ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200422

MetaSelection: Metaheuristic Sub-Structure Selection for Neural Network Pruning Using Evolutionary Algorithm

Zixun Zhang [1 3] and Zhen Li † [2 3] and Lin Lin † [1] and Na Lei [1] and Guanbin Li [4] and Shuguang Cui [2 3]

Abstract. Neural network pruning is widely applied to various mobile applications. Previous pruning methods mainly leverage ad-hoc criteria to evaluate channel importance. In this paper, we propose an effective metaheuristic sub-structure selection (MetaSelection) method for neural network pruning. MetaSelection exploits evolutionary algorithm (EA) to search the proper sub-structure satisfying the resource constraints. In comparison with previous AutoML based methods, MetaSelection can automatically achieve the pruning rate and channel selection at the same time instead of hand-crafted criteria in a cascaded way. Regarding the tremendous search space of channel selection as a combinatorial optimization problem, we further utilize a coarse-to-fine strategy and the novel probability distribution crossover (PDC) to speed up the search procedure. Besides, MetaSelection prunes the network globally rather than in a layer-by-layer way. We evaluate MetaSelection on several appealing deep neural networks, achieving superior results with adaptive depth and width. Concretely, on ImageNet, MetaSelection achieves a top-1 accuracy of 71.5% on MobileNetV2 under 70% FLOPs constraint and a FLOPs reduction of 30% with 76.4% top-1 accuracy for ResNet-50.

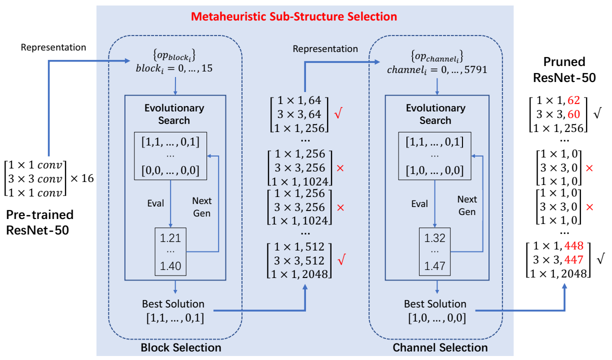
Metaheuristic Sub-Structure Selection
Representation
$\{op_{block_i}\}$ $block_i = 0, \dots, 15$
Evolutionary Search
[1,1, …, 0,1] … [0,0, …, 0,0]
Eval
Next Gen
1.21 … 1.40
Best Solution [1,1, …, 0,1]
Block Selection
$\begin{bmatrix}1\times1\ conv\\3\times3\ conv\\1\times1\ conv\end{bmatrix}\times16$
Pre-trained ResNet-50
Representation
$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}$ √
$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}$ ×
$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}$ √
$\{op_{channel_i}\}$ $channel_i = 0, \dots, 5791$
Evolutionary Search
[1,1, …, 0,1] … [1,0, …, 0,0]
1.32 … 1.47
Best Solution [1,0, …, 0,0]
Channel Selection
Pruned ResNet-50
$\begin{bmatrix}1\times1,62\\3\times3,60\\1\times1,256\end{bmatrix}$ √
$\begin{bmatrix}1\times1,0\\3\times3,0\\1\times1,0\end{bmatrix}$ ×
$\begin{bmatrix}1\times1,448\\3\times3,447\\1\times1,2048\end{bmatrix}$ √

Figure 1. Overview of our proposed MetaSelection model for network pruning on ResNet-50. MetaSelection represents the network by a set of operations and firstly selects the proper subset of blocks from the original residual block set, then searches for the detailed width of each block iteratively. The entire search process is completed using evolutionary algorithm (EA) without any manual metrics. Finally, we get a pruned ResNet-50 both shallower and thinner.

1 Introduction

Deep neural networks have achieved remarkable performance in a variety of computer vision tasks, e.g., image classification [9], object detection [7] and so on. However, these networks with extraordinary capacity are more resource-hungry, which significantly restricts the deploy for mobile applications, e.g., embedded sensors or portable devices with limited computational and power resources. Recently, neural network pruning [19, 21, 12, 24] has been widely studied as an efficient way to compress or accelerate deep neural networks.

Network pruning methods mainly belong to two branches. The first branch is based on handcrafted metrics, e.g., L2-norm [19], sparsity regularization [21, 16], and percentage of zero activations [14]. However, these methods require large amount of human heuristics and expertise for proper ad-hoc criteria. Another stream uses AutoML based methods to automatically prune channels, e.g., exploiting reinforcement learning [12] or progressive barrier method [30] for networks pruning. However, these AutoML methods only determine the pruning rate of each layer voluntarily while the channel selection is still achieved by aforementioned handcrafted matrices. Besides, such methods usually prune the channels layer-by-layer with a greedy strategy without globally considering all channels, thus leading to the sub-optimal performance [31]. Moreover, Neural Architecture Search (NAS) [27] is also leveraged to search for compact networks, but it requires a well-designed search space and only obtains a discrete space for channels quantity (e.g. $\{16, 24, 32, 64\}$ for channels nums). [20] directly find the good pruning structure with weights generation using a meta-learning based PruningNet, under the assumption of optimal pruning structure is most essential [22]. On the contrary, [6] states the subnetworks (lottery ticket) with inherited weights makes pruning effectively. We also conduct neural network pruning as a subnetworks selection problem for appealing models with weights under the hypothesis similar to [6].

In this paper, we reformulate the neural network pruning as a sub-structure selection problem. Representing the network as a set of operations, we can determine the pruning rates of all layers and select the proper operations simultaneously in a global fashion. Due to the high computational complexity of combinatorial optimization problem, we introduce an evolutionary algorithm (EA) to solve this NP-hard problem. Nevertheless, the search space is still enormous for EA, e.g., in Fig. 1, there are 2^{7552} possible combinations for ResNet-50, which motivates us to propose another two modifications for further search acceleration. With the introduction of skip connections, the layer-wise redundancy in multi-branch networks has been pointed out in various works [28, 15]. Thus a coarse-to-fine pruning strategy is leveraged to significantly reduce the search space. Specif-

[1] Dalian University of Technology, China, email: zixun.zhang@mail.dlut.edu.cn; {lin, nalei}@dlut.edu.cn; †: Corresponding author
[2] The Chinese University of Hong Kong (Shenzhen), China, email: {lizhen, shuguangcui}@cuhk.edu.cn; †: Lead corresponding author
[3] Shenzhen Research Institute of Big Data, China
[4] Sun Yat-sen University, China, email: liguanbin@mail.sysu.edu.cn

ically, in Fig. 1, after removing 3 blocks from original ResNet-50 using coarse pruning, the search space reduces from 2^{7552} to 2^{5792} $(5792 = 64 \times 2 \times 3 + 128 \times 2 \times 3 + 256 \times 2 \times 4 + 512 \times 2 \times 3)$. Besides, we exploit a recombination strategy called probability distribution crossover (PDC) to replace the standard crossover in EA. Experiments results and ablation study show that the PDC strategy significantly accelerate the evolutionary process. Fig. 1 illustrates the teaser of our proposed MetaSelection model. Moreover, MetaSelection automatically searches for the proper pruned structure globally rather than pruning layer-by-layer greedily through manual criteria. We evaluate our approach on CIFAR-10 [17] and ImageNet [5] using several appealing deep neural networks. Under the same FLOPs constrains, our algorithm achieves higher accuracy than the uniform baselines on ResNet [10], MobileNetV1 [13] and MobileNetV2 [26]. Compared with state-of-the-art AutoML channel pruning methods, our algorithm produces superior or comparable results and leads to more flexible pruning strategy. Compared with regularization based pruning method, our algorithm also achieves superior performance. In some instances, MetaSelection achieves a FLOPs reduction of 30% on ResNet-50 with 76.4% top-1 accuracy on ImageNet, and 70.6% and 71.5% top-1 accuracy under 0.75x resource constraint on MobileNetV1 and MobileNetV2 correspondingly.

The main contributions of this paper are in three folds:

1. We propose an efficient AutoML method for network pruning, called MetaSelection. Particularly, we reformulate network pruning as a subset selection problem, and introduce an evolutionary algorithm to address it. Specifically, we can determine the pruning rate of each layer and select the proper sub-structure simultaneously rather than using handcrafted metrics or greedily pruning layer-by-layer.
2. Considering high computational complexity of combinatorial optimization problem, we utilize a coarse-to-fine pruning strategy to significantly reduce the search space. Besides, a more effective recombination strategy called probability distribution crossover (PDC) is used to further accelerate the evolutionary search.
3. Experiments of pruning various appealing deep neural networks on both CIFAR and ImageNet dataset demonstrate the effectiveness and superiority of our MetaSelection.

2 Related work

2.1 Criteria Based Network Pruning

Network pruning was first proposed by [18] and [8] to remove unimportant connections and neurons. While weight pruning leads to unstructural sparsity and needs extra hardware or library to accelerate inference, recent works mainly focus on structural pruning (e.g., channel pruning or block pruning). Traditional channel pruning methods use handcrafted criteria to evaluate the importance of channels, such as L2-norm [19], percentage of zero activations [14], energy aware [24] and etc., and then prune channels according to the importance. These methods require to design proper criteria and the pruning rate which needs large amount of domain expertise and human heuristics. Other methods introduce L1 regularization on scaling parameters of batch normalization [21] or extra scaling factors [16] and training with different sparsity weight to achieve different pruning rate. These handcrafted criteria based pruning methods are time consuming and usually sub-optimal. Compared with this kind of handcrafted pruning methods, our proposed MetaSelection can automatically figure out the unimportant channels via EA with few manual efforts.

2.2 AutoML Based Network Pruning

Recently, some AutoML based pruning methods were proposed to tackle the drawbacks of handcrafted pruning methods. AMC [12] leverages reinforcement learning to search for the pruning rate of each layer. NetAdapt [30] uses a progressive barrier method to iteratively prune the network. However, these methods only automatically determine the pruning rate of each layer and still use handcrafted criteria to select channels. What's worse, they use a layer-by-layer pruning strategy which has been proved to be problematic [31]. Most recently, MetaPruning [20] was proposed using MetaLearning to predict the vicarious weight according to a given pruning rate, which bypasses the channel selection problem. Our proposed MetaSelection also aims to alleviate the manual efforts and attemps to solve the automatical channel selection problem. Compared with AMC [12] and NetAdapt [30], our MetaSelection automatically figures out the pruning rate meanwhile selects the proper channels. Moreover, MetaSelection has a more flexible pruning strategy to adaptively adjust depth and width.

2.3 Neural Architecture Search

Searching for optimal neural architecture has attracted increasing attention recently. One stream is to explore the design space by reinforcement learning [1, 32] or evolutionary algorithm [25, 29]. Another stream is to build a supernet with multiple operation choices in each layer then search for the path with the highest accuracy after training, namely one-shot NAS [2, 3]. Different from NAS using discrete channel numbers as search space, the width of each layer is usually consecutive in channel pruning, which leads to an explosion of the search space. MetaSelection addresses the channel pruning problem from the view of sub-structure selection, inherently supporting the continuous search space for channel selection.

3 Methodology

In this section, we propose a pruning method, MetaSelection, that automatically prunes a network to meet resource constraints while remaining high accuracy. Different from the prior AutoML pruning methods, our algorithm determines the pruning rate[5] for each layer globally and selects channels automatically rather than leveraging local information only or using handcraft metrics.

3.1 Problem Formulation

Similar to [30, 20], we first formulate the network pruning problem as follow:

$$\begin{aligned} \max_{\text{Net}} \quad & Acc(Net) \\ \text{s.t.} \quad & Reso(Net) \leq Budget \end{aligned} \tag{1}$$

where Net is a compact network pruned from the original network, Acc is the accuracy of validation dataset, $Reso$ computes the computational resources that the network requires while $Budget$ refers to the predefined resource constrain. Such resource budget can be FLOPs, number of parameters, MACs, latency and etc.

In order to automatically determine the pruning rate and select channels at the same time, we rethink the pruning problem as a

[5] Pruning rate of one layer is defined as the ratio of the number of discarded channels against the number of original channels.

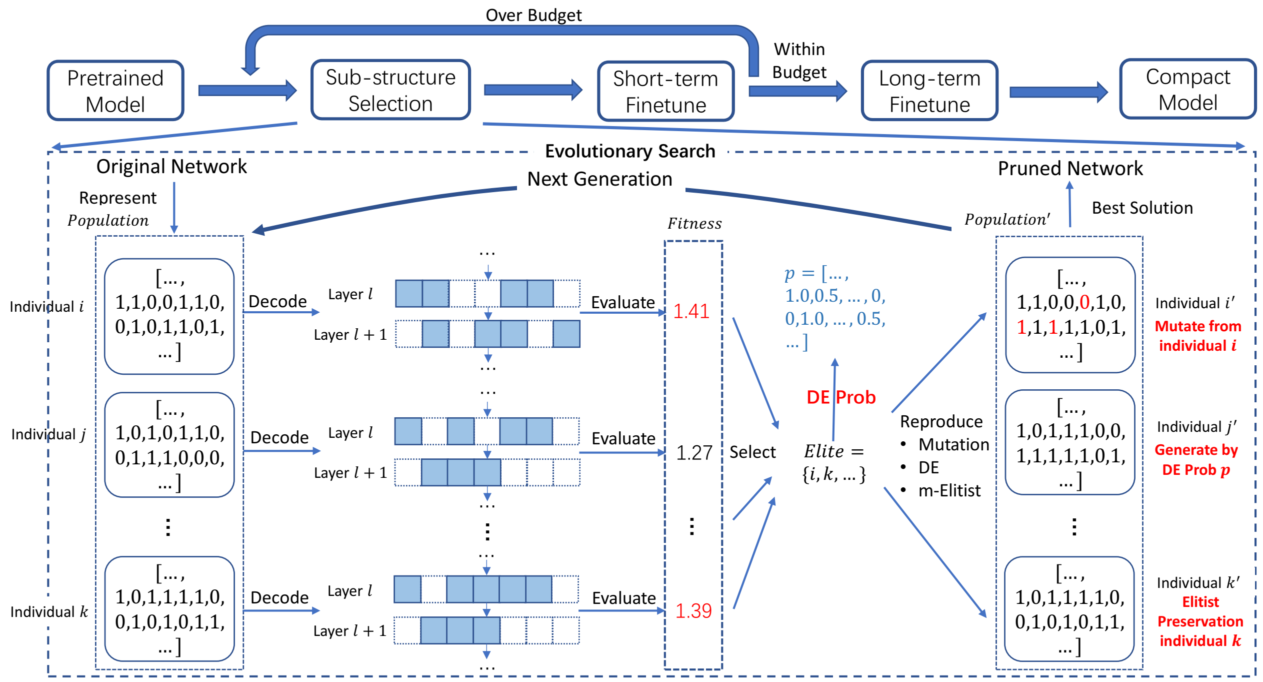

Figure 2. The pipeline of our proposed MetaSelection. At each iteration, a sub-structure is searched via evolutionary algorithm for block/channel selection. If the constraint is not satisfied, we only finetune the pruned network for several epochs to regain the accuracy and continue to prune it. Once the selected sub-structure meets the constraint, it will be finetuned until convergence (i.e., long-term finetune) and returned as the final pruned network.

subset selection problem. Given a network, we can represent it by a set of operations $OP = \{op_1, ..., op_n\}$. The operations can be defined from different scale. At a fine scale, each operation is corresponding to a channel (for CNNs) or a neuron (for MLPs); while at a coarse scale, the operation can be a branch or transformation block like residual block. Considering pruning a network is to reduce the size of the network subject to resource constraints, it can be regarded as selecting a proper subset of the original operations. Thus, this problem can be reformulated as follows:

$$\max_{S \subseteq OP} \quad Acc(Net(S)) \qquad \text{s.t.} \quad Reso(Net(S)) \leq Budget \tag{2}$$

where S is a subset selected from OP, and $Net(S)$ is the corresponding sub-network only retaining the selected operations according to S. Then we can apply an evolutionary algorithm to search for proper pruned network.

3.2 MetaSelection

The pipeline of our proposed MetaSelection is presented in Fig. 2. Taking original networks as inputs, a sub-structure is achieved by an evolutionary algorithm with a soft constraint. If the pruned network does not meet the resource constraint, it will be fine-tuned for only a few epochs to regain the accuracy (**Short-term Finetune**) and then sent back to **Sub-structure Selection** for further compression.

Once the resource constraint is satisfied, the pruned network will be fine-tuned until convergence (**Long-term Finetune**). Besides, for multi-branch networks like ResNet and MobileNetV2, a coarse-to-fine pruning strategy is leveraged to reduce the following channel search space. That is to say, we first search the sub-structure at block/branch level for one step, and then search for the proper width of each layer.

3.2.1 Sub-structure Selection

This process is used for searching proper sub-structure via evolutionary algorithm. It takes a pre-trained network as input and evolves for the proper sub-structure. In practice, we use the following schemes in our evolutionary search process.

1. **Representation:** Each sub-structure is encoded into a vector $\overrightarrow{op}$ of $\{0, 1\}$, where each bit represents the state of the operation. Here 1 denotes the operation is selected. This vector is the chromosome or gene of the individual.
2. **Fitness evaluation:** To evaluate individuals, we define the fitness as follow:

$$Fitness = \frac{Acc(Net_{ind})}{Acc(Net_{ori})} + \alpha\sqrt{1 - \frac{Reso(Net_{ind})}{Reso(Net_{ori})}} \tag{3}$$

where Net_{ind} is an individual from the population, and Net_{ori} is the original full network. Similar to training with sparsity regularization [21, 16], we add the resource constraint as a soft constraint penalty term to the fitness, and use a hyper-parameter α to achieve the trade-off between the accuracy and resource constraint. To make the training process more stable, we normalize these two terms to $[0, 1]$ with the corresponding values of the original network. Note that we directly use the validation accuracy of the pruned model without fine-tuning as final performance since it is an efficient delegate of the fine-tuned accuracy [12].

3. **Selection and reproduction:** In each generation, the *Top K* individuals with highest fitness in the parent population will be selected to reproduce offspring. The reproduction is achieved through mutation, m-Elitist and probability distribution crossover (PDC). The mutation is to randomly reverse several bits of the chromosome. The m-Elitist stands for directly inheriting the elite individual from parents. Instead of using standard crossover which recombines the chromosomes of two parents to generate new offspring, we propose a novel recombination scheme called **probability distribution crossover (PDC)**. PDC is carried out by calculating the weighted mean of the *Top K* individuals for each bit, then generating new individuals using this value as the probability p. The probability p is calculated as follows:

$$p_i = \frac{\sum_{j \in TopK} Fitness_j \times \overrightarrow{op}_i}{\sum_{j \in TopK} Fitness_j}, i = 0, ..., |\overrightarrow{op}| \qquad (4)$$

Standard crossover only leverages information of two parents and has the tendency for more homogeneity with small populations [4]. Our proposed PDC explores the information of elite individuals from the population, and generates new individuals accordingly, which is much more efficient than standard crossover. Ablation study further verifies the superiority of PDC.

After the evolution finished, the individual with the highest fitness in the population is returned as the pruning solution of this step. The evolutionary algorithm for sub-structure selection is presented in Algorithm 1.

Algorithm 1 Sub-structure Selection Using EA

```
Require: Pretrained model $PM$; Max generation $N$; Population
    size $ps$ ($ps = K + m + n$); Elite size $K$; Mutation size $m$;
    Probability distribution crossover size $n$;
Ensure: Pruning solution with best fitness $best$;
 1: Initialize population $S_0$;
 2: Evaluate population($PM$, $S_0$);
 3: for $i = 0 : N$ do
 4:     $elites = TopK(S_i)$;
 5:     $S_{mut} = Mutation(elite, m)$;
 6:     $S_{pdc} = Probabilitydistributioncrossover(elite, n)$;
 7:     $S_{i+1} = (elites, S_{mut}, S_{pdc})$;
 8:     Evaluate population($PM$, $S_{i+1}$);
 9: end for
10: $best = Top1(S_N)$;
11: return $best$;
```

3.2.2 Short/Long-term Finetune

Before the solution candidate meets the resource constraint, we finetune the pruned network for few epochs to regain accuracy for further search (short-term finetune correspondingly). Once the final pruned network is obtained, we finetune the final pruned network until convergence (long-term finetune corresondingly).

4 Experiments

In this section, we evaluate our MetaSelection on two standard datasets: CIFAR [17] and ImageNet LSVRC-2012 [5]. We compare MetaSelction with several state-of-the-art AutoML based pruning methods [12, 30, 20] and handcraft policy based pruning methods [21, 16]. Here we adopt FLOPs as the resource constraint, and it is easy to expend to other constraint like latency.

Meta-parameter for EA There are several meta-parameter for EA, as described in alg. 1. For the elite size K, we set it to 15, and set both mutation size m and probability distribution crossover size n to 25. For block selection, the search space is relatively small, only about $O(2^{20})$ - $O(2^{50})$, so we set the maximum generation to 20. While for channel selection, the search space is tremendous, e.g. $O(2^{7552})$ on ResNet-50 without block pruning on ImageNet, so we set the maximum generation to 30. We discuss more details about the meta-parameter in the analysis section.

4.1 Results on CIFAR-10

Model	Method	Param ↓	FLOPs ↓	Acc
ResNet-56	AMC[12]	-	50.0%	91.9%
	Ours	**70.1%**	**63.4%**	**92.4%**
ResNet-164	NetSlim[21]	35.2%	44.9%	**94.7%**
	SSS[16]	17.7%	48.0%	94.1%
	Ours	**41.6%**	**56.4%**	**94.7%**

Table 1. Results on CIFAR-10. Param ↓ and FLOPs ↓ denote the reduction of parameters and FLOPs, while Acc represents the accuracy. Our method achieves higher accuracy with more reductions under the resource constraint.

Setting For CIFAR-10, we test our MetaSelection on ResNet-56 and ResNet-164 [10]. ResNet-56 has 27 residual blocks each of which consists of two convolutional layers, while ResNet-164 has 54 bottleneck residual blocks consisting three convolutional layers. We first apply block pruning on both network to obtain a shallower network, then apply channel pruning on the shallower network to get a more compact one. For sub-structure selection, we split 5000 images from training set as validation dataset. For short-term finetuning, we train the pruned network for 10 epochs with a constant learning rate of 0.001. While for long-term fintuning, we train the pruned network on the whole training set for 50 epochs with a cosine learning rate starting from 0.01.

Results Table 1 shows the results of our algorithm on CIFAR-10. On ResNet-56, our algorithm achieves a FLOPs reduction of 63.4% with the accuracy of 92.4%. On ResNet-164, our algorithm achieves a parameter reduction of 41.6% and a FLOPs reduction of 56.4% with the accuracy of 94.7%. For block selection, our MetaSelection prunes 17 blocks on ResNet-164, and 9 blocks on ResNet-56 respectively. During the block selection, we found that blocks in early stages are pruned the most, which is similar to the observation made by SSS [16]. Compared with NetSlim[21], SSS[16] and AMC[12], our algorithm achieves both higher FLOPs reduction and accuracy on both ResNet-56 and ResNet-164.

4.2 Results on ImageNet under FLOPs constraint

Setting For ImageNet ILSVRC-2012 dataset, we test our algorithm on two lightweight network, MobileNetV1 [13] and MobileNetv2 [26], and a heavyweight one ResNet-50 [10]. As MobileNetV2 and ResNet-50 are multi-branch networks, we apply both branch pruning and channel pruning to them. We randomly select 25 images of each class from the training set as extra validation dataset for sub-structure search. Similar to the setting on CIFAR-10, after each sub-structure selection step, we finetune the pruned network for 2 or 3 epochs with a constant learning rate of 0.0001 (short-term finetune). Once we obtain the final pruned network which meets the resource constraint, we finetune it for 50 epochs on the whole training dataset with a cosine learning rate starting from 0.01 (long-term finetune). Table 2 shows the results on ImageNet.

Model	Method	FLOPs	Top-1 Acc
MobileNetV1	0.75x[13]	325M	68.4%
	AMC[6][12]	294M	70.5%
	NetAdapt[30]	284M	69.1%
	MetaPruning[20]	281M	**70.6%**
	Ours(F)	285M	**70.6%**
MobileNetV2	0.75x[26]	220M	69.4%
	AMC[12]	220M	70.8%
	MetaPruning[20]	217M	71.2%
	Ours(CF)	216M	71.0%
	Ours(F)	220M	**71.5%**
ResNet-50	SSS[16]	3.5B	75.4%
	Ours(C)	3.4B	**76.6%**
	SSS[16]	2.8B	74.2%
	SFP[11]	2.9B	75.1%
	ThiNet-70[23]	2.9B	75.8%
	MetaPruning[20]	3.0B	76.2%
	Ours(CF)	2.8B	**76.4%**

Table 2. Results on ImageNet under FLOPs constraint. Here, M/B in FLOPs column means million/billion ($10^6/10^9$), respectively. For results on MobileNet, **0.75x** denotes the uniform baseline in this paper. **(C)** denotes result only applying coarse-grained block pruning, **(F)** represents the result applying only fine-grained channel pruning, and so on.

ResNet-50 For original ResNet-50, our algorithm firstly achieves a FLOPs reduction of 16.1% by block selection with a top-1 accuracy of 76.6%. Compared to SSS[16] leveraging L1-norm to prune blocks, we achieve a higher top-1 accuracy under the same FLOPs reduction. Different from the observation on CIFAR-10 dataset, we find that blocks in the third stage are pruned the most, which is also quite different from the results of SSS[16]. We further apply the channel pruning on the pruned ResNet-50 and ahieve a FLOPs reduction of 31.7% with only 0.2% top-1 accuracy drop. In contrast to SSS[16] under the same FLOPs, our algorithm still achieve a higher top-1 accuracy. Note that our algorithm achieves a better trade-off between width and depth on ResNet-50 rather than only a shallower one. Figure 3 shows the width of each block in our pruned ResNet-50. During the channel pruning, we find that channels in the last two stages are more likely to be pruned than earlier stages. Compared with other pruning method, SFP[11], ThiNet[23] and MetaPruning[20], our MetaSelection achieves a higher top-1 accuracy under the similar FLOPs reduction.

MobileNetV1 MobileNetV1 is a single branch network, so we only apply channel selection on MobileNetV1. Due to the use of separable convolution, we only need to select the 1×1 convolution, the corresponding depthwise convolution will be removed as well. Under the 50% FLOPs resource constraint, the original uniform 0.75x baseline has a top-1 accuracy of 68.4%, our pruned MobileNetV1 achieves 70.6% top-1 accuracy. Compared to other AutoML pruning methods, AMC[12], NetAdapt[30], MetaPruning[20], our MetaSelection also produces superior or comparable results.

[6] We report the result of AMC on MobileNetV1 according to the released model on AMC project page.

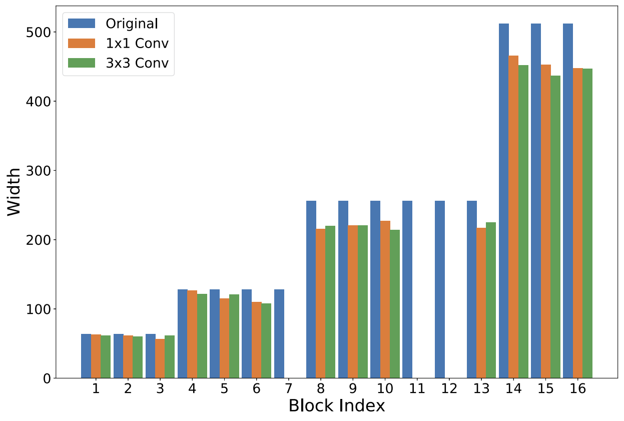

Figure 3. This figure shows the widths of residual blocks in our pruned ResNet-50. Original ResNet-50 consists of 16 residual blocks with bottleneck (1×1 conv – 3×3 conv – 1×1 conv). For channel selection, we only prune the intermediate layers of the residual block (the first two convolutional layers). The 7th, 11th, 12th blocks are removed during the block pruning, so the widths of these blocks are 0.

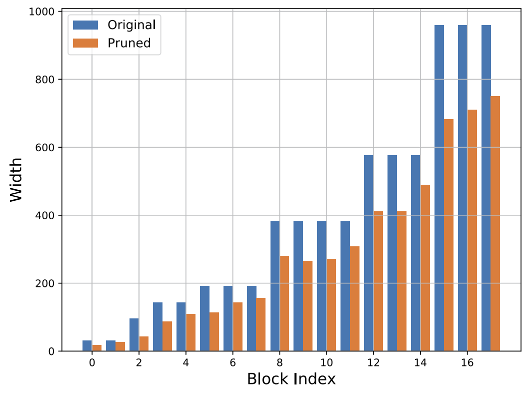

Figure 4. This figure shows the widths of the intermediate layers in our pruned MobileNetV2. Original MobileNetV2 consists of 17 blocks, while 0-th denotes the first convolutional layer in the network.

MobileNetV2 MobileNetV2 uses inverted residual block as building block, thus, we first apply block selection to compress the depth of MobileNetV2. Under the 70% FLOPs constraint, our MetaSelection achieves a top-1 accuracy of 71.0% with both block and channel pruning. During the search process, only one block is pruned after the block selection, which we conjecture that MobileNetV2 is more sensitive to backbone reduction than ResNet due to the use of inverted residual block and separable convolution (MobileNetV2 only contains 3.4M trainable parameters vs 25.6M of ResNet-50). So we further prune MobileNetV2 with only channel pruning, and achieve a top-1 of 71.5% under the 70% FLOPs constraint. During the channel selection, we find that channels in the downsampling blocks are preserverd more than other inverted residual blocks. Figure 4 shows the width of each block in our pruned MobileNetV2. Compared to AMC[12] and MetaPruning[20], our MetaSelection produces superior or comparable results.

4.3 Results on ImageNet under latency constraint

Setting In this experiment, we apply our algorithm on MobileNetV2 under the latency constraint. We directly measure the latency on the targeted I5-8400 CPU used in the fitness evaluation rather than using a look-up table, other settings are similar with experiments under FLOPs constraint.

Results Table 3 shows the results on ImageNet under latency constraint. As a matter of experience, there is always a gap between theoretical (FLOPs) and realistic (latency) speedup. Thus, we directly take latency as the resource constraint. For simplicity, we only apply block pruning and directly measure the latency on CPU with a batch size of 32. Our MetaSelection prunes 5 blocks and 7 blocks respectively to meet the latency constraint and remains 69.8% and 65.3% top-1 accuracy. Under the same latency constraint, our algorithm achieves a higher top-1 accuracy.

Model	CPU Lat. (ms)	GPU Lat. (ms)	Top-1 Acc
0.75x	16.53	0.67	69.4%
Ours	15.46	0.65	**69.8%**
0.5x	10.13	0.58	64.4%
Ours	10.59	0.61	**65.3%**

Table 3. Results of MobileNetV2 on ImageNet under latency constraint. 0.75x and 0.5x MobileNetV2 is the uniform baseline in our implementation. The latency is measured on I5-8400 CPU and TITAN XP GPU with a batch size of 32.

4.4 Analysis

Effect of elite size In our MetaSelection, we use a Top-K selection strategy and reproduce offsprings using the information of parents. The quality of offsprings is affected by the size of parents (elite size), which would affect the search efficiency. Hence, we conduct several experiments with different values of K on CIFAR-100 on a modified VGG-11, and record the best fitness after evolution (Figure 5, left). All populations evolved for the same generation and explored the same amount of new individuals at each step. As shown in the figure, elite size of 15 is the most efficient under the constraint of total exploration amount.

Effect of population size In EA, larger populations could explore the search space more thoroughly, which helps EA reach better optima. However, the exploring direction is updated at each step (the quality of parents), we need to figure out the most effective population size under the same exploration amount (Figure 5, right). As shown in the figure, exploration size of 25 is the most efficient value. Although populations have the similar performance while exploration size is over 25, however, under the constraint of total exploration amount larger population also has a lower efficiency.

Sensitivity of α In the search process, we use a hyper-parameter α, which refers to the step size of each sub-structure selection to balance the accuracy loss and resource constraint. We conduct several experiments on CIFAR-10 to figure out the sensitivity of α with different values on a plain CNN with 15 layers (each layer consists of 32 channels). As shown in figure 6, it can be observed that with the increase of α, the remaining FLOPs and accuracy reduce linearly. In this paper, we use the value of 1.0 for all the experiments.

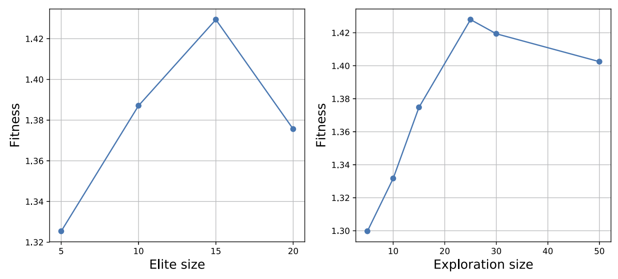

Figure 5. Dependence on elite size and exploration size. All the experiments are conducted on a modified VGG-11 on CIFAR-100. All the populations explore for the same amount of new individuals, which is set to 1500. For the experiments on elite size K, we set mutation size and PDC size $n = m = 15$ and maximum generation $N = 50$. For the experiments on exploration size (value of n and m), we set elite size $K = 15$ and maximum generation $N = \frac{1500}{n+m}$.

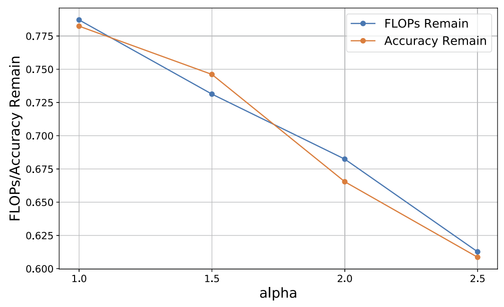

Figure 6. Sensitivity of α. The remaining FLOPs and accuracy without finetuning using different trade-off of α. α satisfies nice monotonicity property (larger value leads to larger compression step size).

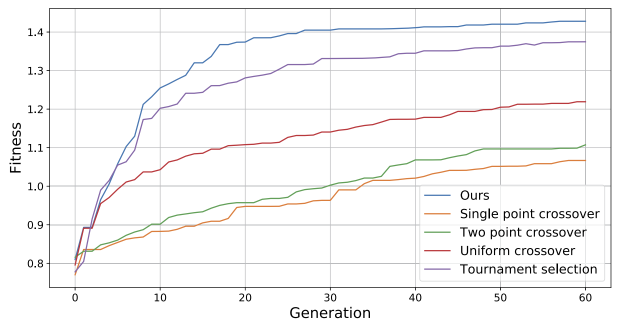

Figure 7. Comparison between different reproduction strategies. All the experiments are conducted on a modified VGG-11 on CIFAR-100. Our proposed PDC has a significant advantage in speed of convergence and performance.

Comparison with other reproduction strategies In this paper, we propose probability distribution crossover (PDC) to replace the normal crossover. We conduct several experiments to figure out the effectiveness of our PDC. In addition, we only use the elite set to generate new offsprings at each step, it may lead to some local minima. Thus, we also conduct an experiment to compare the performance between our Top-K selection and tournament selection. In summary, we conduct several ablation experiments to compare the performance of following reproduction strategies to figure out the effectiveness of ours:

1. Original strategies used in our EA search;
2. Replacing PDC with single-point crossover;
3. Replacing PDC with two-point crossover;
4. Replacing PDC with uniform crossover;
5. Replacing Top-K selection with tournament selection.

The results are shown in figure 7. Under the same constraint of total exploration amount, our reproduction strategy is the most efficient one. Compared with standard crossover, our PDC has a faster speed of convergence. As our observation, standard crossover only uses the information of two parents and more likely to generate a repeated individual at later generations, while PDC combines all the information of elites and uses probability to generate new individual which has a higher ability of exploring new individuals. In addition, PDC needs the information of elites to measure the exploration direction, which takes benefits from Top-K selection, so it is easy to explain the superiority of our Top-K selection over tournament selection.

Diff.	Constraint	Mean Acc.	Variance
Easy	Ori.	84.08%	0.006
	HC	33.68%	0.051
	SC	51.62%	0.036
Hard	Ori.	55.73%	0.015
	HC	19.49%	0.028
	SC	32.79%	0.030

Table 4. Accuracy changes of 1000 classes after pruning with soft/hard constraint of 75% FLOPs without long-term finetuning on MobileNetV1. The classes is divided into two difficulty levels. Ori. denotes the original performance without pruning, HC/SC means pruning under hard/soft constraint, respectively. The generalization ability on easy classes becomes worse and unstable under hard constraint.

One step with hard constraint or iteratively with soft constraint With the flexibility of evolutionary algorithm, there are two ways to deal with the resource constraint. The first way is to obtain the qualified result with hard constraint (HC) in one-step, while the second way is to progressively prune the network with soft constraint (SC) until meeting the resource constraint. Therefore, we carry out several experiments to analyse such two solutions. We first prune the MobileNetV1 with a resource constraint of 75% FLOPs. The original 1000 classes are divided into two difficulty level according to the accuracy (608 easy classes: accuracy over 75%; 392 hard classes: others). We further observe the accuracy changes after sub-structure search. As shown in table 4, the accuracies become to be imbalanced under HC, which means the pruned network begins to lose its generalization capacity. Although the accuracies under SC also appear to be imbalanced, the overall variance is much less than that under HC on easy classes. The final accuracy after finetuning also shows the same phenomenon. However, such a problem reduces a lot on a heavy model like ResNet-50. After pruning with a HC of 75% FLOPs, the pruned ResNet-50 still has a top-1 accuracy of 56.7% without finetuning. And the finetuned top-1 accuracies of SC and HC only have a margin of 0.2%. We conjecture that lightweight networks like MobileNet have higher utilization of parameters which are more sensitive to the reduction. Thus, for stability we progressively prune the network with soft constraint and finetune for a few epochs after each search step.

Stability of EA process In our MetaSelection, the sub-structure selection is based on EA, which is a stochastic process, so the results may vary every execution. Thus, we test the stability of the EA process on ResNet164 and PlainCNN on CIFAR-10 for block/channel selection. All the experiments are repeated 5 times with different random seeds, start with the same pre-trained model and prune for one step with soft resource constraints. The results are shown in table 5. The results show the stability of our method.

Model	FLOPs Reduction	Accuracy
PlainNet	13.07% (0.705)	90.33% (0.048)
ResNet-164	27.78% (1.372)	94.87% (0.014)

Table 5. Stability experiments for EA process. Each experiment runs for 5 times and reports the mean and variance of FLOPs reduction and finetuned accuracy.

5 Conclusion and future work

In this paper, we propose an AutoML pruning method, MetaSelection, to adaptively prune the neural networks. In our method, we reformulate the network pruning problem from the view of subset selection. To solve this combinatorial optimization problem, we introduce an evolutionary algorithm. Different from prior AutoML pruning methods, we simultaneously select the pruning rate and channels globally. In addition, to tackle the high computational complexity, we propose a coarse-to-fine pruning strategy and a modified recombination strategy to further search acceleration. We demonstrate promising results on both heavy and lightweight networks through experiments.

In the future, we plan to combine our MetaSelection with other compression algorithm, e.g., low-precision weights and quantification, to obtain a higher compression performance. What's more, it is also worth exploring to apply our MetaSelection in other applications rather than image classification, such as object detection and image segmentation.

6 Acknowledge

The work was supported by grants No. 2018YFB1800800, NSFC-61902335, No. 2019E0012, No. ZDSYS201707251409055, No. 2017ZT07X152, No. 2018B030338001, and CCF-Tencent Open Fund.

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar, 'Designing neural network architectures using reinforcement learning', *arXiv preprint arXiv:1611.02167*, (2016).

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le, 'Understanding and simplifying one-shot architecture search', in *International Conference on Machine Learning*, pp. 549–558, (2018).

[3] Han Cai, Ligeng Zhu, and Song Han, 'Proxylessnas: Direct neural architecture search on target task and hardware', *arXiv preprint arXiv:1812.00332*, (2018).

[4] Kenneth Alan De Jong, 'Analysis of the behavior of a class of genetic adaptive systems', (1975).

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, 'ImageNet: A Large-Scale Hierarchical Image Database', in *CVPR09*, (2009).

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

[7] Ross Girshick, 'Fast r-cnn', in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, (2015).

[8] Babak Hassibi and David G Stork, 'Second order derivatives for network pruning: Optimal brain surgeon', in *Advances in neural information processing systems*, pp. 164–171, (1993).

[9] 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*, pp. 770–778, (2016).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2016).

[11] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang, 'Soft filter pruning for accelerating deep convolutional neural networks', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2234–2240. AAAI Press, (2018).

[12] 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)*, pp. 784–800, (2018).

[13] 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).

[14] 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).

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, 'Deep networks with stochastic depth', in *European conference on computer vision*, pp. 646–661. Springer, (2016).

[16] Zehao Huang and Naiyan Wang, 'Data-driven sparse structure selection for deep neural networks', in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 304–320, (2018).

[17] Alex Krizhevsky, Geoffrey Hinton, et al., 'Learning multiple layers of features from tiny images', Technical report, Citeseer, (2009).

[18] Yann LeCun, John S Denker, and Sara A Solla, 'Optimal brain damage', in *Advances in neural information processing systems*, pp. 598–605, (1990).

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

[20] 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', *arXiv preprint arXiv:1903.10258*, (2019).

[21] 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*, pp. 2736–2744, (2017).

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

[23] 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*, pp. 5058–5066, (2017).

[24] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, 'Pruning convolutional neural networks for resource efficient inference', *arXiv preprint arXiv:1611.06440*, (2016).

[25] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin, 'Large-scale evolution of image classifiers', in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2902–2911. JMLR. org, (2017).

[26] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, 'Mobilenetv2: Inverted residuals and linear bottlenecks', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (June 2018).

[27] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le, 'Mnasnet: Platform-aware neural architecture search for mobile', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, (2019).

[28] Andreas Veit, Michael J Wilber, and Serge Belongie, 'Residual networks behave like ensembles of relatively shallow networks', in *Advances in neural information processing systems*, pp. 550–558, (2016).

[29] Lingxi Xie and Alan Yuille, 'Genetic cnn', in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1379–1388, (2017).

[30] 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 European Conference on Computer Vision (ECCV)*, pp. 285–300, (2018).

[31] 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*, pp. 9194–9203, (2018).

[32] Barret Zoph and Quoc V Le, 'Neural architecture search with reinforcement learning', *arXiv preprint arXiv:1611.01578*, (2016).