IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2021, accepted December 2, 2021, date of publication December 6, 2021,
date of current version December 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3133334

You Can't Fool All the Models: Detect Adversarial Samples via Pruning Models

RENXUAN WANG[1], ZUOHUI CHEN[1], HUI DONG[2], AND QI XUAN[1], (Member, IEEE)
[1]Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou 310023, China
[2]College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China

Corresponding author: Qi Xuan (xuanqi@zjut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61973273, and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR19F030001.

ABSTRACT Many adversarial attack methods have investigated the security issue of deep learning models. Previous works on detecting adversarial samples show superior in accuracy but consume too much memory and computing resources. In this paper, we propose an adversarial sample detection method based on pruned models and evaluate four different pruning methods. We find that pruned neural network models are sensitive to adversarial samples, i.e., the pruned models tend to output labels different from the original model when given adversarial samples. Moreover, the pruned model has an extremely small model size and computational cost. Based on the detection result, we further propose a simple but effective defense approach to identify the true label of the adversarial sample. Experiments show that, on average, four different pruning methods outperform the SOTA multi-model based detection method (64.15% and 73.70%) by 28.65% and 18.73% on CIFAR10 and SVHN, respectively, with significantly fewer models used. The FLOPs of our structured pruned model are only 49.41% and 25.62% of the original model. Our defense approach achieves 68.60% and 72.03% average classification accuracy on CIFAR10 and SVHN, exceeding other advanced defense methods.

INDEX TERMS Adversarial sample detection adversarial defense adversarial attack DNN model pruning.

I. INTRODUCTION

Though Deep Neural Networks (DNN) have achieved great success in various applications, e.g., computer vision [1], natural language processing [2], and speech recognition [3], the existence of adversarial samples [4] undermines their use in safety critical areas and raises public concern. The Machine Learning (ML) community has proposed many approaches to improve DNN robustness against adversarial samples, including data augmentation [5], [6], adversarial training [7], [8], and robust optimization [9]. These approaches can improve the robustness of the model to a certain extent, but ask for additional data and training, which cost intensive resources, especially for those large models.

The other optional defense strategy is detecting adversarial samples [10]. The ML community has observed that adversarial samples are different from benign samples in multiple aspects, including data distribution [11], [12], decision boundary [13], and neuron activating path [14]. The model developer can distinguish adversarial samples by these characteristics and stop them from attacking the model. The software engineering community proposes the concept of DNN testing that aims to detect bugs in the DNN model [5], [15], [16]. Adversarial samples are a kind of bug hidden in the DNN model. One of these testing methods for detecting adversarial samples is mutation testing [17]. It generates multiple models by randomly shuffling neuron weights, adding noise to the weights, or inversing the activation state of neurons. They find that the generated models are sensitive to adversarial samples, which means the outputs of generated models are different from the original model. For an unknown sample, through the label changes of generated models, we can distinguish whether it is adversarial.

We argue that distinguishing adversarial samples using multiple models is a more reliable strategy for adversarial sample detection. [18] shows that many defense methods, including improving model robustness or detecting adversarial samples, are vulnerable to certain attacks. In extreme cases, such as the attacker grabbing both model details and defense strategy, these methods can be even circumvented [19]. Most of the detection methods evaluated in [19] use indicators from a single model. Since the indicator comes

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

from the victim model and the perturbation updated in the attack also uses the output of the model or its gradient information, it is possible to design an adaptive attack strategy to circumvent these indicators [19] by adding constraints in the attack objective. For the multiple-model method, the indicator comes from various models, it is hard to guarantee that the perturbation works on all the models.

However, mutation testing is barely practical because the detection requires running dozens of models with almost the same size as the original model. There are many redundant parameters in a neural network, which do not contribute much to reducing errors and generalizing the network during training [20]. Generally, DNN pruning refers to the technique of reducing the parameters that has the least effect on the accuracy of DNN [20]. In this paper, we find that pruned models are more sensitive to adversarial samples. Moreover, pruning also reduces the model size, making the detection feasible in practice. Since the adversarial sample is not able to fool all the models, we can infer the true label from generated models output. We conduct experiments on CIFAR10 [21] and SVHN [22] datasets to prove the effectiveness of our method. CIFAR10 is a small dataset for identifying universal objects, which contains RGB images with 10 categories, and SVHN is a real-world street view house numbers dataset.

Specifically, we make the following contributions.

1) We propose a multi-model based adversarial detection method via pruned models and analysis the influence of different pruning methods on the results. Compare with previous works, our method achieves the highest detection accuracy and at the same time greatly reduces the required number of models and model size, achieving a new SOTA performance.
2) Based on pruned models, we propose a simple but effective multi-model defense approach to predict the true label of the adversarial sample. The defense is based on the detection result and requires no extra DNNs.
3) We evaluate our detection and defense approaches on CIFAR10 and SVHN datasets. Pruned models can detect 92.80% and 92.43% adversarial samples on CIFAR10 and SVHN, with 28.95 and 30.97 pruned models, respectively. The accuracy of defending adversarial samples is 68.60% and 72.03%.

Comparing with our previous work [23], we expanded our work in the following aspects: 1) We investigated more related works and presented them in Section II; 2) We expanded our method with more pruning models, added the corresponding experiment, and discussed the difference in the experiment section; 3) We further proposed a defense strategy based on our detection method, and compare it with other advanced defense methods in the experiment. The rest of paper is organized as follows. Section II presents the related work, including adversarial attacks, adversarial sample detection, and the relationship between sparsity and robustness. Section III introduces how we prune the model to detect and defense adversarial samples. Section IV gives the experimental setup and results. Section V concludes the paper.

II. RELATED WORKS

A. ADVERSARIAL ATTACKS

We evaluate 7 most commonly used adversarial attacks in this work, including Fast Gradient Sign Method [24] (FGSM), Jacobian-based Saliency Map Attack [25] (JSMA), Carlini Wagner Attack [26] (CW), Deepfool Attack [27] (DF), Intermediate Level Attack [28] (ILA), Feature Importance-aware Attack [29] (FIA), and Local Search Attack [30] (LS).

1) FAST GRADIENT SIGN METHOD

FGSM assumes that DNN models are too linear to resist adversarial attacks. Most of the DNN models are designed in a linear form to facilitate training and save computational costs. In this case, linear perturbation is sufficient for a successful attack. The process of generating adversarial samples is expressed as follows

$$x' = x + \varepsilon sign(\nabla_x J(x, y)) \tag{1}$$

where x is the original sample, x' is the adversarial sample, J is the loss function, y is the original label, and ε is the step size. Along the gradient direction of the loss function J, FGSM adds noise to make it misclassified.

2) JACOBIAN-BASED SALIENCY MAP ATTACK

JSMA introduces L_0 norm to measure the magnitude of adversarial perturbation, which essentially limits the number of disturbed pixels in the image. The adversarial saliency map in JSMA is based on the forward derivation of the neural network. Assume a DNN model F with given input X, the forward derivative matrix is

$$J_F(x) = \frac{\partial F(X)}{\partial X} = [\frac{\partial F_j(X)}{\partial x_i}] \tag{2}$$

The saliency map is calculated as follows

$$S(X, t)[i] = \begin{cases} 0 & if\ J_{it}(\mathbf{X}) < 0\ or\ \sum_{j \neq t} J_{ij}(\mathbf{X}) > 0 \\ J_{it}(\mathbf{X}) | \sum_{j \neq t} J_{ij}(\mathbf{X}) | & otherwise, \end{cases} \tag{3}$$

where i is the i-th feature in the input space, t is the target label, j is the original model output label, and $J_{it}(\mathbf{X})$ represents the elements of the forward derivative matrix. The model output confidence on the target label will increase if the corresponding saliency map $S(\mathbf{X}, t)[i]$ increases. The final perturbation is added on the top-2 dominating features (2 pixels) of the input, which is selected by

$$\arg\max (p_1, p_2) \left(\sum_{i=p_1,p_2} \frac{\partial F_t(\mathbf{X})}{\partial \mathbf{X}_i} \right) \left| \sum_{i=p_1,p_2} \sum_{j \neq t} \frac{\partial F_j(\mathbf{X})}{\partial \mathbf{X}_i} \right|, \tag{4}$$

where p_1 and p_2 are the candidate pixels.

3) CARLINI WAGNER ATTACK

CW attack optimizes the adversarial perturbation with the objective that minimizing the perturbation magnitude and maximizing the target label probability. The problem can be formulated as follows

$$\begin{aligned} &\textit{minimize } D(x, x+\delta), \\ &\textit{such that } C(x+\delta) = t,\ x+\delta \in [0,1]^n \end{aligned} \tag{5}$$

where x is the original image, δ is the perturbation, C is the victim model, and D is the distance measurement function. Because $C(x+\delta) = t$ is highly nonlinear, CW defines an objective function f, if and only if $f(x+\delta) \leq 0$, $C(x+\delta) = t$. For different distance measurement, i.e., L_0, L_2, and L_{inf}, CW uses different objective function f.

4) DEEPFOOL ATTACK

Deepfool aims to find the shortest distance from the normal sample to the classification hyperplane. It first deduces the minimal perturbation for binary classifier, then generalizes the solution to multi-class classifier. We refer readers to [27] for the details of exact derivations.

5) INTERMEDIATE LEVEL ATTACK

ILA increases the perturbation on the middle layer of the neural network to improve the attack transferability and effectiveness. The idea is that increasing the norm of the perturbation generally improves the effectiveness of the attack, however, it contradicts the imperceptibility requirement of adversarial attacks. Perturbations with large norm added on the middle layer of the neural network will not increase the perceptibility of the input, thus attacks can be enhanced with perturbation in the middle layer. ILA enhances adversarial samples generated by other attacks by maximizing disturbance while maintaining the original direction. The objective is

$$L(y_l', y_l'') = -\alpha \frac{||\nabla y_l''||_2}{||\nabla y_l'||_2} - \frac{\nabla y_l''}{||\nabla y_l''||_2} \frac{\nabla y_l'}{||\nabla y_l'||_2} \tag{6}$$

where y_l'' is the enhanced adversarial sample and α is a parameter to balance the disturbance direction and norm.

B. FEATURE IMPORTANCE-AWARE ATTACK

FIA achieves strong transferability by disrupting object-aware features that dominate model decisions. The generated image distracts the focus of the model from the object, making it fail to capture the important features. The object-aware features are obtained through the aggregate gradient. Specifically, FIA uses random transformations to determine object-aware features, since transformed images preserve structure and texture information while non-semantic details vary with the transformation.

1) LOCAL SEARCH ATTACK

Local search attack constructs numerical approximation to the network gradient based on greedy local search and then uses it to perturb a small part of pixels in the image. The optimization problem is to minimize the confidence on the original image label and generate the smallest possible disturbance. In each round of the optimization, local search attack computes an implicit approximation to the gradient of the current image by understanding the influence of the selected pixels on the output, then uses the approximated gradient to update the image.

C. ADVERSARIAL SAMPLE DETECTION

Adversarial sample detection has been proved to be effective in the arms race against adversarial attacks. The ML community has found many differences between benign and adversarial samples. Zhao *et al.* [31] found that adversarial samples are less robust than benign samples. By attacking the input sample and evaluating the cost, they can identify the less robust inputs as adversarial. Yin *et al.* [13] observed that adversarial samples are close to the decision boundary. They partition the input space into subspaces and train binary classifiers in the subspaces by symmetrical adversarial training to identify adversarial inputs. Ma and Liu [14] found that adversarial samples will activate abnormal neurons in the forward propagation. They define the distribution of the activation values of two consecutive layers as the provenance invariant of the layers. Then, adversarial samples can be identified by checking invariant violations during DNN computation.

However, in the worst case, i.e., the attacker grabs the full model information and defense mechanism, detection can be bypassed with the specially designed objective. Carlini and Wagner [19] tested ten adversarial detection methods in the past years. They found that most of them can be bypassed with adaptive attacks, which means the attacker designs specific attack optimization targets for specific defense mechanisms. Most of the evaluated methods identify adversarial samples based on a single attribute of the model, so they are easy to break. We argue that adversarial samples are not able to fool all the models since they have different decision boundaries. We can exploit its unrobustness and identify it with multiple models.

As far as we know, Model Mutation Testing [17] (MMT) is the only approach to detect adversarial samples using multiple completely different models (not part of the original model). The other multi-model detection methods use models intercepted from the original model [16]. The authors propose to build sub-models with the parameters and structure inherited from the original model and use them to detect adversarial samples. The number of sub-models can be as many as the number of intermediate layers of the original model. They argue that a normal sample should be predicted with increasing confidence, which reflects on the output of sub-models. Their method needs to retrain an output layer based on the inherited layers, but the inherited layers are frozen during training. In MMT [17], the authors propose to generate mutated models through four operators, namely weights fuzzing, weights shuffling, neuron switch, and neuron activation inverse. The operators may cause a significant decrease in accuracy, and the generated models are sensitive

to adversarial samples. Their method usually takes dozens of models to complete one detection, thus it is not practical in real applications.

III. METHOD

A. SPARSITY AND ROBUSTNESS

Recent works on sparse DNN models [32]–[35] have proved that pruning significantly promoted the robustness of DNN models. These works focus on producing a more robust model, while our work focus on detecting adversarial samples, and our defense is based on multiple models instead of a single model. Wang *et al.* [36] showed that pruned DNNs with high compression rates are more vulnerable to adversarial attacks, which can be avoided through adversarial training. Their conclusion is not general since their experiments are conducted with a simple three-layer CNN on MNIST dataset [37]. Wang *et al.* [38] proposed a robustness enhancement method combined with model pruning and logits augmentation. Guo *et al.* [32] revealed the relationship between sparsity and robustness. They found that sparse linear classifier behaves differently under $l_{\inf}$ and l_2 attack, and the higher model sparsity is, the better the robustness of nonlinear DNNs. Matachana *et al.* [39] found that compressed models are resilient to universal adversarial perturbations, which can generalize across various different inputs. They observed that the robustness of compressed models is also application dependant, i.e., the dataset has a significant influence on the compressed model performance.

Generally, a DNN $f_\theta()$ is trained by minimizing the loss $L()$ over the training dataset. The optimization problem is

$$argmin_\theta \mathbb{E}_{(X,Y)\sim\mathcal{D}}[L(\theta, X, Y)], \tag{7}$$

where X and Y are the training images and labels, following the data distribution $\mathcal{D}$, and θ is the model parameters. The adversary aims to construct adversarial sample by

$$\begin{aligned} &minimize D(\varepsilon) \\ &s.t. f_\theta(x+\varepsilon) \neq y, \end{aligned} \tag{8}$$

where ε is the added perturbation, x is an original benign sample, and y is the true label. Model pruning can be divided into structured and unstructured pruning. Structured pruning removes the filters or layers in the model, while unstructured pruning removes individual weights (mask with zero). Model pruning aims to find the smallest model with the least accuracy loss, the optimization strategy is

$$\begin{aligned} &argmin_{\hat{\theta}} L(\hat{\theta}, X, Y) \\ &s.t. \frac{||\hat{\theta}||_0}{||\theta||_0} < 1 - t, \end{aligned} \tag{9}$$

where $\hat{\theta}$ is the pruned model weights, θ is the original model weights, and t is the fraction of pruned weights.

Without loss of generality, we assume all convolutional kernels in the ith layer is the same size. The forward propagation of layer i is initially

$$A_i = W_i * A_{i-1}, \tag{10}$$

Algorithm 1 Framework of Adversarial Sample Detection

Require: Original model: M_o; Pruned models: $M = m_1, m_2, \ldots, m_n$; Unknown sample: X; accept and deny probability ς_a and ς_d; threshold ς_h ;relax scale: σ.

Ensure: The property of sample (adversarial or not).

1: Set the pruned model size S equals to 1, $i = 1$, $p_0 = \varsigma_h + \sigma$ and $p_1 = \varsigma_h - \sigma$;
2: probability ratio $pr = \frac{p_1^z(1-p_1)^{i-z}}{p_0^z(1-p_0)^{i-z}}$
3: **while** $S \leq n$ and $\varsigma_d - \sigma \geq pr \leq \varsigma_a + \sigma$ **do**
4: Feed X into M_o and m_i, obtain the corresponding outputs $M_o(X)$ and $m_i(X)$;
5: Update the the number of models that output different labels z;
6: Update pr;
7: Feed X into m_i;
8: $S = S + 1$ and $i = i + 1$;
9: **end while**
10: **if** $\varsigma_d - \sigma \geq pr$ **then**
11: return X is a clean sample.
12: **end if**
13: **if** $pr \leq \varsigma_a + \sigma$ **then**
14: return X is an adversarial sample.
15: **end if**
16: return X is an adversarial sample.

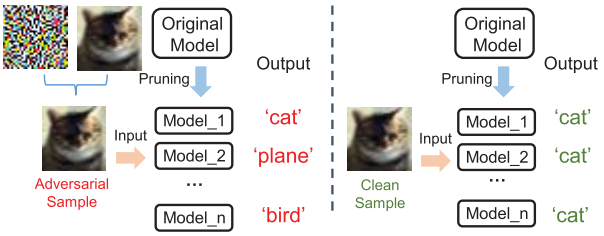


FIGURE 1. The outputs of pruned models when fed with clean sample and adversarial sample.

where A_i and A_{i-1} represent the feature map of the ith and $(i-1)$th layer, $*$ is the convolution operator, and $W_i \in \mathbb{R}^{K,W,H,C}$ is the weights of ith layer. K, W, H, C are the number of kernels, kernel width, kernel height, and kernel depth respectively. After pruning, structured pruning reduces the number of kernels, producing $\hat{W}_i \in \mathbb{R}^{\hat{K},W,H,C}$, while unstructured pruning reduces the kernel width, kernel height, and kernel depth, producing $\hat{W}_i \in \mathbb{R}^{K,\hat{W},\hat{H},\hat{C}}$. With the change of weights shape and retraining of the model, the pruned model learns a different feature space compared with the original model, thus adversarial perturbation is not guaranteed to be effective in the new feature space.

B. ADVERSARIAL SAMPLE DETECTION

We detect adversarial samples by the outputs of pruned models. As shown in Fig. 1, a clean sample with the label cat is still cat in most outputs of additional models, but an adversarial sample makes these models output various labels, e.g., cat, plane, and bird. Because pruning and training rebuild the

decision boundary of the pruned model, making it different from the original model. The diversity of outputs can be measured with Label Change Rate (LCR) and used to identify adversarial samples, which is defined as

$$\varsigma = \frac{\sum_{s\in S} E(f(x), s(x))}{|S|}, \tag{11}$$

where x is the input, $f(x)$ is the original model output, $s(x)$ is the pruned-model output, $|S|$ is the size of used pruned models, and $E(\cdot)$ is defined as

$$E(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise.} \end{cases} \tag{12}$$

A straightforward way to calculate LCR is using the fixed-size sampling test, i.e., adopting a fixed number of models and counting the outputs that are different from the original model. To reduce the computational cost, we use the Sequential Probability Ratio Testing [40] (SPRT) to detect adversarial samples dynamically. The mutual exclusive hypothesis in the testing is

$$\begin{aligned} H_0 &: pr(x) \geq \varsigma_h, \\ H_1 &: pr(x) \leq \varsigma_h, \end{aligned} \tag{13}$$

where ς_h is a threshold determined by the LCR of clean samples (calculated by Eq. (11)). SPRT runs the pruned model successively and calculate the probability ratio pr by

$$pr = \frac{p_1^z(1-p_1)^{n-z}}{p_0^z(1-p_0)^{n-z}}, \tag{14}$$

where z is the number of models that output different labels, n is the total number of used models, $p_0 = \varsigma_h + \sigma$, and $p_1 = \varsigma_h - \sigma$. We set a relax scale σ, which means when the LCR falls in the region $(\varsigma_h - \sigma, \varsigma_h + \sigma)$, neither hypothesis can be denied and the test continues. The accept LCR and deny LCR are defined as follows

$$\begin{aligned} \varsigma_a &= ln\frac{\beta}{1-\alpha}, \\ \varsigma_d &= ln\frac{1-\beta}{\alpha}, \end{aligned} \tag{15}$$

where α and β denote the probability of false positive and false negative, respectively. The test stops when one of the hypothesis is accepted. The input is considered as a clean sample if $pr \geq \varsigma_a$, while it is adversarial otherwise. The complete process is shown in Algorithm 1.

C. PRUNED MODEL GENERATION

Instead of using the original model structure and parameters, we exploit four pruning method (as shown in Fig. 2) to produce sub-models, including Random Channel pruning [41] (RC), L_1 Norm pruning [42] (L_1N), Lottery Ticket Hypothesis [43] pruning (LTH), and Subnetwork Extraction [44] (SE), where RC and L_1N are structured pruning, LTH and SE are unstructured pruning. Compared with the mutated models in MMT [17], the pruned model has a smaller size and is more sensitive to adversarial samples. Our method requires 28.95 and 30.97 models on CIFAR10 and SVHN datasets respectively while MMT needs 62.65 and 51.78. The pruned model runs faster than the mutated model with the FLOPs of 274.9M and 78.6M on CIFAR10 and SVHN respectively, while MMT is 556.65M and 314.03M.

1) RANDOM CHANNEL PRUNING

As shown in Fig. 2(a), channel pruning usually evaluates the importance of different channels of a DNN layer and removes all the input and output connections of the unimportant channels [41], [45], [46]. The advantages of channel pruning include reduction of actual parameters and increase of inference speed. We use random channel pruning to find a set of small models with accuracy close to the original model.

In order to reduce the computational cost while ensuring the pruned models' accuracy and diversity, we set a fixed overall pruning rate for each model and assign a random number of channels to be pruned in each layer. Specifically, every several layers with the same number of channels are divided into a group. A group has an overall pruning rate (e.g., 50%) and layers in a group will be assigned with two random pruning rates (e.g., 30% and 20%), while the sum of which equals the overall pruning rate.

2) L_1 NORM PRUNING

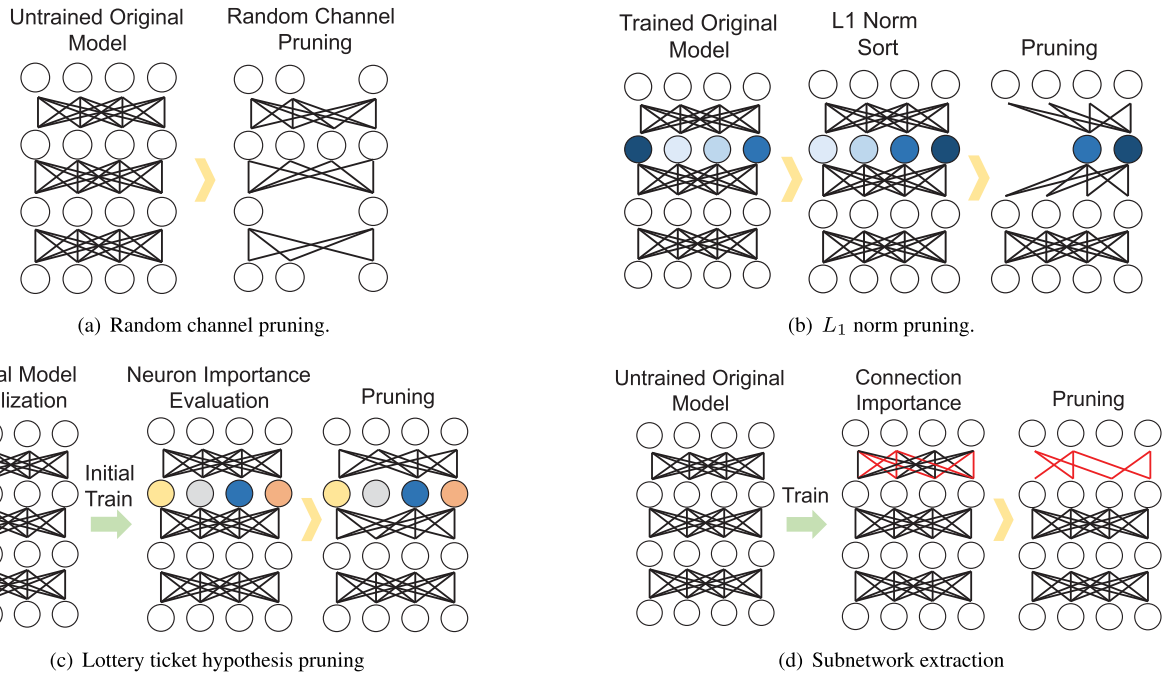
In a DNN model, the relative importance of a filter in each layer can be measured by calculating its L_1 norm [42], i.e., the sum of its absolute weights. The magnitude of the output feature map is mostly determined by this value. The smaller the magnitude, the weaker the activation output, thus pruning the small filters can reduce the model size and maintain the model accuracy.

As shown in Fig. 2(b), for the original model, we sort the filters in each layer with their corresponding L_1 norm. Then, we prune filters for each layer starting with the smallest L_1 norm. The kernels in the next layer corresponding to the pruned feature map are also removed. The pruning rate is randomly set for each layer while keeping the overall pruning rate the same for all models.

3) LOTTERY TICKET HYPOTHESIS PRUNING

The Lottery ticket hypothesis [43] implies that a randomly initialized, dense neural network contains a subnetwork that is initialized such that when trained in isolation it can match the test accuracy of the original network after training for at most the same number of iterations. As shown in Fig. 2(c), it trains a network with a few epochs, then evaluates the importance of neurons and removes those unimportant ones. The subnetwork will be re-initialized with the same parameters, then repeat the training and pruning until meets the pruning rate. The final subnetwork is the winning lottery.

In this work, we evaluate the neuron importance by their absolute value. To produce diverse sub-models, as shown in 2(c), each model is initialized with different parameters and trained 6 epochs after initializing the network. Then we remove the unimportant weights with a fixed pruning rate.



(a) Random channel pruning.

(b) L_1 norm pruning.

(c) Lottery ticket hypothesis pruning

(d) Subnetwork extraction

FIGURE 2. **Pruning methods used in our detection procedure.**

4) SUBNETWORK EXTRACTION

An untrained randomly weighted network contains a subnetwork (with random weights) that has similar performance to a well-trained model with similar size [44]. Ye *et al.* [47] proposed a greedy selection approach to proving that there exists a small subnetwork inside the large network that performs almost as well as the large network. Malach *et al.* [48] proved that in a sufficiently over-parameterized model, there exists a subnetwork with random weights that is roughly the same accuracy as the target network. The subnetwork does not need any further training and it is valid for every target model with bounded weights on every bounded distribution.

We can find these subnetworks by training and learning the importance of the connections between neurons. Specifically, in the forward propagation, if the output of a neuron is aligned with the negative direction of the gradient, we increase the importance of this connection, otherwise, we reduce the importance. As shown in Fig. 2(d), for each model, we randomly set different initial weights and a fixed pruning rate. Then we remove the unimportant connections according to the pruning rate.

There are constraints on the untrained randomly weighted network. The size of the untrained randomly weighted network affects the pruned model accuracy. A wide untrained randomly weighted model produces more accurate subnetwork [44]. A small initial model produces a subnetwork with poor performance [44]. The impact of model size is reflected in the experiment with 2 different models, our method is effective on both of them.

D. ADVERSARIAL SAMPLE DEFENSE

Since the adversarial sample is generated for the original model, in most cases, it can not fool the pruned model. We assume that the true label exists in the output of pruned models. Specifically, in the detection stage, we use a variable number of models to calculate the LCR and identify adversaries. If the input is identified as adversarial, we take the largest category of all model outputs as the true label. Note that only models involved in the calculation of LCR will be counted, thus there is no extra computational cost of any deep learning models.

IV. EXPERIMENTS

A. DATASET AND MODELS

We evaluate our approach on CIFAR10 [21] and SVHN [22]. The former contains 50,000 images for training and 10,000 images for testing, while the latter are 73,257 and 26,032, respectively. The image size of both two datasets is $32 \times 32 \times 3$. We adopt ResNet18 and VGGNet16 for CIFAR10 and SVHN, and their accuracy is 93.03% and 95.63%, respectively.

B. PRUNING SETTING

For channel pruning methods, the pruning rate is set to 50% for all the groups. The overall pruning rate (percentage of parameters and channels removed) is listed in Table 1. The pruning rate is calculated differently according to the pruning type, the structured pruning rate (RC and L_1N) is calculated by the number of removed channels (filters) while the

TABLE 1. Overall Pruning Rates and FLOPs of the Generated Models.

	Method	Channel Pruning Rate	Weight Pruning Rate	FLOPs
CIFAR10	RC	0.5	0.5047	275.2M
	L_1N	0.5	0.5189	274.9M
	LTH	0	0.5	278.3M
	SE	0	0.5	278.3M
	MMT	0	0	556.65M
SVHN	RC	0.5	0.7614	82.3M
	L_1N	0.5	0.7506	78.6M
	LTH	0	0.5	157.0M
	SE	0	0.5	157.0M
	MMT	0	0	314.03M

TABLE 2. Average Model Accuracy of Different Pruning Methods (%).

	original	LTH	SE	RC	L_1N
CIFAR10	93.03	92.20	92.15	89.56	91.65
SVHN	95.63	95.10	94.98	94.34	94.62

unstructured pruning rate (LTH and SE) is determined by the number of individual weights setting to zeros. We give both pruning rates in Table 1. Under this setting, the pruned model accuracy is above 89% as shown in Table 2, while the model size is relatively small. The pruning is finished in one step for RC and L_1N since they do not need an iterative search. For LTH and SE we set the max search steps 6 and 100 respectively, which is empirically enough to find a good subnetwork. Note that LTH still needs training after pruning, same as RC and L_1N. SE does not require training after pruning. The average model size after pruning is listed in Table 3. We compare our approach with the MMT [17], which is a SOTA adversary detection algorithm. There are four mutation operators that can be used to generate mutated models, we choose the best performers for comparison, i.e., Neuron Activation Inversion (NAI). We use the best parameter setting, i.e., the mutation rate is 0.007. Both MMT and our method use SPRT to test the generated models' outputs. For a fair comparison, we set the maximum number of available models in SPRT to 100.

C. ADVERSARIAL SAMPLE GENERATION

We use seven typical adversarial attack methods and set the attack parameters the same as MMT is tested for a fair comparison. The parameters for each attack are summarized as follows:

1) FGSM: the scale of perturbation is 0.03;
2) JSMA: the maximum distortion is 12%;
3) CW: adopt L2 attack, the scale coefficient is 0.6 and the iteration number is 1000;
4) Deepfool (DF): the maximum number of iterations is 50 and the termination criterion is 0.02;
5) One Pixel Attack (OP): the number of pixels for modification is 3 (in order to ensure that enough successful samples are generated) and the differential algorithm runs with a population size of 400 and a max iteration count of 100;

TABLE 3. Average Model Size of MMT and Ours (MB).

	Original	Mutated	Unstructured	Structured
			LTH and SE	RC and L_1N
CIFAR10	85.35	85.35	85.35	43.84
SVHN	112.45	112.45	112.45	27.59

6) Intermediate Level Attack (ILA): the baseline attack is IFGSM, the number of iterations of IFGSM to perform is 10, the learning rate of IFGSM is 0.005, the epsilon of IFGSM is 0.03, the number of ILA to perform is 10, the epsilon of ILA is 0.01, the coefficient of magnitude loss in ILA attack is 1, and the learning rate of ILA is 1.
7) Local Search Attack (LS): the pixel complexity is 1, the perturbation value is 1.5, the half side length of the neighborhood square is 5, the number of pixels perturbed at each round is 5 and the threshold for k-misclassification is 1.
8) Feature Importance-aware Attack (FIA): maximum size of adversarial perturbation is 0.03, number of iterations is 10, step size is 1, momentum is 1, number of random mask input is 30, and randomly mask probability is 0.7.

D. METRICS

We evaluate the detection performance and defense performance with the following metrics. Detection performance is evaluated by how many adversarial samples are detected, while the defense is evaluated by how many adversarial samples are classified correctly when fed with all adversarial samples.

a) AUROC: Our approach takes the LCR of normal samples as the threshold. In order to verify whether the feature is suitable for distinguishing adversarial sample from the normal sample, we calculate the area under the ROC curve to determine whether LCR is an appropriate feature (the closer the AUROC is to 1, the better the feature is).

b) Detection accuracy and the number of models used: in addition to detection accuracy, we also evaluate the number of models required for detection. The higher accuracy and fewer models indicate a better method.

c) Classification accuracy: when fed with adversarial samples, the classification accuracy reflects how strong the defense is. The higher the accuracy, the better the defense ability.

E. RESULTS

As shown in Table 3, structured pruning strategy greatly reduced the model size from 85.35 MB and 112.45 MB to 43.84 MB and 27.59 MB on CIFAR10 and SVHN, respectively. Unstructured pruning and mutation operator do not change the actual model size. The FLoating point OPerations (FLOPs) of the generated models are listed in Table 1. The pruning reduces the number of FLOPs from 556.65M and 314.03M to 276.6M and 118.7M on CIFAR10 and SVHN respectively on average. Note that the mutation operator does

TABLE 4. AUROC of Different Methods and Attacks (%).

	Attack	MMT	RC	L_1N	LTH	SE
CIFAR10	FGSM	87.96	**98.67**	98.50	98.56	98.34
	JSMA	97.07	**99.81**	99.79	99.71	99.66
	CW	88.05	98.07	**98.15**	97.63	97.88
	DF	97.04	99.79	**99.80**	99.73	99.68
	ILA	87.45	**96.98**	96.36	96.61	96.55
	LS	82.78	**96.47**	96.11	94.51	94.17
SVHN	FGSM	91.92	**98.00**	96.47	96.80	96.64
	JSMA	96.43	99.53	99.29	99.55	**99.58**
	CW	86.73	**98.65**	96.94	97.91	98.34
	DF	93.90	**99.19**	98.34	98.97	99.00
	ILA	90.3	**98.89**	97.44	98.62	98.53
	LS	90.38	**98.45**	98.05	96.72	96.46

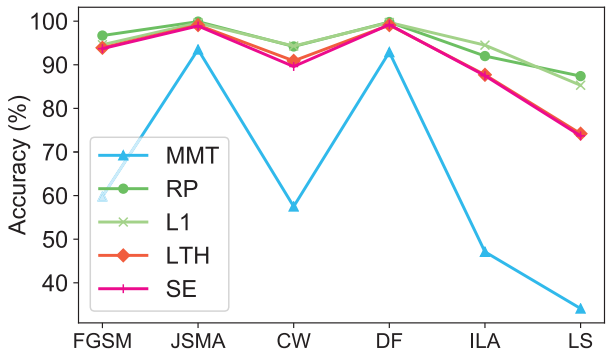


FIGURE 3. The adversarial sample detection accuracy (Accuracy) on CIFAR10.

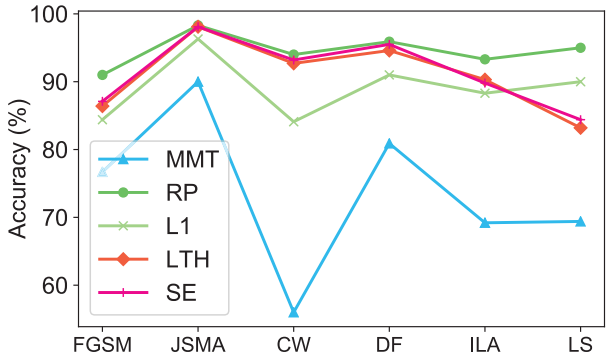


FIGURE 4. The adversarial sample detection accuracy (Accuracy) on SVHN.

not change the model size nor the FLOPs, thus models generated by MMT are the same size as the original model.

AUROC scores are summarized in Table 4 with the best results marked in bold. Pruned models outperform mutated models for all kinds of attacks on CIFAR10 and SVHN. It implies that pruned models are better in distinguishing clean samples and adversarial samples. Among all the attacks, the Local Search (LS) attack has the lowest AUROC, because it is a kind of black-box attack and relatively further away from the decision boundary compared with white-box attacks.

Fig. 3 and Fig. 4 show the adversarial sample detection accuracy. Fig. 5 and Fig. 6 show the number of used models. For all the pruned models, the average detection

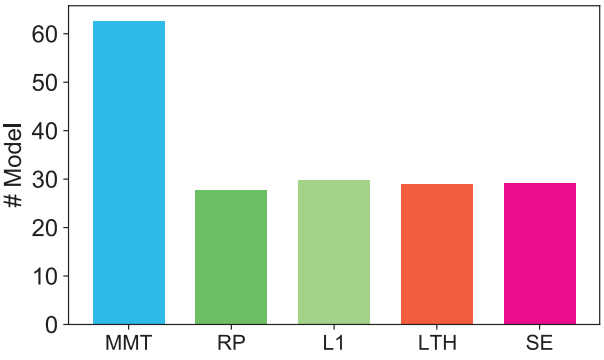


FIGURE 5. The number of used models (#Models) on CIFAR10.

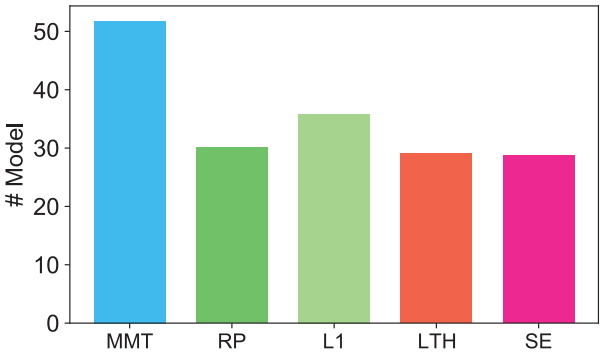


FIGURE 6. The number of used models (#Models) on SVHN.

accuracy on CIFAR10 and SVHN are 92.80% and 92.43%, exceeding MMT (64.15% and 73.70%) 28.65% and 18.73%. All detection methods achieve over 90% accuracy on JSMA and DeepFool generated samples (CIFAR10), which indicates samples generated by the two methods are closer to the decision boundary than other attack methods. MMT shows low detection accuracy, which implies the mutation operator generated models are less diverse on par with pruning. All the detection methods show the lowest detection accuracy on Local Search attack (CIFAR10), which is consistent with the result of AUROC. It indicates that LS generated samples are the farthest from the decision boundary compared with other attacks. Overall, Random Channel pruning has the highest detection accuracy of 95.00% and 94.58% overall attacks on CIFAR10 and SVHN, respectively.

When applied to clean samples, our method shows a low false positive rate, i.e., most of the clean samples are correctly classified as clean samples. As shown in Table 5, we test MMT and our method with 1,000 clean samples. All pruning methods achieve detection accuracy over 85% on clean samples. On average, our approach uses only 28.95 and 30.97 pruned models for all attacks, while MMT requires 62.65 and 51.78 mutated models for CIFAR10 and SVHN, respectively.

To demonstrate the robustness of our method, we test our method with adaptive attacks. For detections using a single indicator or another classifier, Carlini *et al.* [19] showed that they can be circumvented by adding new optimization objects in the generation of perturbation. However, for multi-model

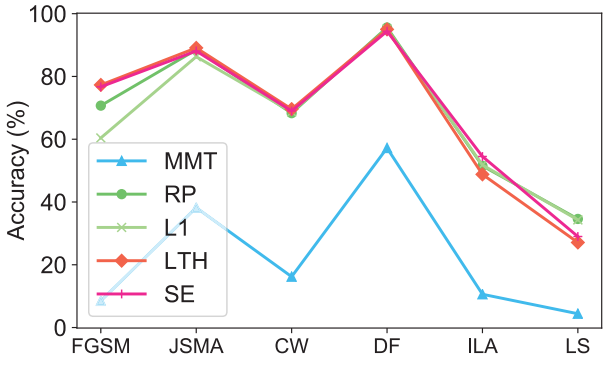


FIGURE 7. The classification accuracy (Accuracy) with defense on CIFAR10.

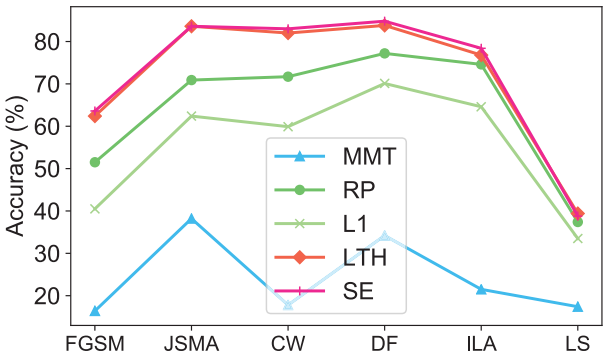


FIGURE 8. The classification accuracy (Accuracy) with defense on SVHN.

based detection, there is no explicit indicator to optimize the perturbation. We argue that the potential threat for multi-model based detection is transferable attacks, which may be able to fool all the pruned models. We utilize the most advanced transferable attacks, i.e. Feature Importance-aware Attack [29] (FIA) under white-box setting to evaluate the detection performance. The attacker knows the model parameters and uses model gradients. We generate 905 and 545 adversarial samples for CIFAR10 and SVHN respectively. The experimental result is shown in Table 6. Pruned models detect 95.13% and 92.33% adaptive adversarial samples on CIFAR10 and SVHN respectively. There is no performance decrease compared with detecting other adversarial samples. MMT shows a significant decrease in performance, with the detection accuracy of 48.25% and 69.9% on CIFAR10 and SVHN respectively.

We compare our defense method with four advanced defense algorithms [49]–[52]. Guo *et al.* [49] used models trained on Transformed Images (TI) to improve robustness. Their transformation contains image cropping and rescaling, bit-depth reduction, JPEG compression, total variance minimization, and image quilting. Madry *et al.* [50] proposed Projected Gradient Descent (PGD), which is the strongest first-order adversarial attack method. Models trained on PGD generated images will be robust to other first-order adversarial attacks. Zhang *et al.* [51] designed a defense method, TRADES, to trade adversarial robustness off against accuracy. Their methodology won

TABLE 5. Detection Accuracy on Clean Samples (%).

	MMT	LTH	SE	RC	L_1N
CIFAR10	81.00	87.70	87.40	85.90	86.40
SVHN	92.30	95.70	95.40	95.10	95.50

TABLE 6. Detection Accuracy Against Adaptive Attacks (%).

	MMT	LTH	SE	RC	L_1N
CIFAR10	49.4	91.4	90.7	100.0	98.4
SVHN	70.6	92.7	92.0	94.3	90.3

TABLE 7. The classification accuracy with defense on CIFAR10 and SVHN (%).

	TI	PGD	TRADES	FGSM-RI
CIFAR10	68.30	61.90	60.20	49.8
SVHN	67.62	60.70	54.80	66.50
	MMT	Ours (SE)	\	\
CIFAR10	22.51	**68.60**	\	\
SVHN	24.26	**72.03**	\	\

TABLE 8. Defense Against Adaptive Attacks (%).

	MMT	LTH	SE	RC	L_1N
CIFAR10	2.4	5.1	6.3	6.2	6.3
SVHN	1.5	2.4	2.6	2.5	2.1

first place in NeurIPS 2018 Adversarial Vision Challenge. Wong *et al.* [52] found that combined with the random initialization, the performance of adversarial training with the FGSM is similar to PGD based training (FGSM-RI), while the cost is significantly lower. They identified the reason for previous failed attempts using FGSM adversarial training, which is catastrophic overfitting.

As shown in Table 7, for all types of adversarial samples, our method has the highest average accuracy compared with other advanced defense methods. Note that the defense is based on the models used in the detection, there is no extra cost of running deep learning models. The detailed performance of different pruning methods is shown in Fig. 7 and Fig. 8. With the defense strategy, pruning models achieve classification accuracy over 95% in the best case (DF attack on CIFAR10). The best case of MMT is 57.2% with the same attack and dataset. Using the best pruning method (SE), the average classification accuracy of pruned models is 68.60% and 72.03% on CIFAR10 and SVHN respectively, while MMT is 22.51% and 24.26%. On SVHN, structured pruning methods achieve significantly higher accuracy than unstructured, while their performance on detection is almost the same. This phenomenon did not appear on the CIFAR10 dataset, which indicates the data complexity has an impact on defense performance. In general, pruned models significantly outperform mutation operators.

In the case of defending against adaptive attacks, our method fails FIA as shown in Table 8. Though our method can not classify FIA adversarial samples correctly, it can still detect them with high accuracy and prevent the model from attacks.

V. CONCLUSION

In this paper, we propose an adversarial sample detection algorithm based on pruned models. Compared with mutation operators, pruning greatly reduces the model size and improves the model sensitivity to the adversarial samples. We use SPRT to test the pruned models outputs and detect adversarial samples through the label changing rate. A simple but effective defense mechanism based on pruned models is proposed. Experimental results show that our method outperforms MMT in AUROC, the number of used models, model size, detection accuracy, and defense success rate. The average AUROC of our method outperforms MMT (90.06% and 90.65%) 7.92% and 7.53% on CIFAR10 and SVHN, respectively. Using only 28.95 and 30.97 models, the average detection accuracy of our method on CIFAR10 and SVHN are 92.80% and 92.43%, while MMT needs 62.65 and 51.78 models with accuracy 64.15% and 73.70%. After adding the defense, the average classification accuracy of adversarial samples reaches 68.60% and 72.03% on CIFAR10 and SVHN respectively, surpassing other advanced adversarial training methods, while MMT only reaches 22.51% and 24.26% under the same mechanism.

ACKNOWLEDGMENT

(Renxuan Wang and Zuohui Chen contributed equally to this work.)

REFERENCES

[1] H. Pham, Z. Dai, Q. Xie, M.-T. Luong, and Q. V. Le, "Meta pseudo labels," 2020, *arXiv:2003.10580*.

[2] T. B. Brown *et al.*, "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[3] J. Pan, J. Shapiro, J. Wohlwend, K. J. Han, T. Lei, and T. Ma, "ASAPP-ASR: Multistream CNN and self-attentive SRU for SOTA speech recognition," 2020, *arXiv:2005.10469*.

[4] X. Fang, Z. Li, and G. Yang, "A novel approach to generating high-resolution adversarial examples," *Appl. Intell.*, vol. 51, pp. 1–17, 2021.

[5] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, May 2018, pp. 303–314.

[6] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[7] F. Yu, Z. Xu, Y. Wang, C. Liu, and X. Chen, "Towards robust training of neural networks by regularizing adversarial gradients," 2018, *arXiv:1805.09370*.

[8] Y. Li, B. Wu, Y. Feng, Y. Fan, Y. Jiang, Z. Li, and S. Xia, "Toward adversarial robustness via semi-supervised robust training," 2020, *arXiv:2003.06974*.

[9] Z. Deng, C. Dwork, J. Wang, and L. Zhang, "Interpreting robust optimization via adversarial influence functions," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2464–2473.

[10] J. Martin and C. Elster, "Detecting unusual input to neural networks," *Int. J. Speech Technol.*, vol. 51, no. 4, pp. 2198–2209, Apr. 2021.

[11] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Defending black-box adversarial attacks on deep neural networks," 2020, *arXiv:2006.14042*.

[12] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," in *Proc. 1st ACM Workshop Secur. Privacy Artif. Intell.*, Oct. 2020, pp. 30–39.

[13] X. Yin, S. Kolouri, and G. K. Rohde, "Adversarial example detection and classification with asymmetrical adversarial training," 2019, *arXiv:1905.11475*.

[14] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "NIC: Detecting adversarial samples with neural network invariant checking," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[15] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang, "DeepMutation: Mutation testing of deep learning systems," in *Proc. IEEE 29th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2018, pp. 100–111.

[16] H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input validation for deep learning applications by crossing-layer dissection," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, Jun. 2020, pp. 727–738.

[17] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, May 2019, pp. 1245–1256.

[18] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100270.

[19] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 3–14.

[20] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artif. Intell. Rev.*, vol. 53, no. 7, pp. 5113–5155, Oct. 2020.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.

[23] Z. Chen *et al.*, "Adversarial sample detection via channel pruning," in *Proc. ICML Workshop Adversarial Mach. Learn.*, 2021, pp. 1–5.

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

[25] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[26] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[28] Q. Huang, I. Katsman, Z. Gu, H. He, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4733–4742.

[29] Z. Wang, H. Guo, Z. Zhang, W. Liu, Z. Qin, and K. Ren, "Feature importance-aware transferable adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 7639–7648.

[30] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1310–1318.

[31] Z. Zhao, G. Chen, J. Wang, Y. Yang, F. Song, and J. Sun, "Attack as defense: Characterizing adversarial examples using robustness," 2021, *arXiv:2103.07633*.

[32] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with improved adversarial robustness," 2018, *arXiv:1810.09619*.

[33] T. Dinh, B. Wang, A. Bertozzi, S. Osher, and J. Xin, "Sparsity meets robustness: Channel pruning for the Feynman-Kac formalism principled robust deep neural nets," in *Proc. Int. Conf. Mach. Learn., Optim., Data Sci.*, 2020, pp. 362–381.

[34] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin, "Adversarial robustness vs. model compression, or both?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 111–120.

[35] S. Kundu, M. Nazemi, P. A. Beerel, and M. Pedram, "DNR: A tunable robust pruning framework through dynamic network rewiring of DNNs," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 344–350.

[36] L. Wang, G. W. Ding, R. Huang, Y. Cao, and Y. C. Lui, "Adversarial robustness of pruned neural networks," in *Proc. ICLR Workshop*, Vancouver, BC, Canada, 2018, pp. 1–5.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[38] S. Wang, X. Wang, S. Ye, P. Zhao, and X. Lin, "Defending DNN adversarial attacks with pruning and logits augmentation," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Nov. 2018, pp. 1144–1148.
[39] A. G. Matachana, K. T. Co, L. Muñoz-González, D. Martinez, and E. C. Lupu, "Robustness and transferability of universal attacks on compressed models," 2020, *arXiv:2012.06024*.
[40] A. Wald, *Sequential Analysis*. Chelmsford, MA, USA: Courier Corporation, 2004.
[41] 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. 1389–1397.
[42] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.
[43] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.
[44] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, "What's hidden in a randomly weighted neural network?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11893–11902.
[45] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-Z. Xu, "Dynamic channel pruning: Feature boosting and suppression," 2018, *arXiv:1810.05331*.
[46] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," 2018, *arXiv:1810.11809*.
[47] M. Ye, C. Gong, L. Nie, D. Zhou, A. Klivans, and Q. Liu, "Good subnetworks provably exist: Pruning via greedy forward selection," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10820–10830.
[48] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6682–6691.
[49] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," 2017, *arXiv:1711.00117*.
[50] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[51] 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.*, 2019, pp. 7472–7482.
[52] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," 2020, *arXiv:2001.03994*.

RENXUAN WANG received the B.S. degree in automation from Anhui Polytechnic University, Wuhu, China, in 2019. He is currently pursuing the master's degree in control theory and engineering with the Zhejiang University of Technology. His current research interests include computer vision and AI security.

ZUOHUI CHEN received the B.S. degree in automation from the Zhejiang University of Technology, Hangzhou, China, in 2019, where he is currently pursuing the Ph.D. degree in control theory and engineering. His current research interests include computer vision, AI application, and AI security.

HUI DONG received the Ph.D. degree in automation from Zhejiang University, Hangzhou, China, in 2007. He is currently a Professor with the College of Information Engineering, Zhejiang University of Technology, Hangzhou. His current research interests include intelligent robot and industrial internet.

QI XUAN (Member, IEEE) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. From 2008 to 2010, he was a Postdoctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010 and 2017. From 2012 to 2014, he was a Postdoctoral Fellow with the Department of Computer Science, University of California at Davis, Davis, CA, USA. He is currently a Professor with the College of Information Engineering, Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou. His current research interests include network science, graph data mining, cyberspace security, machine learning, and computer vision.