Received November 11, 2019, accepted December 5, 2019, date of publication December 9, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958406

A Robust Deep-Neural-Network-Based Compressed Model for Mobile Device Assisted by Edge Server

YUSHUANG YAN[1] **AND QINGQI PEI**[1,2]
[1]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[2]Shaanxi Key Laboratory of Blockchain and Security Computing, Xidian University, Xi'an 710071, China

Corresponding author: Qingqi Pei (qqpei@mail.xidian.edu.cn)

This work was supported in part by the Key Program of NSFC-Tongyong Union Foundation under Grant U1636209, in part by the National Natural Science Foundation of China under Grant 61902292, and in part by the Key Research and Development Programs of Shaanxi under Grant 2019ZDLGY13-07 and Grant 2019ZDLGY13-04.

ABSTRACT Deep neural networks (DNNs) have been extensively used in multiple applications. Due to the over-parameterized DNN-based model, the mobile device has computation and energy limitations to deploy such a model for machine learning tasks. Thus, many works focus on compressing a large-scale model to a small-scale model. In addition, the mobile device training a model assisted by the edge server is an emerging solution in the edge computing environment. However, recent researches have found that DNNs are vulnerable to adversarial examples. These crafted adversarial examples can fool a DNN-based model incorrect predictions. In particular, the DNN-based model can cause risks when used in safety-critical settings. To address this problem, we design a framework for generating a robust deep-convolutional-neural-network-based compressed model in the edge computing environment. The model is partitioned and trained by the mobile device and the edge server. The robust compressed model is constructed mainly via model compression and model robustness. In model robustness, a defensive mechanism is proposed for enhancing the robustness of the compressed model against adversarial examples. Furthermore, the weight distribution of the compressed model is considered for improving the model's accuracy in the defense method. The small-scale compressed model is effective and robust as a collaborative device-server inference for providing recognition tasks for the near devices. On the other hand, it is practical to deploy on the mobile device due to its small-size. Experimental results show that the generated compressed model has strong robustness against adversarial examples while holding high accuracy.

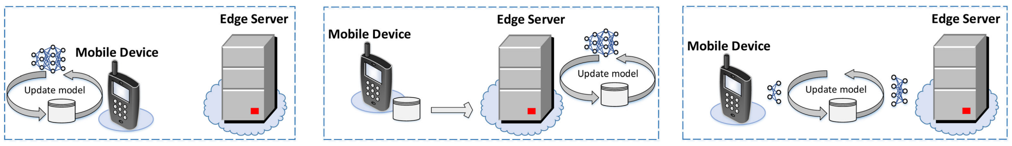
INDEX TERMS Deep neural networks, device-server, compressed model, robustness, edge computing.

I. INTRODUCTION

Deep neural networks (DNNs) have become increasingly popular in multiple application domanis, including image classification [1], [2], object detection [3], [4] and semantic segmentation [5], [6]. In particular, deep convolutional neural networks, such as LeNet, AlexNet, VGGNet, GoogLeNet and ResNet, have achieved outstanding performance on several machine learning tasks.

Unfortunately, DNN-based applications require powerful computation due to the DNN-based model with a large number of parameters. Hence, the DNN-based model cannot well deployed on mobile devices because of their computation and energy limitations. To overcome this problem, researchers have studies how to compress a large-scale model to a small-scale compressed model. Furthermore, many existing works have shown that the compressed model can achieve high utility via deleting the redundancy of the original model. Parameter pruning is extensively used to alleviate computation workload and enhance energy efficiency [7]–[10]. On the other hand, it is natural to consider usage of the emerging edge computing paradigm where all computation tasks are put on the edge server. Generally, the edge server is located in close proximity to the associated mobile device.

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



(a) Device-only method (b) Server-only method (c) Device-server method

FIGURE 1. Device-only, server-only, and device-server methods for training a DNN-based model in the edge computing setting. (a) Device-only method: The mobile device uses its training dataset to construct a model. Due to computation limitation, the device-only method causes latency. (b) Server-only method: The edge server receives the training dataset from the mobile device and constructs a model for the mobile device. This method causes privacy violation since the mobile device needs to send the training dataset to the edge server. (c) Device-server method: The mobile device and the edge server collaboratively construct a model. Compared with the device-only method, this method is effective because parts of the computation tasks are put on the edge server. In addition, the training dataset's privacy can be protected compared with the server-only method since only the intermediate data is transmitted to the edge server.

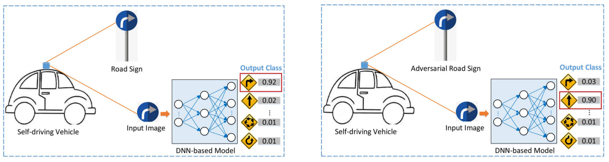
Compared with the traditional cloud-center computing structure [11], [12], the edge server can result in significant low-latency and energy-efficiency [13]–[15].

In the edge computing environment as shown in Fig. 1, the solution to training a DNN-based model on a mobile device with the edge server's assistance (i.e. the device-server method) is superior to the device-only method and the server-only method. Since the edge server has powerful computation, this device-server collaborative training approach can offload parts of computational tasks from the mobile device to the near edge server while yielding low-latency compared with the device-only method [16]–[19]. Meanwhile, training dataset can get some privacy preservation. Compared with the server-only method, the mobile device owns its sensitive training dataset and transfers the intermediate data instead of the raw data to the edge server in the device-server method.

In particular, DNNs have been employed in security-critical settings in recent years [20]–[22]. These applications cause concerns about security of DNNs. Despite their breakthroughs in these complex machine learning tasks, DNNs have been shown to be vulnerable to adversarial examples generated by special methods [23]–[27]. Adversarial examples are maliciously manipulated with small additive perturbations on legitimate samples. Adversarial examples and legitimate samples are indistinguishable to humans. Nevertheless, adversarial examples can fool a DNN-based model as shown in Fig. 2. Therefore, adversarial defense should be considered when a DNN-based model is used in the edge computing setting.

A volume of existing works focus on adversarial defense such as adversarial training [26]–[28], and defensive distillation [29]–[31]. However, adversarial training leads to much computational cost due to additive adversarial examples. Defensive distillation has been shown that the robustness of DNNs is not improved remarkably [32].

To solve the above limitations, we explore a defensive mechanism against this kind of adversarial examples determined by gradients of a model (show in Section V). We focus on enhancing the robustness of a DNN-based model against these adversarial examples. The defensive mechanism



(a) A normal environment (b) An adversarial environment

FIGURE 2. Vulnerability of a DNN-based model shown in the self-driving vehicle application. A self-driving vehicle with a camera can first identify the road sign. Secondly, the input image is fed to a DNN-based model deployed on the self-driving vehicle. Finally, the model outputs the correspond classification of the road sign with the highest probability. (a) shows the vehicle is in a normal environment. In other words, the road sign is a legitimate example. The model will output the road sign as a right-turn sign. In an adversarial environment as (b) shown, the road sign is an adversarial example. However, the adversarial example can fool the model. The model can misclassify the sign for going straight while humans observe it a right-turn sign.

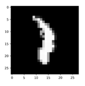


FIGURE 3. Legitimate image. Its label is '1'.

enables high robustness while offering high accuracy of the model (show in Section VI).

We make the following contributions:

- We present a framework for generating a robust DCNN-based compressed model against adversarial examples in the edge computing environment (show in Section IV). Due to the limited computation of the mobile device, the model is partitioned and trained collaboratively by the mobile device and the edge server. The compressed model can hold an attractive trade-off between defensive accuracy and test accuracy.
- To enhance the robustness of the generated compressed model, we propose the defensive mechanism against adversarial attacks. For maintaining high test accuracy of the compressed model, we consider the weight distribution of the compressed model after model compression when adding Laplace noise to the model. Once the robust compressed model is constructed, it can be regarded as a robust collaborative device-server inference against adversarial examples for providing

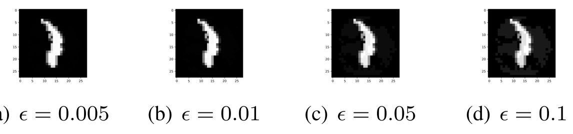


FIGURE 4. Adversarial images are generated by FGSM based the legitimate image shown in Fig. 3. The compressed model1 is constructed without defensive method (describe in Table 1 in Section VI). The model1 can identify the adversarial example's label as '1' when (a) $\epsilon = 0.005$ and (b) $\epsilon = 0.01$. However, it can misclassify the label of the adversarial example as '7' when (c) $\epsilon = 0.05$ and (d) $\epsilon = 0.1$.

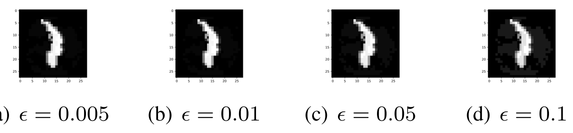


FIGURE 5. Adversarial images are generated by BIM based the legitimate image shown in Fig. 3. The model1 cannot classify the adversarial example correctly when (a) $\epsilon = 0.005$ for 6th step, (b) $\epsilon = 0.01$ for 3rd step, (c) $\epsilon = 0.05$ for first step and (d) $\epsilon = 0.1$ for first step. The label of the adversarial example is identified as '7'.

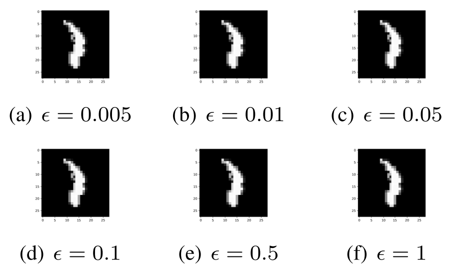


FIGURE 6. Adversarial images are generated by FGSM based the legitimate image shown in Fig. 3. Our compressed robust model is built with defensive method. The robust model is constructed via adding Laplace(+0.5, 1/5.5) and Laplace(−0.5, 1/5.5) to the model1. The robust model can identify the label of the adversarial as '1' with about 1 confidence when $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

recognition services. Furthermore, the generated small-scale compressed model is practical to deployed on the mobile device. It is robust when used in the adversarial setting.

- We extensively evaluate our approach on MNIST dataset under different adversarial attacks including FGSM and BIM. Compared the models with no defense method, the results show that our generated models are more effective against adversarial examples. For example, the compressed model with no defense method can defend the FGSM attack when ϵ is very small ($\epsilon = 0.005, 0.01$ shown in Fig. 4) while suffering from weakness against the BIM attack (show in Fig. 5). However, our generated compressed model shows strong robustness against the FGSM and BIM attacks while having high test accuracy. The results suggest that our approach is high robust against the FGSM attack when a large scale of ϵ is set ($\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ illustrated in Fig. 6). In addition, it can defend the BIM attack successfully when setting $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ for 20th step (describe in Fig. 7).

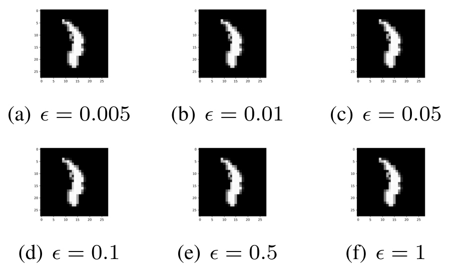


FIGURE 7. Adversarial images are generated by BIM based the legitimate image shown in Fig. 3. Our compressed robust model is built with defensive method. The robust model is constructed via adding Laplace(+0.5, 1/5.5) and Laplace(−0.5, 1/5.5) to the model1. The robust model can identify the label of the adversarial as '1' with about 1 confidence when $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ for 20th step.

The rest of this paper is organized as follows. We introduce the background and related work in Section II. The threat model is briefly presented in Section III. We overview the designed framework in the edge computing environment in Section IV. In Section V, we propose our approach for generating a robust DCNN-based compressed model against adversarial examples. Experimental results are discussed under different adversarial settings in Section VI. Finally, we conclude the work in Section VII.

II. BACKGROUND AND RELATED WORK

In this section, we briefly introduce the method for training a model by the mobile device and the edge server in II-A. Deep neural networks are vulnerable to adversarial examples when used in the edge computing setting in II-B. II-C describes existing adversarial attacks. Existing defense methods are introduced in II-D.

A. MOBILE DEVICE AND EDGE SERVER COLLABORATION

The mobile device assisted by edge server for deep learning has been attracted concerns [18], [19], [33]. Tao and Li proposed a distributed deep learning framework called eSGD [18]. The mobile devices collaboratively train a convolutional neural network model under the help of the edge server. In eSGD, only the important gradient coordinates instead of all gradients are uploaded to cloud for synchronizing so that the communication and computation cost reduce. A privacy-preserving approach for constructing a convolutional neural network model used for face recognition was introduced [19]. Due to the limited computing, the mobile device trains a model with the help of the edge server. The model is partitioned by the mobile device and the edge server. To protect privacy of training dataset, the intermediate results perturbation with differential privacy are sent to the edge server for training the rest of the model. A pruning framework for generating a device-server cooperative inference was presented [33]. Pruning algorithm and DNN partition are used for improving efficiency. The DNN is deployed between the mobile device and the edge server. The input data is fed to the mobile device and the edge server outputs its classification for recognition tasks.

Due to small-size, our generated compressed model is effective. In addition, the training dataset is protected because the mobile device transmits intermediate results instead of the training dataset to the edge server during training. However, latency reduction and confidential training dataset are not our main considerations in this work. We focus on the robustness of our generated model against adversarial attacks in the edge computing environment.

B. ADVERSARIAL DEEP NEURAL NETWORKS IN EDGE COMPUTING

Deep neural networks have achieved superhuman performance on safety and security critical domains, e.g. self-driving vehicles [34], mutations in DNA [35], medical treatments [36] and malware detections [37].

Whereas DNNs perform computer vision tasks successfully, recent researches have shown that DNNs are vulnerable to input examples crafted by adversarial attacks [23]. These examples are called adversarial examples (show in Definition 1). They are perturbed with small modifications, which can fool DNNs [24], [25].

Definition 1 (Adversarial Examples): A DNN-based classifier $f_{\mathbf{w}}(x) : x \in \mathbf{x} \rightarrow y \in Y$ is parameterized by $\mathbf{w}$. $f_{\mathbf{w}}(x)$ takes a legitimate example as input and outputs a corresponding label y. The adversarial attack aims to manufacture x' (i.e. $x' = x + \delta, x' \in \mathbf{x}$) with imperceptible perturbation δ of x. δ is measured by some distance metric. Hence, x' is similar to x so that they are indistinguishable from each other to humans. However, x' enables the classifier a different result as $f_{\mathbf{w}}(x') = y', y' \neq y$.

Edge computing can reduce latency since computing tasks are offloaded from the centralized cloud to the edge server near devices. In addition, the superior performance of DNNs benefits from the capability of providing high accuracy results in learning features from high-dimensional data. DNNs are widely used in many complex applications in the edge computing environment. However, the DNN-based model is not effective to defend adversarial attacks. Hence, we should put much attention on the robustness of the DNN-based model against adversarial examples when the model is employed in edge computing applications.

C. ADVERSARIAL ATTACKS

Adversarial examples can cause security threats in various applications. Many approaches for generating adversarial examples are determined by gradients of the model. In this subsection, we introduce adversarial attacks including Fast Gradient Sign Method (FGSM) and Basic Iterative Method (BIM) against the DNN-based classifier.

1) FAST GRADIENT SIGN METHOD

Goodfellow *et al.* proposed Fast Gradient Sign Method (FGSM) to generate the adversarial example which is very similar to the legitimate example [26]. Given a legitimate example x, the ground-truth label is y. $\mathcal{L}(x, y)$ is a cross-entropy loss function to describe the cost of identifying x as the label y. The sign of the gradient at each pixel decides the update direction. This method only performs one-step gradient update.

The adversarial example x' holds as follows:

$$x' = x + \epsilon \cdot sign(\nabla_x \mathcal{L}(x, y)).$$

ϵ is the volume of the perturbation. The perturbation $\epsilon \cdot sign(\nabla_x \mathcal{L}(x, y))$ can be computed via back propagation. Generally, the perturbation is small.

2) BASIC ITERATIVE METHOD

Kurakin *et al.* presented Basic Iterative Method (BIM) [27]. Compared with FGSM, this method is a finer optimization mechanism for multiple steps. For a small change constraint, the pixel is clipped in each step. $Clip_{x,\epsilon}(x')$ aims to scale the adversarial example x' in ϵ-neighborhood of the legitimate example x.

$$Clip_{x,\epsilon}(x') = min\{255, x + \epsilon, max\{0, x - \epsilon, x'\}\}.$$

The adversarial example is constructed in the jth step:

$$\begin{aligned} x'_0 &= x, \\ x'_{j+1} &= Clip_{x,\epsilon}(x'_j + \alpha \cdot sign(\nabla_x \mathcal{L}(x'_j, y))). \end{aligned}$$

D. DEFENSE MECHANISMS

Various defense mechanisms have been proposed to address the threat of adversarial attacks. We mainly overview adversarial training and defensive distillation in this subsection.

1) ADVERSARIAL TRAINING

Adversarial training is a popular approach to defend against adversarial attacks [26]–[28]. In adversarial training, adversarial examples which are generated by a special adversarial attack method are added to the training dataset for training an improved robust model. On the other hand, the adversarial objective function can also be used to train a robust model. However, online adversarial examples are needed to build. Adversarial training costs more computation than training on legitimate examples only. In addition, the trained model is specific to the adversarial attack. These limitations impede usage of adversarial training.

2) DEFENSIVE DISTILLATION

Existing attack methods depend on optimizing an objective function with respect to the gradient of the model. Naturally, defensive mechanisms focus on how to hide the gradient of the model. Defensive distillation is such a way to train a robust model against adversarial examples so that it is difficult for attackers to generate adversarial examples according to the gradient of the model [29]–[31]. Distillation training techniques are utilized to hide the gradient between the pre-softmax layer and softmax outputs [30]. However, DNNs do not enable remarkable robustness shown in [32].

Adversarial detecting is another approach to detect adversarial examples in the test procedure. The DNN-based model

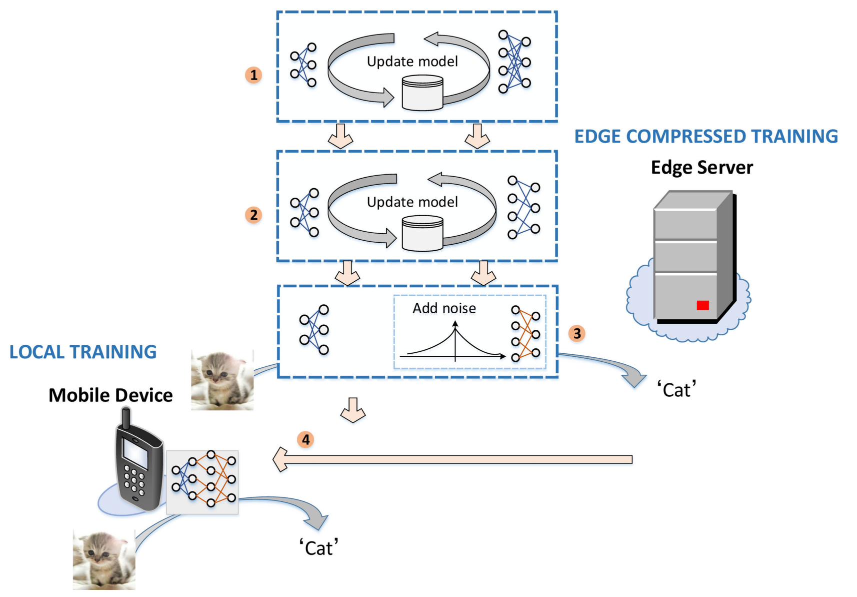


FIGURE 8. **The mobile device and the edge server collaboratively train a robust DCNN-based compressed model in the edge computing environment. The compressed model is partitioning generated via LOCAL TRAINING on the mobile device side and EDGE COMPRESS TRAINING on the edge server side. ① Model initialization: A pre-trained DCNN-based model is built by the mobile device and the edge server. ② Model compression: The compressed model is constructed by deleting the small absolute-value weight connections of the pre-trained model. Model compression is made up of model pruning and model retraining. ③ Model robustness: To guarantee the robustness of the compressed model, the edge server uses the proposed defensive mechanism to the model. ④ Applications: On the one hand, the edge server and the mobile device can offer a collaborative inference for DCNN-based recognition services to the near devices. On the other hand, the edge server sends the noisy part of the model to the mobile device. Finally, the mobile device obtains the robust compressed model. The model is practical to deploy on the mobile device while providing prediction applications. For instance, our generated model can classify the input image as 'cat'.**

constructed is used for detecting to distinguish the legitimate examples or adversarial examples [38]–[41].

To defend adversarial attacks, adversarial training and adversarial detecting do not modify the target model. Defensive distillation aims to mask gradients so that parameters of the target model is changed. In our work, the proposed defensive mechanism perturbs parameters of the target model via adding noise. As a result, gradients of the model are masked.

III. THREAT MODEL

The mobile device and the edge server collaboratively build a DNN-based model in the edge computing setting. We assume that the server has infinite computing power. The communication channel between the mobile device and the edge server is secure.

In this section, we discuss the threat model. The adversary aims to attack the model $f_{\mathbf{w}}(\cdot)$ with adversarial examples. $f_{\mathbf{w}}(\cdot)$ is named the target model. We assume that the adversary has full knowledge of the model, including the model's architecture and parameters.

We also assume that attacks only happen in the test step. They enable only adversarial example generation in the test procedure once the target model is constructed. The adversary has no capacity of modifying the target model such as its parameters, architecture and the training dataset. Specially for the online service setting, it is a general assumption.

IV. FRAMEWORK

In this paper, we concern the robustness of deep convolutional neural networks (DCNNs) in the edge computing setting. We briefly introduce a framework for building a robust DCNN-based compressed model in Fig. 8. The mobile device trains the model with the assistance of the edge server. The DCNN-based model is partitioned between the mobile device and the edge server. The low-level layers which are close to the input layer are deployed on the mobile device. The high-level layers which are close to the output layer are deployed on the edge server. In forward propagation, the mobile device first feeds its training dataset to the lower-level layers and transmits the intermediate results to the edge server via the communication channel. Secondly, the edge server continues to finish loss computing with the high-level layers. In back propagation, the edge server employs the loss result to achieve model's parameter update with the mobile device.

The model generation includes LOCAL TRAINING and EDGE COMPRESSED TRAINING. Training the model on the mobile device side is called LOCAL TRAINING and training the model on the edge server is called EDGE COMPRESSED TRAINING. We construct the robust compressed model by three procedures including model initialization (show in V-A), model compression (show in V-B) and model robustness (show in V-C). In model initialization, the mobile device and the edge server generate a large-scale pre-trained DCNN-based model with proper performance. The purpose is to learn and get the distribution of the important connections with the large absolute-value weight. Secondly, model compression is achieved. Model compression is composed of model pruning and model retraining. The edge server deals with model pruning. The small absolute-value weight connections of the model deployed on the edge server are deleted. Model pruning is only completed on the edge server side. Model retraining is executed by the mobile device and the edge server. The small-scale compressed model is then constructed.

However, the model is vulnerable to adversarial examples. To address the problem, the edge server uses the proposed defensive mechanism for guaranteeing the robustness of the compressed model in model robustness. We directly add Laplace noise to the compressed model after model compression. In particular, we care about the weight distribution of the compressed model for enhancing the model's test accuracy when adding noise. Finally, the robust compressed model with high test accuracy is built. The model is significantly useful for recognition applications. On the one hand, the partitioned compressed model can be regarded as a collaborative device-server inference for recognition tasks. Due to the help of the edge server, latency reduces. Effective recognition services are provided for the near mobile devices. On the other hand, the mobile device receives the noisy part of the model from the edge server to get a small-scale robust compressed model. The model is easy to deploy on the mobile device due to small storage and computation. The DCNN-based model is known to play an increasingly important role in security domains.

V. APPROACH

Deep neural networks, also regarded as deep learning (DL), is a subfield of artificial intelligence (AI). DNNs have the ability to respond to requests like humans do. As a result, DNNs have brought significant performance improvements to many machine learning tasks.

A DNN is composed of the input layer, the hidden layer and the output layer: The neurons in the input layer take the raw data as input; The neurons in the hidden layer compute the weighted sum to feed into the output layer; The neurons in the output layer output the final results of the network. The process of training a DNN-based model can be known as an optimization problem. We define a loss function $\mathcal{L}(\mathbf{w}, \mathbf{x}) = \frac{1}{n}\sum_i \mathcal{L}(\mathbf{w}, x_i)$ which measures the average of the loss made by the model $f_{\mathbf{w}}$ on the training dataset $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. The goal is to find $\mathbf{w}$ corresponding to the expected small loss. One way to approximately solve the above optimization problem is mini-batch stochastic gradient descent (SGD) method [42]. At t iteration of mini-batch SGD, we choose a random batch $\mathbf{B}$ composed of b training examples and update our model at $t+1$ iteration according to $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta_t}{b}\sum_{x_i\in\mathbf{B}} \mathcal{L}(\mathbf{w}_t, x_i)$, where η_t is the learning rate at t iteration, $\mathbf{w}_t$ is the set of the model's parameters at t iteration and $\mathbf{w}_{t+1}$ is the set of the model's parameters at $t+1$ iteration.

A. MODEL INITIALIZATION

Due to the huge ability of computing, the edge server assists the mobile device to train the model collaboratively. Before model compression, the mobile device and the edge server first construct a pre-trained DCNN-based model f_1 by a normal network training. To learn the distribution of important network connections, the pre-trained model should have a proper accuracy. Note that the important connections refer to the connections with the large absolute-value weight. The pruned weight threshold t_d is set for connection pruning. These less important connections should be deleted if the absolute-value of weights is lower than t_d.

B. MODEL COMPRESSION

After model initialization, the model can be compressed. The convolutional layers are known to learn the important features of training dataset. Pruning connections of the convolutional layer will lead to accuracy reduction of the model. Furthermore, compared with the convolutional layer's connections, the fully connected layer's connections are the most part of the model. Thus, we can delete connections of some convolutional layers or only prune that of the fully connected layers.

Our model compressive method is inspired by the pruning approach [9]. Model compression is composed of model pruning and model retraining. Model pruning is accomplished by the edge server. The mobile device and the edge server execute model retraining collaboratively. In model pruning, the less important connections are pruned. To improve the accuracy of the compressed model, model pruning and model retraining should be taken several iterations.

Some methods such as dropout and regularization are widely applied to keep off over-fitting during model training. In dropout, we drop weights probabilistically when retraining but will recover when inference. For preventing over-fitting, regularization is another method used to penalize weights of the model.

In our approach, we use the L_2 norm of weights for regularization.

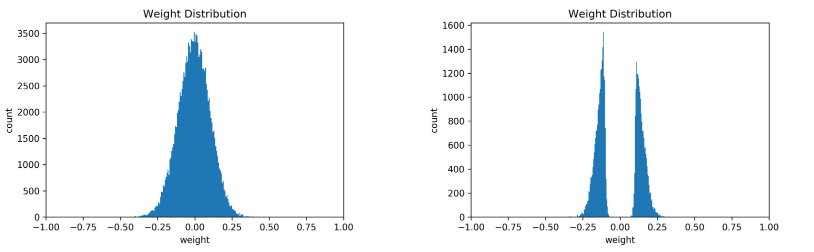
$$\mathcal{J}(\mathbf{w}, \mathbf{x}) = \mathcal{L}(\mathbf{w}, \mathbf{x}) + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$

$\mathcal{L}(\mathbf{w}, \mathbf{x})$ is the loss function of the model and λ is a regularized parameter.

In SGD, at t step, a batch $\mathbf{B}$ randomly made up of b examples $(x_1, x_2, \ldots, x_b)$ are collected to compute the average gradient as following:

$$g_t = \frac{1}{b}\sum\nolimits_{x_i\in\mathbf{B}} \nabla_{\mathbf{w}_t}\mathcal{J}(\mathbf{w}_t, x_i).$$

g_t realizes weight update (show in Algorithm 1).



(a) Pre-trained model (b) Compressed model after model retraining

FIGURE 9. Weight distributions of fully connected layers.

Algorithm 1 Model Compression

```
Require:
    Initial a model denoted as $f_0$;
    Training dataset $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$;
    A batch $\mathbf{B}$ made up of $b$ random training examples;
    Pruned weight threshold $t_d$;
    Size of maximum iteration $T$;
    Learning rate $\eta_t$
 1: Train $f_0$ and get a pre-trained model $f_1$ with appropriate accuracy
 2: Model pruning on the edge server side:
 3: Delete these connections whose absolute-value of weights is lower than $t_d$
 4: Model retraing on the mobile device side and the edge server side:
 5: for $t \in [T]$ do
 6:     $b$ random examples taken by the mobile device
 7:     for $x_i \in \mathbf{B}$ do
 8:         Weight update by the mobile device and the edge server
 9:         $g_t \leftarrow \frac{1}{b} \sum_{x_i \in \mathbf{B}} \nabla_{\mathbf{w}_t} \mathcal{J}(\mathbf{w}_t, x_i)$
10:         $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t g_t$
11:     end for
12: end for
Ensure:
    Compressed model $f$
```

C. MODEL ROBUSTNESS

The compressed DCNN-based model f (describe in Algorithm 1) is obtained after model compression. However, the model without any defensive method is vulnerable to adversarial attacks. An adversary can use adversarial examples to fool a model. Adversarial examples which are built based on gradients of a model are used widely. Therefore, our defensive approach aims to improve the robustness of the compressed model against these adversarial examples. Furthermore, the accuracy of the model should be concerned. Adding noise to the compressed model f is an appropriate approach to sanitize the model. The further goal is to mask gradients of the model. Hence, the vulnerability of the model under the adversarial attack can be addressed.

Intuitively, we inject one single Laplace noise to the compressed model f for enhancing robustness. As described in Section V-B, the convolutional layers are used to extract the important features of training dataset. The additive Laplace noise to the convolutional layers causes low accuracy of the model. Hence, we only add Laplace noise to the fully connected layers. The Laplace noise distribution has a location parameter 0 and a scale parameter $\frac{1}{b}$. We define $f_{FC} = \{w_1, w_2, \ldots, w_N\}$ as the weight set of the whole fully connected layers. Laplace noise is directly added to f_{FC}.

$$w_1^p = w_1 + Laplace(0, \frac{1}{b})$$
$$w_2^p = w_2 + Laplace(0, \frac{1}{b})$$
$$\vdots$$
$$w_N^p = w_N + Laplace(0, \frac{1}{b})$$

The set $\{w_1^p, w_2^p, \ldots, w_N^p\}$ shows the perturbed weights of the fully connected layers. $\mathcal{M}(f_{FC})$ is a random mechanism to confuse the compressed model via the additive Laplace noise for enhancing the robustness of the model.

$$\begin{aligned}\mathcal{M}(f_{FC}) &= (w_1, w_2, \ldots, w_N) + Laplace^N(0, \frac{1}{b}) \\ &= (w_1^p, w_2^p, \ldots, w_N^p) \end{aligned} \tag{1}$$

We conduct some experiments to observe the robustness of the model with this naive defense method against adversarial example. Unfortunately, the results show that the model reveals weakness against adversarial examples. Because of test accuracy guarantee, the additive Laplace noise needs very small. Due to the small noise, the model has low robust performance. Hence, we need to consider our defensive mechanism for getting a good trade-off accuracy and robustness of the model.

Fig. 9 shows that the weight distribution of the pre-trained model's fully connected layers is a unimodal distribution centered on zero. Due to pruning a large number of low-weight connections close to zero, we observe that the weight distribution becomes an approximate symmetrical bimodal distribution from a unimodal distribution after model retraining (indicate in Fig. 9(b)). To maintain high robustness, we propose a defensive mechanism against adversarial examples by injecting two symmetrical Laplace noise distributions. One laplace noise distribution is denoted as *Laplace* $(-E, \frac{1}{n\lambda\mathcal{P}_m})$ which has a location parameter $-E$ and a scale parameter $\frac{1}{n\lambda\mathcal{P}_m}$ and the other is *Laplace* $(+E, \frac{1}{n\lambda\mathcal{P}_m})$ with a location parameter $+E$ and a scale parameter $\frac{1}{n\lambda\mathcal{P}_m}$. The volume of Laplace noise can be determined by three parameters including n, λ and $\mathcal{P}_m$. n is the size of an input training dataset $\mathbf{x}$. λ is a regularized parameter. $\mathcal{P}_m$ is a model robustness parameter.

We introduce the robust metric $\mathcal{P}_m$ in this subsection. $\mathcal{P}_m$ shows the ability of preventing adversarial examples from fooling a model. Smaller $\mathcal{P}_m$ leads to larger additive Laplace noise. That is, it is difficult for adversarial examples to fool the model since gradients are masked greatly. However, noise has a passive effect on predicted outcomes. Thus, small $\mathcal{P}_m$ enables strong robustness of a model while yielding low accuracy for predicting recognition tasks. We should focus on the trade-off between accuracy and robustness of the model. $\mathcal{P}_m$ should be selected carefully.

Algorithm 2 Defensive Mechanism

```
Require:
    A compressed model $f$
1: Noise Added
2: $\mathcal{M}_1(f_{FC1}) \leftarrow f_{FC1} + Laplace^{n_1}(-E, \frac{1}{n\lambda\mathcal{P}_m})$
3: $\mathcal{M}_2(f_{FC2}) \leftarrow f_{FC2} + Laplace^{n_2}(+E, \frac{1}{n\lambda\mathcal{P}_m})$
4: $\mathcal{M}(f_{FC}) \leftarrow \{\mathcal{M}_1(f_{FC1}), \mathcal{M}_2(f_{FC2})\}$
5: $f_r \leftarrow \mathcal{M}(f_{FC})$
Ensure:
    Robust compressed model $f_r$
```

Algorithm 2 describes our defensive mechanism. f_{FC} is made up of f_{FC1} and f_{FC2}.

$$f_{FC1} = \sum_{i=1}^{n_1}\{w_i, |w_i < 0\}$$

$$f_{FC2} = \sum_{i=n_1+1}^{n_1+n_2}\{w_i, |w_i \geq 0\}$$

$$N = n_1 + n_2$$

A random mechanism $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2\}$ takes data $f_{FC} = \{f_{FC1}, f_{FC2}\}$ as input and outputs the robust model f_r. The model f is disturbed by $\mathcal{M}(f_{FC})$ ($\mathcal{M}(f_{FC}) = \{\mathcal{M}_1(f_{FC1}), \mathcal{M}_2(f_{FC2})\}$) if $\mathcal{M}_1$ and $\mathcal{M}_2$ give perturbation with two symmetrical Laplace noise when querying over datasets f_{FC1} and f_{FC2}, respectively.

If the following equations

$$\begin{aligned}\mathcal{M}_1(f_{FC1}) &= (w_1, w_2, \ldots, w_{n_1}) + Laplace^{n_1}(-E, \frac{1}{n\lambda\mathcal{P}_m}) \\ &= (w_1^p, w_2^p, \ldots, w_{n_1}^p) \\ \mathcal{M}_2(f_{FC2}) &= (w_{n_1+1}, w_{n_1+2}, \ldots, w_{n_1+n_2}) \\ &\quad + Laplace^{n_2}(+E, \frac{1}{n\lambda\mathcal{P}_m}) \\ &= (w_{n_1+1}^p, w_{n_1+2}^p, \ldots, w_{n_1+n_2}^p)\end{aligned}$$

TABLE 1. Train and test accuracy of various models with no defense method.

	Pre-trained model	Compressed models			
		Model1	Model2	Model3	Model4
Train	98.4%	95.8%	97.7%	96.1%	97.7%
Test	97.6%	96.0%	97.2%	96.0%	97.3%

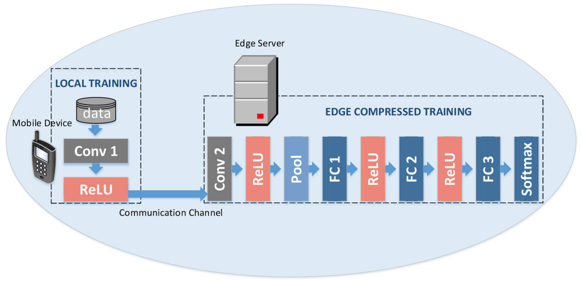


FIGURE 10. Mobile device and edge server collaboratively training a robust DCNN-based compressed model. The mobile device owns its training dataset. Conv1 layer followed by ReLU function is deployed on the mobile device. The other parts of the model are deployed on the edge server.

hold, then the equation

$$\begin{aligned}\mathcal{M}(f_{FC}) &= \{\mathcal{M}_1(f_{FC1}), \mathcal{M}_2(f_{FC2})\} \\ &= (w_1^p, w_2^p, \ldots, w_N^p)\end{aligned}$$

holds.

VI. IMPLEMENTATION AND EVALUATION

A. SETUP

We evaluate our approach on the standard dataset MNIST [43]. The MNIST dataset consists of 55000 training examples, 5000 validation examples and 10000 testing examples of handwritten digits formatted as 28 × 28 gray images in the range [0-9]. They can be classified into 10 classes.

We first conduct a deep convolutional neural network as the pre-trained model for MNIST. The train and test accuracy are 98.4% and 97.6% respectively (describe in Table 1). As shown in Fig. 10, the DCNN-based model is composed of two convolutional (i.e. Conv1 and Conv2) layers and three fully connected (i.e. FC1, FC2, and FC3) layers. FC1 layer and FC2 layer have 500 outputs respectively. FC3 has 10 outputs. L_2-norm regularization is used in the connected fully layers. The convolutional (Conv) layers and pooling (Pool) layers enable feature extraction when an image is fed into a DCNN-based model. The rectified linear unit (ReLU) as the activation function is used for fast training. The softmax layer is the output layer with 10 outputs. The model as our baseline model has more than 60000 parameters including weight and bias. The model is partitioned by the mobile device and the edge server. One convolutional (Conv1) layer is deployed on the mobile device side while one convolutional (Conv2) layer and three fully connected (FC1, FC2 and FC3) layers are deployed on the edge server side.

TABLE 2. Test accuracy of robust models with defense method.

	Models based model1									
	$E = \pm 0.5$					$E = \pm 0.6$				
Scale	1/5.5	1/11	1/22	1/27.5	1/55	1/5.5	1/11	1/22	1/27.5	1/55
Test	92.3%	92.8%	93.5%	93.3%	93.4%	92.5%	92.7%	93.3%	93.3%	93.4%
	Models based model2									
	$E = \pm 0.5$					$E = \pm 0.6$				
Scale	1/5.5		1/55			1/5.5		1/55		
Test	96.4%		96.2			95.8%		96.2%		
	Models based model3									
	$E = \pm 0.5$					$E = \pm 0.6$				
Scale	1/5.5		1/55			1/5.5		1/55		
Test	93.1%		93.4			92.5%		93.3%		
	Models based model4									
	$E = \pm 0.5$					$E = \pm 0.6$				
Scale	1/5.5		1/55			1/5.5		1/55		
Test	96.0%		96.2			95.7%		96.2%		

B. METRICS

Distance: Adversarial examples and legitimate examples are indistinguishable to humans. Thus, we should use a distance metric to evaluate difference between an adversarial example and a legitimate example [32]. We use $\mathbf{L}_2$ (i.e. Euclidean distance) in our experiments. $\mathbf{L}_2$ holds as follows:

$$\|\mathbf{x}\|_2 = (\Sigma_{i=0}^{n}|\mathbf{x}_i|^2)^{\frac{1}{2}}.$$

To evaluate the robustness of the generated compressed model, we investigate two adversarial attacks such as FGSM and BIM. In FGSM, we choose perturbation $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ to evaluate the robustness of our approach. In BIM, we set $\alpha = 1$ and $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ for 20 steps to observe the robust performance of our approach.

We use test accuracy and defensive accuracy to evaluate the performance of the generated compressed model in the experiments. Test accuracy shows the ability for prediction task. Defensive accuracy suggests the robustness of the model against adversarial examples. The higher test accuracy and defensive accuracy are, the higher performance of the model has.

Test accuracy: Given a DCNN-based classifier, test accuracy is the probability of the testing legitimate examples which the classifier can classify correctly. In our experiments, we have 10000 MNIST testing examples.

Defensive accuracy: Given a DCNN-based classifier, defensive accuracy is the probability of adversarial examples which the classifier can identify the label of the adversarial example as the label of the legitimate example used to generate the adversarial example. In our experiments, the adversarial examples based 10000 MNIST testing examples are constructed by FGSM and BIM.

C. RESULTS

In model compression as shown in V-B, we build a set of compressed models such as model1, model2, model3 and model4 described in Table 1 based the above pre-trained model. We can see that they have high accuracy for train

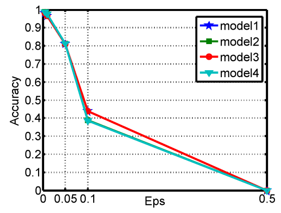


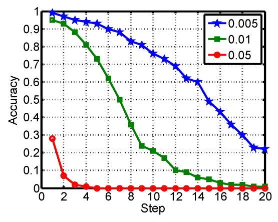
FIGURE 11. Defensive accuracy of model1, model2, model3 and model4 against FGSM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5$.

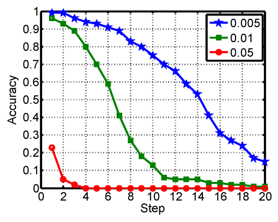
and test. The four models are constructed with different pruned weight threshold t_d and regularized parameter λ. We define a metric called compressed rate R. It holds $R = \frac{m_{delete}}{m_{whole}}$. m_{delete} indicates the number of the deleted connections of the model while m_{whole} shows the number of the whole connections of the model. R is 0.9 when t_d is 0.03. R is 0.8 when t_d is 0.02. Larger R shows more deleted connections so that the size of the model is smaller and the accuracy is lower.

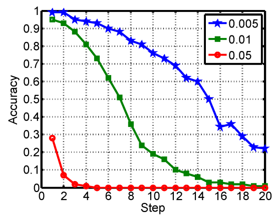
- Model1: $t_d = 0.03, \lambda = 10^{-4}$.
- Model2: $t_d = 0.02, \lambda = 10^{-4}$.
- Model3: $t_d = 0.03, \lambda = 10^{-5}$.
- Model4: $t_d = 0.02, \lambda = 10^{-5}$.

In model robustness as shown in V-C, we add different Laplace noise to the above compressed models (i.e. model1, model2, model3 and model4) for constructing robust compressed models in Table 2. They have high test accuracy. In particular, the generated robust models based model2 and model4 can get over 96.0% test accuracy as shown in Table 2.

We first measure the robustness of model1, model2, model3 and model4 which have no defensive method. As shown in Fig. 11, these compressed models are effective against the adversarial examples generated by FGSM when ϵ is very small (i.e. $\epsilon = 0.005, 0.01$) while they are very vulnerable when $\epsilon = 0.5$. We also examine the robustness of these models under the BIM attack. From Fig. 12, we can see that the defense ability of the compressed models reduces remarkably as step increases though ϵ (i.e. $\epsilon = 0.005$) is


(a) Model1


(b) Model2


(c) Model3

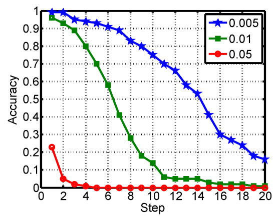

(d) Model4

FIGURE 12. Defensive accuracy of model1, model2, model3 and model4 against BIM for $\epsilon = 0.005, 0.01, 0.05$ within 20 steps.

very small. In other words, these models cannot defend the BIM attack.

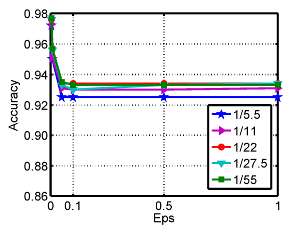
Secondly, Fig. 13 - Fig. 20 suggest the results of our robust compressed models in Table 2 under FGSM and BIM attacks. These models outperform dramatically model1, model2, model3 and model4 with no defense method. Our generated models are robust against FGSM as well as BIM for a large scale ϵ. ϵ can be chosen as 0.005, 0.01, 0.05, 0.1, 0.5, 1.

Furthermore, we investigate different parameters including t_d, λ, E and $\mathcal{P}_m$ on the robust models against FGSM and BIM. As previously discussed, n is the amount of training dataset, which also influences the model's robustness. Since the model's accuracy significantly depends on the size of training dataset, we consistently use $n = 55000$ for guaranteeing the accuracy of the model.

Parameters such as t_d and λ have a direct effect on the train and test accuracy of the compressed model during model compression. Additive Laplace noise to the model is determined by parameters including λ, E and $\mathcal{P}_m$. These three parameters affects the trade-off between test accuracy and robustness of the model in model robustness. We can see the results shown in Fig. 13 - Fig. 20. The robust models have the trade-off between high accuracy and strong robustness for a large scale of ϵ.

1) IMPACT OF PRUNED WEIGHT THRESHOLD

On the one hand, the pruned weight threshold t_d determines the size of the compressed model. Smaller t_d yields smaller R. Meanwhile, smaller R means the larger size of the model. On the other hand, t_d has an important influence on the train and test accuracy of the compressed model. As Table 1 shown, model2 and model4 have higher accuracy than model3 and model4. Smaller t_d leads to smaller R. Hence, the model achieves higher accuracy.


(a) Laplace noise with same $E = \pm 0.5$ and different $scale = 1/5.5, 1/11, 1/22, 1/27.5, 1/55$

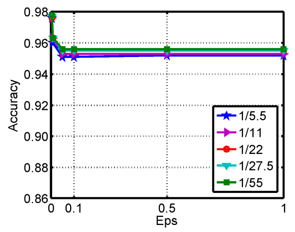
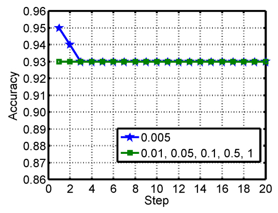
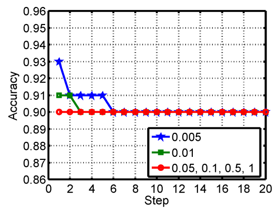
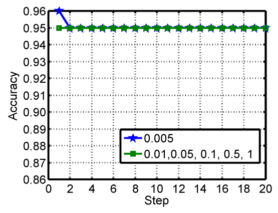

(b) Laplace noise with same $E = \pm 0.6$ and different $scale = 1/5.5, 1/11, 1/22, 1/27.5, 1/55$

FIGURE 13. Defensive accuracy of robust models based model1 against FGSM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.


(a) Laplace noise with $E = \pm 0.5$ and $scale = 1/5.5$


(b) Laplace noise with $E = \pm 0.5$ and $scale = 1/55$


(c) Laplace noise with $E = \pm 0.6$ and $scale = 1/5.5$

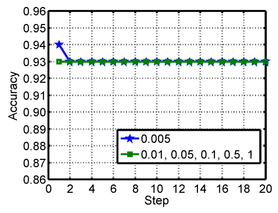
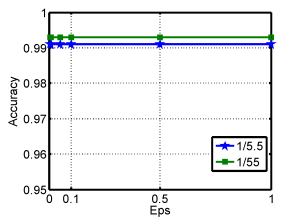

(d) Laplace noise with $E = \pm 0.6$ and $scale = 1/55$

FIGURE 14. Defensive accuracy of robust models based model1 against BIM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ within 20 steps.


(a) Laplace noise with same $E = \pm 0.5$ and different $scale = 1/5.5, 1/55$

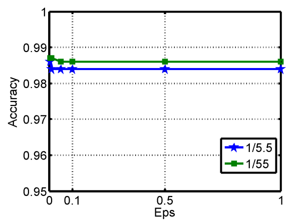
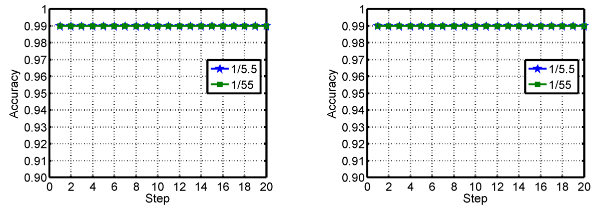

(b) Laplace noise with same $E = \pm 0.6$ and different $scale = 1/5.5, 1/55$

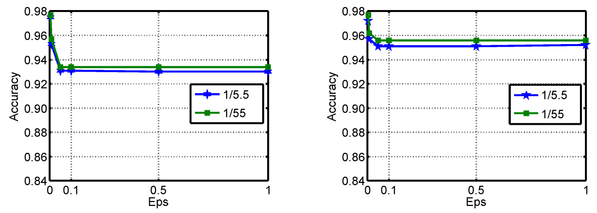
FIGURE 15. Defensive accuracy of robust models based model2 against FGSM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

Furthermore, the robust models generated based model2 and model4 can reach higher test accuracy than the ones generated based model1 and model3 when the same noise is added (illustrate in Table 2). Thus, the robust model with small R holds stronger robustness. For example, the defensive accuracy as illustrated in Fig. 15(b) is over 98.0% for various ϵ against the FGSM attack when the Laplace(± 0.6, $1/5.5$) is injected to the model2. However, the model has about 95.0% defensive accuracy in Fig. 13(b) when the same noise added



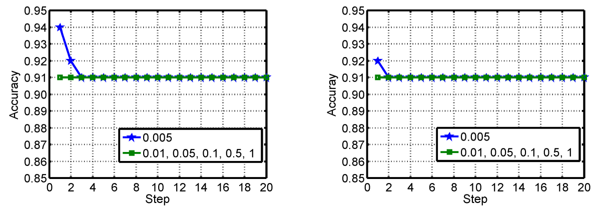
(a) Laplace noise with same $E = \pm\ 0.5$ and different $scale = 1/5.5, 1/55$ (b) Laplace noise with same $E = \pm\ 0.6$ and different $scale = 1/5.5, 1/55$

FIGURE 16. Defensive accuracy of robust models based model2 against BIM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ within 20 steps.

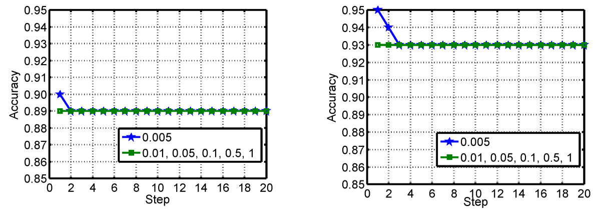


(a) Laplace noise with same $E = \pm\ 0.5$ and different $scale = 1/5.5, 1/55$ (b) Laplace noise with same $E = \pm\ 0.6$ and different $scale = 1/5.5, 1/55$

FIGURE 17. Defensive accuracy of robust models based model3 against FGSM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.



(a) Laplace noise with $E = \pm\ 0.5$, $scale = 1/5.5$ (b) Laplace noise with $E = \pm\ 0.5$, $scale = 1/55$



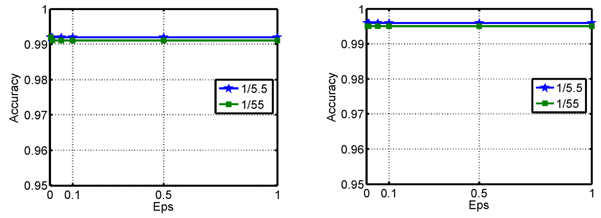
(c) Laplace noise with $E = \pm\ 0.6$, $scale = 1/5.5$ (d) Laplace noise with $E = \pm\ 0.6$, $scale = 1/55$

FIGURE 18. Defensive accuracy of robust models based model3 against BIM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ within 20 steps.

to model1. The model based model2 in Fig. 16(a) gets about 99.0% defensive accuracy for different ϵ under the BIM attack while the model based model1 in Fig. 14(a) holds about 93.0% defensive accuracy when adding the same Laplace(± 0.5, $1/5.5$).

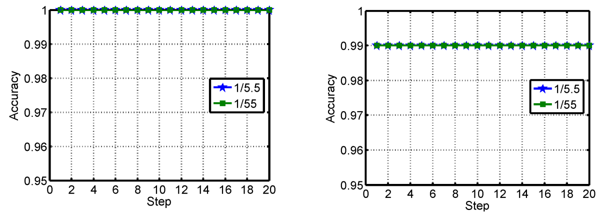
2) IMPACT OF LOCATION PARAMETER

We examine the impact of different location parameter $E = \pm 0.5, \pm 0.6$ on the effectiveness of the models against adversarial examples generated by FGSM and BIM while holding



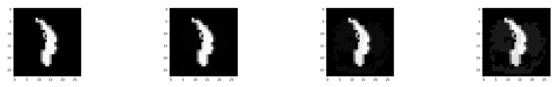
(a) Laplace noise with same $E = \pm\ 0.5$ and different $scale = 1/5.5, 1/55$ (b) Laplace noise with same $E = \pm\ 0.6$ and different $scale = 1/5.5, 1/55$

FIGURE 19. Defensive accuracy of robust models based model4 against FGSM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.



(a) Laplace noise with same $E = \pm\ 0.5$ and different $scale = 1/5.5, 1/55$ (b) Laplace noise with same $E = \pm\ 0.6$ and different $scale = 1/5.5, 1/55$

FIGURE 20. Defensive accuracy of robust models based model4 against BIM for $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$ within 20 steps.



(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$

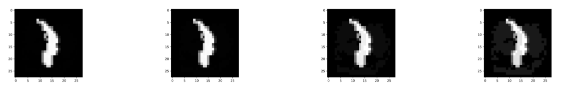
FIGURE 21. Our model2 against FGSM. Our model can identify the adversarial image as the right label '1' with confidence 0.843, 0.816, 0.491 when setting $\epsilon = 0.005, 0.01, 0.05$. However, the model misclassifies the adversarial image as the label '7' with confidence 0.663 when setting $\epsilon = 0.1$.

high test accuracy. Fig. 13, Fig. 15, Fig. 17 and Fig. 19 report that the models provide strong robustness against FGSM when E is ± 0.5 and ± 0.6 respectively. The models can achieve over 92.0% defensive accuracy when different ϵ are set. Specially, the defensive accuracy can reach more than 99.0% in Fig. 19. Fig. 14, Fig. 16, Fig. 18 and Fig. 20 suggest that our robust models can maintain over 90% defensive accuracy within 20 steps under the BIM attack when $E = \pm 0.5, \pm 0.6$. The model can hold as high as about 99.0% defensive accuracy in Fig. 20.

3) IMPACT OF REGULARIZED PARAMETER

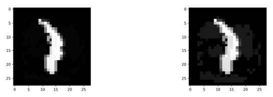
Regularized parameter λ affects the train and test accuracy of the models (i.e. model1, model2, model3 and model4) in model compression as well as the test accuracy and robustness of the models (show in Table 2) in model robustness. From Table 1, we can see that model1, model2, model3 and model4 can hold about 96.0% accuracy for train and test respectively when $\lambda = 10^{-4}$ or $\lambda = 10^{-5}$.

Fig. 13(b) and Fig. 17(b) show high robustness against the FGSM attack when $\lambda = 10^{-4}$ and $\lambda = 10^{-5}$ respectively.



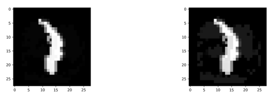
(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$

FIGURE 22. Han's compressed model against FGSM. Han's model can identify the adversarial image as the right label '1' with confidence 0.838, 0.811, 0.488 when setting $\epsilon = 0.005, 0.01, 0.05$. However, the model misclassifies the adversarial image as the label '7' with confidence 0.628 when setting $\epsilon = 0.1$.



(a) $\epsilon = 0.001$ (b) $\epsilon = 0.005$

FIGURE 23. Our model2 against BIM for 20th step. Our model can identify the adversarial image as the right label '1' with confidence 0.708 when setting $\epsilon = 0.001$. However, the model misclassifies the adversarial image as the label '7' with confidence 0.943 when setting $\epsilon = 0.005$.



(a) $\epsilon = 0.001$ (b) $\epsilon = 0.005$

FIGURE 24. Han's compressed model against BIM for 20th step. Han's model can identify the adversarial image as the right label '1' with confidence 0.703 when setting $\epsilon = 0.001$. However, the model misclassifies the adversarial image as the label '7' with confidence 0.939 when setting $\epsilon = 0.005$.

Defensive accuracy can achieve about 95.0% for $\lambda = 10^{-4}$ and $\lambda = 10^{-5}$ respectively. Fig. 14 and Fig. 18 indicate that the models are robust under the BIM attack when $\lambda = 10^{-4}$ or $\lambda = 10^{-5}$. For the same $\mathcal{P}_m = 10$, Fig. 18(a) illustrates that defensive accuracy reaches about 91.0% when $\lambda = 10^{-5}$ while defensive accuracy is about 90.0% in Fig. 14(b) when $\lambda = 10^{-4}$. Compared with Fig. 14(b), Fig. 18(a) has higher defensive accuracy because smaller λ causes larger magnitude of noise so that more noise are injected to the model. More noise means larger perturbation for gradients. Thus, the model in Fig. 18(a) performs better against BIM than the model in Fig. 14(b).

4) IMPACT OF ROBUSTNESS PARAMETER

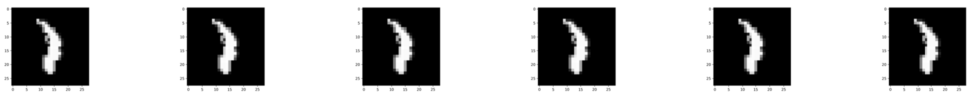
Robustness parameter $\mathcal{P}_m$ has a significant influence on the test accuracy and robustness of the model against adversarial examples. Compared Fig. 14(a) with Fig. 14(b), smaller $\mathcal{P}_m$ leads to stronger robustness against the BIM attack. Smaller $\mathcal{P}_m$ yields larger Laplace noise. Large noise causes big changes of gradients. As we known, the adversarial examples constructed by BIM are based gradients. Larger noise turns out bigger gradient perturbation. As a result, the model has stronger robustness because gradients are better masked. However, the test accuracy goes down when the size of noise increases. Therefore, the noise should select carefully for guaranteeing a good trade-off between test accuracy and robustness of the model. Compared Fig. 14(c) with Fig. 14(d), we can get the same conclusion. In particular, the defensive accuracy can achieve about 99% against the BIM attack as Fig. 16 and Fig. 20 shown. The compressed models i.e. model2 and model4 have high test accuracy because the compressed rate is relatively low. Hence, the robust models generated based model2 and model4 can get high test accuracy. It returns a positive impact on the defensive accuracy of the models. Therefore, these models can support high robustness against BIM.

However, we can see that the robustness of the model against the FGSM attack increases slightly when $\mathcal{P}_m$ increases in Fig. 14(b). Large $\mathcal{P}_m$ yields small Laplace noise. Intuitively, small noise makes small perturbation for gradients of the model so that the model should have low robustness. On the contrary, Fig. 14(a) shows the different result. Small $\mathcal{P}_m$ yields large Laplace noise so that the test accuracy of the model reduces. We can see that the defensive accuracy is affected by the test accuracy of the model. The defensive accuracy reduces as the test accuracy reduces. As a result, the robust performance of the model reduces against adversarial examples generated by FGSM.

5) OUR DEFENSE METHOD AGAINST THE ADVERSARIAL IMAGE

Firstly, we construct a compressed model with Han's pruning method [9] when setting $t_d = 0.02$ and $\lambda = 10^{-4}$. Han's compressed model can get 97.9% for train accuracy and 97.2% for test accuracy. Han's robust compressed model with Laplace(± 0.5, $1/5.5$) has 96.3% for test accuracy. Compared with our model2 (show in Table 1) and the corresponding robust model (show in Table 2), they have similar test accuracy.

From Fig. 21, Fig. 22, Fig. 23, Fig. 24, we investigate the robustness of our model2 and Han's compressed model against the adversarial image generated by FGSM and BIM. As Fig. 21 and Fig. 22 shown, our and Han's compressed model with no defensive mechanism can defend the FGSM attack when ϵ is very small (i.e. $\epsilon = 0.005, 0.01, 0.05$). The adversarial image can fool the models when ϵ is 0.1.



(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$ (e) $\epsilon = 0.5$ (f) $\epsilon = 1$

FIGURE 25. Our robust model against FGSM. Our robust model can identify the adversarial image as label '1' with confidence about 1 when setting $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$ (e) $\epsilon = 0.5$ (f) $\epsilon = 1$

FIGURE 26. Han's robust model against FGSM. Han's robust model can identify the adversarial image as label '1' with confidence about 1 when setting $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$ (e) $\epsilon = 0.5$ (f) $\epsilon = 1$

FIGURE 27. Our robust model against BIM for 20th step. Our robust model can identify the adversarial image as label '1' with confidence about 1 when setting $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

(a) $\epsilon = 0.005$ (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.1$ (e) $\epsilon = 0.5$ (f) $\epsilon = 1$

FIGURE 28. Han's robust model against BIM for 20th step. Han's robust model can identify the adversarial image as label '1' with confidence about 1 when setting $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$.

Similarly, the models can classify the adversarial image constructed by BIM as the right label '1' when ϵ is 0.001. However, the adversarial image can be identified as the wrong label '7' when ϵ is 0.005. Thus, our and Han's compressed model with no defensive method are vulnerable to the FGSM and BIM attacks.

Secondly, we observe the effectiveness of our defense method against the adversarial image built by FGSM and BIM. From Fig. 25, Fig. 26, Fig. 27, Fig. 28, we can see that our robust model and Han's robust model protected with our defensive mechanism can defend successfully the adversarial image generated by FGSM and BIM. The models can classify the adversarial image as the right label with confidence as high as about 1 for a large scale of ϵ (i.e. $\epsilon = 0.005, 0.01, 0.05, 0.1, 0.5, 1$).

VII. CONCLUSION

In this work, we design a framework for constructing a robust DCNN-based compressed model in the edge computing environment. The mobile is partitioned and generated by the mobile device and the edge server. Training the model includes LOCAL TRAINING on the mobile device side and EDGE COMPRESSED TRAINING on the edge server side. The robust compressed model is generated by model initialization, model compression and model robustness. Specially, a defensive mechanism is proposed for enhancing the robustness of the compressed model against adversarial examples in model robustness. For accuracy guarantee, we care about the weight distribution of the compressed model in the defense method. The small-scale compressed model distributed between the mobile device and the edge server can be regarded as a robust collaborative device-server inference for offering recognition tasks. Furthermore, it is practical to deploy on the mobile device due to its small-size. Experimental results show that the generated compressed model has an attractive trade-off between accuracy and robustness.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[4] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," 2014, *arXiv:1412.7062*. [Online]. Available: https://arxiv.org/abs/1412.7062

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[7] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[8] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 164–171.

[9] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[10] 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*. [Online]. Available: https://arxiv.org/abs/1510.00149

[11] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published.

[12] H. Yin, Z. Qin, L. Ou, and K. Li, "A query privacy-enhanced and secure search scheme over encrypted data in cloud computing," *J. Comput. Syst. Sci.*, vol. 90, pp. 14–27, Dec. 2017.

[13] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han, "Mobile big data analytics using deep learning and apache spark," *IEEE Netw.*, vol. 30, no. 3, pp. 22–29, Jun. 2016.

[14] J. Liu, H. Guo, H. Nishiyama, H. Ujikawa, K. Suzuki, and N. Kato, "New perspectives on future smart FiWi networks: Scalability, reliability, and energy efficiency," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1045–1072, 2nd Quart., 2016.

[15] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[16] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 31–36.

[17] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms," in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, 2018, pp. 1–6.

[18] Z. Tao and Q. Li, "eSGD: Communication efficient distributed deep learning on the edge," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–6.

[19] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, Boston, MA, USA, 2018, pp. 1–6.

[20] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[21] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, Feb. 2015.

[22] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 62–79.

[23] 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)*, 2016, pp. 372–387.

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1765–1773.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: https://arxiv.org/abs/1312.6199

[26] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: https://arxiv.org/abs/1412.6572

[27] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: https://arxiv.org/abs/1607.02533

[28] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," 2017, *arXiv:1705.07204*. [Online]. Available: https://arxiv.org/abs/1705.07204

[29] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[30] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[31] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: https://arxiv.org/abs/1503.02531

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

[33] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," 2019, *arXiv:1903.03472*. [Online]. Available: https://arxiv.org/abs/1903.03472

[34] E. Ackerman, "How drive. ai is mastering autonomous driving with deep learning," *IEEE Spectr. Mag.*, to be published.

[35] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, and T. R. Hughes, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, p. 1254806, 2015.

[36] Y. Liu, W. Zhang, S. Li, and N. Yu, "Enhanced attacks on defensively distilled deep neural networks," 2017, *arXiv:1711.05934*. [Online]. Available: https://arxiv.org/abs/1711.05934

[37] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3422–3426.

[38] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and rejecting adversarial examples robustly," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 446–454.

[39] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017, *arXiv:1702.04267*. [Online]. Available: https://arxiv.org/abs/1702.04267

[40] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," 2017, *arXiv:1704.04960*. [Online]. Available: https://arxiv.org/abs/1704.04960

[41] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.

[42] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, 2010, pp. 177–186.

[43] 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.

YUSHUANG YAN received the B.S. degree from the School of Telecommunications Engineering, Shandong University of Science and Technology, in 2012, and the M.S. degree from the School of Telecommunications Engineering, Xidian University, in 2015, where she is currently pursuing the Ph.D. degree. Her research interests include information security, and security and privacy in edge computing and machine learning.

QINGQI PEI received the B.S., M.S., and Ph.D. degrees in computer science and cryptography from Xidian University, in 1998, 2005, and 2008, respectively. He is currently a Professor and a Member of the State Key Laboratory of Integrated Services Networks, also a Professional Member of ACM and a Senior Member of the Chinese Institute of Electronics and the China Computer Federation. His research interests include digital contents protection and wireless networks and security.