Pruning from Scratch

Yulong Wang,[1] **Xiaolu Zhang,**[2] **Lingxi Xie,**[3] **Jun Zhou,**[2] **Hang Su,**[1] **Bo Zhang,**[1] **Xiaolin Hu**[1*]

[1]Tsinghua University, [2]Ant Financial, [3]Huawei Noah's Ark Lab

wang-yl15@mails.tsinghua.edu.cn, {yueyin.zxl, jun.zhoujun}@antfin.com, 198808xc@gmail.com,
{suhangss, dcszb, xlhu}@mail.tsinghua.edu.cn

Abstract

Network pruning is an important research field aiming at reducing computational costs of neural networks. Conventional approaches follow a fixed paradigm which first trains a large and redundant network, and then determines which units (*e.g.*, channels) are less important and thus can be removed. In this work, we find that pre-training an over-parameterized model is not necessary for obtaining the target pruned structure. In fact, a fully-trained over-parameterized model will reduce the search space for the pruned structure. We empirically show that more diverse pruned structures can be directly pruned from randomly initialized weights, including potential models with better performance. Therefore, we propose a novel network pruning pipeline which allows *pruning from scratch* with little training overhead. In the experiments for compressing classification models on CIFAR10 and ImageNet datasets, our approach not only greatly reduces the pre-training burden of traditional pruning methods, but also achieves similar or even higher accuracy under the same computation budgets. Our results facilitate the community to rethink the effectiveness of existing techniques used for network pruning.

Introduction

As deep neural networks are widely deployed in mobile devices, there has been an increasing demand for reducing model size and run-time latency. Network pruning (Han, Mao, and Dally 2016; He, Zhang, and Sun 2017; Liu et al. 2017) techniques are proposed to achieve model compression and inference acceleration by removing redundant structures and parameters. In addition to the early non-structured pruning methods (LeCun, Denker, and Solla 1990; Han, Mao, and Dally 2016), the structured pruning method represented by channel pruning (Li et al. 2016; Luo, Wu, and Lin 2017; He, Zhang, and Sun 2017; Liu et al. 2017) has been widely adopted in recent years because of its easy deployment on general-purpose GPUs. The traditional network pruning methods adopt a three-stage pipeline, namely pre-training, pruning, and fine-tuning (Liu et al. 2019), as shown in Figure 1(a). The pre-training and
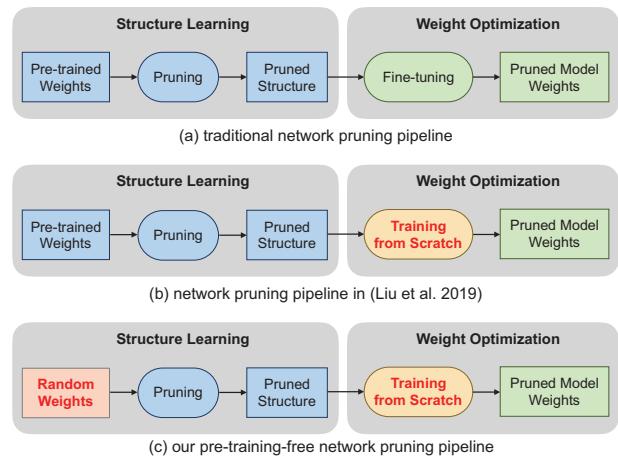
Figure 1: Network pruning pipelines. (a) Traditional network pruning needs pre-trained weights and certain pruning strategy for pruned structure learning, and fine-tuning on full model weights. (b) Recent work (Liu et al. 2019) shows that the pruned model can be trained from scratch without fine-tuning to reach comparable performance. However, the pruned model structure still needs to be obtained by traditional pruning strategies. (c) We empirically show that the pruned model structure can be directly learned from randomly initialized weights without the loss of performance.

pruning steps can also be performed alternately with multiple cycles (He et al. 2018a). However, recent study (Liu et al. 2019) has shown that the pruned model can be trained from scratch to achieve a comparable prediction performance without the need to fine-tune the inherited weights from the full model (as shown in Figure 1(b)). This observation implies that the pruned architecture is more important for the pruned model performance. Specifically, in the channel pruning methods, more attention should be paid to searching the channel number configurations of each layer.

Although it has been confirmed that the *weights* of the pruned model do not need to be fine-tuned from the pretrained weights, the *structure* of the pruned model still needs

to be learned and extracted from a well-trained model according to different criteria. This step usually involves cumbersome and time-consuming weights optimization process. Then we naturally ask a question: Is it necessary for learning the pruned model structure from pre-trained weights?

In this paper, we explored this question through extensive experiments and found that the answer is quite surprising: an effective pruned structure does not have to be learned from pre-trained weights. We empirically show that the pruned structures discovered from pre-trained weights tend to be homogeneous, which limits the possibility of searching for better structure. In fact, more diverse and effective pruned structures can be discovered by directly pruning from randomly initialized weights, including potential models with better performance.

Based on the above observations, we propose a novel network pruning pipeline that a pruned network structure can be directly learned from the randomly initialized weights (as shown in Figure 1(c)). Specifically, we utilize a similar technique in Network Slimming (Liu et al. 2017) to learn the channel importance by associating scalar gate values with each layer. The channel importance is optimized to improve the model performance under the sparsity regularization. What is different from previous works is that we do not update the random weights during this process. After finishing the learning of channel importance, we utilize a simple binary search strategy to determine the channel number configurations of the pruned model given resource constraints (*e.g.*, FLOPS). Since we do not need to update the model weights during optimization, we can discover the pruned structure at an extremely fast speed. Extensive experiments on CIFAR10 (Krizhevsky and others 2009) and ImageNet (Russakovsky et al. 2015) show that our method yields at least $10\times$ and $100\times$ searching speedup while achieving comparable or even better model accuracy than traditional pruning methods using complicated strategies. Our method can free researchers from the time-consuming training process and provide competitive pruning results in future work.

Related Work

Network pruning techniques aim to achieve the inference acceleration of deep neural networks by removing the redundant parameters and structures in the model. Early works (LeCun, Denker, and Solla 1990; Han, Mao, and Dally 2016; Han et al. 2015) proposed to remove individual weight values, resulting in non-structured sparsity in the network. The runtime acceleration cannot be easily achieved on a general-purpose GPU, otherwise with a custom inference engine (Han et al. 2016). Recent works focus more on the development of structured model pruning (Li et al. 2016; He, Zhang, and Sun 2017; Liu et al. 2017), especially pruning weight channels. ℓ_1-norm based criterion (Li et al. 2016) prunes model according to the ℓ_1-norm of weight channels. Channel Pruning (He, Zhang, and Sun 2017) learns to obtain sparse weights by minimizing local layer output reconstruction error. Network Slimming (Liu et al. 2017) uses LASSO regularization to learn the importance of all channels and prunes the model based on a global threshold. Au-

tomatic Model Compression (AMC) (He et al. 2018b) explores the pruning strategy by automatically learning the compression ratio of each layer through reinforcement learning (RL). Pruned models often require further fine-tuning to achieve higher prediction performance. However, recent works (Liu et al. 2019; Frankle and Carbin 2019) have challenged this paradigm and show that the compressed model can be trained from scratch to achieve comparable performance without relying on the fine-tuning process.

Recently, neural architecture search (NAS) provides another perspective on the discovery of the compressed model structure. Recent works (Liu, Simonyan, and Yang 2018; Cai, Zhu, and Han 2019) follow the top-down pruning process by trimming out a small network from a supernet. The one-shot architecture search methods (Brock et al. 2018; Bender et al. 2018) further develop this idea and conduct architecture search only once after learning the importance of internal cell connections. However, these methods require a large amount of training time to search for an efficient structure.

Rethinking Pruning with Pre-Training

Network pruning aims to reduce the redundant parameters or structures in an over-parameterized model to obtain an efficient pruned network. Representative network pruning methods (Liu et al. 2017; Gordon et al. 2018) utilize channel importance to evaluate whether a specific weight channel should be reserved. Specifically, given a pre-trained model, a set of channel gates are associated with each layer to learn the channel importance. The channel importance values are optimized with ℓ_1-norm based sparsity regularization. Then with the learned channel importance values, a global threshold is set to determine which channels are preserved given a predefined resource constraint. The final pruned model weights can either be fine-tuned from the original full model weights or re-trained from scratch. The overall pipeline is depicted in Figure 1(a) and (b).

In what follows, we show that in the common pipeline of network pruning, the role of pre-training is quite different from what we used to think. Based on this observation, we present a new pipeline which allows pruning networks from scratch, *i.e.*, randomly initialized weights, in the next section.

Effects of Pre-Training on Pruning

The traditional pruning pipeline seems to default to a network that must be fully trained before it can be used for pruning. Here we will empirically explore the effect of the pre-trained weights on the final pruned structure. Specifically, we save the checkpoints after different training epochs when we train the baseline network. Then we utilize the weights of different checkpoints as the network initialization weights, and learn the channel importance of each layer by adopting the pipeline described above. We want to explore whether the pre-trained weights at different training stages have a crucial impact on the final pruned structure learning.

Pruned Structure Similarity First, we compare the structure similarity between different pruned models. For each
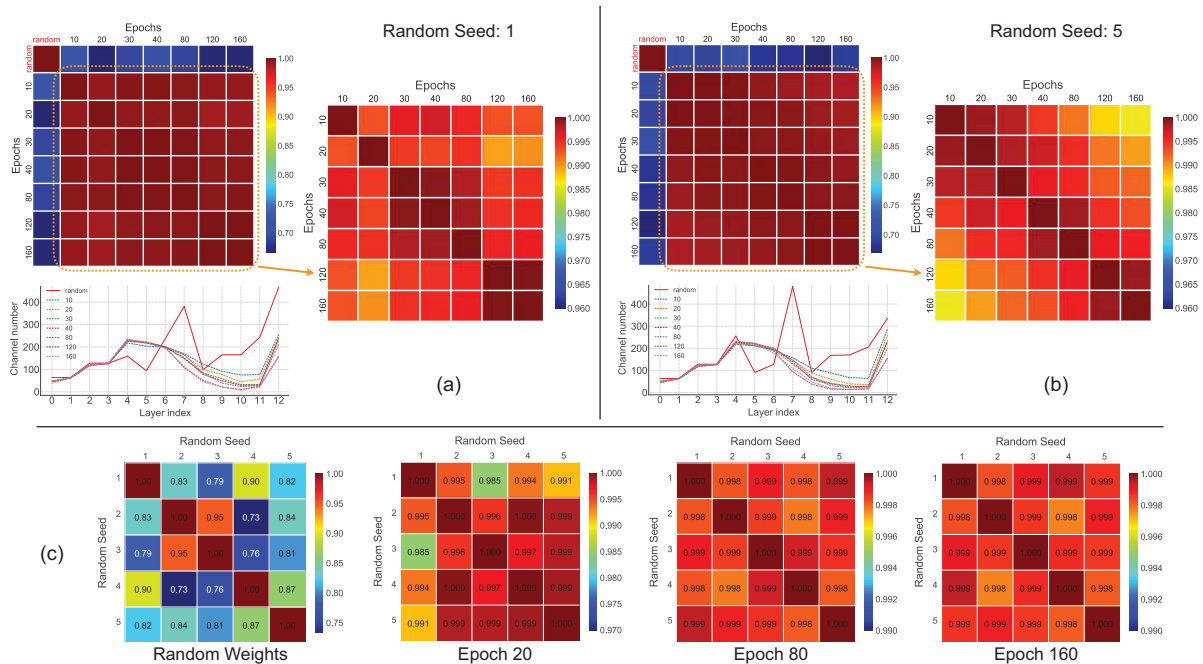
Figure 2: Exploring the effect of pre-trained weights on the pruned structures. All the pruned models are required to reduce 50% FLOPS of the original VGG16 on CIFAR10 dataset. (a) (Top-left) We display the correlation coefficient matrix of the pruned models directly learned from randomly initialized weights ("**random**") and other pruned models based on different checkpoints during pre-training ("**epochs**"). (Right) We display the correlation coefficient matrix of pruned structures from pre-trained weights on a finer scale. (Bottom-left) We show the channel numbers of each layer of different pruned structures. Red line denotes the structure from random weights. (b) Similar results from the experiment with a different random seed. (c) We display correlation coefficient matrices of all the pruned structures from five different random seeds. We mark the names of initialized weights used to get pruned structures below.

pruned model, we calculate the pruning ratio of each layer, *i.e.*, the number of remaining channels divided by the number of original channels. The vector formed by concatenating the pruning ratios of all layers is then considered to be the feature representation of the pruned structure. Then we calculate the correlation coefficient between each of the two pruned model features as the similarity of their structures. In order to ensure the validity, we randomly selected five sets of random seeds for experiments on CIFAR10 dataset with VGG16 (Simonyan and Zisserman 2014) network. We include more visualization results of ResNet20 and ResNet56 (He et al. 2016) in the supplementary material.

Figure 2 shows the correlation coefficient matrices for all pruned models. From this figure, we can observe three phenomena. First, the pruned structures learned from random weights are not similar to all the network structures obtained from pre-trained weights (see top-left figures in Figure 2(a)(b)). Second, the pruned model structures learned directly from random weights are more diverse with various correlation coefficients. Also, after only ten epochs of weights update in the pre-training stage, the resulting pruned network structures become almost homogeneous. (see Figure 2(c)). Third, the pruned structures based on the checkpoints from near epochs are more similar with high correla-

tion coefficients in the same experiment run (see right figures in Figure 2(a)(b)).

The structure similarity results indicate that the potential pruned structure space is progressively reduced during the weights update in the pre-training phase, which may limit the potential performance accordingly. On the other hand, the randomly initialized weights allow the pruning algorithm to explore more diverse pruned structures.

Performance of Pruned Structures We further train each pruned structure from scratch to compare the final accuracy. Table 1 summarizes the prediction accuracy of all pruned structures on the CIFAR10 test set. It can be observed that the pruned models obtained from the random weights can always achieve comparable performance with the pruned structures based on the pre-trained weights. Also, in some cases (such as ResNet20), the pruned structures directly learned from random weights achieves even higher prediction accuracy. These results demonstrate that not only the pruned structures learned directly from random weights are more diverse, but also that these structures are valid and can be trained to reach competitive performance.

The pruned model accuracy results also demonstrate that the pruned structures based on pre-trained weights have little advantages in the final prediction performance under mild

Table 1: Pruned model accuracy (%) on the CIFAR10 test set. All models are trained from scratch based on the training scheme in (Liu et al. 2019). We report the average accuracy across five runs. "Rand" stands for pruned structures from random weights. "RN" stands for ResNet.

Model	Rand	Pre-training Epochs						
		10	20	30	40	80	120	160
VGG16	93.68	93.60	93.83	93.71	93.69	93.64	93.69	93.58
RN20	90.57	90.48	90.50	90.49	90.33	90.42	90.34	90.23
RN56	92.95	92.96	92.90	92.98	93.04	93.03	92.99	93.05

pruning ratio scenario. Considering that the pre-training phase often requires a cumbersome and time-consuming computation process, we think that network pruning can directly start from randomly initialized weights.

Our Solution: Pruning from Scratch

Based on the above analysis, we propose a new pipeline named *pruning from scratch*. Different from existing ones, it enables researchers to obtain pruned structure directly from randomly initialized weights.

Specifically, we denote a deep neural network as $f(x; \boldsymbol{W}, \alpha)$, where x is an input sample, $\boldsymbol{W}$ is all trainable parameters, and α is the model structure. In general, α includes operator types, data flow topology, and layer hyper-parameters as modeled in NAS research. In the network pruning, we mainly focus on the micro-level layer settings, especially the channel number of each layer in the channel pruning strategies.

To efficiently learn the channel importance for each layer, a set of scalar gate values $\boldsymbol{\lambda}_j$ are associated with the j-th layer along the channel dimension. The gate values are multiplied onto the layer's output to perform channel-wise modulation. Therefore, a near-zero gate value will suppress the corresponding channel output, resulting in a pruning effect. We denote the scalar gate values across all the K layers as $\Lambda = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \cdots, \boldsymbol{\lambda}_K\}$. The optimization objective for Λ is

$$\min_{\Lambda} \quad \sum_i^N \mathcal{L}(f(x_i; \boldsymbol{W}, \Lambda), y_i) + \gamma \sum_j^K |\boldsymbol{\lambda}_j|_1$$
$$s.t. \quad \boldsymbol{0} \preceq \boldsymbol{\lambda}_j \preceq \boldsymbol{1}, \quad \forall j = 1, 2, \cdots, K, \quad (1)$$

where y_i is the corresponding label, $\mathcal{L}$ is cross-entropy loss function, γ is a balance factor. Here, the difference from previous works is two-fold. First, we do not update the weights during channel importance learning; Second, we use randomly initialized weights without relying on pre-training.

Following the same approach in Network Slimming, we adopt sub-gradient descent to optimize Λ for the non-smooth regularization term. However, the naive ℓ_1-norm will encourage the gates to be zeroes unconstrainedly, which does not lead to a good pruned structure. Different from the original formulation in Network Slimming, we use the element-wise mean of all the gates to approximate the overall sparsity ratio, and use the square norm to push the sparsity to a pre-defined ratio r (Luo and Wu 2018). Therefore, given a target

sparsity ratio r, the regularization term is

$$\Omega(\Lambda) = (\frac{\sum_j |\boldsymbol{\lambda}_j|_1}{\sum_j C_j} - r)^2, \quad (2)$$

where C_j is the channel number of the j-th layer. Empirically, we find this improvement can obtain more reasonable pruned structure. During the optimization, there can be multiple possible gates for pruning. We select the final gates whose sparsity is below the target ratio r while achieving the maximum validation accuracy.

After obtaining a set of optimized gate values $\Lambda^* = \{\boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \cdots, \boldsymbol{\lambda}_n^*\}$, we set a threshold τ to decide which channels are pruned. In the original Network Slimming method, the global pruning threshold is determined according to a predefined reduction ratio of the target structure's parameter size. However, a more practical approach is to find the pruned structure based on the FLOPS constraints of the target structure. A global threshold τ can be determined by binary search until the pruned structure satisfies the constraints.

Algorithm 1 summarizes the searching strategy. Notice that a model architecture generator $G(\cdot)$ is required to generate a model structure given a set of channel number configurations. Here we only decide the channel number of each convolutional layer and do not change the original layer connection topology.

Algorithm 1 Searching For Pruned Structure

Require: Optimized channel gate values Λ^*, maximum FLOPS C, model architecture generator $G(\Lambda)$, iterations T, relative tolerance ratio ϵ, $\tau_{\min} = 0, \tau_{\max} = 1$
Ensure: Final threshold τ^*, pruned model architecture $\mathcal{A}^*$
1: **for** $t \leftarrow 1$ to T **do**
2: $\quad \tau_t = \frac{1}{2}(\tau_{\min} + \tau_{\max})$
3: $\quad$ Get pruned channel gates Λ_t by threshold τ_t
4: $\quad$ Get pruned model architecture $\mathcal{A}_t = G(\Lambda_t)$
5: $\quad C_t = \texttt{calculate\_FLOPS}(\mathcal{A}_t)$
6: $\quad$ **if** $|C_t - C|/C \leq \epsilon$ **then**
7: $\quad\quad \tau^* = \tau_t, \ \mathcal{A}^* = \mathcal{A}_t$
8: $\quad\quad$ break
9: $\quad$ **end if**
10: $\quad$ **if** $C_t < C$ **then** $\tau_{\min} = \tau_t$ **else** $\tau_{\max} = \tau_t$
11: **end for**

Implementations

Channel Expansion The new pruning pipeline allows us to explore a larger model search space with no cost. We can change the full model size and then obtain the target pruned structure by slimming network. The easiest way to change model capacity is to use uniform channel expansion, which uniformly enlarges or shrinks the channel numbers of all layers with a common width multiplier. As for the networks with skip connection such as ResNet (He et al. 2016), the number of final output channels of each block and the number of channels at the block input are simultaneously expanded by the same multiplier to ensure that the tensor dimensions are the same.

Budget Training A significant finding in (Liu et al. 2019) is that a pruned network can achieve similar performance to a full model as long as it is adequately trained for a sufficient period. Therefore, the authors in (Liu et al. 2019) proposed "Scratch-B" training scheme, which trains the pruned model for the same amount of computation budget with the full model. For example, if the pruned model saves $2\times$ FLOPS, we double the number of basic training epochs, which amounts to a similar computation budget. Empirically, this training scheme is crucial for improving the pruned model performance.

Experiments

Settings

We conduct all the experiments on CIFAR10 and ImageNet datasets. For each dataset, we allocate a separate validation set for evaluation while learning the channel gates. Specifically, we randomly select 5,000 images from the original CIFAR10 training set for validation. For ImageNet, we randomly select 50,000 images (50 images for each category) from the original training set for validation. We adopt conventional training and testing data augmentation pipelines (He et al. 2016).

When learning channel importance for the models on CIFAR10 dataset, we use Adam optimizer with an initial learning rate of 0.01 with a batch-size of 128. The balance factor $\gamma = 0.5$ and total epoch is 10. All the models are expanded by $1.25\times$, and the predefined sparsity ratio r equals the percentage of the pruned model's FLOPS to the full model. After searching for the pruned network architecture, we train the pruned model from scratch following the same parameter settings and training schedule in (He et al. 2018a).

When learning channel importance for the models on ImageNet dataset, we use Adam optimizer with an initial learning rate of 0.01 and a batch-size of 100. The balance factor $\gamma = 0.05$ and total epoch is 1. During training, we evaluate the model performance on the validation set multiple times. After finishing the architecture search, we train the pruned model from scratch using SGD optimizer. For MobileNets, we use cosine learning rate scheduler (Loshchilov and Hutter 2016) with an initial learning rate of 0.05, momentum of 0.9, weight-decay of 4×10^{-5}. The model is trained for 300 epochs with a batch size of 256. For ResNet50 models, we follow the same hyper-parameter settings in (He et al. 2016). To further improve the performance, we add label smoothing (Szegedy et al. 2016) regularization in the total loss.

CIFAR10 Results

We run each experiment five times and report the "mean $\pm$ std." We compare our method with other pruning methods, including naive uniform channel number shrinkage (uniform), ThiNet (Luo, Wu, and Lin 2017), Channel Pruning (CP) (He, Zhang, and Sun 2017), L1-norm pruning (Li et al. 2016), Network Slimming (NS) (Liu et al. 2017), Discrimination-aware Channel Pruning (DCP) (Zhuang et al. 2018), Soft Filter Pruning (SFP) (He et al. 2018a), rethinking the value of network pruning (Rethink) (Liu et al. 2019), and Automatic Model Compression (AMC) (He et al.

Table 2: Network pruning results on CIFAR10 dataset. "Ratio" stands for the percentage of the pruned FLOPS compared to the full model. Larger ratio stands for a more compact model. "Baseline" and "Pruned" columns stand for the accuracy of baseline and pruned models in percentage. "Δ Acc" stands for the difference of the accuracy level between baseline and pruned model, and larger is better.

	Method	Ratio	Baseline (%)	Pruned (%)	Δ Acc (%)
ResNet20	SFP	40%	92.20	90.83±0.31	-1.37
	Rethink	40%	92.41	91.07±0.23	-1.34
	Ours	40%	91.75	91.14±0.32	**-0.61**
	uniform	50%	90.50	89.70	-0.80
	AMC	50%	90.50	90.20	**-0.30**
	Ours	50%	91.75	90.55±0.14	-1.20
ResNet56	uniform	50%	92.80	89.80	-3.00
	ThiNet	50%	93.80	92.98	-0.82
	CP	50%	93.80	92.80	-1.00
	DCP	50%	93.80	93.49	-0.31
	AMC	50%	92.80	91.90	-0.90
	SFP	50%	93.59	93.35±0.31	-0.24
	Rethink	50%	93.80	93.07±0.25	-0.73
	Ours	50%	93.23	93.05±0.19	**-0.18**
ResNet110	L1-norm	40%	93.53	93.30	-0.23
	SFP	40%	93.68	93.86±0.30	+0.18
	Rethink	40%	93.77	93.92±0.13	+0.15
	Ours	40%	93.49	93.69±0.28	**+0.20**
VGG16	L1-norm	34%	93.25	93.40	+0.15
	NS	51%	93.99	93.80	-0.19
	ThiNet	50%	93.99	93.85	-0.14
	CP	50%	93.99	93.67	-0.32
	DCP	50%	93.99	94.16	+0.17
	Ours	50%	93.44	93.63±0.06	**+0.19**
VGG19	NS	52%	93.53	93.60±0.16	+0.07
	Rethink	52%	93.53	93.81±0.14	+0.28
	Ours	52%	93.40	93.71±0.08	**+0.31**

2018b). We compare the performance drop of each method under the same FLOPS reduction ratio. A smaller accuracy drop indicates a better pruning method.

Table 2 summarizes the results. Our method achieves less performance drop across different model architectures compared to the state-of-the-art methods. For large models like ResNet110 and VGGNets, our pruned model achieves even better performance than the baseline model. Notably, our method consistently outperforms Rethink method, which also utilizes the same budget training scheme. This validates that our method discovers a more efficient and powerful pruned model architecture.

ImageNet Results

In this section, we test our method on ImageNet dataset. We mainly prune three types of models: MobileNet-V1 (Howard et al. 2017), MobileNet-V2 (Sandler et al. 2018), and ResNet50 (He et al. 2016). We compare our method with uniform channel expansion, ThiNet, SFP, CP, AMC, and NetAdapt (Yang et al. 2018). We report the top-1 accuracy of

Table 3: Network pruning results on ImageNet dataset. For uniform channel expansion models, we expand the channels of each layer with a fixed ratio m, denoted as "$m\times$". "Baseline $1.0\times$" stands for the original full model. "Params" column summarizes the sizes of the total parameters of each pruned models.

	Model	Params	Latency	FLOPS	Top-1 Acc (%)
MobileNet-V1	Uniform $0.5\times$	1.3M	20ms	150M	63.3
	Uniform $0.75\times$	3.5M	23ms	325M	68.4
	Baseline $1.0\times$	4.2M	30ms	569M	70.9
	NetAdapt	–	–	285M	70.1
	AMC	2.4M	25ms	294M	70.5
	Ours $0.5\times$	1.0M	20ms	150M	**65.5**
	Ours $0.75\times$	1.9M	21ms	286M	**70.7**
	Ours $1.0\times$	4.0M	23ms	567M	**71.6**
MobileNet-V2	Uniform $0.75\times$	2.6M	39ms	209M	69.8
	Baseline $1.0\times$	3.5M	42ms	300M	71.8
	Uniform $1.3\times$	5.3M	43ms	509M	**74.4**
	AMC	2.3M	41ms	211M	70.8
	Ours $0.75\times$	2.6M	37ms	210M	**70.9**
	Ours $1.0\times$	3.5M	41ms	300M	**72.1**
	Ours $1.3\times$	4.5M	42ms	511M	74.1
ResNet50	Uniform $0.5\times$	6.8M	50ms	1.1G	72.1
	Uniform $0.75\times$	14.7M	61ms	2.3G	74.9
	Uniform $0.85\times$	18.9M	62ms	3.0G	75.9
	Baseline $1.0\times$	25.5M	76ms	4.1G	76.1
	ThiNet-30	–	–	1.2G	72.1
	ThiNet-50	–	–	2.1G	74.7
	ThiNet-70	–	–	2.9G	75.8
	SFP	–	–	2.9G	75.1
	CP	–	–	2.0G	73.3
	Ours $0.5\times$	4.6M	44ms	1.0G	**72.8**
	Ours $0.75\times$	9.2M	52ms	2.0G	**75.6**
	Ours $0.85\times$	17.9M	60ms	3.0G	**76.7**
	Ours $1.0\times$	21.5M	67ms	4.1G	**77.2**

each method under the same FLOPS constraint.

Table 3 summarizes the results. When compressing the models, our method outperforms both uniform expansion models and other complicated pruning strategies across all three architectures. Since our method allows the base channel expansion, we can realize the neural architecture search by pruning the model from an enlarged supernet. Our method achieves comparable or even better performance than the original full model design. We also measure the model CPU latency under batch size 1 on a server with two 2.40GHz Intel(R) Xeon(R) CPU E5-2680 v4. Results show that our model achieves similar or even faster model inference speed than other pruned models. These results validate that it is both effective and scalable to prune model from a randomly initialized network directly.

Comparison with Lottery Ticket Hypothesis

According to the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin 2019), a pruned model can only be trained to a competitive performance level if it is re-initialized to the original full model initialization weights ("winning tick-

Table 4: We compare the pruned model performance under the same pruning ratio (PR). All the models are trained for five runs on CIFAR10 dataset. "Random" stands for our method. "Lottery" stands for lottery-ticket hypothesis, which uses the original full model initialization for pruning when re-training the pruned model from scratch.

Model	PR	Random (Ours)	Lottery (Frankle'19)
ResNet20	40%	**91.14±0.32**	90.94±0.26
ResNet20	50%	**90.44±0.14**	90.34±0.36
ResNet56	50%	**93.05±0.19**	92.85±0.14
ResNet110	40%	**93.69±0.28**	93.55±0.37
VGG16	50%	**93.63±0.06**	92.95±0.22
VGG19	52%	**93.71±0.08**	93.51±0.21

ets"). In our pipeline, we do not require that the pruned model has to be re-initialized to its original states for re-training the weights. Therefore, we conduct comparison experiments to testify whether LTH applies in our scenario. Table 4 summarizes the results. We trained all the models for five runs on CIFAR10 dataset. From the results, we conclude that our method achieves higher accuracy of the pruned models in all the cases. For Lottery Ticket Hypothesis, we do not observe the necessity of its usage. Similar phenomena are also observed in (Liu et al. 2019). There are several potential explanations. First, our method focuses on structured pruning, while LTH draws conclusions on the unstructured pruning, which can be highly sparse and irregular, and a specific initialization is necessary for successful training. Second, as pointed by (Liu et al. 2019), LTH uses Adam optimizer with small learning rate, which is different from the conventional SGD optimization scheme. Different optimization settings can substantially influence the pruned model training. In conclusion, our method is valid under the mild pruning ratio in the structured pruning situation.

Computational Costs for Pruning

Since our pruning pipeline does not require updating weights during structure learning, we can significantly reduce the pruned model search cost. We compare our approach to traditional Network Slimming and RL-based AMC pruning strategies. We measure all model search time on a single NVIDIA GeForce GTX TITAN Xp GPU.

When pruning ResNet56 on the CIFAR10 dataset, NS and AMC take 2.3 hours and 1.0 hours, respectively, and our pipeline only takes 0.12 hours. When pruning ResNet50 on ImageNet dataset, NS takes approximately 310 hours (including progressive training process) to complete the entire pruning process. For AMC, although the pruning phase takes about 3.1 hours, a pre-trained full model is required, which is equivalent to about 300 hours of pre-training. Our pipeline takes only 2.8 hours to obtain the pruned structure from a randomly initialized network. These results illustrate the superior pruning speed of our method.

Visualizing Pruned Structures

Figure 3 displays the channel numbers of the pruned models on CIFAR10 and ImageNet datasets. For each network ar-
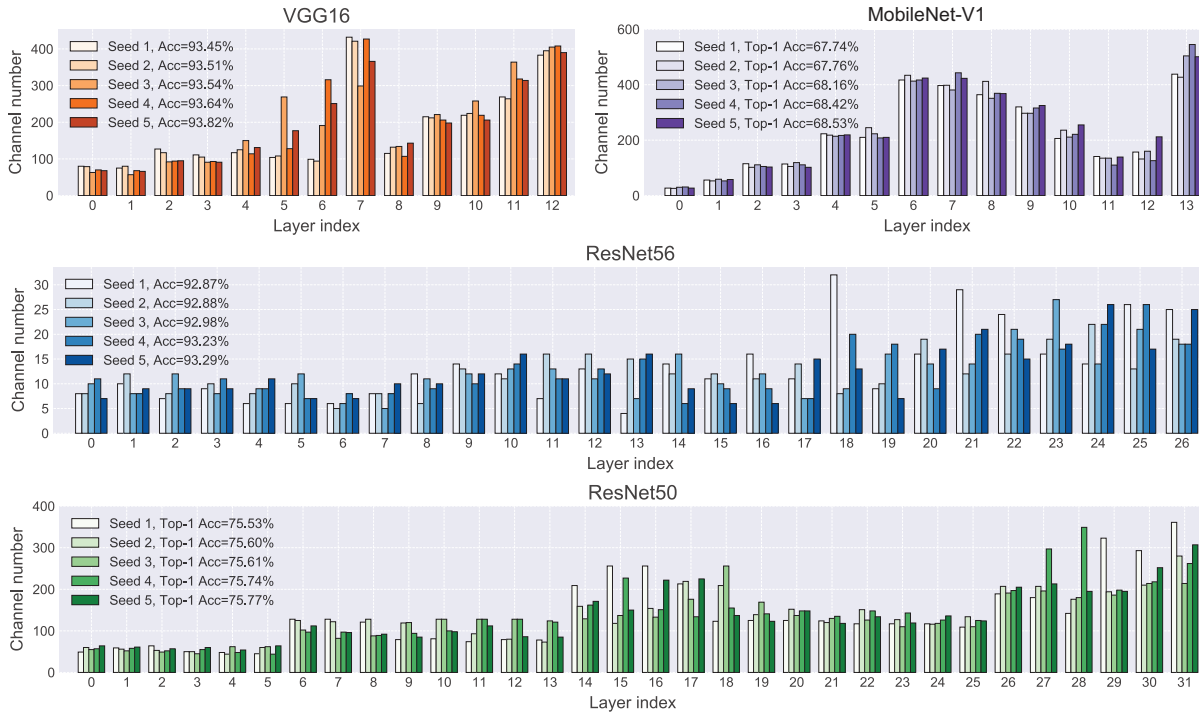
Figure 3: Visualization of channel numbers of the pruned models. For each network architecture, we learn the channel importance and prune 50% FLOPS compared to the full model under five different random seeds. VGG16 and ResNet56 are trained on CIFAR10, and MobileNet-V1 and ResNet50 are trained on ImageNet.
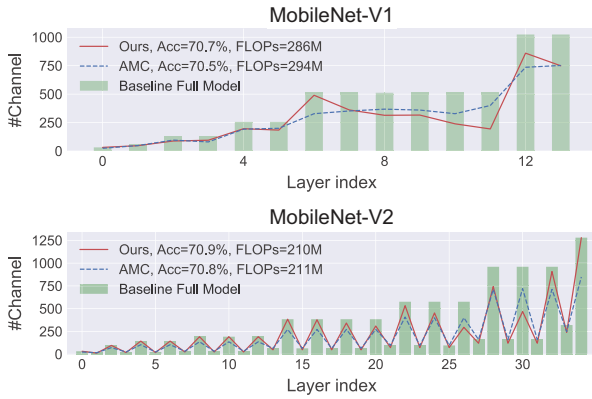

Figure 4: Pruned model structure compared with AMC. Both models are trained on the ImageNet dataset. We include the top-1 accuracy and FLOPS of each model in the legend.

chitecture, we learn the channel importance and prune 50% FLOPS compared to the full model under five different random seeds. Though there are some apparent differences in the channel numbers of the intermediate layers, the resulting pruned model performance remains similar. This demonstrates that our method is robust and stable under different initialization methods.

We also compare the pruned structures with those identified by AMC (He et al. 2018b), which utilizes a more complicated RL-based strategy to determine layer-wise pruning

ratios. Figure 4 summarizes the difference. On MobileNet-V1, our method intentionally reduces more channels between the eighth and eleventh layers, and increases channels in the early stage and the final two layers. The similar trend persists in the last ten layers of MobileNet-V2. This demonstrates that our method can discover more diverse and efficient structures.

Discussion and Conclusions

In this work, we demonstrate that the pipeline of pruning from scratch is efficient and effective through extensive experiments on various models and datasets. An important observation is that pre-trained weights reduce the search space for the pruned structure. Meanwhile, we also observe that even after a short period of pre-training weights, the possible pruned structures have become stable and limited. This perhaps implies that the learning of structure may converge faster than weights. Although our pruning pipeline fixes the random initialization weights, it needs to learn the channel importance. This is equivalent to treating each weight channel as a single variable and optimizing the weighting coefficients. The pruned structure learning may become easier with reduced degree of variables.

search Program.

References

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *Proceedings of the International Conference on Machine Learning (ICML)*, 549–558.

Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2018. Smash: one-shot model architecture search through hypernetworks. In *International Conference on Learning Representations (ICLR)*.

Cai, H.; Zhu, L.; and Han, S. 2019. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*.

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

Gordon, A.; Eban, E.; Nachum, O.; Chen, B.; Wu, H.; Yang, T.-J.; and Choi, E. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1586–1595.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems (NeurIPS)*, 1135–1143.

Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M. A.; and Dally, W. J. 2016. Eie: efficient inference engine on compressed deep neural network. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 243–254.

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

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

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018a. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2234–2240.

He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; and Han, S. 2018b. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 784–800.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1389–1397.

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*.

Krizhevsky, A., et al. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

LeCun, Y.; Denker, J. S.; and Solla, S. A. 1990. Optimal brain damage. In *Advances in neural information processing systems (NeurIPS)*, 598–605.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2736–2744.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR)*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*.

Loshchilov, I., and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*.

Luo, J.-H., and Wu, J. 2018. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *arXiv preprint arXiv:1805.08941*.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 5058–5066.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

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

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

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

Yang, T.-J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sandler, M.; Sze, V.; and Adam, H. 2018. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 285–300.

Zhuang, Z.; Tan, M.; Zhuang, B.; Liu, J.; Guo, Y.; Wu, Q.; Huang, J.; and Zhu, J. 2018. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 875–886.