EPSD: Early Pruning with Self-Distillation for Efficient Model Compression

Dong Chen[1,2*], Ning Liu[2*], Yichen Zhu[2], Zhengping Che[2], Rui Ma[1†], Fachao Zhang[2], Xiaofeng Mou[2], Yi Chang[1], Jian Tang[2†]

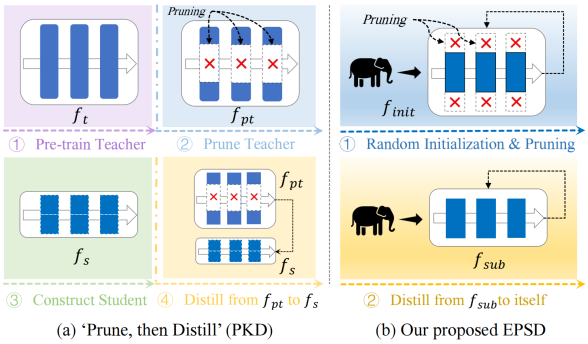
[1]School of Artificial Intelligence, Jilin University
[2]Midea Group
chendong21@mails.jlu.edu.cn, {ruim, yichang}@jlu.edu.cn,
{liuning22, zhuyc25, chezp, zhangfc, mouxf, jiantang22}@midea.com

Abstract

Neural network compression techniques, such as knowledge distillation (KD) and network pruning, have received increasing attention. Recent work 'Prune, then Distill' reveals that a pruned student-friendly teacher network can benefit the performance of KD. However, the conventional teacher-student pipeline, which entails cumbersome pre-training of the teacher and complicated compression steps, makes pruning with KD less efficient. In addition to compressing models, recent compression techniques also emphasize the aspect of efficiency. Early pruning demands significantly less computational cost in comparison to the conventional pruning methods as it does not require a large pre-trained model. Likewise, a special case of KD, known as self-distillation (SD), is more efficient since it requires no pre-training or student-teacher pair selection. This inspires us to collaborate early pruning with SD for efficient model compression. In this work, we propose the framework named Early Pruning with Self-Distillation (EPSD), which identifies and preserves distillable weights in early pruning for a given SD task. EPSD efficiently combines early pruning and self-distillation in a two-step process, maintaining the pruned network's trainability for compression. Instead of a simple combination of pruning and SD, EPSD enables the pruned network to favor SD by keeping more distillable weights before training to ensure better distillation of the pruned network. We demonstrated that EPSD improves the training of pruned networks, supported by visual and quantitative analyses. Our evaluation covered diverse benchmarks (CIFAR-10/100, Tiny-ImageNet, full ImageNet, CUB-200-2011, and Pascal VOC), with EPSD outperforming advanced pruning and SD techniques.

Introduction

Resource-limited edge devices struggle to handle the computational demands of large deep neural networks (DNNs). Therefore, compressing deep models is crucial to eliminate redundancy, facilitating the effective deployment of DNNs on edge devices (Gong et al. 2019; Liu et al. 2020; Guo, Xu, and Ouyang 2023). Various compression methods have been well studied, including network pruning (Han et al. 2015; Guo, Ouyang, and Xu 2020; Huang et al. 2023), knowledge distillation (KD) (Hinton et al. 2014), parameter quantization (Hubara et al. 2017; Wei et al. 2022) and low-rank decomposition (Zhang et al. 2015). Among them, KD and pruning have received increasing attention.



(a) 'Prune, then Distill' (PKD) (b) Our proposed EPSD

Figure 1: Comparison of different model compression schemes. (a) PKD (Park and No 2022) follows four steps to combine pruning and KD. (b) Our Early Pruning with SD (EPSD) needs only two steps for compression.

The concept behind KD is to train a smaller student network to approximate a larger, pre-trained teacher network with higher accuracy (Hinton et al. 2014). The cost of pre-training and the capacity gap issue between teachers and students inevitably limit the usage of KD (Mirzadeh et al. 2020; Xu and Liu 2019). To overcome these limitations, self-distillation (SD) is proposed to enable students to distill knowledge from themselves (Shen et al. 2022; Yang et al. 2019; Zhang, Bao, and Ma 2021; Zhang et al. 2019). Namely, SD allows the student network to learn from its predictions (Mobahi, Farajtabar, and Bartlett 2020), enabling a more streamlined training procedure that requires much fewer computational resources. However, the potential risk in SD is that the student could result in overfitting if the training process is not properly regularized (Kim et al. 2021). Therefore, regularizing the student model becomes crucial to ensure effective knowledge transfer.

Network pruning removes the redundancy inside the original network and generates a sub-network with comparable accuracy performance (LeCun, Denker, and Solla 1989;

*These authors contributed equally. This work was done during Dong Chen's internship at Midea Group.
†Corresponding authors.

Liu et al. 2021a). In addition to reducing the computational requirements, pruning also helps prevent overfitting of DNNs (Han, Mao, and Dally 2016). Current pruning works focus on pruning in the early stage (Frankle and Carbin 2019; de Jorge et al. 2021; Alizadeh et al. 2022). Pruning takes place either during initialization or shortly after a few training steps. Early pruning methods efficiently operate without the need for a pre-trained model. Recent work 'Prune, then Distill' (Park and No 2022) (we refer to it as PKD) explores the regularization effect of pruning on KD. They analyze the distillation process regularized by the pruned teacher and combine pruning and KD in four steps as shown in Fig. 1(a): 1) Pre-train a teacher network f_t, 2) prune f_t and obtain a pruned teacher f_{pt}, 3) construct a student network f_s according to f_{pt}, and 4) distill knowledge from f_{pt} to f_s. Though PKD reveals that *pruning student-friendly teacher can boost the performance of KD*, the required cumbersome pre-training of the teacher and the complicated steps make it suffer from heavy training efforts.

To mitigate the complicated compression process, we attempt to collaborate early pruning with SD for efficient model compression. An intuitive approach is to prune the network and then finetune the pruned network with SD. However, different from PKD, in the context of the SD, pruning the teacher network also affects the student and leads to inadequate regularization if the pruned student presents weak trainability. Empirically, applying a simple combination of pruning and SD results in severe performance degradation especially under the large sparsity ratios (as shown in Fig. 2). Therefore, the key question is: *How to effectively prune DNNs with SD to yield performance gains?*

A promising solution is to make the pruned network favorable to SD, $i.e.$, to preserve more *distillable* weights to ensure the efficacy of the SD process. To this end, we propose a novel framework named EPSD that collaborates **E**arly **P**runing and **S**elf-**D**istillation for efficient model compression. Specifically, EPSD has two main steps as shown in Fig. 1 (b): 1) **Early Pruning**: Prune an initialized network f_{init} to obtain a pruned sub-network f_{sub} with distillable weights. 2) **Self-Distillation**: Train the pruned network f_{sub} by SD. Namely, given a desired sparsity level, EPSD globally ranks the weights in f_{init} according to their influence (quantified by the absolute gradients) on the SD loss and removes weights with less influence. By doing so, the trainability of the student network can be enhanced since the subnetwork maintains objective consistency with SD loss and preserves more distillable weights. Next, EPSD applies SD to recover the accuracy of the student network with distillable weights. Our contributions are summarized as follows:

- We present EPSD, which collaborates early pruning with SD, to compress models efficiently in only two steps. Meanwhile, EPSD preserves the trainability of the pruned network to improve performance.
- EPSD identifies distillable weights that ensure objective consistency between early pruning and SD, and we present quantitative and visualized analysis to demonstrate the efficacy of EPSD.
- Extensive results with three advanced SD methods on multiple benchmarks show that EPSD outperforms advanced pruning and SD methods while showcasing its scalability on two downstream tasks.

Related Works

Knowledge Distillation. Knowledge Distillation (KD) transfers various 'knowledge' in networks (Romero et al. 2015; Hinton et al. 2014), acting as a potent regularization method to enhance generalization by utilizing learned softened targets (Shen et al. 2022). However, the capacity gap prevents well-performing teachers from making students better (Mirzadeh et al. 2020).

Self-Distillation. To improve the efficiency of knowledge transfer, Self-Distillation (SD) leverages knowledge from the student network without involving additional teachers (Wang and Yoon 2021; Yun et al. 2020). The key to SD is creating soft targets, where the student network generates its valuable knowledge to guide its training (Lee, Hwang, and Shin 2020; Zhang, Bao, and Ma 2021; Yang et al. 2019; Shen et al. 2022). SD's efficiency arises from avoiding teacher network pre-training and addressing teacher-student capacity gaps. Yet, the student network might be over-fitting due to insufficient training regularization (Kim et al. 2021). Recently, PKD (Park and No 2022) revealed the positive regularizing impact of pruning teacher networks on KD, which inspires us to regularize the SD process by pruning.

Network Pruning. Network pruning aims to identify and remove unnecessary weights, reducing complexity while preserving training performance (Reed 1993; Lee et al. 2020). Traditional approaches (Han et al. 2015; Molchanov et al. 2017) typically follow pre-training, pruning, and re-training to prune, which requires much training effort. Another paradigm named Dynamic Sparse Training (DST) (Mocanu et al. 2018; Bellec et al. 2018; Evci et al. 2020; Liu et al. 2021b) starts from a (random) sparse neural network and allows the sparse connectivity to evolve dynamically during training. DST can significantly improve the trainability of sparse DNNs without increasing the training FLOPs. Recently, early pruning (Lee, Ajanthan, and Torr 2019; Wang, Zhang, and Grosse 2020; de Jorge et al. 2021; Alizadeh et al. 2022) has been widely studied as it identifies sparse subnetworks before training without cumbersome pre-training. Many early pruning works evaluate the importance of individual weights regarding the impact on loss, i.e., the gradients of a network. Though early pruning is efficient, it is considered under-performance (Wang et al. 2022): pruning neural networks breaks the dynamical isometry (Saxe, McClelland, and Ganguli 2014) and results in the trainability degradation (Lee et al. 2020). In this work, we empirically show that SD greatly enhances the performance of early pruned networks and improves their trainability by ensuring alignment between pruning and SD objectives.

Early Pruning with Self-Distillation

We first introduce a simple combination of early pruning and SD and show that it suffers performance degradation. To address this issue, we introduce the concept of distillable weights, along with quantitative and visualized analysis. Fi-

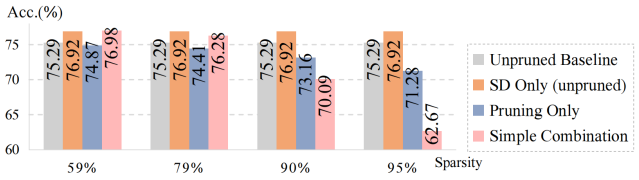


Figure 2: Performance comparison among the 'Simple Combination', pre-trained network ('Unpruned Baseline'), the network only performs pruning without fine-tuning with SD ('Pruning Only'), and the network only performs SD without any sparsity ('SD Only') on CIFAR-100 of ResNet-18. The 'Simple Combination' suffered severe performance degradation, especially under the high sparsity ratio 95%.

nally, we present the overall framework of EPSD, demonstrating the efficiency by comparing the required training efforts with other compression techniques.

The 'Simple Combination'

A straightforward way to combine pruning and SD requires two steps. Step-1: Network pruning without pre-training and step-2: Distill knowledge to itself. Specifically, step-1 is to identify a sub-network from the randomly initialized network by pruning. Step-2 is to fine-tune the sub-network via SD. Since our goal is to efficiently compress the model, the early pruning method ProsPr (Alizadeh et al. 2022) is utilized as the representative method in step-1.

Step-1: Identify Redundancy Before Training. Lee *et al.* first proposed SNIP (Lee, Ajanthan, and Torr 2019) to prune unnecessary weights in random initialized networks that are least salient for the loss. They compute the gradients Δ to generate saliency scores for initial weight θ_{init} with random samples x_{rand} and remove the weights with the lowest scores. Specifically, an all-one mask m is attached to initial weights to get $\theta_0 \leftarrow m \odot \theta_{init}$, and the saliency scores can be computed as:

$$\Delta(w_p, x_{rand}) = \frac{\partial \mathcal{L}(\theta_0, x_{rand})}{\partial m_p}, \tag{1}$$

$$s_{w_p} = \frac{|\Delta(w_p, x_{rand})|}{\sum_q |\Delta(w_q, x_{rand})|}, \tag{2}$$

where m is the pruning mask with values 0s or 1s (initial value is 1s), Δ denotes gradients derived from labels, w_p is p-th weight in θ_{init}, s_{w_p} is the saliency score for measuring the importance of w_p. Recently, Milad et al. pointed out that pruning should consider the trainability of a certain weight, instead of only its immediate impact on the loss before training (Alizadeh et al. 2022), they measured the impact of pruning on loss across i gradient descent steps during initial training, rather than assessing alterations in loss at initialization. The saliency scores are calculated based on the updated weights θ_i as in Eq. (3):

$$\Delta(w_p, x_i) = \frac{\partial \mathcal{L}(\theta_i, x_i)}{\partial m_p}, \tag{3}$$

where x_i denotes *i-th* random sampled batch of data for computing the gradients. In the classification tasks, the cross-entropy (CE) loss runs through the entire process, from pruning to training. The difference between predictions and labels is used to evaluate the importance of weights and optimize the pruned network.

Step-2: Distilling Knowledge from Soften Targets. In classification task, we denote $\mathbf{x} \in \mathcal{X}$ as input and $y \in \mathcal{Y} = \{1, \ldots, C\}$ as its ground-truth label. Given the input $\mathbf{x}$, the predictive distribution of a softmax classifier is:

$$P(y \mid \mathbf{x}; \theta, \tau) = \frac{\exp\left(l_y(\mathbf{x}; \theta)/\tau\right)}{\sum_{i=1}^{C} \exp\left(l_i(\mathbf{x}; \theta)/\tau\right)}, \tag{4}$$

where l_i denotes the logit of DNNs for class i which are parameterized by θ, and $\tau > 0$ is the temperature scaling factor. To improve the generalization ability, traditional KD (Hinton et al. 2014) transfers pre-trained teacher's knowledge by optimizing an additional Kullback-Leibler (KL) divergence loss between the softened outputs $\widetilde{P}$ from teacher and student in every mini-batch x_i:

$$\mathcal{L}_{KD} = \frac{1}{n} \sum_{i=1}^{n} \tau^2 \cdot D_{KL}\left(\widetilde{P}(x_i; \theta_t) \| \widetilde{P}(x_i; \theta_s)\right). \tag{5}$$

The original KD matches the predictions of the same inputs from two different networks, while the SD replaces the teacher's prediction with that of the student network itself. Various works (Lee, Hwang, and Shin 2020; Xu and Liu 2019; Zhang et al. 2019; Zhang, Bao, and Ma 2021; Yang et al. 2019; Shen et al. 2022) have explored enhancing the SD method in different ways. Our work focuses on the gradients of SD loss rather than specific improvements. We further discuss the gradients of SD loss in Sec. . Without loss of generality, we formulate SD loss as follows:

$$\mathcal{L}_{SD} = \frac{1}{n} \sum_{i=1}^{n} \tau^2 \cdot D_{KL}\left(\widetilde{P}(\overline{x}_i; \overline{\theta}_s) \| \widetilde{P}(x_i; \theta_s)\right), \tag{6}$$

where $\widetilde{P}(\overline{x}_i; \overline{\theta}_s)$ represents the soft targets produced by the student networks in SD, and different SD methods have different definitions of $\overline{x}_i$ and $\overline{\theta}_s$. We refer the readers to the Appendix for a more detailed explanation of these symbols.

Remarks. A baseline method for combing early pruning and SD involves applying the two techniques sequentially (we name it 'Simple Combination.'). This straightforward approach produces a distilled, sparse network. The preliminary study shown in Fig. 2 demonstrates that under the sparsity ratio of 95%, the accuracy of the 'Simple Combination' is only 62.67%, lower nearly 13% than the 'Unpruned Baseline' and 8% than the 'Pruning Only'. These anomalous results indicate that the pruned network can not effectively learn via SD when directly applying SD to the early-pruned network. To this end, we raised one question: *"Why does the early-pruned network degrade accuracy when training with SD?"* In the pruning step of the 'Simple Combination', the gradient only reflects the difference between the network output and the hard labels, without considering the soft targets generated in SD. We argue that it is difficult for the early-pruned network to learn knowledge from itself during SD when directly combining the early pruning and SD.

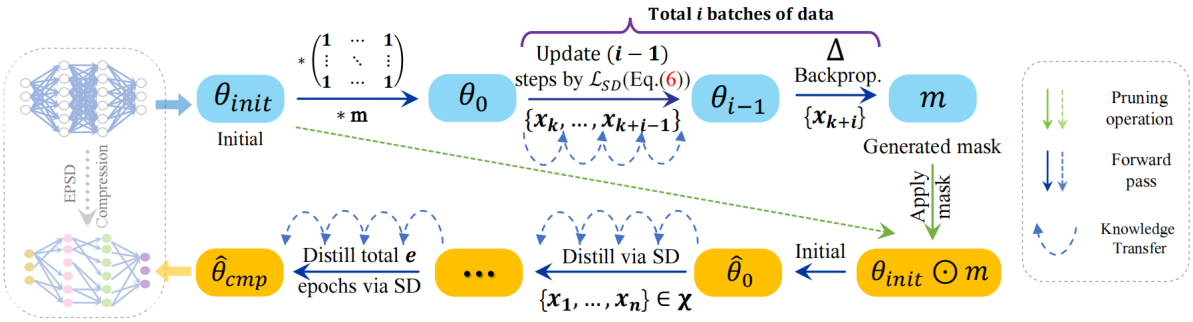


Figure 3: EPSD prunes a random initialized network with weights θ_{init} in step-1 (blue block) and then employs the SD algorithm to train the pruned network in step-2 (orange block). In Step 1, EPSD identifies and retains distillable weights by measuring the impact of SD loss on individual weights after i steps of training.

Identify Distillable Weights via SD

As introduced in the previous section, a simple combination of early pruning and SD does not lead to performance gains and even results in severe degradation at large sparsity. In the SD scenario, when the teacher network is pruned, the students are also affected, potentially leading to inadequate distillation if a weak student is involved. A desirable mitigation solution is to make pruning results favorable to SD, *i.e.*, to *preserve more distillable weights* to ensure that the pruned model can be better distilled. Intuitively, obtaining distillable weights implies that the pruned network should be consistent with the optimized objective of SD.

As a result, we propose to identify distillable weights with SD loss before training. More specifically, during pruning, we establish a knowledge transmission path to facilitate the model to learn from its own outputs. We evaluate the importance of the weights by conducting a few SD iterations to derive the necessary gradients. Formally, the salience score for an individual weight can be derived from Eq. (3) and (6):

$$\widetilde{\Delta}(w_p, x_i) = \frac{\partial \mathcal{L}_{SD}(\theta_i, x_i)}{\partial m_p}, \tag{7}$$

$$\widetilde{s}_{w_p} = \frac{\left|\widetilde{\Delta}(w_p, x_i)\right|}{\sum_q \left|\widetilde{\Delta}(w_q, x_i)\right|}. \tag{8}$$

We remove weights that have the least impact on SD loss according to the desired sparsity ratio, and the *weights with higher salience scores $\widetilde{s}$ are regarded as distillable to be preserved*. We thoroughly assess weight importance by considering both hard label influences and network-generated knowledge during pruning, resulting in more reliable saliency criteria driven mainly by the SD loss.

To analyze the trainability of the pruned model, we leverage loss surface (Li et al. 2018) to visualize the loss landscape and assess the ease of optimization. Additionally, we utilize the mean Jacobian singular values (Mean-JSV) as a quantitative metric to gauge compliance with the dynamic isometry conditions (Wang et al. 2021; Wang and Fu 2023). The top of Fig. 4 shows the contour plots of loss. We observed that the loss surface of EPSD is flatter than the 'Simple Combination', and reaches local minima

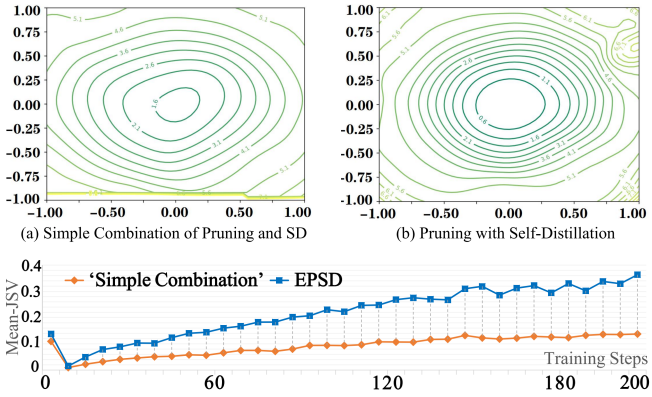


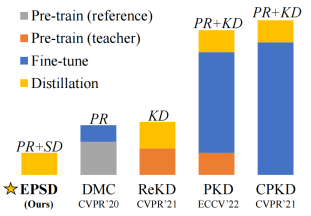
Figure 4: Trainability analysis. Top: Loss contour plots of early-pruned networks using (a) 'Simple Combination' and (b) EPSD. Bottom: Comparison of Mean-JSV curves of EPSD and the 'Simple Combination' approach.

faster (minimum loss value 0.6 v.s 1.6 within equal training steps), implying the pruned model by EPSD is easier to optimize (Arora et al. 2018; Dinh et al. 2017). The bottom of Fig. 4 shows Mean-JSV curves over the first 200 training steps for pruned model obtained by EPSD and the 'Simple Combination', respectively. In theory, a larger Mean-JSV (closer to 1) indicates better trainability of the model. The Mean-JSV of EPSD better meets dynamic isometry requirements than the 'Simple Combination', revealing the potential of keeping objective consistency between pruning and SD in preserving trainable weights.

Remarks. To tackle the degradation issue raised by the 'Simple Combination', we aim to pinpoint distillable weights preferred by SD for improved accuracy. Our visual and quantitative analysis reveals that sub-networks identified by maintaining objective consistency exhibit superior trainability compared to those identified solely through pruning.

Towards Efficient Model Compression

Fig. 3 shows the overall compression procedure of EPSD. There are mainly two steps as mentioned in Sec. . In step-1, given randomly initialized weights θ_{init}, EPSD estimates



Method	Model	#Ep.	Time
PKD	VGG19-VGG19$_{79\%}$	1300	11.3h
EPSD	VGG19$_{79\%}$	200	1.4h
ReKD	ResNet56-ResNet20	480	4.2h
EPSD	ResNet56	200	1.3h
	ResNet20	200	1.2h

Figure 5: Training efforts comparisons among various representative compression approaches. Left: Total training epochs of CPKD (Aghli and Ribeiro 2021), PKD (Park and No 2022), ReKD (Chen et al. 2021), DMC (Gao et al. 2020). 'PR' and 'KD (SD)' denote pruning and knowledge distillation (self-distillation), respectively. Right: Comparison of total training wall time under identical conditions.

the effect of pruning on the SD loss (Eq. 6) over i steps of gradient descent. By doing so, EPSD preserves more distillable weights, which become crucial since they offer superior trainability and are more easily optimized by the SD loss as discussed in Sec. . Once the pruning mask m is generated by the gradients $\widetilde{\Delta}$, we apply it to the initial weights θ_{init} to get a pruned network. In Step 2, we train the pruned network by SD until it reaches convergence.

We emphasize that EPSD is efficient, which is attributed to: 1) the absence of pre-training for pruning, 2) the elimination of teacher training, and 3) the pruned network's distillable weights, which contribute to improved trainability and faster convergence during SD. In Fig. 5, we demonstrate the training efforts of EPSD and compare them against other representative compression methods. Among them, EPSD combines early pruning and SD (*PR+SD*), DMC uses advanced pruning (*PR*), ReKD is a KD method (*KD*), and the other two are combinations of pruning and KD (*PR+KD*). EPSD achieves efficient training with fewer epochs than other methods. For instance, the training time of PKD is about eight times that of EPSD (11.3 vs. 1.4 hours).

Experiments

We evaluate EPSD on various benchmarks, including CIFAR-10/CIFAR-100 (Krizhevsky, Hinton et al. 2009), Tiny-ImageNet, and full ImageNet (Deng et al. 2009) using diverse networks and comparing with the 'Simple Combination' approach, advanced pruning and SD methods. We also assess EPSD's adaptability and scalability in two downstream tasks. More details can be found in the Appendix.

EPSD equipped with Various SD Methods

We incorporate three distinct SD algorithms (CS-KD (Yun et al. 2020), PS-KD (Kim et al. 2021), and DLB (Shen et al. 2022)) into EPSD to ensure a comprehensive evaluation. Our experiments are conducted on CIFAR-10/100 and Tiny-ImageNet datasets across five sparsity ratios (36%, 59%, 79%, 90%, 95%). To ensure fairness in comparison, we employ identical hyper-parameters for training each dataset. For each variant of EPSD utilizing a specific SD method, we conduct a comprehensive comparison with 1) the unpruned network without any pruning or SD (Unpruned Baselines'), 2) network training using the respective SD method ('SD Only'), and 3) the simple combination of pruning and the specific SD method ('Simple Combination'). Figure 6 illustrates the specific comparison results.

Backbone	VGG-19				ResNet-50			
Sparsity	90%		95%		90%		95%	
Accuracy	top1	top5	top1	top5	top1	top5	top1	top5
Unpruned	73.1	91.3	73.1	91.3	75.6	92.8	75.6	92.8
SNIP$_{19'}$	68.5	88.8	63.8	86.0	61.5	83.9	44.3	69.6
GraSP$_{20'}$	69.5	89.2	67.0	87.4	65.4	86.7	46.2	66.0
FORCE$_{21'}$	70.2	89.5	65.8	86.8	64.9	86.5	59.0	82.3
DOP$_{22'}$	-	-	-	-	64.1	-	48.1	-
ProsPr$_{22'}$	70.7	89.9	66.1	87.2	65.9	86.9	59.6	82.8
Sim.Cmb.	17.3	25.8	15.4	23.0	9.9	16.4	8.3	15.3
EPSD	**71.2**	**90.1**	**67.1**	**87.6**	**66.3**	**87.3**	**60.1**	**83.0**

Table 1: Comparing test accuracy of various advanced early pruning methods at 90% and 95% sparsity on full ImageNet. 'Sim.Cmb.' refers to the 'Simple Combination'.

Based on the results, we have the following observations:

- EPSD consistently outperformed the 'Simple Combination' overall settings. Moreover, under high sparsity conditions ($e.g.$, 95%), EPSD remained competitive while the 'Simple Combination' heavily declined.
- On the more challenging Tiny-ImageNet, the 'Simple Combination' degraded more severely than EPSD for all three SD methods. For instance, with DLB and VGG-19 on Tiny-ImageNet at sparsity 90%, the accuracy of the 'Simple Combination' is 20.80% lower than 'Unpruned Baseline' (29.08% vs. 49.88%), while EPSD achieved 53.91% accuracy, increasing 1.80% and 3.93% compared to 'SD Only' and 'Unpruned Baseline', respectively.
- EPSD outperformed 'Unpruned Baseline' and 'SD Only' over all three SD methods in most settings, indicating that early pruning with SD can boost the performance of SD. EPSD maintains an advantage over the 'Simple Combination', affirming its efficacy in preserving more distillable weights and achieving promising performance.

Comparison of Pruning Methods

To illustrate the effectiveness of EPSD, we compared EPSD with advanced pruning methods on CIFAR-10/100 (See Appendix) and ImageNet. Further, we extended EPSD with structured pruning to show the extensibility of our method.

CIFAR-10/100. We perform extensive comparisons with recent early pruning methods on CIFAR-10 and CIFAR-100, and we applied EPSD to two popular lightweight networks (MobileNet-v2 (Sandler et al. 2018) and MobileViT (Mehta and Rastegari 2022)), which is not a common practice in previous early pruning works. We also investigated the iterative version of EPSD. Please refer to the Appendix.

ImageNet. We evaluated EPSD on the challenging full ImageNet dataset. Table 1 compared EPSD with advanced pruning methods in terms of top-1 and top-5 accuracy under

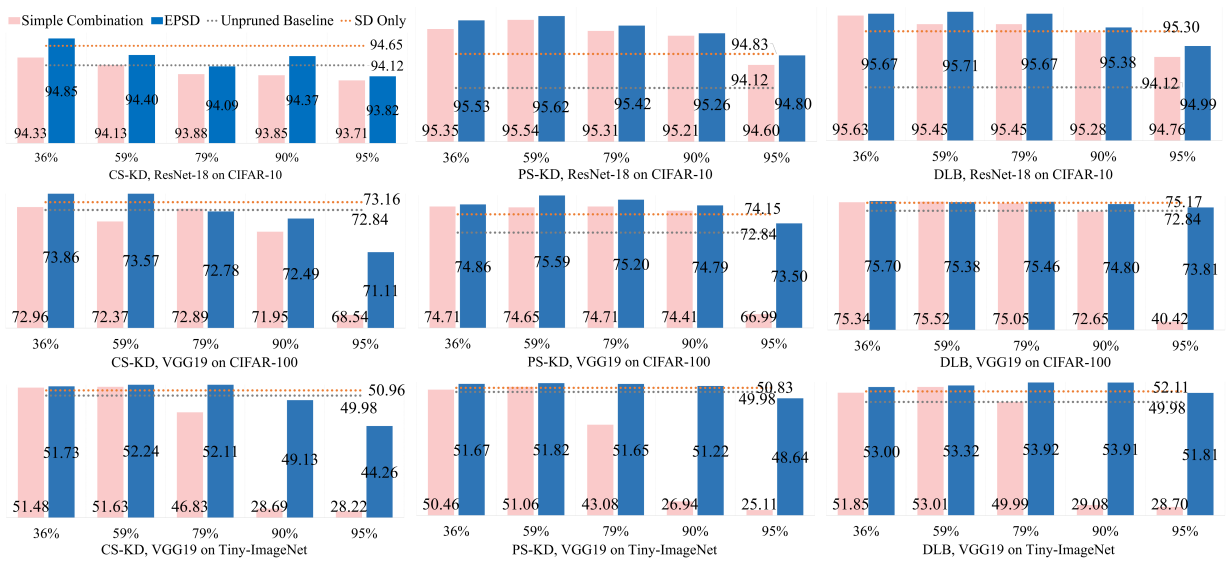


Figure 6: Test accuracy among 'Unpruned Baseline', 'SD Only', 'Simple Combination' and EPSD with three equipped SD methods (CS-KD, PS-KD, DLB) on CIFAR-10/100 and Tiny-ImageNet under various sparsity. The three rows illustrate the three datasets and the three columns display the three equipped SD methods.

90% and 95% sparsity ratio with VGG-19 and ResNet-50. EPSD surpasses other early pruning methods and notably addresses the degradation problem of 'Simple Combination' on challenging datasets. For instance, EPSD leads GraSP by 0.9% and improves by 0.4% over ProsPr at sparsity 90% with ResNet-50. This highlights EPSD's effective synergy of early pruning and SD, leading to enhanced performance.

Structured Pruning. To illustrate the extensibility of EPSD, We evaluate structured pruning, where entire channels are eliminated rather than individual weights. We compare EPSD against 3SP (van Amersfoort et al. 2020), ProsPr (Alizadeh et al. 2022), and random structure pruning reported in ProsPr. The results are summarized in Table 2, and our EPSD achieves the best accuracy performance compared with other structured pruning methods.

Comparison of SD Methods

Since EPSD is to explore the *effective combination* of early pruning and SD, we compare EPSD with SD methods to show the effectiveness. Specifically, we compare EPSD with LSR (Szegedy et al. 2016), TFKD (Yuan et al. 2020), CSKD (Yun et al. 2020), PSKD (Kim et al. 2021), and DLB (Shen et al. 2022) using various models (ResNet-32/110 and VGG-16/19) on CIFAR-10/100. When compared to SD methods, EPSD prunes networks at 80% sparsity. Table 3 shows the comparison results. Surprisingly, though EPSD removes most of the weights, it still achieved comparable or better performance than other advanced SD methods. Please be aware that directly comparing early-pruned models with unpruned self-distilled models is uncommon in prior research. This is because models obtained through early pruning are often considered less trainable (Lee et al. 2020; Frankle et al. 2021; Wang et al. 2022). However, we demonstrate that combining early pruning with self-distillation is a viable and competitive approach.

Sparsity	Method	CIFAR-10	CIFAR-100
-	Unpruned	93.88(%)	72.84 (%)
80%	Random	92.00	67.50
	3SP	93.40	69.90
	ProsPr	93.61	72.29
	EPSD	**93.82**	**73.16**
90%	Random	90.40	63.80
	3SP	93.10	68.30
	ProsPr	93.64	71.12
	EPSD	**93.72**	**71.80**

Table 2: Test accuracy among various structured pruning methods using VGG-19 on CIFAR-10 and CIFAR-100 under sparsity ratios 80% and 90%.

Impact of SD-based Pre-training

In previous sections, we showed that a simple combination of early pruning and SD can lead to performance degradation. To verify the key idea of EPSD that identifying more distillable weights enhances the accuracy performance, we design another way for combination: 1) start by training the network from scratch with SD, then 2) prune it, and 3) fine-tune the pruned model with SD to regain performance. We name this method 'Simple Combination-2' (SC-2). Compared to 'Simple Combination' (SC-1), SC-2 requires more pre-training effort. To explore the potential impact of SD-based pre-training on the pruned model, we tested SC-2's effect on ImageNet using ResNet-50. Experiments shown in Table 4 indicated that SC-2 achieved *comparable* accuracy to EPSD (66.4% vs. 66.2%). We argue this happened

Net.	U.P.	LSR	TFKD	CSKD	PSKD	DLB	**EPSD**
R32	93.46	93.27	93.68	93.12	94.04	94.15	**94.68**
R110	94.79	94.40	95.08	93.88	94.91	95.15	**95.32**
V16	93.97	94.09	94.08	93.78	94.10	**94.62**	94.51
V19	93.88	93.95	94.09	93.62	93.93	94.42	**94.45**
R32	71.74	71.79	73.91	70.79	72.51	74.00	**74.30**
R110	76.36	76.68	72.98	76.59	77.15	78.18	**78.45**
V16	73.63	74.19	74.06	74.19	74.05	76.12	**76.31**
V19	74.61	73.25	72.54	73.35	73.64	75.47	**76.11**

Table 3: Comparing test accuracy against advanced SD methods and the unpruned baseline (U.P.). The top section shows CIFAR-10 and the lower section displays CIFAR-100. We use 'R' for ResNet and 'V' for VGG. EPSD is 80% sparsity, while the other approaches remain unpruned.

Method	P.T. w/ SD #Epochs	PR. w/ CE	PR. w/ SD	(Re-)Train w/ SD #Epochs	(Re-)Train w/ SD Top1 Acc.(%)
SC-1	0	✓		100	9.9
EPSD	0		✓	100	66.2
SC-2	100	✓		100	66.4
P.T.+EPSD	100		✓	100	66.6

Table 4: Investigation of the impact of SD-based Pre-training. 'P.T.' means pre-training and 'PR.' is the pruning process with a 90% sparsity ratio.

because *SD-based pre-training in SC-2 produced distillable weights*. After pruning with a standard cross-entropy (CE) loss, the remaining weights still kept their distillable nature, allowing the pruned model to regain from fine-tuning with SD. In addition, building upon SC-2, we used EPSD to compress the SD-pre-trained model (instead of starting from random initialization), resulting in further accuracy improvement (66.6%), which is attributed to retaining more distillable weights through pruning with the SD loss.

Downstream Tasks

We further verify the robustness of EPSD on two downstream tasks presented below. See the Appendix for details.

Weakly Supervised Object Localization. As shown in Table 5, we reported the error rates with a pruning ratio of 50%. Compared to ProsPr, EPSD achieved lower errors (Cls. Err of 24.40% vs. 25.39%, Top-1 Loc. Err. as low as 41.23%). Compared to the unpruned baseline, EPSD only saw a slight 0.27% drop in localization accuracy, showing improved generalization in weakly supervised scenarios.

Semantic Segmentation. As shown in Table 6, across two different metrics, EPSD outperforms ProsPr and the 'Simple Combination'. Specifically, EPSD achieves a 1.63% higher than ProsPr in mean IoU and 2.63% higher in pixel accuracy. Compared to the 'Simple Combination', the improvements are even more significant, with increases of 5.26% and 5.76% in two metrics, respectively.

Method	*s.p.*	Cls.Err. (↓)	Loc.Err. (↓) Top-1	Loc.Err. (↓) Gt-Known
Unpruned	-	23.90%	40.96%	23.97%
ProsPr	50%	25.39%	48.65%	32.69%
Sim.Cmb.	50%	27.53%	50.57%	33.33%
EPSD	50%	24.40%	41.23%	25.08%

Table 5: Results of weakly supervised object localization task on CUB-200-2011. The top-1 classification error (Cls.Err.) and localization error rates (Loc.Err.) are reported.

Method	*s.p.*	Mean IoU (↑)	pixAcc (↑)
Unpruned	-	46.46%	85.70%
ProsPr	40%	42.87%	80.34%
Sim.Cmb.	40%	39.24%	77.21%
EPSD	40%	44.50%	82.97%

Table 6: Results of semantic segmentation task on Pascal VOC 2012. The mean intersection-over-union (Mean IOU) and pixel accuracy (pixAcc) are reported.

Discussion and Limitation

This paper explores an efficient model compression framework. By effectively combining early pruning with SD, EPSD improved performance for pruned models without the burden of extensive training. Importantly, we address the degradation issue arising in a simple combination of early pruning and SD, shedding light on a promising research direction for combining these two techniques, which might offer enlightening insights to the community. However, this paper mainly addresses fundamental vision models in computer vision. Our focus has yet to encompass the presently prevalent large-scale language or multi-model networks. It remains a potential direction for our future research.

Conclusion

In this study, we introduce the Early Pruning with Self-Distillation (EPSD) framework, which identifies and retains distillable weights during pruning for a specific SD task. EPSD seamlessly integrates early pruning and SD in just two steps, ensuring the trainability of pruned networks for effective model compression. We unveil that a straightforward combination of pruning and SD can result in performance decline, particularly at high sparsity ratios. Extensive visual and quantitative analysis show that EPSD enhances the trainability of pruned networks, and outperforms advanced pruning and SD methods. We believe EPSD will inspire more follow-ups for efficient compression of other multi-modal networks, which will accelerate the deployment of the latest deep models to edge devices.

Acknowledgments

This work is supported in part by the National Natural Science Foundation (NSF) of China (No. 62202199, No. 61976102, and No.U19A2065).

References

Aghli, N.; and Ribeiro, E. 2021. Combining weight pruning and knowledge distillation for cnn compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3191–3198.

Alizadeh, M.; Tailor, S. A.; Zintgraf, L. M.; van Amersfoort, J.; Farquhar, S.; Lane, N. D.; and Gal, Y. 2022. Prospect Pruning: Finding Trainable Weights at Initialization using Meta-Gradients. *International Conference on Learning Representations*.

Arora, S.; Ge, R.; Neyshabur, B.; and Zhang, Y. 2018. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, 254–263. PMLR.

Bellec, G.; Kappel, D.; Maass, W.; and Legenstein, R. 2018. Deep rewiring: Training very sparse deep networks. *International Conference on Learning Representations*.

Chen, P.; Liu, S.; Zhao, H.; and Jia, J. 2021. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5008–5017.

de Jorge, P.; Sanyal, A.; Behl, H. S.; Torr, P. H.; Rogez, G.; and Dokania, P. K. 2021. Progressive skeletonization: Trimming more fat from a network at initialization. *International Conference on Learning Representations*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 248–255.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 1019–1028. PMLR.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2943–2952. PMLR.

Frankle, J.; and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations*.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2021. Pruning neural networks at initialization: Why are we missing the mark? *International Conference on Learning Representations*.

Gao, S.; Huang, F.; Pei, J.; and Huang, H. 2020. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1899–1908.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4852–4861.

Guo, J.; Ouyang, W.; and Xu, D. 2020. Channel pruning guided by classification loss and feature importance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 10885–10892.

Guo, J.; Xu, D.; and Ouyang, W. 2023. Multidimensional Pruning and Its Extension: A Unified Framework for Model Compression. *IEEE Transactions on Neural Networks and Learning Systems*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28.

Hinton, G.; Vinyals, O.; Dean, J.; et al. 2014. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems Workshop*.

Huang, Y.; Liu, N.; Che, Z.; Xu, Z.; Shen, C.; Peng, Y.; Zhang, G.; Liu, X.; Feng, F.; and Tang, J. 2023. CP3: Channel Pruning Plug-In for Point-Based Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5302–5312.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1): 6869–6898.

Kim, K.; Ji, B.; Yoon, D.; and Hwang, S. 2021. Self-knowledge distillation with progressive refinement of targets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6567–6576.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *(Technical Report)*.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2.

Lee, H.; Hwang, S. J.; and Shin, J. 2020. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, 5714–5724. PMLR.

Lee, N.; Ajanthan, T.; Gould, S.; and Torr, P. H. 2020. A signal propagation perspective for pruning neural networks at initialization. *International Conference on Learning Representations*.

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

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31.

Liu, N.; Ma, X.; Xu, Z.; Wang, Y.; Tang, J.; and Ye, J. 2020. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4876–4883.

Liu, N.; Yuan, G.; Che, Z.; Shen, X.; Ma, X.; Jin, Q.; Ren, J.; Tang, J.; Liu, S.; and Wang, Y. 2021a. Lottery Ticket Preserves Weight Correlation: Is It Desirable or Not? In *International Conference on Machine Learning*, 7011–7020. PMLR.

Liu, S.; Yin, L.; Mocanu, D. C.; and Pechenizkiy, M. 2021b. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, 6989–7000. PMLR.

Mehta, S.; and Rastegari, M. 2022. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *International Conference on Learning Representations*.

Mirzadeh, S. I.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; and Ghasemzadeh, H. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5191–5198.

Mobahi, H.; Farajtabar, M.; and Bartlett, P. 2020. Self-distillation amplifies regularization in hilbert space. *Advances in Neural Information Processing Systems*, 33: 3351–3361.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1): 1–12.

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

Park, J.; and No, A. 2022. Prune your model before distill it. In *European Conference on Computer Vision*, 120–136. Springer.

Reed, R. 1993. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5): 740–747.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. *International Conference on Learning Representations*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

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

Shen, Y.; Xu, L.; Yang, Y.; Li, Y.; and Guo, Y. 2022. Self-Distillation from the Last Mini-Batch for Consistency Regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11943–11952.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

van Amersfoort, J.; Alizadeh, M.; Farquhar, S.; Lane, N.; and Gal, Y. 2020. Single shot structured pruning before training. *arXiv preprint arXiv:2007.00389*.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking winning tickets before training by preserving gradient flow. *International Conference on Learning Representations*.

Wang, H.; and Fu, Y. 2023. Trainability Preserving Neural Pruning. *International Conference on Learning Representations*.

Wang, H.; Qin, C.; Bai, Y.; and Fu, Y. 2021. Dynamical isometry: The missing ingredient for neural network pruning. *arXiv preprint arXiv:2105.05916*.

Wang, H.; Qin, C.; Bai, Y.; Zhang, Y.; and Fu, Y. 2022. Recent advances on neural network pruning at initialization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 23–29.

Wang, L.; and Yoon, K.-J. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6): 3048–3068.

Wei, X.; Zhang, Y.; Zhang, X.; Gong, R.; Zhang, S.; Zhang, Q.; Yu, F.; and Liu, X. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35: 17402–17414.

Xu, T.-B.; and Liu, C.-L. 2019. Data-distortion guided self-distillation for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5565–5572.

Yang, C.; Xie, L.; Su, C.; and Yuille, A. L. 2019. Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2859–2868.

Yuan, L.; Tay, F. E.; Li, G.; Wang, T.; and Feng, J. 2020. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3903–3911.

Yun, S.; Park, J.; Lee, K.; and Shin, J. 2020. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13876–13885.

Zhang, L.; Bao, C.; and Ma, K. 2021. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8): 4388–4403.

Zhang, L.; Song, J.; Gao, A.; Chen, J.; Bao, C.; and Ma, K. 2019. 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*, 3713–3722.

Zhang, X.; Zou, J.; He, K.; and Sun, J. 2015. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10): 1943–1955.