Cascaded channel pruning using hierarchical self-distillation

Roy Miles
r.miles18@imperial.ac.uk
Krystian Mikolajczyk
k.mikolajczyk@imperial.ac.uk

Imperial College London
Department of Electrical and Electronic Engineering
London, UK

Abstract

In this paper, 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.

1 Introduction

Convolutional neural networks (CNNs) have demonstrated state-of-the-art results on a range of computer vision tasks, such as image classification [8, 28], depth-estimation [4, 6], and object detection [5, 21]. Despite their success, these models rely on a large number of parameters, which limits their deployment on resource-constrained devices and motivates the need for model compression techniques. 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 degradation in task accuracy. However, although pruning individual weights [17, 18] 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 [23]. To enable the contribution of

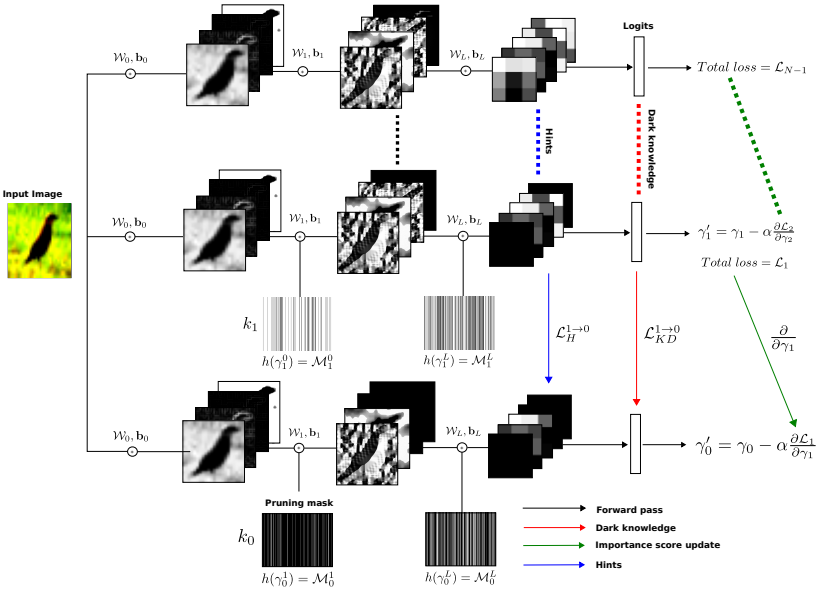


Figure 1: Proposed hierarchical self-distillation strategy for channel pruning. Each of the models are jointly trained with shared convolutional weights but with independent binary masks, batch normalisation layers, and classification layers. The lesser constrained models provide knowledge distillation and importance score gradients down the hierarchy. The frozen teacher for model T_N has been omitted for clarity.

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 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 [28] 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 1). We also consider ResNet50 [8] on the ImageNet2012 classification 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 2).

2 Related Work

We group the relevant works into three main categories that include filter pruning, knowledge distillation, and efficient architectures.

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 [19], while [10] performed the pruning with channel selection based on a LASSO-regression. In [22] the pruning of a given layer is guided by subsequent layer statistics. Similarly, NISP [34] formulates the pruning problem as a binary integer program by which the error-propagation across layers is considered. Discrimination-aware losses were proposed by [38] 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 [36, 37]. 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 [32] proposed the training of a single slimmable network that is iteratively slimmed and evaluated to ensure minimal accuracy drop, while MorphNet [7] optimizes the model using shrinking and cycle phases. The inefficient filters are then removed using sparsifying regularizers.

In a different approach [25] 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 [11] 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 [26] 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 [23] 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 [33] 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 [13] 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 [29], and all the MobileNet variants [8, 12, 13]. MobileNetV2 [8] proposed a linear bottleneck and an inverted residual connection to enforce feature re-use. ShuffleNet [35] built upon this idea by using group pointwise convolutions followed by a shuffle operation for enabling cross-group in-

formation flow. EfficientNets [30] 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. We demonstrate this idea in our experiments using the popular MobileNetV1 architecture in section 4.2.

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.

3.1 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:

$$\mathcal{L}(f(\mathcal{X}, \mathcal{W} \cdot \mathcal{M})) \approx \mathcal{L}(f(\mathcal{X}, \mathcal{W})), \quad \frac{\|\mathcal{M}\|_0}{|\mathcal{W}|} = p \tag{1}$$

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.

3.2 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 [9, 17] 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 [2, 15] of its gradients. We also use the derivation from [25] for the γ update rule, which we describe below. Consider the following masked convolutional layer:

$$\mathcal{Y}_{h,w,n} = \mathcal{X} * (\mathcal{W} \cdot \mathcal{M}) = \sum_{k_h,k_w}^{K} \sum_{i}^{C} \mathcal{W}_{k_h,k_w,i,n} \cdot \mathcal{X}_{h',w',i} \cdot h(\gamma_{k_h,k_w,i,n}) \tag{2}$$

Where $*$ is the convolution operation and $\cdot$ is 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 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.

$$\frac{\partial\mathcal{Y}}{\partial\gamma}_n = \sum_{w}^{W}\sum_{h}^{H}\sum_{k_h,k_w}^{K}\sum_{i}^{C}\mathcal{W}_{k_h,k_w,i,n}\cdot\mathcal{X}_{h',w',i} = \sum_{w}^{W}\sum_{h}^{H}\mathcal{W}*\mathcal{X} \tag{3}$$

The final update[1] for γ is given as follows:

$$\gamma' = \gamma - \alpha\sum_{w=1}^{W}\sum_{h=1}^{H}\frac{\partial\mathcal{L}}{\partial\mathcal{Y}}\cdot(\mathcal{X}*\mathcal{W}) \tag{4}$$

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.

3.3 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 $k_{N-1} = 1.0$. 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). 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*. [33] for networks that are executable at different widths, while the independent classifications layers are needed to enable sufficient diversity between the models. Figure 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 [33]. 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

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

can use the importance score gradients from the teacher to update the student. The proposed modification can be reflected in the γ update rule:

$$\gamma_i' = \gamma_i - \alpha \sum_{w=1}^{W} \sum_{h=1}^{H} \frac{\partial \mathcal{L}_{i+1}}{\partial \mathcal{Y}_{i+1}} \cdot (\mathcal{X}_{i+1} * \mathcal{W}) \tag{5}$$

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 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 5.1.

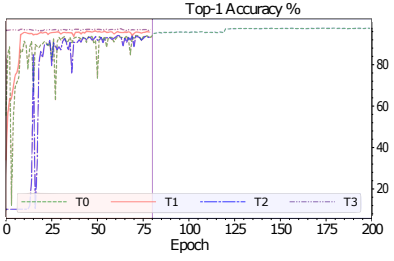
3.4 Knowledge distillation

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 [11]. 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 [26] provide teacher-student supervision for the intermediate representations. For this, we consider the simple reconstruction error term [38] 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 Experiments

In this section, we emperically validate the cascaded pruning approach on CIFAR10, CIFAR100 and ImageNet 2012 classification tasks, in which we consider the VGG16 [28], MobileNetV1 [13] and ResNet50 [8] 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%
Ours-0.2	2.95M	376M	↓0.29%
Ours-0.5	1.77M	156M	↓1.42%
Ours-0.7	0.38M	75M	↓3.69%

Figure 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 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 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.

4.1 Implementation details

All our experiments are implemented in Tensorflow [1] with an NVIDIA 2080Ti GPU. We use SGD with Nesterov [24] 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

Method	Top-1 Baseline Accuracy (%)	Params	FLOPs	Top-1 Accuracy (%)
Original		14.98M	313M	93.26%
Variational pruning [36]	93.25%	3.92M	190M	↓0.07%
Geometric median [9]	93.53%	–	237M	↓0.34%
Try-and-learn [24]	92.77%	2.59M	140M	↓1.10%
Magnitude pruning [19]	93.25%	5.40M	206M	↓0.15%
Discrimination-aware [38]	93.99%	7.80M	157M	↓0.17%
Bayesian Pruning [37]	91.60%	**0.38M**	89M	↓0.60%
Ours-0.3	93.25%	7.76M	134M	↑**0.28**%
Ours-0.6	93.25%	2.50M	83M	↑0.07%
Ours-0.8	93.25%	0.97M	**52M**	↓0.28%

Table 1: Comparison to other filter-level pruning methods on the CIFAR10 benchmark and with the VGG16 architecture. Ours-$k0$ indicates a cascaded pruned student model using a filter pruning ratio of k0, whereas the "-" indicates that the results were not reported. For each model, the drop in accuracy is in reference to their own baseline.

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.

4.2 Comparisons on CIFAR10 and CIFAR100

The CIFAR10 dataset [16] consist of 60K 32×32 RGB images across 10 classes and with a 5:1 training/testing split. The chosen VGG16 [28] architecture is modified for this dataset by adding independent batch normalisation layers [33] 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 of k_0, $1 + \frac{k_0 - 1}{1.5}$, $1 + \frac{k_0 - 1}{2.5}$, and 1.0, respectively. We observe that any reasonably uniform allocation between $k0$ and 1.0 is sufficient to balance the gradient bias and the capacity gaps.

Table 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.

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 2 (right) and demonstrate the validity of this proposed training methodology on an already parameter efficient architecture.

4.3 Comparisons on ILSVRC-12

We evaluate cascaded pruning on ResNet-50 [8] for the ImageNet 2012 classification task [27]. Unlike most other pruning strategies [22, 37], 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

Method	Top-1 Baseline Accuracy (%)	Params	FLOPs	Top-1 Accuracy (%)	Top-5 Accuracy (%)
Original		25.5M	3.86B	75.03%	92.11%
Discrimination-aware [38]	76.01	12.38M	1.72B	↓1.06%	↓0.61%
Bayesian Pruning [37]	76.10	-	1.68B	↓3.10%	↓2.90%
NISP [42]	-	18.58M	2.81B	↓0.21%	-
Filter Sketch [20]	76.13	14.53M	2.23B	↓1.45%	↓0.69%
ThiNet [23]	72.88	12.28M	1.71B	↓1.87%	↓1.12%
S-ResNet-50 [33] $[0.25, 0.5, 0.75, 1.0]\times$	76.10%	25.5M	4.1B	↑**0.10**%	-
		14.7M	2.3B	↓1.20%	-
		6.9M	1.1B	↓4.00%	-
		2.0M	278M	↓11.10%	-
Cascaded pruning	75.03	7.12M	1.04B	↓2.50%	↓1.29%

Table 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.

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 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 [31], 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.

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.

5.1 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 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 uni-

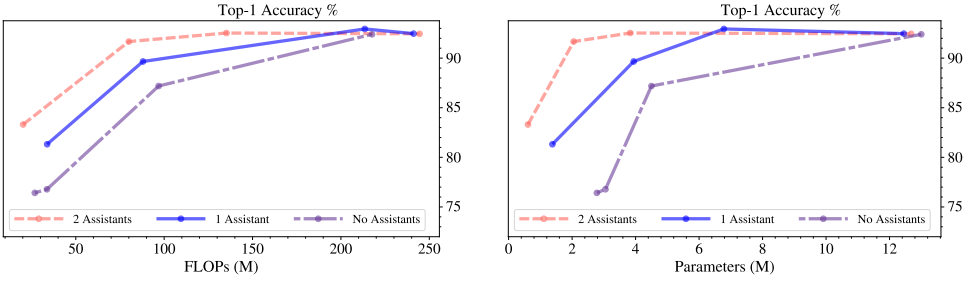


Figure 3: Evaluation of the computational complexity (left) and the number of parameters (right) for a student model with a varying number of teaching assistants. Each data point on the graph is ordered according to their pre-defined filter-pruning ratio using the modified VGG16 architecture on the CIFAR10 dataset.

form 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.

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.

Acknowledgement. This work was supported by UK EPSRC EP/S032398/1 & EP/N007743/1 grants.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint*, 2016.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint*, 2013.

[3] Michael H. Fox, Kyungmee Kim, and David Ehrenkrantz. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *CVPR*, 2018.

[4] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. *CVPR*, 2018.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[6] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. *CVPR*, 2017.

[7] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien Ju Yang, and Edward Choi. MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks. *CVPR*, 2018.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. ResNet - Deep Residual Learning for Image Recognition. *CVPR*, 2015.

[9] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. *CVPR*, 2018.

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

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *NeurIPS*, 2015.

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. *ICCV*, 2019.

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, and Marco Andreetto. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint*, 2017.

[14] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. *WACV*, 2018.

[15] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *ICLR*, 2017.

[16] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[17] Yann Lecun. Optimal Brain Damage. *NeurIPS*, 1990.

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

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters For Efficient Convnets. *ICLR*, 2017.

[20] Mingbao Lin, Rongrong Ji, Shaojie Li, Qixiang Ye, Yonghong Tian, and Jianzhuang Liu. Filter Sketch for Network Pruning. 2019.

[21] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal Loss for Dense Object Detection. *ICCV*, 2017.

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

[23] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher. *AAAI*, 2020.

[24] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 1983.

[25] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's Hidden in a Randomly Weighted Neural Network? *CVPR*, 2019.

[26] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints For Thin Deep Nets. *ICLR*, 2015.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2014.

[28] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks For Large-scale Image Recognition. *ICLR*, 2015.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. GoogLeNet/Inception - Going deeper with convolutions. In *CVPR*, 2015.

[30] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019.

[31] Hinton. Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA*, 2012.

[32] Jiahui Yu and Thomas Huang. AutoSlim: Towards One-Shot Architecture Search for Channel Numbers. *arXiv preprint*, 2019.

[33] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks. *ICLR*, 2018.

[34] Ruichi Yu, Ang Li, Chun Fu Chen, Jui Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching Yung Lin, and Larry S. Davis. NISP: Pruning Networks Using Neuron Importance Score Propagation. *CVPR*, 2018.

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

[36] Chenglong Zhao, Bingbing Ni, Jian Zhang, and Qiwei Zhao. Variational Convolutional Neural Network Pruning. *CVPR*, 2019.

[37] Yuefu Zhou, Ya Zhang, Yanfeng Wang, and Qi Tian. Accelerate CNN via Recursive Bayesian Pruning. *ICCV*, 2018.

[38] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware Channel Pruning for Deep Neural Networks. *NeurIPS*, 2018.