PRELIMINARY PREPRINT VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference.

Prior Gradient Mask Guided Pruning-aware Fine-tuning

Linhang Cai[1,2], Zhulin An[1*], Chuanguang Yang[1,2], Yangchun Yan[3], Yongjun Xu[1]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Horizon Robotics, Beijing, China
Email: {cailinhang19g, anzhulin, yangchuanguang, xyj}@ict.ac.cn, yangchun.yan@horizon.ai

Abstract

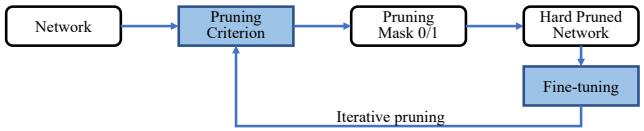
We proposed a Prior Gradient Mask Guided Pruning-aware Fine-tuning (PGMPF) framework to accelerate deep Convolutional Neural Networks (CNNs). In detail, the proposed PGMPF selectively suppresses the gradient of those "unimportant" parameters via a prior gradient mask generated by the pruning criterion during fine-tuning. PGMPF has three charming characteristics over previous works: (1) Pruning-aware network fine-tuning. A typical pruning pipeline consists of training, pruning and fine-tuning, which are relatively independent, while PGMPF utilizes a variant of the pruning mask as a prior gradient mask to guide fine-tuning, without complicated pruning criteria. (2) An excellent tradeoff between large model capacity during fine-tuning and stable convergence speed to obtain the final compact model. Previous works preserve more training information of pruned parameters during fine-tuning to pursue better performance, which would incur catastrophic non-convergence of the pruned model for relatively large pruning rates, while our PGMPF greatly stabilizes the fine-tuning phase by gradually constraining the learning rate of those "unimportant" parameters. (3) Channel-wise random dropout of the prior gradient mask to impose some gradient noise to fine-tuning to further improve the robustness of final compact model. Experimental results on three image classification benchmarks CIFAR10/100 and ILSVRC-2012 demonstrate the effectiveness of our method for various CNN architectures, datasets and pruning rates. Notably, on ILSVRC-2012, PGMPF reduces 53.5% FLOPs on ResNet-50 with only 0.90% top-1 accuracy drop and 0.52% top-5 accuracy drop, which has advanced the state-of-the-art with negligible extra computational cost.

Introduction

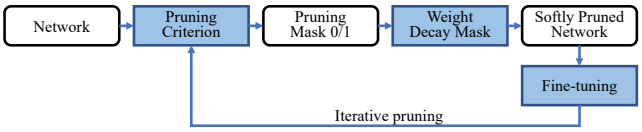
Despite the superior performance of deep Convolutional Neural Networks in various tasks, e.g., image classification (He et al. 2016; Xu et al. 2021), object detection (Bochkovskiy, Wang, and Liao 2020), image retrieval (Hu et al. 2020), semantic segmentation (He et al. 2017), the deployment of CNNs to resource-limited mobile devices have posed great challenges. Network pruning is a powerful method to compress the model with little performance loss, which can be divided into two categories: *weight pruning* and *filter pruning* based on the granularity (Zhu and Gupta 2018; Liu et al. 2019c; Frankle and Carbin 2019). Weight pruning methods remove unimportant connections or weights in the network, inducing unstructured sparsity in filters, thus requiring specialized libraries for real acceleration. In contrast, filter pruning structurally remove unimportant filters, capable of compressing both the model size and the computational burden. Hence we focus on *filter pruning*.

A typical three-step filter pruning pipeline consists of: training a network, evaluating the importance of every filter to generate a pruning mask to mask out unimportant filters and then fine-tuning the pruned network to compensate for the performance degradation incurred by pruning. The pruning and fine-tuning phases could be iteratively used to greedily compress the model.

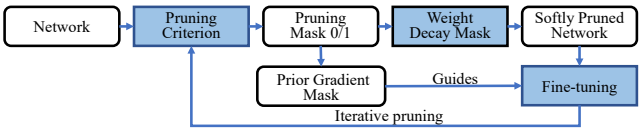
These phases are relatively independent as the pruning operation is non-differentiable, while our Prior Gradient



(a) Hard Filter Pruning



(b) Asymptotic SofteR Filter Pruning



(c) PGMPF

Figure 1: Comparison of three kinds of pruning pipelines. Our PGMPF devises a prior gradient mask generated by the Boolean pruning mask at the pruning stage to guide the next fine-tuning stage, making the fine-tuning stage to be pruning-aware. Weight decay mask, proposed in Asymptotic SofteR Filter Pruning (ASRFP), is a variant of the Boolean pruning mask to soften the pruning operation to maintain training information inside those pruned filters.

*Corresponding author.
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Mask Guided Pruning-aware Fine-tuning (PGMPF) utilizes a modified version of the pruning mask generated by the pruning stage as a prior gradient mask to guide fine-tuning, as shown in Figure 1. Previous Soft Filter Pruning (SFP) based methods, e.g., Asymptotic Soft Filter Pruning (ASFP) and Asymptotic SofteR Filter Pruning (ASRFP) (He et al. 2018, 2019a; Cai et al. 2021b), also allow pruned filters to update their parameters to maintain a large model capacity during fine-tuning to pursue better performance, shown in Figure 1(b), where weight decay mask is a modified version of the Boolean pruning mask to smoothly soften the pruning operation in order to maintain more training information inside those filters chosen to be pruned. However, these methods confronted with catastrophic non-convergence of the pruned model for relatively large pruning rates. The *"catastrophic non-convergence of the pruned model for large pruning rates"* means that the **Test Accuracy Drop** before and after pruning would be very huge, where **0** denotes no evident accuracy drops incurred by pruning. Note that soft pruning based methods allow all filters to unconstrainedly update the parameter during fine-tuning, ignoring the uneven importance of filters.

Unlike Hard Filter Pruning (HFP) that disables the update of pruned filters, gradually reducing the model capacity or SFP based methods that encounter catastrophic non-convergence of the pruned model, our PGMPF allows the update of pruned filters via a prior gradient mask generated by the pruning criterion, striking an excellent tradeoff between large model capacity during fine-tuning and stable convergence speed to obtain the final compact model.

Our contribution points are as follows: (1) We proposed a novel Prior Gradient Mask Guided Pruning-aware Fine-tuning (PGMPF) method to compress and accelerate deep models, which provides state-of-the-art performance without complicated handcrafted or learnt pruning criteria. (2) Our PGMPF greatly stabilizes the fine-tuning phase by gradually constraining the learning rate of those "unimportant" parameters, achieving an excellent tradeoff between large model capacity during fine-tuning and stable convergence speed to obtain the final compact model. (3) We proposed channel-wise random dropout of the prior gradient mask to impose some gradient noise to fine-tuning to further improve the robustness of final compact model.

Related Works

Prevalent works on compressing and accelerating CNN models mainly consist of *network pruning*, *knowledge distillation*, *model quantization*, *low-rank approximation* and *efficient network module design*.

Network pruning focuses on compressing the model without incurring obvious performance loss. Recently, much attention has been paid to filter pruning, since filter pruning is much friendlier to hardwares, capable of compressing both the model size and the computational cost. Until now, diverse filter pruning methods have been proposed.

In addition, pruning can be categorized into **static pruning** and **dynamic pruning**. Static pruning removes unimportant filters statically, eventually obtaining a fixed and static compact model invariant to different inputs. In contrast, given a unique input, dynamic pruning uses channel-wise or spatial-wise attention modules to adaptively predict the importance of each channel and skip the computation of unimportant channels and locations, or replace the computation with a low-precision version (Hua et al. 2019; Gao et al. 2019; Liu et al. 2020). Even though dynamic pruning surpasses static pruning by learning instance-level network activation paths, potential drawbacks are that the model size is not compressed and the actual inference speed is hindered by the computational cost of reindexing the dynamic network structure for each input (Chen et al. 2019; Liu et al. 2019a). Thus we focus on static pruning.

Pruning Criteria. Existing criteria for evaluating the importance of a filter mainly include ℓ_1-norm, ℓ_2-norm, weight similarity, feature redundancy, scaling factors in Batch Normalization layers, the rank of the feature map, cross-layer weight dependency and so on (Li et al. 2016; Liu et al. 2017; Ayinde and Zurada 2018; Wang et al. 2019; Lin et al. 2020). Some approaches compare the importance of each filter layer-wisely, while others compare the importance in the while network. A potential disadvantage of global pruning is that how to design a global filter importance criterion as magnitudes of filters vary from layer to layer. Recently, Channel Pruning via Multi-Criteria (CPMC) method takes three aspects, i.e., cross-layer filter dependency, the parameter numbers and FLOPs of each filter into account, and then normalizes these criteria to generate a global multi-criteria importance to measure the importance in a global manner (Yan et al. 2021). Filter Pruning via Geometric Median (FPGM) approach prunes filters via Geometric Median, claiming that the prevalent smaller-norm-less-important criterion demands large deviation of filter norms and near zero norms of unimportant filters (He et al. 2019b). AutoPruner proposes to use a channel-wise attention module and a scaled sigmoid function to gradually scale each channel and find unimportant filters automatically during training (Luo and Wu 2020), however, evidently increasing training-time computational costs and requiring heavy tuning of parameters in the scaled sigmoid function for each network and dataset.

Inspired by Differentiable Architecture Search (DARTS) (Liu, Simonyan, and Yang 2019), Learning Filter Pruning Criteria (LFPC) proposes a Differentiable Criteria Sampler (DCS) to learn layer-wise importance criteria (He et al. 2020b), which is computationally expensive and time-consuming. MetaPruning adopts Meta Learning and evolutionary algorithm for automatic channel pruning, whose training cost is very expensive (Liu et al. 2019b). Likewise, EagleEye also relies on evolutionary algorithm together with adaptive batch normalization to search an optimal structure (Li et al. 2020). In short, how to design a pruning criterion is still an open issue.

In contrast, our proposed PGMPF does not rely on complicated handcrafted or learnt pruning criteria. For simplicity, we adopt the simple ℓ_2-norm criterion. We utilize a modified version of the pruning mask generated by the pruning stage as a prior gradient mask to guide fine-tuning. Unlike conventional HFP based methods which disable the update

of pruned filters, gradually reducing the model capacity, our proposed PGMPF allows the update of pruned filters via a prior gradient mask generated by the pruning criterion, balancing well between large search space during fine-tuning and stable convergence speed to obtain the pruned model.

Gradually Hard Filter Pruning (GHFP) (Cai et al. 2021a) alleviates the issue of catastrophic non-convergence of the pruned model via a monotonically increasing parameter to control the proportion of soft pruning and hard pruning to balance between performance and convergence speed. While GHFP still suffers from relatively large pruning rates, our PGMPF greatly stabilizes the fine-tuning phase by gradually constraining the learning rate of those "unimportant" parameters. Moreover, our PGMPF are totally pruning-aware, meaning that the pruning phase could intimately affect the fine-tuning phase via our prior gradient mask, while in most previous methods, pruning and fine-tuning are relatively independent, shown in Figure 2.

Low-rank approximation of convolutional filters reduces model size and computation by decomposing large matrices into small matrices, however, achieving relatively tiny speedups on small-size convolutional kernels (Jaderberg, Vedaldi, and Zisserman 2014; Alvarez and Salzmann 2017). Model quantization quantizes the weights and activations into fewer bits to reduce model size and computational budgets (Hubara et al. 2016; Han et al. 2020). Efficient network module design aims at designing more lightweight modules, e.g., MobileNet, CondConv, ACNet, HCGNet (Howard et al. 2017; Yang et al. 2019; Ding et al. 2019; Yang et al. 2020). Knowledge distillation (KD) methods define various knowledge, e.g., the activation or attention map (Hinton, Vinyals, and Dean 2015; Yuan et al. 2020; Yang, An, and Xu 2021), and then transfer the knowledge from a large teacher model to a small student model, which could be regarded as a kind of instance-level label smoothing. Recently, self-supervised learning (SSL) (Chen et al. 2020; He et al. 2020a; Yang et al. 2021) is defined as one kind of knowledge to improve the performance (Yin et al. 2020), introducing auxiliary tasks, e.g., rotation, jigsaw, image inpainting, color distortion, to push the model to learn more generalized or task-specific representations. These approaches can be combined with our PGMPF to achieve further improvement.

Methods

Formulation

For a network with L convolutional layers, the weight of the l-th convolutional layer W_l can be denoted by $\mathbb{R}^{n\times m\times s\times s}$, where $1 \le l \le L$. In detail, s denotes the kernel size. m and n are the number of input channels and output channels respectively. We denote I_l and O_l as the input and output feature maps of the l-th layer. The shape of the input tensor I_l and the output tensor $O_l = W_l * I_l$ are $m \times h_l \times w_l$ and $n \times h_{l+1} \times w_{l+1}$ respectively, represented as

$$O_{l,j} = W_{l,j} * I_l \ \ for\ 1 \le j \le n, \tag{1}$$

where $O_{l,j} \in \mathbb{R}^{h_{l+1}\times w_{l+1}}$ and $W_{l,j} \in \mathbb{R}^{m\times s\times s}$ denote the j-th output channel and the j-th filter individually in the l-th layer. If the filter pruning rate for the l-th layer is P_l, then $n \times P_l$ filters in the l-th layer would be removed. After pruning, the size of the pruned output tensor $\hat{O}_l$ is $n \times (1 - P_l) \times h_{l+1} \times w_{l+1}$.

Pruning Mask. During pruning, given the weight tensor W_l and the pruning rate P_l, we adopt a simple ℓ_2-norm filter importance criterion to generate a Boolean pruning mask $M_{l,j}$. Specifically, $M_{l,j} = 0$ if $W_{l,j}$ is pruned. Otherwise, $M_{l,j} = 1$ means that the filter $W_{l,j}$ is not pruned.

According to ASRFP, the pruned weights of the l-th layer are gradually zeroized, given by

$$\hat{W}_{l,j} = W_{l,j} \odot M_{l,j} + \alpha\, W_{l,j} \odot (1 - M_{l,j}) \ \ for\ 1 \le j \le n, \tag{2}$$

where $\odot$ denotes the element-wise multiplication. α is a monotonically decreasing parameter to control the decaying speed of pruned filters and to better utilize the trained information of pruned filters. ASRFP exponentially decays α from 1 towards 0 as the pruning and fine-tuning procedure goes on.

Prior Gradient Mask Guided Pruning-aware Fine-tuning

As shown in Figure 2, the prior gradient mask categorizes filters into important ones and unimportant ones, based on the ℓ_2-norm of each filter. The closer a filter gets to the center of concentric circles, the less important the filter is. The weight decay mask at the pruning stage would push unimportant filters towards the center via a monotonically decreasing parameter α. During fine-tuning, Both unconstrained fine-tuning (UFT) and PGMPF calculate the gradient as normal, and the aimed direction of gradient update is denoted by solid arrow. Then we can get the new position of each filter. UFT just moves each filter to the aimed position, ignoring the pruning objective, treating each filter as equally important during fine-tuning, which would incur catastrophic non-convergence of the pruned model for relatively large pruning rates. Our PGMPF is pruning-aware, gradually scaling down the learning rate of unimportant filters. Besides, the guidance of a prior gradient mask obtained in the last pruning stage would continue for an epoch. After each fine-tuning epoch, **the roles of important and unimportant filters may change**, and a new prior gradient mask can be obtained.

After obtaining the Boolean pruning mask $M_{l,j}$, we define a modified asymptotic variant $\hat{M}_{l,j}$, named as prior gradient mask, given by

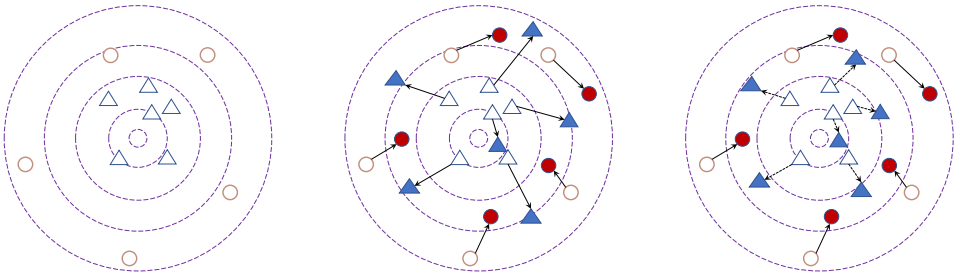
$$\hat{M}_{l,j} = M_{l,j} + \beta(1 - M_{l,j}) \ \ for\ 1 \le j \le n, \tag{3}$$

where β constrains the learning rate of those pruned parameters, decreasing from 1 to 0, given by

$$\beta(t) = \left(\frac{t_{max} - 1 - t}{t_{max} - 1}\right)^3 \ \ for\ 0 \le t < \ t_{max}, \tag{4}$$

where t_{max} is the maximal number of pruning and fine-tuning epochs.

Once we obtain the prior gradient mask $\hat{M} = \{\hat{M}_{l,j} | l \in [1, L], j \in [1, n]\}$, we adopt it to guide the next fine-tuning stage. Assume that g_t is the normal gradient computed by regular backpropagation during fine-tuning in the t-th epoch.



(a) Importance Assignment (b) UFT (c) PGMPF

Figure 2: Comparison of UFT in SFP based methods and our PGMPF. (a) The prior gradient mask categorizes filters into important ones and unimportant ones, denoted by red solid circles and blue solid triangles respectively. Purple dotted concentric circles indicate the importance of each filter, measured by the ℓ_2-norm of the filter. (b) During fine-tuning, Both UFT and PGMPF calculate the gradient as normal, and the aimed direction of gradient update is denoted by solid arrow. Then we can get the new position of each filter, denoted by blue filled triangle or red filled circle. UFT just moves each filter to the aimed position, ignoring the pruning objective. (c) PGMPF scales down the learning rate of those unimportant filters, denoted by dashed arrows. PGMPF is pruning-aware, gradually constraining the learning rate of those "unimportant" parameters.

We impose our prior gradient mask to constrain the learning rate of those unimportant parameters determined by the last pruning stage, and obtain a modified gradient $\hat{g}_t$, given by

$$\hat{g}_t = g_t \odot \hat{M} = g_t \odot M + \beta\, g_t \odot (1 - M) \quad for\ 0 \leq t < t_{max}, \quad (5)$$

where $M = \{M_{l,j} | l \in [1, L], j \in [1, n]\}$.

Then with the current learning rate η, we update the weight tensor by

$$\tilde{W} = W - \eta \cdot \hat{g}_t = W - \eta \cdot \hat{M} \odot g_t. \quad (6)$$

where $\hat{M}_{l,j} = 1$ for unpruned filters and $\hat{M}_{l,j} = \beta$ for pruned filters. In effect, we inject a prior gradient mask into the learning rate to penalize the update step of those filters considered as unimportant by the pruning criterion, while other filters will not be affected. Specifically, previous SFP based methods can be regarded as a special case when $\hat{M}_{l,j} = 1$ for all filters, regardless of the importance measured by the pruning criterion. Maintaining a large model capacity during fine-tuning would encounter catastrophic non-convergence of the pruned model for relatively large pruning rates, while our PGMPF greatly stabilizes the fine-tuning phase by gradually constraining the learning rate of those "unimportant" parameters.

Random Dropout of the Prior Gradient Mask. During fine-tuning, evert time before we update parameters, we adopt channel-wise random dropout of the prior gradient mask to impose some gradient noise to fine-tuning to further improve the robustness of final compact model. In detail, we define a random matrix $R = \{R_{l,j} | R_{l,j} \sim Bernoulli(p), l \in [1, L], j \in [1, n]\}$, where we set $p = 0.5$ for simplicity. Then we multiply $R_{l,j}$ with $\hat{M}_{l,j}$ to obtain a random dropout version of prior gradient mask, denoted by $R_{l,j} \odot \hat{M}_{l,j}$, where $R_{l,j}$ is shared within the same filter. Hence, the modified update rule is given by

$$\tilde{W} = W - \eta \cdot R \odot \hat{M} \odot g_t. \quad (7)$$

where $\hat{M}$ is fixed within a single fine-tuning epoch and R is fixed within each batch. The Eq.(7) assumes the optimizer to be SGD (Stochastic Gradient Descent). Yet it can be naturally extended to SGD with Momentum.

We present our PGMPF method in Algorithm 1. By default, we set $\alpha_0 = 1$ following ASRFP and the probability of using the prior gradient mask $p = 0.5$.

Experimental Results

Experimental Settings

We empirically evaluate our PGMPF for VGGNet and ResNet (Simonyan and Zisserman 2015; He et al. 2016) on three datasets: CIFAR-10/100 and ILSVRC-2012 (Krizhevsky 2009; RussakovskyOlga et al. 2015). Both CIFAR-10 and CIFAR-100 consist of 50,000 training images and 10,000 test images of size 32×32 pixels, drawn from 10 classes and 100 classes respectively. ILSVRC-2012 contains 1.28 million training images and 50k validation images divided into 1,000 classes.

On CIFAR-10/100, we follow the parameter scheme and the training configuration in GHFP and CPMC. On ILSVRC-2012, we follow the parameter setting and the data augmentation scheme in ASFP. The total number of pruning and fine-tuning epochs CIFAR-10/100 and ILSVRC-2012 are 200 and 100 respectively, following the settings of ASFP, ASRFP and GHFP (He et al. 2019a; Cai et al. 2021b,a).

Models are either pruned from scratch or pruned from pre-trained models. For pruning pre-trained models, we set the initial learning rate as one-tenth of the original learning rate. We compare our methods with other state-of-the-art methods, e.g., ASFP, ThiNet, AutoPruner, GHFP, CPMC, FPGM, ASRFP, PARI (Cai et al. 2021c).

Single-Branch Network Pruning

VGG16 on CIFAR-10/100. We compare our PGMPF with several state-of-the-art structural pruning algorithms. (1) GAL utilizes generative adversarial learning (GAL) (Lin et al. 2019) to optimize the network structure. (2) VCNNP (Zhao et al. 2019) is a variational Bayesian framework for channel pruning. (3) HRank (Lin et al. 2020) regards the Rank of the feature map as a criterion to eval-

Algorithm 1: PGMPF Algorithm

inputs : training set: X, final pruning rate: P_l, initial decay rate: α_0, the model with parameters $W = \{W_i, 0 \leq i \leq L\}$.
output: The pruned model with parameters $W^* = W^{t_{max}}$
Initialize $\beta(0) = 1$, $p = 0.5$ and pruning rate $P_l(0) = 0$
Initialize prior gradient mask $\hat{M}^{-1}$ with all ones
for $t = 0, ..., t_{max} - 1$ **do**
  Decrease weight decay rate α based on SRFP
  Decrease β with Eq.(4)
  Increase pruning rate $P_l(t)$ based on ASFP
  for *each batch in* X **do**
    Draw a random $R = \{R_{l,j} | R_{l,j} \sim Bernoulli(p)\}$
    Compute the gradient g_t
    Update the weight by $\tilde{W}^t = W^t - \eta \cdot R \odot \hat{M}^{t-1} \odot g_t$
  end
  Get trained model parameters $\hat{W}^{t+1}$
  for $l = 1, ..., L$ **do**
    Compute the ℓ_2-norm of each filter $\|\hat{W}_{l,j}^{t+1}\|_2, 1 \leq j \leq n$
    Generate a pruning mask $M_{l,j}^t$
    Obtain the prior gradient mask $\hat{M}_{l,j}^t$ via Eq.(3)
    Select $n \times P_l$ filters with minimal ℓ_2-norm values to be softly pruned via α
  end
  Get the pruned model parameters W^{t+1} based on $\hat{W}^{t+1}$
end
Get the pruned model with final parameters $W^* = W^{t_{max}}$

uate the importance of each filter. (4) CPGMI (Lee et al. 2020) uses gradients of mutual information to measure the importance of each filter. (5) CPMC simultaneously considers cross-layer filter dependency, the parameter numbers and FLOPs of each filter, and then normalizes these aspects to get a global multi-criteria importance of each filter.

Unlike above methods, we do not rely on complicated pruning criteria. For simplicity, we adopt the simple ℓ_2-norm criterion. We utilize a modified version of the pruning mask generated by the pruning stage as a prior gradient mask to guide fine-tuning, balancing well between large search space during fine-tuning and stable convergence speed to obtain the pruned model. Results are shown in Table 1 and Table 2, where "PGMPF-cfg1" means using a simple pre-defined layer-wise pruning rate configuration to prune each layer with the same pruning rate and "PGMPF-cfg2" means using a layer-wise pruning rate configuration that gradually increasing the pruning rate from shallow layers to deep layers. Besides, the "baseline" denotes the test accuracy of the unpruned model. In Table 2, "PGMPF-SFP" means using the constant pruning rate strategy in SFP, while we use the asymptotic pruning rate strategy in ASFP and GHFP by default. Unless specifically clarified, we prune each layer with the same pruning rate for simplicity, which may put our method at a disadvantage. Still, PGMPF outperforms other methods with a simple pruning rate configuration "PGMPF-cfg1". With better pruning rate configuration "PGMPF-cfg2", our method outperform other methods by a moderate margin.

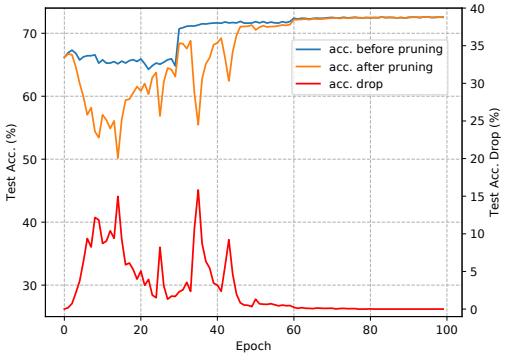


Figure 3: Test accuracies before and after pruning, as well as the accuracy drop of the pre-trained ResNet-34 on ILSVRC-2012 during fine-tuning when the pruning rate is 30%.

Multiple-Branch Network Pruning

ResNet on CIFAR-10/100. On ResNet-20/56, We mainly compare our PGMPF with most related methods, i.e., SFP, ASFP, ASRFP and GHFP as they also maintain a large model capacity during fine-tuning and the training costs are roughly the same. As shown in Table 1 and Table 2, our PGMPF evidently outperforms other methods. For example, when pruning pre-trained ResNet-56 for CIFAR-10, ASFP, ASRFP and GHFP accelerate ResNet-56 by 72.6% speedup ratio with 5.13%, 4.31% and 2.31% accuracy drops respectively, while our PGMPF further narrows the gap to 1.70%. When pruning ResNet-20 from scratch for CIFAR-100, ASFP, ASRFP and GHFP accelerate ResNet-56 by 29.3% speedup ratio with 1.97%, 2.44% and 2.07% accuracy drops respectively, while our PGMPF further reduces the gap to 1.30%. Our PGMPF greatly stabilizes the fine-tuning phase by gradually constraining the learning rate of those "unimportant" parameters.

ResNet on ILSVRC-2012. For ILSVRC-2012, we evaluate our PGMPF on ResNet-18/34/50 with the same pruning rate for each layer, following the same settings in SFP, ASFP and ASRFP. For pruning pre-trained models, we use the official pre-trained models provided by the Pytorch library. As shown in Table 3, PGMPF still outperforms previous methods, even though the importance criterion and the pruning rate configuration we use are quite simple, which means that our PGMPF could greatly relax the need of complicated importance criteria and pruning rate configurations. Moreover, unlike other methods, e.g. AutoPruner, MetaPruning, we do not introduce obvious training burdens because we just elegantly post-process the learning rate of those unimportant filters, without extra learnt parameters.

AutoPruner accelerates pre-trained ResNet-50 by 51.2% speedup ratio with 1.39% top-1 accuracy drops, relying on an extra channel-wise attention module and a scaled sigmoid function to find unimportant filters. For pre-trained

Table 1: Pruning results on CIFAR-10.

Model	Alg	Pre-trained?	Baseline (%)	Accu. (%)	Accu. Drop (%)	FLOPs	Pruned FLOPs(%)
VGG16	GAL-0.05	✓	93.68	92.03	1.65	1.89E8	39.6
	VCNNP	✓	93.68	93.18	0.50	1.90E8	39.1
	HRank	✓	93.68	92.34	1.34	1.09E8	65.3
	CPMC	✓	93.68	93.40	0.28	1.07E8	66.0
	PGMPF-cfg1	✓	93.68	93.46	0.22	1.07E8	66.0
	PGMPF-cfg2	✓	93.68	**93.60**	**0.08**	1.07E8	**66.0**
ResNet-20	ASFP	×	92.89	90.57	2.32	2.43E7	54.0
	ASRFP	×	92.89	90.65	2.24	2.43E7	54.0
	GHFP	×	92.89	90.82	2.07	2.43E7	54.0
	PGMPF	×	92.89	**91.54**	**1.35**	2.43E7	**54.0**
ResNet-56	GHFP	×	94.85	92.08	2.77	3.43E7	72.6
	PGMPF	×	94.85	**92.81**	**2.14**	3.43E7	**72.6**
	ASFP	✓	94.85	89.72	5.13	3.43E7	72.6
	ASRFP	✓	94.85	90.54	4.31	3.43E7	72.6
	GHFP	✓	94.85	92.54	2.31	3.43E7	72.6
	PGMPF	✓	94.85	**93.15**	**1.70**	3.43E7	**72.6**

Table 2: Pruning results on CIFAR-100.

Model	Alg	Pre-trained?	Baseline (%)	Accu. (%)	Accu. Drop (%)	FLOPs	Pruned FLOPs(%)
VGG16	VCNNP	✓	73.80	73.33	0.47	2.56E8	18.0
	CPGMI	✓	73.80	73.53	0.27	1.98E8	37.1
	CPMC	✓	73.80	73.01	0.79	1.62E8	48.4
	PGMPF	✓	73.80	73.45	0.35	1.63E8	48.2
	PGMPF-SFP	✓	73.80	**73.66**	**0.14**	1.63E8	**48.2**
ResNet-20	SFP	×	68.11	66.23	1.88	2.87E7	29.3
	ASFP	×	68.92	66.95	1.97	2.87E7	29.3
	ASRFP	×	68.92	66.48	2.44	2.87E7	29.3
	GHFP	×	68.92	66.85	2.07	2.87E7	29.3
	PGMPF	×	68.92	**67.62**	**1.30**	2.87E7	**29.3**
ResNet-56	ASFP	×	72.92	69.35	3.57	5.94E7	52.6
	ASRFP	×	72.92	69.16	3.76	5.94E7	52.6
	GHFP	×	72.92	69.62	3.30	5.94E7	52.6
	PGMPF	×	72.92	**70.21**	**2.71**	5.94E7	**52.6**

ResNet-50, SRFP and FPGM accelerate ResNet-50 by 53.5% speedup ratio with 4.04% and 1.32% top-1 accuracy drops respectively, while our PGMPF further reduces the gap to 0.90%. Meta-Pruning is more like a neural architecture search (NAS) approach than a pruning method, which relies on evolutionary algorithm to search an optimal structure. Consequently, the training-phase computational costs are extremely heavy. Meta-Pruning accelerates ResNet-50 by 50.0% speedup ratio with 1.20% top-1 accuracy drops, which is surpassed by our lightweight PGMPF.

Convergence Analysis. We present test accuracies before and after pruning, as well as the accuracy drop of the pre-trained ResNet-34 on ILSVRC-2012 during fine-tuning when the pruning rate is 30% in Figure 3. The Test Accuracy Drop is the difference between the Top-1 accuracy before pruning and the Top-1 accuracy after pruning, where 0 denotes no evident accuracy drops incurred by pruning. Our PGMPF allows pruned filters to update their parameters during fine-tuning, thus maintaining a relatively large model capacity to obtain better performance before pruning. Nevertheless, the test accuracy drop caused by pruning reaches 15% in the first half of the training epochs due to the contradiction between large model capacities during fine-tuning and the need for a compact model. With our PGMPF, we gradually constrain the learning rate of those "unimportant" parameters. Thus, we asymptotically reduce the test accuracy drop caused by pruning to nearly 0 while still maintaining a relatively large model capacity during fine-tuning.

Ablation Study

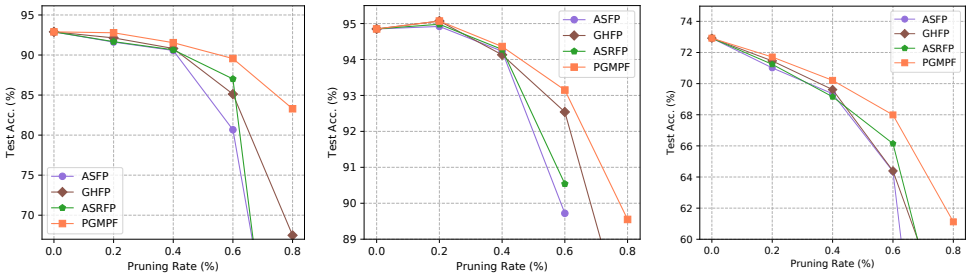
We conducted extensive ablation experiments to analyze the effect of our PGMPF.

Varying pruning rates. We present test accuracies of various pruning rates for ResNet-20/56 on CIFAR-10/100 in Fig. 4(a), Fig. 4(b) and Fig. 4(c), where ResNet-20 is pruned from scratch and ResNet-56 is trained from a pre-trained model on CIFAR-10. On CIFAR-100, ResNet-56 is pruned from scratch. As the pruning rate increases, the test accuracies of our PGMPF decline much steadier than those of ASFP, ASRFP and GHFP. Our PGMPF significantly surpasses other methods in terms of stability and test accuracy across various pruning rates on different datasets, achieving an elegant tradeoff between large model capacity during fine-tuning and stable convergence speed to obtain the final compact model by gradually constraining the learning rate of those "unimportant" parameters.

Influence of dropout type of prior gradient mask. While we use channel-wise random dropout of prior gradient mask by default, here, we present test accuracies of different dropout types on CIFAR-10, shown in Table 4. Layer-wise dropout is a simple extension of channel-wise dropout that all filters in each layer shares the same random variable. Compared with not using any random dropout, in general, channel-wise random dropout could further improve the per-

Table 3: Pruning results on ImageNet.

Model	Alg	Pre-trained?	Top-1 Baseline(%)	Top-1 (%)	Top-5 Baseline(%)	Top-5 (%)	Top-1 Accu. Drop(%)	Top-5 Accu. Drop(%)	Pruned FLOPs(%)
ResNet-18	ASFP	×	70.23	66.02	89.51	86.92	4.21	2.59	53.5
	ASRFP	×	70.23	66.35	89.51	87.06	3.88	2.45	53.5
	PGMPF	×	70.23	**66.67**	89.51	**87.36**	**3.56**	**2.15**	53.5
ResNet-34	ASFP	×	73.27	68.79	91.43	88.95	4.48	2.48	52.7
	ASRFP	×	73.27	70.42	91.43	89.63	2.85	1.80	52.7
	PGMPF	×	73.27	**70.64**	91.43	**89.87**	**2.63**	**1.56**	**52.7**
	ASRFP	✓	73.27	69.82	91.43	89.51	3.45	1.92	52.7
	PGMPF	✓	73.27	**71.59**	91.43	**90.45**	**0.78**	**0.98**	**52.7**
ResNet-50	ThiNet	✓	72.88	72.04	91.14	90.67	0.84	0.47	36.7
	SFP	✓	76.15	62.14	92.87	84.60	14.01	8.27	41.8
	PARI	✓	76.15	75.08	92.87	92.49	1.07	0.38	42.2
	Meta-Pruning	-	76.60	75.40	-	-	1.20	-	50.0
	SRFP	✓	76.13	72.09	92.86	90.75	4.04	2.11	53.5
	AutoPruner	✓	76.15	74.76	92.87	92.15	1.39	0.72	51.2
	FPGM	✓	76.15	74.83	92.87	92.32	1.32	0.55	53.5
	PGMPF	✓	76.01	**75.11**	92.93	**92.41**	**0.90**	**0.52**	**53.5**



(a) ResNet-20 on CIFAR-10 (b) ResNet-56 on CIFAR-10 (c) ResNet-56 on CIFAR-100

Figure 4: Pruning results of ResNet-20/56 on CIFAR-10/100 among ASFP/ASRFP/GHFP/PGMPF with diverse pruning rates.

formance of our prior gradient mask.

Influence of weight decay mask. Weight decay mask is a variant of the Boolean pruning mask to smoothly soften the pruning operation to maintain more training information inside those pruned filters in the early stage of iterative pruning retraining, controlled by α_0. When $\alpha_0 = 0$, the information of those pruned filters is totally discarded, which is equivalent to normal pruning operation all the time, as in SFP and ASFP. While we use $\alpha_0 = 1$ as SRFP and ASRFP by default, here, we present test accuracies of two kinds of α_0 on CIFAR-10, shown in Table 5. While $\alpha_0 = 1$ is a better choice, our PGMPF is very insensitive to α_0, which reveals the stability of our approach.

Conclusion

In short, we propose a novel pruning-aware network fine-tuning framework PGMPF to accelerate deep CNNs. Unlike previous methods that require complicated pruning criteria or heavy training costs, our PGMPF elegantly unifies pruning and fine-tuning using our prior gradient mask together with channel-wise random dropout without introducing obvious training burdens. Thanks to this, an excellent trade-off between large model capacity during fine-tuning and stable convergence speed to obtain the final compact model is achieved. Extensive experiments demonstrate the effectiveness of our method for various CNN architectures, datasets and pruning rates.

Table 4: Influence of dropout type of the prior gradient mask on CIFAR-10.

Model	Pruned percent(%)	dropout Type	Accu.(%)	FLOPs(PR%)
ResNet-20	20	-	92.34	2.87E7(29.3)
	20	layer-wise	92.60	2.87E7(29.3)
	20	channel-wise	**92.78**	2.87E7(29.3)
ResNet-56	20	-	93.84	8.98E7(28.4)
	20	layer-wise	94.25	8.98E7(28.4)
	20	channel-wise	**95.07**	8.98E7(28.4)
	40	-	93.74	5.94E7(52.6)
	40	layer-wise	93.63	5.94E7(52.6)
	40	channel-wise	**94.36**	5.94E7(52.6)

Table 5: Influence of α_0 on CIFAR-10.

Model	Pruned percent(%)	α_0	Accu.(%)	FLOPs(PR%)
ResNet-20	20	0	92.56	2.87E7(29.3)
	20	1	**92.78**	2.87E7(29.3)
	40	0	91.49	2.43E7(54.0)
	40	1	**91.54**	2.43E7(54.0)
ResNet-56	20	0	94.97	8.98E7(28.4)
	20	1	**95.07**	8.98E7(28.4)
	40	0	**94.47**	5.94E7(52.6)
	40	1	94.36	5.94E7(52.6)

References

Alvarez, J. M.; and Salzmann, M. 2017. Compression-aware training of deep networks. *Advances in Neural Information Processing Systems*, 2017-December(Nips): 857–868.

Ayinde, B. O.; and Zurada, J. M. 2018. Building Efficient ConvNets using Redundant Feature Pruning. 1–9.

Bochkovskiy, A.; Wang, C.-Y.; and Liao, H. 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv*, abs/2004.10934.

Cai, L.; An, Z.; Yang, C.; and Xu, Y. 2021a. Soft and Hard Filter Pruning via Dimension Reduction. *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Cai, L.; An, Z.; Yang, C.; and Xu, Y. 2021b. Softer Pruning, Incremental Regularization. *2020 25th International Conference on Pattern Recognition (ICPR)*, 224–230.

Cai, Y.; Yin, Z.; Guo, K.; and Xu, X. 2021c. Pruning the Unimportant or Redundant Filters? Synergy Makes Better. *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Chen, J.; Zhu, Z.; Li, C.; and Zhao, Y. 2019. Self-Adaptive Network Pruning. *ArXiv*, abs/1910.08906.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *ArXiv*, abs/2002.05709.

Ding, X.; Guo, Y.; Ding, G.; and Han, J. 2019. ACNet: Strengthening the Kernel Skeletons for Powerful CNN via Asymmetric Convolution Blocks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1911–1920.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv: Learning*.

Gao, X.; Zhao, Y.; Dudziak, L.; Mullins, R.; and Xu, C. 2019. Dynamic Channel Pruning: Feature Boosting and Suppression. *ArXiv*, abs/1810.05331.

Han, K.; Wang, Y.; Xu, Y.; Xu, C.; Wu, E.; and Xu, C. 2020. Training Binary Neural Networks through Learning with Noisy Supervision. *ArXiv*, abs/2010.04871.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. B. 2020a. Momentum Contrast for Unsupervised Visual Representation Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9726–9735.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December: 770–778.

He, Y.; Ding, Y.; Liu, P.; Zhu, L.; Zhang, H.; and Yang, Y. 2020b. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006–2015.

He, Y.; Dong, X.; Kang, G.; Fu, Y.; Yan, C.; and Yang, Y. 2019a. Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks. *IEEE Transactions on Cybernetics*, PP: 1–11.

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018. Soft filter pruning for accelerating deep convolutional neural networks. *IJCAI International Joint Conference on Artificial Intelligence*, 2018-July: 2234–2240.

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2019b. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4335–4344.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. 1–9.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, B.; Song, R.-J.; Wei, X.-S.; Yao, Y.; Hua, X.; and Liu, Y. 2020. PyRetri: A PyTorch-based Library for Unsupervised Image Retrieval by Deep Convolutional Neural Networks. *Proceedings of the 28th ACM International Conference on Multimedia*.

Hua, W.; Sa, C. D.; Zhang, Z.; and Suh, G. 2019. Channel Gating Neural Networks. In *NeurIPS*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *Advances in Neural Information Processing Systems*.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. In *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *... Science Department, University of Toronto, Tech. ...*.

Lee, M. K.; Lee, S. H.; Lee, S. H.; and Song, B. 2020. Channel Pruning Via Gradient Of Mutual Information For Light-Weight Convolutional Neural Networks. *2020 IEEE International Conference on Image Processing (ICIP)*, 1751–1755.

Li, B.; Wu, B.; Su, J.; Wang, G.; and Lin, L. 2020. EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning. In *ECCV*.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning Filters for Efficient ConvNets. (2016): 1–13.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020. HRank: Filter Pruning Using High-Rank Feature Map. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1526–1535.

Lin, S.; Ji, R.; Yan, C.; Zhang, B.; Cao, L.; Ye, Q.; Huang, F.; and Doermann, D. 2019. Towards Optimal Structured CNN Pruning via Generative Adversarial Learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2785–2794.

Liu, C.; Wang, Y.; Han, K.; Xu, C.; and Xu, C. 2019a. Learning Instance-wise Sparsity for Accelerating Deep Models. *ArXiv*, abs/1907.11840.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. *ArXiv*, abs/1806.09055.

Liu, L.; Deng, L.; Chen, Z.; Wang, Y.; Li, S.; Zhang, J.; Yang, Y.; Gu, Z.; Ding, Y.; and Xie, Y. 2020. Boosting Deep Neural Network Efficiency with Dual-Module Inference. In *ICML*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning Efficient Convolutional Networks through Network Slimming. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob: 2755–2763.

Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, K.; and Sun, J. 2019b. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3295–3304.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019c. Rethinking the Value of Network Pruning. *ArXiv*, abs/1810.05270.

Luo, J. H.; and Wu, J. 2020. AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, 107.

RussakovskyOlga; DengJia; Suhao; KrauseJonathan; SatheeshSanjeev; MaSean; HuangZhiheng; KarpathyAndrej; KhoslaAditya; BernsteinMichael; BergAlexander, C.; and Fei-FeiLi. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Wang, W.; Fu, C.; Guo, J.; Cai, D.; and He, X. 2019. COP: Customized deep model compression via regularized correlation-based filter-level pruning. In *IJCAI International Joint Conference on Artificial Intelligence*. ISBN 9780999241141.

Xu, Y.; Wang, Q.; An, Z.; Wang, F.; Zhang, L.; Wu, Y.; Dong, F.; Qiu, C.-W.; Liu, X.; Qiu, J.; Hua, K.; Su, W.; Xu, H.; Han, Y.; Cao, X.; ju Liu, E.; Fu, C.; Yin, Z.; Liu, M.; Roepman, R.; Dietmann, S.; Virta, M.; Kengara, F.; Huang, C.; Zhang, Z.; Zhang, L.; Zhao, T.; Dai, J.; Yang, J.; Lan, L.; Luo, M.; Huang, T.; Liu, Z.; Qian, S.; An, T.; Liu, X.; Zhang, B.; He, X.; Cong, S.; Liu, X.; Zhang, W.; Wang, F.; Lu, C.; Cai, Z.; Lewis, J. P.; Tiedje, J. M.; and bing Zhang, J. 2021. Artificial Intelligence: A Powerful Paradigm for Scientific Research. *The Innovation*.

Yan, Y.; Li, C.; Guo, R.; Yang, K.; and Xu, Y. 2021. Channel Pruning via Multi-Criteria based on Weight Dependency. *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Yang, B.; Bender, G.; Le, Q. V.; and Ngiam, J. 2019. CondConv: Conditionally Parameterized Convolutions for Efficient Inference. In *NeurIPS*.

Yang, C.; An, Z.; Cai, L.; and Xu, Y. 2021. Hierarchical Self-supervised Augmented Knowledge Distillation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 1217–1223.

Yang, C.; An, Z.; and Xu, Y. 2021. Multi-View Contrastive Learning for Online Knowledge Distillation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3750–3754.

Yang, C.; An, Z.; Zhu, H.; Hu, X.; Zhang, K.; Xu, K.; Li, C.; and Xu, Y. 2020. Gated convolutional networks with hybrid connectivity for image classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 12581–12588.

Yin, J.; Qiu, J.; Zhang, S.; Ma, Z.; and Guo, J. 2020. SSKD: Self-Supervised Knowledge Distillation for Cross Domain Adaptive Person Re-Identification. *ArXiv*, abs/2009.05972.

Yuan, L.; Tay, F. E. H.; Li, G.; Wang, T.; and Feng, J. 2020. Revisiting Knowledge Distillation via Label Smoothing Regularization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3902–3910.

Zhao, C.; Ni, B.; yu Zhang, J.; Zhao, Q.; Zhang, W.; and Tian, Q. 2019. Variational Convolutional Neural Network Pruning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2775–2784.

Zhu, M.; and Gupta, S. 2018. To prune, or not to prune: exploring the efficacy of pruning for model compression. *ArXiv*, abs/1710.01878.