Learning to Explore Distillability and Sparsability: A Joint Framework for Model Compression

Yufan Liu, Jiajiong Cao, Bing Li, Weiming Hu, Stephen Maybank, *Fellow, IEEE*

Abstract—Deep learning shows excellent performance usually at the expense of heavy computation. Recently, model compression has become a popular way of reducing the computation. Compression can be achieved using knowledge distillation or filter pruning. Knowledge distillation improves the accuracy of a lightweight network, while filter pruning removes redundant architecture in a cumbersome network. They are two different ways of achieving model compression, but few methods simultaneously consider both of them. In this paper, we revisit model compression and define two attributes of a model: distillability and sparsability, which reflect how much useful knowledge can be distilled and how many pruned ratios can be obtained, respectively. Guided by our observations and considering both accuracy and model size, a dynamically distillability-and-sparsability learning framework (DDSL) is introduced for model compression. DDSL consists of teacher, student and dean. Knowledge is distilled from the teacher to guide the student. The dean controls the training process by dynamically adjusting the distillation supervision and the sparsity supervision in a meta-learning framework. An alternating direction method of multiplier (ADMM)-based knowledge distillation-with-pruning (KDP) joint optimization algorithm is proposed to train the model. Extensive experimental results show that DDSL outperforms 24 state-of-the-art methods, including both knowledge distillation and filter pruning methods.

Index Terms—Knowledge distillation, Filter pruning, Structured sparsity pruning, Deep learning.

1 Introduction

In recent years, deep neural networks (DNNs) have achieved huge success in various fields such as image classification [1], object detection [2] and video understanding [3]. To solve challenging problems and pursue high performance, deeper and wider architectures have been proposed at the expense of increased storage and computation time. However, these cumbersome DNNs fail to satisfy the efficiency requirement of edge devices. Model compression has become a fundamental topic in the search for greater efficiency. The aim is to obtain a lightweight DNN model with acceptable performance. In this field, diverse techniques have been tried, including filter pruning [4], knowledge distillation [5], low-rank decomposition [6], parameter quantization [7], *etc.* Among them, filter pruning and knowledge distillation are two widely explored techniques, because of their hardware-friendly capabilities and no requirement for special libraries.

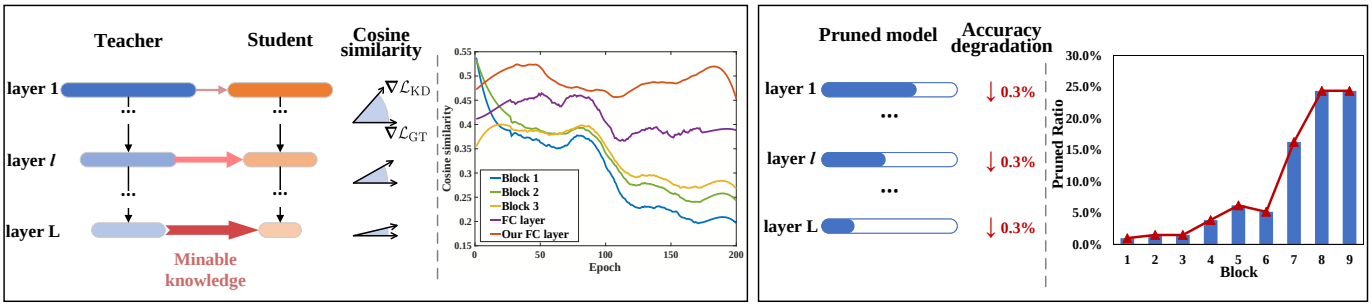
For knowledge distillation (KD) methods, a lightweight network (called a student network) is trained to mimic a large network (called a teacher network) [8]. With the guidance of the teacher network, the student network can achieve better performance, compared with the performance obtained by training without the guidance. However, KD methods only aim to improve the accuracy of the student network. They ignore the problem of selecting the architecture. The student's architecture is pre-defined and fixed during training. Recent work [9] shows that the architecture is crucial to the performance of compressed models.

In filter pruning methods, a cumbersome network is reduced to a sparse network. The filter pruning methods impose a structured sparsity regularization of the initial large network. However, model pruning is only used to obtain a compact architecture. Current methods do not take full advantage of the knowledge in the original large model. Some recent works [10], [11], [12] explore methods for balancing performance and model size, by combining KD and structured sparsity pruning. Most of the methods are limited to a straightforward loss combination. In contrast, our experiments show that dynamical joint learning of these two aspects is crucial to the final model performance.

We analyze the two aspects of performance and architecture during compression-aware training. In particular, we find that there are two important characteristics of the student model, *i.e.*, distillability and sparsability. Distillability refers to the density of useful knowledge that can be distilled from the teacher network. It measures the performance gain of the student under the guidance of the teacher. For instance, one student network with higher distillability obtains higher performance gain compared with that of another student with lower distillability. Distillability can be quantitatively analyzed in a layer-wise manner. As illustrated in Fig. 1-(a), the bar graph shows the cosine similarity between gradients of KD loss and those of ground truth (GT) loss. A larger cosine similarity value indicates that the current distilled knowledge is more beneficial to performance. In this way, the cosine similarity can be a measure of distillability. According to Fig. 1-(a), distillability increases as the layer becomes deeper. It explains why common practice usually adds KD supervision only to the last few layers. Further, the student also has different distillability at different training epochs, because the cosine similarity changes as training goes on. Thus, it is necessary to analyze the distillability in a dynamic layer-wise manner during training.

- *Corresponding Author: Bing Li*
- *Y. Liu, B. Li and W. Hu are with National Laboratory of Pattern Recognition, Institution of Automation, Chinese Academy of Sciences; School of Artificial Intelligence, University of Chinese Academy of Sciences and CAS Center for Excellence in Brain Science and Intelligence Technology. B. Li is also with People AI, Inc. (e-mail: yufan.liu@ia.ac.cn;{bli, wmhu}@nlpr.ia.ac.cn).*
- *Jiajiong Cao is with Ant Financial, China (e-mail: jiajiong.caojiajio@antgroup.com).*
- *Stephen Maybank is with the Department of Computer Science and Information Systems, Birkbeck College, University of London, London WC1E 7HX, U.K. (e-mail: sjmaybank@dcs.bbk.ac.uk).*

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317



(a) Distillability (b) Sparsability

Fig. 1. Distillability and sparsability of a DNN. (a) Illustration of distillability. The curves plot the cosine similarity between the KD loss and the ground truth loss at different blocks, as the training epoch changes. Note that "Our FC layer" represents the results of the DNN trained by our method. (b) Illustration of sparsability. The bar and line graph depict the pruned ratios that the model can achieve with the same accuracy degradation (*i.e.*, 0.3%). Each bar represents the compressed model in which only the corresponding layer is pruned. Note that the experiment is conducted on CIFAR10 using ResNet20.

On the other hand, the sparsability of the student network is a measure of the number of pruned ratios that can be obtained, under certain accuracy degradation. Higher sparsability indicates a larger potential pruned ratio. As depicted in Fig. 1-(b), different layers show different sparsability. Similar to distillability, sparsability can be analyzed at the layer-level and the epoch-level. Nevertheless, to the best of our knowledge, none of the existing methods explore and analyze both distillability and sparsability as they change during training. The methods that adopt a fixed training scheme may fail to obtain an optimal solution.

To overcome the shortcomings of existing methods and properly utilize distillability and sparsability, we firstly analyze the training procedure for model compression. Inspired by the results of the analysis, we propose a dynamically distillability-and-sparsability learning framework (DDSL) for model compression. It dynamically integrates KD and structure sparsity pruning and adjusts the joint training procedure by assessing the distillability and sparsability of the model. Rather than a "teacher-student" framework, the proposed DDSL can be described as a "learning-in-school" framework, which consists of three components: teacher, student and dean. The teacher teaches the student as before. The dean controls the learning intensity and learning manner. Given the current states of the teacher and the student, the dean network assesses the distillability and sparsability of the student network, and then it dynamically balances and controls the supervision intensities of KD and structure sparsity pruning. To optimize the proposed DDSL during training, we present an alternating direction method of multiplier (ADMM)-based knowledge-distillation-with-pruning (KDP) joint optimization algorithm to update the student network. To optimize the dean network, a meta-learning-based dean optimization algorithm is presented. Thanks to the dynamical adjustment of supervision of our method, the distillation supervision in reverse influences distillability. As can be seen in Fig. 1-(a), our method delays the downward trend of the distillability, and even increase the distillability by properly utilizing the distilled knowledge. To the best of our knowledge, this paper is the first attempt to explore the distillability and sparsability of DNN for model compression, and to balance these two aspects of model performance and pruned architecture. Our main contributions in this paper are three-fold:

- We explore the attributes of model compression during training, and obtain several observations which may inspire future model compression study. In particular, we find that distillability and sparsability are two crucial attributes for model compression.
- We propose a dynamically distillability-and-sparsability learning framework (DDSL) for model compression, which dynamically integrates KD and structure sparsity pruning and adjusts the joint training procedure by assessing the distillability and sparsability of the model.
- We propose an ADMM-based knowledge-distillation-with-pruning (KDP) joint optimization algorithm and a meta-learning-based dean optimization algorithm to optimize the overall framework.

2 Related Work

2.1 Knowledge distillation

The concept of knowledge distillation is introduced by Hinton *et al.* [5] based on a teacher-student framework. This method transfers knowledge from the trained teacher to the student network. Recently, it has been applied mainly to two areas: model compression [13] and knowledge transfer [14]. For model compression, a compact small student model is trained to mimic the pre-trained cumbersome teacher model.

Most knowledge distillation methods explore distilled knowledge in order to guide the student network, including instance feature, instance feature relationship and feature space transformation, etc. For instance feature, the related methods [5], [15] in the early time distill *logits* at the end of the network. The *logits* reflect the class distribution and contain more information than one-hot label. In this manner, the student network can be improved by learning more information. After that, features containing richer spatial information from intermediate layers [16], [17], [18] are extracted as the distilled knowledge. For example, FitNet [16] extracts the feature maps of the intermediate layers as well as the final output to teach the student network. Zagoruyko *et al.* [17] define Attention Transfer (AT) based on attention maps to improve the performance of the student network. More recently, structural knowledge [19], [20], [21], e.g., instance feature relationship and feature space transformation, has been presented, which represents more comprehensive information. For example, Liu *et al.* [19] propose the Instance Relationship Graph (IRG) to represent instance feature relationship and feature space transformation. It considers the geometry of the feature spaces and allows for dimension-agnostic transfer of knowledge. Yim *et al.* [21] present the Flow of Solution Procedure (FSP) to transfer the inference procedure of

the teacher, which can be seen as a feature space transformation rather than the intermediate layer results.

Though the above methods have reached a milestone in knowledge distillation, all of them follow a classic single-teacher-single-student framework. Recently, some works have explored new frameworks for knowledge distillation. For instance, [22] and [23] propose a mutual learning framework where multiple peer networks learn from each other. The papers [24] and [25] present self-distillation frameworks that enable the network to distill from itself. Meta learning methods are adopted to design new frameworks. Jang *et al.* [26] make use of meta learning to determine which information should be transferred during knowledge transfer. Liu *et al.* [27] directly learn soft targets via a meta network for self-distillation. However, nearly all of the previous works perform optimization with a fixed student network. A better resource-performance trade-off can be achieved, if the architecture design is considered during training.

2.2 Structured sparsity pruning

In model compression, structured sparsity pruning directly removes redundant neurons and channels rather than irregular weights. Thus, it is hardware-friendly and has been widely applied in recent years. Some works [28], [29], [30], [31], [32] aim to exploit a criterion of the filter importance and prune the unimportant filters, while some other works [33], [34], [35], [36] devote to training the network with additional sparse constraints and removing the sparse part of the network. For example, Li *et al.* [28] consider that the parameters with small L_1-norm are less important. He *et al.* [29] calculate the geometric median of the filters within the same layer and prune the filters near the geometric median. Afterwards, HRank [30] uses rank to assess the filter importance and pruned filters with low-rank feature maps. He *et al.* [31], [32] exploit a measure of the filter importance. The unimportant filters are pruned in a soft manner. In particular, the unimportant filters are just set to be zero but they may still be updated in the next training epoch. In contrast, some works [33], [34] impose sparse regularization to learn the importance of each channel. Huang *et al.* [35] present a scaling factor to scale the outputs of specific structures and add sparsity constraints on these factors, so that the structure corresponding to a zero-value scaling factor can be removed. ThiNet [36] regards filter pruning as an optimization problem, and prune each filter layer using statistical information from their next layer.

More recently, some works [37], [38], [39], [40], [41], [42] learn the sparse allocation of pruning, to meet budget constraints. For example, Gordon *et al.* [37] propose a general technique, *i.e.*, MorphNet, for resource-constrained optimization of DNN architecture. But the width multiplier that uniformly expands all layer sizes does not consider the difference among layers so that the resource allocation may not be optimal. ECC [38] introduces an energy consumption model to optimize the DNN compression problem and update the pruned ratio, under an energy constraint. ADMM is leveraged to solve the gradient-based learning problem. Besides, some works [39], [40], [41], [42] automatically learn the pruned ratio of each DNN layer. For instance, AMC [39] uses reinforcement learning to find a proper sparsity ratio for each layer. MetaPruning [41] constructs a meta network to directly generate the weights of the compressed model, given the sparse allocated ratios. Ning *et al.* [42] present a differentiable pruning process to learn the sparse allocation. ADMM is also used for the budgeted pruning problem. Though these previous works use complex optimization processes to meet the compression budget, no extra operation is adopted to enhance the model performance.

Recent works [10], [11], [12] combine knowledge distillation and model compression to obtain a compact model with high accuracy. Li *et al.* [11] first compress a teacher network to obtain a student network, and then add a 1×1 convolution layer at the end of each block to make the student mimic the teacher. After that, they merged the 1×1 convolution layer into the previous layer. Bai *et al.* [12] combine cross distillation and network pruning by adding regularization to a loss function. However, these methods either treat knowledge distillation and model compression as two independent stages or simply combine the loss functions. Without a framework-level re-design, it is difficult to achieve an optimal trade-off between performance and model complexity.

3 Exploration of Distillability and Sparsability

We define and explore the distillability and sparsability of DNN, and analyze the factors that may influence distillability and sparsability. Several important observations are obtained to inspire us for model compression. Here, our analysis is based on the methods in [4], [5] as applied to the CIFAR database. Specifically, we leverage *logits* as distilled knowledge and we utilize group Lasso regularization to achieve structured sparsity.

Definition: Distillability is the density of useful knowledge that can be distilled from the teacher network. It can be measured using the cosine similarity between gradients of the KD loss and those of the GT loss. **Sparsability** is the probability of the parameters being sparse. It can be measured using the number of pruned ratios that can be obtained, under certain accuracy degradation. It is hard to directly calculate distillability and sparsability at each time and in each layer. Thus, it is necessary to analyze these two characteristics and learn to adapt them.

Our analysis has five aspects, as follows.

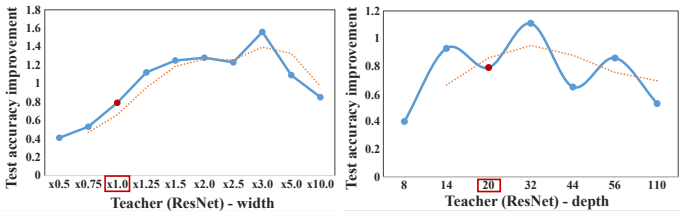
Observation 1*: Distillability and sparsability are influenced by the architectures of teacher and student.*

Analysis: For distillability, intuitively, larger teacher is supposed to result in higher distillability since there is richer knowledge to be distilled. To verify this, we train the student network using various teachers with different depths and widths. However, as Fig. 2 depicted, teachers with larger capacity do not necessarily produce a better student. For example, teacher-student (T-S) combination {T-S: ResNet20-ResNet20} shows significant higher performance than that of {T-S: ResNet110-ResNet20} in Fig. 2-(b). It can be concluded that different teacher-student pairs with various architectures have different distillability.

For sparsability, we compress different networks with similar model size by structure sparsity pruning. From Fig. 3, the compressed models have different pruned ratios, even with similar initial model size and similar accuracy degradation. This indicates that the model architecture has an influence on sparsability. This completes the analysis of *observation 1*.

Observation 2*: The distillability and sparsability vary in different layers.*

Analysis: We analyze distillability and sparsability of each DNN layer shown in Fig. 1. As mentioned in Sec. 1, Fig. 1-(a) illustrates that different layers have various distillability, and deeper layers have higher distillability. In Fig. 1-(b), different layers achieve different pruned ratios, when remaining the same



(a) (b)

Fig. 2. The accuracy curves of the student network ResNet20, which is trained under the guidance of teacher networks with different capacities. In (a), the teachers vary in width. For example, "x1.0" denotes the channel number of this teacher is 1.0 time of ResNet20. In (b), the teachers vary in depth, including ResNet8, ResNet14, ResNet20, ResNet32, ResNet44, ResNet56 and ResNet110. Note that the orange dotted lines represent the trendline.

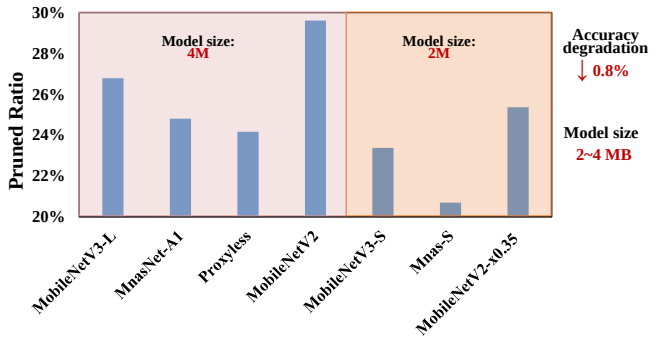


Fig. 3. Pruned ratios of different models which have similar accuracy degradation and initial model sizes. The models are MobileNetV2, MobileNetV2-x0.35 [43], MobileNetV3-Large, MobileNetV3-Small [44], MnasNet-A1, Mnas-Small [45] and Proxyless [46].

accuracy degradation. This manifests that sparsability varies in different layers, and similarly, deeper layers have higher sparsability. This completes the analysis of *observation 2*. This observation indicates that layer-wise fine-grained distillability and sparsability are necessary to improve the model performance.

Observation 3*: The distillability and sparsability dynamically changes when training goes on.*

Analysis: For distillability, we find that distillability decreases as training goes on, and the KD supervision may even harm the model performance towards the end of training. As shown in Fig. 4, the KD loss has a high correlation with the GT loss at the early times in the training. As training goes on, the correlation decreases and may even become negative. This indicates that KD does help the converge of the GT loss, but towards the end of the training the KD loss may reduce accuracy, *i.e.*, the "distillability" is reduced. Furthermore, the smaller the teacher network, the lower the correlation. Thus, small teachers are not recommended. During training, it is necessary to adjust the intensity of the knowledge distillation. To further verify this observation, we remove KD supervision at the end of training. As reported in Fig. 5(a), on ImageNet, the students removing KD supervision in the end (*i.e.*, "ResNet-KD-Partial") obtain higher performance than the students that use KD (*i.e.*, "ResNet-KD-Full") throughout the training. However, models trained on CIFAR10 have reverse behavior. This may be because CIFAR10 is a simple database, and a small student can minimize both the cross entropy loss and the knowledge distillation loss. In contrast, ImageNet is a challenging database. The student cannot simultaneously minimize the two losses, especially when these two losses have a low correlation at the end of the training.

For sparsability, it has been verified in *observation 1* that different architectures lead to different sparsability. Since the model architecture changes during the training procedure, it is obvious that sparsability changes as training goes on. It is interesting to notice that the sparsity loss itself has little or no direct negative impact on model performance. As depicted in Fig. 4(b), its correlation with the GT loss is nearly zero. This suggests that the sparsity loss does not reduce the accuracy. Fig. 5(b) shows that some pruned networks have a better performance than the baseline. When the sparsity ratio is increased, the pruned networks with high sparsity ratios have low accuracy, since they have low capacities and thus underfit the data. Hence, the factor that affects the model performance is model capacity, rather than the sparsity loss. Finally, this completes the analysis of *Observation 3*.

Observation 4*: The intensities of KD and sparse supervision, which can be adjusted to adapt distillability and sparsability, are crucial to model performance.*

Analysis: We firstly analyze different static values of the supervision intensity. For knowledge distillation, we use different coefficients in the KD regularizer to test the student performance. As can be seen in Fig. 6, different teacher-student pairs have different optimal coefficients. In the case of structured sparsity, Fig. 5(b) shows that different sparse regularizer coefficients yield different accuracy and sparsity ratios. A proper coefficient is important to balance the model performance and model size. Thus, the accurate selection of the supervision intensities for both KD and structured sparsity is crucial to model performance.

Besides static supervision intensity above, the effects of dynamically changing the supervision intensities are examined next. We construct sequences of supervision intensities with various trends, including increasing trend (*i.e.*, "Sigmoid" and "Cosine") , decreasing trend (*i.e.*, "Sigmoid ↓" and "Cosine ↓") , increasing followed by decreasing (*i.e.*, "Cosine ↑↓"), and decreasing followed by increasing (*i.e.*, "Cosine ↓↑"). The details are described in *supplemental material*. On embedding these sequences to control the intensity of the supervision, we obtain different performances of the model, as shown in Fig. 7. It is clear that most of the "dynamically" trained models are superior to "statically" trained model in both KD and sparsity pruning. This indicates that dynamically adjusting the supervision yields better performance, compared with the static supervision. However, these sequences are designed manually and are fixed, thus they may not be suitable for every student. In contrast, the proposed method, dynamically adjusting the supervision driven by network status, achieves a much higher performance gain. Thus, dynamic training can improve model performance. *Observation 4* is verified experimentally.

Observation 5*: The proper combination of knowledge distillation and sparsity pruning can obtain higher performance.*

Analysis: In order to evaluate the effectiveness of direct combination, we firstly prune a model using [47] and then use KD [5] to improve the model accuracy. As illustrated in Fig. 8, most pruned models retrained with KD surpass the original pruned models. This indicates KD can improve the performance of a pruned model. However, when the pruned ratio reaches to an extremely large value (e.g., more than 80%), KD degrades the accuracy of the pruned model. As mentioned in *Observation 1* and *Observation 3*, the capacity gap may lead to this phenomenon. In addition, *Observation 4* tells us that a fixed KD supervision usually fails to obtain the best performance. Hence, a direct combination of KD and pruning is obviously sub-optimal, since it neglects all the previous observations. It motivates us to propose a better

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

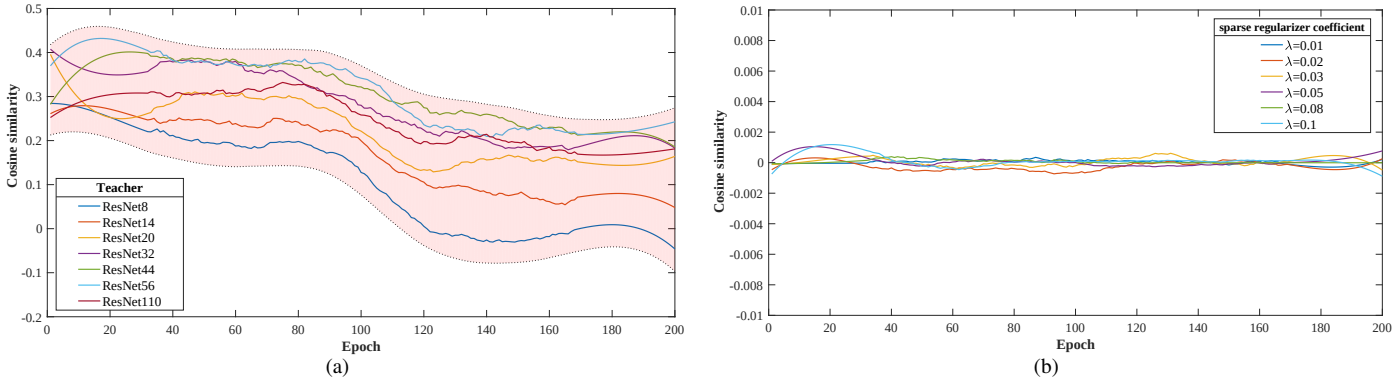


Fig. 4. Cosine similarity between different losses. (a) Cosine similarity between knowledge distillation loss and ground truth loss. (b) Cosine similarity between sparsity loss and ground truth loss. Note that the student network in (a) and the initial network in (b) are both ResNet20.

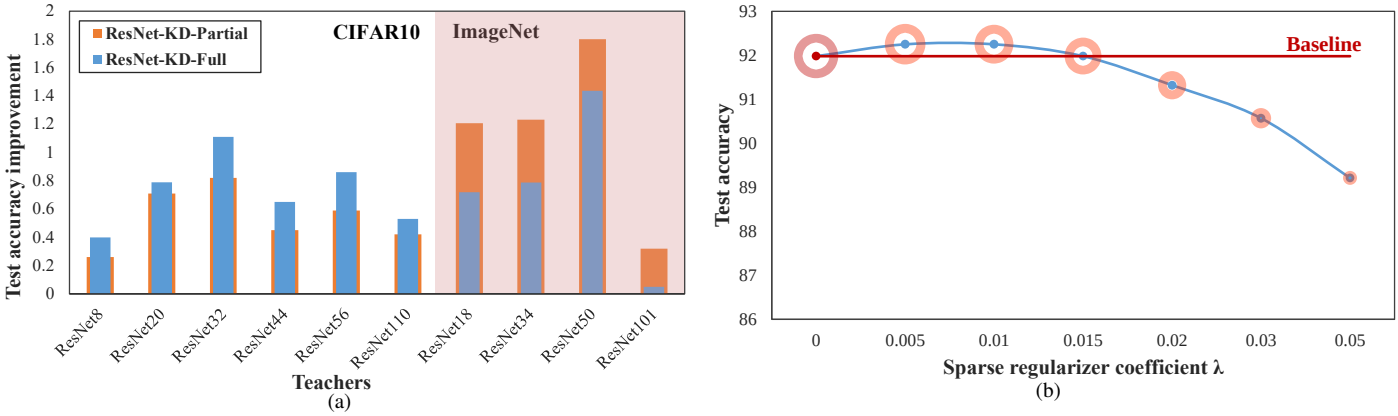


Fig. 5. Test accuracy comparisons at different situations. (a) The test accuracy improvement of ResNet20 on CIFAR10 and ResNet18 on ImageNet. ResNet20 and ResNet18 are distilled from different teacher networks. Note that "ResNet-KD-Full" represents the models that use KD during the whole training procedure, while "ResNet-KD-Partial" represents the models that remove KD at the end of training (*i.e.*, after the 150-th epoch). (b) The test accuracy of different pruned networks using different sparse regularizer coefficients. A larger sparse regularizer coefficient corresponds to a higher sparsity ratio in the pruned model. And "baseline" means the initial network trained without sparsity constraint. Note that the student network in (a) and the initial network in (b) are both ResNet20.

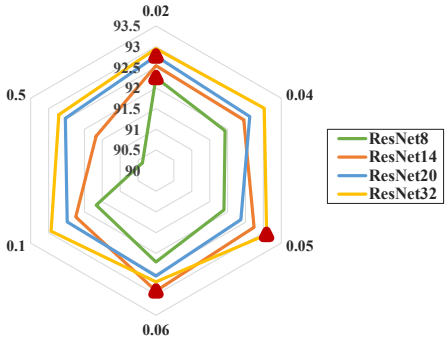


Fig. 6. Radar chart of the student accuracy, at different coefficients of KD regularizer (*i.e.*, 0.02, 0.04, 0.05, 0.06, 0.1, 0.5). Note that these full lines with different colors represent different teacher networks (*i.e.*ResNet8, ResNet14, ResNet 20, ResNet32). The student network is ResNet20.

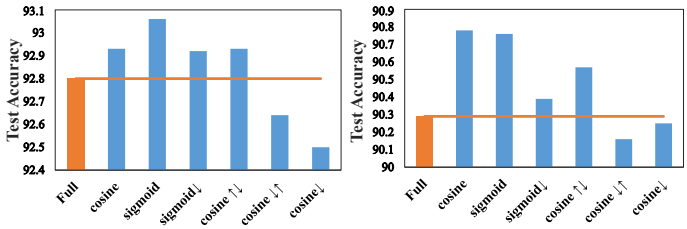


Fig. 7. Test accuracy for different models training with different sequences of supervision intensities. (a) training of ResNet20 on CIFAR10, utilizing KD supervision. The teacher is ResNet20-2x. (b) Training of ResNet20 on CIFAR10 with sparse supervision. The pruned ratios of these models are all 50%. Note that "Full" denotes that the supervision coefficient is static.

combination guided by the previous observations.

Given the above observations, we construct the DDSL framework. It sufficiently considers the information about the teacher and the student, and adjusts the supervisions in layer wise and time wise by dynamically assessing the distillability and sparsability.

4 Approach

In Sec. 4.1, we review knowledge distillation and structured sparsity pruning. We introduce the proposed method in Sec. 4.2. Descriptions of the framework, formulation, optimization and training procedure are included.

4.1 Background: knowledge distillation and pruning

Knowledge distillation, first presented by Hinton *et al.* [5], is an efficient teacher-student learning framework. On distilling knowledge from a teacher network $\mathbf{\Theta}^{\mathrm{T}}$ to guide a student network $\mathbf{\Theta}^{\mathrm{S}}$, the student network can obtain better performance. Knowledge distillation is easily applied to obtain a compact and small

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

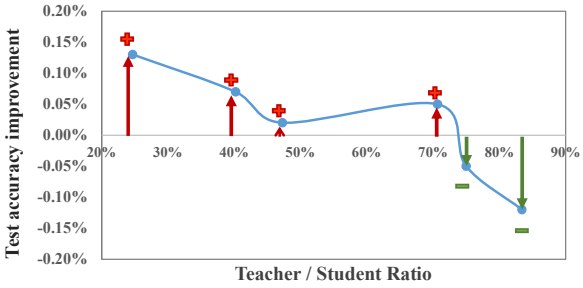


Fig. 8. Test accuracy improvements of pruned models retrained with KD, at different parameter pruned ratios. Note that the initial network is ResNet20, which is regarded as the teacher in KD. The student networks are the pruned networks at different pruned ratios.

network by learning from a large and powerful teacher. Given the training data $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, where $\mathbf{x}$ is the input sample and $\mathbf{y}$ is the corresponding label, the objective of training the student network is formulated as:

$$\min_{\mathbf{\Theta}^{\mathrm{S}}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}_{\mathrm{CE}}(\mathcal{F}(\mathbf{\Theta}^{\mathrm{S}}, \mathbf{x}), \mathbf{y}) + \lambda \mathcal{L}_{\mathrm{D}}(\mathbf{x}, \mathbf{\Theta}^{\mathrm{T}}, \mathbf{\Theta}^{\mathrm{S}}), \tag{1}$$

where $\mathcal{F}(\cdot)$ is the deep neural network (DNN) forward function. $\mathcal{L}_{\mathrm{CE}}$ is the standard cross entropy loss and $\mathcal{L}_{\mathrm{D}}$ is the distillation loss which measures the similarity of the output distributions of the teacher and student. The distillation loss is usually set to be the Kullback-Leibler (KL) divergence. The balancing parameter λ must be tuned manually. This is inconvenient and may not yield an optimal solution.

Structured sparsity pruning does not focus on improving the performance of a pre-defined light-weight network. Instead, it concentrates on obtaining a high compression ratio for a large existing model with marginal accuracy degradation. Recently, achieving a target compression ratio is a new requirement. In this manner, the objective of structured sparsity pruning is formulated as:

$$\begin{aligned} &\min_{\mathbf{\Theta}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}_{\mathrm{CE}}(\mathcal{F}(\mathbf{\Theta}, \mathbf{x}), \mathbf{y}), \\ &\text{s.t.} \quad PR_{\mathrm{goal}} - PR(\mathbf{\Theta}) \leq 0. \end{aligned} \tag{2}$$

Note that $PR(\cdot)$ is the pruned ratio of parameters or FLOPs, and PR_{goal} is the target pruned ratio. However, it is not convenient to directly optimize under the constraint in Eqn. 2. To convert the original problem (2) to an unconstrained optimization problem, a structured sparsity regularization $\mathcal{R}(\cdot)$ is introduced:

$$\min_{\mathbf{\Theta}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathcal{L}_{\mathrm{CE}}(\mathcal{F}(\mathbf{\Theta}, \mathbf{x}), \mathbf{y}) + \gamma \mathcal{R}(\mathbf{\Theta}), \tag{3}$$

where γ is the penalty coefficient. A higher value of γ usually yields a higher pruned ratio. However, there is no direct relation between γ and the pruned ratio. This makes it difficult to tune γ to achieve a specified pruned ratio. To meet the target pruned ratio, existing methods directly prune the network including the non-zero parameters, which harms the performance. Thus, the pruned model needs to be fine-tuned to recover the accuracy.

In summary, knowledge distillation and structured sparsity pruning achieve model compression in two different ways. Knowledge distillation improves the accuracy of the compressed model, while structured sparsity pruning compresses the model and obtains a proper pruned ratio. The proper integration of these two tasks has potential prospects.

4.2 Dynamically distillability-and-sparsability learning

Here, we introduce the proposed method, including framework, formulation, optimization and procedure aspects.

4.2.1 Framework

According to the observations in Sec. 3, the choice of a sequence of supervision intensities is crucial to model performance. In addition, it is important to integrate knowledge distillation and structured sparsity pruning. Therefore, we propose a dynamically distillability-and-sparsability learning framework (DDSL) for model compression. This method balances knowledge distillation and model parameters sparsity, by assessing the distillability and sparsability of the current model and controlling the supervision intensities accordingly, during the training procedure. The proposed method takes into account two aspects of model compression, *i.e.*, accuracy and compressed ratio, to obtain a smaller but more robust pruned model.

The overall framework is summarized in Fig. 9. Rather than "teacher-student" framework, it is a "learning in school" framework, containing teacher, student and dean. The "teacher" guides the "student" to learn the knowledge. At the same time, the "student" also learns to become slim. The "dean" dynamically adjusts the curricula that the student learns, taking into account the current states of the teacher and the student. At the beginning of the training, a network that needs to be compressed and a dean module are prepared. Then, the initial network $\mathbf{\Theta}^{\mathrm{T}}$ is regarded as the teacher, and the network $\mathbf{\Theta}^{\mathrm{S}}$ in the subsequent epochs is regarded as the student. In each epoch, the dean $\mathbf{\Phi}$ is fed with teacher information and student information to assess current distillability and sparsability. It then controls the intensity of the supervision by KD and sparsity. In this manner, the student dynamically learns to be distilled and sparse at each epoch. When the student has a higher distillability, a stronger KD supervision signal is imposed (see the pink background in the pruned network in Fig. 9). In contrast, when the student has a higher sparsability, a stronger sparse supervision signal is imposed (see the orange background in Fig. 9).

4.2.2 Formulation

In this subsection, we formulate the proposed method. Based on Eqn. 1 and 2, the objective of the proposed DDSL is written as:

$$\begin{aligned} \min_{\mathbf{\Theta}^{\mathrm{S}}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \Big[&\mathcal{L}_{\mathrm{CE}}(\mathcal{F}(\mathbf{\Theta}^{\mathrm{S}}, \mathbf{x}), \mathbf{y}) \\ &+ \mathcal{L}_{\mathrm{KD}}(\mathbf{x}, \mathbf{\Theta}^{\mathrm{T}}, \mathbf{\Theta}^{\mathrm{S}}, \mathbf{\Phi}) + \mathcal{L}_{\mathrm{SP}}(\mathbf{\Theta}^{\mathrm{S}}, \mathbf{\Phi}) \Big]. \end{aligned} \tag{4}$$

Note that $\mathcal{L}_{\mathrm{KD}}$ is the knowledge distillability loss while $\mathcal{L}_{\mathrm{SP}}$ is the sparsability loss. The term $\mathbf{\Phi}$ is the dean network, which dynamically adjusts these two losses. We describe the dean network and these two losses as follows.

(1) Dean network

Dean network is fed with the features of teacher and the features of student, from which two signals (*i.e.*, $\boldsymbol{\alpha} = \{\boldsymbol{\alpha}_l\}_{l=1}^{L}$ and $\boldsymbol{\beta} = \{\boldsymbol{\beta}_l\}_{l=1}^{L}$) are obtained to assess current distilability and sparsability and adjust the supervisions. In detail, $\boldsymbol{\alpha}_l$ reflects the distilability and is used to adjust the KD supervisions from the l-th layer. The term $\boldsymbol{\beta}_l$ reflects the sparsability and is used to adjust the sparsity supervisions from the l-th layer. These two signals

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

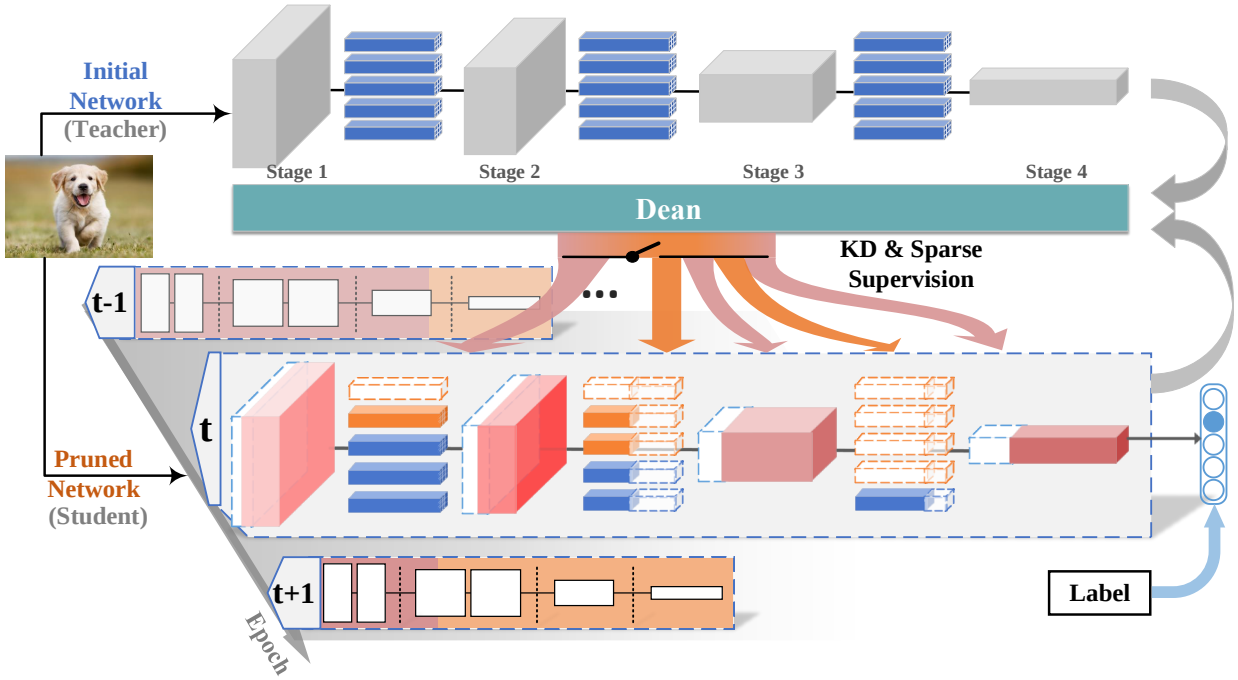


Fig. 9. The overall framework of the proposed method, which contains teacher, student and dean.

can also be regarded as the supervision control signals. They are computed as:

$$\{\boldsymbol{\alpha},\ \boldsymbol{\beta}\} = \bigcup_{l=1}^{L} \mathcal{F}\left(\boldsymbol{\Phi}_l, \{\mathbf{h}^{\mathrm{T}}_{g(l)}, \mathbf{h}^{\mathrm{S}}_{l}\}\right), \tag{5}$$

where $\mathbf{h}^{\mathrm{T}}_{g(l)} = \mathcal{F}_{g(l)}(\boldsymbol{\Theta}^{\mathrm{T}}, \mathbf{x})$ and $\mathbf{h}^{\mathrm{S}}_{l} = \mathcal{F}_{l}(\boldsymbol{\Theta}^{\mathrm{S}}, \mathbf{x})$ are the features of the teacher from the $g(l)$-th layer and that of the student from the l-th layer, respectively. The term $\mathcal{F}_l$ is the l-th layer output of the DNN forward function. Note that when the student is the initial network before compression, there is one-to-one correspondence between the layer indexes of teacher and that of student: $g(l) = l$. When the teacher has a different architecture from the student, $g(\cdot)$ is a linear function that maps the student layer to the corresponding teacher layer. In the dean network $\boldsymbol{\Phi}$, each layer corresponds to a dean module $\boldsymbol{\Phi}_l$. In our implementation, the l-th dean network $\boldsymbol{\Phi}_l$ is comprised of one fully-connected (FC) layer followed by a activation function of *Rectified Linear Unit 6 (ReLU6)* and a normalization operation. The *ReLU6* and the normalization are used to make the output distribute in a reasonable range.

(2) Knowledge distillability loss

The knowledge distillability loss $\mathcal{L}_{\mathrm{KD}}$ is used to promote the model performance. It is defined by:

$$\mathcal{L}_{\mathrm{KD}}(\mathbf{x}, \boldsymbol{\Theta}^{\mathrm{T}}, \boldsymbol{\Theta}^{\mathrm{S}}, \boldsymbol{\Phi}) = \sum_{l=1}^{L} \boldsymbol{\alpha}_l \cdot \mathrm{Dist}(\mathcal{K}(\mathbf{h}^{\mathrm{T}}_{g(l)}) || \mathcal{K}(\mathbf{h}^{\mathrm{S}}_{l})). \tag{6}$$

Note that $\boldsymbol{\alpha}_l$ denotes the distillation supervision control signal at the l-th layer, which also reflects the knowledge distillability. The value of $\boldsymbol{\alpha}_l$ changes over time. The function $\mathrm{Dist}(\cdot)$ measures the similarity of the teacher and the student. Possible choices for $\mathrm{Dist}(\cdot)$ include the Euclidean distance and the KL divergence. The term $\mathcal{K}(\cdot)$ is the knowledge distilled from the teacher network. It can take the form of an attention map [17] or a feature distribution. Here, we use an informative knowledge: the instance relationship graph (IRG), proposed by [19]. It contains both instance feature and instance relationship. For simplification, the distance of the intermediate layer is computed using the IRG edges difference:

$$\mathrm{Dist}(\mathcal{K}(\mathbf{h}^{\mathrm{T}}_{g(l)}) || \mathcal{K}(\mathbf{h}^{\mathrm{S}}_{l})) = ||\mathbf{A}^{T}_{g(l)} - \mathbf{A}^{S}_{l}||^2_2, \tag{7}$$

in which $\mathbf{A}^{T}_{g(l)}$ and $\mathbf{A}^{S}_{l}$ denote the IRG adjacency matrices of teacher and student at the l-th layer, respectively. As can be seen, this distance and corresponding loss (6) are differentiable and can be easily optimized.

(3) Sparsability loss

The sparsability loss $\mathcal{L}_{\mathrm{SP}}$ is used to make the model compact. It is defined by:

$$\mathcal{L}_{\mathrm{SP}}(\boldsymbol{\Theta}^{\mathrm{S}}, \boldsymbol{\Phi}) = \sum_{l=1}^{L} \boldsymbol{\beta}_l \cdot \mathcal{R}(\boldsymbol{\Theta}^{\mathrm{S}}_l), \tag{8}$$

where $\boldsymbol{\beta}_l \in \mathbb{R}^{c_l}$ denotes the sparsity supervision control signal at the l-th layer, which also reflects the sparsability. The elements of $\boldsymbol{\beta}_l$ correspond to c_l channels. And c_l is the overall number of channels in the l-th layer. The term $\mathcal{R}(\boldsymbol{\Theta}^{\mathrm{S}}_l)$ is the sparse regularization imposed on the network parameters of the l-th layer. To achieve the structure sparsity, we introduce an ℓ_{21}-norm based regularization:

$$\mathcal{R}(\boldsymbol{\Theta}^{\mathrm{S}}_l) = \sum_{j \in \mathcal{C}^{(l)}_{\mathrm{sel}}} \left(\left\| (\mathbf{W}^{(l)}_{j,:})^{\mathrm{2D}}_{\mathrm{out}} \right\|_2 + \left\| (\mathbf{W}^{(l+1)}_{j,:})^{\mathrm{2D}}_{\mathrm{in}} \right\|_2 \right). \tag{9}$$

Note that $(\mathbf{W}^{(l)})^{\mathrm{2D}}_{\mathrm{out}} \in \mathbb{R}^{c_l \times c_{l-1} k^{(l)}_1 k^{(l)}_2}$ and $(\mathbf{W}^{(l)})^{\mathrm{2D}}_{\mathrm{in}} \in \mathbb{R}^{c_{l-1} \times c_l k^{(l)}_1 k^{(l)}_2}$ denote the matrix-forms of tensor $\boldsymbol{\Theta}^{\mathrm{S}}_l \in \mathbb{R}^{c_l \times c_{l-1} \times k^{(l)}_1 \times k^{(l)}_2}$ along output and input channels, respectively. And $k^{(l)}_1 \times k^{(l)}_2$ indicates the size of the 2-dimensional spatial kernel. In addition, $\mathcal{C}^{(l)}_{\mathrm{sel}}$ represents the set of the selected "unimportant" channels that are restricted by the sparse regularization. In this manner, "unimportant" channels are forced to be zero and a compact model is thus obtained.

In summary, the pruned ratio can be controlled to meet the target pruned ratio as follows. We propose a progressively layer-wise sparsity allocation scheme (see details in the following

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

paragraph). In this scheme, to achieve the target pruned ratio, each layer of the student network is allocated with different sparsity ratios progressively during training. For each training epoch, first, the "importance" of each channel is calculated according to the proposed channel importance metric (described in the following paragraph). After ranking by this importance metric, a corresponding ratio of "unimportant" filters is selected (*i.e.*, the set of $\mathcal{C}_{\text{sel}}^{(l)}$). This ratio of each layer is initialized by the target pruned ratio of the whole network and is updated for each epoch (see details in Alg. 1). Then, the sparsity regularization is imposed onto these unimportant filters, whose intensity is controlled by $\boldsymbol{\beta}$. After a training epoch, a part of these filters is forced to be zero and a new iteration of sparsity allocation is started by selecting "unimportant" filters from non-zero ones. After several iterations of sparsity allocations, the student network achieves the target pruned ratio.

Channel importance metric. As mentioned above, the sparse constraint is only imposed on certain "unimportant" channels, it is necessary to select these "unimportant" channels under a metric. Based on [47], we present a channel importance metric to identify the "unimportant" channels. In particular, channel importance is evaluated from two aspects. On one hand, sparsability of the channel reflects the channel importance. It represents the necessity that the channel need to be removed, thus higher sparsability corresponds to lower channel importance. On the other hand, channel sensitivity, namely, how much impact on model performance when removing the channel also reflects the channel importance. Here, we use cross entropy loss to represent the model performance. Begin with a single parameter w. If removing it causes a large change in the cross entropy loss, then the corresponding channel is important. Thus, the sensitivity $\text{Sen}(w)$ of the parameter can be computed:

$$\text{Sen}(w) = |\mathcal{L}_{\text{CE}}(\mathcal{F}(\boldsymbol{\Theta}_{w\to 0}^{\text{S}}, \mathbf{x}), \mathbf{y}) - \mathcal{L}_{\text{CE}}(\mathcal{F}(\boldsymbol{\Theta}_{w}^{\text{S}}, \mathbf{x}), \mathbf{y})|, \quad (10)$$

where $\boldsymbol{\Theta}_{w\to 0}$ denotes that parameter w tends to 0. On using Taylor series, the loss function can be approximated by

$$\begin{aligned}\mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}_{w}^{\text{S}}, \mathbf{x}), \mathbf{y}\right) = \mathcal{L}_{\text{CE}}(\mathcal{F}(\boldsymbol{\Theta}_{w\to 0}^{\text{S}}, \mathbf{x}), \mathbf{y}) + \\ \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}_{w\to 0}^{\text{S}}, \mathbf{x}), \mathbf{y}\right)}{\partial w} w + O\left(w^2\right),\end{aligned} \quad (11)$$

where $O\left(w^2\right)$ is a remainder term. Because w is very small, $O\left(w^2\right)$ can be ignored. After substituting Eqn. 11 into Eqn. 10, we obtain the parameter sensitivity for w. To achieve structured sparsity, we extend the parameter sensitivity to channel sensitivity:

$$\begin{aligned}&\text{Sen}((\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}) \\ &= \left|\mathcal{L}_{\text{CE}}(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}) - \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}}, \mathbf{x}), \mathbf{y})\right| \\ &\approx \left|\frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}} (\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}\right| \\ &= \left|\sum_{m=1}^{c_{l-1}k_1^{(l)}k_2^{(l)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}}\right|.\end{aligned} \quad (12)$$

When the output channels are removed in the current layer, the corresponding input channels in the next layer are also pruned. Hence, we need to consider channel sensitivity at the next layer:

$$\begin{aligned}&\text{Sen}((\mathbf{W}_{j,:}^{(l+1)})_{\text{in}}^{\text{2D}}) \\ &\approx \left|\sum_{m=1}^{c_l k_1^{(l+1)} k_2^{(l+1)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l+1)})_{\text{in}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}\right|.\end{aligned} \quad (13)$$

Thus, the channel sensitivity of the j-th channel at the l-th layer can be computed by:

$$\begin{aligned}&\text{Sen}(\mathcal{W}_j^{(l)}) \\ &= \Bigg|\sum_{m=1}^{c_{l-1}k_1^{(l)}k_2^{(l)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}} \\ &+ \sum_{m=1}^{c_l k_1^{(l+1)} k_2^{(l+1)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l+1)})_{\text{in}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}\Bigg|.\end{aligned} \quad (14)$$

Note that $\mathcal{W}_j^{(l)}$ denotes the parameter tensor of the j-th channel at the l-th layer.

To summarize, by considering both channel sparsability and channel sensitivity, we obtain the final channel importance score:

$$\begin{aligned}&I(\mathcal{W}_j^{(l)}) = \frac{1}{\boldsymbol{\beta}_l(j)} \cdot \text{Sen}(\mathcal{W}_j^{(l)}) \\ &= \frac{1}{\boldsymbol{\beta}_l(j)} \Bigg|\sum_{m=1}^{c_{l-1}k_1^{(l)}k_2^{(l)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l)})_{\text{out}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l)})_{\text{out}}^{\text{2D}} \\ &+ \sum_{m=1}^{c_l k_1^{(l+1)} k_2^{(l+1)}} \frac{\partial \mathcal{L}_{\text{CE}}\left(\mathcal{F}(\boldsymbol{\Theta}^{\text{S}}_{(\mathbf{W}_{j,:}^{(l+1)})_{\text{in}}^{\text{2D}}\to \mathbf{0}}, \mathbf{x}), \mathbf{y}\right)}{\partial (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}} (\mathbf{W}_{j,m}^{(l+1)})_{\text{in}}^{\text{2D}}\Bigg|.\end{aligned} \quad (15)$$

Here, $\boldsymbol{\beta}_l(j)$ is the j-th element of $\boldsymbol{\beta}_l$. It represents the sparsability of the j-th channel at the l-th layer. We integrate the channel sparsability and channel sensitivity by taking $\boldsymbol{\beta}_l(j)$ as the attention weight to adjust $\text{Sen}(\mathcal{W}_j^{(l)})$. After obtaining the channel importance score of each channel, we identify the "unimportant" channels by sorting all the channels from large to small. The lower ranked channels comprise the set $\mathcal{C}_{\text{sel}}^{(l)}$ and are restricted by the sparse regularization.

Progressively layer-wise sparsity allocation. Previous works usually adopt a uniform prior for sparsity allocation, namely, each layer is optimized with the same sparsity ratio. However, this usually leads to a sub-optimal architecture, as mentioned in [9]. Therefore, it is crucial to allocate a customized sparsity ratio $sr^{(l)}$ to each layer. At the beginning of the training, it is difficult to determine the final sparsity allocations. Thus, a progressive layer-wise sparsity allocation scheme is proposed to iteratively update the sparsity ratio $sr^{(l)}$ of each layer during training. In particular, the sparsity ratio $sr^{(l)}$ is first initialized as PR_{goal} equally for each layer. Then, at the beginning of each training epoch, the channels are divided into two types, namely dead channels which are zeros, and active channels which are non-zero. The ratio of dead channels of l-th layer is presented as the actual sparsity ratio $sr_{\text{act}}^{(l)}$ and is computed as:

$$sr_{\text{act}}^{(l)} = \frac{\#Param_{\text{zero}}^{(l)}}{\#Param_{\text{total}}^{(l)}}, \quad (16)$$

where $\#Param_{\text{zero}}^{(l)}$ is the number of zero parameters and $\#Param_{\text{total}}^{(l)}$ is the total number of parameters in the l-th layer.

Algorithm 1: Progressively layer-wise sparsity allocation.

Input: $\mathbf{\Theta}^{\mathrm{S}}$, PR_{goal}, current epoch e.

if $e = 0$ **then**
  Initilization: $(sr_{\mathrm{act}}^{(l)})^{(0)} = 0, \quad l = 1, 2, ..., L$;
  $(sr_{\mathrm{rmng}})^{(0)} = PR_{\mathrm{goal}}$ equally assigned to each layer;
else
  $(sr_{\mathrm{act}}^{(l)})^{(e)} = \frac{(\#Param_{\mathrm{zero}}^{(l)})^{(e)}}{\#Param_{\mathrm{total}}^{(l)}}, \quad l = 1, 2, ..., L$;
  $(sr_{\mathrm{rmng}})^{(e)} = (sr_{\mathrm{rmng}})^{(e-1)}$ equally assigned to each layer;
end
repeat
  Update sparsity ratio:
  $(sr^{(l)})^{(e)} = (sr_{\mathrm{act}}^{(l)})^{(e)} + (sr_{\mathrm{rmng}})^{(e)}, \ l = 1, 2, ..., L$;
  $\hat{\mathbf{\Theta}}^{\mathrm{S}} \leftarrow$ Prune each layer of $\mathbf{\Theta}^{\mathrm{S}}$ using ratio $sr^{(l)}$;
  Search a proper sr_{rmng} via a step size, to meet PR_{goal}:
  if $PR(\hat{\mathbf{\Theta}}^{\mathrm{S}}) > PR_{\mathrm{goal}}$ **then**
    $sr_{\mathrm{rmng}} = sr_{\mathrm{rmng}} - \Delta$;
  else if $PR(\hat{\mathbf{\Theta}}^{\mathrm{S}}) < PR_{\mathrm{goal}}$ **then**
    $sr_{\mathrm{rmng}} = sr_{\mathrm{rmng}} + \Delta$;
  end
until $|PR(\hat{\mathbf{\Theta}}^{\mathrm{S}}) - PR_{\mathrm{goal}}| < \Delta$;
return $\{sr^{(l)}\}_{l=1}^{L}$;

Note that $\#Param_{\mathrm{zero}}^{(l)}$ changes as training goes on. To meet PR_{goal}, it is necessary to determine the remaining sparsity ratio sr_{rmng}. Under the actual sparsity ratio $sr_{\mathrm{act}}^{(l)}$, remaining sparsity is re-allocated equally to only active channels, namely, sr_{rmng} is the same for each-layer. As training goes on, though remaining sparsity re-allocation is uniformly performed, the final sparsity ratio $sr^{(l)}$ of each layer is diverse. Finally, customized layer-wise sparsity ratios are achieved. The detailed sparsity allocation procedure is illustrated in Alg. 1. Note that we set the pruned ratio error Δ to be 0.001. An error this small is usually negligible in real-world.

4.2.3 Optimization

In this subsection, we optimize the overall objective in Eqn. 4 to solve the training problem. Based on Eqn. 4, the overall loss function $\mathcal{L}_{\mathrm{total}}$ is given by:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{KD}} + \mathcal{L}_{\mathrm{SP}}. \tag{17}$$

Namely, the goal of the proposed method is to minimize the loss function $\mathcal{L}_{\mathrm{total}}$. In this subsection, the optimization of the student network and the dean network is introduced. Then, the overall training scheme is described in the next subsection.

(1) Optimization of the student network.

The goal of the student network is to achieve the optimization in Eqn. 4, thus obtaining a compact but strong model. However, the term $\mathcal{L}_{\mathrm{SP}}$ in Eqn. 17 is not differentiable, thus it is not possible to use a gradient descent scheme for the optimization. To mitigate this problem, we introduce ADMM [48].In particular, we decompose the original optimization problem into two sub-problems and solve them iteratively. The solutions to small local subproblems are coordinated to find a solution to a large global problem. Specifically, for Eqn. 4, we rewrite the overall objective into ADMM form:

$$\begin{aligned} \min_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2} \quad & f(\boldsymbol{\theta}_1) + h(\boldsymbol{\theta}_2) \\ s.t. \quad & \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2 = 0, \end{aligned} \tag{18}$$

in which $f(\boldsymbol{\theta}_1) = \mathcal{L}_{\mathrm{CE}}(\mathcal{F}(\mathbf{\Theta}^{\mathrm{S}}, \mathbf{x}), \mathbf{y}) + \mathcal{L}_{\mathrm{KD}}(\mathbf{x}, \mathbf{\Theta}^{\mathrm{T}}, \mathbf{\Theta}^{\mathrm{S}}, \mathbf{\Phi})$ is differentiable, and $h(\boldsymbol{\theta}_2) = \mathcal{L}_{\mathrm{SP}}(\mathbf{\Theta}^{\mathrm{S}}, \mathbf{\Phi})$ is non-differentiable. Note that $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ both include the student network $\mathbf{\Theta}^{\mathrm{S}}$. We convert the original optimization problem into a constrained convex optimization problem, and decouple this problem into two subproblems (*i.e.*, $f(\boldsymbol{\theta}_1)$ and $h(\boldsymbol{\theta}_2)$). On exploiting the special form of f and h, the ADMM algorithm procedure is expressed as:

$$\boldsymbol{\theta}_1^{t+1} := \arg\min_{\boldsymbol{\theta}_1} \left(f(\boldsymbol{\theta}_1) + \frac{\rho}{2} ||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2^t + \mathbf{u}^t||_2^2 \right), \tag{19}$$

$$\boldsymbol{\theta}_2^{t+1} := \arg\min_{\boldsymbol{\theta}_2} \left(h(\boldsymbol{\theta}_2) + \frac{\rho}{2} ||\boldsymbol{\theta}_1^{t+1} - \boldsymbol{\theta}_2 + \mathbf{u}^t||_2^2 \right), \tag{20}$$

$$\mathbf{u}^{t+1} := \mathbf{u}^t + (\boldsymbol{\theta}_1^{t+1} - \boldsymbol{\theta}_2^{t+1}), \tag{21}$$

where ρ is a penalty parameter, and $\mathbf{u}$ is the scaled dual variable. The stopping criteria are:

$$||\boldsymbol{\theta}_1^{t+1} - \boldsymbol{\theta}_2^{t+1}||_2^2 \leq \epsilon, \quad ||\boldsymbol{\theta}_2^{t+1} - \boldsymbol{\theta}_2^t||_2^2 \leq \epsilon, \tag{22}$$

or the model convergence. Note that $\epsilon > 0$ is a feasible tolerance, which is commonly set to be 10^{-3}. To conduct the ADMM algorithm, the two subproblems (19) and (20) are solved. For subproblem (19), the first term $f(\boldsymbol{\theta}_1)$ is the sum of a cross entropy loss and KD loss. The second term can be regarded as an ℓ_2 regularizer. Subproblem (19) is differentiable and can be solved by using stochastic gradient decent. The gradient with respect to $\boldsymbol{\theta}_1$ is expressed as:

$$\begin{aligned} &\frac{\partial \left(f(\boldsymbol{\theta}_1) + \frac{\rho}{2} ||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2^t + \mathbf{u}^t||_2^2 \right)}{\partial \boldsymbol{\theta}_1} \\ &= \frac{\partial f(\boldsymbol{\theta}_1)}{\partial \boldsymbol{\theta}_1} + \rho \cdot (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2^t + \mathbf{u}^t). \end{aligned} \tag{23}$$

For subproblem (20), the first term $h(\boldsymbol{\theta}_2)$ is an ℓ_{21} regularizer which is not differentiable. To solve this subproblem, we compute a simple closed-form solution utilizing the subdifferential calculus. Explicitly, Eqn. 20 can be derived as:

$$\boldsymbol{\theta}_2^{t+1} := \mathrm{S}_{\beta/\rho}(\boldsymbol{\theta}_1^{t+1} + \mathbf{u}^t), \tag{24}$$

where $S_{1/\rho}$ is the soft thresholding operator. In particular, the student network parameters at the l-th layer are updated by:

$$\begin{aligned} &S_{\beta/\rho}((\mathbf{W}_{j,\,m}^{(l)})^{2\mathrm{D}}) \\ &= \begin{cases} (\mathbf{W}_{j,\,m}^{(l)})^{2\mathrm{D}} - \frac{\beta_l(j)(\mathbf{W}_{j,\,m}^{(l)})^{2\mathrm{D}}}{\rho ||(\mathbf{W}_{j,\,:}^{(l)})^{2\mathrm{D}}||_2}, \text{if } ||(\mathbf{W}_{j,\,:}^{(l)})^{2\mathrm{D}}||_2 > \frac{\beta_l(j)}{\rho}, \\ 0 \qquad\qquad\qquad\qquad\quad , \text{otherwise}. \end{cases} \end{aligned} \tag{25}$$

Note that $(\mathbf{W}_{j,\,m}^{(l)})^{2\mathrm{D}}$ is the matrix-form of the student network parameters at the l-th layer which is obtained after the computation of $(\boldsymbol{\theta}_1^{t+1} + \mathbf{u}^t)$ in Eqn. 24.

(2) Optimization of the dean network.

The goal of the dean network is to help the student network achieve high performance. As mentioned above, the outputs of the dean network are $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ which are regarded as the supervised intensities. The two parameters and the dean network $\mathbf{\Phi}$ cannot

be directly adjusted and updated by the back propagation (BP) algorithm. Because the solution converges to local optimum, *i.e.*, $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\beta} = \mathbf{0}$, if using BP algorithm. It degrades to the situation that the student is trained from scratch without any sparsity and KD constraints.

Thus, to optimize the dean network and maximize the performance of the student network, we use the bilevel optimization framework in meta learning [49], [50]. In particular, our overall loss function is rewritten as:

$$\begin{aligned}\mathcal{L}_{\text{total}}(\boldsymbol{\Theta}^{\text{S}}|\mathbf{x},\mathbf{y},\boldsymbol{\Phi}) &= \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}}|\mathbf{x},\mathbf{y}) \\ &+ \mathcal{L}_{\text{KD}}((\boldsymbol{\Theta}^{\text{S}}|\mathbf{x},\boldsymbol{\Phi})) + \mathcal{L}_{\text{SP}}((\boldsymbol{\Theta}^{\text{S}}|\mathbf{x},\boldsymbol{\Phi})).\end{aligned} \tag{26}$$

For the dean network, the objective is formulated as:

$$\begin{aligned}&\min_{\boldsymbol{\Phi}} \ \ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}}) \\ &s.t. \ \ \boldsymbol{\Theta}^{\text{S}*} = \arg\min_{\boldsymbol{\Theta}^{\text{S}}} \ \mathcal{L}_{\text{total}}(\boldsymbol{\Theta}^{\text{S}}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi}),\end{aligned} \tag{27}$$

in which $(\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}})$ and $(\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}})$ are a training sample and a validation sample in the database, respectively. From Eqn. 27, it is usual to calculate $\nabla_{\boldsymbol{\Phi}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}})$ to update $\boldsymbol{\Phi}$. However, it is hard to directly compute $\nabla_{\boldsymbol{\Phi}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}})$, because $\boldsymbol{\Theta}^{\text{S}*}$ is an implicit function of $\boldsymbol{\Phi}$ and the calculation has a huge computational cost. Based on a standard approach in meta learning [50], we obtain our solution for updating the dean network.

In detail, firstly, we update $\boldsymbol{\Theta}^{\text{S}}$ to minimize $\mathcal{L}_{\text{total}}(\boldsymbol{\Theta}^{\text{S}}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi})$:

$$\boldsymbol{\Theta}^{\text{S}*}_{t+1} := \boldsymbol{\Theta}^{\text{S}*}_{t} - \zeta_{\text{in}}\nabla_{\boldsymbol{\Theta}^{\text{S}}}\ \mathcal{L}_{\text{total}}(\boldsymbol{\Theta}^{\text{S}}_{t}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi}), \tag{28}$$

where ζ_{in} is the learning rate of the inner-loop optimization. Note the updated $\boldsymbol{\Theta}^{\text{S}*}$ in the above equation is only used to optimize $\boldsymbol{\Phi}$. It is not used in the optimization of the student network.

Then, we update $\boldsymbol{\Phi}$ to carry out the minimization in Eqn. 27. The distillability control module and the sparsability control module may influence each other and obstruct the optimization. To avoid this problem, the distillability control module and the sparsability control module are decoupled from the dean network, by dividing the dean into two parts: $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_{\text{KD}}, \boldsymbol{\Phi}_{\text{SP}}\}$ in which $\boldsymbol{\Phi}_{\text{KD}}$ is the distillability control module and $\boldsymbol{\Phi}_{\text{SP}}$ is the sparsability control module. In this manner, $\boldsymbol{\Phi}_{\text{KD}}$ and $\boldsymbol{\Phi}_{\text{SP}}$ are optimized alternately. The term $\boldsymbol{\Phi}_{\text{KD}}$ is updated to minimize $\mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}})$:

$$\boldsymbol{\Phi}^{t+1}_{\text{KD}} := \boldsymbol{\Phi}^{t}_{\text{KD}} - \zeta_{\text{out}}\nabla_{\boldsymbol{\Phi}_{\text{KD}}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}}). \tag{29}$$

Note that ζ_{out} is the learning rate of the outer-loop optimization. To summarize, Eqn. 28 is used in the inner-loop of the meta learning, while Eqn. 29 is used in the outer-loop in which $\mathcal{L}_{\text{CE}}$ is regarded as the meta objective to help learn a better student network. Likewise, for $\boldsymbol{\Phi}_{\text{SP}}$, the updated step is:

$$\boldsymbol{\Phi}^{t+1}_{\text{SP}} := \boldsymbol{\Phi}^{t}_{\text{SP}} - \zeta_{\text{out}}\nabla_{\boldsymbol{\Phi}_{\text{SP}}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}*}|\mathbf{x}^{\text{val}},\mathbf{y}^{\text{val}}). \tag{30}$$

Similar to [26], the proposed dean network $\boldsymbol{\Phi}$ has only a small influence on $\mathcal{L}_{\text{CE}}$, through the terms $(\mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{SP}})$. Hence, it is difficult to update $\boldsymbol{\Phi}$ using the single gradient descent step in Eqn. 29 and Eqn. 30. To increase the influence of $\boldsymbol{\Phi}$, we update $\boldsymbol{\Theta}^{\text{S}*}$ T times and modify the update step of Eqn. 28 to:

$$\begin{aligned}&\boldsymbol{\Theta}^{\text{S}*}_{T+1} := \boldsymbol{\Theta}^{\text{S}*}_{T} - \zeta_{\text{in}}\nabla_{\boldsymbol{\Theta}^{\text{S}}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}}_{T}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi}), \\ &\boldsymbol{\Theta}^{\text{S}*}_{t+1} := \boldsymbol{\Theta}^{\text{S}*}_{t} - \zeta_{\text{in}}\nabla_{\boldsymbol{\Theta}^{\text{S}}}\ \Big(\mathcal{L}_{\text{KD}}(\boldsymbol{\Theta}^{\text{S}}_{t}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi}) \\ &\qquad + \mathcal{L}_{\text{SP}}(\boldsymbol{\Theta}^{\text{S}}_{t}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi})\Big). \\ &(t = 0, 1, ..., T-1)\end{aligned} \tag{31}$$

Note that we set $T = 2$ in our experiments. Here, the $\boldsymbol{\Theta}^{\text{S}*}$ update similarly follows the optimization of the student network proposed in Eqns. 19, 20 and 21.

Algorithm 2: The procedure of DDSL.

Input: Database $\mathcal{D}^{\text{train}} = \{\mathbf{x}^{\text{train}}, \mathbf{y}^{\text{train}}\}$, $\boldsymbol{\Theta}^{\text{S}}$, $\boldsymbol{\Theta}^{\text{T}}$, $\boldsymbol{\Phi}$, target pruned ratio PR_{goal}.

For each layer l, initialize the sparsity allocation ratio $sr^{(l)} = sr^{(l)}_{\text{act}} + sr_{\text{rmng}}$, where $sr^{(l)}_{\text{act}} = 0$ and $sr_{\text{rmng}} = PR_{\text{goal}}$ initially;
Initialize the "unimportant" channel set $\mathcal{C}^{(l)}_{\text{sel}} = \varnothing$;
Initialize $\boldsymbol{\Theta}^{\text{S}}$ and $\boldsymbol{\Phi}$;
repeat
  Update $\boldsymbol{\Theta}^{\text{S}}$ using Eqn. 19, 20, 21;
  For each layer l, calculate the importance score of each channel: $I(\mathcal{W}^{(l)}_j), j = 1, 2, ..., c_l$ via Eqn. 15;
  Update spasity allocation ratio $sr^{(l)}$ for each layer l via Alg. 1.
  For each layer l, sort and truncate the smallest-$\lceil sr^{(l)} * c_l \rceil$ of I to identify $\mathcal{C}^{(l)}_{\text{sel}}$;
  for $t = 0$ *to* $T-1$ **do**
    Update $\boldsymbol{\Theta}^{\text{S}*}_t$ using Eqn. 31 and Eqn. 19, 20, 21;
  end
  $\boldsymbol{\Theta}^{\text{S}*}_{T+1} \leftarrow \boldsymbol{\Theta}^{\text{S}*}_{T} - \zeta_{\text{in}}\nabla_{\boldsymbol{\Theta}^{\text{S}}}\ \mathcal{L}_{\text{CE}}(\boldsymbol{\Theta}^{\text{S}}_{T}|\mathbf{x}^{\text{train}},\mathbf{y}^{\text{train}},\boldsymbol{\Phi})$ (using Eqn. 31);
  Update $\boldsymbol{\Phi}$ using Eqn. 29, 30;
  For each layer l, recalculate $sr^{(l)}$;
until *done*;
Prune the redundant filters ($\left\|\mathbf{W}^{(l)}_{j,:}\right\|_2 = 0$) and return the compressed network with acceptable accuracy.

4.2.4 *Training procedure*

The overall training procedure of the proposed DDSL is summarized in Alg. 2. In detail, after a series of initializations, we update the student network using GT supervision, KD supervision and sparsity supervision. To dynamically impose the sparsity constraint, a set of "unimportant" channels with an allocated sparsity ratio are selected to be restricted. Then, the dean network is updated using the proposed meta scheme. After a period of training, the training procedure stops when the student network has converged or the stop condition (*i.e.*, Eqn. 22) has been satisfied. The channels with zero parameters are pruned and a final compressed model with acceptable performance is obtained.

5 Experiments and Results

5.1 Settings

Databases and Networks. We evaluate the proposed method on two databases: CIFAR [58] and ImageNet [59]. The data are

TABLE 1
The results of pruning ResNet56 and ResNet110 on CIFAR-10. The term "Pre-defined?" refers to whether a fixed pruned ratio is assigned to each layer. The "--" indicates that the results are not reported in the original paper.

Model Architecture	Method	Training Settings		Test Accuracy	PR_{Param}	PR_{FLOPs}
		Pre-defined?	Fine-tuning?			
ResNet56	Baseline	N/A	N/A	(93.81±0.14%)	0	0
	Li *et al.* [28]	✔	✔	93.06%	13.70%	27.60%
	CP [34]	✔	✔	91.80%	--	50.00%
	AMC [39]	✗	✔	91.90%	--	50.00%
	HRank [30]	✗	✔	93.17%	42.40%	50.00%
	SFP [31]	✔	✗	(92.26±0.31)%	--	52.60%
	FPGM [29]	✔	✗	(92.93±0.49)%	--	52.60%
	ASFP [32]	✔	✗	(93.12±0.20)%	--	52.60%
	DPFPS [47]	✗	✗	(93.20±0.11)%	46.84%	52.86%
	DCP [51]	✗	✔	93.49%	49.24%	49.75%
	CAC [52]	✗	✗	93.48%	29.43%	30.16%
	ResRep [53]	✗	✗	93.71%	--	52.91%
	EagleEye [54]	✗	✔	**94.66%**	--	50.40%
	Logits [5]	✔	✗	92.63%	43.50%	43.68%
	AT [17]	✔	✗	92.76%	43.50%	43.68%
	IRG [19]	✔	✗	92.97%	43.50%	43.68%
	MetaDistiller [27]	✔	✗	92.78%	43.50%	43.68%
	VID [55]	✔	✗	92.69%	43.50%	43.68%
	CRD [56]	✔	✗	92.91%	43.50%	43.68%
	SAD [57]	✔	✗	92.96%	43.50%	43.68%
	DPFPS [47] + IRG [19]	✗	✔	93.57%	45.62%	50.27%
	GSKD [10]	✗	✗	92.45%	62.75%	69.28%
	FSKD [11]	✔	✗	92.44%	72.91%	73.00%
	FSCD [12]	✗	✗	92.78%	72.91%	73.00%
	Ours	✗	✗	(94.05±0.09)%	47.89%	56.99%
	Ours	✗	✗	93.98%	**74.98%**	**75.89%**
ResNet110	Baseline	N/A	N/A	(94.69±0.18%)	0	0
	Li *et al.* [28]	✔	✔	93.30%	32.40%	38.60%
	HRank [30]	✗	✔	92.65%	68.70%	68.60%
	SFP [31]	✔	✗	(93.38±0.30)%	--	40.80%
	FPGM [29]	✔	✗	(93.85±0.11)%	--	52.30%
	ASFP [32]	✔	✗	(93.10±0.06)%	--	52.30%
	DPFPS [47]	✗	✗	(92.61±0.17)%	74.32%	74.36%
	CAC [52]	✗	✗	93.54%	52.31%	51.15%
	Logits [5]	✔	✗	92.38%	74.78%	74.96%
	AT [17]	✔	✗	92.23%	74.78%	74.96%
	IRG [19]	✔	✗	92.65%	74.78%	74.96%
	MetaDistiller [27]	✔	✗	92.57%	74.78%	74.96%
	VID [55]	✔	✗	92.49%	74.78%	74.96%
	CRD [56]	✔	✗	92.46%	74.78%	74.96%
	SAD [57]	✔	✗	92.71%	74.78%	74.96%
	DPFPS [47] + IRG [19]	✗	✔	92.85%	71.67%	72.38%
	Ours	✗	✗	**(93.87±0.35)%**	**75.13%**	**75.31%**

augmented using the same strategies as in the PyTorch official examples [60]. On the CIFAR database, we evaluate the proposed method using VGG network [61] and ResNets [62]. As the original VGG-16 is specially designed for ImageNet classification, we use a variant version (*i.e.*, VGG-Small) taken from [63] in our experiments with CIFAR. On the ImageNet database, we evaluate our model on ResNets (including ResNet 34, 50, and 101) and MobileNet v2 [43]. In addition, in ablation analysis, we analyze the proposed method by using ResNet20 evaluated on CIFAR10.

Evaluation Metrics. To evaluate the performance of different model compression methods, we use the parameters pruned ratio,

$$PR_{\text{Param}} = 1 - \frac{\#Param_{\text{remain}}}{\#Param_{\text{total}}}, \tag{32}$$

where $\#Param_{\text{remain}}$ is the number of remaining parameters. The term $\#Param_{\text{total}}$ is the total number of parameters in the original network. We also use the FLoating-point OPerations (FLOPs) pruned ratio,

$$PR_{\text{FLOPs}} = 1 - \frac{\#FLOP_{\text{remain}}}{\#FLOP_{\text{total}}}, \tag{33}$$

where $\#FLOP_{\text{remain}}$ is the number of remaining FLOPs after the model is pruned. Besides, $\#FLOP_{\text{total}}$ represents the total number of FLOPs in the original network.

Implementation Details. We train all the networks from scratch. For CIFAR-10 and CIFAR-100, the total number of epochs is 200 with a standard batch size of 64. The inner loop learning rate ζ_{in} is initialized as 0.1 and multiplied by 0.1 at epoch 100 and epoch 150, while the outer loop learning rate ζ_{out} is set to be 0.1 times the inner loop learning rate, as suggested empirically. For

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

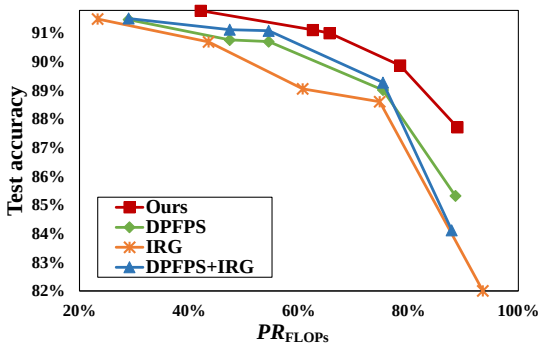


Fig. 10. Accuracy curves at different pruned ratios. The proposed method works best among different compression methods.

ImageNet, the total number of epochs is 120 with a 256 batch size. The inner loop learning rate ζ_{in} is initialized as 0.1 and multiplied by 0.1 at epoch 30, epoch 60 and epoch 90, respectively, while the outer loop learning rate ζ_{out} is 0.1 times the inner loop learning rate. A standard stochastic gradient descent (SGD) optimizer with 10^{-4} weight decay and 0.9 momentum is adopted. All the experiments are conducted on a platform with 8 Nvidia Tesla GPU cards and 96-core Intel(R) Xeon(R) Platinum 8163 CPU. In addition, every single setting is repeated 10 times with different random seeds on Pytorch.

5.2 Comparison with the state-of-the-art

We compare the performance of our method with 24 state-of-the-art model compression methods, including Li *et al.* [28], HRank [30], SFP [31], CP [34], AMC [39], FPGM [29], ASF-P [32], IE [64], DPFPS [47], ResRep [53], EagleEye [54], DMCP [65], DCP [51], CAC [52], GSKD [10], FSKD [11], FSCD [12], Logits [5], AT [17], IRG [19], MetaDistiller [27], VID [55], CRD [56] and SAD [57]. Among them, Li *et al.*, IE, HRank, SFP, CP, AMC, FPGM, ASFP, DPFPS, ResRep, EagleEye, DCP and CAC are sparsity pruning methods. GSKD, FSKD, FSCD and DPFPS+IRG are methods using both knowledge distillation and pruning. Others are knowledge distillation methods. Note that "DPFPS+IRG" denotes the model first pruned via DPFPS (*i.e.*, a state-of-the-art channel pruning method) and then trained with IRG (*i.e.*, a state-of-the-art KD method). For a fair comparison, we adopt a pre-defined compression model as the student network for knowledge distillation methods, because it has a similar pruned ratio as sparsity pruning methods with which it is compared. For example, in Tab. 1, ResNet56-x0.75 is the student network, since it has a similar FLOPs pruned ratio as the competing pruning methods.

Evaluation on CIFAR. Tab. 1 presents the model compressed results of test accuracy, pruned ratio of PR_{Param} and PR_{FLOPs} on CIFAR 10. As shown in this table, the proposed method has the best performance among almost all the methods, including the sparsity pruning methods, the knowledge distillation methods and the methods using both pruning and knowledge distillation. In particular, for ResNet56, our compressed model achieves an accuracy of 94.05% with a 56.99% pruned ratio of FLOPs. The performance even surpasses that of the baseline model. With a higher pruned ratio, namely, 75.89% pruned ratio of FLOPs, the accuracy of the proposed method only degrades slightly and is still higher than that of the baseline. For the large DNN ResNet110, our method performs the best. It surpasses DPFPS, which achieves the highest pruned ratio among all the compared methods, by 1.07% test accuracy. For traditional DNN VGG-Small, our method prunes VGG-Small more than 94.27% parameters and 74.19% FLOPs, with a slight accuracy loss (*i.e.*, less than 0.09%). It also achieves the best performance. Due to the space limitation, the detailed experimental results of VGG and other results are reported in the *supplemental material*.

Further, Fig. 10 evaluates the proposed DDSL at different pruned ratios, compared with IRG (*i.e.*, a KD state-of-the-art method), DPFPS (*i.e.*, a sparsity pruning state-of-the-art method) and a direct combination of them. It is obvious that the proposed DDSL surpasses the compared methods at all the assigned values of pruned ratio. When PR_{FLOPs} exceeds 80%, the test accuracies of all the compared methods drop sharply. In contrast, the graph of our method has a steady trend. At 90% pruned ratio, the proposed DDSL even surpasses the best of the compared method by $\sim$3% test accuracy. The direct combination of KD and sparsity pruning "DPFPS+IRG" (*i.e.*, ResNet20 is first pruned via DPFPS and then the pruned network is trained by IRG with a pre-trained ResNet20 teacher network) performs similar to IRG. It indicates that the performance gain of a direct combination is small. On the contrary, via joint dynamic optimization, the proposed method outperforms "DPFPS+IRG" by a large margin, which verifies the effectiveness of our dynamically distillability-and-sparsability learning scheme.

Evaluation on ImageNet. On the large-scale challenging database ImageNet, our method also performs the best as reported in Tab. 2, in comparison with sparsity pruning methods and KD methods. For compression performance, the proposed method not only achieves the highest pruned ratio, but also maintains the highest accuracy. In particular, our method has the best performance for all three network architectures with different depths (*i.e.*, ResNet34, 50 and 101). It even surpasses the baseline for ResNet101. When compressing a lightweight network (*i.e.*, MobileNet v2), the proposed method is also superior to the compared methods. Our method has an efficient training scheme which does not need a pre-defined compressed architecture, fine-tuning steps or multi-pass steps. In contrast, nearly all KD methods need to pre-define the student architecture. Some sparsity pruning methods need a pre-defined network or fine-tuning to enhance the accuracy. DPFPS, ResRep and CAC do not need pre-defining and fine-tuning, but it has inferior performance compared with our method. d DPFPS+IRG method, which directly combines pruning and knowledge distillation, has a high pruned ratio but is still worse than the proposed method for both performance and pruned ratio.

These experimental results indicate that the proposed method is effective and efficient, and is superior to almost all the compared methods on both small-scale and large-scale databases when compressing various DNNs with different depths and architectures, with the help of the proposed dynamically distillability-and-sparsability learning scheme.

5.3 Ablation analysis

(1) Analysis of distillability and sparsability. It is hard to accurately calculate distillability and sparsability, but the learned supervision control signals (*i.e.*, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) do include information about distillability and sparsability. Fig. 11 shows the curves of supervision control signals at different layers during training. These graphs show that as distillability and sparsability change during training, the supervision control signals are adaptively adjusted at different layers. For distillability, the supervision control signals $\boldsymbol{\alpha}$ increase in the deeper layers, especially at the last convolutional layer and the FC layer. This indicates that deeper layers have

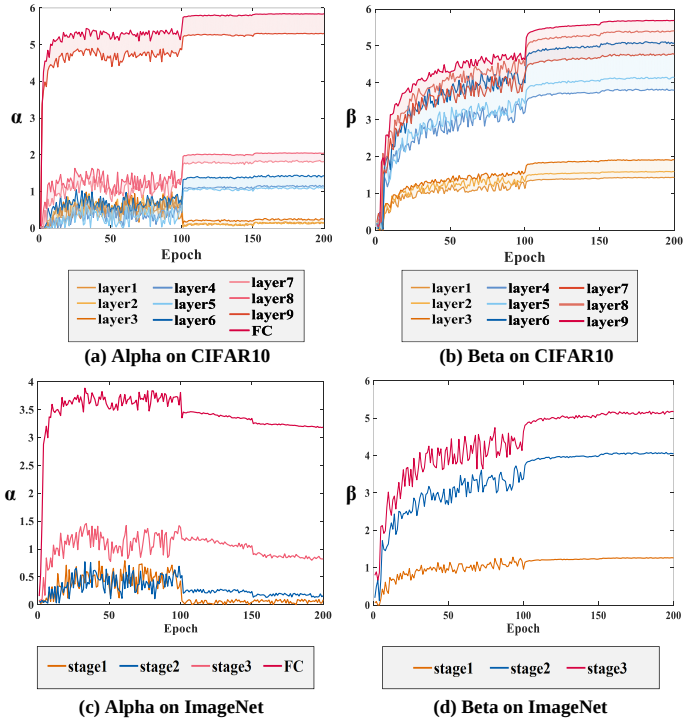
This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

TABLE 2
The results of pruning ResNets and MobileNet v2 on ImageNet.

Model Architecture	Method	Training Settings		Test Accuracy		PR_{FLOPs}	Model Architecture	Method	Training Settings		Test Accuracy		PR_{FLOPs}
		Pre-defined?	Fine-tuning?	Top-1	Top-5				Pre-defined?	Fine-tuning?	Top-1	Top-5	
ResNet34	Baseline	N/A	N/A	73.92%	91.62%	0	ResNet50	Baseline	N/A	N/A	76.15%	92.87%	0
	SFP [31]	✔	✗	71.83%	90.33%	41.10%		SFP [31]	✔	✗	74.61%	92.06%	41.80%
	FPGM [29]	✔	✗	72.11%	90.69%	41.10%		FPGM [29]	✔	✗	75.03%	92.40%	42.20%
	ASFP [32]	✔	✗	71.72%	90.65%	41.10%		ASFP [32]	✔	✗	74.88%	92.39%	41.80%
	DPFPS [47]	✗	✗	72.25%	90.80%	43.29%		DPFPS [47]	✗	✗	75.55%	92.54%	46.20%
	Li *et al.* [28]	✔	✔	72.17%	−−	24.20%		HRank [30]	✗	✔	74.98%	92.33%	43.77%
	IE [64]	✔	✔	72.83%	−−	24.20%		IE [64]	✔	✔	74.50%	−−	45.00%
	DMCP [65]	✗	✔	72.96%	90.99%	39.87%		DMCP [65]	✗	✔	75.67%	92.58%	38.34%
	EagleEye [54]	✔	✔	72.87%	90.94%	39.37%		EagleEye [54]	✔	✔	75.73%	92.59%	42.07%
	ResRep [53]	✗	✗	73.02%	91.02%	41.65%		CAC [52]	✗	✗	75.79%	92.35%	**55.16%**
	Logits [5]	✔	✗	72.42%	90.90%	28.38%		Logits [5]	✔	✗	74.86%	92.19%	28.43%
	AT [17]	✔	✗	72.38%	90.88%	28.38%		AT [17]	✔	✗	74.91%	92.29%	28.43%
	IRG [19]	✔	✗	72.73%	90.99%	28.38%		IRG [19]	✔	✗	75.07%	92.27%	28.43%
	MetaDistiller [27]	✔	✗	72.58%	90.92%	28.38%		MetaDistiller [27]	✔	✗	74.78%	92.11%	28.43%
	VID [55]	✔	✗	72.56%	90.92%	28.38%		VID [55]	✔	✗	74.99%	92.30%	28.43%
	CRD [56]	✔	✗	72.71%	90.99%	28.38%		CRD [56]	✔	✗	75.08%	92.38%	28.43%
	SAD [57]	✔	✗	72.67%	90.97%	28.38%		SAD [57]	✔	✗	75.12%	92.39%	28.43%
	DPFPS+IRG	✗	✔	72.53%	90.89%	42.56%		DPFPS+IRG	✗	✔	75.69%	92.58%	47.38%
	Ours	✗	✗	**73.26%**	**91.09%**	**49.14%**		**Ours**	✗	✗	**76.02%**	**92.74%**	50.01%
ResNet101	Baseline	N/A	N/A	77.37%	93.56%	0	MobileNet v2	Baseline	N/A	N/A	72.00%	90.65%	0
	DPFPS [47]	✗	✗	77.27%	93.68%	44.97%		DPFPS [47]	✗	✗	71.10%	89.87%	24.89%
	SFP [31]	✔	✗	77.03%	93.46%	42.20%		AMC [39]	✗	✔	70.80%	−−	26.54%
	FPGM [29]	✔	✔	77.32%	93.56%	42.20%		MobileNet v2 0.75x [43]	✔	✗	69.80%	88.97%	26.54%
	IE [64]	✔	✔	77.35%	−−	39.80%							
	DMCP [65]	✗	✔	76.98%	93.36%	43.27%		DMCP [65]	✗	✔	71.11%	89.61%	29.67%
	EagleEye [54]	✔	✔	76.92%	93.32%	42.81%		EagleEye [54]	✔	✔	71.18%	89.64%	30.00%
	ResRep [53]	✗	✗	77.02%	93.44%	44.39%		DCP [51]	✗	✔	64.22%	85.95%	44.75%
	Logits [5]	✔	✗	76.33%	93.01%	28.61%		Logits [5]	✔	✗	70.68%	89.65%	27.02%
	AT [17]	✔	✗	76.21%	92.93%	28.61%		AT [17]	✔	✗	70.51%	89.57%	27.02%
	IRG [19]	✔	✗	76.57%	93.16%	28.61%		IRG [19]	✔	✗	70.83%	89.69%	27.02%
	MetaDistiller [27]	✔	✗	76.36%	93.03%	28.61%		MetaDistiller [27]	✔	✗	70.72%	89.71%	27.02%
	VID [55]	✔	✗	76.62%	93.19%	28.61%		VID [55]	✔	✗	70.75%	89.72%	27.02%
	CRD [56]	✔	✗	76.78%	93.30%	28.61%		CRD [56]	✔	✗	70.65%	89.62%	27.02%
	SAD [57]	✔	✗	76.91%	93.39%	28.61%		SAD [57]	✔	✗	70.59%	89.59%	27.02%
	DPFPS+IRG	✗	✔	77.31%	93.53%	46.28%		DPFPS+IRG	✗	✔	71.15%	89.98%	28.32%
	Ours	✗	✗	**77.38%**	**93.81%**	**50.96%**		**Ours**	✗	✗	**71.37%**	**90.42%**	**32.37%**

higher distillability and that there is more useful knowledge to be distilled for these layers. In addition, as shown in Fig. 11-(c), $\boldsymbol{\alpha}$ decreases at the end of training, which indicates that distillability may decrease during training for the same challenge databases (*e.g.*, ImageNet). It is consistent with our *Observation 3* for ImageNet. Fortunately, by properly mining the distilled knowledge, our method slows the downward trend of distillability and has a satisfactory performance. For sparsability, deeper layers have relatively higher supervision intensities, because there are more redundant channels, *i.e.*, higher sparsability. This phenomenon is consistent with our sparsity allocation results. Besides, other than a sudden network structure change as traditional channel pruning does [33], [34], the proposed supervision control signal $\boldsymbol{\beta}$ increases, making the student network to be sparse progressively. As a result, the student network fits the data in the early stages of the training, making it is less likely that the optimization yields a sub-optimal network.

(2) Effectiveness of different components. We thoroughly analyze the effectiveness of each component of the proposed method. Tab. 3 reports the performance of different components in our method using ResNet20 tested on CIFAR10. In this table, "Ours-KD" refers to our method of training a pre-defined and fixed student network, while "Ours-SP" refers to our method without knowledge guidance from a teacher. As reported, "Ours-KD" and "Ours-SP" both outperform state-of-the-art KD and SP methods by a significant margin. The effectiveness of our dean component is verified. With a layer-wise dynamic supervision intensity adjustment scheme, the performance is improved compared with a predefined supervision intensity scheme (*e.g.*, IRG and DPFPS). Further, the combination of all components (*i.e.*, the proposed DDSL framework) shows the best performance. The student baseline is surpassed by a 1.94% accuracy with a higher pruned ratio. The feeding of the student information and the teacher knowledge into the dean ensures that the proposed DDSL automatically balances the importance of KD and SP during training. As a result, our dean gives better supervision control signals than those obtained from



(a) Alpha on CIFAR10 (b) Beta on CIFAR10

(c) Alpha on ImageNet (d) Beta on ImageNet

Fig. 11. The graphs of supervision control signals at different layers during training. Note that (a) and (c) are the graphs of supervision control signals for distillability from different layers. The graphs of (b) and (d) are the supervision control signals for sparsability from different layers. The point values on the lines are the sum of the elements in β_l.

TABLE 3
Performance of different components in our method.

Methods	Accuracy	PR_{Param}	PR_{FLOPs}
Baseline $\mathbf{\Theta}^{\text{T}}$	91.97%	0	0
Baseline $\mathbf{\Theta}^{\text{S}}$	88.43%	74.23%	74.73%
IRG [19]	89.35%	74.23%	74.73%
DPFPS [47]	89.03%	74.88%	76.86%
Ours-KD	89.68%	74.23%	74.73%
Ours-SP	89.32%	75.09%	77.27%
Ours-separately	89.57%	75.09%	77.27%
Ours	**90.37%**	74.99%	76.97%

* Note that "Baseline $\mathbf{\Theta}^{\text{T}}$" and "Baseline $\mathbf{\Theta}^{\text{S}}$" represent the teacher network and the student network trained from scratch, respectively. "Ours-KD" and "Ours-SP" are the proposed method using only KD and only structured sparsity pruning, respectively. Besides, "Ours-separately" denotes our method optimized separately (*i.e.*, use structured sparsity pruning first and then train with KD).

methods which use single component.

(3) Joint Optimization Versus Separate Optimization. The proposed DDSL optimize Eqn. 17 jointly, with KD component (*i.e.*, $\mathcal{L}_{\text{KD}}$) and sparsity pruning component (*i.e.*, $\mathcal{L}_{\text{SP}}$). To evaluate the effectiveness of our dynamically joint optimization scheme, we compare it with separate optimization "Ours-separately". For this method, we first use our dynamic sparsability learning component "Ours-SP" to obtain a pruned model. Then this pruned model is trained utilizing our dynamic distillability learning component, "Ours-KD". As reported in Tab. 3, the test accuracy of joint optimization is 0.8% higher than that of separate optimization, with a similar pruned ratio. Fig. 12 shows that our joint optimization is much better than separate optimization at different pruned ratios. Joint optimization is an end-to-end framework with fewer hyper-

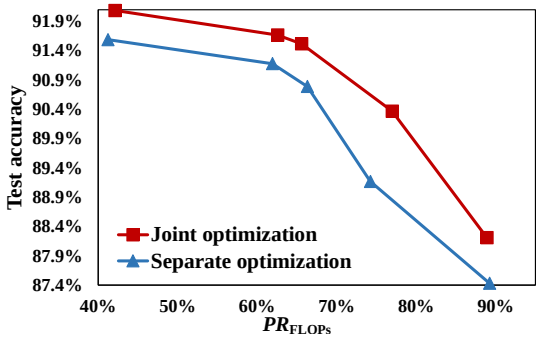


Fig. 12. Accuracy curves at different compression ratios of joint optimization and separate optimization.

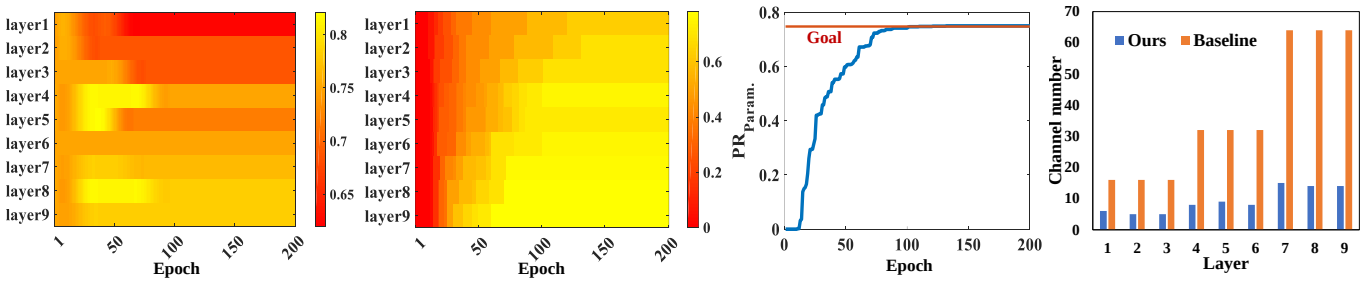
parameters, which makes DDSL a more efficient and effective method.

(4) Analysis of progressively layer-wise sparsity allocation. The proposed progressively layer-wise sparsity allocation aims to distribute proper sparsity ratio for each layer in a progressive manner. It ensures that DDSL obtains a compact and high-performance model architecture. Fig. 13-(a) and Fig. 13-(b) show the layer-wise allocated sparsity ratio $sr^{(l)}$ computed by the proposed method and the actual sparsity ratio $sr^{(l)}_{\text{act}}$ after training, respectively. For both sub-figures, the deeper layers have larger sparsity ratios, which demonstrates that the deeper layers have more redundant channels. The actual sparsity ratios change as the allocated ratios are progressively adjusted. After training, the actual sparsity ratio in each layer is equal to the final allocated ratio. This indicates that DDSL is able to achieve the target sparsity ratio, while many previous methods can not accurately control the final sparsity ratio by simply changing the sparsity supervision intensity. Fig. 13-(c) shows the overall sparsity ratio changes compared with the target ratio. With the dynamic layer-wise sparsity allocation control, the overall sparsity ratio can easily approach the target ratio. Consequently, a more compact architecture, compared with baseline, is obtained as shown in Fig. 13-(d). The above analysis fully verifies the superiority of the proposed progressively layer-wise sparsity allocation algorithm.

(5) Training efficiency analysis. Besides the number of parameters and FLOPs, we also analyze the convergent of the proposed method during training. Fig. 14 compares the training loss and test accuracy curves of different methods. As shown in the figure, the proposed DDSL has the fastest convergence and the best performance (even faster than the baseline). It is interesting to note that DPFPS, with a manual pre-defined sparsity constraint adjustment scheme (*i.e.*, the constraint strength changes following a fixed function), has a suddenly steep performance drop at around epoch 50. It is because the sparsity constraint strength increases sharply from zero to a large value around epoch 50. This leads to a sudden network structure change and performance drop. It takes more than 50 epochs to recover the performance. In contrast, the performance of DDSL increases steadily during the whole training process, since the constraint strength is dynamically adjusted by the dean in a progressive manner. As a result, the problem of DPFPS is significantly alleviated for DDSL and the final performance is improved by a large margin.

(6) Applications on other tasks. The proposed DDSL can be easily generalized to other tasks. As shown in Tab. 4, DDSL is evaluated on three visual tasks including object detection [66], instance segmentation [67] and liveness detection [68]. For detection and segmentation, three popular backbones are used,

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317



(a) Sparsity ratio (b) Actual sparsity ratio (c) Overall actual sparsity ratio (d) Channel number

Fig. 13. Analysis of progressively layer-wise sparsity allocation.

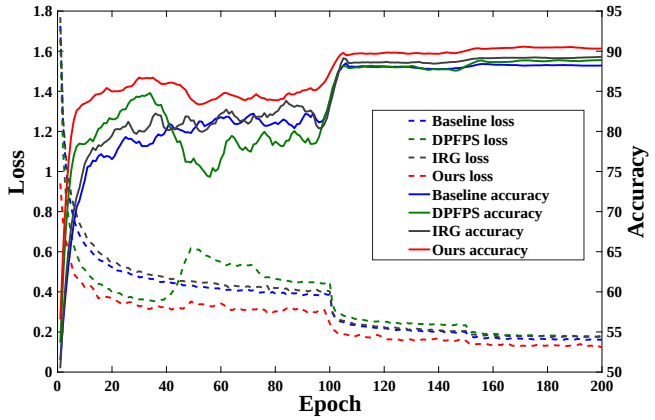


Fig. 14. Analysis of training convergence. Note that "Baseline" represents the pruned network (*i.e.*, student network) trained from scratch.

namely, ResNet50, ResNet101 and ResNext101-64x4d. We follow the recent protocol of COCO database [69] and report the standard COCO metric average precision (AP). Note that AP in segmentation is computed using mask intersection over union (IoU). According to the experimental results in Tab. 4, under similar performance degradation, DDSL shows 8%-10% more FLOPs compression compared with DPFPS+IRG. In addition, for liveness detection, which is a binary classification task, we adopt ResNet18, Inception-v3 and ResNext26 as backbones. Equal Error Rate (EER) is reported as the evaluation metric. And the experiments are conducted on CelebA-Spoof [68], which is one of the largest datasets for liveness detection. From Tab. 4, DDSL performs similar or equal compared with DPFPS+IRG but obtains 10%-13% more FLOPs compression. It is interesting to notice that, though DDSL is designed for classification task, it has good generalization when it is directly applied to other tasks such as detection and segmentation.

(7) The stability of the framework. The performance fluctuation under different hyper-parameter configurations is analyzed to verify the stability of the proposed framework. In detail, we use different hyper-parameters, including the batch size and the outer loop learning rate (*i.e.*, the learning rate for the dean network), to evaluate the proposed DDSL. The pruning results of ResNet and MobileNet on ImageNet are depicted in Fig. 15. As is seen, under different learning rates and batch sizes, the Top-1 accuracy fluctuates within the range of 0.4% and the pruned ratios have a 2% averaged fluctuation range. It depicts that both the performance and the pruned ratio stay steady when the hyper-parameters vary.

TABLE 4
Evaluation on other visual tasks, including object detection, instance segmentation and liveness detection.

Task	Backbone Architecture	Method	AP	PR_{FLOPs}
Object Detection	ResNet50	Baseline	34.5	0
		DPFPS+IRG	33.8	27.98%
		Ours	34.0	36.77%
	ResNet101	Baseline	37.1	0
		DPFPS+IRG	36.5	30.27%
		Ours	36.6	38.13%
	ResNeXt101-64x4d	Baseline	39.2	0
		DPFPS+IRG	38.6	26.97%
		Ours	38.6	33.82%
Task	Backbone Architecture	Method	AP	PR_{FLOPs}
Instance Segmentation	ResNet50	Baseline	32.6	0
		DPFPS+IRG	31.7	25.71%
		Ours	31.7	32.88%
	ResNet101	Baseline	33.9	0
		DPFPS+IRG	33.0	26.87%
		Ours	33.1	34.26%
	ResNeXt101-64x4d	Baseline	35.1	0
		DPFPS+IRG	34.9	23.79%
		Ours	35.0	32.17%
Task	Backbone Architecture	Method	EER	PR_{FLOPs}
Liveness Detection	ResNet18	Baseline	1.6	0
		DPFPS+IRG	1.7	29.62%
		Ours	1.7	37.34%
	Inception-v3	Baseline	1.4	0
		DPFPS+IRG	1.5	39.69%
		Ours	1.5	52.83%
	ResNeXt26	Baseline	1.3	0
		DPFPS+IRG	1.3	36.58%
		Ours	1.3	48.32%

6 Conclusion

In this paper, we proposed a new method for model compression, which simultaneously considers model accuracy and model size. In particular, we first revisited model compression and analyzed the factors that influence the performance of model compression. Specifically, we found that there are two attributes for model compression: distillability and sparsablity. Distillability indicates how much useful knowledge can be extracted from a teacher network. Sparsablity indicates the extent to which the model can be pruned. By exploring distillability and sparsablity, a guide for model compression during training can be obtained. Inspired by our observations, we proposed a novel dynamically distillability-and-sparsablity learning framework (DDSL), comprising a teacher, a student and a dean. The teacher guides the student using the distilled knowledge, while the student

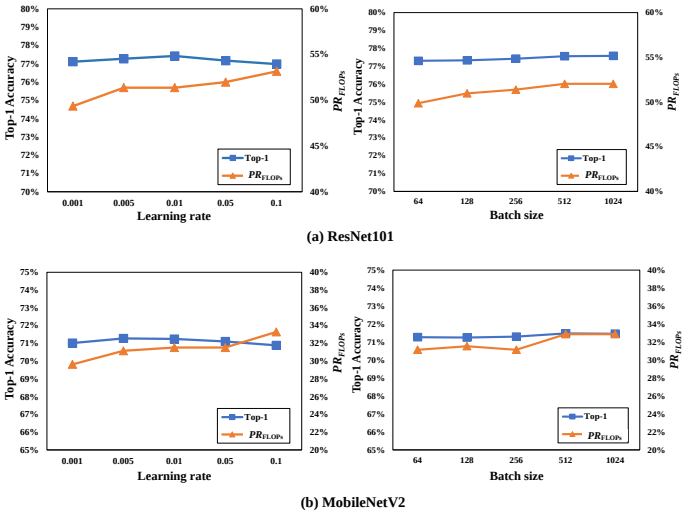


Fig. 15. The test Top-1 accuracy curves and pruned ratio PR_{FLOPS} curves at different outer loop learning rates ζ_{out} and batch sizes. Note that the inter loop learning rate ζ_{in} is commonly set to 0.1. (a) The curves of pruning ResNet101 on ImageNet. (b) The curves of pruning MobileNetV2 on ImageNet.

learns to become more accurate and more compact. In addition, the dean controls the whole learning process. In this manner, the supervisions are dynamically adjusted and a good balance is achieved. In order to optimize the proposed framework, an ADMM-based knowledge distillation-with-pruning (KDP) joint optimization algorithm was presented for updating the model. The joint optimization improves the accuracy and the pruned ratio of our method. Finally, experimental results showed that our method outperforms 24 state-of-the-art methods in terms of accuracy and pruned ratio on both small-scale and large-scale databases.

We foresee three directions for future research in this area. First, it would be promising to extend our method to data-free situation. Second, the acceleration and simplification of the training procedure is another future work, for making it more convenient and efficient. Third, in addition to the convolutional neural network, it is interesting to extend our method to other architectures, such as Transformer [70].

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2020AAA0106800), the Natural Science Foundation of China (Grant No.61902401, No. 62192785, No. 61972071, No. U1936204, No. 62122086, No. 62036011, No. 62192782 and No. 61721004), the Beijing Natural Science Foundation No. M22005, the CAS Key Research Program of Frontier Sciences (Grant No. QYZDJ-SSW-JSC040). The work of Bing Li was also supported by the Youth Innovation Promotion Association, CAS.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017. 1

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. 1

[3] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4534–4542. 1

[4] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *arXiv preprint arXiv:1608.03665*, 2016. 1, 3

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. 1, 2, 3, 4, 5, 11, 12, 13

[6] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 1943–1955, 2015. 1

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017. 1

[8] L. Wang and K.-J. Yoon, "Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[9] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018. 1, 8

[10] J. Cho and M. Lee, "Building a compact convolutional neural network for embedded intelligent sensor systems using group sparsity and knowledge distillation," *Sensors*, vol. 19, no. 19, p. 4307, 2019. 1, 3, 11, 12

[11] T. Li, J. Li, Z. Liu, and C. Zhang, "Few sample knowledge distillation for efficient network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 639–14 647. 1, 3, 11, 12

[12] H. Bai, J. Wu, I. King, and M. Lyu, "Few shot network compression via cross distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3203–3210. 1, 3, 11, 12

[13] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017. 2

[14] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279. 2

[15] L. J. Ba and R. Caruana, "Do deep nets really need to be deep?" *arXiv preprint arXiv:1312.6184*, 2013. 2

[16] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014. 2

[17] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016. 2, 7, 11, 12, 13

[18] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," *arXiv preprint arXiv:1707.01219*, 2017. 2

[19] Y. Liu, J. Cao, B. Li, C. Yuan, W. Hu, Y. Li, and Y. Duan, "Knowledge distillation via instance relationship graph," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7096–7104. 2, 7, 11, 12, 13, 14

[20] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1365–1374. 2

[21] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141. 2

[22] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328. 3

[23] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3430–3437. 3

[24] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3713–3722. 3

[25] S. Yun, J. Park, K. Lee, and J. Shin, "Regularizing class-wise predictions via self-knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 876–13 885. 3

[26] Y. Jang, H. Lee, S. J. Hwang, and J. Shin, "Learning what and where to transfer," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3030–3039. 3, 10

[27] B. Liu, Y. Rao, J. Lu, J. Zhou, and C.-J. Hsieh, "Metadistiller: Network self-boosting via meta-learned top-down distillation," in *European Conference on Computer Vision*. Springer, 2020, pp. 694–709. 3, 11, 12, 13

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016. 3, 11, 12, 13

[29] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349. 3, 11, 12, 13

[30] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1529–1538. 3, 11, 12, 13

[31] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *arXiv preprint arXiv:1808.06866*, 2018. 3, 11, 12, 13

[32] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic soft filter pruning for deep convolutional neural networks," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3594–3604, 2019. 3, 11, 12, 13

[33] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744. 3, 13

[34] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397. 3, 11, 12, 13

[35] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 304–320. 3

[36] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "Thinet: pruning cnn filters for a thinner net," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 10, pp. 2525–2538, 2018. 3

[37] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "Morphnet: Fast & simple resource-constrained structure learning of deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1586–1595. 3

[38] H. Yang, Y. Zhu, and J. Liu, "Ecc: Platform-independent energy-constrained deep neural network compression via a bilinear regression model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 206–11 215. 3

[39] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–800. 3, 11, 12, 13

[40] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4876–4883. 3

[41] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "Metapruning: Meta learning for automatic neural network channel pruning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3296–3305. 3

[42] X. Ning, T. Zhao, W. Li, P. Lei, Y. Wang, and H. Yang, "Dsa: More efficient budgeted pruning via differentiable sparsity allocation," *arXiv preprint arXiv:2004.02164*, 2020. 3

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520. 4, 11, 13

[44] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324. 4

[45] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828. 4

[46] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018. 4

[47] X. Ruan, Y. Liu, B. Li, C. Yuan, and W. Hu, "Dpfps: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2495–2503. 4, 8, 11, 12, 13, 14

[48] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. 9

[49] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of operations research*, vol. 153, no. 1, pp. 235–256, 2007. 10

[50] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135. 10

[51] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018. 11, 12, 13

[52] Z. Chen, T.-B. Xu, C. Du, C.-L. Liu, and H. He, "Dynamical channel pruning by conditional accuracy change for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 799–813, 2020. 11, 12, 13

[53] X. Ding, T. Hao, J. Tan, J. Liu, J. Han, Y. Guo, and G. Ding, "Resrep: Lossless cnn pruning via decoupling remembering and forgetting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4510–4520. 11, 12, 13

[54] B. Li, B. Wu, J. Su, and G. Wang, "Eagleeye: Fast sub-net evaluation for efficient neural network pruning," in *European conference on computer vision*. Springer, 2020, pp. 639–654. 11, 12, 13

[55] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171. 11, 12, 13

[56] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," *arXiv preprint arXiv:1910.10699*, 2019. 11, 12, 13

[57] M. Ji, B. Heo, and S. Park, "Show, attend and distill: Knowledge distillation via attention-based feature matching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7945–7952. 11, 12, 13

[58] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009. 10

[59] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 10

[60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Advances in Neural Information Processing Systems Workshop*, 2017. 11

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 11

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 11

[63] S. Zagoruyko, "92.45% on cifar-10 in torch," *Torch Blog*, 2015. 11

[64] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 264–11 272. 12, 13

[65] S. Guo, Y. Wang, Q. Li, and J. Yan, "Dmcp: Differentiable markov channel pruning for neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1539–1547. 12, 13

[66] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015. 14

[67] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. 14

[68] Y. Zhang, Z. Yin, Y. Li, G. Yin, J. Yan, J. Shao, and Z. Liu, "Celeba-spoof: Large-scale face anti-spoofing dataset with rich annotations," in *European Conference on Computer Vision*. Springer, 2020, pp. 70–85. 14, 15

[69] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755. 15

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 16

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3185317

Yufan Liu received her B.S. degree from Zhejiang University in 2015, and the M.S. degree from Beihang University in 2018. She is currently a research associate with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Her research interests include computer vision, model compression and video understanding.

Jiajiong Cao received BA and Master degree from the college of information science and electronic engineering, Zhejiang University in 2018. Now he is a reseacher in the IoT Business Unit, Ant Group. His research interests are in deep learning, biometrics and face recognition.

Bing Li received the Ph.D. degree from the Department of Computer Science and Engineering, Beijing Jiaotong University, Beijing, China, in 2009. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing. His current research interests include video understanding, color constancy, visual saliency, multi-instance learning, and Web content security.

Weiming Hu received the Ph.D. degree from the Department of Computer Science and Engineering, Zhejiang University, Zhejiang, China, in 1998. From 1998 to 2000, he was a postdoctoral research fellow with the Institute of Computer Science and Technology, Peking University, Beijing. He is currently a professor with the Institute of Automation, Chinese Academy of Sciences(CASIA), Beijing. His research interests are visual motion analysis, recognition of web objectionable information, and network intrusion detection.

Stephen Maybank received a BA in Mathematics from King's College Cambridge in 1976 and a PhD in computer science from Birkbeck college, University of London in 1988. Now he is a professor emeritus in the Department of Computer Science and Information Systems, Birkbeck College. His research interests include the geometry of multiple images, camera calibration, visual surveillance, *etc.*