Robust Neural Network Pruning by Cooperative Coevolution*

Jia-Liang Wu[1], Haopu Shang[1], Wenjing Hong[2], and Chao Qian[1]

[1] State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{wujl, shanghp, qianc}@lamda.nju.edu.cn
[2] Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
hongwj@sustech.edu.cn

Abstract Convolutional neural networks have achieved success in various tasks, but often lack compactness and robustness, which are, however, required under resource-constrained and safety-critical environments. Previous works mainly focused on enhancing either compactness or robustness of neural networks, such as network pruning and adversarial training. Robust neural network pruning aims to reduce computational cost while preserving both accuracy and robustness of a network. Existing robust pruning works usually require expert experiences and trial-and-error to design proper pruning criteria or auxiliary modules, limiting their applications. Meanwhile, evolutionary algorithms (EAs) have been used to prune neural networks automatically, achieving impressive results but without considering the robustness. In this paper, we propose a novel robust pruning method CCRP by cooperative coevolution. Specifically, robust pruning is formulated as a three-objective optimization problem that optimizes accuracy, robustness and compactness simultaneously, and solved by a cooperative coevolution pruning framework, which prunes filters in each layer by EAs separately. The experiments on CIFAR-10 and SVHN show that CCRP can achieve comparable performance with state-of-the-art methods.

Keywords: Model compression · Neural network pruning · Robustness · Evolutionary algorithm · Cooperative coevolution.

1 Introduction

Recently, convolutional neural networks (CNNs) have achieved great success in the field of computer vision, such as image classification [9] and object detection [6]. Despite the impressive performance, the high computational cost of CNNs inhibits their deployments in resource-limited scenarios. The CNNs are also vulnerable to malicious attacks, challenging their reliability in safety-critical scenarios. Therefore, in many real-world applications like autonomous

* This work was supported by the NSFC (62022039, 62106098) and the Jiangsu NSF (BK20201247). Chao Qian is the corresponding author.

driving [4,5], enhancing the compactness and robustness of CNNs simultaneously is essential.

Most previous works, however, only focus on enhancing either compactness or robustness of networks. On the one hand, various model compression methods have been proposed to reduce the computational cost of neural networks, such as neural network pruning [8] and quantization [30]. Among them, neural network pruning aims to remove the redundant parameters in networks while preserving accuracy, which has achieved impressive success. On the other hand, methods like adversarial training [7], which aims to minimize the training loss on adversarial examples, can significantly improve the robustness of neural networks.

Recently, several works [21,22,27] took the network robustness into consideration when pruning neural networks. They usually use criteria designed by experts to measure the importance of network weights and prune the networks accordingly. However, the designing and tuning of such criteria require plenty of expertise and tiring trials, making them difficult to be applied to the practical scenarios where the data sets and neural network architectures can be various. Meanwhile, these works mainly focus on unstructured neural network pruning [2], which can hardly reduce the computation cost in practical applications, since the consequent irregular structures are incompatible with the mainstream software and hardware frameworks. Therefore, an automatic structured robust pruning method is essential in real-world applications.

Robust neural network pruning can be naturally formulated as an optimization problem that aims to search for a sub-net of the original network which still maintains high accuracy and robustness but has less computation cost. Evolutionary algorithms (EAs) [1] are a kind of heuristic randomized optimization algorithms inspired by natural evolution, which have been used for pruning neural networks automatically since the 1990s [26]. However, unlike artificial neural networks in the last century, modern CNNs usually consist of dozens of layers and millions of parameters, implying a huge search space. It is difficult for EAs to obtain satisfactory solutions within a limited computational overhead. Recently, Shang et al. have proposed CCEP [23], an evolutionary pruning method inspired by cooperative coevolution, which has achieved encouraging results on the large-scale pruning problem. However, they only focused on the accuracy but did not take robustness into consideration.

In this paper, we propose a novel Cooperative Coevolution method for Robust Pruning (CCRP). The robust pruning problem is formulated as an explicit three-objective optimization problem, i.e., optimizing accuracy, robustness and compactness simultaneously. A cooperative coevolution framework is adopted to solve the formulated robust pruning problem, which divides the search space by layer and applies an EA to optimize each group. Besides, since the process of generating adversarial examples for each pruned network is time-consuming, we design an adversarial example generating method to improve the efficiency of robustness evaluation.

The contributions of this paper are summarized as follows.

1. We propose a novel framework, CCRP, that considers network robustness during the pruning process and automatically solves the three-objective robust pruning problem by cooperative coevolution. To the best of our knowledge, this is the first application of EAs to robust neural network pruning.
2. We propose an adversarial example generating method to improve the efficiency of evaluating the robustness of pruned networks.
3. We compare CCRP with previous methods through experiments on three network architectures and two data sets. Experimental results show that CCRP can achieve comparable performance with state-of-the-art methods.

2 Related work

2.1 Neural Network Pruning

Neural network pruning aims to enhance the efficiency of a network by removing redundant components. Existing methods can be generally classified into two categories, i.e., unstructured pruning and structured pruning [2]. Unstructured pruning methods directly prune the weights in the parameter matrices of the network. Even though such methods can achieve impressive theoretical acceleration, the resulted sparse matrices and broken structures are incompatible with the mainstream software and hardware platforms, which can hardly obtain actual acceleration. Instead, structured pruning methods focus on pruning structured components such as filters in convolution layers, which have shown better overall performance in real-world applications, and thus have prevailed and attracted more attention nowadays.

Based on how to identify the redundant component, previous structured pruning methods can be generally categorized into criteria-based and learning-based methods. Criteria-based methods (e.g., [15,16]) use expert-designed criteria to identify unimportant components and prune them, while learning-based methods (e.g., [18]) use auxiliary modules to measure the importance of components and conduct pruning accordingly. However, both of these methods heavily rely on the expertise, limiting their application and extensibility.

To get rid of the reliance on expertise, it is natural to use EAs to search for the good pruned networks automatically, which has been studied since the 1990s [26]. However, the huge search space of a deep neural network brings severe challenge to EAs [14,13]. Recently, a novel pruning method inspired by cooperative coevolution named CCEP [23] is proposed, which employs the idea of divide-and-conquer to settle the huge search space and has achieved impressive performance, showing the great potential of EA-based methods for neural network pruning. But the previous EA-based pruning methods never considered the network robustness, which is important to many application scenarios [4,5].

2.2 Robustness of Neural Network

In application scenarios [4,5], neural networks are typically vulnerable to adversarial attacks [7]. Generally, adversaries utilize the model information to generate adversarial examples for attack. An adversarial example can be defined as

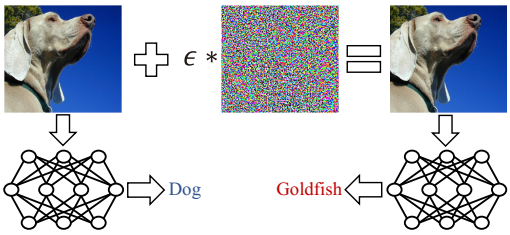

Figure 1: Illustration of an adversarial example in the image classification task.

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta} \qquad s.t. \quad \|\boldsymbol{\eta}\|_\infty \leq \epsilon, \tag{1}$$

where $\boldsymbol{x}$ is the original example and $\boldsymbol{\eta}$ is the perturbation subject to budget ϵ. As shown in Fig 1, an adversarial example in the image classification task is generated by adding perturbations to the original image. The perturbations are imperceptible to the human eyes, but will mislead the neural network to incorrect prediction. When facing adversarial attacks, the robustness of a neural network $\mathcal{N}$ is usually measured by the robust accuracy on an adversarial data set D_a, i.e.,

$$\mathrm{ACC}_r(\mathcal{N}) = \frac{1}{|D_a|} \sum_{\hat{\boldsymbol{x}}, y \in D_a} \mathbb{I}(\mathcal{N}(\hat{\boldsymbol{x}}) = y), \tag{2}$$

where $|D_a|$ denotes the size of D_a (i.e., the number of adversarial examples), $\mathcal{N}(\hat{\boldsymbol{x}})$ denotes the prediction of the neural network $\mathcal{N}$ on the adversarial example $\hat{\boldsymbol{x}}$, and $\mathbb{I}(\cdot)$ is the indicator function that is 1 if the inner expression is true and 0 otherwise.

Adversarial training [7,19,29] is one of the primary defense methods against adversarial attacks. The main idea is to minimize the training loss on adversarial examples generated by adversarial attacks, such as fast gradient sign method (FGSM) [7]. Thus, the objective of adversarial training can be formulated as

$$\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\eta}\|_\infty \leq \epsilon} L(\boldsymbol{x} + \boldsymbol{\eta}, y, \boldsymbol{\theta}) \right], \tag{3}$$

where $\boldsymbol{\theta}$ denotes the parameters of the neural network, and L is a loss function. Previous empirical results have indicated that adversarial training requires the networks owning a larger capacity for better overall performance. Therefore, neural networks with robustness are usually too computationally intensive to be deployed on resource-constrained applications.

2.3 Robust Neural Network Pruning

Recently, some works [10,25] have studied on the relationship between robustness and network capacity, revealing that a sub-net of the original network can have similar or even better robustness than the original one, and different sub-nets can have quite different robustness. This finding has inspired robust neural network

pruning, which aims to find a compact neural network and retain the robustness. The few existing methods usually train a network by adversarial training and conduct unstructured pruning based on expert-designed criteria. For example, ADV-LWM [21] prunes weights with small l_1-norm, and fine-tunes the obtained network by adversarial training to recover robustness. Ye *et al.* [27] adopts the ADMM pruning framework by replacing the original training loss with an adversarial one. HYDRA [22] adds importance scores to all the weights in the network, and optimizes the adversarial loss by adjusting importance scores while freezing weights. Then the weights with small importance scores are pruned. DNR [12] chooses the feature matrices with small Frobenius norm and prunes the corresponding filters. Furthermore, these methods need proper pruning ratios of each layer, which also often require a lot of expert experience and trial-and-error.

3 CCRP Method

Let $\mathcal{N}$ denote a well-trained neural network with n convolution layers $\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_n\}$, where $\mathcal{L}_i$ denotes the ith layer which has l_i filters and $\mathcal{L}_{ij}$ denotes the jth filter in the ith layer. Robust neural network pruning can be formulated as an optimization problem, with the aim of searching for a subset of filters in $\mathcal{N}$, which can maximize the accuracy and robustness while minimizing computational cost simultaneously. Let the mask vector $\mathcal{M} = \{m_{ij} \mid m_{ij} \in \{0,1\}, i \in \{1,2,...,n\}, j \in \{1,2,...,l_i\}\}$, where $m_{ij} = 1$ if and only if $\mathcal{L}_{ij}$ is retained. Thus, a pruned network can be represented by the mask $\mathcal{M}$ as

$$\mathcal{N}_{\mathcal{M}} = \bigcup_{i=1}^{n} \bigcup_{j=1}^{l_i} m_{ij}\mathcal{L}_{ij}. \tag{4}$$

Let $\mathrm{ACC}(\mathcal{N}_{\mathcal{M}})$ denote the accuracy of the pruned network $\mathcal{N}_{\mathcal{M}}$ on the clean data sets, $\mathrm{ACC}_r(\mathcal{N}_{\mathcal{M}})$ denote the robust accuracy on the generated adversarial examples, and $\mathrm{FLOPs}(\mathcal{N}_{\mathcal{M}})$ denote the number of FLoating point OPerations, which is a common metric to measure the computational cost. The robust neural network pruning problem can be formulated as

$$\underset{\mathcal{M}\in\{0,1\}^{\sum_{i=1}^{n} l_i}}{\arg\max} \left(\mathrm{ACC}(\mathcal{N}_{\mathcal{M}}), \mathrm{ACC}_r(\mathcal{N}_{\mathcal{M}}), -\mathrm{FLOPs}(\mathcal{N}_{\mathcal{M}})\right) \tag{5}$$

Because the number of alternative filters to be pruned in a CNN, i.e. $\sum_{i=1}^{n} l_i$, can be very large, this is essentially a challenging large-scale optimization problem. To solve this problem, we propose a novel robust pruning method named CCRP. Inspired by our previous work CCEP [23], we adopt a cooperative coevolution pruning framework which divides the search space by layer and conducts an EA on each layer independently. Robust accuracy is set as an optimization objective to directly guide pruned neural networks towards robustness. Note that the evaluation of $\mathrm{ACC}_r(\mathcal{N}_{\mathcal{M}})$ is time-consuming since specialized adversarial examples need to be generated for each pruned network. To settle this, we propose an adversarial example generating method that needs to generate adversarial

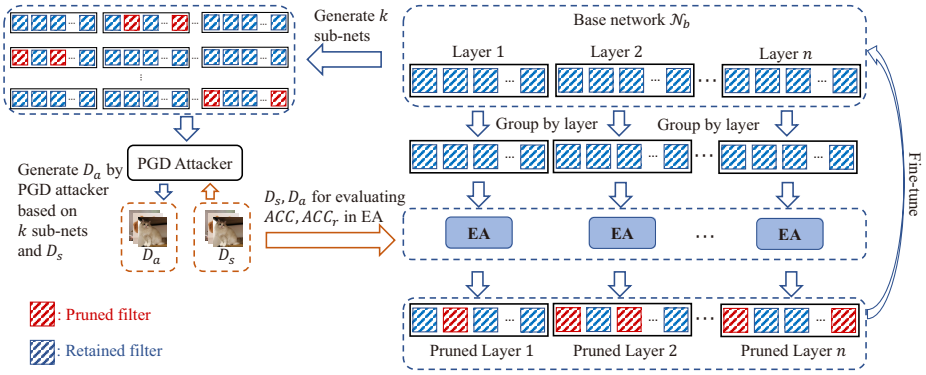

Figure 2: Illustration of the framework of CCRP.

samples only once in each iteration of CCRP. The generated examples can be used to evaluate all the pruned networks in this iteration.

3.1 Framework of CCRP

The framework of CCRP is shown in Algorithm 1. It prunes a well-trained network iteratively and finally outcomes a set of pruned networks with robustness for user selection. Each iteration works as follows. Firstly, a mask $\mathcal{M}$ is generated based on the network to be pruned in line 4. Then the mask $\mathcal{M}$ will be split into n groups by layer in line 5. After that, in line 6, a set D_a of adversarial examples is generated for the evaluation of the pruned networks, which is shown in Algorithm 3 in detail. For each group, an EA is employed to optimize it and obtain $\boldsymbol{m}_i'$ representing the pruned result of the ith layer. The process of EA in each group is described in Algorithm 2. By collecting all the $\boldsymbol{m}_i'$ of n layers and applying it to $\mathcal{N}_b$, the corresponding pruned network $\mathcal{N}'$ is obtained, as presented in line 8. After pruning, to recover the accuracy and robust accuracy, the pruned network $\mathcal{N}'$ will be fine-tuned by adversarial training in line 9. The fine-tuned model will be used as the new base network $\mathcal{N}_b$ to be pruned in the next iteration, and added into archive H in line 11. After T iterations, the CCRP method will stop and return the pruned networks in archive H. An illustration of the framework of CCRP is also shown in Fig 2.

3.2 EA in Each Group

For EA in each group, we use a typical evolutionary process: generate an initial subpopulation randomly, breed new individuals by applying reproductive operators, evaluate the fitness of each individual, and select better individuals to remain in the next generation. When the termination condition is reached, it selects an individual from the final subpopulation, which represents the corresponding pruned layer.

The detailed description of EA in each group is shown in Algorithm 2. At the beginning, it generates the initial subpopulation P with d individuals, as

Algorithm 1 CCRP Framework

Input: A well trained CNN $\mathcal{N}$ with n layers, maximum number T of iterations, training set D_t, a randomly sampled part D_s of the training set D_t
Output: A set of pruned networks with different sizes

1: Let $H = \emptyset$, $i = 0$;
2: Set base network $\mathcal{N}_b = \mathcal{N}$;
3: **while** $i < T$ **do**
4: Generate a mask $\mathcal{M}$ based on $\mathcal{N}_b$ and initialize it with all bits equal to 1;
5: $\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_n = \text{Decompose}(\mathcal{M})$;
6: $D_a = $ Generate adversarial samples based on $(\mathcal{N}_b, D_s)$;
7: $\boldsymbol{m}'_1, \boldsymbol{m}'_2, \cdots, \boldsymbol{m}'_n = \text{EA}(\boldsymbol{m}_1, D_s, D_a), \text{EA}(\boldsymbol{m}_2, D_s, D_a), \cdots, \text{EA}(\boldsymbol{m}_n, D_s, D_a)$;
8: $\mathcal{N}' = \bigcup_{i=1}^{n} \bigcup_{j=1}^{l_i} m'_{ij} \mathcal{L}_{ij}$, where $\mathcal{L}_{ij}$ is a filter of $\mathcal{N}_b$;
9: $\mathcal{N}_b = \text{Fine-tune}(\mathcal{N}', D_t)$;
10: $H = H \cup \mathcal{N}_b$;
11: $i = i + 1$
12: **end while**
13: **return** H

shown in line 2. An individual $\boldsymbol{m}_0$ with all bits equal to 1 is added into P to encourage conservative pruning. The rest $d - 1$ individuals are generated by applying a modified bit-wise mutation operator with mutation rate p_1. In each generation of EA, it generates d new offspring individuals by uniformly randomly choosing d individuals from the subpopulation with replacement and applying the bit-wise mutation operator with mutation rate p_2. Following the prior work CCEP [23], we make a modification to the standard bit-wise mutation operator to prevent the pruning process from being too violent. That is, a ratio bound r is introduced to limit the number of filters to be pruned. Specifically, a standard bit-wise mutation operator with mutation rate p is first performed on an individual $\boldsymbol{m}$; if the number of 0s in the mutated $\boldsymbol{m}$, denoted by $|\boldsymbol{m}|_0$, is larger than $\text{Len}(\boldsymbol{m}) \times r$, it will randomly select $|\boldsymbol{m}|_0 - \text{Len}(\boldsymbol{m}) \times r$ bits from where $m_i = 0$, and flip them to 1.

When evaluating the fitness of an offspring individual, noting that each individual only corresponds to a single pruned layer, it first obtains a complete network by splicing this single layer with the other layers obtained from the base network $\mathcal{N}_b$. Then, we can evaluate the accuracy, robust accuracy, and FLOPs of offspring individuals. More specifically, ACC is evaluated on the clean data set D_s, which is randomly sampled from the training set D_t; ACC_r is evaluated on the adversarial data set D_a, which is generated by Algorithm 3 and will be introduced in Section 3.3; FLOPs can be calculated directly. After evaluation in lines 6-7, the d offspring individuals and d individuals in the current subpopulation P will be merged into a collection Q. Since we consider three objectives as in Eq. (5), it is not easy to find a proper ranking of the individuals. For simplicity, the individuals in Q are ranked by the average value of ACC and ACC_r in descending order. As for two individuals with the same average value, the one with less FLOPs is ranked ahead. Other techniques (e.g., non-dominated sorting [3]) may also be employed, and will be investigated in our future work.

Algorithm 2 EA in each group

Input: A mask $\boldsymbol{m}_i$ of the ith layer, population size d, a randomly sampled part D_s of the training set D_t, adversarial data set D_a, mutation rate p_1, p_2, ratio bound r, maximum number G of generations
Output: Mask vector $\boldsymbol{m}_i'$

1: Let $j = 0$, $\boldsymbol{m}_0 = \boldsymbol{m}_i$;
2: Initialize a subpopulation P with $\boldsymbol{m}_0$ and $d-1$ individuals generated from $\boldsymbol{m}_0$ by applying the bit-wise mutation operator with p_1 and r;
3: **while** $j < G$ **do**
4: Uniformly randomly select d individuals from P with replacement as the parent individuals;
5: Generate d offspring individuals by applying the bit-wise mutation operator with p_2 and r on each parent individual;
6: Calculate the ACC and ACC_r of d offspring individuals by using D_s and D_a;
7: Calculate the FLOPs of d offspring individuals;
8: Set Q as the union of P and d offspring individuals;
9: Rank the $2d$ individuals in Q in descending order by $\frac{\text{ACC}+\text{ACC}_r}{2}$; for two individuals with the same value, the one with less FLOPs is ranked ahead;
10: Replace the individuals in P with the top d individuals in Q;
11: $j = j + 1$
12: **end while**
13: Select the rank one individual in P as $\boldsymbol{m}_i'$
14: **return** $\boldsymbol{m}_i'$

After the evolution of G generations, the individual that ranks first in the final subpopulation is chosen as the pruned result of the corresponding group.

3.3 Robustness Evaluation

Typically, the robustness of a neural network is based on its ability against adversarial attacks. In this paper, we use the robust accuracy on the generated adversarial examples as the metric of robustness, which is denoted as ACC_r. CCRP applies the state-of-the-art white-box attack algorithm PGD[19] to generate the adversarial examples. PGD is designed to attack a specialized network in an iterative style, which is time-consuming. If we use PGD to generate specialized adversarial examples when evaluating each pruned network, the computational overhead will be prohibitive. To settle this issue, we design an adversarial example generating method shown in Algorithm 3 to generate an adversarial data set D_a, which can be shared in one iteration of CCRP. The method samples a sub-net $\mathcal{N}'$ of base network $\mathcal{N}_b$ by randomly selecting $\lceil n/k \rceil$ layers in $\mathcal{N}_b$ and applying bit-wise mutation operator with p_1 and r to them in lines 4-5, and then employs PGD on $\mathcal{N}'$ to generate adversarial examples based on D_s in line 6. This process will be repeated k times independently, and all the generated adversarial examples constitute the adversarial data set D_a, which will be used to evaluate the robustness of all the pruned networks in the current iteration of CCRP. Note that the goal of generating adversarial examples from diverse sub-nets of $\mathcal{N}_b$ is to better measure the robustness of a pruned network.

Algorithm 3 Adversarial Example Generating

Input: Base network $\mathcal{N}_b$ with n layers, a randomly sampled part D_s of the training set D_t, number k of sampled sub-nets.
Output: Adversarial data set D_a

Let $D_a = \emptyset$, $i = 0$;
while $i < k$ **do**
    Randomly select $\lceil n/k \rceil$ layers from $\mathcal{N}_b$;
    Apply mutation with p_1 and r on these layers to obtain a sub-net $\mathcal{N}'$;
    $A = \mathrm{PGD}(\mathcal{N}', D_s)$;
    $D_a = D_a \cup A$;
    $i = i + 1$
end while
return D_a

3.4 Comparison with CCEP

In this subsection, we make a comparison between CCEP and CCRP. CCEP applies cooperation coevolution to neural network pruning and achieves impressive results. CCRP is inspired by CCEP and extended to robust neural network pruning, i.e., by taking the robustness of networks into consideration. These two methods use a similar decomposition strategy that splits the search space by layer. The most significant difference between them is the problem formulation. CCRP introduces robustness as an optimization objective while CCEP concerns accuracy and compactness only. An adversarial example generation method has been introduced into CCRP, which can reduce the cost of robustness evaluation. In addition, adversarial training is applied in fine-tuning to retain the robustness of the pruned network.

4 Experiments

We conduct experiments from three aspects. First, we compare CCRP with the state-of-the-art unstructured robust pruning methods. Second, we extend several popular structured pruning methods to robust pruning and compare CCRP with them. In the third aspect, we conduct repeated experiments to examine the stability of CCRP and visualize the architecture of pruned networks.

Two popular image classification data sets CIFAR-10 [11] and SVHN [20], and three typical neural networks VGG [24], ResNet [9] and WRN [28] are used for examination. Following the common filter pruning settings, CCRP prunes all convolution layers for VGG and the first convolution layer of the residual blocks for ResNet and WRN. The popular adversarial training method, TRADES [29], is used in the pre-train and fine-tune processes. The settings of CCRP are described as follows. It runs for 16 iterations, i.e., $T = 16$. For EA in each group, the population size d is 5, the mutation rate p_1 and p_2 are 0.05 and 0.1, respectively, the ratio bound r is 0.1, the maximum generation G is 10, and D_s is generated by selecting 10% of the training set randomly. When generating adversarial examples, the number k of sampled sub-nets is 5.

Table 1: Comparison in terms of ACC, ACC_r, and inference speed with unstructured robust pruning methods. The best results of each objective are shown in bold.

Data Set	Architecture	Method	Base ACC (%)	Base ACC_r (%)	ACC↓ (%)	ACC_r ↓ (%)	Speed (images/s)
CIFAR-10	VGG-16	ADV-LWM	82.70	51.90	3.90	4.20	2082.13
		ADV-ADMM	78.36	47.07	3.50	3.76	2114.77
		HYDRA	82.70	51.90	2.20	2.40	2077.57
		CCRP	81.57	61.71	**-1.39**	**0.63**	**6842.39**
	WRN-28-4	ADV-LWM	85.60	57.20	2.80	3.40	4142.74
		ADV-ADMM	78.22	51.56	2.46	2.50	4375.58
		HYDRA	85.60	57.20	1.90	2.00	4016.55
		CCRP	85.91	53.42	**-0.05**	**-9.12**	**4737.09**
SVHN	VGG-16	ADV-LWM	90.50	53.50	1.30	2.00	2308.65
		ADV-ADMM	89.35	54.61	-0.23	4.10	2322.72
		HYDRA	90.50	53.50	1.30	**1.10**	2334.29
		CCRP	86.86	53.18	**-1.58**	2.36	**11124.56**
	WRN-28-4	ADV-LWM	93.50	60.10	1.20	0.70	5259.51
		ADV-ADMM	92.14	59.07	1.32	4.53	5482.91
		HYDRA	93.50	60.10	-0.90	**-2.70**	5294.31
		CCRP	90.07	57.47	**-1.63**	-0.18	**6467.55**

For adversarial training by TRADES [29], the common settings are used. The optimizer is SGD with an initial learning rate 0.1, and a Cosine Annealing scheduler [17] is employed to adjust the learning rate during fine-tuning. The weight decay is 0.0001 and the momentum is 0.9. The number of epochs in each process of fine-tuning is 30. The batch size for training is 128. For adversarial attack by PGD, the l_∞ perturbation budget, number of steps, and perturbation per step are set as 8/255, 10, 2/255 respectively in adversarial training and 8/255, 40, 2/255 for evaluation and testing.

We compare CCRP with various methods, including three state-of-the-art unstructured robust pruning methods: ADV-LWM [21], ADV-ADMM [27] and HYDRA [22], as well as two structured pruning methods L1 [15] and HRank [16] extended to robust pruning. The results of HYDRA and ADV-LWM are obtained from their released models. All the experiments are realized based on PyTorch and carried out on a single Nvidia GeForce RTX-3090 GPU.

Comparison with Unstructured Robust Pruning Methods: We first compare CCRP with state-of-the-art unstructured robust pruning methods in terms of accuracy drop, robust accuracy drop, and inference speed, as shown in Table 1. Inference speed is used to measure the computation cost here since FLOPs drop of unstructured models cannot reflect the actual computational performance in applications. The inference speed is tested on 100,000 32×32 images with a batch size of 128. For CCRP, the solution in the 10th iteration is presented in Table 1 for comparison. CCRP achieves a smaller drop in accuracy and robust accuracy with faster inference speed in most cases. On SVHN, HYDRA [22] and ADV-LWM [21] achieve a smaller drop in robust accuracy than

CCRP but more drop in accuracy and slower inference speed. It is worth noting that CCRP achieves the fastest inference speed in all cases.

Comparison with Structured Robust Pruning Methods: For a more comprehensive comparison, two structured pruning methods, L1 [15] and HRank [16], are extended to the scenario of robust pruning by introducing adversarial training in the pre-train and fine-tune steps. For CCRP, we select the solution in the 16th iteration for comparison. The results in Table 2 show that compared with L1 and HRank, CCRP can always achieve better performance on at least two of the three metrics. Sometimes CCRP prevails on two metrics but only with minor disadvantage on the third metric.

Table 2: Comparison in terms of ACC, ACC_r, and pruning ratio with structured robust pruning methods. The best results of each objective are shown in bold.

Data Set	Architecture	Method	Base ACC (%)	Base ACC_r (%)	ACC↓ (%)	ACC_r ↓ (%)	FLOPs ↓ (%)
CIFAR-10	VGG-16	L1	81.57	61.71	2.00	3.37	69.23
		HRank	81.91	61.11	7.05	**3.01**	65.85
		CCRP	81.57	61.71	**0.05**	6.22	**77.95**
	ResNet-56	L1	80.31	48.95	2.47	-4.62	68.53
		HRank	80.31	48.95	**0.13**	-2.22	50.02
		CCRP	80.31	48.95	0.23	**-8.31**	**72.30**
	WRN-28-4	L1	85.91	53.61	2.00	3.37	**69.23**
		CCRP	85.91	53.61	**-0.88**	**-8.06**	66.92
SVHN	VGG-16	L1	86.86	53.18	2.17	4.35	**85.88**
		HRank	86.06	54.53	0.40	5.03	65.85
		CCRP	86.86	53.18	**-0.98**	**2.68**	80.44
	ResNet-56	L1	85.91	52.24	**-2.04**	-1.78	60.08
		HRank	87.09	55.57	-1.53	3.25	50.02
		CCRP	85.91	52.24	-1.95	**-2.01**	**70.71**
	WRN-28-4	L1	90.07	57.47	1.45	4.19	**72.11**
		CCRP	90.07	57.47	**-1.25**	**1.46**	70.99

Further Studies: Because experiments of network pruning are very time-consuming and may require dozens of hours, previous works [15,16,21,22] usually conducted experiments only once. However, considering the stochastic characteristic of EAs, we conduct a repeated test on a relatively small data set CIFAR-10, to prune ResNet-56 and VGG-16 for ten independent runs. The ACC, ACC_r, and ACC_a (i.e., the average of ACC and ACC_r), are recorded and shown in Fig 3. The solid line is the mean value, and the shadow area represents the 95% confidence interval. We can observe that the ACC and ACC_r are even better than the base model when the pruning ratio is low and get a slight drop as the pruning ratio increases; the ACC_a is always better than the base model. The 95% confidence interval implies the good stability of CCRP.

In Fig 4, we visualize the architecture (i.e., the number of filters in each layer or residual block) of pruned networks on CIFAR-10. The results show that

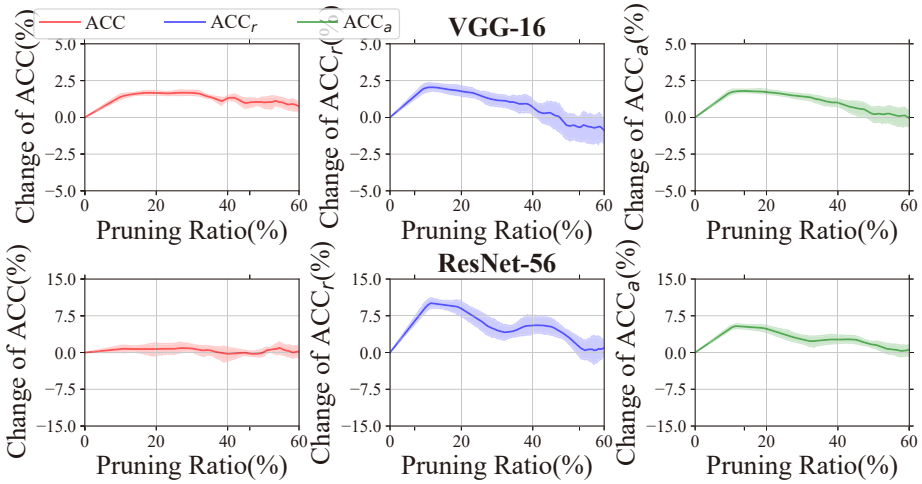

Figure 3: Repeated test of CCRP on CIFAR-10.

CCRP can choose different pruning ratios at different layers (or residual blocks) automatically. For ResNet-56, CCRP prunes fewer filters around the expansion of channels, while on VGG-16, CCRP prunes more filters after the 6th layer. As for WRN-28-4, more filters at 9th block are preserved. Note that previous robust pruning methods may require lots of trial-and-error to design the proper pruning ratios at each layer manually.

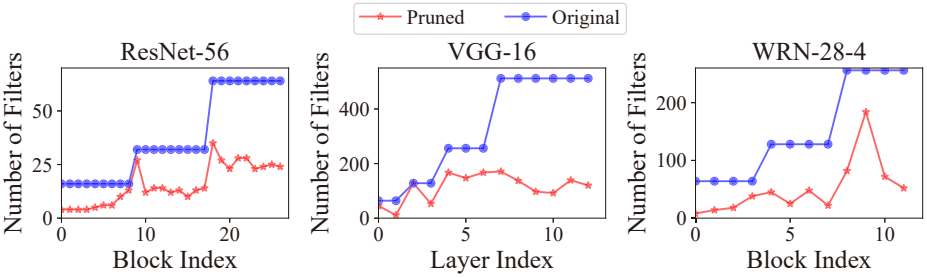

Figure 4: Visualization of the original networks and the pruned networks obtained by CCRP on CIFAR-10.

5 Conclusion

This paper proposes the automatic robust neural network pruning method, which formulates robust pruning as a three-objective optimization problem considering robustness, and solves it by an adapted cooperative coevolution framework. To the best of our knowledge, this is the first application of EAs to robust neural network pruning. Experiments show that CCRP can achieve a comparable performance with the state-of-the-art methods. In the future, we will try to perform theoretical analysis [31], as well as incorporate more advanced multi-objective optimization techniques to improve the performance of CCRP.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford, UK (1996)
2. Cheng, J., Wang, P., Li, G., Hu, Q., Lu, H.: Recent advances in efficient computation of deep convolutional neural networks. Frontiers of Information Technology and Electronic Engineering **19**(1), 64–77 (2018)
3. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
4. Duan, R., Ma, X., Wang, Y., Bailey, J., Qin, A.K., Yang, Y.: Adversarial camouflage: Hiding physical-world attacks with natural styles. In: Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 997–1005. Seattle, WA (2020)
5. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1625–1634. Salt Lake City, UT (2018)
6. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 580–587. Columbus, OH (2014)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR). San Diego, CA (2015)
8. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: Proceedings of the 4th International Conference on Learning Representations (ICLR). San Juan, Puerto Rico (2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. Las Vegas, NV (2016)
10. Huang, H., Wang, Y., Erfani, S., Gu, Q., Bailey, J., Ma, X.: Exploring architectural ingredients of adversarially robust deep neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 34, pp. 5545–5559. New Orleans, LA (2021)
11. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Canada (2009)
12. Kundu, S., Nazemi, M., Beerel, P.A., Pedram, M.: DNR: A tunable robust pruning framework through dynamic network rewiring of dnns. In: Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASPDAC). pp. 344–350. Tokyo, Japan (2021)
13. Li, G., Qian, C., Jiang, C., Lu, X., Tang, K.: Optimization based layer-wise magnitude-based pruning for DNN compression. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI). pp. 2383–2389. Stockholm, Sweden (2018)
14. Li, G., Yang, P., Qian, C., Hong, R., Tang., K.: Magnitude-based pruning for recurrent neural networks. IEEE Transactions on Neural Networks and Learning Systems (in press)

15. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: Proceedings of the 5th International Conference on Learning Representations (ICLR). Toulon, France (2017)
16. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the 2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1526–1535. Los Alamitos, CA (2020)
17. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: Proceedings of the 5th International Conference on Learning Representations (ICLR). Toulon, France (2017)
18. Luo, J., Wu, J.: Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. Pattern Recognition **107**(107461), 107461 (2020)
19. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: Proceedings of the 6th International Conference on Learning Representations (ICLR). Vancouver, Canada (2018)
20. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: Advances in Neural Information Processing Systems, Workshop (NeurIPS) (2011)
21. Sehwag, V., Wang, S., Mittal, P., Jana, S.: Towards compact and robust deep neural networks. CoRR p. abs/1906.06110 (2019)
22. Sehwag, V., Wang, S., Mittal, P., Jana, S.: HYDRA: Pruning adversarially robust neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 33, pp. 19655–19666. Vancouver, Canada (2020)
23. Shang, H., Wu, J.L., Hong, W., Qian, C.: Neural network pruning by cooperative coevolution. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI). Vienna, Austria (2022)
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR). San Diego, CA (2015)
25. Wu, B., Chen, J., Cai, D., He, X., Gu, Q.: Do wider neural networks really help adversarial robustness? In: Advances in Neural Information Processing Systems (NeurIPS). vol. 34, pp. 7054–7067. New Orleans, LA (2021)
26. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE **87**(9), 1423–1447 (1999)
27. Ye, S., Lin, X., Xu, K., Liu, S., Cheng, H., Lambrechts, J., Zhang, H., Zhou, A., Ma, K., Wang, Y.: Adversarial robustness vs. model compression, or both? In: Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV). pp. 111–120. Seoul, Korea (South) (2019)
28. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of the 2016 British Machine Vision Conference (BMVC). York, UK (2016)
29. Zhang, H., Yu, Y., Jiao, J., Xing, E.P., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: Proceedings of the 36th International Conference on Machine Learning (ICML). pp. 7472–7482. Long Beach CA (2019)
30. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. In: Proceedings of the 5th International Conference on Learning Representations (ICLR). Toulon, France (2017)
31. Zhou, Z., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer, Singapore (2019)