Practical Network Acceleration with Tiny Sets

Guo-Hua Wang Jianxin Wu
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, China
wangguohua@lamda.nju.edu.cn, wujx2001@nju.edu.cn

Abstract

Due to data privacy issues, accelerating networks with tiny training sets has become a critical need in practice. Previous methods mainly adopt filter-level pruning to accelerate networks with scarce training samples. In this paper, we reveal that dropping blocks is a fundamentally superior approach in this scenario. It enjoys a higher acceleration ratio and results in a better latency-accuracy performance under the few-shot setting. To choose which blocks to drop, we propose a new concept namely recoverability to measure the difficulty of recovering the compressed network. Our recoverability is efficient and effective for choosing which blocks to drop. Finally, we propose an algorithm named PRACTISE *to accelerate networks using only tiny sets of training images.* PRACTISE *outperforms previous methods by a significant margin. For 22% latency reduction,* PRACTISE *surpasses previous methods by on average 7% on ImageNet-1k. It also enjoys high generalization ability, working well under data-free or out-of-domain data settings, too. Our code is at https://github.com/DoctorKey/Practise.*

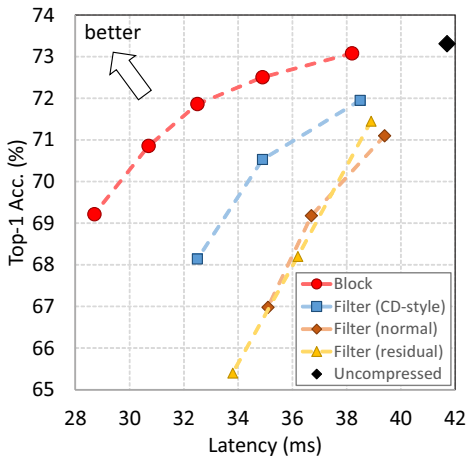

Figure 1. Comparison of different compression schemes with only 500 training images. We propose dropping blocks for few-shot network acceleration. Our method ('Block') outperforms previous methods dominantly for the latency-accuracy tradeoff. The ResNet-34 model was compressed on ImageNet-1k and all latencies were tested on an NVIDIA TITAN Xp GPU.

1. Introduction

In recent years, convolutional neural networks (CNNs) have achieved remarkable success, but they suffer from high computational costs. To accelerate the networks, many network compression methods have been proposed, such as network pruning [11, 17, 19, 21], network decoupling [6, 15] and network quantization [2, 7]. However, most previous methods rely on *the original training set* (i.e., all the training data) to recover the model's accuracy. But, to preserve data privacy and/or to achieve fast deployment, only scarce training data may be available in many scenarios.

For example, a customer often asks the algorithmic provider to accelerate their CNN models, but due to privacy concerns, the whole training data *cannot* be available. Only the raw uncompressed model and a few training examples are presented to the algorithmic provider. In some extreme cases, not even a single data point is to be provided. The algorithmic engineers need to synthesize images or collect some out-of-domain training images by themselves. Hence, to learn or tune a deep learning model with only very few samples is emerging as a critical problem to be solved.

In this few-shot compression scenario, most previous works [1, 12, 29] adopt filter-level pruning. However, it *cannot achieve a high acceleration ratio* on real-world computing devices (e.g., on GPUs). To make compressed models indeed run faster than the uncompressed models, lots of FLOPs (number of floating point operations) are required to be reduced by filter-level pruning. And without the whole training dataset, it is difficult to recover the compressed model's accuracy. Hence, previous few-shot compression methods often exhibit a poor latency (wall-clock timing) vs. accuracy tradeoff.

In this paper, we advocate that we need to focus on

KD [10]	FSKD [12]	CD [1]	MiR [29]	BP (blocks)
44.5	45.3	56.2	64.1	**66.5**

Table 1. Top-1 validation accuracy (%) on ImageNet-1k for different compression schemes. ResNet-34 was accelerated by reducing 16% latency with 50 training images. Previous methods prune filters with the 'normal' style. For the block-level pruning, we simply remove the first k blocks and finetune the pruned network by back propagation, i.e., 'BP (blocks)' in this table.

latency-accuracy rather than FLOPs-accuracy, and reveal that block-level pruning is fundamentally superior in the few-shot compression scenario. Compared to pruning filters, dropping blocks enjoys a higher acceleration ratio. Therefore it can keep more capacity from the original model and its accuracy is easier to be recovered by a tiny training set under the same latency when compared with filter pruning. Fig. 1 shows dropping blocks dominantly outperforms previous compression schemes for the latency-accuracy tradeoff. Table 1 further reports that *an embarrassingly simple dropping block baseline (i.e., finetune without any other processing) has already surpassed existing methods which use complicated techniques*. The baseline, 'BP (blocks)', simply removes the first few blocks and finetune the pruned network with the cross-entropy loss.

To further improve block pruning, we study the strategy for choosing which blocks to drop, especially when only scarce training samples are available. Several criteria [20, 30, 32] have been proposed for pruning blocks on the whole dataset. However, some [30, 32] require a large amount of data for choosing, whereas others [20] only evaluate the output difference before/after block removal. In this paper, we notice that although dropping some blocks significantly changes the feature maps, they are easily recovered by end-to-end finetuning even with a tiny training set. So simply measuring the difference between pruned/original networks is not valid. To deal with these problems, a new concept namely *recoverability* is proposed in this paper for better indicating blocks to drop. And we propose a method to compute it efficiently, with only a few training images. At last, our recoverability is surprisingly consistent with the accuracy of the finetuned network.

Finally, we propose PRACTISE, namely Practical network acceleration with tiny sets of images, to effectively accelerate a network with scarce data. PRACTISE significantly outperforms previous few-shot pruning methods. For 22.1% latency reduction, PRACTISE surpasses the previous state-of-the-art (SOTA) method on average by *7.0%* (percentage points, not relative improvement) Top-1 accuracy on ImageNet-1k. It is also robust and enjoys high generalization ability which can be used on synthesized/out-of-domain images. Our contributions are:

• We argue that the FLOPs-accuracy tradeoff is a misleading metric for few-shot compression, and advocate that the latency-accuracy tradeoff (which measures real runtime on devices) is more crucial in practice. For the first time, we find that in terms of latency vs. accuracy, block pruning is an embarrassingly simple but powerful method—dropping blocks with simple finetuning has already surpassed previous methods (cf. Table 1). Note that although dropping blocks is previously known, we are *the first to reveal its great potential in few-shot compression*, which is both a surprising and an important finding.

• To further boost the latency-accuracy performance of block pruning, we study the optimal strategy to drop blocks. A new concept *recoverability* is proposed to measure the difficulty of recovering each block, and in determining the priority to drop blocks. Then, we propose PRACTISE, an algorithm for accelerating networks with tiny sets of images.

• Extensive experiments demonstrate the extraordinary performance of our PRACTISE. With 22.1% latency reduction, PRACTISE outperforms previous SOTAs by 7.0% Top-1 accuracy on ImageNet-1k. In the extreme data-free scenario, PRACTISE also improves results by a significant margin. It is versatile and widely applicable for different network architectures, too.

2. Related Works

Filter-level pruning accelerates networks by removing filters in convolutional layers. Different criteria for choosing filters have been proposed [9, 11, 17, 20–22], along with different training strategies [4, 13, 18]. However, most these methods rely on the whole training data to train the network. When facing data privacy issues, these filter-level pruning methods suffer from poor latency-accuracy performance with tiny training sets [1, 12]. In this paper, we argue that the main drawback of filter pruning is its low acceleration ratio. It requires pruning lots of parameters and FLOPs to reduce latency. That results in a large capacity gap between the pruned and the original networks. So it is challenging to recover the pruned network accuracy on only a tiny training set. Instead, we advocate block-level pruning for few-shot compression. Dropping blocks enjoys a higher acceleration ratio. We claim that it is a superior way to accelerate networks with only tiny training sets.

Block-level pruning removes the whole block (*e.g.*, a residual block) in a network. Some works have been proposed for the whole training data case, but rarely studied in the few-shot scenario. DBP [30] proposes using linear probing to evaluate the accuracy of each block's features, and dropping blocks with low accuracy. ϵ-ResNet adds a sparsity-promoting function to discard the block if all responses of this block are less than a threshold ϵ. Both DBP and ϵ-ResNet require a large dataset for training and testing. CURL [20] uses a proxy dataset to evaluate the KL-

divergence change before/after block removal. However, it neglects the finetuning process. Actually, we care more about the accuracy of the pruned network *after finetuning*. But there are no existing criteria to measure it well.

In this paper, we propose a new concept named *recoverability* to evaluate if the network pruned by dropping blocks can recover the accuracy well. Our method for computing recoverability is efficient that only requires a few training samples. And it is effective to predict the accuracy of the finetuned network. Based on it, we propose PRACTISE , an algorithm for practical network acceleration with tiny sets. Our PRACTISE outperforms previous few-shot compression methods by a significant margin.

Data Limited Knowledge Distillation aims at training a student network by a pretrained teacher with limited original training data. Few-Shot Knowledge Distillation (FSKD) [12] inserts 1×1 conv. after the pruned conv. layer and trains each layer by making the pruned network's feature maps mimic the original network's. The layer-wisely training can obtain more supervised signals from the teacher, but easily results in error accumulation. CD [1] proposes cross distillation to reduce the layer-wisely accumulated errors. MiR [29] proposes a mimicking then replacing framework to optimize the pruned network holistically. For a more extreme case, not even a single original training sample is available, Data Free Knowledge Distillation (DFKD) was proposed in [31]. The core of DFKD is to synthesize alternatives of the original training data. Due to the promising results, more and more studies try to improve it, such as accelerating the synthesis process [5] and enhancing the performance by multi-teacher [14]. Pruning and quantization [3, 16, 33] are two main applications of DFKD.

However, most FSKD and DFKD methods adopt filter-level pruning to compress networks and result in poor latency-accuracy performance. We claim that dropping blocks is a more data-efficient acceleration scheme. Our PRACTISE outperforms previous FSKD works significantly. It is also robust to work on synthesized/out-of-domain images and improves the accuracy in the data-free scenario by a large margin.

3. The Proposed Method

First, we analyze the benefits of dropping blocks (*e.g.*, dropping residual blocks in ResNet). Compared with previous few-shot compression methods, dropping blocks enjoys a high acceleration ratio that achieves superior latency-accuracy performance with the tiny training set. Second, we propose a new concept named recoverability for choosing which blocks to drop. Different from previous criteria, the recoverability measures the hardness of finetuning a pruned network, which is closely and more directly related to the model's accuracy. The recoverability is also efficient to compute, which suits the few-shot scenario well.

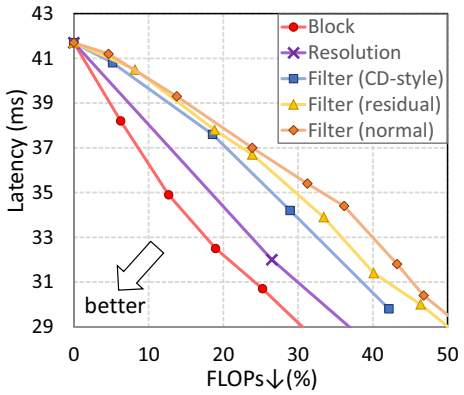

Figure 2. The relationship between latency and FLOPs reduction. Dropping blocks enjoys the highest acceleration ratio compared with other compression schemes.

Based on the recoverability, we propose PRACTISE, an algorithm for practical network acceleration with tiny training sets. Our PRACTISE does not require the label of the training set, and it is even able to work without using any image from the original training dataset.

3.1. The motivation to drop blocks

For accelerating neural networks in real-world applications, latency and accuracy are the two most important metrics. Lower latency means the model runs faster on devices. To accelerate the model with *tiny* training sets, many compression schemes have been proposed. Fig. 2 compares the acceleration ratios of different schemes. FSKD [12] prunes filters within residual blocks according to the L_1 norm (namely 'normal'). CD [1] proposes pruning conv. layers only in shallow layers, and keeps deeper conv. layers unchanged (namely 'CD-style'). MiR [29] trims the residual connection (namely 'residual'). All these pruning schemes suffer from inefficient acceleration ratios. As shown in Fig. 2, with about 30% FLOPs reduction, compressed models achieve only 16.1% latency reduction (41.7→35 ms). Simply resizing the input image's resolution (namely 'Resolution') achieves better acceleration. At last, dropping blocks outperforms all these methods. To achieve 35 ms latency, it only needs to reduce 12.7% FLOPs, significantly less than 30% in pruning filters.

Obviously, dropping blocks is more effective for model acceleration than pruning filters, but it is neglected in few-shot compression. One possible reason is that pruning filters has achieved extraordinary performance with the *whole* training dataset [4]. However, when only a tiny training set is available, finetuning the pruned model suffers from overfitting and unstable problems, especially for large FLOPs reductions [12, 29]. For the same latency reduction, dropping blocks keeps more parameters and capacity from the

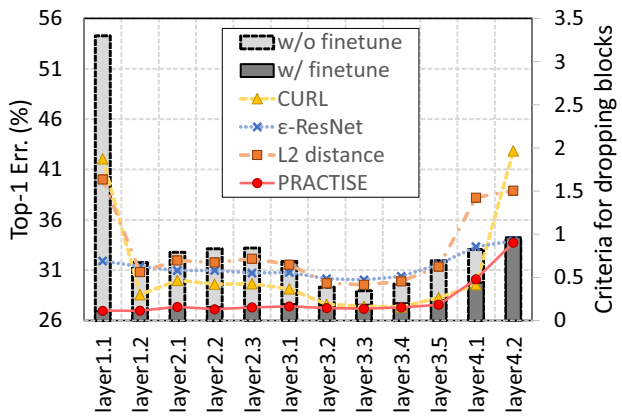

Figure 3. Illustration of the recoverability of each block and different criteria for dropping blocks. ResNet-34 has 12 blocks that can be dropped from 'layer1.1' to 'layer4.2'. We drop each block and evaluate the Top-1 error with/without finetuning, respectively. Scores of different criteria are computed for each block and presented in this figure, too. Note that the Top-1 error (%) is evaluated on the ImageNet-1k validation set (50000 images), while finetuning and evaluating criteria take only 500 training images. Our PRACTISE predicts the finetuned network's error almost perfectly.

original model. Therefore, it requires less data for finetuning and achieves a superior latency-accuracy tradeoff with the *tiny* training set (cf. Fig. 1). We have demonstrated that even a naive dropping blocks method has already outperformed most existing methods.

3.2. The recoverability of the pruned model

To further improve dropping blocks, we study how to choose blocks to drop. As analyzed in the related works section, existing criteria perform poorly in the few-shot scenario, as they either require a large training set [30, 32] or fail to predict the finetuned accuracy [20]. We argue that an effective metric should be consistent with the finetuned (*i.e.*, recovered) accuracy. To this end, we propose a new concept namely *recoverability*. Recoverability measures the ability of a pruned model to recover accuracy. As shown in Fig. 3, we will take the 'layer1.1' block as an example. Simply dropping this block results in a 54.3% Top-1 error. Both the KL-divergence ('CURL') and 'L2 distance' variations before/after block removal are large, which indicates the model's outputs are indeed changed dramatically. However, the accuracy can be recovered effectively by finetuning with a tiny training set. These existing criteria cannot reveal this trend, which however directly determines compression quality. To this end, we propose a method to evaluate the recoverability of each dropped block. As illustrated in Fig. 3, ours ('PRACTISE') enjoys a high consistency with the Top-1 error of the finetuned model.

Fig. 4 presents our method. Given the original model $\mathcal{M}_O$, the pruned model $\mathcal{M}_{P(\mathcal{B}_i)}$ is obtained by dropping a block $\mathcal{B}_i$. We insert adaptors in the positions connected to this block. These adaptors are conv. layers with kernel size 1×1 and placed before/after the raw conv. layers according to different positions. For blocks in front of the dropped block, adaptors are inserted after conv. layers. On the contrary, adaptors are inserted before conv. layers. Because convolutions are linear, all adaptors can be fused in the neighbor conv. layers while keeping the outputs unchanged. That means these adaptors will not be overhead to the pruned model. We use the tiny training set to optimize these adaptors to minimize the distance of original/pruned models' features. The recoverability is calculated as

$$\mathcal{R}(\mathcal{B}_i) = \min_{\alpha} \mathbb{E}_{x\sim p(x)} \|\mathcal{M}_O(x;\theta) - \mathcal{M}_{P(\mathcal{B}_i)}(x;\theta\backslash b_i,\alpha)\|_F^2 , \tag{1}$$

where θ means parameters in the original model, and $\backslash b_i$ means excluding the parameters in the dropped block $\mathcal{B}_i$, and α denotes parameters in adaptors.

Our method is effective to estimate the recoverability of each block, as Fig. 3 shows. In fact, our adaptors aim at eliminating the effect of dropping one block. Because nonlinear operations (*e.g.*, ReLU) exist in blocks, in theory the linear adaptors cannot eliminate the dropping effect perfectly. But a surprising observation is these adaptors can recover most accuracy loss (cf. Fig. 5). Hence it results in an excellent metric to predict the accuracy of the finetuned model. This phenomenon suggests that *finetuning the whole network on the tiny set mainly recovers the model's linearity part*, although CNNs are considered highly nonlinear models. Finally, our method is efficient for evaluation. Thanks to the limited computations and parameters in adaptors, calculating the recoverability by Eq. 1 requires only a few training samples and little training time.

Another factor for dropping blocks we should take care of is the latencies of different blocks are different. With the same recoverability, the block with a higher latency should have a higher priority to drop. To this end, we calculate the acceleration ratio of the block $\mathcal{B}_i$ by

$$\tau(\mathcal{B}_i) = \frac{lat_{\mathcal{M}_O} - lat_{\mathcal{M}_{P(\mathcal{B}_i)}}}{lat_{\mathcal{M}_O}} , \tag{2}$$

where lat denotes the latency. Finally, we define the pruning score for each block $\mathcal{B}_i$ as

$$s(\mathcal{B}_i) = \frac{\mathcal{R}(\mathcal{B}_i)}{\tau(\mathcal{B}_i)} . \tag{3}$$

Our pruning score considers both the recoverability and latency of each block. A lower score means the block has a higher priority to drop. We compute the pruning score for each block and drop the top k blocks with minimum scores.

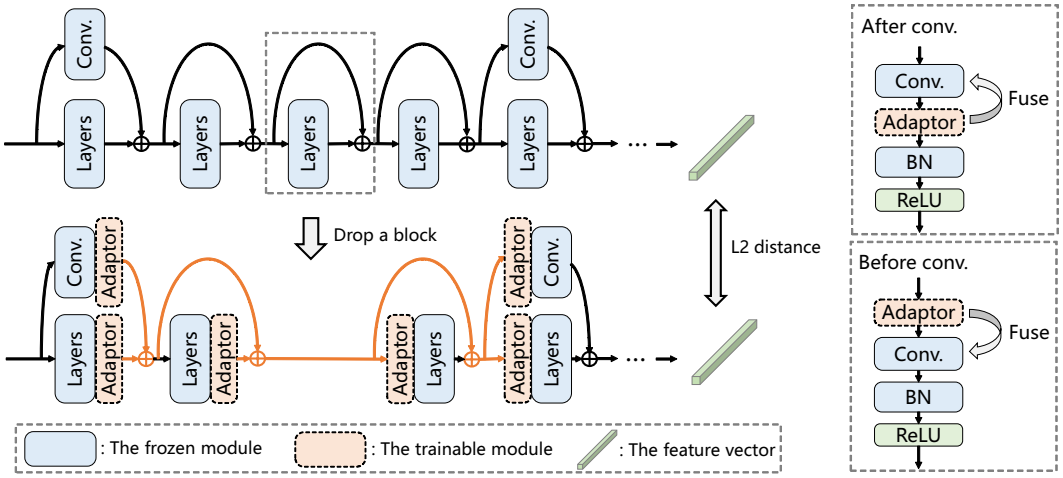

Figure 4. Illustration of our method for determining which block to drop. One block is dropped, and adaptors are inserted around the dropped position to alleviate the loss from dropping this block. Then, the L2 distance of pruned/original networks' features is computed as the *recoverability* of this dropped block. Finally, we drop several blocks that are easily recovered to obtain the pruned network.

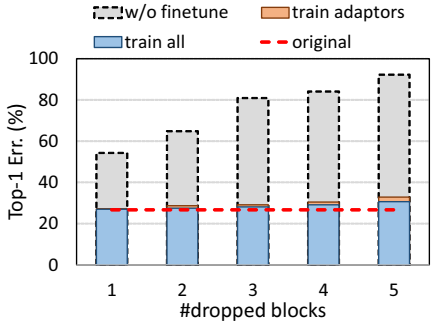

Figure 5. Comparison of training all and only adaptors. ResNet-34 was pruned by dropping different numbers of blocks with only 500 training samples. Top-1 error (%) was evaluated on the ImageNet-1k validation set. Note that the Top-1 error of the original model is 26.98%.

Algorithm 1: PRACTISE

Input: The original model $\mathcal{M}_O$, the number of dropped blocks k, the tiny training data $\mathcal{D}_\mathcal{T}$

Test the latency of $\mathcal{M}_O$;
for *each block* $\mathcal{B}_i$ **do**
  Drop $\mathcal{B}_i$ to obtain the pruned model $\mathcal{M}_{P(\mathcal{B}_i)}$;
  Test latency of $\mathcal{M}_{P(\mathcal{B}_i)}$ and find $\tau(\mathcal{B}_i)$ (Eq. 2);
  Insert adaptors;
  Compute $\mathcal{R}(\mathcal{B}_i)$ with $\mathcal{D}_\mathcal{T}$ (Eq. 1);
  Compute the score $s(\mathcal{B}_i)$ (Eq. 3);
  Add $\mathcal{B}_i$ back and remove all adaptors;
Choose the top k blocks with the minimum scores;
Drop these k blocks to obtain $\mathcal{M}_P$;
Finetune $\mathcal{M}_P$ with $\mathcal{D}_\mathcal{T}$ by minimizing $\mathcal{L}$ (Eq. 4);
return The pruned model $\mathcal{M}_P$

3.3. Recover the accuracy of the pruned model

Once the pruned model's structure is determined, the last problem is how to recover the accuracy. A naive method is using the tiny training set to finetune with the cross-entropy loss. However, the pruned model easily suffers from overfitting, as previous works pointed out [1, 12]. Knowledge distillation [10], especially feature-level distillation [24,28], can alleviate the overfitting problem and achieve superior accuracy. MiR [29] proposed to use the features before the global average pooling for the pruned model to mimic and achieved state-of-the-art performance with few training samples. We follow MiR and finetune the pruned model by minimizing

$$\mathcal{L} = \|\mathcal{M}_O(x;\theta_O) - \mathcal{M}_P(x;\theta_P)\|_F^2\,, \tag{4}$$

where θ_O denotes the frozen parameters of the original model and θ_P denotes the trainable parameters of the pruned model.

Overall, the whole algorithm of PRACTISE is presented in Alg. 1. Our PRACTISE enjoys *zero* extra hyperparameters. With feature mimicking, we accelerate models *without* using training labels.

Our PRACTISE can even work in data-free scenarios. One choice is treating the synthesized images from DFKD methods [31] as the training images. Most existing works [14, 31] adopt filter pruning, whereas our PRACTISE improves the latency-accuracy performance by a significant margin. That helps a lot in data-free scenarios. Another choice is collecting out-of-domain data. On the other hand, with a large amount of out-of-domain images, the accuracy

Method	Latency (ms)	50	100	500	1000
BP (filter)	35.1 (15.8%↓)	$39.0_{\pm 1.41}/68.9_{\pm 1.17}$	$41.0_{\pm 0.33}/70.5_{\pm 0.66}$	$51.8_{\pm 0.30}/78.1_{\pm 0.38}$	$57.8_{\pm 0.30}/81.5_{\pm 0.18}$
BP (block)	**34.9** (**16.3**%↓)	$\mathbf{66.5}_{\pm 0.81}/\mathbf{78.4}_{\pm 0.44}$	$\mathbf{66.8}_{\pm 0.23}/\mathbf{87.7}_{\pm 0.23}$	$\mathbf{68.6}_{\pm 0.18}/\mathbf{88.8}_{\pm 0.09}$	$\mathbf{69.8}_{\pm 0.12}/\mathbf{89.3}_{\pm 0.07}$
KD [10]	35.1 (15.8%↓)	$44.5_{\pm 1.20}/72.3_{\pm 0.87}$	$46.4_{\pm 0.34}/74.0_{\pm 0.58}$	$54.7_{\pm 0.26}/79.7_{\pm 0.19}$	$57.9_{\pm 0.21}/81.6_{\pm 0.12}$
FSKD [12]	35.1 (15.8%↓)	$45.3_{\pm 0.77}/71.5_{\pm 0.62}$	$51.2_{\pm 0.30}/76.8_{\pm 0.23}$	$57.6_{\pm 0.21}/81.6_{\pm 0.15}$	$59.4_{\pm 0.13}/82.7_{\pm 0.06}$
CD [1]	35.1 (15.8%↓)	$56.2_{\pm 0.37}/80.8_{\pm 0.31}$	$59.1_{\pm 0.22}/82.8_{\pm 0.11}$	$63.7_{\pm 0.18}/86.0_{\pm 0.05}$	$64.4_{\pm 0.03}/86.3_{\pm 0.07}$
MiR [29]	35.1 (15.8%↓)	$64.1_{\pm 0.10}/86.3_{\pm 0.11}$	$65.1_{\pm 0.19}/87.0_{\pm 0.11}$	$67.0_{\pm 0.09}/88.1_{\pm 0.07}$	$67.8_{\pm 0.06}/88.5_{\pm 0.02}$
PRACTISE	**34.9** (**16.3**%↓)	$\mathbf{70.3}_{\pm 0.16}/\mathbf{89.6}_{\pm 0.06}$	$\mathbf{71.5}_{\pm 0.74}/\mathbf{90.3}_{\pm 0.37}$	$\mathbf{72.5}_{\pm 0.04}/\mathbf{90.9}_{\pm 0.03}$	$\mathbf{72.5}_{\pm 0.05}/\mathbf{91.0}_{\pm 0.02}$

Table 2. Top-1/Top-5 validation accuracy (%) on ImageNet-1k for pruning ResNet-34. All models were accelerated by reducing about 16% latency with 50, 100, 500, and 1000 training samples (in the top row). Previous methods pruned filters within the residual block (*i.e.*, 'normal' in Fig. 2). The Top-1/Top-5 accuracy of the original ResNet-34 are 73.31%/91.42%.

Method	Latency (ms)	50	100	500	1000
BP (filter)	33.8 (18.9% ↓)	$24.2_{\pm 0.92}/52.7_{\pm 1.36}$	$27.6_{\pm 0.41}/56.7_{\pm 0.62}$	$42.9_{\pm 0.28}/70.5_{\pm 0.27}$	$51.2_{\pm 0.32}/76.5_{\pm 0.16}$
BP (block)	**32.5** (**22.1**% ↓)	$\mathbf{60.6}_{\pm 0.62}/\mathbf{83.5}_{\pm 0.42}$	$\mathbf{61.6}_{\pm 0.31}/\mathbf{84.3}_{\pm 0.36}$	$\mathbf{65.0}_{\pm 0.19}/\mathbf{86.5}_{\pm 0.20}$	$\mathbf{66.8}_{\pm 0.18}/\mathbf{87.5}_{\pm 0.13}$
KD [10]	33.8 (18.9% ↓)	$30.1_{\pm 0.69}/57.7_{\pm 1.10}$	$33.1_{\pm 0.43}/61.0_{\pm 0.53}$	$45.7_{\pm 0.26}/72.2_{\pm 0.25}$	$50.5_{\pm 0.29}/75.9_{\pm 0.23}$
FSKD [12]	33.8 (18.9% ↓)	$31.1_{\pm 0.90}/56.5_{\pm 1.10}$	$36.6_{\pm 0.44}/63.1_{\pm 0.46}$	$42.8_{\pm 0.49}/69.1_{\pm 0.58}$	$44.9_{\pm 0.20}/70.5_{\pm 0.29}$
MiR [29]	33.8 (18.9% ↓)	$59.9_{\pm 0.30}/83.2_{\pm 0.31}$	$62.1_{\pm 0.22}/84.8_{\pm 0.18}$	$65.4_{\pm 0.07}/87.0_{\pm 0.03}$	$66.6_{\pm 0.05}/87.7_{\pm 0.04}$
PRACTISE	**32.5** (**22.1**% ↓)	$\mathbf{68.0}_{\pm 1.36}/\mathbf{88.2}_{\pm 0.77}$	$\mathbf{70.4}_{\pm 0.42}/\mathbf{89.7}_{\pm 0.23}$	$\mathbf{71.8}_{\pm 0.07}/\mathbf{90.5}_{\pm 0.02}$	$\mathbf{71.9}_{\pm 0.05}/\mathbf{90.6}_{\pm 0.04}$

Table 3. Top-1/Top-5 validation accuracy (%) on ImageNet-1k for pruning ResNet-34. Our model was accelerated by reducing 22.1% latency. Previous methods pruned filters both inside and outside the residual connection (*i.e.*, 'residual' in Fig. 2). The Top-1/Top-5 accuracy of the original ResNet-34 are 73.31%/91.42%.

of the pruned network is even close to that of using original training images. That demonstrates the high generalization ability of PRACTISE.

4. Experimental Results

In this section, we evaluate the performance of PRACTISE. Following previous works [1, 12, 29], ResNet-34 [8] and MobileNetV2 [26] will be pruned on tiny training sets of ImageNet-1k [25]. Then, to test the generalization ability of PRACTISE, we will prune ResNet-50 with synthesized images and out-of-domain data, respectively. Finally, ablation studies are conducted for further analyzing PRACTISE.

Implementation details. As presented in Alg. 1, PRACTISE requires computing the latency and recoverability of each pruned model $\mathcal{M}_{P(\mathcal{B}_i)}$, then finetuning the pruned model $\mathcal{M}_P$. In this paper, we only consider dropping the block with the same input and output dimensionality. For the latency, we tested the model with the $64 \times 3 \times 224 \times 224$ input by 500 times and take the mean as the latency number. Note that all latency numbers in this paper were tested on the same computer with an NVIDIA TITAN Xp GPU. To compute the recoverability in Eq. 1, we used SGD with batch size 64 to optimize adaptors by 1000 iterations. The initial learning rate was 0.02 and decreased by a factor of 10 per 40% iterations. Note that if the size of training data $\mathcal{D}_\mathcal{T}$ is less than 64, the batch size will equal this number. After optimizing adaptors, $\mathcal{R}(\mathcal{B}_i)$ was computed by Eq. 1 with the training set $\mathcal{D}_\mathcal{T}$. For the final finetuning, all parameters in the pruned network were updated by SGD with minimizing $\mathcal{L}$ in Eq. 4. Finetuning ran 2000 iterations by default. The settings of batch size and learning rate schedules were the same as those of optimizing adaptors. For a fair comparison, we used the data argumentation strategy supplied by PyTorch official examples, which is the same as that of previous works [1, 12, 29]. All experiments were conducted with PyTorch [23].

To compare with other few-shot pruning methods, we are mainly concerned about the latency-accuracy tradeoff of the pruned model. We tested the latency of networks pruned by previous methods and directly cite their accuracy number. The results of PRACTISE were run by five times with different sampled tiny training sets. We report the mean accuracy along with standard deviation.

4.1. Different amounts of training data

We compare our PRACTISE with the state-of-the-art few-shot pruning methods. Following the previous setting, we prune ResNet-34 with different amounts of training images on ImageNet-1k. Table 2 summarizes results. All previous methods pruned filters within residual blocks (cf. 'normal' in Fig. 2), whereas our dropping blocks achieves better results. First, we just removed the first few blocks to reach the latency goal and then simply finetuned the network with

Latency (ms)	BP (filter)	KD [10]	FSKD [12]	CD [1]	MiR [29]	PRACTISE
32.5 (22.1%↓)	$47.54_{\pm 0.41}$	$49.34_{\pm 0.25}$	$33.19_{\pm 0.60}$	$59.65_{\pm 0.12}$	$68.14_{\pm 0.04}$	$\mathbf{71.75}_{\pm 0.07}$
34.9 (16.3%↓)	$58.94_{\pm 0.36}$	$61.01_{\pm 0.24}$	$62.56_{\pm 0.13}$	$68.17_{\pm 0.07}$	$70.53_{\pm 0.10}$	$\mathbf{72.52}_{\pm 0.04}$
38.3 (8.2%↓)	$65.02_{\pm 0.30}$	$67.22_{\pm 0.18}$	$69.59_{\pm 0.09}$	$71.12_{\pm 0.06}$	$71.95_{\pm 0.07}$	$\mathbf{73.04}_{\pm 0.06}$

Table 4. Top-1 validation accuracy (%) on ImageNet-1k for pruning ResNet-34. The model was pruned at three different compression ratios with 500 training samples. Previous methods prune filters by CD-style, while we drop blocks. The Top-1 accuracy and the latency of the original ResNet-34 are 73.31% and 41.7 ms, respectively.

the cross-entropy loss. This simple baseline for dropping blocks, 'BP (block)', has already outperformed previous methods. Note that this is our first contribution that revealing the advantage of dropping blocks in the few-shot scenario. PRACTISE further improves results. With similar latency reduction, it dramatically outperforms previous SOTA by an average of 5.7% Top-1 accuracy on ImageNet-1k. Table 3 compares these methods with a larger latency reduction. PRACTISE surpasses MiR by a significant margin again, on average 7.0% Top-1 accuracy. Note that our model even is faster than previous ones by 1.3 ms. Both Table 2 and 3 demonstrate that dropping blocks is a superior manner for accelerating networks with tiny sets.

4.2. Different acceleration ratios

Table 4 compares PRACTISE with previous methods for different acceleration ratios. For 8.2% latency reduction, outperforms MiR by 1.1% Top-1 accuracy. In further reducing latency by 22.1%, PRACTISE surpasses MiR by 3.6% Top-1 accuracy. That indicates PRACTISE enjoys higher accuracy than others when the acceleration ratio becomes larger. Fig. 1 presents curves for the latency-accuracy tradeoffs of different methods. Our PRACTISE outperforms previous methods dominantly and makes a new milestone in the field of few-shot compression.

4.3. The data-latency-accuracy tradeoff

Previous experiments have demonstrated the advantage of PRACTISE . Next, for a better understanding of the data-latency-accuracy tradeoff in the few-shot compression scenario, we pruned ResNet-34 by PRACTISE with different latency reductions and amounts of training images. Fig. 6 presents the results. With less latency reduction, the accuracies for different amounts of training data are comparable. But for a large latency reduction, it is challenging to recover the accuracy by only a tiny training set. And the accuracy gap becomes large w.r.t. different amounts of training data.

4.4. Results on MobileNetV2

MobileNetV2 [26] is a lightweight model and is popularly applied on mobile devices. It has 10 blocks which can be dropped by our PRACTISE. Table 5 summarizes results. PRACTISE outperforms previous methods by a significant margin. Compared with MiR, PRACTISE obtains pruned models with both lower latency and higher accuracy.

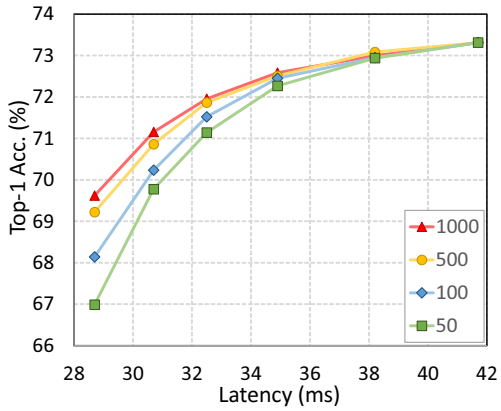

Figure 6. Illustration of the data-latency-accuracy tradeoff. ResNet-34 was pruned by PRACTISE on ImageNet-1k. Numbers in the legend mean different amounts of training images.

Method	Latency (ms)	Top-1/Top-5
Original	37.6	71.9/90.3
BP (filter)	31.5 (16.2% ↓)	$45.0_{\pm 0.34}/71.8_{\pm 0.38}$
KD [10]	31.5 (16.2% ↓)	$48.4_{\pm 0.34}/73.9_{\pm 0.32}$
MiR [29]	31.5 (16.2% ↓)	$67.6_{\pm 0.05}/87.9_{\pm 0.04}$
PRACTISE	**30.4** (**19.1**% ↓)	$\mathbf{69.3}_{\pm 0.05}/\mathbf{88.9}_{\pm 0.05}$
BP (filter)	34.1 (9.3% ↓)	$55.5_{\pm 0.16}/80.3_{\pm 0.26}$
KD [10]	34.1 (9.3% ↓)	$59.1_{\pm 0.17}/82.5_{\pm 0.15}$
MiR [29]	34.1 (9.3% ↓)	$69.7_{\pm 0.04}/89.2_{\pm 0.03}$
PRACTISE	**31.9** (**15.2**% ↓)	$\mathbf{70.3}_{\pm 0.03}/\mathbf{89.5}_{\pm 0.03}$

Table 5. Comparison of PRACTISE and few-shot pruning methods on ImageNet-1k. MobileNetV2 was pruned with 500 samples.

4.5. Train with synthesized/out-of-domain images

In some extreme scenarios, not even a single original training sample is to be provided. Zero-shot pruning is required. Because PRACTISE does not need ground-truth labels, it is able to accelerate networks with the images synthesized by data-free knowledge distillation methods. Table 6 summarizes results. Note that most zero-shot pruning methods adopt filter pruning, which are inferior to pruning

Network	Method	Pruning	Latency	Top-1
ResNet-50	Original		83.8	76.1
	DI [31]	filter	-	72.0
	MixMix [14]	filter	-	69.8
	ADI [31]	filter	-	73.3
	ADI* [31]	filter	79.9 (4.7%↓)	73.5
	PRACTISE	block	**66.2** (**21.0**%↓)	**74.8**
MobileNetV2	Original		37.6	71.9
	DI [31]	filter	-	15.3
	MixMix [14]	filter	-	42.5
	ADI* [31]	filter	30.8 (18.1%↓)	62.8
	PRACTISE	block	**30.4** (**19.1**%↓)	**68.0**

Table 6. Comparison of PRACTISE and data-free methods on ImageNet-1k. Latency (ms) and Top-1 validation accuracy (%) on ImageNet-1k are reported. * denotes the method reimplemented by ourselves.

Dateset	50	500	1000	5000	All
ImageNet [25]	74.22	74.58	74.58	75.14	75.24
ADI [31]	69.85	72.68	73.01	74.40	74.79
CUB [27]	72.49	73.71	73.94	74.86	74.92
Place365 [34]	**72.80**	**74.10**	**74.18**	**75.05**	**75.21**

Table 7. Top-1 validation accuracy (%) on ImageNet-1k for different out-of-domain training datasets. ResNet-50 was accelerated by reducing 21% latency with PRACTISE.

blocks as we have shown. We adopt synthesized images produced by ADI [31] as the training set. For both ResNet-50 and MobileNetV2, our PRACTISE achieves higher Top-1 accuracy with more latency reductions. And we advocate that we should adopt dropping blocks for DFKD in the future to accelerate networks more effectively.

Another choice of zero-shot pruning is collecting out-of-domain training images. Our PRACTISE is also robust to work with these data. Table 7 presents results. The original ResNet-50 was trained on ImageNet-1k, and we pruned it on other datasets by PRACTISE. ADI consists of images synthesized by DeepInversion [31]. CUB [27] contains images of birds with 200 categories. Place365 [34] consists of scene pictures. Our PRACTISE enjoys a high generalization ability to work with all these datasets. Another benefit of the out-of-domain data is the unlimited number of images. We notice the accuracy is boosted by using more training samples and even close to that of using original training data.

4.6. Different criteria for dropping blocks

Finally, we compare PRACTISE with other criteria for dropping blocks. Fig. 3 shows results for removing only one block. Obviously, PRACTISE is better than others, and is very consistent with the finetuned models' accuracies. Most existing methods mainly measure the gap between the original network and the pruned network without finetuning. They neglect the recoverability of each dropped block, hence resulting in an inferior strategy for dropping blocks. Our PRACTISE pays attention to the recoverability and the acceleration ratio of each block. Therefore the pruned network enjoys higher accuracy and lower latency.

#Dropped blocks	1	2	3	4	5
CURL [20]	72.33	71.08	69.14	65.48	64.97
ϵ-ResNet [32]	72.51	71.20	69.00	67.90	64.75
L2 distance	72.51	71.20	69.00	68.61	64.93
PRACTISE	**73.02**	**72.52**	**71.92**	**70.86**	**69.35**

Table 8. Top-1 validation accuracy (%) on ImageNet-1k for pruning ResNet-34 with 500 images. We compare different criteria for dropping different numbers of blocks.

Table 8 compares different criteria for dropping more blocks. As the number of dropped blocks increases, our PRACTISE outperforms others by even larger margins. To sum up, our PRACTISE is able to find inefficient blocks to drop, compared with other methods.

5. Conclusions and Future Works

This paper aims at accelerating networks with tiny training sets. For the first time we revealed that dropping blocks is more effective than previous filter-level pruning in this scenario. We believe this finding makes significant progress in the few-shot model compression. To determine which blocks to drop, we proposed a new concept namely *recoverability* to measure the difficulty of recovering the pruned network's accuracy with few samples. Compared with previous pruning criteria, our recoverability is more related to the model's accuracy after finetuning. Our method for computing recoverability reveals that the end-to-end finetuning with a tiny set mainly recovers the model's linear ability. Finally, PRACTISE was proposed. It enjoys high latency-accuracy performance and is robust to deal with synthesized/out-of-domain images. Extensive experiments demonstrated that PRACTISE outperforms previous methods by a significant margin (on average 7% Top-1 accuracy on ImageNet-1k for 22% latency reduction).

PRACTISE has limitations such as it is confined to recognition, which leads to future explorations. It is promising to extend PRACTISE for other models (*e.g.*, Transformer) or other vision tasks (*e.g.*, object detection and segmentation). For network acceleration, how to compute the recoverability of other compression schemes is also an interesting problem. Recently, finetuning and accelerating a large pretrained network on downstream tasks are emerging as critical needs. Applying PRACTISE for tuning a pretrained model on downstream tasks is also promising.

References

[1] Haoli Bai, Jiaxiang Wu, Irwin King, and Michael Lyu. Few shot network compression via cross distillation. In *AAAI*, volume 04, pages 3203–3210, 2020. 1, 2, 3, 5, 6, 7

[2] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *NeurIPS 31*, pages 5151–5159, 2018. 1

[3] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. ZeroQ: A novel zero shot quantization framework. In *CVPR*, pages 13169–13178, 2020. 3

[4] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. ResRep: Lossless cnn pruning via decoupling remembering and forgetting. In *ICCV*, pages 4510–4520, 2021. 2, 3

[5] Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. Up to 100x faster data-free knowledge distillation. In *AAAI*, volume 36, pages 6597–6604, 2022. 3

[6] Jianbo Guo, Yuxi Li, Weiyao Lin, Yurong Chen, and Jianguo Li. Network decoupling: From regular to depthwise separable convolutions. In *BMVC*, page 248, 2018. 1

[7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, pages 1–14, 2016. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6

[9] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, pages 4340–4349, 2019. 2

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 5, 6, 7

[11] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 2

[12] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *CVPR*, pages 14639–14647, 2020. 1, 2, 3, 5, 6, 7

[13] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. In *CVPR*, pages 191–201, 2022. 2

[14] Yuhang Li, Feng Zhu, Ruihao Gong, Mingzhu Shen, Xin Dong, Fengwei Yu, Shaoqing Lu, and Shi Gu. MixMix: All you need for data-free compression are feature and data mixing. In *ICCV*, pages 4410–4419, 2021. 3, 5, 8

[15] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic CNN compression via low-rank decomposition with knowledge transfer. *IEEE TPAMI*, 41(12):2889–2905, 2018. 1

[16] Yuang Liu, Wei Zhang, and Jun Wang. Zero-shot adversarial quantization. In *CVPR*, pages 1512–1521, 2021. 3

[17] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2736–2744, 2017. 1, 2

[18] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, pages 1–21, 2018. 2

[19] Zechun Liu, Xiangyu Zhang, Zhiqiang Shen, Yichen Wei, Kwang-Ting Cheng, and Jian Sun. Joint multi-dimension pruning via numerical gradient update. *IEEE TIP*, 30:8034–8045, 2021. 1

[20] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *CVPR*, pages 1458–1467, 2020. 2, 4, 8

[21] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017. 1, 2

[22] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, pages 11264–11272, 2019. 2

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS 32*, pages 8026–8037, 2019. 6

[24] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *ICLR*, pages 1–13, 2015. 5

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6, 8

[26] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 6, 7

[27] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 8

[28] Guo-Hua Wang, Yifan Ge, and Jianxin Wu. Distilling knowledge by mimicking features. *IEEE TPAMI*, 44(11):8183–8195, 2022. 5

[29] Huanyu Wang, Junjie Liu, Xin Ma, Yang Yong, Zhenhua Chai, and Jianxin Wu. Compressing models with few samples: Mimicking then replacing. In *CVPR*, pages 701–710, 2022. 1, 2, 3, 5, 6, 7

[30] Wenxiao Wang, Shuai Zhao, Minghao Chen, Jinming Hu, Deng Cai, and Haifeng Liu. DBP: Discrimination based block-level pruning for deep model acceleration. *arXiv preprint arXiv:1912.10178*, 2019. 2, 4

[31] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *CVPR*, pages 8715–8724, 2020. 3, 5, 8

[32] Xin Yu, Zhiding Yu, and Srikumar Ramalingam. Learning strict identity mappings in deep residual networks. In *CVPR*, pages 4432–4440, 2018. 2, 4, 8

[33] Xiangguo Zhang, Haotong Qin, Yifu Ding, Ruihao Gong, Qinghua Yan, Renshuai Tao, Yuhang Li, Fengwei Yu, and Xianglong Liu. Diversifying sample generation for accurate data-free quantization. In *CVPR*, pages 15658–15667, 2021. 3

[34] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016. 8