Revisiting Random Channel Pruning for Neural Network Compression

Yawei Li[1] Kamil Adamczewski[2] Wen Li[3] Shuhang Gu[4] Radu Timofte[1] Luc Van Gool[1,5]
[1]Computer Vision Lab, ETH Zürich [2]MPI-IS [3]UESTC [4]USYD [5]KU Leuven
{yawei.li, radu.timofte, vangool}@vision.ee.ethz.ch

Abstract

Channel (or 3D filter) pruning serves as an effective way to accelerate the inference of neural networks. There has been a flurry of algorithms that try to solve this practical problem, each being claimed effective in some ways. Yet, a benchmark to compare those algorithms directly is lacking, mainly due to the complexity of the algorithms and some custom settings such as the particular network configuration or training procedure. A fair benchmark is important for the further development of channel pruning.

Meanwhile, recent investigations reveal that the channel configurations discovered by pruning algorithms are at least as important as the pre-trained weights. This gives channel pruning a new role, namely searching the optimal channel configuration. In this paper, we try to determine the channel configuration of the pruned models by random search. The proposed approach provides a new way to compare different methods, namely how well they behave compared with random pruning. We show that this simple strategy works quite well compared with other channel pruning methods. We also show that under this setting, there are surprisingly no clear winners among different channel importance evaluation methods, which then may tilt the research efforts into advanced channel configuration searching methods. Code will be released at https://github.com/ofsoundof/random_channel_pruning.

1. Introduction

Since the advent of deep learning based computer vision solutions, network compression has been at the core of reducing the computational complexity of neural networks, accelerating their inference, and enabling their deployment on resource constrained devices [20,21,31,41,55,56,63,65]. Channel pruning (or structured pruning, filter pruning) is one of the approaches that can achieve the acceleration of convolutional neural networks (CNNs) [10, 18, 30, 32, 40].

The goal of this paper is to conduct an empirical study on channel pruning procedure that is not paid enough attention to, *i.e.* random channel pruning. By random pruning, we mean that the pruning ratio of each layer is randomly selected and the channels to be pruned within the layer are determined by some criterion. Random pruning is frequently referred as a baseline to show the improvements of the state-of-the-art channel pruning methods [11, 12, 35, 39, 42, 47, 49, 61, 64]. Yet, the power of random pruning is not fully released. By the rigorous study in this paper, we have several striking findings as follows.

- F1 When brought to the same setting under random pruning, the recent proposed channel pruning criteria [19, 36, 44, 49] performs just comparable with the simple L1 and L2 norm based pruning criteria.
- F2 Compared with channel pruning algorithms that start with a pre-trained model [9, 17–19, 23, 37–39, 45, 60, 66] (See results in Table 3), random pruning can find a pruned model with comparable or even superior performances.
- F3 Even compared with advanced pruning methods that optimize the overall network architecture such placement of pooling layers [40] and expansion of available network width [57], random pruning still narrows the performance gap (less than 0.5% on ImageNet classification).
- F4 Fine-tuning epochs has a strong influence on the performance of the pruned network. High-performing pruned networks usually comes with prolonged fine-tuning epochs.

Those findings lead to several implications. *First of all*, considering F1, since L1/L2 based channel pruning could perform as well as the other pruning criteria, by the law of Occam's razor, **most of the cases, the simple L1 and L2 based pruning criteria can just serve the purpose of channel pruning.** *Secondly*, combining F2 and F3, **random pruning as a neutral baseline, reveals the fundamental development in the field of network pruning.** For algorithms that rely on the predefined network architecture and pre-trained network weight, we haven't gone far

since the advent of network pruning. Beyond that, overall network architecture optimization brings additional benefits. The performance difference of most methods fall into a narrow range of 1%, which is close to the performance of the original network. This on the one hand shows the characteristic of channel pruning, *i.e.* the performance of the channel pruned network is upper bounded by the original network [1]. On the other hand, it shows the difficulty of the problem, *i.e.* every small improvement comes with huge efforts (mostly computation). *Thirdly*, considering F4, **for a fair comparison and a long-lasting development of the field, fine-tuning epoch should be standardized.** We encourage researchers in this field to explain in detail the training and fine-tuning protocol especially the number of epochs. As such, computational cost could be kept in mind for both researchers and industrial practitioners.

The discussion above leads to the unique role that random pruning could play in channel pruning, *i.e.* serving as a baseline to benchmark different channel pruning methods [5]. On the one hand, random channel pruning could bring different pruning criteria under the same regime. As such, the different channel importance estimation methods becomes a meta component which is fit to work with the existing methods. On the other hand, random pruning can become a baseline for other algorithms. Since the performance of channel pruning algorithms can be influenced by a couple of factors especially the fine-tuning procedure, decoupling the influential factors and neutrally showing costs and benefits helps creating clarity. Random channel pruning also simplifies the pruning algorithm. Instead of resorting to sophisticated algorithms such as reinforcement learning [18], evolutionary algorithms [40], and proximal gradient descent [30], channel pruning can be simplified to randomly sampling a pool of sub-networks and selecting the best from them.

In this paper, random pruning is studied in two settings. In the first setting, the task is to prune a pre-trained network. In the second setting, a pre-trained network is not needed and the pruning algorithm starts with a randomly initialized network. The problem is formulated as an architecture search problem. To cope with the searching, the network is reparameterized with an architecture similar to that of the original network. Since the network is trained and pruned from scratch, the second setting is referred to as 'pruning from scratch' in this paper. In both cases, random pruning aims at searching the optimal numbers of channels for a compact network, by randomly sampling the space of all possible channel configurations. Although being extremely easy, random pruning performs surprisingly well compared to the carefully designed pruning algorithms. The surprising success of random pruning also call for an optimized sampling method that improves the search efficiency.

[1] More discussion in the supplementary.

In short, the contributions of this paper are as follows.

1) We present random pruning, a simplified channel pruning method as a strong baseline to benchmark other channel pruning methods. The properties of random pruning are analyzed in this paper.
2) We formalize the basic concepts in channel pruning and try to analyze the reason why random pruning could lead to results comparable to those of carefully designed algorithms.
3) We benchmark a number of channel pruning methods, incl. criteria for random pruning, to get a feel for the current status of channel pruning.

2. Related Works

Channel pruning methods are one of the primary ways to compress neural networks along with the reduction of number of bits in weights via quantization [7, 13] and low rank approximation [24, 31, 62, 63]. The purpose of channel pruning methods is to create a thinner architecture while incurring minimal loss in performance relative to that of the original network.

The early pruning methods concentrate around, so-called, unstructured pruning which removes single parameters from networks [14, 48]. These approaches, though interesting theoretically, are more difficult to implement within current hardware and software settings. Therefore, much recent work has focused on structured pruning where network channels can be removed and the models can be practically compressed and accelerated [2].

The pruning methods fit in different paradigms. Most common pruning approaches rely on pruning the parameters based on the magnitude of the weights, such as L1/L2 norm [56], or more recent median pruning [19]. When convolved, weights provide direct way to compute and, after pruning, approximate the feature maps [45]. Assessing output feature maps, which corresponds to the channels can be an alternative to analyze the importance of the parameters in the network [37, 66]. Another group of pruning methods which have been developed over a few decades utilize the gradient of the loss function with respect to the weights by means of first-order or second-order Taylor series approximations [15, 27, 49]. In this line of work, the weights of smaller importance have smaller impact on the loss function and therefore can be removed. Recent approaches are varied and include assessing channel importance by KL-divergence [44], simulated annealing [50], importance sampling [3], and learning Dirichlet distribution over parameters [1].

Recently, pruning methods have intertwined with knowledge distillation where two networks, a large and a small one share output information to produce similar results [21, 30, 33, 54]. Such approach can be also combined with generative adversarial learning for pruning [38].

Nevertheless, the issue with these methods is that although they provide the importance score for the weights, they neither indicate how many parameters should be pruned nor provide little justification as to choices of the pruned architecture. However, it is widely considered that some of the pruned architectures can be better than others [12]. Our work suggests a random architecture search, a simple, unbiased and general approach to compare most of the pruning methods and allows to find a good architecture given a pre-defined model.

We also noticed other works that try to compare different methods [22, 41]. Yet, this paper is fundamentally different from those works in the aim and the enlightenment from the analysis. The aim of [41] is to identify the value of network pruning as discovering the network architecture whereas our aim is to propose random pruning as a neutral baseline to compare different pruning methods. The study in [22] "guides and motivates the researchers to design more reasonable criteria" while our study finds out that advanced pruning criteria behaves just comparable with the naive L1/L2 norm and calls for an optimized sampling method for efficient search. More discussion is given in the supplementary.

3. Definition and Preliminaries

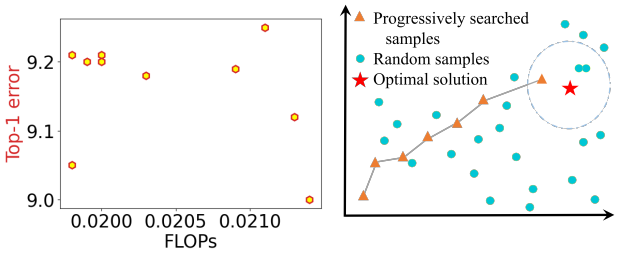
3.1. Basic concepts and formalization

Before delving into the details of the random pruning procedure in this paper, a couple of concepts are first introduced in this section.

Definition 1 (Random selection in channel pruning). As far as random pruning in the network is concerned, the randomness could occur in different ways. *I. Fully random.* The channels to be pruned are fully randomly selected without any constraint across layers. This is often used as a weak baseline [12, 42, 49]. *II. Constrained Random.* The pruning ratio of each layer is determined according to some prior knowledge. The pruned channels within a layer are randomly selected. This is studied in [47]. *III. Random channel number selection.* The pruning ratio of each layer is randomly sampled and the filters in a layer are pruned according certain criteria. In this paper, the third case of random pruning is studied.

Definition 2 (Channel Configuration Space). The channel configuration space $\mathbb{E}$ of a network is defined as the space that contains all of the possible channel number configurations. Let c_{l_i} be the number of channels in a layer l_i, then the number of channel configurations within a layer is $2^{c_{l_i}} - 1$ (we need at least one channel in a layer) and the space of all the configurations contains $\prod_i^n 2^{c_{l_i}} - 1$ samples, where n is the number of layers in an architecture.

Different configurations in the space have varying model complexity (computation, number of parameters, latency) and accuracy. Channel pruning methods aim at finding a target channel configuration that maximizes the accuracy of the network given a fixed model complexity. The configuration space is very different from the parameter space of a network. In the following, two properties that highly influence the channel pruning algorithms are summarized.



(a) Performance of networks in a neighborhood. (b) Searching of networks in the configuration space.

Figure 1. (a) Slightly modified ResNet20 on CIFAR10 image classification. The accuracy of the networks in the local region of the configuration space does not vary a lot. (b) Random pruning only needs to get a sample in the neighborhood of the optimal solution in the configuration space.

Property 1: The channel configuration space is discrete. Conducting differentiable analysis in this space is impossible. This property constitutes a major challenge for channel pruning and architecture search methods. To conduct a search in the space, reinforcement learning, evolutionary algorithm, and also proximal gradient descent have been utilized [18, 32, 40].

Property 2: Slightly changing the channel number of a network does not change the accuracy of the network too much, which means that channel configurations in a local region of the configuration space tend to have similar accuracy. This property is shown in Fig. 1a, where the accuracy of the network in the top-left region does change a lot.

This property means that the solution to channel pruning problem is not unique. Instead, a group of solutions can exist. This sheds light on the effectiveness of random pruning. Regularization-based methods gradually update from initial networks to the optimal solutions [30, 56]. By contrast, random pruning only needs to get a sample in the neighborhood of the optimal solution instead of optimal solution itself (See Fig. 1b). As mentioned in the introduction, we study random pruning for channel pruning in two settings. We describe them below.

Setting 1: Pruning pre-trained networks. In this setting, channel pruning methods take a pre-trained network and prune the less important channels according to an importance score.

Setting 2: Pruning from scratch. In this setting, the network is trained from scratch [4, 6, 32, 40, 58, 59]. During each

mini-batch iteration, sub-networks in the allowable channel configuration space in Sec. 5.1 are trained in parallel such that four sub-networks are sampled and used for parameter update. To cope with the parallel training, a network with architectures similar to the original network is rebuilt according to the description in Sec. 5. After the training, optimized searching method is used to seek the candidate networks [4,6,58]. A recent work also incorporates the searching phase into the training phase by penalizing parameters in the rebuilt network, achieving faster convergence [32].

In the process of pruning the network, the crucial benchmark is the evaluation of the pruned model itself. When pruning and finetuning are done iteratively, it is possible to evaluate the performance of the network during pruning. But if the network is severely pruned, the accuracy of the network drops drastically. For example, when directly pruning 30% of the computation in MobileNetV2, Top-1 error could deteriorate to 90%. Directly evaluating the network in this case becomes unreliable. In short, we are faced with the challenge: *how to evaluate the performance of a pruned network in an efficient way?*

For the two pruning settings, there exist different solutions. When pruning a pre-trained network with random pruning, the parameters of the pruned network are updated by minimizing the difference between the feature maps of the pruned network and the original network layer by layer Compared with finetuning the network for several epochs, the updating the parameters is more efficient, especially when the number of random samples is large. When pruning from scratch, the solution lies in the parallel training procedure of the network. During training, a large number of sub-networks are sampled. The network is trained such that the accuracy of all of the sub-networks tends to decrease. Parallel training arms the network with the capability of interpolating the accuracy of unsampled sub-networks. Thus, after training, it is possible to evaluate the performance of the sampled sub-networks reliably.

3.2. Pruning criteria

For channel pruning, it is crucial to evaluate the relative importance of the channels. There exist several methods that try to measure the channel importance score from different perspectives. The most straightforward method is based on the L1/L2 norms of the filters. Consider an individual layer in a network with weight parameters $\mathbf{W} = [\mathbf{W}_1, \cdots, \mathbf{W}_n]$, where $\mathbf{W} \in \mathbb{R}^{n\times c\times w\times h}$, $\mathbf{W}_i \in \mathbb{R}^{c\times w\times h}$ denotes the i-th output channel of the network (for clarity, we omit the bias). n, c, $w \times h$ denote the number of output channel, input channel, and kernel size of the layer. Then the L1/L2 norm based importance score is computed as $\mathcal{I}_{norm} = \|\mathbf{W}_i\|_p$, where p could be 1 or 2. The filters with smaller norms are likely to be pruned since they generate output feature map with smaller magnitude.

Yet, some work point out that relying on L1/L2 norms could be problematic since the batch normalization layer could recalibrate the magnitude of the feature map [56]. In addition, the "smaller-norm-less-informative" criteria does not respect the distribution of filters in the network [19]. Thus, in [19] geometric median is proposed to overcome the problem. This criteria discovers the similar filters which could be replaced by the other filters, $\mathcal{I}_{gm} = \sum_j \mathcal{S}(\mathbf{W}_i, \mathbf{W}_j)$, where $\mathcal{S}(\cdot,\cdot)$ denotes the similarity between two filters.

The above criteria are only based on the distribution of the filters in the network, which may not fully respect their influence on the accuracy of the network. Thus, in [44], Kullback–Leibler divergence is used to measure the importance of a channel by masking out it in the network, $\mathcal{I}_{kl} = \sum_k P_k \log(\frac{P_k}{Q_k^i})$, where P is the output probability of the original network and Q_k^i is the probability of the pruned network by masking out the single channel in the network. Channels with smaller KL divergence score have weaker influence on the output probability and can be pruned. However, this method requires to conduct one forward-pass for every channel in the network. This is quite slow compared with other methods. In [49], an accelerated computing method by estimating the prediction error with and without a specific parameter. The estimation is done by taking the first- or second- order Taylor expansion of the prediction error. In short, the importance score of a channel is computed by $\mathcal{I}_{te} = (\sum_s g_s w_s)^2$, where w_s denotes a single weight in the channel $\mathbf{W}_i$ and g_s denotes the gradient. Furthermore, in [3, 36], an empirical sensitivity based on the feature map is proposed. Intuitively, the sensitivity of a feature map reflects the relative impact it has on the pre-activations in the next layer. In this paper, we try to compare the six metrics under random pruning.

4. Pruning Pre-trained Networks

In this section, the random procedure for pruning a pre-trained model is introduced. The pipeline is shown in Fig. 2. The pruning algorithm starts with a pre-trained network. The importance score of individual channels in the pre-trained network is first computed. The importance score is the indicator of which channels should be pruned in the next step. Then we select a number of sub-architectures and prune the channels with the lowest score. A sub-architecture is formed by sampling pruning ratios for each layer separately, and then pruning the number of channels given by the ratio. A minimum ratio of remaining channels is set. That is, the range for sampling the pruning ratio is $[\eta, 1]$ Next, the parameters of the pruned network are updated by minimizing the squared difference between features maps of the pruned network and the original network, and the accuracy of the pruned network is evaluated on the

validation set. The top-5 accurate models are selected and fine-tuned for several epochs to further recover the accuracy of the network. Finally, the model with the best accuracy is selected and fine-tuned for longer epochs. Next, the important steps in the pipeline are explained in detail.

4.1. Random sampling

The sub-networks are derived by random sampling the pruning ratio for each layer independently. In total, a population of N sub-networks are sampled. The configurations that meet the target computational complexity are kept. Specifically, let $\mathcal{C}_{prune}$ and $\mathcal{C}_{orig}$ denotes the floating point operations (FLOPs) of the pruned network and the original network, respectively. Then the samples that meet the following criteria are kept, *i.e.*

$$\left| \frac{\mathcal{C}_{prune}}{\mathcal{C}_{orig}} - \gamma \right| <= \mathcal{T}, \tag{1}$$

where γ is the overall pruning ratio of the network and $\mathcal{T}$ is the threshold that confines the difference between the actual and target pruning ratio. During the sampling, the minimum ratio of remaining channels η is empirically set around (equal to or slightly smaller than) the overall pruning ratio γ based on the following considerations. 1) This setting is simple enough and does not involve complicated hyper-parameter tuning. 2) It allows for a reasonably constrained random sampling sub-space for the algorithm to explore. The setting of η prevents the case where a major part of the channels in a layer is pruned. A bottleneck in the network could harm the performance of the pruned network. The random sampling procedure searches the configuration space. Although it seems to be quite easy, it is shown in the experiments that this procedure is surprisingly competitive.

4.2. Updating network parameter

For each sampled sub-architecture, the network is directly pruned according to the per-layer pruning ratio. Yet, the accuracy of the network is very likely to drop drastically after pruning, especially when the pruning ratio is high. Directly evaluating the pruned network is not reliable. The common practice is to fine-tune the network for a few epochs. But this could be time-consuming considering that a large population of sub-networks are sampled. Instead, we opt for another solution, *i.e.* minimizing the distance between the feature maps of the pruned network and the original network [20, 29, 45]. Let $\mathbf{F}_p \in \mathbb{R}^{n' \times d}$ and $\mathbf{F}_o \in \mathbb{R}^{n \times d}$ denote the feature map of the pruned network and the original network, respectively. Note that the feature maps are reshaped into matrices. Since the network is pruned, its feature map has less channels than the original network, *i.e.* $n' < n$. The parameters in the pruned network is updated by minimizing the following loss function

$$\mathcal{L} = \arg\min_{\mathbf{X}} \|\hat{\mathbf{F}}_o - \mathbf{X}\mathbf{F}_p\|_2^2, \tag{2}$$

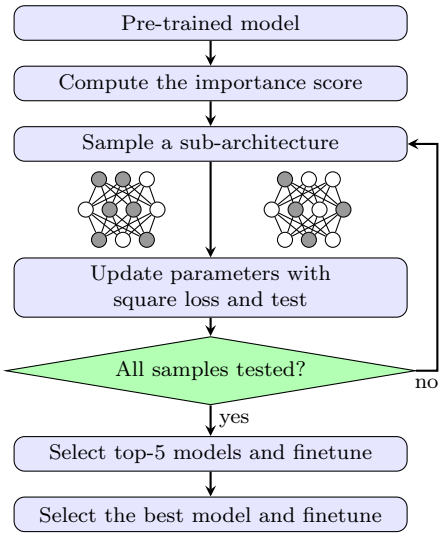


Figure 2. The pipeline of random pruning a pre-trained model.

where $\hat{\mathbf{F}}_o \in \mathbb{R}^{n' \times d}$ is the feature map of the original network with the corresponding channels removed and $\mathbf{X} \in \mathbb{R}^{n' \times n'}$ is the additional parameter that updates the pruned network. The parameter $\mathbf{X}$ can be derived with least square solvers. It can be further merged with the original parameter in the layer of the network. Thus, in fact, no additional parameter or computation is introduced in the pruned network. This parameter updating procedure is done layer-wise.

5. Pruning From Scratch

In this section, the procedure used to prune a network from scratch is described. The pipeline is shown in Fig 3. In this setting, the pipeline starts with the architecture of the original network. We build a slimmable network according to [59]. The permissible channel configurations are described in Sec. 5.1. Then the network is initialized and trained from scratch. Parallel training is conducted. That is, for each mini-batch iteration during training, four sub-networks are sampled including the complete network and three random samples. Four forward and backward passes are conducted. The gradients during the four backward passes are accumulated and used to update the parameters in the network. The maximum network is always sampled, which guarantees that all of the parameters are updated during one iteration. In-place knowledge distillation is used.

After the training stage, the channel configuration is still searched by random sampling. Thus, a population of N sub-networks satisfying Eqn. 1 are derived. Owing to the parallel training, the network gains the capability of interpolating the accuracy of unsampled sub-networks. Thus, the sub-networks can be evaluated directly on the validation set and the accuracy is reliable. After that, the top 50 models

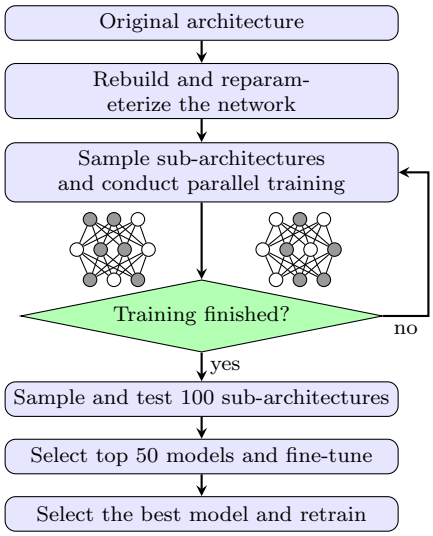


Figure 3. The pipeline of the random pruning from scratch.

are further trained for a few epochs. Finally, the best model among the 50 models is selected and retrained from scratch. In the next subsection, the considerations for rebuilding the network are described.

5.1. Designing the network pruning space

One problem encountered in this setting is that the total number of sub-networks is quite large. Searching in that large search space is a challenge. Thus, to ease the problem, the pruning space is restricted as follows.

1) The number of channels is confined to be multiples of 8. Although making the channel number selection discrete, this strategy reduces the possible network samples significantly. For example in the case ResNet-18, the number of possible sub-network configurations is reduced from 2.4×10^{24} to 2.8×10^{14}. This design is inspired by **Property 2** of the configuration space.
2) The minimum number of channels is reset and rounded to multiples of 8. This again avoids the very narrow bottleneck in the network. For example, when pruning 30% of the FLOPs of ResNet-18, we empirically require 40% of the channels must be kept.
3) For a fair comparison with pruning pre-trained networks, the maximum network width is not expanded.

6. Experimental Results

The experimental results are shown in this section. The experiments are conducted on three commonly used networks, including VGG [53], ResNet [16] and its variants, and MobileNetV2 [52]. For ImageNet [8] experiments, the pre-trained models provided by PyTorch [51] are used as the baseline. For CIFAR [26] experiments, the original network is trained for 300 epochs with the initial learning rate of 0.1 and the batch size of 64. The learning rate decays by 0.1 at the epochs 150 and 225. When pruning the pre-trained models, the pruned architectures with the channels selected by the above methods are tested. The top-5 pruned models are selected and fine-tuned for 5 epochs. Eventually to narrow down the search we choose the best model and fine-tune it again to obtain the final pruned model. For ImageNet and CIFAR, the networks are fine-tuned for 25 and 50 epochs respectively unless otherwise stated. When pruning from scratch, the network is initially trained for 40 epochs. After pruning, the network is reinitialized and retrained for 90 epochs. The population of the sampled sub-network is 100. The threshold $\mathcal{T}$ for random sampling is set to 0.02.

Criterion	Top-1 Error (%)	Top-5 Error (%)	FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
VGG16, Target FLOPs Ratio 70 %				
Baseline	26.63	8.5	15.50 /100.00	138.4 /100.00
L1	27.14	8.68	11.11 /71.67	112.3 /81.14
L2	27.01	8.82	10.87 /70.10	128.8 /93.09
GM	27.96	8.77	11.09 /71.55	108.4 /78.34
TE	27.04	8.77	10.65 /68.70	130.4 /94.27
ES	26.76	8.60	10.38 /66.98	130.9 /94.63
KL	27.22	8.88	10.91 /70.37	128.9 /93.15
ResNet18, Target FLOPs Ratio 70 %				
Baseline	30.24	10.92	1.82 /100.00	11.69 /100.00
L1	32.08	12.02	1.28 /70.53	8.90 /76.10
L2	31.69	11.79	1.31 /71.92	9.58 /81.92
GM	31.76	11.87	1.30 /71.67	9.97 /85.27
TE	31.76	11.83	1.30 /71.38	9.79 /83.77
ES	31.66	11.86	1.30 /71.31	9.41 /80.53
KL	31.90	11.93	1.31 /72.14	9.77 /83.54
Scratch	31.68	11.71	1.28 /70.38	10.06 /86.05
ResNet18, Target FLOPs Ratio 50 %				
Baseline	30.24	10.92	1.82 /100.00	11.69 /100.00
L1	34.98	13.81	0.94 /51.80	7.49 /64.06
L2	35.18	13.97	0.93 /51.30	6.88 /58.88
GM	34.50	13.44	0.92 /50.57	8.04 /68.80
TE	34.66	13.81	0.94 /51.69	8.02 /68.60
ES	35.34	14.03	0.94 /51.90	6.74 /57.67
KL	35.10	13.94	0.93 /50.88	7.78 /66.53
ResNet50, Target FLOPs Ratio 70 %				
Baseline	23.85	7.13	4.11 /100.00	25.56 /100.00
L1	24.77	7.51	2.81 /68.41	18.24 /71.35
L2	24.33	7.35	2.94 /71.48	20.23 /79.15
GM	24.65	7.40	2.87 /69.89	18.60 /72.80
TE	24.69	7.43	2.89 /70.32	20.58 /80.53
ES	24.66	7.48	2.89 /70.26	18.06 /70.66
KL	24.66	7.49	2.92 /70.94	18.60 /72.76
MobileNetV2, Target FLOPs Ratio 70 %				
Baseline	28.12	9.71	0.314 /100.00	3.50 /100.00
L1	32.22	12.04	0.224 /71.22	2.65 /75.74
L2	31.84	11.85	0.225 /71.63	2.62 /74.81
GM	31.89	11.88	0.223 /71.12	2.69 /76.65
TE	32.09	12.01	0.223 /70.87	2.63 /75.16
ES	31.93	11.77	0.223 /71.03	2.63 /75.10
KL	31.96	11.93	0.225 /71.54	2.64 /75.36

Table 1. Benchmarking channel pruning criteria on ImageNet classification under the scheme of random pruning.

6.1. Benchmarking channel pruning criteria

It is worth noting that the implemented random pruning method indicates how many channels of each layer should

Criterion	Top-1 Error (%)	Top-5 Error (%)	FLOPs [G] / Ratio (%)	Params / Ratio (%)
VGG, CIFAR10				
Baseline	5.67	0.58	313.80 /100.00	14.73M /100.00
L1	6.1	0.69	160.50 /51.15	5.05M /34.32
L2	6.06	0.67	150.60 /47.99	6.20M /42.11
GM	5.99	0.52	154.60 /49.27	4.13M /28.04
TE	6.51	0.61	157.00 /50.03	5.84M /39.63
ES	6.21	0.64	157.20 /50.10	7.06M /47.90
KL	6.19	0.66	161.50 /51.47	6.52M /44.26
ResNet56, CIFAR10				
Baseline	5.58	0.26	126.80 /100.00	855.8k /100.00
L1	6.72	0.79	63.60 /50.16	503.6k /58.85
L2	6.52	0.76	64.70 /51.03	471.4k /55.08
GM	6.39	0.77	65.40 /51.58	504.0k /58.89
TE	6.86	0.59	65.70 /51.81	442.4k /51.69
ES	6.59	0.67	65.80 /51.89	545.6k /63.75
KL	7.12	0.67	65.20 /51.42	443.3k /51.80
ResNet20, CIFAR100				
Baseline	31.53	9.87	41.20 /100.00	278.3k /100.00
L1	33.41	10.42	20.80 /50.49	176.2k /63.29
L2	33.39	10.62	21.00 /50.97	175.9k /63.20
GM	33.32	10.35	20.60 /50.00	183.8k /66.03
TE	34.24	10.92	20.00 /48.54	168.8k /60.65
ES	33.81	10.13	21.00 /50.97	176.3k /63.34
KL	33.32	10.62	21.20 /51.46	187.5k /67.35

Table 2. Benchmarking channel pruning criteria on CIFAR10 and CIFAR100 image classification under the scheme of random pruning. More results are given in the supplementary.

be pruned and *which* channels are pruned is decided by the external criteria. A range of pruning methods are compared and benchmarked under the scheme of random pruning, including the traditional L1 and L2 norm of the filters (L1, L2), and the recent method based on geometric median (GM) [19], Taylor expansion (TE) [49], KL-divergence importance metric (KL) [44] and empirical sensitivity analysis (ES) [36]. In addition, the method of pruning from scratch based on slimmable networks [58] is also included.

The benchmark results for ImageNet and CIFAR are shown in Table 1 and Table 2, respectively. The FLOP metric is relatively fixed. Since a threshold $\mathcal{T} = 0.02$ is set, the difference between the target overall pruning ratio and the actual overall pruning ratio is within 2%. During the random sampling, it is difficult to fix both FLOPs and the number of parameters. Thus, the number of parameters of the pruned networks vary. Several conclusions can be drawn by analyzing the results in Table 1 and Table 2. **I.** When comparing different pruning criteria across different networks and datasets under the scheme of random pruning, their performance is close to each other. It is quite surprising that the advanced pruning criteria such as KL and ES do not necessarily outperform the naive ones such as L1 and L2 norm. **II.** The number of parameters have significant influence on the accuracy of the pruned network. When the computational complexity is about the same, pruned networks with more parameters tend to have lower error rate. **III.** Considering the above two observations, we conclude that there

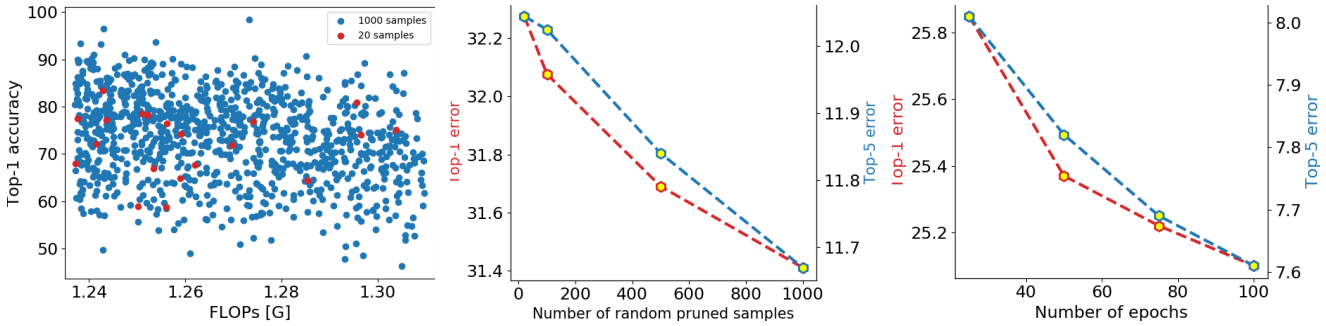
Methods	Epoch	Top-1 Err. (%)	Top-5 Err. (%)	FLOPs Ratio	Params Ratio
ResNet50, ImageNet					
SFP [17]	100	25.39	7.94	58.2	–
GAL-0.5 [38]	30	28.05	9.06	56.97	83.14
SSS [23]	100	28.18	9.21	56.96	61.15
HRank [37]	480	25.02	7.67	56.23	63.33
Random Pruning	25	25.85	8.01	50.72	54.99
Random Pruning	75	25.22	7.69	50.72	54.99
Random Pruning	120	24.87	7.48	48.99	54.12
AutoPruner [43]	32	25.24	7.85	48.79	–
Adapt-DCP [39]	120	24.85	7.70	47.59	45.01
FPGM [19]	90	25.17	7.68	46.5	–
DCP [66]	60	25.05	7.68	44.50	48.44
ThiNet [45]	87	27.97	9.01	44.17	–
MetaPruning [40]	160	24.60	–	48.78	–
AutoSlim [57]	150	24.40	–	80.60	–
MobileNetV2, ImageNet2012					
MetaPruning [40]	160	28.80	–	72.33	–
Random Pruning	120	29.10	–	70.87	–
AMC [18]	120	29.20	–	70.00	–
Adapt-DCP [39]	310	28.55	–	68.92	–
ResNet56, CIFAR10					
GAL-0.5 [38]	100	6.62	–	63.40	88.20
[28]	40	6.94	–	62.40	86.30
NISP [60]	–	6.99	–	56.39	57.40
Random Pruning	50	6.52	–	51.03	55.08
CaP [46]	–	6.78	–	50.20	–
ENC [25]	–	7.00	–	50.00	–
AMC [18]	–	8.1	–	50.00	–
Hinge [30]	300	6.31	–	50.00	48.73
KSE [34]	200	6.77	–	48.00	45.27
FPGM [19]	200	6.74	–	47.4	–
SFP [17]	300	6.65	–	47.4	–

Table 3. Benchmarking different channel pruning methods.

is no clear winner among the seven compared pruning criteria. **IV.** Thus, when the pruned networks are fine-tuned for long enough epochs (*e.g.* , more than 25 epochs), the benefits of advanced pruning criteria is substituted by the prolonged training. **V.** This means that for both pruning a pre-trained model and pruning from scratch, efficient search of the channel configuration space should be at least one of the major research directions.

6.2. Benchmarking channel pruning methods

To further study the status of channel pruning, we incorporate the results of more methods in Table 3. Two networks are used to benchmark different methods including ResNet50 for ImageNet and ResNet56 for CIFAR10. The results are from the original paper. Note that the number of fine-tuning epochs is also included. This is crucial for comparing different channel pruning methods since the number of training epochs has quite important influence on the accuracy of the final pruned networks. More fine-tuning epochs usually leads to more accurate pruned networks. Ideally, for a fair comparison between different methods, the influence of fine-tuning strategy of the pruning algorithm itself should be decoupled. That is, the number of fine-tuning epochs should be fixed. Yet, this is almost impossible since different methods adopt different training and fine-tuning



(a) Comparison between 1000 and 20 samples for ResNet18.

(b) Random samples *vs.* error rate in ResNet18.

(c) Epochs *vs.* error rate in ResNet50.

Figure 4. The influence of the random sample size and fine-tuning epochs on the prediction accuracy. Experiments done on ImageNet.

strategies according to requirements of the methods. In any case, the number of training epochs is still an indicator of the compared algorithms. When the accuracy of two algorithms is close, the one with fewer fine-tuning epochs is obviously better. When the fine-tuning epochs of two algorithms are different, we have a tolerance for the accuracy drop of the one with fewer fine-tuning epochs.

We have a couple conclusions from Table 3. **I.** On CIFAR10, random pruning performs no worse than any of the compared methods. This shows that for this easy case, random pruning could just serve the purpose. **II.** On ImageNet, compared with earlier channel pruning methods including SFP [17], GAL [38], and SSS [23], random pruning outperforms under fewer fine-tuning epochs and severer pruning ratio. **III.** Random pruning is even comparable with the recent work HRank [37] considering the longer fine-tuning epochs and larger remained model. **IV.** Compared with advanced searching methods such as MetaPruning [40], random pruning performs a little bit worse. Yet, we also need to be aware that the slightly changed baseline network for MetaPruning is already in favor of FLOPs reduction. The fine-tuning is also longer. In addition, the potential of random pruning could be fully released as shown in the next subsection. **V.** Compared with methods that only prune the pre-trained networks, overall architecture optimization such as the placement of pooling layers and expansion of maximum network width could bring additional benefits.

6.3. Ablation study

The characteristics of random pruning is ablated.

Influence of the sampling population. In the former experiments, the number of random samples is fixed to 100. In Fig. 4b, to study the influence of the population size, the number of random samples is increased gradually from 20, 100, 500, to 1000. As expected, the Top-1 and Top-5 error drop steadily from 20 samples to 1000 samples. Meanwhile, the gain of more random sampled does not get saturated. For the studied range, the empirical Top-1 error curve is a monotonically decreasing and convex. This means that the gain of accuracy diminished with increase number of samples. As shown in Fig. 4a, when increasing the number of random samples, both better and worse sub-networks could be sampled, which shows the randomness of random pruning. This is acceptable since we are searching for well-performed samples. But from another perspective, this phenomenon also calls for advanced searching methods.

Influence of fine-tuning epochs. The importance of fine-tuning epochs is already emphasized in Sec. 6.2. Here we quantify the influence of fine-tuning epochs by studying ResNet. The result for ResNet-50 is shown in Fig. 4c. The result for ResNet-18 is shown in the supplementary. When the number of fine-tuning epochs is increased from 25 to 100, the Top-1 and Top-5 error of ResNet-50 drops by 0.75% and 0.4%, respectively. This shows the significant influence of fine-tuning epochs. Again, when benchmarking, the fine-tuning strategy should be considered.

Analysis of additional computational cost. Except fine-tuning, other additional computational cost for random pruning includes the evaluation of the pruned models. For pruning pre-trained models, updating the parameters also needs to compute the feature maps, which introduces additional computation. The additional computational cost for evaluation could be reduced by taking out a smaller part (say 5000 for ImageNet) of the validation set for evaluation. This is adopted by some works [18].

7. Conclusion

This work studies the problem of pruning neural network as an unbiased random search for an optimal network architecture. The search can be applied both for learning the architecture from scratch as well as applying it to the pre-trained model with predefined importance score of the channels. As a result, random pruning is a simple, general and explainable baseline which performs well and can be used as a benchmark to more complex pruning methods.

Acknowledgement: This work is partially supported by the ETH Zürich Fund (OK), the National Natural Science Foundation of China (Grant No. 62176047), and an Amazon AWS grant.

References

[1] Kamil Adamczewski and Mijung Park. Dirichlet pruning for neural network compression. *arXiv preprint arXiv:2011.05985*, 2020. 2

[2] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017. 2

[3] Cenk Baykal, Lucas Liebenwein, Igor Gilitschenski, Dan Feldman, and Daniela Rus. Sipping neural networks: Sensitivity-informed provable pruning of neural networks. *arXiv preprint arXiv:1910.05422*, 2019. 2, 4

[4] Maxim Berman, Leonid Pishchulin, Ning Xu, Matthew B Blaschko, and Gérard Medioni. AOWS: Adaptive and optimal network width search with latency constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11217–11226, 2020. 3, 4

[5] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020. 2

[6] Ting-Wu Chin, Ari S Morcos, and Diana Marculescu. PareCO: Pareto-aware channel optimization for slimmable neural networks. *arXiv preprint arXiv:2007.11752*, 2020. 3, 4

[7] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. 6

[9] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014. 1

[10] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal SGD for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019. 1

[11] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021. 1

[12] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 1, 3

[13] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015. 2

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proceedings of International Conference on Learning Representations*, 2015. 2

[15] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 164–171, 1993. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[17] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 1, 7, 8

[18] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for model compression and acceleration on mobile devices. In *Proceeding of the European Conference on Computer Vision*, pages 784–800, 2018. 1, 2, 3, 7, 8

[19] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 1, 2, 4, 7

[20] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 1, 5

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[22] Zhongzhan Huang, Wenqi Shao, Xinjiang Wang, Liang Lin, and Ping Luo. Rethinking the pruning criteria for convolutional neural network. In *Advances in Neural Information Processing Systems*, 2021. 3

[23] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceeding of the European Conference on Computer Vision*, pages 304–320, 2018. 1, 7, 8

[24] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*, 2014. 2

[25] Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. Efficient neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12569–12577, 2019. 7

[26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[27] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605, 1990. 2

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 7

[29] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14639–14647, 2020. 5

[30] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 3, 7

[31] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5623–5632, 2019. 1, 2

[32] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. In *Proceeding of the European Conference on Computer Vision*, pages 608–624. Springer, 2020. 1, 3, 4

[33] Yawei Li, Wen Li, Martin Danelljan, Kai Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. The heterogeneity hypothesis: Finding layer-wise differentiated network architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2144–2153, 2021. 2

[34] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable CNN compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 7

[35] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4410–4419, 2021. 1

[36] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019. 1, 4, 7

[37] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. HRank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020. 1, 2, 7, 8

[38] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019. 1, 2, 7, 8

[39] Jing Liu, Bohan Zhuang, Zhuangwei Zhuang, Yong Guo, Junzhou Huang, Jinhui Zhu, and Mingkui Tan. Discrimination-aware network pruning for deep model compression. *arXiv preprint arXiv:2001.01050*, 2020. 1, 7

[40] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 2, 3, 7, 8

[41] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *Proceedings of International Conference on Learning Representations*, 2019. 1, 3

[42] Zechun Liu, Xiangyu Zhang, Zhiqiang Shen, Zhe Li, Yichen Wei, Kwang-Ting Cheng, and Jian Sun. Joint multi-dimension pruning. *arXiv preprint arXiv:2005.08931*, 2020. 1, 3

[43] Jian-Hao Luo and Jianxin Wu. AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, page 107461, 2020. 7

[44] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1458–1467, 2020. 1, 2, 4, 7

[45] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 1, 2, 5, 7

[46] Breton Minnehan and Andreas Savakis. Cascaded projection: End-to-end network compression and acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10715–10724, 2019. 7

[47] Deepak Mittal, Shweta Bhardwaj, Mitesh M Khapra, and Balaraman Ravindran. Recovering from random pruning: On the plasticity of deep convolutional neural networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 848–857. IEEE, 2018. 1, 3

[48] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017. 2

[49] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 1, 2, 3, 4, 7

[50] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *Advances in Neural Information Processing Systems*, pages 1977–1987, 2019. 2

[51] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. 6

[52] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 6

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[54] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, pages 1365–1374, 2019. 2

[55] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 1

[56] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *Proceedings of International Conference on Learning Representations*, 2018. 1, 2, 3, 4

[57] Jiahui Yu and Thomas Huang. AutoSlim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 1, 7

[58] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1803–1811, 2019. 3, 4, 7

[59] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 3, 5

[60] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 1, 7

[61] Sixing Yu, Arya Mazaheri, and Ali Jannesari. Auto graph encoder-decoder for neural network pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6362–6372, 2021. 1

[62] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017. 2

[63] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2015. 1, 2

[64] Yanfu Zhang, Shangqian Gao, and Heng Huang. Exploration and estimation for model compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 487–496, 2021. 1

[65] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016. 1

[66] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 1, 2, 7