EAPruning: Evolutionary Pruning for Vision Transformers and CNNs

Qingyuan Li
liqingyuan02@meituan.com
Bo Zhang
zhangbo97@meituan.com
Xiangxiang Chu
chuxiangxiang@meituan.com

Meituan
Beijing, China

Abstract

Structured pruning greatly eases the deployment of large neural networks in resource-constrained environments. However, current methods either involve strong domain expertise, require extra hyperparameter tuning, or are restricted only to a specific type of networks, which prevents pervasive industrial applications. In this paper, we undertake a simple and effective approach that can be easily applied to *both vision transformers and convolutional neural networks*. Specifically, we consider pruning as an *evolution process of sub-network structures that inherit weights through reconstruction techniques*. We achieve 50% FLOPS reduction for ResNet50 and MobileNetV1, leading to **1.37**× and **1.34**× speedup respectively. For DeiT-Base, we reach nearly **40%** FLOPs reduction and **1.4**× speedup. Our code will be made available.

1 Introduction

With the large-scale application of AI in the cloud, mobile, autonomous vehicles, and IoT devices, deep neural networks face the challenge of reducing the high computational intensity and great power consumption on restricted devices. Researchers from both academia and industry have endeavored on model compression and acceleration. A large volume of acceleration methods has been proposed. Some design lightweight networks like manually like MobileNet [17] and ShuffleNet [38], or through *neural architecture search* (NAS) [16], which requires deep expertise to invent new structures or to design appropriate search spaces. In contrast, pruning from pretrained networks can be put into direct use. To name a few, one can resort channel pruning [14], network trimming [18], data-free l_1-norm filter pruning [21], structured sparsity learning [32], and greedy search-based ThiNet [25] etc.

In this paper, we attempt to further investigate pruning for its versatility and efficiency. Due to the large number of neural network parameters, the existence of redundant neurons becomes inevitable. It has been a natural idea to cut off "unimportant" neurons. Methods fall in this category can be roughly summarized as *heuristic-based* and *learning-based*. The former includes l_1, l_2 regularization [12, 21], deep compression [11] etc. The latter comprises network slimming [23] and fine-grained sparse pruning [39]. However, according to AlexNet

[19], VGG [27], ResNet [13], it is agreed that network structure plays a decisive role in performance than neurons' position. This observation is echoed by [24]. With the emergence of automated exploration methods like AMC [15], NetAdapt [34], MetaPruning [23], and APQ [30], "structure" search has gradually become the mainstream.

Therefore, we also consider pruning as a process of seeking extreme values in a finite space. Despite above-mentioned advances, searching-based approaches have been hindered for long, since the searching space is limited but still very huge. It becomes impractical to exhaustively evaluate every structure, which goes exponential (2^N, given N as the channel number). However, the optimal choices of "solutions" (i.e. network structures) tend to be similar, for which we can resort to evolutionary algorithm (EA) for searching. EA keeps good structures and discards bad ones until we gradually approach the optimal solution. Specifically, we encode channel by its number as in [22] instead of channel-wise encoding [8, 31, 41] to achieve compression of evolutionary space, we then use weight reconstruction [14] instead of "cumbersome" fine-tuning to recover subnetworks for proxy evaluation. We finally employ a multi-objective evolutionary process NSGA-III [5] to better balance the computation and accuracy trade-off.

In a nutshell, our contribution can be summarized as follows,

- We propose an evolutionary-based pruning method (called EAPruning) for both vision transformers and CNNs, which is proved very effective, easy to use, and comes with low cost. To our best knowledge, it is also the first pruning algorithm that *works for both types of mainstream networks*.
- Our method employs *weight reconstruction* for the fast evaluation of sampled child networks, and *multi-objective evolutionary search* to select a series of target models at once. Benefitting from this paradigm, we don't require tedious hyperparameter tuning and much domain expertise.
- We prune DeiT-Base by nearly **40%** FLOPs reduction with only 0.5% loss in accuracy, and ResNet50 by **50%** FLOPs reduction with merely 0.3% accuracy loss.

2 Related work

Pruning is usually divided into *unstructured pruning* [4, 9, 10, 11, 12, 20] and *structured pruning* depending on the granularity of connections. The former prunes every connection to achieve sparse kernels in the end that won't enjoy much benefit unless it is efficiently supported by the underlying hardware. Therefore, most community effort is put into structured pruning.

Criteria-based Pruning. Structured pruning usually removes redundant filters according to some human-crafted criteria. [21] directly identifies the relative importance of filters by their l_1 norms. Instead, [18] calculates the percentage of activations of a neuron. Some use regularization terms as in [32] to sparsify the network where filters have smaller norms are considered less important and can be safely pruned. Other strategies like network slimming [23] penalize the scaling factor in batch normalization to find the unnecessary channels. In contrast, [26] involves Taylor expansion to determine the importance of filters. These methods commonly come with costly iterative finetuning and require predefined pruning ratios or empirics to obtain appropriate compression. ResRep [6] is another form of criteria that divides neurons into remembering parts and forgetting ones, but it is limited to CNN only.

Search-based Pruning. [24] suggests that it is useful to consider pruning as an architecture search paradigm. Automated pruning methods have also begun to emerge, such as reinforcement learning in AMC [15], greedy search in NetAdapt [34], neural architecture search in OFA [1], APQ [30], and BigNAS [36]. Note OFA and BigNAS benefits from supernet training strategy so that subnetworks with inherited weights don't require finetuning any further.

Vision Transformer Compression. Making vision transformers [7] more efficient is an urgent task. VTP [42] adds soft gates to learn the importance of filters. NViT [33] instead imposes Taylor importance score. AutoFormer [2] applies one-shot weight-sharing neural architecture search to find optimal subnetworks in predefined discrete search spaces.

3 Method

3.1 Motivation

A universal pruning framework is indispensable to satisfy diverse model compression needs. Given the success of vision transformers and CNNs, we are driven to develop an efficient and that is easily applicable to both kinds of networks. It has been shown that structure is more important than weights [24]. There is no obvious difference for channels in the same layer, which disables a family of algorithms that tend to identify the importance of filters, such as Lasso [14], KL divergence [35], Taylor expansion [26]. In this regard, we'd like to resort to searching algorithms to find a better structure. However, previous evolutionary-based pruning methods face several challenges, such as *huge search space* that is too fine-grained down to channel-wise level, and *computationally expensive finetuning* for the evaluation of each subnetwork. Searching becomes very costly and infeasible for increasingly big models and large datasets. To solve these problems, we need to choose a coarse pruning space for CNNs and to identify the effective prunable parts in Vision Transformer that relieves the searching difficulty. To get rid of costly iterative finetuning, meanwhile being general, we wish to apply minimum weight reconstruction to recover pruned structures. We also want to utilize evolutionary algorithms to enable proficient searching, while considering the computation vs. accuracy trade-off.

3.2 Pruning Space

Defining a proper pruning space is critical. The common channel-wise one-hot encoding is too fine-grained. Although it is useful to differentiate important channels from others for the iterative pruning paradigm, this creates unnecessary difficulty for searching-based algorithms. Take ResNet50 as an example, there are a total of 24576 channels excluding the first layer, whose search space is as huge as 2^{24576}, effectively disabling evolutionary search. Instead, we follow the practices in neural architecture search that encodes channel numbers only. Therefore, for ResNet50, we need $\sum \log_2(C_i) \approx 286$ bit long encoding, where C_i is the number of channels of the i-th filter.

For vision transformers, we find it is enough to reduce the prunable space to the number of attention heads (N_{head}) and MLP ratios (r), see Figure 1. We keep the embedding dimension (Dim) unchanged for large models like DeiT-Base as later shown these models tend to be ignored very early in evolution. Specifically for each attention block, we prune the number of heads for Q, K, and V, whose weights are pruned to $N_{head} \times Dim_{head} \times Dim$. Correspondingly,

the weights in projection $Dim \times Dim$ are pruned to $M \times Dim$, where $M = N_{head} \times Dim_{head}$. For MLP layers, we tune the hidden dimension by the ratio r where the weights of FC1 are pruned to $Dim \times r \times Dim_{hidden}$, and FC2 to $r \times Dim_{hidden} \times Dim$.

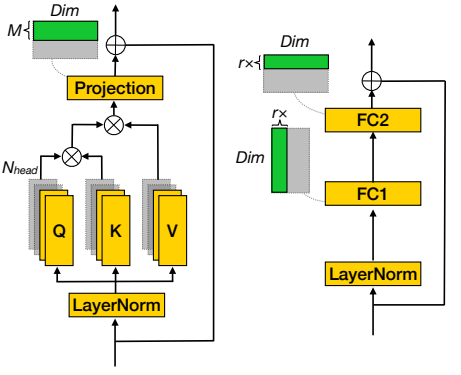

Figure 1: Our pruning space for an attention block in Vision Transformers. We prune the number of heads for Q, K and V (correspondingly the projection dimension) and MLP inner dimension only. The dotted gray area shows weights to be pruned, green area shows the remaining weights for convolutions. The shape of input and output of each block is retained.

3.3 Channel Selection

Since we only encode the number of channels, it is still a problem to decide which channels to preserve. Common practices use l_1-norm as in AMC [15], lasso regression as in channel pruning [14] , or greedy search as in NetAdapt [34] and ThiNet [25]. As we assume that the performance of the subnet only has to do with the structure, we just randomly sample channels to a target number. We later show that using l_1-norm doesn't make much difference.

3.4 Weight Reconstruction

Direct channels pruning will cause performance collapse for subnetworks. To resolve such collapse, it is common to apply a finetuning process as in [11, 12, 34]. Despite being time-consuming especially on large datasets like ImageNet, the subnetwork after finetuning has not completely converged. To avoid the performance collapse and to have a justified evaluation of subnetworks, we use the technique of network *weight reconstruction* as in [14, 15, 25] to recover the remaining weights.

Specifically, given $F_{N\times C_{in}\times H\times W}$ is the input feature, $W_{C_{out}\times C_{in}\times K_1\times K_2}$ is the convolutional kernels. To save computational cost, we randomly select d patches $X_{N\times d\times C_{in}\times K_1\times K_2}$ from F, whose corresponding output feature is $Y_{N\times d\times C_{out}} = X \times W^T$. We then sample C'_{in} channels from X to have $X'_{N\times d\times C'_{in}\times K_1\times K_2}$, the weights of convolution kernel becomes $W'_{C_{out}\times C'_{in}\times K_1\times K_2}$, hence the output is $Y'_{N\times d\times C_{out}} = X' \times W'^T$. We seek to have $Y' = Y$ after channel selection. To do so, we have to update the weights W that minimizes the reconstruction error as defined in Equation 1.

$$W' = \arg\min_{W'} \sum_{i=1}^{m} \left(Y - X'W'^T\right)^2 \tag{1}$$

Such a linear regression problem can be solved by the common least-square method to find $W' = (X'^T X')^{-1} X'^T Y$. In addition, since the finetuning on the dataset is eliminated, the evolutionary speed can be greatly improved.

3.5 Pruning As Searching

We choose NSGA-III [5] as the evolutionary algorithm, instead of vanilla evolution for it has two following advantages. Firstly, it introduces a reference point of the hyperplane to maintain the diversity of the population in the evolution process, which is very important to find a set of "optimal network structures". Secondly, after the evolutionary process ends, we can sample multiple subnets that meet different constraints from the Pareto front. The knee-guided algorithm is also excluded [4.1], because its goal is to find a compromise in the "multi-objective", but we want to find different optimal solutions instead. Our whole pipeline can be viewed in Figure 2 and summarized in Algorithm 1.

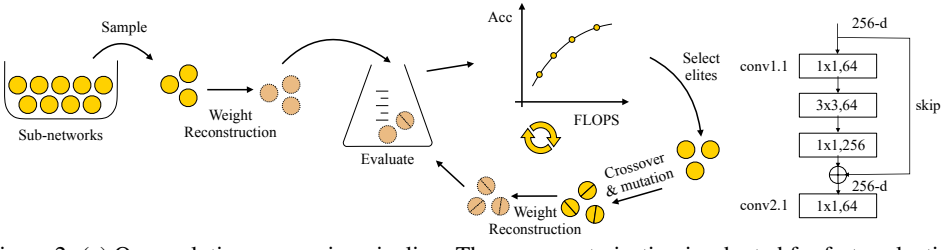

Figure 2: (a) Our evolutionary pruning pipeline. The reparameterization is adopted for fast evaluation within NSGA-III iterations. (b) Filters with channel dependency (*e.g.* conv1.1 and conv2.1) shall be pruned correspondingly.

Algorithm 1 Evolutionary Pruning Algorithm

Input: Original Network: $\mathcal{N}$, Pretrained Weights: $\mathcal{W}$, Population Size: $\mathcal{P}$, Number of Mutation: $\mathcal{M}$, Number of Crossover: $\mathcal{S}$, Max Number of Iterations: $\mathcal{T}$.
Output: $\mathcal{K}$ optimal Sub-Networks: $\mathcal{G}_\mathcal{K}$.

$\mathcal{G}_0$ = Random($\mathcal{N}$, $\mathcal{P}$);
for $i = 1 : \mathcal{T}$ **do**
  $\mathcal{G}_{metric}$ = Infer(Reconstruct($\mathcal{G}_{i-1}$, $\mathcal{W}$));
  $\mathcal{G}_i$ = NSGA-III.NextGen($\mathcal{G}_{metric}$, $\mathcal{M}$, $\mathcal{S}$);
end for
$\mathcal{G}_\mathcal{K}$ = ParetoFront($\mathcal{G}_\mathcal{T}$, $\mathcal{K}$);
return $\mathcal{G}_\mathcal{K}$;

4 Experiment

4.1 Models and Datasets

We conduct the experiments on a variety of networks, specifically, MobileNet-V1, ResNet50, and DeiT-Base. For pure CNN networks, we let every channel be prunable, i.e., for each layer

we select from a range of N channels with step 1. Note channel dependency is required, see Figure 2 (b). We directly perform evolutionary search on ImageNet, however only a small portion is used. We randomly select a set of images (1.5k for ResNet50 and DeiT-Base, 3k for MobileNet-V1) for weight reconstruction, and another 5k images for the evaluation of subnetworks' performance. We take images from the training set to avoid overfitting.

4.2 Evolutionary configurations

We evaluate nearly 1500 models per search space (takes 3.17 GPU hours on a A100 machine). Table 1 gives our configurations when performing evolutionary searching. We simply use a common set of hyper-parameters for three tasks (for ResNet50, it was due to an artifact of implementation, but the total number of models are ≈1.5k).

Search Space	Initial	Population	Iterations	Total
MobileNet-V1	64	50	30	1564
ResNet50	64	32	47	1568
DeiT-Base	64	50	30	1564

Table 1: Evolutionary configurations of selected search spaces.

4.3 Details for pruning DeiT-Base

Search space. Vision Transformer [7] is stacked with multiple transformer encoders. Each encoder is composed of a multi-head self-attention (MHSA) module and linear projection layers. Each has the same head number and head dimension. We choose to prune the popular DeiT-Base, whose prunable parts are a total of 4 linear projection layers: FC_{qkv}, FC_{proj}, FC_1, and FC_2. We follow the similar encoding paradigm as done in CNNs.

Weight reconstruction. Given input resolution is 224×224 and patch size 16×16, we have 197 tokens (including *cls*). To speed up reconstruction, we only choose 20 tokens which include *cls* token, and another 19 randomly sampled ones for weight reconstruction.

Finetune configuration. We use the same training hyper-parameters as DeiT-Base for finetuning, except that the number of epochs is reduced from 300 to 100, following VTP [42]. Our results are shown in Table 2. Though our performance is somewhat inferior compared with AutoFormer [3], EAPruning requires much less expertise and hyper-parameter tuning. Besides, AutoFormer is a NAS method for vision transformers only.

Model	FLOPs	Reduction	Top-1	Epochs	Training
DeiT-Base ([29])	17.8G	-	81.8%	300	Scratch
VTP ([42])	13.8G	22.4%	81.3%	100	Finetune
EAPruning (Ours)	13.5G	24.2%	81.3%	100	Finetune
AutoFormer ([3])	11.0G	38.2%	82.4%	500	Supernet
EAPruning (Ours)	11.0G	38.2%	81.6%	500	Scratch

Table 2: Pruning DeiT-Base on ImageNet, compared with state-of-the-art search-based methods.

4.4 Details for pruning ResNet50

Training configuration. According to [24], we retrain all pruned models from scratch with the same recipe, an initial learning rate of 0.2 with cosine annealing for 200 epochs, a batch size of 512, and we weight decay to 1e-4. Table 3 shows our finetuned model compared with others. Note MetaPruning [22] and ResRep [6] are for CNNs only. Figure 3 attests that finetuning and retraining give similar results (it is the structure that matters, not weight).

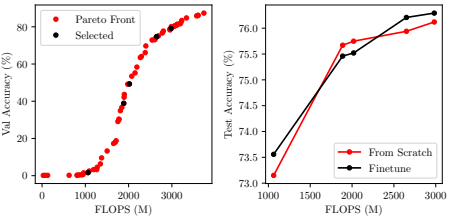

Figure 3: (a) Pareto front when pruning ResNet50 with the proposed method. (b) Finetuning selected models (at different prunning ratios) vs. training them from scratch, both give the similar results.

Model	FLOPs	Reduction	Top-1
ResNet50 [13]	4111M	-	76.0%
AMC [15]	2047M	50.3%	75.5%
NetAdapt [34]	2239M	45.6%	75.9%
MetaPruning [22]	2G	51.4%	75.4%
N2NSkip [28]	≈2G	50%	74.6%
ResRep [6]	≈1871M*	54.5%	76.2%
EAPruning (Ours)	2019M	50.9%	75.7%
OTO [3]	≈1418M*	65.5%	74.7%
EAPruning (Ours)	1554M	62.2%	74.8%
EAPruning (Ours)	1063M	74.1%	73.6%

Table 3: Pruned ResNet50 on ImageNet at 2G and 1G FLOPs level. *: estimated by reduction ratio.

4.5 Details for pruning MobileNetV1

Finetune configuration We select a subnetwork of the same scale as AMC [15] and finetune with the same strategy, i.e., an initial learning rate 0.05 with cosine annealing for 150 epochs, a batch size of 256, weight decay 4e-5.

Table 4 shows that our pruned network outperforms AMC by 0.6% improved accuracy while having the same FLOPs. We also surpass MetaPruning [22] with fewer FLOPs.

Model	FLOPs	Reduction	Top-1
MobileNetV1 ([17])	569M		70.6%
MobileNetV1 ([17])	325M	0.75×	68.4%
AMC ([15])	301M	0.89×	70.4%
NetAdapt ([34])	284M	1×	69.1%
MetaPruning ([22])	281M	1×	70.6%
MetaPruning ([22])	324M	0.75×	70.9%
EAPruning (Ours)	302M	0.88×	**71.1%**

Table 4: Comparison of pruned MobileNetV1 models on ImageNet.

4.6 Ablation study

NSGA-III vs. Random To verify the effectiveness of the evolutionary algorithm, we randomly sample the same number of subnetworks from the ResNet50's search space, going through weight reconstruction and evaluation likewise, to obtain a new Pareto front. As shown in the Figure 4 (a), NSGA-III's advantage is obvious.

l_1-norm vs. Random To choose which channels to keep, we conduct a control experiment: l_1-norm vs. random sampling. As shown in Figure 4 (b), the Pareto fronts obtained by the two methods almost overlap, indicating that both can find subnetwork structures with

the same performance. It is again assured that the network structure is more decisive for performance than weights.

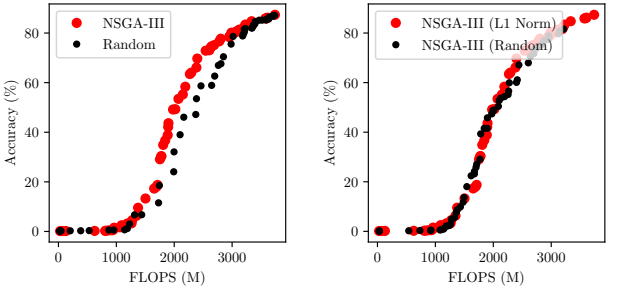

Figure 4: (a) Pareto front of NSGA-III vs. Random Searching (b) l_1-norm vs. random sampling.

Head number vs. dimension For DeiT-Base pruning, [37] concludes that each head is only responsible for a projection subspace. To cut off the subspaces that have not learned effective information, by default we fixed the subspace size (Dim_{Head}) in the MSA of each block, and only pruned the number of projection spaces (N_{Head}). We also carried out a comparative experiment, fixing N_{Head} in MSA and pruning Dim_{Head} to reduce the size of the subspace. It can be seen from Figure 5 that pruning N_{Head} is better than Dim_{Head}. The model of the same FLOPs is selected from the last generation Pareto boundary for finetuning, which also confirms the case, see Table 5. We can see that shortening the subspace size of the heads in the same encoder will damage those heads that have learned effective information [37].

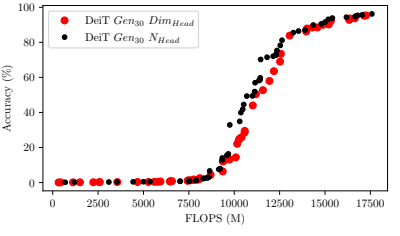

Figure 5: Pareto-front at the 30th generation when pruning DeiT-Base by head number and dimension.

Type	FLOPS	Top-1	Training
Head Num	12.5G	**80.93%**	finetune
Head Dim	12.5G	80.57%	finetune
Head Num	10.3G	**80.02%**	finetune
Head Dim	10.3G	79.74%	finetune
Head Num	13.5G	**81.45%**	from scratch
Head Num	13.5G	81.28%	finetune
Head Num	10.3G	**81.27%**	from scratch
Head Num	10.3G	80.02%	finetune

Table 5: Pruning DeiT-Base by head number vs. dimension, either finetuned or trained from scratch.

Finetuning vs. Training from scratch To verify that structure of the network is more important than weights on Vision Transformers, we retrained the pruned DeiT-Base model from scratch using the same hyper-parameters as [29]. The finetuning uses the same recipe, except that the epoch is set to 100. The results are shown in Table 5. After training, the accuracy of the 13.5 GFLOPs model trained from scratch is improved by 0.17% compared to finetuning, and the 10.3 GFLOPs model is improved by 1.25%.

Hardware speedup The acceleration on hardware is an important factor for pruning. We conducted experiments on NVIDIA A30 GPU which has FP32 Tensor cores. We set model input to 128×3×224×224, with FP32 precision, and we use TensorRT for deployment. The throughput of pruned CNN models has +30% increase while the accuracy loss is less than 0.5%. The throughput of DeiT is increased by 40% at the pruning ratio 40%, see Table 6.

Model	Throughputs (img/s)	Acc (%)	Speedup
ResNet50	3753		
ResNet50×0.5	5147	-0.30	**1.37**×
MobileNetV1	9176		
MobileNetV1×0.5	12296	-0.09	**1.34**×
DeiT-Base	777		
DeiT-Base×0.6	1086	-0.35	**1.40**×

Table 6: Our EA pruned models enjoys obvious speedup on NVIDIA A30 GPUs.

5 Importance of the Projection Layers in DeiT

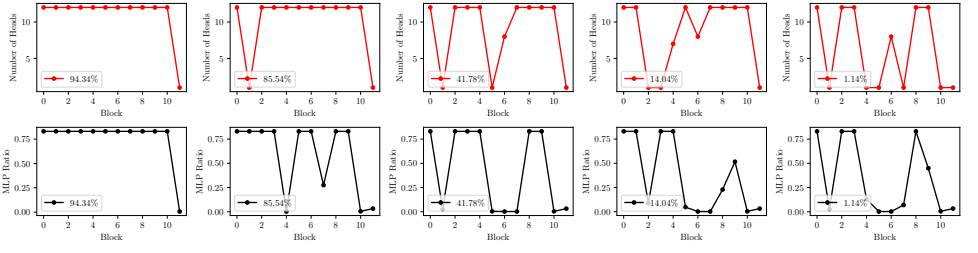

Figure 6: Head numbers and MLP ratios of EA pruned DeiT-Base selected from the Pareto front. The legend shows the top-1 accuracy on a reserved small dataset after weight reconstruction.

For DeiT-Base, we select networks with different pruning ratios from the Pareto front for structural analysis. Shown in Figure 6, EA prunes the last layer first, and the layer near the head next, forming a "cone" structure [33]. As the pruning rate further increases, it starts to prune the projection layer in the middle of the network, forming a "double-hump" structure. Hence we speculate that the importance of the projection layer of DeiT for ImageNet classification can be ranked as follows: last layer < beginning layers < middle layers < other layers. This observation coincides with [40], which finds that deeper layers in vision transformers develop similar attention maps and the performance saturates thereafter.

6 Conclusion

In this work, we have proposed a general pruning framework that is *effective for both Transformers and CNNs*. We seek to rejuvenate the evolutionary search as an efficient and practical pruning paradigm by reducing the search space and utilizing weight reconstruction for fast evaluation of subnetworks. We also take advantage of the multi-objective nature of the NSGA-III. The entire pruning process is made simple enough, requiring less effort to tune hyper-parameters or to involve extra components. This is in line with the "Occam's razor" principle. Our method is evaluated on a variety of networks to achieve abundant reduction and speedup.

References

[1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020.

[2] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12270–12280, 2021.

[3] Tianyi Chen, Bo Ji, Tianyu Ding, Biyi Fang, Guanyi Wang, Zhihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. In *NeurIPS*, volume 34, 2021.

[4] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.

[5] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4): 577–601, 2013.

[6] 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.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] Francisco E Fernandes Jr and Gary G Yen. Pruning deep convolutional neural networks architectures with evolution strategy. *Information Sciences*, 552:29–47, 2021.

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

[10] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493*, 2016.

[11] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[13] 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.

[14] 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.

[15] 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 *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[18] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[20] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

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

[22] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019.

[23] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017.

[24] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[25] 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.

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

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

[28] Arvind Subramaniam and Avinash Sharma. N2nskip: Learning highly sparse networks using neuron-to-neuron skip connections. In *BMVC*, 2022.

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[30] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2078–2087, 2020.

[31] Yunhe Wang, Chang Xu, Jiayan Qiu, Chao Xu, and Dacheng Tao. Towards evolutional compression. *arXiv preprint arXiv:1707.08005*, 2017.

[32] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29:2074–2082, 2016.

[33] Huanrui Yang, Hongxu Yin, Pavlo Molchanov, Hai Li, and Jan Kautz. Nvit: Vision transformer compression and parameter redistribution. *arXiv preprint arXiv:2110.04869*, 2021.

[34] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the ECCV (ECCV)*, pages 285–300, 2018.

[35] Hao Yu and Jianxin Wu. A unified pruning framework for vision transformers. *arXiv preprint arXiv:2111.15127*, 2021.

[36] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, pages 702–717. Springer, 2020.

[37] Qinglong Zhang and Yubin Yang. Rest: An efficient transformer for visual recognition. *arXiv preprint arXiv:2105.13677*, 2021.

[38] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[39] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n: M fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.

[40] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

[41] Yao Zhou, Gary G Yen, and Zhang Yi. A knee-guided evolutionary algorithm for compressing deep neural networks. *IEEE transactions on cybernetics*, 51(3):1626–1638, 2019.

[42] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021.