Toward Compact and Robust Model Learning Under Dynamically Perturbed Environments

Hui Luo, Zhuangwei Zhuang, Yuanqing Li, *Fellow, IEEE*, Mingkui Tan, *Member, IEEE*, Cen Chen, *Senior Member, IEEE*, and Jianlin Zhang

***Abstract*— Network pruning has been widely studied to reduce the complexity of deep neural networks (DNNs) and hence speed up their inference. Unfortunately, most existing pruning methods ignore the changes in the model's robustness before and after pruning, which makes pruned models vulnerable under dynamically perturbed environments (e.g., autonomous driving). Only a few works have explored the robustness of pruned models against adversarial attacks that significantly differ from perturbations in real-world scenarios. To bridge the gap between real-world applications and existing studies, in this work, we propose an adversarial pruning scheme, which automatically identifies and preserves robust channels to obtain robust pruned models that are suitable for practical deployment in dynamically perturbed environments. Specifically, to simulate real-world perturbations, we first employ multi-type adversarial attack samples and adversarial perturbation samples generated by an adversarial perturbation generator to create mixed noise samples. Then, we propose a plug-and-play feature scoring module and a novel contribution difference loss to evaluate the robustness of intermediate features dynamically. Next, to leverage robust intermediate features to identify robust channels, we have developed a simple but effective gating mechanism that evaluates the robustness of channels and preserves robust channels during training. Lastly, we compress the model in a layer-wise or block-wise manner. Compared to existing methods, our scheme enhances the robustness of the pruned model in a broader sense, making it better able to against dynamic perturbations in the real world. Extensive experimental results on well-known dataset benchmarks and popular network architectures demonstrate the effectiveness of our method.**

***Index Terms*— Network pruning, robustness, dynamically perturbed environments, intermediate features, adversarial pruning.**

I. Introduction

In RECENT years, researchers have extensively explored the application of DNNs on numerous computer vision tasks, such as image classification [3], [4], object detection [5], [6], and video analysis [7], [8], [9]. However, deep models are often difficult to deploy to some resource-constrained devices (e.g., smart bracelets, mobile phones, sensors) due to the enormous computational cost and memory footprint. To address this issue, various model compression methods have been proposed to compress and accelerate the deep model, including network pruning [10], quantization [11], [12], knowledge distillation [13] and tensor factorization [14], etc.

Network pruning is an important approach, which aims to identify and remove unimportant neurons or connections based on some criterion without significant degradation in performance. Existing pruning methods can be divided into two categories: unstructured pruning [15] and structured pruning [16], [17], [18]. Unstructured pruning can achieve a higher compression ratio than structured pruning by discarding unimportant weights. However, the pruned weight matrices are irregularly sparse, which may limit the actual network acceleration. Therefore, unstructured pruning requires dedicated hardware and software to accelerate inference. In contrast, structured pruning is a more prevalent approach, which prunes unimportant channels to get a structured pruned model. Since structured pruning does not rely on additional hardware or libraries, it is more commonly used in practice.

Although existing network pruning can effectively improve the model's efficiency, it may compromise the robustness of the model. As shown in Fig. 1, existing pruning methods cause a decrease in the robustness of the pruned model. We analyze existing pruning methods, as shown in Fig. 2 (a), most current pruning methods using a three-stage pipeline: 1) Training a model on a target data to obtain a well-trained pre-trained model, 2) Pruning the pre-trained model on the same data based on some criteria, 3) Fine-tuning the pruned

Manuscript received 10 July 2023; revised 13 October 2023 and 6 November 2023; accepted 26 November 2023. Date of publication 28 November 2023; date of current version 6 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072190; in part by the Key-Area Research and Development Program Guangdong Province under Grant 2018B010107001; in part by the Key Realm Research and Development Program of Guangzhou under Grant 202007030007; and in part by the Frontier Research Fund of Institute of Optics and Electronics, China Academy of Sciences, under Grant C21K005. This article was recommended by Associate Editor X. Chang. *(Corresponding authors: Jianlin Zhang; Mingkui Tan; Cen Chen.)*

Hui Luo and Jianlin Zhang are with the National Key Laboratory of Optical Field Manipulation Science and Technology and the Key Laboratory of Optical Engineering, Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu 610209, China, also with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Pazhou Laboratory, Guangzhou 510330, China (e-mail: luohui19@mails.ucas.ac.cn; jlin_zh@163.com).

Zhuangwei Zhuang and Mingkui Tan are with the School of Software Engineering and the Key Laboratory of Big Data and Intelligent Robot, Ministry of Education, South China University of Technology, Guangzhou 510641, China (e-mail: z.zhuangwei@mail.scut.edu.cn; mingkuitan@scut.edu.cn).

Yuanqing Li is with the Pazhou Laboratory, Guangzhou 510330, China (e-mail: auyqli@scut.edu.cn).

Cen Chen is with the School of Future Technology, South China University of Technology, Guangzhou 510641, China, and also with the Pazhou Laboratory, Guangzhou 510330, China (e-mail: chencen@scut.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2023.3337538.

Digital Object Identifier 10.1109/TCSVT.2023.3337538

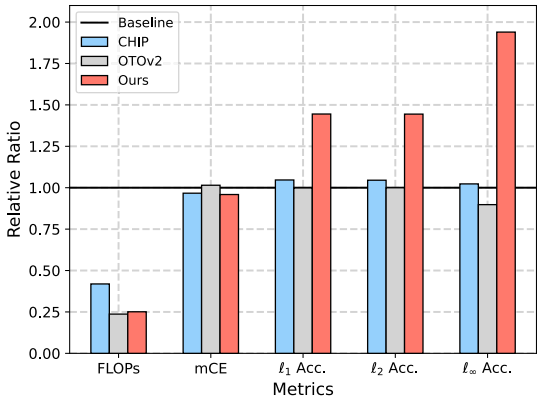


Fig. 1. Comparison of our method with CHIP [1] and OTOv2 [2] regarding efficiency and robustness before and after pruning. Experiments are conducted on CIFAR-10 and VGG-16. FLOPs refer to floating point operations. ℓ_1 Acc., ℓ_2 Acc. and ℓ_∞ Acc. are the attack accuracy under ℓ_1 attack, ℓ_2 attack and ℓ_∞ attack, respectively. mCE represents the mean Corruption Error on CIFAR-10-C, the lower the better. Best viewed in color.

model on the same data to recover its accuracy. The above three stages are all executed on static target data, which makes the pruned model have no apparent performance loss on static data but may perform poorly in dynamically perturbed environments. On the one hand, dynamic perturbations can lead to a mismatch in the test and training data distributions, i.e., data distribution shift. Even if the shift is mild, the performance of models can be severely affected [19]. On the other hand, the information preserved in pruned models obtained using conventional pruning methods is relatively dependent on the distribution of the original data. Although some adversarial attack-based pruning methods have been proposed to obtain robust pruned models, these pruned models are still unsatisfactory in practice due to the large difference between adversarial attacks and real-world dynamic perturbations. Actually, the pruned model may inevitably be applied in various real-world scenarios with dynamically perturbed environments. For example, deploying the pruned model in autonomous driving, which requires the pruned model to be able to make reliable predictions under changing driving conditions, such as severe weather conditions (e.g., rain, snow and fog) and sensor degradation (e.g., gaussian noise, elastic transform and motion blur), etc. These highlight the importance of considering the robustness of the pruned model as an evaluation metric. How to simultaneously reduce the computational overhead and maintain performance, including robustness, on resource-constrained devices becomes a challenging but important problem to be solved.

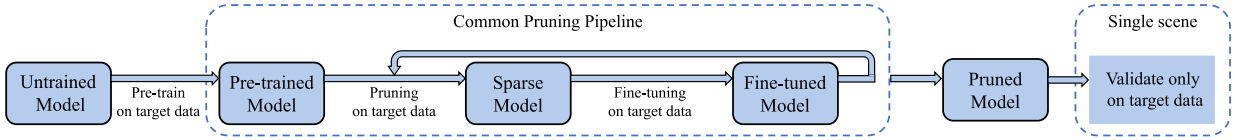
To tackle the above challenge, previous works have explored enhancing the robustness of pruned models against adversarial attacks while pruning, such as [20] and [21]. However, these methods focus only on improving robustness against artificially designed adversarial attacks, which is less relevant to most real-world applications than natural corruptions [22], [23]. On the other hand, some recent work [24], [25] has not yet agreed on whether improving the robustness of a model against adversarial attacks will simultaneously improve the robustness against natural corruptions. In order to improve the robustness of the model in a broader sense, we should consider the characteristics that a robust model should embody. We believe that robust models are more tolerant to noisy samples than weak models. Specifically, a pair of clean and noisy samples is fed into the robust model, and the model should make similar predictions. Furthermore, an ideal robust neuron extracts intermediate features from a pair of clean and noisy samples that should contribute equally to the final prediction of the model.

Under this view, in this work, we propose an adversarial pruning scheme to improve the robustness of pruned models in a broader sense. Our scheme is based on mixed noise samples to obtain a robust pruned model by identifying and preserving robust channels. First, we learn mixed noise samples to simulate complex real-world environments. Specifically, we utilize multi-type iterative adversarial attacks to generate a variety of input-dependent adversarial attack samples. In order to better identify robust features, we further learn input-independent adversarial perturbation samples by proposing a Gaussian noise-based adversarial perturbation generator that randomly perturbs the input during adversarial pruning. After obtaining adversarial attack samples and adversarial perturbation samples, we merge the two to get mixed noise samples for identifying robust features. Unlike other adversarial pruning methods that directly identify robust channels or weights, our scheme identifies robust channels from the perspective of identifying robust intermediate features, which is beneficial to dynamically identifying robust channels based on dynamic input. To this end, we then design a plug-and-play feature scoring module that dynamically evaluates the robustness score of intermediate features. Moreover, in light of the above view on ideal robust neurons, we propose a novel contribution difference loss to align the contributions of clean and noisy intermediate features at each layer of the model to identify robust intermediate features. Next, we adopt a simple yet effective gating mechanism, which exploits the robustness score of intermediate features from clean and noisy samples to identify the robustness channels of the model. Finally, we follow the three-stage pipeline in network pruning to compress the model with our proposed adversarial pruning scheme in a layer-wise or block-wise manner, see Fig. 2 (b). Our work improves the robustness of pruned models in a broader sense, as our work not only investigates to improve the robustness against adversarial attacks, but also tries to improve the robustness against natural corruptions.

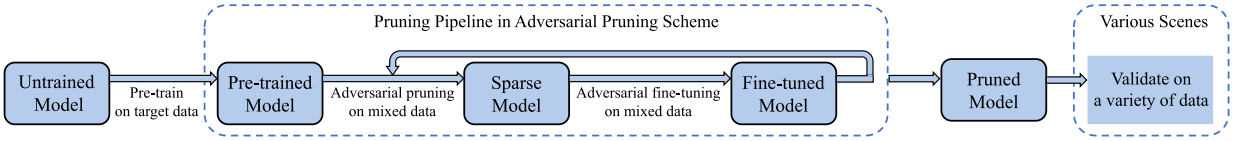
We evaluate our proposed adversarial pruning scheme on VGG-16, ResNets, and MobileNet-V2. We also report the robustness of pruned models obtained by some state-of-the-art pruning methods, and the evaluation results show that they reduce the robustness of the model when pruning. Compared with them, our method shows a clear advantage: pruned models with similar memory footprint and computational overhead possess higher robustness.

To summarize, we highlight our main contributions as follows:

(1) We consider the robustness of the pruned model in a broader sense. In light of this, we propose an adversarial pruning scheme, which can prune the model without causing obvious degradation in the model's performance, including its robustness.



(a) Common scheme when compressing models by pruning.



(b) Our adversarial pruning scheme.

Fig. 2. An illustration of the common pruning scheme and our adversarial pruning scheme. The main difference is that our scheme involves more data in both the pruning pipeline and the application scene than the common scheme.

(2) To be more flexible in identifying robust channels, we used various means to generate mixed noise samples to simulate dynamically changing environments, which is beneficial to identifying channels that are robust enough to perform well in real-world scenarios. Moreover, we propose a plug-and-play feature scoring module and a novel contribution difference loss to identify robust intermediate features. Based on the robustness of the intermediate features, we construct a gating mechanism to identify and preserve robust channels.

(3) We conduct extensive experiments to evaluate the robustness of pruned models obtained from state-of-the-art methods and our adversarial pruning scheme. The experimental results show that pruned models obtained by our method have higher robustness than other methods. We also conduct ablation studies to demonstrate the effectiveness of each component of the proposed scheme.

II. Related Work

Our work is most closely related to channel pruning, robustness against adversarial attacks, and robustness against natural corruptions. In this section, we provide a brief overview of each of these topics.

A. Channel Pruning

Channel pruning [16], [17], [26], [27] is a popular method that reduces model complexity by removing unimportant channels. Various algorithms are proposed to evaluate the importance of channels. Reference [28] utilizes "smaller-norm-less-important" assumption to prune channels with smaller norm values. However, this hypothesis usually requires two pre-conditions to be satisfied: the distribution deviation of the norm value of the channel is large enough and the norm value of the channel that will be removed should be small enough. Hence, [29] proposes a geometric median based filter pruning method that achieves competitive performance even when the above two pre-conditions cannot be fully satisfied. Reference [30] proposes a novel low-rank guided pruning scheme to obtain skeleton neural networks by alternatively training and pruning CNNs. Reference [31] explores the discriminative feature of feature maps and explicitly investigates the two-adjacent-layer features of each layer to propose a three-phase hierarchical pruning framework. Reference [32] proposed a new method based on LASSO regression to measure the importance of channels by minimizing the feature reconstruction error between the pre-trained feature maps and the compressed ones. Zhuang et al. [16] introduce additional discrimination-aware losses in the fine-tuning and channel selection stages, which was able to increase the discriminative power of the features that are preserved after pruning. Reference [33] proposes to simultaneously address the problems of pruning indicator, pruning ratio, and efficiency constraint. Reference [28] first proposes a soft filter pruning method, which allows pruned filters to be updated during training. Since Batch Normalization (BN) layers are widely used in DNNs, network slimming [18] imposes regularization on the scaling factor γ of the channels in each layer, and then prunes filter with smaller scaling factors. Lin et al. [17] proposed pruning the filters that generate low-rank feature maps.

B. Robustness Against Adversarial Attacks

Adversarial examples were first proposed by Szegedy et al. [34]. An adversarial example is a clean image perturbed by a small distortion to fool a neural network. Popular techniques for improving the robustness of models against adversarial examples are: adversarial training [35], modifying the model structure [36], data augmentation [37] and designing the loss function [38], etc. In this paper, we only discuss the most relevant technique: adversarial training. Adversarial training and its variants improve the robustness of the model by minimizing the training loss on adversarial samples. Goodfellow et al. [39] train DNNs on adversarial examples generated by Fast Gradient Sign Method (FGSM). Besides FGSM, some works have also explored adversarial training based on Projected Gradient Descent (PGD) [40]. Since adversarial robustness and standard accuracy are negatively correlated, to balance the two, TRADES [35] is proposed. Reference [41] proposes a vulnerability suppression loss to minimize the vulnerability of latent features and a Bayesian-based approach to remove latent features with high vulnerability.

Due to the requirements of practical applications, some works have investigated whether model compression and adversarial robustness can be performed together. Reference [40] argues that adversarial robustness is proportional to the

capacity of the model. In contrast, Guo et al. [42] demonstrated that appropriately increasing the sparsity of the model is beneficial to improve the adversarial robustness of the model. Reference [20] proposes a framework of parallel adversarial training and weight pruning to achieve a compact and robust model. HYDRA [21] transforms the weight pruning problem into an empirical risk minimization problem with a robust training objective and solves it by Stochastic Gradient Descent (SGD). Reference [43] experimentally demonstrated that pruned models do not inherit the adversarial vulnerability of original models. Unlike other adversarial pruning methods that are studied only for adversarial attacks, our scheme is not only for adversarial attacks, but also for natural perturbations that are closer to real-world perturbations. In short, we improve the robustness of the pruned model in a broader sense. Furthermore, compared with existing methods that directly identify channels, we identify robust channels from the perspective of identifying robust intermediate features. This flexible mechanism helps the model adaptively identify channels according to the characteristics of dynamic input, resulting in a robust pruned model.

C. Robustness Against Natural Corruptions

In addition to being vulnerable to adversarial attacks, deep models are also vulnerable to natural corruptions, such as impulse noise, defocus blur, and snow. Essentially, natural corruptions is data shift, where models trained on clean data have struggled to perform well on unseen data with distortion. There have been some methods proposed for resolving this problem. To improve generalization performance, some works have explored data augmentation. For example, Cutout [44], CutMix [45] and AugMix [46], etc. Calian et al. [47] proposes adversarial augmentations, which find the parameters of image-to-image models to generate adversarially corrupted augmented images. Guo et al. [48] enhances the weak subnets that show particularly poor robustness on perturbations of the model to improve the robustness of the model on perturbed inputs. Reference [22] demonstrates that data augmentation with Gaussian or Speckle noise can effectively improve robustness against natural corruptions, and designs an adversarial perturbation generator that generates auxiliary noise distributions. Reference [49] proposes to dynamically suppress the most discriminative features of DNNs to yield richer discriminative features, which contributes to the robustness of classification performance. To the best of our knowledge, there are few existing studies investigating the robustness of pruned models against natural corruptions, and most works evaluate the robustness of pruned models only on adversarial attack samples.

D. Relationship Between Adversarial Robustness and Robustness to Natural Corruptions

Based on previous work, we find that these works have no agreement on the relationship between adversarial robustness and robustness to natural corruptions. Some studies have shown that improving adversarial robustness also improves robustness to natural corruptions, but others hold the opposite view. Gilmer et al. [25] claims that improving one type of robustness can also improve another type of robustness. On the other hand, Engstrom et al. [24] found that improving adversarial robustness against ℓ_∞ attacks does not improve robustness against translations and rotations. Reference [22] report adversarially trained model have drops in accuracy on all noise classes of ImageNet-C. Laugros et al. [23] suggests that the robustness of neural networks should be addressed in a broader sense. While existing works have extensively explored a variety of effective approaches to address one of these challenges, only a few works have studied them jointly.

III. Methodology

Our proposed adversarial pruning scheme aims to build robust pruned models by automatically identifying and preserving robust channels. Specifically, we first generate mixed noise samples to simulate dynamic perturbations in real-world scenarios. Then, based on the generated mixed noise samples, we utilize the proposed feature scoring module to dynamically evaluate the robustness scores of intermediate features during training. Besides, we introduce a novel contribution difference loss to identify robust intermediate features more efficiently by aligning contribution scores and making contribution scores sparse. Next, to identify robust channels using the robustness scores of intermediate features, we propose a simple but effective gating mechanism to identify and preserve robust channels. Finally, fine-tuning the pruned model on both clean and noisy samples to recover accuracy and robustness. Repeat the above steps until all the layers or blocks that need to be pruned in the model have been traversed. See Algorithm 1, which gives a more detailed introduction to the pipeline of our proposed method.

In this section, we detail the principles of our scheme. The overall of our proposed scheme is shown in Fig. 3. First, some preliminary contents related to our method are briefly presented in Subsection A. Then, the process of generating mixed noise samples is detailed in Subsection B. Next, the feature scoring module and the contribution difference loss proposed by our work are introduced in Subsection C. To identify robust channels based on robust intermediate features, we construct a gating mechanism to identify and preserve robust channels in Subsection D. Finally, we give the overall objective function of our method in Subsection E.

A. Preliminaries

To better introduce what is in our method, we first formally introduce some preliminaries related to our work. Suppose a model $f_\theta(\cdot)$ with trainable parameters $\theta = \{\mathcal{W}_1, \ldots, \mathcal{W}_L\}$. Under the assumption that bias and rectified linear units are not considered, the l-th layer of the model can be parameterized as $\mathcal{W}_l \in \mathcal{R}^{N_{l+1} \times N_l \times K \times K}$, $1 \leq l \leq L$, where L is the number of convolutional layers in the network, N_{l+1}, N_l and K are the number of output channels, the number of input channels and the size of the convolution kernels of the l-th layer, respectively. Let O_l and O_{l+1} denote the input and output of the l-th layer, respectively. The convolution process of the l-th layer can be represented as $W_l \otimes O_l$, where $\otimes$ represents the

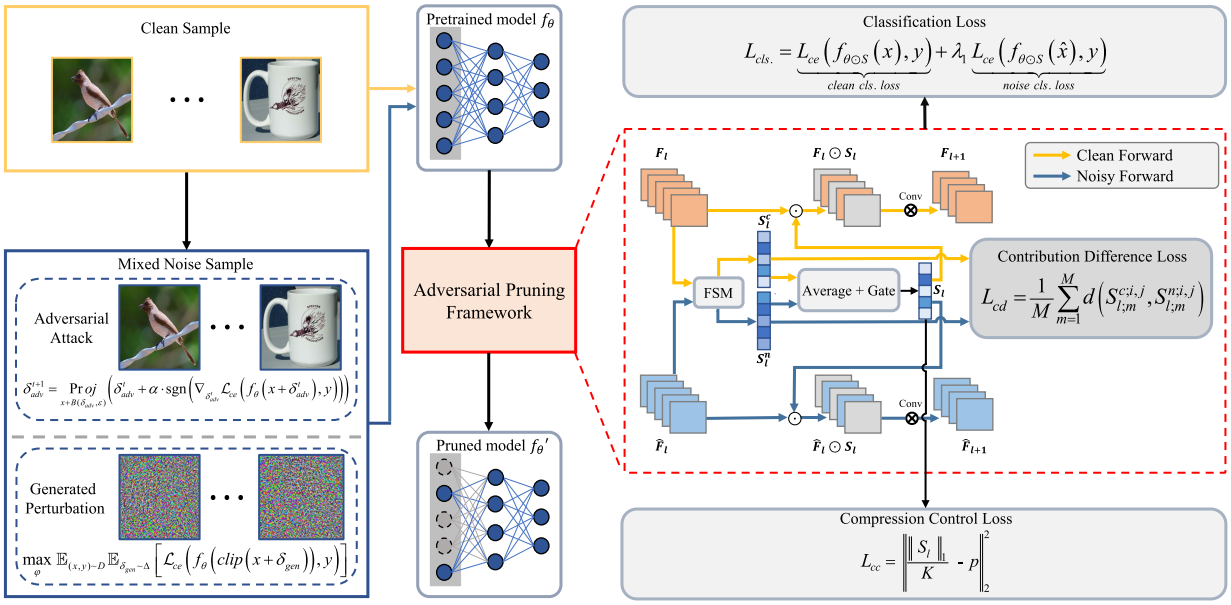


Fig. 3. An overview of our proposed adversarial pruning scheme. Our method consists of two main components: generating mixed noise samples, identifying and preserving robust channels. Multi-type adversarial attacks and generated perturbations from the adversarial perturbation generator are used to obtain mixed noise samples. The feature scoring module (FSM) dynamically evaluates the robustness score (i.e., S_l^c and S_l^n) of intermediate features (i.e., F_l and $\hat{F}_l$), and the contribution difference loss aligns the robustness score of both intermediate features. The robustness score of intermediate features is translated into the robustness score of the channels (i.e., S_l) using the constructed gating mechanism (Average + Gate) to automatically remove the corresponding channels (0: remove), and the compression control loss is introduced to meet the desired compression ratio.

convolution operation. Suppose a dataset $\mathcal{D} = \{X, Y\}$, random sampled input $x \in X$ and its corresponding label $y \in Y$. The prediction of the model can be defined as:

$$y_p = f_\theta(x) = \mathcal{W}_L \otimes (\mathcal{W}_{L-1} \otimes (\cdots (\mathcal{W}_1 \otimes x))), \quad (1)$$

where y_p denotes the final prediction of the model when the input is x. For the vanilla image classification tasks, optimize the parameters of the model by solving:

$$\min_\theta \mathbb{E}_{(x,y)\sim D} \mathcal{L}_{ce}(f_\theta(x), y), \quad (2)$$

where $\mathcal{L}_{ce}$ is the loss function often used for classification tasks, such as cross-entropy loss and 0-1 loss, etc.

Channel pruning aims to optimize the channel layout and parameters to obtain a compact model, most channel pruning methods can be simply formulated as:

$$\min_{\theta,M} \mathbb{E}_{(x,y)\sim D} \mathcal{L}_{ce}(f_{\theta\odot M}(x), y) + \lambda \|M\|_1, \quad \text{subject to } \|M\|_0 \le R, \quad (3)$$

where $M = \{\mathcal{M}_1, \cdots, \mathcal{M}_L\}, \mathcal{M}_l \in \mathcal{R}^{N_{l+1}}, 1 \le l \le L$ is the binary mask used for channel selection, $\theta \odot M = \{\mathcal{W}_1 \odot \mathcal{M}_1, \cdots, \mathcal{W}_L \odot \mathcal{M}_L\}$, where $\odot$ represents element-wise multiplication. λ is a penalty coefficient, R is the number of reserved channels.

Adversarial training is an effective method to improve the robustness of a model. Adversarial training maximizes perturbations by introducing data with perturbations to fool the model, while minimizing the adversarial risk on the overall task. In essence, this is a min-max optimization problem:

$$\min_\theta \mathbb{E}_{(x,y)\sim D} \left[\max_{\|\delta\| \le \varepsilon} \mathcal{L}_{ce}(f_\theta(x+\delta), y) \right], \quad (4)$$

where δ refers to the learned adversarial perturbation, which is limited by the allowed perturbation size ε. $x + \delta$ is a noisy sample. In this paper, noisy samples do not exclusively refer to artificially designed adversarial attacks, but rather refer to various forms of adversarial noise, e.g., adversarial perturbation samples and adversarial attack samples. To obtain a robust pruned model, there are two immediate solutions. One is to obtain an adversarially trained model by using adversarial samples, and then prune it. The other solution is to fine-tune or retrain the pruned model on adversarial samples. However, Ye et al. [20] argue that both solutions have significant challenges. The adversarially trained model is less sparse than the non-adversarially trained model, which makes pruning more difficult. Additionally, the adversarially trained model requires a larger model size to achieve better standard and adversarial accuracy, which makes it difficult to obtain robust pruned models by fine-tuning or retraining pruned models.

B. Generating Mixed Noise Samples

Our objective is to achieve robust pruned models capable of simultaneously addressing natural corruptions and adversarial attacks. This goal distinguishes our work from existing research, as conventional studies tend to focus solely on enhancing robustness in one of these aspects or improving the robustness of unpruned models. Hence, to obtain high-quality adversarial samples, simulate real-world dynamic changes, and thereby enhance the robustness of the pruned model, we propose to generate mixed noise samples.

Randomly sampling a pair of data (x, y) from data distribution $\mathcal{D} = \{X, Y\}$, for the model $f_\theta(\cdot)$, the adversarial attack δ_{adv} on sample x can be obtained by performing multi-step

Algorithm 1 Pipeline of the Adversarial Pruning Scheme

Input: Well-trained model f_θ; Clean data distribution $\mathcal{D}$; Adversarial perturbation generator $\mathcal{G}_\varphi$; Maximum iterations T_i for identifying robust channels; Maximum iterations T_f for fine-tuning the pruned model; Batch size B; Feature scoring module M_F; Number of layers or blocks to be pruned $\mathcal{L}$; Maximum iterations for fine-tuning in the last layer or block T_{fl}.

Output: A robust pruned model $f_\theta{}'$.

for $l = (1, \ldots, \mathcal{L})$ **do**
  Insert the feature scoring module M_F into the l-th layer or block.
  for $t_i = (1, \ldots, T_i)$ **do**
    Randomly sample mini-batches of size B from clean data distribution $\mathcal{D}$.
    Use a portion of the mini-batches to generate adversarial attack samples (Eq. 5).
    Use another portion of the mini-batches to generate adversarial perturbation samples utilizing the designed adversarial perturbation generator $\mathcal{G}_\varphi$ (Eq. 6).
    Fed the sample mini-batches of size B and the corresponding mixed noise samples into the model f_θ.
    Use the feature scoring module and the contribution difference loss to evaluate the robustness score of intermediate features (Eqs. 7, 8, 9).
    Compute the sparse robustness score of channels (Eq. 10).
    Use the gating mechanism and compression control loss to identify and preserve robust channels (Eq. 11).
  end for
  Perform real pruning on the current layer or block.
  if $l == \mathcal{L}$ **then**
    $T_f \leftarrow T_{fl}$
  end if
  for $t_f = (1, \ldots, T_f)$ **do**
    Fine-tune the pruned model (Eq. 12).
  end for
end for
return A robust pruned model $f_\theta{}'$.

gradient based attacks, such as PGD as follows:

$$\delta_{adv}^{t+1} = \underset{x+B(\delta_{adv},\varepsilon)}{Proj}\left(\delta_{adv}^{t} + \alpha \cdot \operatorname{sgn}\left(\nabla_{\delta_{\mathbf{adv}}^{\mathbf{t}}} \mathcal{L}_{ce}\left(f_\theta\left(x+\delta_{adv}^{t}\right), y\right)\right)\right), \tag{5}$$

where $Proj$ represents a projection operation for projecting the input onto the allowed perturbation set $B(\delta_{adv}, \varepsilon) = \left\{\delta_{adv} \in R^m \mid \|\delta_{adv}\|_p \leq \varepsilon\right\}$, α is the step size of PGD, $\operatorname{sgn}(\cdot)$ returns the sign of the vector, δ_{adv}^t is the adversarial attack at the t-th PGD step. In particular, the initial perturbation δ_{adv}^0 is chosen randomly from the perturbation set $B(\delta_{adv}, \varepsilon)$. After obtaining the adversarial attack δ_{adv}^T of the last PGD step, we add it to the clean input x to obtain adversarial attack samples, i.e., $x_{adv} = x + \delta_{adv}^T$, where T is the maximum number of attack steps, x_{adv} is attack adversarial sample. Inspired by [50], in our work, we introduce multi-type adversarial attack samples obtained by using stochastic multi-type iterative adversarial attacks. Specifically, we randomly choose the type of adversarial attack from ℓ_1 attack, ℓ_2 attack and ℓ_∞ attack to obtain multi-type adversarial attack samples.

To improve the robustness of the model against natural corruptions, we propose a Gaussian noise-based adversarial perturbation generator to generate input-independent perturbation. For a given image x and its corresponding label y, we input $z \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ that is consistent with the shape of x to our generator $\mathcal{G}_\varphi$ with parameters φ. We fix the parameters θ of the model to optimize the generator:

$$\max_{\varphi} \mathbb{E}_{(x,y)\sim D}\, \mathbb{E}_{\delta_{gen}\sim\Delta}\left[\mathcal{L}_{ce}\left(f_\theta\left(clip\left(x+\delta_{gen}\right)\right), y\right)\right], \tag{6}$$

where $\delta_{gen} \in \mathbb{R}^d$ is the output of the generator $G_\varphi(z)$, $\Delta = \left\{\delta_{gen} \in \mathbb{R}^d \mid \left\|\delta_{gen}\right\|_2 \leq \xi\right\}$ denotes that the δ_{gen} is projected onto a norm-ball $\left\|\delta_{gen}\right\|_2 \leq \xi$ by scaling the output of the generator, ξ is the maximum allowed perturbation size, *clip* is to constrain the value of the generated perturbation sample $x_{gen} = x + \delta_{gen}$ to an interval $[0, 1]^d$ to ensure it does not exceed its natural range.

Our designed perturbation generator consists of four convolutional layers with one residual connection from input to output. The first three convolutional layers have ReLU activation and the number of output channels is set to 20. Except for the second convolutional layer, the kernel size of each convolutional layer is 1. The kernel size of the second convolutional layer is 3, and after the convolutional operation, an Instance Normalization Layer is introduced to learn detailed perturbations. It is worth noting that several previous studies [22], [50] have proposed to learn the input-independent perturbation samples using an adversarial perturbation generator to improve the robustness of models. However, these generated perturbation samples are only used independently to improve the robustness of unpruned models against either adversarial attacks or natural corruptions, which is different from our purpose of designing perturbation generator to improve the robustness of pruned models against both adversarial attacks and natural corruptions.

To consistently identify robust intermediate features during training, we merge the two noisy samples to obtain mixed noise samples. The specific procedure for obtaining mixed noise samples is to take $\eta\%$ of clean samples from each mini-batches to obtain adversarial attack samples, then, use another $(100-\eta)\%$ of clean samples to obtain generated perturbation samples, and finally, merge adversarial attack samples x_{adv} and generated perturbation samples x_{gen} to obtain the mixed noise samples $\hat{x}$.

C. Identifying Robust Features

Inspired by dynamic neural networks, it is possible to enhance efficiency and maintain good accuracy by adjusting the network's structure or parameters based on the input. In our work, a feature scoring module is proposed to evaluate

the robustness of intermediate features dynamically. In contrast to dynamic neural networks, our proposed module uses a pre-trained model to evaluate the robustness of intermediate features. Since the structure and parameters of the pre-trained model do not change drastically during training in our method, the representational power of each neuron does not change drastically. In other words, the properties of the features extracted by each neuron are relatively stable during training. Therefore, for an ideal robust neuron, the features extracted from a pair of clean and noisy samples are typically distinct, but the contribution of these features to the final prediction of the model should be similar. In our work, to identify robust intermediate features, we propose a contribution difference loss to align the contribution of each pair of intermediate features (clean features and noisy features).

We denote the intermediate features of l-th layer by $F_l \in \mathbb{R}^{B\times C\times H\times W}$, where B is the mini-batch size, C, H and W are the number of channels, height and width of the intermediate feature, respectively. First, to avoid high computational overhead, we use a 5×5 global average pooling to reduce the spatial size of every intermediate feature to obtain $F_{l;gap} = Avgpool(F_l) \in \mathbb{R}^{B\times C\times 5\times 5}$. Then, two 3×3 convolution operations and one ReLU activation are performed on the pooled intermediate features to score the robustness of every intermediate feature:

$$S_l^c = Conv_2(ReLU(Conv_1(F_{l;gap}))), \tag{7}$$

where $S_l^c = \left\{ S_l^{c;i,j} \in \mathbb{R}^{1\times 1} | 1 \le i \le B, 1 \le j \le C \right\}$ indicates the robustness score of clean intermediate features of l-th layer. It is worth noting that the feature scoring module will be removed after the robustness of the feature is evaluated and before pruning, which does not bring additional cost.

We utilize the same feature scoring module to evaluate the robustness of intermediate features from clean samples and noisy samples. Based on this, we propose a contribution difference loss to identify robust intermediate features. When given a pre-trained model, since the representational power of neurons is fixed, even though the features extracted by the same neuron on different inputs may be very different, i.e., the extracted features contain different information, but these features should show similar contributions to the final predictions of the model. Motivated by this, we propose to align the contribution between clean features and noisy features of each layer of the network.

In our work, in each training iteration, a mini-batch of clean samples and a mini-batch of noisy samples are fed into the model at the same time. As mentioned above, the output of the feature scoring module, i.e., the robustness score of intermediate features: S_l^c and S_l^n, can be used to indicate the contribution of the corresponding intermediate features to the final prediction. Therefore, based on our proposed feature scoring module, we define the contribution of features from clean samples and noisy samples to the final prediction of the model as $S_l^{c;:,:}$ and $S_l^{n;:,:}$, respectively. Thus, the difference between the contribution of clean features and noisy features of l-th layer to the final prediction can be formulated as:

$$d\left(S_l^{c;i,j}, S_l^{n;i,j}\right) \overset{\text{def}}{=} \frac{1}{B\times C}\sum_{i=1}^{B}\sum_{j=1}^{C}\left\| S_l^{c;i,j} - S_l^{n;i,j} \right\|, \tag{8}$$

where d indicates the Manhattan distance. In particular, for ResNet and MobileNet, multiple feature scoring modules are simultaneously inserted in the block that will be pruned. Therefore, we consider all inserted feature scoring modules to evaluate the difference between the contribution of clean and noisy features in the entire block:

$$L_{cd} \overset{\text{def}}{=} \frac{1}{M}\sum_{m=1}^{M} d\left(S_{l;m}^{c;i,j}, S_{l;m}^{n;i,j}\right), \tag{9}$$

where M is the number of feature scoring modules inserted in the block.

D. Identifying and Preserving Robust Channels

In order to identify robust channels using robust intermediate features, we construct a simple yet effective gating mechanism. We first average the robustness scores of intermediate features for clean samples S_l^c and corresponding noisy samples S_l^n, i.e., $S_{l;m} = mean(S_l^c, S_l^n)$, where *mean* indicates that the operation is to average $S_l^c \in \mathbb{R}^{B\times C\times 1\times 1}$ and $S_l^n \in \mathbb{R}^{B\times C\times 1\times 1}$ to obtain the robustness score $S_{l;m} \in \mathbb{R}^{1\times C\times 1\times 1}$ of channels. Then, we introduce a scaled gate function that makes the robustness score of channels sparse:

$$S_l = 0.5 \times \left(\tanh\left(\beta \times S_{l;m}\right) + 1\right), \tag{10}$$

where S_l is the sparse robustness score of channels in the $l-$th layer, $S_l \in \mathcal{R}^{N_{l+1}}, 1 \le l \le L$, β is a hyper-parameter that controls the strength of the scaled gate to limit the output. As the value of β gradually increases, the scaled gate function will output a nearly binary robustness score of channels. Next, we apply a smaller global pruning threshold $\mathcal{T}$ to all layers in training to make the smaller channel robustness scores become zero, which helps to pay more attention to the learning of robust channels during training while ensuring the final prediction does not suffer from vulnerable channels. In other words, some channels with robustness scores lower than $\mathcal{T}$ will be identified and pre-removed (set to zero) during training. Further, to provide precise control over the pruning ratio to obtain pruned models with different performances, we introduce a compression control loss function as follows:

$$L_{cc} = \left\| \frac{\|S_l\|_1}{K} - p \right\|_2^2, \tag{11}$$

where K is the length of the sparse robustness score vector S_l, $p \in [0, 1]$ is the percentage of channels expected to be preserved. Finally, the channels corresponding to zero values of the sparse robustness score vector will be removed after training to achieve real pruning.

E. Overall Objective Function

In this subsection, we give the overall objective function. Identifying robust channels or fine-tuning pruned models is

performed on clean samples and corresponding noisy samples. Therefore, our classification loss needs to consider the loss on both clean and noisy samples. Moreover, in order to gradually reduce the interference of less robust channels to the final prediction during training, their learning is stopped according to their sparse robustness score S. The classification loss function is denoted as follows:

$$L_{cls.} = \underbrace{L_{ce}\left(f_{\theta\odot S}\left(x\right), y\right)}_{\text{clean cls.loss}} + \lambda_1 \underbrace{L_{ce}\left(f_{\theta\odot S}\left(\hat{x}\right), y\right)}_{\text{noise cls. loss}}, \tag{12}$$

where $\theta \odot S = (\mathcal{W}_1 \odot S_1, \cdots, \mathcal{W}_L \odot S_L)$, $\hat{x}$ is mixed noise sample that corresponds to clean sample x, $cls.$ is the abbreviation of *classification*, λ_1 controls the magnitude of classification loss on noisy samples. By combining the compression control loss and the contribution difference loss mentioned above, the overall loss is as follows:

$$L_{overall} = L_{cls.} + \lambda_2 L_{cc} + \lambda_3 L_{cd}\ , \tag{13}$$

where λ_2 and λ_3 are hyper-parameters used to adjust the relative magnitude of each loss. To recover the accuracy and robustness of the pruned model, the pruned model is fine-tuned using Eq.12 after the pruning is completed. L_{cc} in Eq. 13 is the compression control loss used to control the pruning ratio. When fine-tuning the pruned model, there is no need to control the pruning ratio, so L_{cc} is not needed. Moreover, L_{cd} in Eq. 13 is the contribution difference loss, which aligns the contribution between clean and noisy intermediate features. The robust channels preserved in the pruned model can extract intermediate features with similar contributions for clean and noisy inputs, thus not requiring L_{cd}.

IV. Experiments

In this section, we empirically study the benefits of our proposed adversarial pruning scheme.

A. Experimental Setup

1) Datasets and Networks: We conduct experiments based on CIFAR-10 [51] and ImageNet [52], in addition, CIFAR-10-C [53], ImageNet-C [53] and ImageNet-R [54] are used to test the robustness of the pruned model. CIFAR-10 contains 50,000 natural images for training and 10,000 natural images for testing, which are categorized into 10 classes. ImageNet contains over one million natural images for training and 50,000 natural images for testing, which are categorized into 1,000 classes. CIFAR-10-C and ImageNet-C are designed to evaluate the robustness of the model against natural corruptions. It was generated by adding 15 different natural corruptions to the original test set images, with 5 levels of severity for each corruption type, resulting in an overall 75 different corruptions. ImageNet-R has renditions of 200 ImageNet classes, resulting in 30,000 images. These renditions include art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, etc. ImageNet-200 is the original ImageNet data restricted to ImageNet-R's 200 classes. They are commonly used to evaluate the out-of-distribution robustness of models. To effectively demonstrate the benefits of our work, several popular network structures are adopted in our experiments. For CIFAR-10, we evaluate our method on three network structures: VGG-16 [55], ResNet-56 [4] and ResNet-110 [4]. For ImageNet, we evaluate our method on three network structures: ResNet-18, ResNet-50 and MobileNet-V2 [56]. Since our method is based on pre-trained models, all network structures will be pre-trained before being fed into the adversarial pruning scheme.

2) Evaluation Metrics: To compare our proposed method with some state-of-the-art pruning methods, we evaluate the performance of the pruned model by multiple metrics. Parameters and FLOPs are employed to evaluate the model size and computational overhead. For CIFAR-10, we use Top-1 accuracy to evaluate the classification performance of the model. For ImageNet, we use Top-1 accuracy and Top-5 accuracy to evaluate the classification performance of the model on a large-scale dataset. To evaluate the robustness against adversarial attacks, we evaluate the attack accuracy of the model under multiple types of PGD adversarial attacks. e.g., ℓ_∞, ℓ_1, and ℓ_2 attack. For CIFAR-10, we use $\varepsilon = \left\{\frac{2000}{255}, \frac{128}{255}, \frac{8}{255}\right\}$ and $\alpha = \left\{\frac{255}{255}, \frac{25}{255}, \frac{1}{255}\right\}$ for ℓ_1, ℓ_2 and ℓ_∞ attacks respectively. For ImageNet, we use $\varepsilon = \left\{\frac{2000}{255}, \frac{80}{255}, \frac{4}{255}\right\}$ and $\alpha = \left\{\frac{255}{255}, \frac{25}{255}, \frac{1}{255}\right\}$ for ℓ_1, ℓ_2 and ℓ_∞ attacks respectively. For CIFAR-10 and ImageNet, we use 200 steps and 20 steps of PGD attack for attack, respectively, during evaluation. Moreover, to evaluate the robustness of the pruned model against natural corruptions, we adopted mean Corruption Error (mCE) on the CIFAR-10-C and ImageNet-C. mCE is the mean of the corruption error of the pruned model over all corruptions. In particular, for ImageNet-C, the corruption error (CE) is defined as:

$$\mathrm{CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}}\right), \tag{14}$$

where $E_{s,c}^f$ and $E_{s,c}^{\mathrm{AlexNet}}$ are the Top-1 errors of model f and AlexNet for a corruption c with severity s, respectively. For ImageNet-R, we use Top-1 accuracy to evaluate the out-of-distribution robustness of the pruned model. The batch size for all the robustness tests mentioned above is set to 256.

It is worth noting that existing studies [20], [22], [35] have shown that adversarially trained models often struggle to achieve similar standard accuracy compared to non-adversarially trained models, and this phenomenon is especially evident for pruned models. Therefore, in our work, pruned models obtained by our method are usually slightly weaker than pruned models obtained by other methods in terms of standard accuracy, but standard accuracy is still competitive. We evaluate the performance of pruned models in terms of comprehensive performance. For a fair comparison, we refer to the results reported by other methods on some evaluation metrics: standard accuracy, parameters, and FLOPs.

3) Implementation Details: We use PyTorch [61] to implement the proposed adversarial pruning scheme and train the network with stochastic gradient descent (SGD) with a momentum of 0.9. All experiments are conducted on Nvidia

GeForce RTX 3090 GPU. For the setting of the baseline model on CIFAR-10, the batch size for training, the batch size for testing, weight decay, initial learning rate and epoch are set to 128, 256, 1e-4, 0.1 and 200, respectively. For the baseline model on ImageNet, we use the pre-trained model officially provided by PyTorch as the baseline model. Set η to 30 for generating adversarial attack samples. The remaining samples are used to generate adversarial perturbation samples. For both CIFAR-10 and ImageNet, we set the allowed maximum perturbation size, the learning rate of the adversarial perturbation generator to 135, and 0.001 for generating adversarial perturbations, respectively. For CIFAR-10, we use $\varepsilon = \left\{\frac{2000}{255}, \frac{128}{255}, \frac{8}{255}\right\}$ and $\alpha = \left\{\frac{255}{255}, \frac{25}{255}, \frac{1}{255}\right\}$ for ℓ_1, ℓ_2 and ℓ_∞ attacks respectively. For ImageNet, we use $\varepsilon = \left\{\frac{2000}{255}, \frac{80}{255}, \frac{4}{255}\right\}$ and $\alpha = \left\{\frac{255}{255}, \frac{25}{255}, \frac{1}{255}\right\}$ for ℓ_1, ℓ_2 and ℓ_∞ attacks respectively. We use 10 steps of PGD attack for ℓ_1, ℓ_2 and ℓ_∞ to generate adversarial perturbations. Other hyper-parameters are the same as those in the training of the baseline model. For CIFAR-10, we set the initial learning rate, epoch and pruning threshold to 0.01, 20 and 0.05 respectively to identify and preserve robust channels. To identify and preserve robust channels on ImageNet, we set the initial learning rate, epoch and pruning threshold to 0.001, 12 and 0.05, respectively. After each layer or block to be pruned is pruned, we fine-tune the entire model to recover performance. When fine-tuning, we also set η to 30 to generate mixed noise samples. We set the initial learning rate and epoch to 0.01 and 20 for fine-tuning on CIFAR-10. We set the initial learning rate and epoch to 0.001 and 20 for fine-tuning on ImageNet. Specifically, the epoch is set to 30 for fine-tuning the entire model after the last layer or block that needs to be pruned is pruned. We set p in Eq. 11 to 0.5 to prune the model. Other implementation details are based on specific experiments and network structures, e.g., the value of β in Eq. 10, the values of λ_1, λ_2 and λ_3 in Eqs. 12 and 13. Different network structures are pruned in different manners. For VGG-16, we use the proposed adversarial pruning scheme to perform pruning in a layer-wise manner, and for other network structures, we perform pruning in a block-wise manner. For a fair comparison, we try to use the weights officially provided by other pruning methods. For individual methods that do not provide weights, we reimplement them using the officially provided code, following their implementation details.

B. Results and Analysis on CIFAR-10

To verify the effectiveness of the proposed method, we first conduct experiments on CIFAR-10. We use VGG-16, ResNet-56 and ResNet-110 to compare the pruning results of our adversarial pruning scheme and some state-of-the-art pruning methods, such as L1 Norm [57], Network Slimming [18], Hinge [58], HRank [17], CHIP [1], DCP [16], SFP [28], FPGM [29], FilterSketch [60] and ABCPruner [59].

1) Results and Analysis on VGG-16: VGG-16 is a neural network with plain architecture, which has 16 layers. Therefore, we adopt a layer-wise manner to prune VGG-16. Specifically, according to the proposed adversarial pruning scheme, we insert the proposed feature scoring module after the ReLU activation of each convolutional layer in a layer-wise manner to perform pruning. We take the values of λ_1, λ_2 and λ_3 in Eqs. 12 and 13 as 5e-2, 50 and 5e-2 respectively, and take the value of β in Eq. 10 as 50 to prune VGG-16. The pruning results are shown in Table I. Compared to L1 Norm with the same baseline model, our method provides similar parameters reduction and FLOPs reduction, with significantly better performance in terms of standard accuracy (92.66% *vs*. 90.63%) and robustness (26.35% *vs*. 29.84% in mCE, 81.39% *vs*. 52.87% in ℓ_1 accuracy, 81.33% *vs*. 52.82% in ℓ_2 accuracy, 36.25% *vs* 11.39% in ℓ_∞ accuracy), which demonstrates the superiority of our method over the magnitude-based method. Compared with HRank and ABCPruner, our method yields similar standard accuracy (92.66% *vs*. 92.34% by HRank and 93.08% by ABCPruner), while achieving better mCE (26.35% *vs*. 29.32% by HRank and 28.95% by ABCPruner) and attack accuracy. Compared with Network Slimming, Hinge, CHIP and FPGM, even though our method is slightly lower in standard accuracy, it significantly improves mCE and attack accuracy, especially achieving a significant enhancement in attack accuracy. Hence, our method demonstrates its ability to obtain a robust and compact model on a neural network with a plain structure.

2) Results and Analysis on ResNet-56 and ResNet-110: In contrast to VGG-16, ResNet-56 and ResNet-110 consist of three stages. Moreover, each stage is composed of multiple basic blocks, which contain two convolutional layers and a shortcut connection. We prune ResNet-56 and ResNet-110 in a block-wise manner. Since the existence of shortcut connections requires that the input and output of basic blocks must match in dimension, existing works usually only prune the first convolutional layer in each basic block. In this paper, we also follow this pruning paradigm, so the proposed feature scoring module is inserted after the ReLU activation between the first convolutional layer and the second convolutional layer of each basic block to prune the first convolutional layer. We take the values of λ_1, λ_2, λ_3 and β as 1e-1, 30, 5e-2 and 50 respectively to prune ResNet-56 and ResNet-110. The comparison of pruning results on ResNet-56 and ResNet-110 is also shown in Table I. We first analyze the experimental results on ResNet-56. Obviously, under similar FLOPs reduction(49.53% *vs*. 49.77% by L1 Norm and 54.72% by SFP) and parameters reduction (49.41% *vs*. 49.41% by L1 Norm and 49.41% by Network Slimming) with L1 Norm, Network Slimming and SFP, our method shows excellent performance in standard accuracy (92.18% *vs*. 90.49%, 90.90% and 91.87% by L1 Norm, Network Slimming and SFP, respectively), mCE (28.11% *vs*. 31.71%, 30.14% and 35.41% by L1 Norm, Network Slimming and SFP, respectively) and attack accuracy. Besides, compared to other methods (e.g., Hinge, HRank, DCP and FilterSketch, etc.) with similar parameters reduction and FLOPs reduction, our method significantly improves the robustness of the model against natural corruptions and adversarial attacks, although our method leads to lower but comparable standard accuracy. Next, we analyze the experimental results on ResNet-110. Compared with some methods with similar standard accuracy reduction (0.82% *vs*. 1.08%, 1.72% and 0.79% by L1 Norm, Network Slimming and

TABLE I

PERFORMANCE COMPARISON OF PROPOSED ADVERSARIAL PRUNING SCHEME AGAINST SOME STATE-OF-THE-ART PRUNING METHODS. THE EXPERIMENT IS CONDUCTED USING VGG-16/RESNET-56/RESNET-110 ON CIFAR-10. "ACC." DENOTES THE ACCURACY OF THE MODEL ON CIFAR-10. "ACC.↓" DENOTES THE DROP IN ACCURACY OF THE MODEL ON CIFAR-10 BEFORE AND AFTER PRUNING. "PARAMS.↓" AND "FLOPS↓" DENOTE PARAMETERS REDUCTION AND FLOPS REDUCTION BEFORE AND AFTER PRUNING, RESPECTIVELY. "*" DENOTES THAT THE EXPERIMENT IS EXECUTED BY REIMPLEMENTING THE OFFICIALLY PROVIDED CODE. "A(B)" DENOTES THE RESULT OF THE MODEL AFTER (BEFORE) PRUNING. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

Model	Method	Acc. (%)	Acc.↓ (%)	Params.↓ (%)	FLOPs↓ (%)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
VGG-16	L1 Norm [57]*	90.63(93.27)	2.64	75.00	74.83	29.84(27.48)	52.87(56.33)	52.82(56.31)	11.39(18.69)
	Network Slimming [18]*	93.24(93.27)	0.03	76.50	33.74	29.32(27.48)	56.75(56.33)	56.70(56.31)	16.32(18.69)
	Hinge [58]	93.59(94.02)	0.43	80.05	39.07	28.45(26.18)	58.45(60.36)	58.35(60.31)	12.71(19.63)
	HRank [17]	92.34(93.96)	1.62	82.10	65.30	29.32(27.25)	52.03(56.95)	51.93(56.87)	12.08(20.05)
	CHIP [1]	93.86(93.96)	0.10	81.60	58.10	26.57(27.25)	58.96(56.95)	58.86(56.87)	19.12(20.05)
	ABCPruner [59]	93.08(93.02)	-0.06	88.68	73.68	28.95(27.52)	66.89(55.35)	66.84(55.30)	24.58(17.97)
	FPGM [29]	93.23(93.58)	1.35	-	35.90	28.01(27.46)	67.39(56.73)	67.28(56.68)	28.42(19.93)
	Ours	92.66(93.27)	0.61	74.73	74.91	**26.35**(27.48)	**81.39**(56.33)	**81.33**(56.31)	**36.25**(18.69)
ResNet-56	L1 Norm [57]*	90.49(93.11)	2.62	49.41	49.77	31.71(27.91)	50.95(54.05)	50.87(53.96)	6.88(8.58)
	Network Slimming [18]*	90.90(93.11)	2.21	49.41	52.26	30.14(27.91)	49.59(54.05)	49.53(53.96)	7.72(8.58)
	SFP [28]	91.87(93.31)	1.44	-	54.72	35.41(27.67)	42.46(52.04)	41.10(52.01)	13.44(14.86)
	Hinge [58]	93.69(92.95)	-0.74	51.27	50.00	29.18(26.18)	53.04(54.69)	52.99(54.66)	9.31(8.16)
	HRank [17]	93.17(93.26)	0.09	42.40	50.00	29.02(28.92)	52.44(55.94)	52.36(55.82)	8.29(9.16)
	DCP [16]	93.49(93.80)	0.31	49.41	49.75	28.16(26.21)	53.07(56.72)	53.03(56.58)	8.21(11.47)
	CHIP [1]	94.16(93.26)	-0.90	42.80	47.40	28.77(28.92)	51.39(55.94)	51.31(55.82)	9.71(9.16)
	FilterSketch [60]	93.19(93.26)	0.07	41.20	41.50	30.40(28.92)	64.64(62.29)	64.52(62.19)	11.97(13.57)
	ABCPruner [59]	93.23(93.26)	0.03	54.20	54.13	30.96(28.92)	60.06(55.93)	59.90(55.81)	17.70(9.08)
	Ours	92.18(93.11)	0.93	49.41	49.53	**28.11**(27.91)	**81.97**(54.05)	**81.92**(53.96)	**28.76**(8.58)
ResNet-110	L1 Norm [57]*	90.95(92.03)	1.08	49.71	49.86	25.09(23.15)	55.63(57.38)	55.57(57.32)	15.71(19.96)
	Network Slimming [18]*	90.31(92.03)	1.72	48.55	53.11	27.92(23.15)	51.88(57.38)	51.83(57.32)	11.15(19.96)
	SFP [28]	93.15(93.94)	0.79	-	43.49	39.74(26.38)	53.13(56.50)	52.96(56.43)	10.90(14.35)
	HRank [17]	94.23(93.50)	-0.73	39.40	41.20	27.04(26.15)	54.88(56.66)	54.83(56.56)	8.93(9.79)
	CHIP [1]	94.44(93.50)	-0.94	48.30	52.10	26.84(26.05)	54.56(56.66)	54.49(56.56)	11.23(9.79)
	FilterSketch [60]	93.44(93.50)	0.06	59.90	63.30	29.31(26.18)	62.76(67.66)	62.68(67.60)	10.75(25.26)
	ABCPruner [59]	93.58(93.50)	-0.08	67.41	65.04	27.92(26.15)	65.53(56.66)	65.41(56.55)	12.82(9.81)
	Ours	91.21(92.03)	0.82	50.29	50.76	**24.76**(23.15)	**80.45**(57.38)	**80.41**(57.32)	**36.73**(19.96)

SFP, respectively), our method shows significant superiority in improving robustness against natural corruptions (24.76% *vs*. 25.09%, 27.92% and 39.74% by L1 Norm, Network Slimming and SFP, respectively) and adversarial attacks (e.g., 80.41% *vs*. 55.57%, 51.83% and 52.96% in ℓ_2 accuracy by L1 Norm, Network Slimming and SFP, respectively), which proves that our method is effective in mining the robust features. Similar to what we have found with ResNet-56, since our pruned model is adversarially trained, although our method shows lower but competitive results in terms of standard accuracy compared to some methods, our method outperforms in terms of mCE and attack accuracy. For example, our method is slightly lower than FilterSketch in standard accuracy (91.21% *vs*. 93.44%), but we outperform FilterSketch in both mCE (24.76% *vs*. 29.31%) and attack accuracy (80.45 % *vs*. 62.76 % in ℓ_1 accuracy, 80.41 % *vs*. 62.68 % in ℓ_2 accuracy, 36.73 % *vs* 10.75 % in ℓ_∞ accuracy). Therefore, our proposed adversarial pruning scheme can also be applied to neural networks with residual blocks.

Moreover, from Table I, we observe that except for our method and CHIP (e.g., CHIP reduces mCE from 27.25% to 26.57% on VGG-16 and from 28.92% to 28.77% on ResNet-56), other methods show a decrease in performance in terms of mCE. This again proves that most existing pruning methods can harm the robustness of the model. This also demonstrates that the pruned model obtained from CHIP by measuring the channel independence of the feature maps has better robustness against natural corruptions. There are some methods (e.g., Network Slimming, CHIP, ABCPruner, FilterSketch and FPGM) that can slightly improve the robustness of the pruned model against adversarial attacks compared to the original model. We analyze that this may be related to the pruning method, the pruning ratio or pruning threshold, network structure and datasets. For example, it may be due to they introduce noise during pruning, which increases the robustness of the model, or the lower complexity of the dataset involved. This phenomenon will disappear with the increase of the pruning ratio, which is proved in Fig. 4. Furthermore, subsequent experimental results on ImageNet and complex networks will further show that this phenomenon is difficult to occur on complex datasets and networks. Besides, Khoury and Hadfield-Menell [65] experimentally demonstrated that since in general there is no single decision boundary that can be optimally robust in all norms, there is no necessary connection between robustness under different norm attacks. For example, pruned models obtained by some methods exhibit a decrease in ℓ_1 accuracy and ℓ_2 accuracy, but an improvement in ℓ_∞ accuracy.

C. Results and Analysis on ImageNet

We conduct experiments for ResNet-18, ResNet-50 and MobileNet-V2 on ImageNet to demonstrate the effectiveness

TABLE II

PERFORMANCE COMPARISON OF PROPOSED ADVERSARIAL PRUNING SCHEME AGAINST SOME STATE-OF-THE-ART PRUNING METHODS. THE EXPERIMENT IS CONDUCTED USING RESNET-18/RESNET-50/MOBILENET-V2 ON IMAGENET. "TOP-1 ACC." AND "TOP-5 ACC." DENOTES TOP-1 ACCURACY AND TOP-5 ACCURACY OF THE MODEL ON IMAGENET, RESPECTIVELY. "TOP-1 ACC.↓" AND "TOP-5 ACC.↓" DENOTES THE DROP IN TOP-1 ACCURACY AND TOP-5 ACCURACY OF THE MODEL ON IMAGENET BEFORE AND AFTER PRUNING, RESPECTIVELY. "FLOPS↓" DENOTES FLOPS REDUCTION BEFORE AND AFTER PRUNING. "A(B)" DENOTES THE RESULT OF THE MODEL AFTER (BEFORE) PRUNING. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY. "BASELINE" REFERS TO THE PRE-TRAINED WEIGHTS OFFICIALLY PROVIDED BY PYTORCH

Model	Method	Top-1 Acc. (%)	Top-1 Acc.↓ (%)	Top-5 Acc. (%)	Top-5 Acc.↓ (%)	FLOPs↓ (%)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
ResNet-18	Baseline	69.76	0.00	89.08	0.00	0.00	84.66	25.32	25.31	1.66
	SFP [28]	67.10(70.28)	3.18	87.78(89.63)	1.85	41.80	89.71	9.81	9.80	2.05
	FPGM [29]	68.34(70.28)	1.94	88.53(89.63)	1.10	41.80	93.18	6.12	6.12	1.17
	ABCPruner [59]	67.28(69.66)	2.38	87.67(89.08)	1.41	44.88	89.91	6.51	6.51	1.95
	ManiDP [62]	68.88(69.76)	0.88	88.76(89.08)	0.32	51.00	90.31	7.53	7.52	1.85
	DCP [16]	67.35(69.64)	2.29	87.60(88.98)	1.38	46.14	90.61	21.37	21.37	1.76
	Ours	66.97(69.76)	2.79	87.28(89.08)	1.80	45.92	**89.32**	**43.37**	**43.37**	**3.70**
ResNet-50	Baseline	76.15	0.00	92.86	0.00	0.00	76.70	34.42	34.42	1.56
	SFP [28]	74.61(76.15)	1.54	92.06(92.87)	0.81	41.80	83.84	11.19	11.19	2.36
	FPGM [29]	75.59(76.15)	0.56	92.63(92.87)	0.24	42.20	83.37	7.88	7.88	2.05
	HRank [17]	75.56(76.15)	0.59	92.63(92.87)	0.24	44.80	81.22	8.60	8.59	0.95
	MetaPruning [63]	75.40(76.60)	1.20	-	-	51.22	93.66	22.49	22.49	2.58
	FilterSketch [60]	74.68(76.13)	1.45	92.17(92.86)	0.69	45.50	82.38	8.52	8.52	2.35
	CHIP [1]	75.26(76.15)	0.89	92.53(92.87)	0.34	62.80	81.98	8.32	8.31	0.81
	ABCPruner [59]	73.86(76.01)	2.15	91.69(92.96)	1.27	54.29	82.65	8.33	8.32	2.55
	Polar [64]	75.63(76.15)	0.52	-	-	54.00	83.41	8.51	8.48	1.79
	DCP [16]	74.95(76.01)	1.06	92.32(92.93)	0.61	55.37	**80.96**	29.71	29.71	2.47
	Ours	73.87(76.15)	2.28	91.90(92.86)	0.96	53.39	81.24	**49.12**	**49.12**	**4.63**
MobileNet-V2	Baseline	71.87	0.00	90.28	0.00	0.00	86.18	27.15	27.15	0.77
	MetaPruning [63]	68.20(72.00)	3.80	-	-	55.27	98.54	16.49	16.49	1.56
	ManiDP [62]	71.42(71.80)	0.38	90.28(90.43)	0.15	37.20	95.31	6.68	6.68	1.98
	Polar [64]	71.80(72.00)	0.20	-	-	28.00	90.70	6.46	6.44	1.70
	Ours	70.64(71.87)	1.73	89.37(90.28)	0.91	43.85	**90.45**	**30.03**	**30.01**	**2.87**

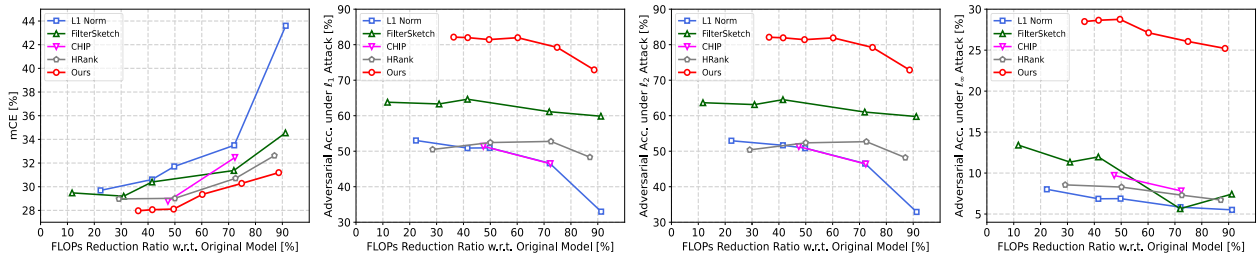


Fig. 4. Robustness comparison of our proposed adversarial pruning scheme against L1 Norm [57], FilterSketch [60], CHIP [1] and HRank [17] under different FLOPs reduction. All pruned models are obtained by pruning ResNet-56 on CIFAR-10. The (a) mCE, (b) ℓ_1 accuracy, (c) ℓ_2 accuracy and (d) ℓ_∞ accuracy of the pruned ResNet-56 under different FLOPs reduction are reported, respectively. Best viewed in color.

of our proposed scheme on challenging datasets. We compare the pruning results of our method with that of some state-of-the-art pruning methods, such as SFP [28], FPGM [29], ABCPruner [59], ManiDP [62], DCP [16], HRank [17], MetaPruning [63], FilterSketch [60], CHIP [1] and Polar [64]. For a fair comparison, our method uses the pre-trained model officially provided by PyTorch as the baseline model.

1) Results and Analysis on ResNet-18 and ResNet-50: We first analyze the experimental results on ResNet-18. Similar to ResNet-56, we also only prune the first convolutional layer in each basic block to prune ResNet-18. We take the values of λ_1, λ_2 and λ_3 in Eqs. 12 and 13 as 1e-1, 60 and 5e-2 respectively, and take the value of β in Eq. 10 as 50 to prune ResNet-18. The pruning results are shown in Table II. Compared with SFP, our method not only has lower Top-1 accuracy drop (2.79% *vs*. 3.18%) and Top-5 accuracy drop (1.80% *vs*. 1.85%), but also outperforms in FLOPs reduction (45.92% *vs*. 41.80%), mCE (89.32% *vs*. 89.71%) and attack accuracy (43.37% *vs*. 9.81% in ℓ_1 accuracy, 43.37% *vs*. 9.80% in ℓ_2 accuracy, 3.70% *vs* 2.05% in ℓ_∞ accuracy). Under similar FLOPs reduction, our method shows significant superiority in both mCE and attack accuracy compared to FPGM, ABCPruner, ManiDP and DCP, especially attack accuracy, although it leads to slightly lower yet comparable standard accuracy. To further verify the applicability of our method, we next analyze the experimental results on ResNet-50. Unlike ResNet-56 and ResNet-18, which are composed of multiple basic blocks, ResNet-50 consists of multiple bottleneck blocks. Each bottleneck block contains a 3×3 convolutional layer and two 1×1 convolutional layers. Following the pruning paradigm of existing methods for ResNet-50, we insert the proposed feature scoring module into after the ReLU activations after the first two convolutional layers of each bottleneck block to prune ResNet-50. We set the value of λ_2 in Eq. 13 to 100, and the

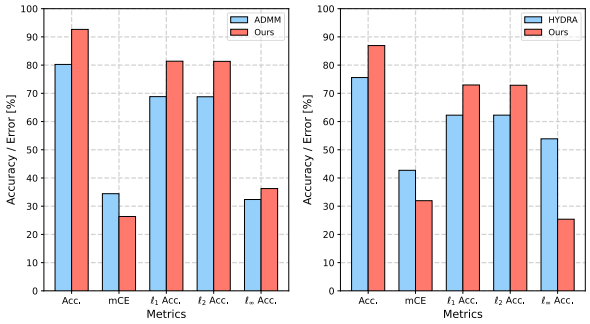


Fig. 5. Performance comparison of our method with two adversarial attack-based pruning methods at similar parameters reduction. Experiments are conducted on CIFAR-10 and VGG-16. Left: Comparison with ADMM [20] at similar parameters reduction (74.73% by Ours *vs*. 76.68% by ADMM). Right: Comparison with HYDRA [21] at similar parameters reduction (98.03% by Ours *vs*. 95.00% by HYDRA). Acc. is the standard accuracy. Best viewed in color.

TABLE III

PERFORMANCE COMPARISON OF PROPOSED ADVERSARIAL PRUNING SCHEME AGAINST SOME STATE-OF-THE-ART PRUNING METHODS. THE EXPERIMENT IS CONDUCTED USING RESNET-50 ON IMAGENET-200 AND IMAGENET-R. "FLOPS↓" DENOTES FLOPS REDUCTION BEFORE AND AFTER PRUNING. TOP-1 ACCURACY IS REPORTED. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY. "BASELINE" REFERS TO THE PRE-TRAINED WEIGHTS OFFICIALLY PROVIDED BY PYTORCH

Method	FLOPs↓ (%)	ImageNet-200 (%)	ImageNet-R (%)
Baseline	0.00	92.12	36.18
DCP [16]	55.37	**91.61**	36.04
ABCPruner [59]	54.29	90.62	34.13
HRank [17]	44.80	91.59	36.03
FilterSketch [60]	45.50	91.29	34.91
CHIP [1]	62.80	91.38	34.97
Ours	53.39	91.36	**36.11**

TABLE IV

PERFORMANCE COMPARISON OF THE PROPOSED ADVERSARIAL PRUNING SCHEME UNDER DIFFERENT NOISE SOURCES. THE EXPERIMENT IS CONDUCTED USING VGG-16 AND RESNET-56 ON CIFAR-10. "ATT." AND "GEN." DENOTE THE ADVERSARIAL ATTACK SAMPLES AND GENERATED PERTURBATION SAMPLES, RESPECTIVELY. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

Model	Att.	Gen.	Acc. (%)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
VGG-16	✗	✗	**92.84**	29.93	57.73	57.68	15.86
	✓	✗	91.56	29.86	**82.29**	**82.27**	**37.25**
	✗	✓	92.12	26.59	52.56	52.50	15.52
	✓	✓	92.66	**26.35**	81.39	81.33	36.25
ResNet-56	✗	✗	91.96	30.83	51.67	51.60	8.18
	✓	✗	91.45	29.97	**82.91**	**82.89**	**31.59**
	✗	✓	**92.36**	**27.89**	50.78	50.70	7.89
	✓	✓	92.18	28.11	81.97	81.92	28.76

TABLE V

PRUNING RESULTS OF RESNET-110 WITH DIFFERENT η ON CIFAR-10. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

η	Acc. (%)	FLOPs↓ (%)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
10	89.83	50.38	24.81	77.54	77.48	34.16
30	91.21	50.76	**24.76**	80.45	80.41	36.73
50	91.47	50.30	24.92	80.87	80.82	36.91
70	**91.53**	50.11	25.37	**80.97**	**80.91**	**36.95**

TABLE VI

PRUNING RESULTS OF RESNET-56 WITH DIFFERENT NOISE DISTRIBUTION ON CIFAR-10. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

Distribution Type	Acc. (%)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
Gaussian Distribution	92.18	**28.11**	**81.97**	**81.92**	**28.76**
Uniform Distribution	**92.21**	29.08	80.43	80.38	27.83
Poisson Distribution	91.36	28.97	80.29	80.25	27.76
Gamma Distribution	91.54	28.26	79.89	79.85	27.25
Exponential Distribution	91.73	29.40	78.51	78.46	26.48

other hyper-parameters to the same settings as when pruning ResNet-18. From Table II, it can be observed that our method outperforms ABCPruner in all aspects of robustness, including mCE (81.24% *vs*.82.65%) and attack accuracy (49.12% *vs*. 8.33% in ℓ_1 accuracy, 49.12% *vs*. 8.32% in ℓ_2 accuracy, 4.63% *vs* 2.55% in ℓ_∞ accuracy), while maintaining similar performance in terms of Top-1 accuracy (73.87% *vs*.73.86%), Top-5 accuracy (91.90% *vs*.91.69%), and FLOPs reduction (53.39% *vs*.54.29%). Moreover, some methods (e.g., HRank and DCP) slightly outperform our method in terms of mCE, but are much lower than our method in terms of attack accuracy, which indicates that our method can improve the robustness of the pruned model across the board. Compared with other methods (e.g., FPGM, MetaPruning and CHIP, etc.), our method can significantly improve the robustness of the pruned model when achieving comparable Top-1 accuracy drop, Top-5 accuracy drop and FLOPs reduction. The experimental results of pruning ResNet-50 on ImageNet demonstrate that our scheme can be effectively applied to complex datasets and network structures to obtain robust pruned models.

2) Results and Analysis on MobileNet-V2: To validate the effectiveness of our proposed method on lightweight models, we perform experiments to prune MobileNet-V2 on ImageNet. MobileNet-V2 consists of multiple blocks, each consisting of three convolutional layers: pointwise convolution, depthwise convolution and pointwise convolution. To accommodate the structure of depthwise convolution, we insert the proposed feature scoring module after the first pointwise convolution, and the generated robustness scores of the channels are shared by the first pointwise convolution and depthwise convolution to prune MobileNet-V2. We take the values of λ_1, λ_2, λ_3 and β as 5e-2, 70, 5e-2 and 50, respectively. The experimental results are also shown in Table II. Compared with ManiDP, our method outperforms in FLOPs reduction (43.85% *vs*.37.20%), mCE (90.45% *vs*.95.31%) and attack accuracy (30.03% *vs*. 6.68% in ℓ_1 accuracy, 30.01% *vs*. 6.68% in ℓ_2 accuracy, 2.87% *vs* 1.98% in ℓ_∞ accuracy) while possessing slightly weaker but comparable Top-1 accuracy (70.64% *vs*.71.42%). Since our method has higher FLOPs reduction (43.85% *vs*.28.00%), it is slightly weaker than Polar in Top-1 accuracy (70.64% *vs*.71.80%), but the pruned model obtained by our method has higher robustness, especially in terms of attack accuracy. Hence, it demonstrates that our proposed adversarial pruning scheme can be effectively applied to lightweight models.

3) More Evaluation Results on ImageNet-R: To further make the conclusions stronger, we conduct experiments on ImageNet-R to compare with some state-of-the-art methods regarding out-of-distribution robustness. The experimental results (in Table III) show that the pruned model obtained using our scheme exhibits superior out-of-distribution robustness.

To sum up, we show that the proposed adversarial pruning scheme has an advantage: the resulting pruned model is more robust compared to other channel pruning methods while maintaining similar standard accuracy, parameters and FLOPs.

D. Comparison With Adversarial Attack-Based Pruning Methods

To further demonstrate the superiority of our method, we compare our method with the pruning method based on adversarial attacks. We choose ADMM [20] and HYDRA [21], two classic pruning methods based on adversarial attacks, for comparison. ADMM is a pruning framework of concurrent adversarial training and model compression based on the Alternating Direction Method of Multipliers, which preserves the adversarial robustness of the model while compressing it. We implement ADMM under the scheme of filter pruning. The experimental results are shown on the left side of Fig. 5, and we can see that our method exhibits superior performance on multiple metrics with similar parameters reduction. HYDRA integrates robust training objective into the pruning technique itself by transforming pruning into an optimization problem to improve both benign and adversarial accuracy. We utilize the weights provided by HYDRA to perform the evaluation. The experimental results on the right side of Fig. 5 show that our method outperforms HYDRA in multiple metrics, except for ℓ_∞ accuracy. These results demonstrate that the setting adopted by our method to identify and preserve robust channels to obtain robust compact models is more effective than adversarial attack-based pruning methods. Dynamically perturbed environments are usually a mixture of multi-distribution shifts, so adversarial attack-based pruning methods struggle to comprehensively improve the model's ability to against multiple types of perturbations, while our method improves the robustness of the pruned model in a broader sense. As can be seen from Table IV, using both adversarial attacks and generated perturbations as noise sources has an overall advantage over using only adversarial attacks.

E. Ablation Study

In this section, we conduct ablation studies about the proposed adversarial pruning scheme. This section is composed of five parts: sources of mixed noise samples, the effect of the trade-off hyper-parameters, FLOPs reduction versus robustness, contribution difference loss and compression control loss, and visualization of the performance of the pruned model.

1) Sources of Mixed Noise Samples: In our work, two different sources of mixed noise samples exist: multi-type adversarial attacks and generated adversarial perturbations. We want to explore the influence of different noise sources on the effectiveness of the proposed adversarial pruning scheme.

TABLE VII

PRUNING RESULTS OF RESNET-56 WITH DIFFERENT β ON CIFAR-10. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

β	Acc. (%)	Params. (M)	FLOPs (M)	mCE (%)	ℓ_1 Acc. (%)	ℓ_2 Acc. (%)	ℓ_∞ Acc. (%)
30	**92.45**	0.43	63.76	28.18	81.39	81.33	27.16
40	92.11	0.43	63.58	28.24	81.41	81.38	27.97
50	92.18	0.43	63.46	**28.11**	**81.97**	**81.92**	**28.76**
60	91.99	0.43	63.71	28.57	81.37	81.34	27.35
70	91.64	0.43	63.90	29.18	81.36	81.31	27.12

TABLE VIII

PRUNING RESULTS OF RESNET-56 WITH DIFFERENT λ_1, λ_2 AND λ_3 ON CIFAR-10. THE BEST AND SECOND-BEST RESULTS ARE HIGHLIGHTED IN **BOLD** AND UNDERLINE RESPECTIVELY

λ_1	1e-2	5e-2	5e-1	1e-1			1e-1		
λ_2	30			10	30	50	30		
λ_3	5e-2			5e-2			1e-2	1e-1	5e-1
Acc. (%)	**92.70**	92.15	91.78	91.98	92.18	92.12	91.89	92.03	91.54
Params. (M)	0.43	0.43	0.43	0.43	0.43	0.43	0.42	0.43	0.43
FLOPs (M)	63.09	62.52	63.61	63.11	63.46	63.31	62.56	63.28	62.98
mCE (%)	29.84	29.12	28.52	28.75	**28.11**	28.64	28.16	28.78	29.19
ℓ_1 Acc. (%)	81.05	81.64	81.78	81.47	**81.97**	81.75	81.91	81.67	81.71
ℓ_2 Acc. (%)	81.01	81.60	81.75	81.45	**81.92**	81.73	81.86	81.63	81.65
ℓ_∞ Acc. (%)	27.24	27.00	27.81	**28.81**	28.76	27.91	27.95	26.65	27.92

Table IV compares the performance of our method on pruning VGG-16 and ResNet-56 under different noise sources. Here, since the parameters and FLOPs of the pruned model are almost constant under different noise sources, they are not reported. As can be seen, in general, using both noise sources, the pruned model achieves excellent performance in terms of both standard accuracy and robustness-related metrics. Besides, it can be observed that the generated adversarial perturbations are significantly effective in improving the robustness of the pruned model against natural corruptions. For example, with generated adversarial perturbations, pruned VGG-16 and ResNet-56 reduce mCE by 3.51% (29.86% without and 26.35% with generated adversarial perturbations) and 1.86% (29.97% without and 28.11% with generated adversarial perturbations), respectively. Moreover, we also observe that using only multi-type adversarial attacks can significantly improve the robustness of the pruned model against adversarial attacks, but compromises the robustness against natural corruptions. One possible reason is that the distributions of samples obtained by adversarial attacks and by natural corruptions are far apart, resulting in models trained on adversarial attacks performing poorly on natural corruptions. This phenomenon is more evident in pruned models.

Given that mixed noise samples consist of two types of noise samples, we conducted experiments to investigate how the ratio of these two types within mixed noise samples affects the results (in Table V). The results show a gradual increase in standard accuracy as η increases. This observation implies that generated perturbation samples have a more significant impact on standard accuracy than adversarial attack samples. Furthermore, it becomes evident that beyond a value of 30 for η, the improvements in several metrics are not significant. Considering the higher time overhead associated with obtaining adversarial attack samples through adversarial attack training compared to generating perturbation samples,

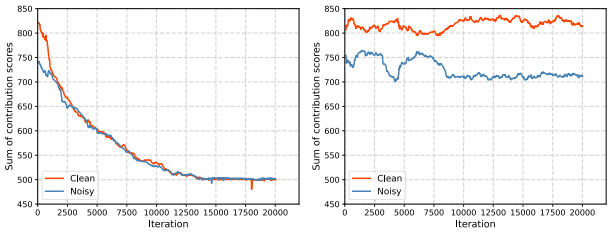


Fig. 6. Illustration of the change in the contribution score of intermediate features during training when pruning ResNet-50 on ImageNet. The red and blue lines represent the sum of contribution scores of intermediate features from clean and noisy samples during training, respectively. Left: With the contribution difference loss. Right: Without the contribution difference loss. Best viewed in color.

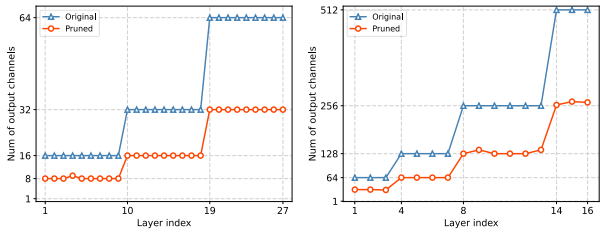


Fig. 7. Comparison of the output channels before and after pruning. Left: Pruning ResNet-56 on CIFAR-10. Right: Pruning ResNet-50 on ImageNet. Best viewed in color.

we have set η to 30 in this work. Furthermore, we explore the effect of employing different noise distributions as inputs to the adversarial perturbation generator (in Table VI). Here, all distributions are utilized in their standard forms. The results indicate that the pruned model exhibits a more comprehensive advantage when utilizing the Gaussian distribution. We attribute this to the favorable properties of the Gaussian distribution, such as the central limit theorem, which allows it to fit complex noise distributions better.

2) The Effect of The Trade-off Hyper-Parameters: To study the effect of using different hyper-parameters, we prune ResNet-56 on CIFAR-10 and evaluate the pruned model using multiple metrics. We mainly study β in Eq. 10, λ_1, λ_2 and λ_3 in Eq. 12 and Eq. 13. The experimental results are shown in Table VII and Table VIII, respectively. From the perspective of the comprehensive performance of the pruned model, we find that the pruned model shows overall superiority in multiple metrics when $\beta = 50$, $\lambda_1 = 1e-1$, $\lambda_2 = 30$ and $\lambda_3 = 5e-2$. We use the same way to select suitable hyper-parameters to fit other network structures and data sets. Moreover, it is easily observed that our method is insensitive to hyper-parameters, which proves the reliability of our method.

3) FLOPs Reduction Versus Robustness: To study the robustness of the pruned model under different FLOPs reduction, we evaluate the proposed method and some methods (e.g., L1 Norm, FilterSketch, CHIP and HRank). We prune ResNet-56 on CIFAR-10 with different FLOPs reduction. We show mCE, ℓ_1 accuracy, ℓ_2 accuracy and ℓ_∞ accuracy in Fig. 4. From Fig. 4 (a), as can be seen, compared with the original model, the mCE of the pruned model may not get worse or even perform better at a lower FLOPs reduction ratio, but its mCE will increase rapidly as the FLOPs reduction ratio increases. This phenomenon, which is not only manifested on mCE, can also be observed on ℓ_1 accuracy, ℓ_2 accuracy and ℓ_∞ accuracy (see Fig. 4 (b)-(d)). This phenomenon shows that a small number of channels are removed, which has little impact on the performance of the model, and when more channels are removed, the robustness or representation ability of the model is impaired dramatically, resulting in a large performance drop. Besides, compared with other methods, the robustness of the pruned model obtained by our method shows superiority at high FLOPs reduction ratio.

4) Contribution Difference Loss and Compression Control Loss: In our work, we propose a contribution difference loss to align the contribution of two types of intermediate features to the final prediction. Here, we choose to prune the first block of ResNet-50 as an example to verify the effectiveness of this contribution difference loss. During the training, we conducted statistical analysis on the layers to be pruned, specifically the sum of contribution scores (i.e., robustness scores) of intermediate features from clean and noisy samples to the final prediction, respectively. The statistical results are shown in Fig. 6. From the left side of Fig. 6, we can see that in the case of using the contribution difference loss, with the progress of training, the sum of contribution scores from both types not only decreases but also gets closer. However, without using the contribution difference loss (see the right side of Fig. 6), the sum of contribution scores from both types is not only difficult to decrease, but also maintains a large gap consistently. This demonstrates that the proposed contribution difference loss not only aligns contribution scores, but also makes contribution scores sparse. Besides, from the distribution of the category scores in Fig. 8, we can also see that the contribution difference loss allows the pruned model to make reliable predictions in the face of perturbed inputs by aligning the contribution of intermediate features to the final prediction. Further, we visualize the number of channels of ResNet-56 and ResNet-50 before and after pruning to verify the effect of the proposed compression control loss. The number of output channels before and after pruning for each layer is shown in Fig. 7. From this figure, we can see that the compression control loss can efficiently control the compression ratio of each layer, which is beneficial to control the compression ratio flexibly to meet the demand.

5) Visualization: We visualize the feature maps w.r.t. the first block in the pruned ResNet-50 and the distribution of the final category scores of the pruned ResNet-50 in Fig. 8. From the results (Middle of Fig. 8), we observe that the pruned model extracts similar and rich information on both clean and perturbed images (e.g., snow and elastic transform). It proves that the proposed adversarial pruning scheme can select the channels with robustness for the network. Besides, we can observe that the pruned ResNet-50 has similar distributions of final category score when faced with clean and perturbed samples (Right side of Fig. 8), especially for the larger classification score. This validates that our proposed adversarial pruning scheme can yield robust pruned models, especially in terms of the relative stability of Top-1 accuracy and Top-5 accuracy when facing perturbed inputs. In other words, the pruned model obtained by our proposed scheme is able to make reliable predictions in the face of perturbed inputs.

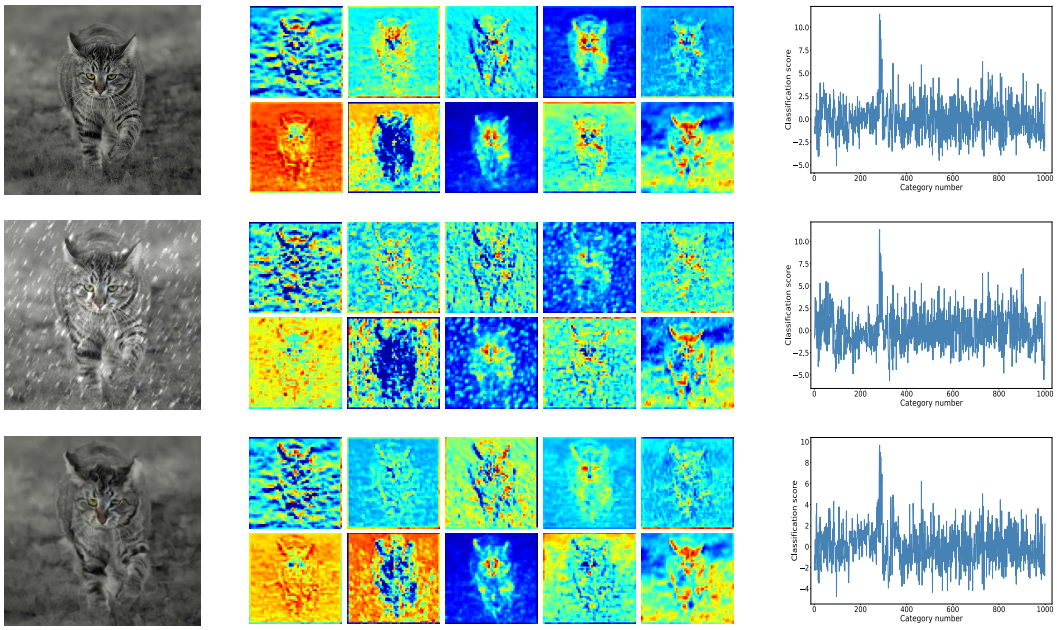


Fig. 8. Visualization of intermediate feature maps and category scores. On the left are different input images, i.e., from top to bottom, clean image, image with snow, and image with elastic transformation, respectively. In the middle is the feature map output from the first block of the pruned ResNet-50. On the right side is the distribution of the category scores output from the FC layer of the pruned ResNet-50. The input images are from ImageNet and ImageNet-C, respectively. Best viewed in color.

V. Conclusion

In this paper, we point out that the existing problem in network pruning is that most pruning methods usually damage the robustness of the model, and the pruned model is difficult to perform well in the real world. To address this issue, we propose an adversarial pruning scheme to compress models to obtain robust pruned models. The scheme is simple yet effective. Extensive experimental results on benchmark datasets demonstrate that the pruned model obtained from the proposed adversarial pruning scheme can make reliable predictions with lower computational overhead than the original model in the face of perturbed inputs.

However, our proposed method also has some limitations. On the one hand, our approach performs pruning in a layer-wise or block-wise manner, which results in high pruning costs. In our future research, we will try to solve this problem with better methods. On the other hand, our method performs pruning with the same pruning ratio per layer or per block, which is not conducive to discovering a more appropriate final structure. In future work, we will try to let the scheme explore a flexible final structure given a desired global pruning rate.

References

[1] Y. Sui, M. Yin, Y. Xie, H. Phan, S. A. Zonouz, and B. Yuan, "Chip: Channel independence-based pruning for compact neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24604–24616.

[2] T. Chen, L. Liang, D. Tianyu, Z. Zhu, and I. Zharkov, "OTOv2: Automatic, generic, user-friendly," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[3] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[6] C. Chen, K. Li, S. G. Teo, X. Zou, K. Li, and Z. Zeng, "Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks," *ACM Trans. Knowl. Discovery From Data*, vol. 14, no. 4, pp. 1–23, Aug. 2020.

[7] M. Liu, S. Jin, C. Yao, C. Lin, and Y. Zhao, "Temporal consistency learning of inter-frames for video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 4, pp. 1507–1520, Apr. 2023.

[8] E.-J. Ong, S. S. Husain, M. Bober-Irizar, and M. Bober, "Deep architectures and ensembles for semantic video classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3568–3582, Dec. 2019.

[9] K. Liao et al., "IR feature embedded BOF indexing method for near-duplicate video retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3743–3753, Dec. 2019.

[10] J. Guo, J. Liu, and D. Xu, "3D-pruning: A model compression framework for efficient 3D action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 12, pp. 8717–8729, Dec. 2022.

[11] S. Xu, S. Zhang, J. Liu, B. Zhuang, Y. Wang, and M. Tan, "Generative data free model quantization with knowledge matching for classification," *IEEE Trans. Circuits Syst. Video Technol.*, early access, May 23, 2023, doi: 10.1109/TCSVT.2023.3279281.

[12] W. Xu et al., "Improving extreme low-bit quantization with soft threshold," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 4, pp. 1549–1563, Apr. 2023.

[13] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11943–11952.

[14] V. Aggarwal, W. Wang, B. Eriksson, Y. Sun, and W. Wang, "Wide compression: Tensor ring nets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9329–9338.

[15] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[16] Z. Zhuang et al., "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018, pp. 1–12.

[17] M. Lin et al., "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.

[18] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.

[19] S. Niu et al., "Towards stable test-time adaptation in dynamic wild world," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023, pp. 1–27.

[20] S. Ye et al., "Adversarial robustness vs. model compression, or both?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 111–120.

[21] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 19655–19666.

[22] E. Rusak et al., "A simple way to make neural networks robust against diverse image corruptions," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 53–69.

[23] A. Laugros, A. Caplier, and M. Ospici, "Are adversarial robustness and common perturbation robustness independent attributes?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–11.

[24] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proc. Int. Conf. Mach. Learn. (ICML)*. PMLR, 2019, pp. 1802–1811.

[25] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 2280–2289.

[26] K.-Y. Feng, X. Fei, M. Gong, A. K. Qin, H. Li, and Y. Wu, "An automatically layer-wise searching strategy for channel pruning based on task-driven sparsity optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 5790–5802, Sep. 2022.

[27] J. Guo, W. Zhang, W. Ouyang, and D. Xu, "Model compression using progressive channel pruning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 1114–1124, Mar. 2021.

[28] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2234–2240.

[29] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4335–4344.

[30] L. Yang, S. Gu, C. Shen, X. Zhao, and Q. Hu, "Skeleton neural networks via low-rank guided filter pruning," *IEEE Trans. Circuits Syst. Video Technol.*, early access, May 18, 2023, doi: 10.1109/TCSVT.2023.3277689.

[31] H. Zhang, L. Liu, H. Zhou, L. Si, H. Sun, and N. Zheng, "FCHP: Exploring the discriminative feature and feature correlation of feature maps for hierarchical DNN pruning and compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 6807–6820, Oct. 2022.

[32] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406.

[33] Y. Li et al., "Weight-dependent gates for network pruning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 6941–6954, Oct. 2022.

[34] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[35] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 7472–7482.

[36] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 501–509.

[37] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving robustness using generated data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 4218–4233.

[38] C. Qin et al., "Adversarial robustness through local linearization," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 1–10.

[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–23.

[41] D. Madaan, J. Shin, and S. J. Hwang, "Adversarial neural pruning with latent vulnerability suppression," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 6575–6585.

[42] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with improved adversarial robustness," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018, pp. 1–10.

[43] A. Jordão and H. Pedrini, "On the effect of pruning on adversarial robustness," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1–11.

[44] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[45] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6022–6031.

[46] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "AugMix: A simple data processing method to improve robustness and uncertainty," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–15.

[47] D. A. Calian et al., "Defending against image corruptions through adversarial augmentations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022, pp. 1–27.

[48] Y. Guo, D. Stutz, and B. Schiele, "Improving robustness by enhancing weak subnets," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022, pp. 320–338.

[49] H. Liu, H. Wu, W. Xie, F. Liu, and L. Shen, "Group-wise inhibition based feature regularization for robust classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 468–476.

[50] D. Madaan, J. Shin, and S. J. Hwang, "Learning to generate noise for multi-attack robustness," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 7279–7289.

[51] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[53] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–16.

[54] D. Hendrycks et al., "The many faces of robustness: A critical analysis of Out-of-Distribution generalization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 8320–8329.

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

[56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[57] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, pp. 1–13.

[58] Y. Li, S. Gu, C. Mayer, L. Van Gool, and R. Timofte, "Group sparsity: The Hinge between filter pruning and decomposition for network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8015–8024.

[59] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," 2020, *arXiv:2001.08565*.

[60] M. Lin et al., "Filter sketch for network pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7091–7100, Dec. 2022.

[61] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 1–12.

[62] Y. Tang et al., "Manifold regularized dynamic network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 5016–5026.

[63] Z. Liu et al., "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3295–3304.

[64] T. Zhuang, Z. Zhang, Y. Huang, X. Zeng, K. Shuang, and X. Li, "Neuron-level structured pruning using polarization regularizer," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 9865–9877.

[65] M. Khoury and D. Hadfield-Menell, "On the geometry of adversarial examples," 2018, *arXiv:1811.00525*.

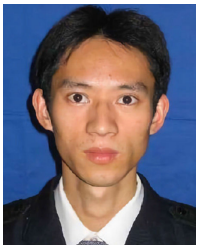
Hui Luo is currently pursuing the Ph.D. degree in signal and information processing with the University of Chinese Academy of Sciences, Beijing, China. His research interests include computer vision, model compression, and acceleration.

Zhuangwei Zhuang received the bachelor's degree in automation and engineering and master's degree in software engineering from the South China University of Technology, Guangzhou, China, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the School of Software Engineering. His research interests include model compression and 3D scene understanding for autodriving.

Yuanqing Li (Fellow, IEEE) received the bachelor's degree in applied mathematics from Wuhan University, Wuhan, China, in 1988, and the master's degree in applied mathematics and Ph.D. degree in control theory and applications from South China Normal University, Guangzhou, China, in 1994 and 1997, respectively. He is currently a Professor with the School of Automation and Engineering, South China University of Technology. His research interests include blind signal processing, sparse representation, machine learning, brain-computer interface, and EEG and functional magnetic resonance imaging data analysis.

Mingkui Tan (Member, IEEE) received the bachelor's degree in environmental science and engineering and master's degree in control science and engineering from Hunan University, Changsha, China, in 2006 and 2009, respectively. He received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2014. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. From 2014 to 2016, he worked as a Senior Research Associate on computer vision at the School of Computer Science, University of Adelaide, Australia. His research interests include machine learning, sparse analysis, deep learning, and large-scale optimization.

Cen Chen (Senior Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, in 2019. He previously worked as a Scientist with the Institute of Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. He currently works as a Professor with the School of Future Technology of South China University of Technology, Shenzhen Institute of Hunan University. He has published more than 60 articles in international conferences and journals on machine learning algorithms and parallel computing, such as HPCA, DAC, IEEE TC, IEEE TPDS, AAAI, ICDM, ICPP, and ICDCS. His research interest includes parallel and distributed computing, machine learning, and deep learning. He has served as a Guest Editor for Pattern Recognition and Neurocomputing.



Jianlin Zhang received the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, China, in 2008. He is currently a Professor with the Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu, China. From 2014 to 2015, he worked as a Visiting Scholar with the Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD, USA. His research interests include machine learning and computer vision.