aggregating information across all the layers using trainable attention blocks. Neuron selectivity transfer [81], similarity-preserving KD [180], and relational KD [135] construct relational batch and feature matrices that can be used as inputs for the distillation losses. Similarly FSP matrices [196] were proposed to extract the relational information through a residual block. In contrast to this theme, we show that a simple projection layer can implicitly capture most relational information, thus removing the need to construct any expensive relational structures.

Representation distillation was originally proposed alongside a contrastive based loss [172] and has since been extended using a Wasserstein distance [29], information theory (discussed in chapter 5), graph theory [115], and complementary gradient information [216]. Distillation also been empirically shown to benefit from longer training schedules and more data-augmentation [19], which is similarly observed with HSAKD [193] and SSKD [189]. Distillation between CNNs and transformers has also been a very practically motivated task for data-efficient training [177] and has shown to benefit from an ensemble of teacher architectures [147]. However, we show that just a simple extension of some fundamental distillation design principles is much more effective.

Self-Supervised Learning Self-supervised learning (SSL) is an increasingly popular field of machine learning whereby a model is trained to learn a useful representation of unlabelled data. Its popularity has been driven by the increasing cost of manual labelling and has since been crucial for training large transformer models. Various pretext tasks have been proposed to learn these representations, such as image inpainting [67], colorization [208], or prediction of the rotation [56] or position of patches [49, 25]. SimCLR [32] approached self-supervision using a contrastive loss with multi-view augmentation to define the positive and negative pairs. They found a large memory bank of negative representations was necessary to achieve good performance, but would incur a significant memory overhead. MoCo [68] extending this work with a momentum encoder, which was subsequently extended by MoCov2 [36] and MoCov3 [35].

Asymmetric architectures were proposed as an alternative to contrastive learning, whereby no negative samples are needed. Most noticeable works in this area are BYOL [59] and SimSiam [34] which both use stop gradients to avoid representation collapse. DirectPred [173] provided a theoretical understanding of this non-contrastive SSL setting. They introduced a series of conceptual insights into the functional roles of many crucial ingredients used. In doing so they derived a simple linear predictor with competitive performance to some much more complex predictor architectures. In our work we explore knowledge distillation from a similar perspective as DirectPred but observe some unique observations and results pertaining to this distillation setting.

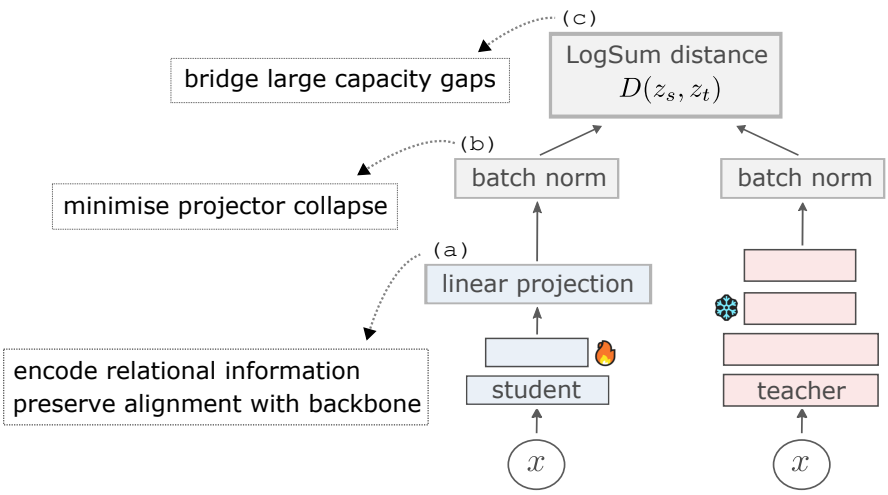
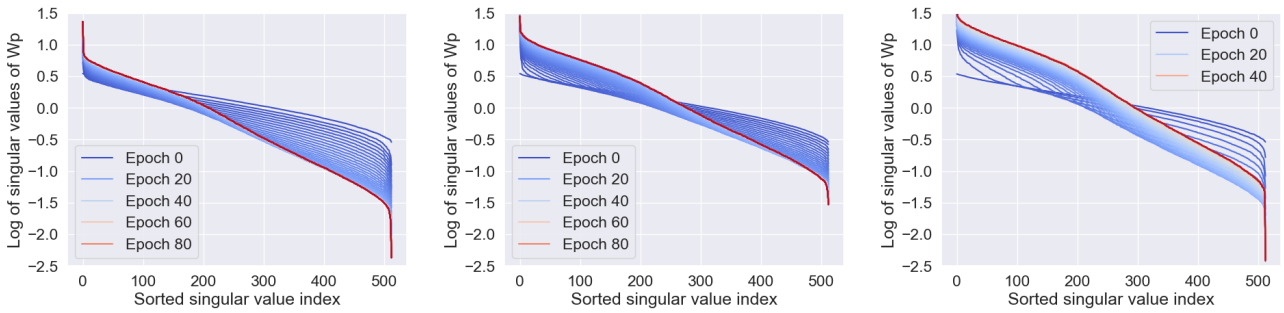


Figure 6.1: Proposed feature distillation pipeline using three distinct components: linear projection (a), batch norm (b), and a *LogSum* distance (c). We provide an interpretable explanation for each each of these three components, which results in a very cheap and effective recipe for distillation.



(a) no normalisation. (b) L2 normalisation. (c) batch normalisation.

Figure 6.2: Evolution of singular values of the projection weights $\mathbf{W}_p$ under three different representation normalisation schemes. The student is a Resnet-18, while the teacher is a ResNet-50. The three curves shows the evolution of singular values for the projector weights when the representations undergo no normalisation, L2 normalisation, and batch norm respectively.

Feature decorrelation is another approach to SSL that avoids the need for negative pairs to address representation collapse. Both Barlow twins [207] and VICReg [15] achieve this by maximising the variance within a batch, while preserving invariance to augmentations. Both contrastive learning and feature decorrelation have since been extended to dense prediction tasks [16] and unified with knowledge distillation [123].

6.3 Understanding the Role of the Projector

Knowledge distillation (KD) is a technique used to transfer knowledge from a large, powerful model (teacher) to a smaller, less powerful one (student). In the classification setting, it can be done using the soft teacher predictions as pseudo-labels for the student. Unfortunately, this approach does not trivially generalise to non-classification tasks [105] and the classifier may collapse a lot of information [176] that can be useful for distillation. Another approach is to use feature maps from the earlier layers for distillation [149, 205], however, its usage presents two primary challenges: the difficulty in ensuring consistency across different architectures [30] and the potential degradation in the student's downstream performance for cases where the inductive biases of the two networks differ [172] A compromise, which strikes a balance between the two approaches discussed above, is to distill the representation directly before the output space. Representation distillation has been successfully adopted in past works [172, 216, 122]

and is the focus of this paper. The exact training framework used is described in figure 6.1 alongside an extension to incorporates a logit distillation loss. The projection layer shown was originally used to simply match the student and teacher dimensions [149], however, we will show that its role is much more important and it can lead to significant performance improvements even when the two feature dimensions already match. The two representations are typically both followed by some normalisation scheme as a way of appropriately scaling the gradients. However, we find this normalisation has a more interesting property in its relation to what information is encoded in the learned projector weights.

In this work we provide a theoretical perspective to motivate some simple and effective design choices for knowledge distillation. In contrast to the recent works [180, 122], we show that an explicit construction of complex relational structures, such as feature kernels [66] is not necessary. In fact, most of this structure can be learned implicitly through the tightly coupled interaction of a learnable projection layer and an appropriate normalisation scheme. In the following sections we investigate the training dynamics of the projection layer with the choice of normalisation scheme. We explore the impact and trade-offs that arise from the architecture design of the projector. Finally, we propose a simple modification to the distance metric to address issues arising from a large capacity gap between the student and teacher models. Although we do not aim to necessarily propose a new method for distillation, we uncover a cheap and simple recipe that can transfer to various distillation settings and tasks. Furthermore, we provide a new theoretical perspective on the underlying principles of distillation that can translate to large scale vision tasks.

The projection weights encode relational information from previous samples. The projection layer plays a crucial role in KD as it provides an implicit encoding of previous samples and its weights can capture the relational information needed to transfer information regarding the correlation between features. We observe that even a single linear projector layer can provide significant improvements in accuracy (see Supplementary). This improvement suggests that the projections role in distillation can be described more concisely as being an encoder of essential information needed for the distillation loss itself. Most recent works propose a

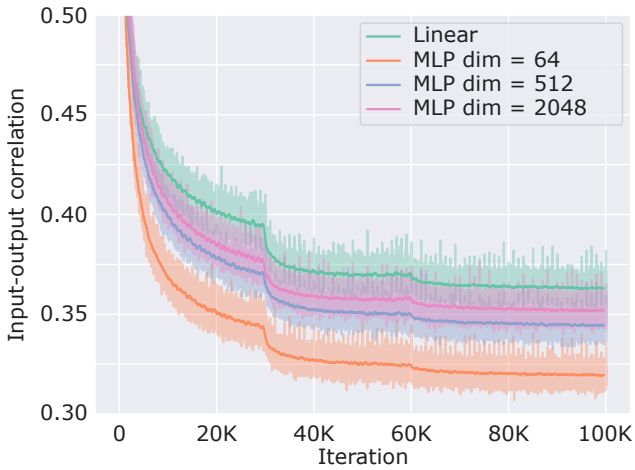


Figure 6.3: Correlation between input-output features using different projector architectures. All projector architectures considered will gradually decorrelate the input-output features. Although this decorrelation is attributed to the layer removing irrelevant information, it can degrade the efficacy of distilling through to the student backbone.

manual construction of some relational information to be used as part of a loss [135, 180], however, we posit that an implicit and learnable approach is much more effective. To explore this phenomenon in more detail, we consider the update equations for the projector weights and its training dynamics. Without loss in generality, consider a simple L2 loss and a linear projection layer without any bias term.

$$D(\mathbf{Z}_s, \mathbf{Z}_t\ ; \mathbf{W}_p) = \frac{1}{2} \left\| \mathbf{Z}_s \mathbf{W}_p - \mathbf{Z}_t \right\|_2^2 \tag{6.1}$$

Where $\mathbf{Z}_s$ and $\mathbf{Z}_t$ are the **s**tudent and **t**eacher representations respectively, while $\mathbf{W}_p$ is the matrix representing the linear **p**rojection. Using the trace property of the Frobenius norm, we can then express this loss as follows:

$$D(\mathbf{Z}_s, \mathbf{Z}_t\ ; \mathbf{W}_p) = \frac{1}{2} tr\left(\left(\mathbf{Z}_s \mathbf{W}_p - \mathbf{Z}_t\right)^T \left(\mathbf{Z}_s \mathbf{W}_p - \mathbf{Z}_t\right) \right) \tag{6.2}$$

$$= \frac{1}{2} tr(\mathbf{W}_p^T \mathbf{Z}_s^T \mathbf{Z}_s \mathbf{W}_p - \mathbf{Z}_t^T \mathbf{Z}_s \mathbf{W}_p \tag{6.3}$$

$$- \mathbf{W}_p^T \mathbf{Z}_s^T \mathbf{Z}_t + \mathbf{Z}_t^T \mathbf{Z}_t) \tag{6.4}$$

Taking the derivative with respect to $\mathbf{W}_p$, we can derive the update rule $\dot{\mathbf{W}}_p$

$$\dot{\mathbf{W}}_p = -\frac{\partial D(\mathbf{W}_p)}{\partial \mathbf{W}_p} \tag{6.5}$$

$$= -\mathbf{Z}_s^T \mathbf{Z}_s \mathbf{W}_p + \mathbf{Z}_s^T \mathbf{Z}_t \tag{6.6}$$

which can be further simplified

$$\boxed{\dot{\mathbf{W}}_p = \mathbf{C}_{st} - \mathbf{C}_s \mathbf{W}_p} \tag{6.7}$$

where $\mathbf{C}_s = \mathbf{Z}_s^T \mathbf{Z}_s \in \mathbb{R}^{d_s \times d_s}$ and $\mathbf{C}_{st} = \mathbf{Z}_s^T \mathbf{Z}_t \in \mathbb{R}^{d_s \times d_t}$ denote self and cross correlation matrices that capture the relationship between features. Due to the capacity gap between the student network and the teacher network, there is no perfect linear projection between these two representation spaces. Instead, the projector will converge on an approximate projection that we later show is governed by the normalisation being employed.

<u>Whitened features:</u> consider using self-supervised learning in conjunction with distillation whereby the student features are whitened to have perfect decorrelation [52], or alternatively, they are batch normalised and sufficiently regularised with a feature decorrelation term [15]. In this setting, the fixed point solution for the projection weights will be symmetric and will capture the cross relationship between student and teacher features.

$$\mathbf{C}_{st} - \mathbf{C}_s \mathbf{W}_p = 0 \quad \text{where} \quad \mathbf{C}_s = \mathbf{I} \tag{6.8}$$

$$\rightarrow \mathbf{W}_p = \mathbf{C}_{st} \tag{6.9}$$

Other normalisation schemes, such as those that jointly normalise the projected features and the teacher features, will have a much more involved analysis but will unlikely provide any additional insights on the dynamics of training itself. Thus, we propose to empirically explore the training trajectories of the projector weights singular values. This exploration will help quantify how the projector is mapping the student features to the teachers space. We cover this in the next section along with some additional insights into what is being learned and distilled.

The choice of normalisation directly affects the training dynamics of $\mathbf{W}_p$. Equation 6.7 shows that the projector weights can encode relational information between the student and teacher's features. This suggests redundancy in explicitly constructing and updating a large memory bank of previous representations [172]. By considering a weight decay η and a learning rate α_p, the update equation can be given as follows:

$$\mathbf{W}_p \rightarrow \mathbf{W}_p + \alpha_p \dot{\mathbf{W}}_p - \eta \mathbf{W}_p \tag{6.10}$$

$$= (1 - \eta)\mathbf{W}_p + \alpha_p \dot{\mathbf{W}}_p \tag{6.11}$$

By setting $\eta = \alpha_p$ we can see that the projection layer will reduce to a moving average of relational features, which is very similar to the momentum encoder used by CRD [172]. Other works suggest to extract relational information on-the-fly by constructing correlation or Gram matrices [122, 138]. We show that this is also not necessary and more complex information can be captured through a simple linear projector. We also demonstrate that, in general, the use of a projector will scale much more favourably for larger batch sizes and feature dimensions. We also note that the handcrafted design of kernel functions [85, 66] may not generalise to real large scale datasets without significant hyperparameter tuning.

From the results in table 6.1, we observe that when fixing all other settings, the choice of normalisation can significantly affect the student's performance. To explore this in more detail, we consider the training trajectories of $\mathbf{W}_p$ under different normalisation schemes. We find that the choice of normalisation not only controls the training dynamics, but also the fixed point solution (see equation 6.7). We argue that the efficacy of distillation is dependent on how much relational information can be encoded in the learned weights and how much information is lost through the projection. To jointly evaluate these two properties we show the evolution of singular values of the projector weights during training. The results can be seen in figure 6.2 and show that the better performing normalisation methods (table 6.1) are shrinking far

fewer singular values towards zero. This shrinkage can be described as collapsing the input along some dimension, which will induce some information loss and it is this information loss that degenerates the efficacy of the distillation process.

Description	RegNet-Y → MBv2	ViT → MBv3	ConvNext → EffNet-b0
No Distillation	50.89	54.13	64.48
L2 Norm	52.91	54.65	64.23
Group Norm	55.63	59.08	65.52
Batch Norm	**56.09**	**59.28**	**67.95**

Table 6.1: Normalisation ablation for distillation across a range of architecture pairs on an ImageNet-1K 20% subset. Although distillation improves performance with a variety of normalisation schemes, we find batch normalisation is consistently the most effective.

Larger projector networks learn to decorrelate the input-output features. One natural extension of the previous observations is to use a larger projector network to encode more information relevant for the distillation loss. Unfortunately, we observe that a trivial expansion of the projection architecture does not necessarily improve the students performance. To explain this observation we evaluate a measure of decorrelation between the input and output features of these projector networks. The results can be seen in figure 6.3 and we can see that the larger projectors learn to decorrelate more and more features from the input. This decorrelation can lead to the projector learning features that are not shared with the student backbone, which will subsequently diminish the effectiveness of distillation. These observations suggest that there is an inherent trade-off between the projector capacity and the efficacy of distillation. We note that the handcrafted design of the projector architecture is a motivated direction for further research [37, 129]. However, in favour of simplicity, we choose to use a linear projector for all of our large scale evaluations.

Teacher → Student	1.0	1.5	2.0	2.5	3.0	4.0	5.0
ResNet50 → ResNet18	61.74	62.18	62.23	62.84	63.10	63.32	**63.40**
ConvNext → EffNet-b0	65.52	66.69	66.61	67.10	67.72	**68.51**	67.69

Table 6.2: Ablating the importance of α. Distillation is generally robust for various values of α, but consistently optimal in range 4-5 across various architecture pairs.

Description	ViT → MBv3	ConvNext → EffNet-b0	ResNet-50 → ResNet-18
wo/ LogSum	59.28	67.95	70.03
w/ LogSum	**59.80**	**68.51**	**71.29**

Table 6.3: LogSum ablation across various architecture pairs. Left: 20% subset. Right: Full ImageNet. The soft maximum function provides consistent improvement across both the CNN→CNN and ViT→CNN distillation settings.

Teacher Student	WRN40-2 WRN16-2	WRN40-2 WRN40-1	R56 R20	R32×4 R8×4	VGG13 MBv2	R50 MBv2	R50 VGG8	R32×4 ShuffleV1	R32×4 ShuffleV2	WRN40-2 ShuffleV1
Teacher	76.46	76.46	73.44	79.63	75.38	79.10	79.10	79.63	79.63	76.46
Student	73.64	72.24	69.63	72.51	65.79	65.79	70.68	70.77	73.12	70.77
KD	74.92	73.54	70.66	73.33	67.37	67.35	73.81	74.07	74.45	74.83
FitNet	75.75	74.12	71.60	74.31	68.58	68.54	73.84	74.82	75.11	75.55
AT	75.28	74.45	71.78	74.26	69.34	69.28	73.45	74.76	75.30	75.61
CRD	76.04	75.52	71.68	75.90	68.49	70.32	74.42	75.46	75.72	75.96
SSKD	76.04	**76.13**	71.49	76.20	71.53	72.57	75.76	**78.44**	78.61	77.40
Our Method	**76.14**	75.42	**71.75**	**76.44**	**71.47**	**72.81**	**76.20**	77.32	**79.06**	**79.22**
KD†	75.94	75.32	71.10	75.84	70.79	71.29	75.75	77.80	78.43	78.00
CRD†	77.27	**76.15**	72.21	77.69	71.65	72.03	75.73	78.57	79.01	78.54
DKD†	74.96	75.89	70.95	77.52	72.01	73.30	76.88	**79.71**	**80.08**	77.86
Our Method	**77.61**	76.04	**72.25**	**78.37**	**72.82**	**73.51**	**77.08**	78.99	79.86	**78.79**

Table 6.4: KD between Similar and Different Architectures. Top-1 accuracy (%) on CIFAR100. Bold is used to denote the best results. All reported models are trained using pairs of augmented images. Those reported in the top box use RandAugment [44] strategy, while those in the bottom box use pre-defined rotations, as used in SSKD. † denotes reproduced results in a new augmentation setting using the authors provided code.

The soft maximum function can address distilling across a large capacity gap. When the capacity gap between the student and the teacher is large, representation distillation can become challenging. More specifically, the student network may have insufficient capacity to perfectly align these two spaces and in attempting to do so may degrade its downstream performance. To addresses this issue we explore the use of a soft maximum function which will soften the contribution of relatively close matches in a batch. In this way the loss can be adjusted to compensate for poorly aligned features which may arise when the capacity gap is large. The family of functions which share these properties can be more broadly defined through a property of their gradients. In favour of simplicity, we use the simple *LogSum* function throughout our experiments.

$$D(\mathbf{Z}_s, \mathbf{Z}_t\ ; \mathbf{W}_p) = log \sum_i |\mathbf{Z}_s \mathbf{W}_p - \mathbf{Z}_t|_i^\alpha \tag{6.12}$$

where α is a smoothing factor. We also note that other functions, such as the $LogSumExp$, with a temperature parameter τ, have been used in SimCLR and CRD to a similar effect. Table 6.1 shows the importance of feature normalisation across a variety of student-teacher architecture pairs. Batch normalisation provides the most consistent improvement that even extends to the Transformer $\rightarrow$ CNN setting. In table 6.3 we highlight the importance of the $LogSum$ function, which is most effective in the large capacity gap settings, as evident from the 1% improvement for R50 $\rightarrow$ R18. Table 6.2 provides an ablation of the importance of the α parameter, whereby we observe that the performance is relatively robust to a wide range of values, but consistently optimal in the range 4-5.

6.4 Benchmark Evaluation

Implementation details. We follow the same training schedule as CRD [172] for both the CIFAR100 and ImageNet experiments. For the object detection, we use the same training schedule as ReviewKD [30], while for the data efficient training we use the same as Co-Advice [147]. All experiments were performed on a single NVIDIA RTX A5000. When using batch normalisation for the representations, we removed the affine parameters and set $\epsilon = 0.0001$.

6.4.1 Classification on CIFAR100 and ImageNet

Experiments on the CIFAR-100 classification task [90] consist of 60K 32×32 RGB images across 100 classes with a 5:1 training/testing split. Table 6.4 shows the results for several student-teacher pairings. To enable a fair evaluation, we have only included the methods that use the same teacher weights provided by SSKD[189]. In these experiments we use an MLP projector with a hidden size of 1024 and no additional KL divergence loss. We confirm that not only

is the choice of augmentation critical for good performance [19] on this dataset, but applying our principles can attain state-of-the-art across most architecture pairs. The most significant improvements pertain to the cross-architecture experiments or where the capacity gap is large. We provide two sets of experiments with and without introducing a wider set of augmentations. In both settings we maintain the same optimiser, learning rate scheduler, and training duration.

The ImageNet [153] classification uses 1.3 million images that are classified into 1000 distinct classes. The input size are set to 224 x 224, and we employed a typical augmentation procedure that includes cropping and horizontal flipping. We used the torchdistill library with the standard configuration, which involves 100 training epochs using SGD and an initial learning rate of 0.1, which is decreased by a factor of 10 at epochs 30, 60, and 90. The results can be seen in table 6.6 and although the choice of architectures is not in favour of our method since the capacity gap is small, we are still able to attain competitive performance. Other methods, such as ICKD [104] or SimKD [26] either modify the original training settings or modify the underlying student architecture, and so have been omitted from this evaluation.

6.4.2 Data efficient training for transformers

Transformers have emerged as a viable replacement for convolution-based neural networks (CNN) in visual learning tasks. Despite the promise of these models, their performance will suffer when there is insufficient training data available, such as in the case of ImageNet. DeiT [177] was the first to address this problem through the use of knowledge distillation. Although the authors show improved alignment with the teacher, we believe this fails to capture why less data is needed.

We posit that the distillation process encourages the student to learn layers which are "more" translational equivariant in attempt to match the teacher's underlying function. Although this is the principle that motivates using an ensemble of teacher models with different inductive biases [147], there is still no thorough demonstration on if the inductive biases are actually being transferred. In this section we attempt to address this gap by introducing a measure of equivariance. We show that applying our distillation principles to this task can achieve

significant improvements over state-of-the-art as a result of transferring more of the translational equivariance.

The results of these experiments are shown in table 6.5. We use the exact same training methodology as co-advice [147] and choose to use batch normalisation, a linear projection layer, and $\alpha = 4$ as the parameters for distillation. We observe a significant improvement over both DeiT and CivT when the capacity gap is large. However, as the capacity gap diminishes, and the student approaches the same performance as the teacher, this improvement is much less significant. Multiple factors, such as the soft maximum function and the batch normalisation, will be contributing to this observed result. However, the explanation is more concisely described by the fact that our distillation loss transfers more translational equivariance, which is discussed in the next section.

Cross architecture distillation can implicitly transfer inductive biases. CNNs use convolutions, which are spatially local operations, whereas transformers use self-attention, which are global operations. We expect that a benefit of this cross-architecture distillation setting is that the students learn to be "more" spatially equivariant in an attempt to match the teachers underlying function. It is this strong inductive bias that can reduce the amount of training data needed. A layer is translation equivariant if the following property holds:

$$\phi(T\mathbf{x}) = T\phi(\mathbf{x}) \tag{6.13}$$

In other words, if we take a translated input $T\mathbf{x}$ and pass it through a layer ϕ, the result should be equivalent to first applying ϕ to $\mathbf{x}$ and then performing the translation. A natural measure of equivariance can then be the difference between the left and right-hand side of this equation 6.13.

$$\mu_T(\phi) = \|\phi(T\mathbf{x}) - T\phi(\mathbf{x})\|_2^2 \tag{6.14}$$

We evaluate this measure on a block of self-attention layers by first removing the distillation and class tokens and then rolling the patch tokens to recover the spatial dimensions. This

Network	acc@1	Teacher	#params
RegNetY-160	82.6	none	84M
BiT-M R152x2	84.5	none	236M
DeiT-Ti	72.2	none	5M
CivT-Ti	74.9	ensemble	6M
DeiT-Ti⚗	74.5	regnety-160	6M
↳ 1000 epochs	76.6	regnety-160	6M
DearKD	74.8	regnety-160	6M
↳ 1000 epochs	77.0	regnety-160	6M
USKD	75.0	regnety-160	6M
Our Method	**77.2**	regnety-160	6M
ResNet-50	76.5	none	25M
FunMatch	80.3	bit-m r152x2	25M
↳ 9600 epochs	**82.8**	bit-m r152x2	25M
DeiT-S	79.8	none	22M
CivT-S	82.0	ensemble	22M
DeiT-S⚗	81.2	regnety-160	22M
↳ 1000 epochs	82.6	regnety-160	22M
DearKD	81.5	regnety-160	22M
↳ 1000 epochs	82.8	regnety-160	22M
USKD	80.8	regnety-160	22M
Our Method	**82.1**	regnety-160	22M

Table 6.5: Data-efficient training of transformers and CNNs on the ImageNet-1K dataset. Unless specified, all student models are trained for 300 epochs.

	Teacher	Student	AT	KD	CC	CRD	ReviewKD	Ours
acc@1	26.69	30.25	29.30	29.34	30.04	28.62	28.39	**28.13**
acc@5	8.58	10.93	10.00	10.12	10.83	9.51	9.42	**9.29**

Table 6.6: Top-1 and Top-5 error rates (%) on ImageNet. ResNet18 as student, ResNet34 as teacher.

operation can then be performed on the input and output tensors before applying a translation. Table 6.7 shows this measure of equivariance after training with and without distillation. In general, we observe that the distilled models do in fact learn to preserve spatial locality between feature maps, which aligns with the function matching perspective for distillation (see table 6.7). We also find that our distillation method can transfer a lot more of this equivariance property to the student. Although Co-advise does learn this spatial locality to some extent, it is much less significant than using our feature based distillation, despite both attaining a similar level of performance. Intermediate feature map losses may be able to transfer even more of this translational equivariance property, however, its usage may degrade the benefit of using a transformer in the first place. For example, although we observe that most self-attention blocks (trained using distillation) do preserve a lot of this spatial locality, there is still some global context between patch tokens that is still being preserved.

Network	$\mu_T(\phi)$
DeiT-S	1.52 ± 0.15
CivT-S	0.13 ± 0.05
Our Method	$\mathbf{0.04} \pm \mathbf{0.02}$

Table 6.7: Measure of translational equivariance of a DeiT-S transformer model trained with and without distillation. These results confirm that distillation can transfer explicit inductive biases from the teacher.

6.4.3 Object Detection on COCO

We extend the application of our method to object detection, whereby we employ a similar approach as used in the classification task by distilling the backbone output features of both the student and teacher networks. To evaluate the efficacy of our method, we conduct experiments on the widely-used COCO2017 dataset [102] under the same settings provided in ReviewKD. We then further demonstrate the applicability of our distillation principles on the more recent and efficient YOLOv5 model [217]. In both cases we show improved student performance on the downstream task, whereby competitive performance is achieved with ReviewKD despite being significantly simpler and cheaper to integrate into a given distillation pipeline. Our method even outperforms FPGI [183], which is directly designed for detection.

Model	mAP (50-95)	mAP 50
YOLOv5m (teacher)	64.1	45.4
YOLOv5s	56.8	37.4
↳ Our Method	**57.3**	**37.5**
	mAP	AP50
Faster R-CNN w/ R50 (teacher)	40.22	61.02
Faster R-CNN w/ MV2	29.47	48.87
↳ KD	30.13	50.28
↳ FitNet	30.20	49.80
↳ FPGI	31.16	50.68
↳ ReviewKD	**33.71**	**53.15**
↳ Our Method	32.92	52.96

Table 6.8: Object detection on COCO. (top) We report the standard COCO metric of mAP averaged over IOU thresholds in $[0.5 : 0.05 : 0.95]$ along with the standard PASCAL VOC's metric, which is the average mAP@0.5. (bottom) For the R-CNN results, we report the mAP and AP50 metrics to enable a consistent comparison with ReviewKD.

6.5 Conclusion

In this paper, we revisited the core underlying principles of knowledge distillation and have performed an extensive ablation on the most effective and scaleable components. In doing so, we have provided a new theoretical perspective for understanding these results through analyzing the projector training dynamics. By extending these principles to a wide range of tasks, we achieve competitive or improved performance to state-of-the-art across image classification, object detection, and data efficient training of transformers. Our proposed distillation recipe can significantly reduce the complexity and memory consumption of existing pipelines by avoiding the need to construct expensive relational object, many trainable layers, or enforcing very long training schedules. We further show improved performance for the large capacity gap settings and evidence for the distillation of explicit inductive biases from the teacher. Looking ahead to future research in this area, we expect to see the joint development of more sophisticated normalisation schemes and projection networks, which will encode more complex and informative features for the distillation process.

Code Reproducibility. To facilitate the reproducibility of results, we will release all the training code and pre-trained weights. The ImageNet experiments are also performed using

the popular *torchdistill* [116] framework, while the CIFAR100 and data-efficient training code is based on those provided by CRD [172] and co-advice [147] respectively.

6.6 Supplementary Material

Small capacity gap setting. We ablate the importance of the repulsive force in the low-capacity gap setting, whereby we observe a much smaller performance improvement (table 6.9). This observation is likely attributed to the teacher no longer providing any sufficiently more discriminative representations to aid in the knowledge distillation process.

Description	Repulsive force	acc@1
KD	$KL(p_s \parallel p_t)$	71.37
Mean-square	$\|z_s - z_t\|_2^2$	71.08
no projection		
Mean-square	$\|z_s - z_t\|_2^2$	71.53
Batch-normalised	$\|BN(z_s) - BN(z_t)\|_2^2$	71.34
Correlation	$\|BN(z_s) \cdot BN(z_t) - \mathbf{1}\|^2$	71.35
Higher-order	$\|BN(z_s) \cdot BN(z_t) - \mathbf{1}\|^4$	71.31
Soft maximum	$log \sum \|BN(z_s) \cdot BN(z_t) - \mathbf{1}\|^4$	**71.63**

Table 6.9: Ablating the importance in the choice of metric function using a ResNet34 and a ResNet18 student on the ImageNet dataset. The loss modifications are highlighted in red.

6.6.1 Measure of translational equivariance

In this section we provide more details on the proposed measure of translational equivariance and the motivation for its usage. In the cross inductive-bias distillation literature it is common to evaluate the effectiveness of a distillation pipeline through a measure of agreement between the student and the teacher [177, 147]. We argue that this metric fails to encapsulate why one distillation method is more data efficient than another. We hypothesise that the most effective cross architecture distillation methods are those that transfer most of the inductive biases. To verify this claim, we introduce a measure of equivariance and show that a good distillation loss does indeed minimise this. For the convolution $\rightarrow$ transformer distillation setting, the most appropriate choice of measure is a measure of translational equivariance. This is because the

teacher will have this equivariance explicitly enforced through the underlying convolutional layers, whereas the self-attention student will not. Recent works have shown that the self-attention layer can, in principle, learn the same operation as a convolution [95]. The authors then show that injecting this bias will improve the data-efficiency, however, we show that this is not necessary since the bias can be learned implicitly through distillation.

Self-attention layers are known to be permutation equivariant. However, with the use of additive positional encodings, this restriction can be relaxed and can enable these layers to learn any arbitrary sequence-to-sequence function [204]. When we apply transformers to the vision domain, the tokens are provided as non-overlapping patches of the image. Thus, to perform a spatial translation on the sequence of tokens, we must first roll the sequence back to have the $H \times W$ dimensions in the same way in which we unrolled it at the input. The exact details can be seen in algorithm 2. Using this defined measure, we find that non-distilled transformers are over 15× less equivariant than their distilled counterparts.

6.6.2 Few-shot distillation experiments

Due to the limited compute resources available, we were unable to perform some ablation experiments using the full ImageNet training data. To address this concern, and to avoid using a poor surrogate dataset (i.e. CIFAR100), we propose a more difficult few-shot distillation setting. In this setting, the models are still trained and evaluated on the large-scale ImageNet dataset, but with only a subset of the training data available. This results in the distillation objective being much more challenging. In all of these experiments we sample the same 20% of images from each class to ensure the overall class balance is maintained.

Listing 2: Translational equivariance measure

```
# x: Image representation B x N x C
# T: Spatial translation
# blk: Block of self-attention layers
def compute(x, T):
    # 14 x 14 patches
    h = 14
    w = 14
    b, n, c = x.shape

    # remove positional encodings
    xp = x[:, 2:]

    # roll token-dim into H x W dimensions
    xp = xp.transpose(1, 2).reshape(b, c, h, w)
    Tx = T(xp)

    # unroll H x W dimensions
    Tx = Tx.flatten(2).transpose(1,2)

    # add back the positional encodings
    Tx = torch.cat((x[:, 0:2], Tx), dim=1)

    # forward pass w/ and wo/ translation
    Fx = blk(x)
    FTx = blk(Tx)

    # remove position encoding
    Fxp = Fx[:, 2:]
    b, n, c = Fxp.shape

    Fxp = Fxp.transpose(1,2).reshape(b,c,h,w)
    TFxp = T(Fxp)
    TFxp = TFxp.flatten(2).transpose(1,2)

    # add back position encoding
    TFx = torch.cat((x[:, 0:2], TFxp), dim=1)

    return F.mse_loss(TFxp, FTx)
```

6.6.3 Model Architectures

In experiments, we use the following model architectures.

- Wide Residual Network (WRN) [206]: WRN-d-w represents wide ResNet with depth d and width factor w.
- resnet [69]: We use ResNet-d to represent CIFAR-style resnet with 3 groups of basic blocks, each with 16, 32, and 64 channels, respectively. In our experiments, resnet8x4 and resnet32x4 indicate a 4 times wider net- work (namely, with 64, 128, and 256 channels for each of the blocks).
- ResNet [69]: ResNet-d represents ImageNet-style ResNet with bottleneck blocks and more channels.

- MobileNetV2 [54]: In our experiments, we use a width multiplier of 0.5.
- vgg [163]: The vgg networks used in our experiments are adapted from their original ImageNet counterpart.
- ShuffleNetV1 [209], ShuffleNetV2 [114]: ShuffleNets are proposed for efficient training and we adapt them to input of size 32x32.

Implementation Details. Both the ImageNet and CIFAR experiments follow the same training procedures as CRD [172]. However, for completeness, we choose to restate the details here.

For the CIFAR-100 experiments we use the SGD optimizer with an initial learning rate of 0.05, and with a decay of 0.1 every 30 epochs after the first 150 epochs until the last 240 epoch. For MobileNetV2, ShuffleNetV1 and ShuffleNetV2, we use a learning rate of 0.01 as this learning rate is optimal for these models in a grid search, while 0.05 is optimal for other models.

For the ImageNet experiments we train for 100 epochs with a 0.1 decay at epochs 30, 60, and 90. Further details are provided in the *torchdistill* library.

Finally, for the data-efficient training of transformers, we use the same training schedule as DeIT [177]. This training pipeline uses the AdamW optimizer with Mixup, Cutmix and RandAugment. We choose a batch size of 512 on a single GPU.

Chapter 7

Conclusion

7.1 Summary of Contributions

The goal of this PhD was to improve our understanding of distillation and push the boundaries of model compression in the field of computer vision and machine learning in general. The initial work in this thesis proposed a novel low-rank decomposition for descriptor learning that was motivated and derived from an emergent property in the pre-trained weights. Subsequent work introduced a novel decomposition of convolutional weights alongside a mathematical re-formulation to enable an efficient CUDA implementation on device. By coupling this decomposition with standard pruning techniques, we could learn an appropriate rank for this decomposition in each layer. Later works saw the generalisation of slimmable networks [199] to enable arbitrary filter-wise pruning masks, thus improving the diversity of subnetworks and subsequently the attainable downstream accuracy. This was then followed be a focus on knowledge distillation, whereby two perspectives are proposed. The first being from an information-theoretic background leading to state-of-the-art across all standard classification benchmarks, while being significantly cheaper to adopt than other proposed in the literature. The follow up work explored the training dynamics throughout the distillation procedure and in doing so led to a simple method for distilling between transformers and CNNs.

7.2 Limitations

Although we do provide extensive experiments showing the generality of the proposed approaches given in this thesis, there are some obvious limitations. One limitation is applicable to much of the empirical research in deep learning and that is on insufficient model guarantees. These guarantees can be related to the robustness and failure modes found in practice. Unfortunately, it is very difficult to provide guarantees on any particular deep learning method without imposing strict constraints which in turn make the guarantees less applicable in practice. We are hoping that the divide between theory and practice will shrink in the near future and subsequently aid in improving both the interpretability and robustness of models being deployed in the wild.

7.3 Future Work

One immediate consequence of the later work in this PhD is on the topic of cross-architecture distillation. There are still a lot of open questions about the design of projector networks for distillation and how the differing inductive biases of the student-teacher pair comes into play. By diving more into this theory, some novel applications may emerge that can address some very relevant problems in the field of machine learning. The most obvious of which is on reducing the amount of data needed to train and deploy large language model. Models such as GPT-4 require a very large corpus of training data, which is inaccessible to most research or industry labs. Knowledge distillation, and specifically the work that may build upon this thesis, can directly begin to address and bridge this gap to make the training and deployment of these models more accessible.

7.3.1 Data-Efficient Training

Geometric deep learning [21, 22] offers a framework for designing neural architectures that exploit the inherent symmetries present in the data. By explicitly incorporating these symmetries

into the network structure, we can enhance the data efficiency of the models. This enhanced efficiency is also observed when transferring knowledge across different architectural designs.

When training deep learning models, one of the key challenges is to obtaining accurate results with limited data, especially in domains like medical imaging where data scarcity is prevalent. Geometric deep learning provides a way to mitigate this challenge. By enforcing these symmetries into underlying model architecture, we can effectively capture and exploit the underlying structural patterns, leading to more efficient and accurate learning.

Moreover, when distilling knowledge from one model to another, the benefits of geometric deep learning become even more pronounced. The ability to transfer knowledge across architectures is not limited to explicit approaches alone. In fact, a combination of both explicit and implicit approaches can be employed to maximize the advantages of data-efficient training. By combining the explicit enforcement of symmetries with the implicit regularisation from representation distillation, we can create models that are not only capable of capturing intricate patterns but also perform optimally with limited data resources.

7.3.2 Multi-Modality Models

Multi-modality models can understand information from multiple modalities, such as text, images, audio, and video. In the context of robotics, incorporating diverse sources of sensory information can enable models to have a more unified and complete understanding of the environment in a similar way to which humans perceive and interact with each other. Most of the work developed in this thesis has focused on a single modality and tasks, but can very naturally be extended to train more general and efficient models. Recent work has shown the promise of utilising information from multiple modalities [78, 109, 8] and tasks [10], but a more general approach, and likely a very effective approach, in the context of knowledge distillation has yet to be explored.

Bibliography

[1] Martin Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". In: *arXiv preprint* (2016).

[2] Madhu Advani, Artemy Kolchinsky, and Brendan D Tracey. "On the information bottleneck theory of deep learning". In: *ICLR* (2018).

[3] Sungsoo Ahn et al. "Variational information distillation for knowledge transfer". In: *CVPR* (2019).

[4] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: *BMVC* (2013).

[5] Zeyuan Allen-Zhu and Yuanzhi Li. "Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning". In: (Dec. 2020). URL: `http://arxiv.org/abs/2012.09816`.

[6] Relja Arandjelovic. "Three Things Everyone Should Know to Improve Object Retrieval". In: *CVPR* (2012).

[7] P A Y Attention, T O Snapshots, and O F Pruning. "Paying more Attention to Snapshots of Iterative Pruning : Improving Model Compression via Ensemble Distillation". In: (2020).

[8] Dylan Auty and Krystian Mikolajczyk. "ObjCAViT: Improving Monocular Depth Estimation Using Natural Language Models And Image-Object Cross-Attention". In: *arXiv preprint* (2022).

[9] Dylan Auty et al. "Learning to Project for Cross-Task Knowledge Distillation". In: *arXiv preprint* (2024).

[10] Dylan Auty et al. “Learning to Project for Cross-Task Knowledge Distillation”. In: *arXiv preprint* (2024).

[11] Mart Van Baalen et al. “Bayesian Bits : Unifying Quantization and Pruning”. In: NeurIPS (2020).

[12] Davide Bacciu and Danilo P. Mandic. “Tensor Decompositions in Deep Learning”. In: (Feb. 2020).

[13] Vassileios Balntas et al. “HPatches: A benchmark and evaluation of handcrafted and learned local descriptors”. In: *CVPR*. 2017.

[14] Vassileios Balntas et al. “Learning local feature descriptors with triplets and shallow convolutional neural networks”. In: *BMVC* (2017).

[15] Adrien Bardes, Jean Ponce, and Yann LeCun. “VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning”. In: *ICLR* (2022).

[16] Adrien Bardes, Jean Ponce, and Yann LeCun. “VICRegL: Self-Supervised Learning of Local Visual Features”. In: *NeurIPS* (Oct. 2022).

[17] H Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* (2008).

[18] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation”. In: *arXiv preprint* (2013).

[19] Lucas Beyer et al. “Knowledge distillation: A good teacher is patient and consistent”. In: *CVPR* (2022).

[20] Rajendra Bhati. “Infinitely Divisible Matrices”. In: *Transactions of the American Mathematical Society* (1969).

[21] Michael M Bronstein et al. *Geometric deep learning: going beyond Euclidean data*. Tech. rep.

[22] Michael M. Bronstein et al. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: (Apr. 2021). URL: `http://arxiv.org/abs/2104.13478`.

[23] Adrian Bulat and Georgios Tzimiropoulos. “XNOR-Net++: Improved Binary Neural Networks”. In: *BMVC* (Sept. 2019).

[24] Michael Calonder et al. “Binary Robust Independent Elementary Features”. In: *ECCV* (2010).

[25] Fabio M. Carlucci et al. “Domain Generalization by Solving Jigsaw Puzzles”. In: *CVPR* (2019).

[26] Defang Chen et al. “Knowledge Distillation with the Reused Teacher Classifier”. In: *CVPR* (2022).

[27] Hanting Chen et al. “AdderNet: Do We Really Need Multiplications in Deep Learning?” In: *CVPR* (2020).

[28] Hong Yen Chen and Chung Yen Su. “An Enhanced Hybrid MobileNet”. In: *iCAST* (2018).

[29] Liqun Chen et al. “Wasserstein Contrastive Representation Distillation”. In: *CVPR* (2020).

[30] Pengguang Chen et al. “Distilling Knowledge via Knowledge Review”. In: *CVPR* (2021).

[31] Tianyi Chen et al. “Only Train Once: A One-Shot Neural Network Training And Pruning Framework”. In: *NeurIPS* (2021).

[32] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *ICML* (2020).

[33] Xiaohan Chen et al. “The Elastic Lottery Ticket Hypothesis”. In: *NeurIPS* ().

[34] Xinlei Chen and Kaiming He. “Exploring Simple Siamese Representation Learning”. In: *CVPR* (2021).

[35] Xinlei Chen, Saining Xie, and Kaiming He. “An Empirical Study of Training Self-Supervised Vision Transformers”. In: *ICCV* (Apr. 2021).

[36] Xinlei Chen et al. “Improved Baselines with Momentum Contrastive Learning”. In: *arXiv preprint* (Mar. 2020).

[37] Yudong Chen et al. “Improved Feature Distillation via Projector Ensemble”. In: *NeurIPS* (2022).

[38] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *CVPR* (2017).

[39] Krzysztof Choromanski et al. “Rethinking Attention with Performers”. In: *ICLR* (Sept. 2020).

[40] Adam Coates, Honglak Lee, and Andrew Y. Ng. “An analysis of single-layer networks in unsupervised feature learning”. In: *JMLR* (2011). ISSN: 15324435.

[41] Taco S. Cohen and Max Welling. “Group equivariant convolutional networks”. In: *ICML* (2016).

[42] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *CVPR* (2016).

[43] Matthieu Courbariaux et al. “Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or -1”. In: *arXiv preprint* (2016).

[44] Ekin D. Cubuk et al. “Randaugment: Practical automated data augmentation with a reduced search space”. In: *CVPR Workshop* (2020).

[45] Daniel Detone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-supervised interest point detection and description”. In: *ICPR* (2018).

[46] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (2019).

[47] Ruizhou Ding et al. “Regularizing activation distribution for training binarized deep networks”. In: *CVPR* (2019).

[48] Xiaohan Ding et al. “Approximated Oracle Filter Pruning for Destructive CNN Width Optimization”. In: *ICML* (May 2019).

[49] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. “Unsupervised Visual Representation Learning by Context Prediction”. In: *ICCV* (May 2015).

[50] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. "Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon". In: *NeurIPS* (2017).

[51] Alaaeldin El-Nouby et al. "XCiT: Cross-Covariance Image Transformers". In: *NeurIPS* (2021).

[52] Aleksandr Ermolov et al. "Whitening for Self-Supervised Representation Learning". In: *ICML* (July 2020).

[53] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the Manifold Hypothesis". In: *arXiv preprint* (Oct. 2013).

[54] Michael H. Fox, Kyungmee Kim, and David Ehrenkrantz. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *CVPR* (2018).

[55] Jonathan Frankle et al. "Linear mode connectivity and the lottery ticket hypothesis". In: *ICML* (2020).

[56] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *ICLR* (Mar. 2018).

[57] Ruihao Gong et al. "Differentiable soft quantization: Bridging full-precision and low-bit neural networks". In: *ICCV* (2019). ISSN: 15505499.

[58] Ariel Gordon et al. "MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks". In: *CVPR* (2018).

[59] Jean Bastien Grill et al. "Bootstrap your own latent a new approach to self-supervised learning". In: *NeurIPS* (2020).

[60] Jia Guo et al. "Reducing the Teacher-Student Gap via Spherical Knowledge Distillation". In: *arXiv preprint* (2020).

[61] Shaopeng Guo et al. "DMCP: Differentiable Markov Channel Pruning for Neural Networks". In: *CVPR* (2020).

[62] Kai Han et al. "GhostNet: More Features from Cheap Operations". In: *CVPR* (2019).

[63] Song Han, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *ICLR* (2015).

[64] Babak Hassibi and David G Stork. "Second order derivatives for network pruning: Optimal Brain Surgeon". In: *NeurIPS* (1993).

[65] Kohei Hayashi et al. "Einconv: Exploring Unexplored Tensor Decompositions for Convolutional Neural Networks". In: *NeurIPS 2019* (2019).

[66] Bobby He and Mete Ozay. "Feature Kernel Distillation". In: *ICLR* (2022).

[67] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *CVPR* (Nov. 2022).

[68] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *CVPR* (Nov. 2020).

[69] Kaiming He et al. "ResNet - Deep Residual Learning for Image Recognition". In: *CVPR* (2015).

[70] Yang He et al. "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration". In: *CVPR* (2018).

[71] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel Pruning for Accelerating Very Deep Neural Networks". In: *ICCV*. 2017.

[72] Zhiqiang He et al. "Filter Pruning via Feature Discrimination in Deep Neural Networks". In: *ECCV*. 2022.

[73] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: *NeurIPS* (2015).

[74] Geoffrey Hinton et al. "On the importance of initialization and momentum in deep learning". In: *ICML* (2013).

[75] Frank L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *JMP* (2015).

[76] Andrew Howard et al. "Searching for MobileNetV3". In: *ICCV* (2019).

[77] Andrew G Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: *arXiv preprint* (2017).

[78] Lukas Hoyer et al. *Three Ways to Improve Semantic Segmentation with Self-Supervised Depth Estimation*. 2021.

[79] Hengyuan Hu et al. “Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures”. In: *arXiv* (2016).

[80] Qiangui Huang et al. “Learning to prune filters in convolutional neural networks”. In: *WACV* (2018).

[81] Zehao Huang and Naiyan Wang. “Like What You Like: Knowledge Distill via Neuron Selectivity Transfer”. In: *arXiv preprint* (2017).

[82] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. “Speeding up convolutional neural networks with low rank expansions”. In: *BMVC*. 2014.

[83] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *ICLR* (2017).

[84] Donggyu Joo et al. “Linearly Replaceable Filters for Deep Network Channel Pruning”. In: *AAAI*. 2021.

[85] Chaitanya K. Joshi et al. “On Representation Knowledge Distillation for Graph Neural Networks”. In: *arXiv preprint* (Nov. 2021).

[86] Andrew Kerr, Dan Campbell, and Mark Richards. “QR decomposition on GPUs”. In: *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, GPGPU-2* (2009).

[87] Jangho Kim, Seong Uk Park, and Nojun Kwak. “Paraphrasing complex network: Network compression via factor transfer”. In: *NeurIPS* (2018).

[88] Yong-Deok Kim et al. “Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications”. In: *ICLR* (2015).

[89] Benedikt Kolbeinsson and Krystian Mikolajczyk. “DDOS: The Drone Depth and Obstacle Segmentation Dataset”. In: *arXiv preprint* (2023).

[90] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: (2009).

[91] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NeurIPS* (2012).

[92] Yann Lecun. “Optimal Brain Damage”. In: *NeurIPS* (1990).

[93] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H.S. Torr. “SnIP: Single-shot network pruning based on connection sensitivity”. In: *ICLR*. 2019.

[94] Hao Li et al. “Pruning Filters For Efficient Convnets”. In: *ICLR* (2017).

[95] Shanda Li et al. “Can Vision Transformers Perform Convolution?” In: *arXiv preprint* (Nov. 2021).

[96] Yawei Li et al. “Group Sparsity: The Hinge Between Filter Pruning and Decomposition for Network Compression”. In: *CVPR* (2020).

[97] Tailin Liang et al. “Pruning and Quantization for Deep Neural Network Acceleration: A Survey”. In: *arXiv preprint* (Jan. 2021).

[98] Mingbao Lin et al. “Channel Pruning via Automatic Structure Search”. In: *IJCAI*. Jan. 2020.

[99] Mingbao Lin et al. “Filter Sketch for Network Pruning”. In: *arXiv* (2019).

[100] Mingbao Lin et al. “HRank: Filter Pruning using High-Rank Feature Map”. In: *CVPR* (2020).

[101] Mingbao Lin et al. “Rotated binary neural network”. In: *NeurIPS* (2020).

[102] Tsung Yi Lin et al. “Microsoft COCO: Common objects in context”. In: *ECCV* (2014).

[103] Jiayi Liu et al. “Pruning Algorithms to Accelerate Convolutional Neural Networks for Edge Applications: A Survey”. In: *arXiv preprint* (May 2020).

[104] Li Liu et al. “Exploring Inter-Channel Correlation for Diversity-preserved Knowledge Distillation”. In: *ICCV* (2021).

[105] Yifan Liu et al. “Structured Knowledge Distillation for Semantic Segmentation”. In: *CVPR* (2019).

[106] Yuchen Liu, David Wentzlaff, and S. Y. Kung. "Rethinking Class-Discrimination Based CNN Channel Pruning". In: *arXiv preprint*. Apr. 2020.

[107] Zechun Liu and Tim Kwang-ting Cheng. "MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning". In: *ICCV*. 2019.

[108] Zhuang Liu et al. "Rethinking The Value Of Network Pruning". In: *ICLR* (2019).

[109] Adrian Lopez-Rodriguez and Krystian Mikolajczyk. *DESC: Domain Adaptation for Depth Estimation via Semantic Consistency*. 2020.

[110] Christos Louizos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through L_0 Regularization". In: *ICLR* (Dec. 2017).

[111] David G Lowe. *SIFT - Distinctive Image Features from Scale-Invariant Keypoints*. Tech. rep. 2004.

[112] Jian Hao Luo, Jianxin Wu, and Weiyao Lin. "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression". In: *ICCV*. 2017.

[113] Zixin Luo et al. "GeoDesc: Learning local descriptors by integrating geometry constraints". In: *ECCV* (2018).

[114] Ningning Ma et al. "Shufflenet V2: Practical guidelines for efficient cnn architecture design". In: *Lecture Notes in Computer Science*. 2018.

[115] Yuchen Ma, Yanbei Chen, and Zeynep Akata. "Distilling Knowledge from Self-Supervised Teacher by Embedding Graph Alignment". In: *BMVC* (Nov. 2022).

[116] Yoshitomo Matsubara. "torchdistill : A Modular, Configuration-Driven Framework for Knowledge Distillation". In: *ICPR Workshop on Reproducible Research in Pattern Recognition* (2020).

[117] Roy Miles, Ismail Elezi, and Jiankang Deng. "V_kD: Improving Knowledge Distillation using Orthogonal Projections". In: *CVPR* (2024).

[118] Roy Miles and Krystian Mikolajczyk. "Cascaded channel pruning using hierarchical self-distillation". In: *BMVC* (2020).

[119] Roy Miles and Krystian Mikolajczyk. "Compression of descriptor models for mobile applications". In: *ICASSP* (2021).

[120] Roy Miles and Krystian Mikolajczyk. "Reconstructing Pruned Filters using Cheap Spatial Transformations". In: *ICCV Workshop on Resource Efficient Deep Learning for Computer Vision* (2023).

[121] Roy Miles and Krystian Mikolajczyk. "Understanding the Role of the Projector in Knowledge Distillation". In: *AAAI* (2024).

[122] Roy Miles, Adrian Lopez Rodriguez, and Krystian Mikolajczyk. "Information Theoretic Representation Distillation". In: *BMVC* (Dec. 2022).

[123] Roy Miles et al. "MobileVOS: Real-Time Video Object Segmentation Contrastive Learning meets Knowledge Distillation". In: *CVPR* (Mar. 2023).

[124] Ilya Mironov. "Rényi Differential Privacy". In: *Proceedings - IEEE Computer Security Foundations Symposium* (2017). ISSN: 19401434.

[125] Seyed-Iman Mirzadeh et al. "Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher". In: *AAAI* (2020).

[126] Anastasiya Mishchuk et al. "Working hard to know your neighbor's margins: Local descriptor learning loss". In: *NeurIPS* (2017).

[127] Pavlo Molchanov et al. "Importance Estimation for Neural Network Pruning". In: *CVPR* (2019).

[128] Shinichi Nakajima et al. "Global Analytic Solution of Fully-observed Variational Bayesian". In: *JMLR* (2013).

[129] K L Navaneet et al. "SimReg: Regression as a Simple Yet Effective Tool for Self-supervised Knowledge Distillation". In: *BMVC* (Jan. 2021).

[130] Michal Nazarczuk and Krystian Mikolajczyk. "SHOP-VRB: A Visual Reasoning Benchmark for Object Perception". In: *ICRA* (2020).

[131] Kien Nguyen et al. "Iris Recognition with Off-the-Shelf CNN Features: A Deep Learning Perspective". In: *IEEE Access* (Dec. 2017).

[132] Alexander Novikov et al. “Tensorizing Neural Networks”. In: *NeurIPS* (2015).

[133] Junghun Oh et al. “Batch Normalization Tells You Which Filter is Important”. In: *WACV* (2022).

[134] Yuki Ono et al. “LF-Net: Learning local features from images”. In: *NeurIPS* (2018).

[135] Wonpyo Park et al. “Relational Knowledge Distillation”. In: *CVPR* (2019).

[136] Nikolaos Passalis and Anastasios Tefas. “Learning Deep Representations with Probabilistic Knowledge Transfer”. In: *ECCV* (2018).

[137] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: (2017).

[138] Baoyun Peng et al. “Correlation congruence for knowledge distillation”. In: *CVPR* (2019).

[139] Qi Qian, Hao Li, and Juhua Hu. “Efficient Kernel Transfer in Knowledge Distillation”. In: *arXiv* (2020).

[140] Haotong Qin et al. “Forward and Backward Information Retention for Accurate Binary Neural Networks”. In: *CVPR* (2020).

[141] Zhuwei Qin et al. “CAPTOR: A Class Adaptive Filter Pruning Framework for Convolutional Neural Networks in Mobile Applications”. In: *ASPDAC*. 2019.

[142] Ilija Radosavovic et al. “Data Distillation: Towards Omni-Supervised Learning”. In: *CVPR* (2018).

[143] Pranav Rajpurkar et al. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text”. In: (2016).

[144] Vivek Ramanujan et al. “What’s Hidden in a Randomly Weighted Neural Network?” In: *CVPR* (2019).

[145] Mohammad Rastegari et al. “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks”. In: *ECCV* (2016).

[146] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, faster, stronger”. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. 2017. ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.690.

[147] Sucheng Ren et al. "Co-advise: Cross Inductive Bias Distillation". In: *CVPR* (2022).

[148] Alfréd Rényi. "On Measures of Entropy and Information". In: *Proceedings of the fourth Berkeley Symposium on Mathematics, Statistics and Probability* (1960).

[149] Adriana Romero et al. "FitNets: Hints For Thin Deep Nets". In: *ICLR* (2015).

[150] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI* (May 2015).

[151] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection". In: *ECCV*. 2006.

[152] Xiaofeng Ruan et al. "DPFPS: Dynamic and Progressive Filter Pruning for Compressing Convolutional Neural Networks from Scratch". In: *AAAI*. 2021.

[153] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *IJCV* (2014).

[154] Noveen Sachdeva and Julian McAuley. "Data Distillation: A Survey". In: (Jan. 2023). URL: `http://arxiv.org/abs/2301.04272`.

[155] Luis G. Sanchez Giraldo and Jose C. Principe. "Information theoretic learning with infinitely divisible kernels". In: *ICLR* (2013).

[156] Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C. Principe. "Measures of entropy from data using infinitely divisible Kernels". In: *IEEE Transactions on Information Theory* (2015).

[157] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing* (2019).

[158] B. H. Shekar et al. "Face recognition using kernel entropy component analysis". In: *Neurocomputing* (2011).

[159] Nicholas D. Sidiropoulos et al. "Tensor Decomposition for Signal Processing and Machine Learning". In: *IEEE Transactions on Signal Processing* (2017). ISSN: 1053587X. DOI: `10.1109/TSP.2017.2690524`.

[160] Laurent Sifre and Stephane Mallat. “Rigid-Motion Scattering For Image Classification”. PhD thesis. 2014.

[161] Vin de Silva and Lek-Heng Lim. “Tensor rank and the ill-posedness of the best low-rank approximation problem”. In: *SIAM* (2006).

[162] B.W. Silverman. “Density estimation for statistics and data analysis”. In: *Monographs on Statistics and Applied Probability* (1986).

[163] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks For Large-scale Image Recognition”. In: *ICLR* (2015).

[164] Brijraj Singh, Durga Toshniwal, and Sharan Kumar Allur. “Shunt connection: An intelligent skipping of contiguous blocks for optimizing MobileNet-V2”. In: *Neural Networks* (2019).

[165] Arvind Subramaniam and Avinash Sharma. “N2NSkip: Learning Highly Sparse Networks using Neuron-to-Neuron Skip Connections”. In: *BMVC* (2020).

[166] Christian Szegedy et al. “GoogLeNet/Inception - Going deeper with convolutions”. In: *CVPR*. 2015.

[167] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking model scaling for convolutional neural networks”. In: *ICML* (2019).

[168] Mingxing Tan et al. “MnasNet: Platform-Aware Neural Architecture Search for Mobile”. In: *CVPR* (2018).

[169] Yehui Tang et al. “SCOP: Scientific Control for Reliable Neural Network Pruning”. In: *NeurIPS*. Oct. 2020.

[170] “The lottery ticket hypothesis : Finding sparse, Trainable neural Networks”. In: *ICLR* (2019).

[171] Yonglong Tian, Olivier J. Henaff, and Aaron van den Oord. “Divide and Contrast: Self-supervised Learning from Uncurated Data”. In: *ICCV* (May 2021).

[172] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive representation distillation”. In: *ICLR* (2019).

[173] Yuandong Tian, Xinlei Chen, and Surya Ganguli. "Understanding self-supervised Learning Dynamics without Contrastive Pairs". In: *ICML* (2021).

[174] Yurun Tian, Bin Fan, and Fuchao Wu. "L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space". In: *CVPR* (2017).

[175] Yurun Tian et al. "Sosnet: Second order similarity regularization for local descriptor learning". In: *CVPR* (2019).

[176] Naftali Tishby. "Deep Learning and the Information Bottleneck Principle". In: *IEEE Information Theory Workshop (ITW)* (2015).

[177] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *PMLR* (2021).

[178] Linh Tran et al. "Hydra: Preserving Ensemble Diversity for Model Distillation". In: (2020). URL: `http://arxiv.org/abs/2001.04694`.

[179] Ledyard R Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* (1966).

[180] Fred Tung and Greg Mori. "Similarity-preserving knowledge distillation". In: *ICCV* (2019).

[181] Yaman Umuroglu et al. "FINN: A framework for fast, scalable binarized neural network inference". In: *FPGA 2017*. 2017.

[182] Ashish Vaswani et al. "Attention is all you need". In: *NeurIPS* (2017).

[183] Tao Wang et al. "Distilling Object Detectors with Fine-grained Feature Imitation". In: *CVPR* (June 2019).

[184] Wenqi Wang, Brian Eriksson, and Wenlin Wang. "Wide Compression : Tensor Ring Nets". In: *CVPR* (2018).

[185] Wenxiao Wang et al. "COP: Customized deep model compression via regularized correlation-based filter-level pruning". In: *IJCAI* (2019).

[186] Wei Wen et al. "Learning Structured Sparsity in Deep Neural Networks". In: *NeurIPS* (2016).

[187] Paul L. Williams and Randall D. Beer. "Nonnegative Decomposition of Multivariate Information". In: (2010).

[188] Tong Xiao et al. "Sharing Attention Weights for Fast Transformer". In: *IJCAI* (June 2019).

[189] Guodong Xu et al. "Knowledge Distillation Meets Self-supervision". In: *ECCV* (2020).

[190] Keyulu Xu et al. "How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks". In: *ICLR* (2020).

[191] Zihan Xu et al. "ReCU: Reviving the Dead Weights in Binary Neural Networks". In: *ICCV* (2021).

[192] Le Ya and Yang Xuan. "Tiny imagenet visual recognition challenge". In: (2015).

[193] Chuanguang Yang et al. "Hierarchical Self-supervised Augmented Knowledge Distillation". In: *IJCAI* (2021).

[194] Zhaohui Yang et al. "Searching for low-bit weights in quantized neural networks". In: *NeurIPS* (2020). ISSN: 10495258.

[195] Ziqing Yang et al. "TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing". In: *ACL* (2020).

[196] Junho Yim. "A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning". In: *CVPR* (2017).

[197] Jiahui Yu and Thomas Huang. "AutoSlim: Towards One-Shot Architecture Search for Channel Numbers". In: *arXiv preprint* (2019).

[198] Jiahui Yu and Thomas Huang. "Universally Slimmable Networks and Improved Training Techniques". In: *ICCV* (2019).

[199] Jiahui Yu et al. "Slimmable Neural Networks". In: *ICLR* (2018).

[200] Ruichi Yu et al. "NISP: Pruning Networks Using Neuron Importance Score Propagation". In: *CVPR* (2018).

[201] Shujian Yu and José C. Príncipe. "Understanding autoencoders with information theoretic concepts". In: *Neural Networks* (2019).

[202] Shujian Yu et al. "Understanding Convolutional Neural Networks With Information Theory: An Initial Exploration". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[203] Xi Yu, Shujian Yu, and José C. Príncipe. "Deep deterministic information bottleneck with matrix-based entropy functional". In: *ICASSP* (2021).

[204] Chulhee Yun et al. "Are Transformers universal approximators of sequence-to-sequence functions?" In: *ICLR* (2020).

[205] Sergey Zagoruyko and Nikos Komodakis. "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer". In: *ICLR*. 2019.

[206] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *BMVC* (2016).

[207] Jure Zbontar et al. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *ICML* (2021).

[208] Richard Zhang, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization". In: Mar. 2016.

[209] Xiangyu Zhang, Xinyu Zhou, and Mengxiao Lin. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *CVPR* (2018).

[210] Youcai Zhang et al. "Prime-Aware Adaptive Distillation". In: *ECCV* (2020).

[211] Borui Zhao, Renjie Song, and Yiyu Qiu. "Decoupled Knowledge Distillation". In: *CVPR* (2022).

[212] Chenglong Zhao et al. "Variational Convolutional Neural Network Pruning". In: *CVPR* (2019).

[213] Hattie Zhou et al. "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask". In: *NeurIPS* NeurIPS (2019).

[214] Shuchang Zhou et al. "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients". In: *arXiv preprint* (2016).

[215] Yuefu Zhou et al. "Accelerate CNN via Recursive Bayesian Pruning". In: *ICCV* (2018).

[216] Jinguo Zhu et al. "Complementary Relation Contrastive Distillation". In: *CVPR* (2021).

[217] Xingkui Zhu et al. “TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios”. In: *VisDrone ICCV workshop* (Aug. 2021).

[218] Tao Zhuang et al. “Neuron-level Structured Pruning using Polarization Regularizer”. In: *NeurIPS*. 2022.

[219] Zhuangwei Zhuang et al. “Discrimination-aware Channel Pruning for Deep Neural Networks”. In: *NeurIPS* (2018).