DefQ: Defensive Quantization Against Inference Slow-Down Attack for Edge Computing

Han Qiu, Tianwei Zhang, *Member, IEEE*, Tianzhu Zhang, Hongyu Li, *Member, IEEE*, and Meikang Qiu, *Senior Member, IEEE*

***Abstract*—The novel multiexit deep neural network (DNN) architectures provide a new optimization solution for efficient model inference in edge systems. Inference of most samples can be completed within the first few layers on an edge device without the need to transmit them to a remote server. This can significantly increase the inference speed and system throughput, which is particularly beneficial to the resource-constrained scenarios. Unfortunately, researchers proposed an inference slow-down attack against this technique, where an external adversary can add imperceptible perturbations on clean samples to invalidate the multiexit mechanism. In this article, we propose a defensive quantization (`DefQ`) method as the *first* defense against the inference slow-down attack. It is designed to be lightweight and can be easily implemented in off-the-shelf camera sensors. Particularly, `DefQ` introduces a novel quantization operation to preprocess the input images. It is capable of removing the perturbations from the malicious samples and preserving the correct inference exit points and prediction accuracy. Meanwhile, it has little impact on the clean samples. Extensive evaluations show that `DefQ` can effectively defeat the inference slow-down attack and well protect the efficiency of edge systems.**

***Index Terms*—Deep learning (DL), edge computing, inference slow-down, security.**

I. Introduction

In THE past decade, researchers have proposed various deep learning (DL) algorithms and models (e.g., convolutional neural networks (CNNs) [1], recurrent neural networks (RNNs) [2], deep reinforcement learning (DRL) [3]), to solve complex tasks in different domains, such as computer vision, natural language processing (NLP), and autonomous driving. Those state-of-the-art models have been extensively commercialized in many products and perfectly integrated with modern edge computing systems. By hosting deep neural network (DNN) models on the edge devices, the Internet-of-Everything (IoE) system becomes more intelligent to interpret and interact with the physical world in a more efficient fashion.

New emerging DNN models are becoming more complicated, with an increased number of parameters, layers, and computations. This guarantees they have higher performance and generalization to handle different types of data, at the cost of powerful computing capabilities. In general, the required computation for DL research doubles every few months, resulting in an increase of about 300 000× from 2012 to 2018.[1] This phenomenon adds difficulties to deploy state-of-the-art models on the IoT devices, which are generally resource- and computation-constrained. This becomes more severe in some critical scenarios which have high requirements for the inference speed and throughput [4]. Novel solutions are urgently needed to optimize the utilization of DNN models on tiny computing devices and small-scale systems.

In order to deploy increasingly complex DNN models into the edge computing scenarios such as Intelligent IoT, many different methodologies have been proposed to simplify the computation of the DNN inference process or to compress the DNN-related storage. They can be classified into the following categories. First, more compact DNN architectures were designed to adapt to the edge and mobile devices, such as MobileNets [5], SqueezeNet [6], and ShuffleNets [7]. They can achieve comparable accuracy with those complicated models, using much fewer parameters and computations. Besides, adaptive neural networks (AdNNs) [8] were proposed aiming at saving the energy consumption of inference by dynamically deactivating parts of its model based on the need of the inputs. However, this technique cannot reduce the model size, hindering the deployment of large models on small devices. Second, researchers proposed model compression [9] to reduce the model size while preserving the performance. Different methods were introduced to achieve this goal, including model pruning [9], [10], quantization [11], precision reduction [12], distillation [13], etc. According to [14], a typical ResNet-50 model can be compressed by removing 75% of its parameters while preserving similar performance. There still exists a tradeoff between the compression ratio and model accuracy drop. Third, split learning was proposed for distributed inference [15]–[19]. Instead of compressing and putting the entire DNN model to one edge device, they aim to fragment the

Manuscript received 9 July 2021; revised 14 November 2021; accepted 20 December 2021. Date of publication 29 December 2021; date of current version 6 February 2023. This work was supported by the Natural Science Foundation of China under Grant 62106127. *(Corresponding author: Meikang Qiu.)*

Han Qiu is with the Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing 100084, China (e-mail: qiuhan@tsinghua.edu.cn).

Tianwei Zhang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: tianwei.zhang@ntu.edu.sg).

Tianzhu Zhang is with Nokia Bell Labs, 91620 Nozay, France (e-mail: tianzhu.zhang@nokia-bell-labs.com).

Hongyu Li is with the Department of Information and Telecommunications Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: l543306408@bupt.edu.cn).

Meikang Qiu is with the Department of Computer Science, Texas A&M University at Commerce, Commerce, TX 77843 USA (e-mail: qiumeikang@yahoo.com).

Digital Object Identifier 10.1109/JIOT.2021.3138935

[1] openai.com/blog/ai-and-compute

DNN model into multiple parts by the layer, and distribute them to different devices. A common practice is to deploy the first few layers on the resource-constrained edge device while offloading the rest layers to the remote server for acceleration. This collaborative mode can remarkably enhance the inference speed and throughput.

Recently, researchers proposed the multiexit DNN architecture [20], which allows the model to make predictions for certain input samples at earlier stages. When a sample can be classified by a front layer with high confidence, the model can simply finish the inference task and output the results, without involving the rest layers. This early-exit mechanism can significantly save inference time and energy. It can be adopted cooperatively with the split learning: by deploying different layers (exit points) to the edge device and remote server, the prediction of most samples can be completed at the local side, avoiding the time cost from network transmission and remote server computation. This is much more efficient compared to the original edge-cloud systems.

Unfortunately, the multiexit DNN architecture is vulnerable to the inference slow-down attack, which can invalidate the early-exit mechanism and force the inference samples to go through the entire model. Hong *et al.* [21] designed the DeepSloth attack to achieve this goal. The adversary can use the gradient-based technique to generate imperceptible perturbations and add them to the clean samples. Then, these malicious samples will confuse each internal classifier for prediction, and fail to meet the early-stop criteria at each point. Thus, they have to travel across each layer on the edge device and remote server, as well as the network, to obtain the final results. Experiments in [21] demonstrated that the DeepSloth attack can achieve 100% success rate against the existing multiexit models. Moreover, the model accuracy is also terribly compromised. Although this attack can bring severe threats to the edge systems in terms of efficiency and performance, to the best of our knowledge, there are currently no defense solutions to address this issue yet.

In this article, we propose a defensive auantization (`DefQ`), the *first* methodology to defeat the inference slow-down attack in the edge computing scenario. The essential component of `DefQ` is a lightweight transformation function to process all the input samples before feeding them into the multiexit DNN models. Particularly, we use a statistical manner to understand the malicious perturbations of the inference slow-down attack in the frequency domain and design a defensive quantization table for the preprocessing operation. During inference, the transformation function adopts this defensive quantization table to remove the potential malicious perturbations in the frequency domain. Meanwhile, it will not affect the normal samples. We conducted extensive evaluations on three mainstream multiexit models and three l bound-based DeepSloth attacks. Experimental results show that `DefQ` can effectively mitigate the inference slow-down threat, and recover the early-exit mechanism for compromised samples. Besides, `DefQ` can also improve the prediction accuracy of these samples.

The key contributions of this article are as follows.

1) The first defense method against the inference slow-down attack in intelligent edge computing systems.
2) A statistical method to understand the malicious perturbations brought by the inference slow-down attack.
3) An effective yet lightweight defensive quantization method to mitigate the inference slow-down attack and preserve the model's accuracy.

This article is organized as follows. Section II discusses the research background and related works. Section III presents the threat model and defense requirements. Section IV analyzes the attack and describes the defense details. Section V presents the evaluation results. We discuss and list future works in Section VI and conclude in Section VII.

II. Background and Related Works

In this section, we briefly present the background and relevant works about intelligent edge systems, multiexit DNN models, and inference slow-down attacks.

A. Artificial Intelligence on the Edge

The development of DL technology makes today's IoT ecosystems more intelligent and powerful. It becomes practical to deploy DNN models on edge devices for complex tasks. A conventional Artificial Intelligence-of-Things (AIoT) system consists of multiple frontend sensors for data collection and preprocessing, backend edge devices, and remote servers [22] for DNN inference. They are normally connected via wireless channels (e.g., 5G and Wi-Fi) to collaboratively sense, interpret and understand the environment. Such intelligent IoT systems have been widely adopted in many scenarios, such as face authentication [23], fire alarming system [24], and remote monitoring [25].

Modern edge systems are required to process the sensor data promptly. For instance, there can be a large number of high-resolution cameras generating real-time data continuously for online analysis [26]. However, the processing speed is restricted by the limited computing capabilities and resources of edge devices. One promising solution is to leverage the powerful compute servers to accelerate the inference process. We can split a DNN model into two parts, and offload the second part to the remote server. The collaborative model across the edge and server can increase the inference speed even when we consider the network latency. The design of multiexit DNN models can better support such model split and enhance the edge system efficiency, as introduced below.

B. Multiexit DNN Architecture

To accelerate the inference on resource-constrained devices, one promising research direction is to build multiexit DNN architectures, which can selectively make predictions at earlier layers [27]. The key idea is to insert multiple classifiers (i.e., exit points) at different layers in a DNN model. The sample to be analyzed will go through each layer one by one, and each classifier attempts to make a prediction. If one classifier has sufficient confidence to predict this sample, i.e., the stop criteria are met, the inference task for this sample will be done at this exit point, and the predicted label will be assigned. Fig. 1 shows the structure of a multiexit DNN model. A number of

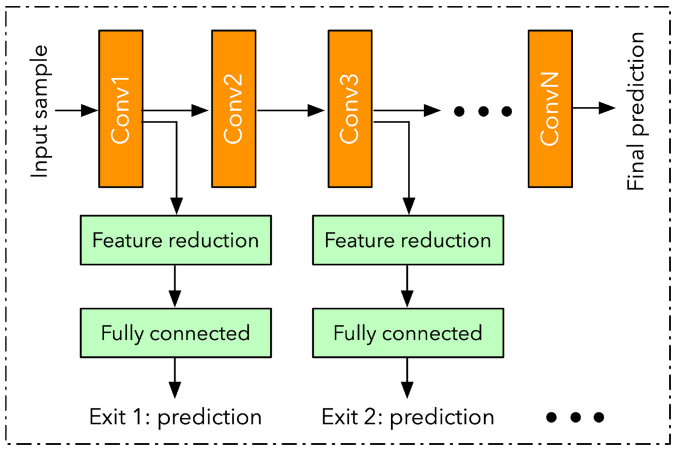


Fig. 1. Building an SDN [28] multiexit model by adding multiple exit points for prediction on a typical CNN model.

works have proposed different algorithms and methods to support this feature. For instance, Huang *et al.* [20] designed the multiscale dense networks (MSDNets) architecture, which can provide any time predictions and avoid any loss of prediction accuracy when directly modifying existing DNN architectures. Kaya *et al.* [28] proposed the shallow-deep network (SDN), which can directly convert a well-trained DNN model into a multiexit one by inserting internal classifiers. The basic structure of these internal classifiers is designed as a feature reduction step followed by a fully connected layer classifiers. The inference will stop and the sample will exit once the sample's result at one exit point meets a preset threshold.

This multiexit DNN architecture provides great opportunities to enable distributed inference with higher efficiency on the edge computing system. For instance, we can deploy the first few layers and exit points of any DNN model on the local edge device, and migrate the rest computation to the remote powerful server. Then, some of the input samples can be predicted and exit only on the edge device without any requirement for transmission and further computation cost on remote servers. Hence, even a tiny edge device can be involved for the complex model serving tasks since inference of a certain ratio of samples can be completed within the first few samples on the edge device to reduce the network latency. This is much more efficient than the traditional DNN split learning task [16]–[19], where every sample has to go through all the layers on the edge and server to finish the prediction.

C. Inference Slow-Down Attack

Inference slow-down attack is proposed [21], aiming to slow down the inference speed of multiexit models. An adversary can inject imperceptible perturbation on a clean sample, which makes the internal classifier fail to meet the early-exit threshold at each exit point. Then, this sample has to go through every layer and obtain the final prediction results only at the last layer. This can significantly increase the inference time and computation energy. Such perturbation can be generated using the gradient-based approach to disable all the exit points. Note that the goal of this attack is different from adversarial examples (AEs), which aim to change the prediction label.

This inference slow-down attack is particularly severe for the edge computing context. First, it significantly compromises the inference efficiency by forcing all the samples to pass all the convolutional layers, resulting in a huge waste of time and energy for the multiexit model inference. Second, it causes a majority of samples to be transmitted from the edge to the cloud for the final prediction. The incurred network latency can amplify the inference time by 1.5×–5×, negating the benefits of split learning for the edge–cloud setting. Third, as a side effect, the model accuracy will also be decreased by the added perturbation. To the best of our knowledge, there is currently a lack of defense solutions for this severe threat.

III. Threat Model and Defense Requirements

Threat Model: We consider an artificial intelligent edge system for computer vision tasks (e.g., image classification, object detection, etc). This system consists of three entities: frontend sensors keep collecting the images at a high sampling rate and sending them to edge devices and remote servers, which collaboratively host a multiexit DNN model for inference. Since an edge device has limited computing capabilities and memory, it only deploys the first few layers and exit points, while offloading the rest computations to the remote server. We focus on computer vision applications for two reasons. First, vision sensors, such as cameras are one of the most widely used IoT devices in our daily life. Computer vision tasks are also commonly adopted in many scenarios, e.g., video surveillance [29], face authentication [23], autonomous driving [30], etc. Second, compared to other sensor data, vision sensors can produce a larger volume of real-time streaming data with higher throughput. Thus, edge systems for computer vision are in more urgent need of efficient inference solutions.

The entire edge system (e.g., frontend sensors, edge devices, remote servers, and their communication networks) is assumed to be trusted. So we do not consider the security threats from inside the system (e.g., botnet Mirai [31], Hajime [32]) or man-in-the-middle network attacks. The adversary is outside of the edge system, attempting to spoof the sensor data by adding malicious perturbations on the physical objects, camera lens, or digital images. He has full knowledge of the target DNN model but is not allowed to tamper with the parameters (e.g., DNN backdoor attacks [33]). The adversary's goal is to feed malicious images to the system to significantly reduce its efficiency: the DNN model has to go through all the layers to process these images with much longer time cost, resulting in more energy consumption and less throughput. Such inference slow-down attacks have been introduced in [21]. Attacks aiming at other purposes such as AEs, backdoor, and data poisoning are not within the scope of this article.

Defense Requirements: The goal of this article is to design a novel and effective methodology for mitigating inference slow-down attacks against multiexit DNN models. We assume the defender will not modify or enhance the target model itself. For instance, the DNN model can be purchased from a model vendor and deployed with specific settings. It may not be

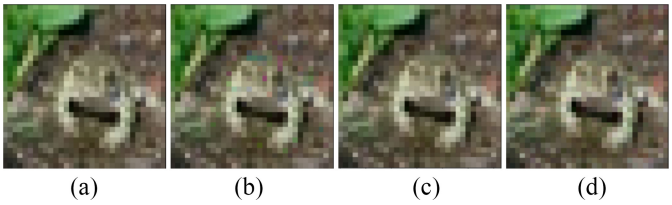


Fig. 2. Example of DeepSloth attack on a ResNet-56 model: (a) a clean sample exits at the 1st exit point; while (b) L_1-based; (c) L_2-based; and (d) L_∞-based attack samples miss all the 27 exit points.

allowed or practical for users to alter or retrain the model. The defender only implements transformation functions in front of the DNN model to preprocess the input images. The functions need to meet the following requirements.

1) *Effectiveness:* They should be able to rectify the malicious samples, forcing them to follow the correct exits in the model. Meanwhile, they should not affect the exit points of benign samples.
2) *Accuracy-Preserving:* They should have little impact on the prediction accuracy of malicious or benign samples.
3) *Lightweight:* The computation of these functions should not be too heavy to affect the operation of the edge systems, considering the limited onboard computing capabilities and resources of edge devices.

IV. Proposed Defense Methodology

We present a novel approach DefQ to protect multi-exit DNN models from inference slow-down attacks. First, we give a detailed analysis on a representative attack – DeepSloth [21] in Section IV-A. Then, we provide the methodology overview in Section IV-B, followed by the detailed designs in Section IV-C and defense security analysis in Section IV-D.

A. DeepSloth Attack Analysis

Hong *et al.* [21] proposed the DeepSloth attack, which can slow down the inference process of a multiexit model by adding malicious perturbations to the benign samples. The core idea of the DeepSloth attack is to make the samples never meet the exit thresholds at each exit point of the DNN model. Specifically, by adding carefully crafted perturbations on input samples, an adversary can manipulate the representation of each layer in the model and push the outputs of the internal classifier at each exit point toward a uniform distribution. Apparently, a uniform distribution of the internal classifiers' outputs cannot meet any thresholds to exit at these exit points. Hence, each internal classifier cannot make confident decisions about the samples, and deeper layers are needed for classification. Fig. 2 shows the visual results of an attack example on a ResNet-56-SDN model with 27 exit points. A clean sample (a) will exit at the 1st exit point. DeepSloth generates imperceptible perturbations bounded by the L_1-norm (b), L_2-norm (c), or L_∞-norm (d), which make the samples miss all the 27 exit points and obtain the prediction at the final layer. Note that this attack is fundamentally different from the previous adversarial attacks, which aim to change the predictions and trigger the misclassification at earlier layers.

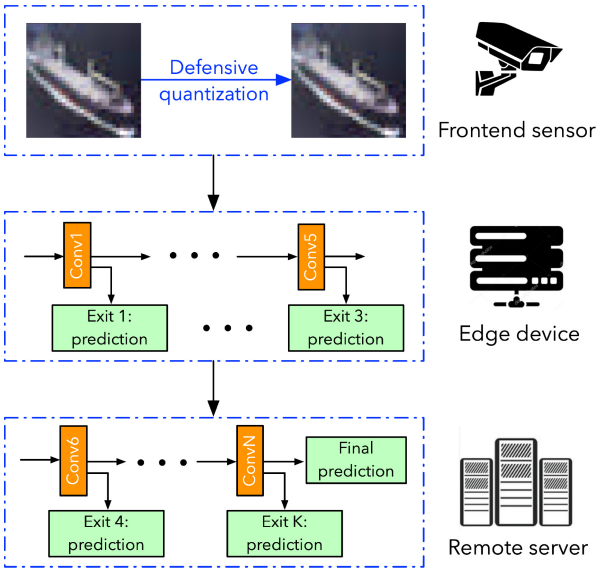


Fig. 3. Overview of our defensive quantization-based methodology.

Efficiency Analysis: The primary target of the DeepSloth attack is inference efficiency. The adversary's goal is to cause the inference samples to fail the criteria of all the early exit points, until reaching the final prediction layer (Fig. 1). This will compromise the system efficiency from two aspects: for each infected sample, the inference time will be significantly delayed as it needs to go through more convolutional layers or even the network between the edge device and remote server. For the entire system, the edge device needs to spend more time to classify the samples; hence, the inference throughput is reduced and more energy is wasted.

Accuracy Analysis: Preserving or decreasing the prediction accuracy is not a target of DeepSloth. Although Hong *et al.* [21] claimed that they try not to affect the prediction results, our experimental results indicate the prediction accuracy is still decreased to some extent. How to design more efficient inference slow-down attacks without affecting the model accuracy is beyond the scope of this work.

B. Overview of DefQ

Fig. 3 shows an overview of our proposed methodology, DefQ. The core of DefQ is a defensive quantization-based function deployed in the frontend sensors. It preprocesses each captured image before sending it to the edge server. It will not affect the execution of benign images, while effectively removing the malicious perturbations added by the inference slow-down attack. This quantization step can be integrated with the image compression and formatting operations to reduce the size of transmitted images.

In a benign case, the analysis of most images will be completed in the first few layers residing on the edge device, without going to the remote server. However, the malicious samples will go through more layers across the edge device and remote server with longer execution time and network latency. Our quantization function aims to remove the perturbations from these malicious samples and make their exit on

the edge device again. Besides, as the DeepSloth attack can also affect the prediction results of malicious samples, our quantization function can even improve the model accuracy over these samples.

In order to design a qualified defensive quantization function, we generate the quantization table by statistically computing the influence on the frequency domain in DeepSloth. We aim to use a small set of malicious samples generated from DeepSloth to get a statistical frequency influence, which can inspire us to further design the quantization table (more details are described in Section IV-C).

C. Defensive Quantization

The key idea of `DefQ` is to utilize a defensive quantization operation to remove the perturbations in the frequency domain in a lightweight manner. For an input image, we first transform it from the spatial domain to the frequency domain based on a discrete cosine transform (DCT) [34]. This DCT coefficient distribution represents the energy contribution of each frequency band. Previous works [35], [36] have shown that different frequency bands have different influence levels on DNN learning and inference.

Here, we adopt a 2D-DCT which first cut images into grids with a setting size. Pixels in each grid are transformed into the frequency space via 2D-DCT as shown in (1): $f(x, y)$ refers to the input image and C is the coefficients for the DCT calculation. The values of C can then be calculated according to the grid size $N \times N$ (in this article, we set $N = 8$)

$$
\begin{aligned}
F(u,v) &= \frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C_{x,y}(u,v) f(x,y) \\
C_{x,y}(u,v) &= \alpha(u)\alpha(v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2x+1)v}{2N}\right] \\
\alpha(x) &= \begin{cases} \sqrt{\frac{1}{N}}, & \text{when } x = 0 \\ \sqrt{\frac{2}{N}}, & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

After the 2D-DCT, we design a specific quantization method by analyzing the modification in the frequency domain caused by the DeepSloth attack. Based on the adversary's viewpoint, to bypass a quantization-based defense, the malicious perturbation in terms of pixel values must be large enough to introduce significant modifications in the frequency domain to further influence the quantized results. Therefore, the quantization-based defense can make it significantly difficult or even impossible for the adversary to craft the perturbations. Previous works aim to design a specific quantization table to mitigate the adversarial attacks for misleading the DNN prediction [35]. For the DeepSloth attack, we find the malicious modifications in the frequency domain have special patterns that can be used to design a defensive quantization table.

Our novel method to generate the quantization table is presented in Algorithm 1. We generate the table Q_{def} in a statistical learning manner by summarizing the frequency patterns of the malicious samples. Here, we first build a clean sample set I_M by collecting M clean samples from the data set ($\mathbb{R}$) with width H, height W, and three layers of colors. Then, we use the DeepSloth L_∞ attack to generate a malicious sample set $\widehat{I}_M$ from the I_M. Therefore, $\widehat{I}_M$ contains M images that miss all exit points of the model. These steps will build the input of our algorithm as $I_M \in \mathbb{R}^{M\times H\times W\times 3}$ and $\widehat{I}_M \in \mathbb{R}^{M\times H\times W\times 3}$. Note that using other perturbation bounds (L_1, or L_2) for malicious sample generation will lead to similar results. Our algorithm first fragments all the images into 8×8 blocks (lines 3 and 4) on all three color layers. Then, these blocks in the spatial domain (I in line 6) are collected from all the images' color channels for both the benign image set and malicious set ($\widehat{I}$ in Line 7). By conducting DCT-2D on all the 8×8 blocks, we compare the difference of DCT frequency coefficients (line 8) to understand the perturbation ratio in the frequency domain brought by the DeepSloth attack. This is calculated by comparing the difference of all the blocks in the frequency coefficients from the clean and malicious sets accordingly. By calculating the average modification ratio, we get the modification levels for different frequency bands (line 13). Then, we normalize these statistical results of the modification in the frequency space, and remap them back into a range between 20 and 60 (line 14) as our quantization table Q_{def}. We also try different ranges such as [20, 80] or [20, 100], which yield similar defense results (see the ablation study in Section V-D).

Algorithm 1: Generating the Quantization Table Q_{def}

Input: clean set $I_M \in \mathbb{R}^{M\times H\times W\times 3}$,
malicious set $\widehat{I}_M \in \mathbb{R}^{M\times H\times W\times 3}$,
a zero matrix of size 8×8 Q_0
Output: defensive quantization table Q_{def}

1 **for** i *in* $1 \sim M$ **do**
2 **for** $I_{i,channel}$ *in* I_i **do**
3 $n_w = W/8$, $n_h = H/8$;
4 $\mathcal{G}(I_i) = \{(x_m, y_n) | (m, n) \in (0 \sim n_w, 0 \sim n_h)\}$;
5 **for** (x_m, y_n) *in* $\mathcal{G}(I_i)$ **do**
6 $F(I_{i,channel}) = DCT(I_{i,channel}(x_{m-1}{:}x_m, y_{n-1}{:}y_n))$;
7 $F(\widehat{I}_{i,channel}) = DCT(\widehat{I}_{i,channel}(x_{m-1}{:}x_m, y_{n-1}{:}y_n))$;
8 $Dif(i)_{m,n} = (|F(I_i)| + 1)/(|F(\widehat{I}_i)| + 1)$;
  $Q_0 = Q_0 + Dif_{m,n}$;
9 **end**
10 **end**
11 **end**
12 $Q_0 = Average(Q_0, M \times n_w \times n_h \times 3)$;
13 $Q_{def} = (Q_0 - min(Q_0)/max(Q_0)) \times 40 + 20$;
14 **return** Q_{def};

The statistical results in the frequency domain of comparing the clean and malicious samples are given in Fig. 4(a). We observe that the low-frequency bands, including the dc value, are slightly changed. The high-frequency bands (the lower right corner) are modified by the DeepSloth attack, but the middle frequency bands (the upper right corner) are changed the most. Fig. 4(b) shows our quantization table Q_{def} after remapping the statistical results.

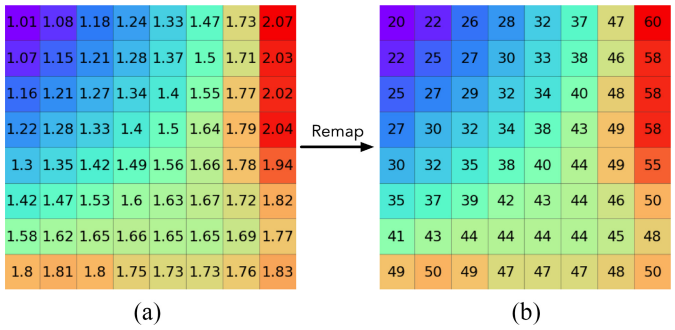


Fig. 4. DCT frequency space statistical results of DeepSloth attack for (a) random 1000 samples and (b) our defensive quantization table Q_{def}.

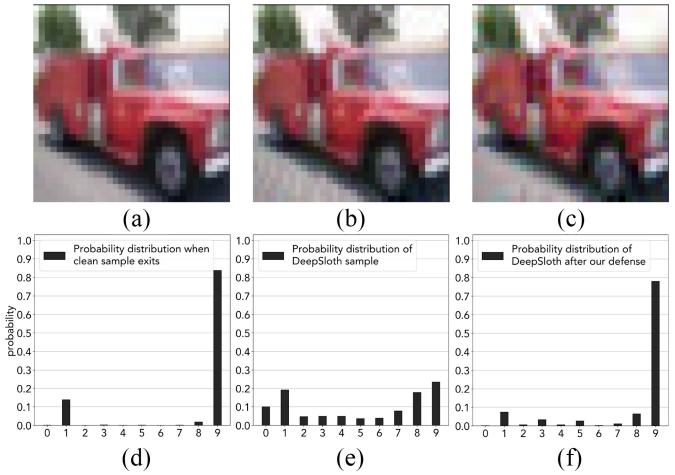


Fig. 5. Example of our defense on DeepSloth attack. The first row shows a clean image (a), a malicious image before without our defense (b), and with our quantization preprocess (c). The second row shows the corresponding confidence scores from the classifier at the 2nd exit point, (d) and (f) quit at the 2nd exit point, while image (e) quit at the final layer.

D. Security Analysis

We use a simple example to demonstrate the effects of the feature modification by our defensive quantization. Fig. 5 shows (a) a clean image, (b) malicious DeepSloth image, and (c) the output of our quantization function for this malicious image. The second row shows the predicted confidence scores at the 2nd exit point corresponding to the images in the first row. We observe that for the clean image, the classifier has high confidence to predict it as a "truck" (d). This meets the early-exit criteria and the task will be completed with the correct label. However, for the malicious image without our defense, the probability distribution of the confidence score is rather even, and the classifier cannot make the decision (e). It has to send the image to the subsequent layers for further analysis, and finally, the task is done at the last layer. Now with our defensive quantization method, the malicious perturbation in the frequency space is removed. The confidence score is recovered to the correct one, and the task can be done at the 2nd exit point. This indicates the DeepSloth attack is successfully defeated, and the multiexit scheme works again.

V. Evaluations

We comprehensively evaluate our proposed methodology over three mainstream models. We first evaluate the defense effectiveness of DefQ on DeepSloth attacks with different techniques (L_1, L_2, and L_∞). Second, we measure the impact of DefQ on the model accuracy to validate its accuracy-preserving property. Considering the defense requirements in Section III, we evaluate and give the results about the *effectiveness* in Section V-B, *accuracy preserving* and *lightweight* in Section V-C, respectively.

A. Experimental Configuration

We consider the image classification task on the CIFAR-10 data set. It contains 50 000 images for training and 10 000 images for testing. Each image has a size of $32 \times 32 \times 3$ and belongs to one of ten classes. All pixel values are normalized within the range of [0, 1]. We consider three state-of-the-art DNN models (ResNet-56 [37], VGG-16, and MobileNet [28]), and use the SDN structure [21] to convert them into multiexit models. The corresponding SDN-ResNet-56, SDN-VGG-16, and SDN-MobileNet models have 27, 14, and 14 exit points, respectively. To reproduce the DeepSloth attack, we follow the configurations in [21] under a white-box setting (i.e., the adversary has full knowledge of the victim SDN model) and generate the malicious samples with the L_1, L_2, and L_∞-based techniques. We use Pytorch 1.8 [38] as the backend for model training and inference. All the experiments are run on a server with an Intel Core i9-10900K CPU@3.70 GHz and two Nvidia GeForce RTX 3090 GPUs.

B. Defense Effectiveness Evaluation

We adopt two metrics to quantify the effectiveness of DefQ. The first one is the Top-3 Exit Ratio (T3-ER), which measures the percentage of inference samples that can be completed within the first three exit points deployed on the edge device (Fig. 3). A higher T3-ER indicates higher inference efficiency. The second metric is Efficacy (EFCY) from [21]. This metric is defined to quantify a model's ability to utilize its exit points for inference. It is a normalized value between 0 and 1: a value closer to 1 indicates more input samples will exit earlier to save inference time. We consider different thresholds for early-exit in SDN-based models. At each exit point, a confidence score will be selected and compared with a threshold to determine whether this sample should take this exit point. A smaller threshold will lead more samples to exit earlier, but the accuracy may be decreased. We set two thresholds to make the relative accuracy drop (RAD) is within 5% and 10% of its maximum accuracy, respectively.

We randomly select 1000 clean samples from the testing data set and generate the malicious samples. Table I presents the T3-ER values of different samples without and with DefQ. First, we observe that about 55%, 20%, and 53% clean samples can meet the threshold of RAD = 5% within the first three exit points for the three models, respectively. The percentage is even higher when we set the threshold as RAD = 10%. This can significantly reduce the computation cost and edge–server transmission cost. The introduced quantization function has little impact on the clean samples. In the SDN-MobileNet model, DefQ can even slightly improve T3-ER. Second, we observe that with the DeepSloth attack, almost zero samples will exit

TABLE I
TOP-3 EXIT RATIO OF SAMPLES, INCLUDING CLEAN SAMPLE AND DEEPSLOTH SAMPLES OF DIFFERENT CONSTRAINS WITH OR WITHOUT DEFQ

RAD	Sample	SDN-VGG-16		SDN-ResNet-56		SDN-MobileNet	
		original	DefQ	original	DefQ	original	DefQ
5%	Clean	0.55	0.54	0.20	0.21	0.53	0.57
	DeepSloth (L_∞)	0.00	0.26	0.00	0.07	0.00	0.27
	DeepSloth (L_2)	0.35	0.45	0.09	0.14	0.36	0.44
	DeepSloth (L_1)	0.27	0.34	0.08	0.11	0.27	0.30
10%	Clean	0.70	0.69	0.34	0.34	0.77	0.81
	DeepSloth (L_∞)	0.00	0.41	0.00	0.12	0.05	0.48
	DeepSloth (L_2)	0.50	0.58	0.21	0.26	0.57	0.69
	DeepSloth (L_1)	0.40	0.56	0.17	0.21	0.45	0.61

TABLE II
EFCY OF SAMPLES, INCLUDING CLEAN SAMPLE AND DEEPSLOTH SAMPLES OF DIFFERENT CONSTRAINS WITH OR WITHOUT DEFQ

RAD	Sample	SDN-VGG-16		SDN-ResNet-56		SDN-MobileNet	
		original	DefQ	original	DefQ	original	DefQ
5%	Clean	0.78	0.76	0.56	0.54	0.82	0.81
	DeepSloth (L_∞)	0.02	0.53	0.00	0.44	0.01	0.55
	DeepSloth (L_2)	0.42	0.66	0.16	0.44	0.43	0.70
	DeepSloth (L_1)	0.31	0.52	0.12	0.30	0.31	0.59
10%	Clean	0.85	0.84	0.69	0.67	0.92	0.92
	DeepSloth (L_∞)	0.06	0.66	0.01	0.41	0.06	0.76
	DeepSloth (L_2)	0.57	0.78	0.27	0.58	0.65	0.85
	DeepSloth (L_1)	0.45	0.76	0.21	0.55	0.54	0.79

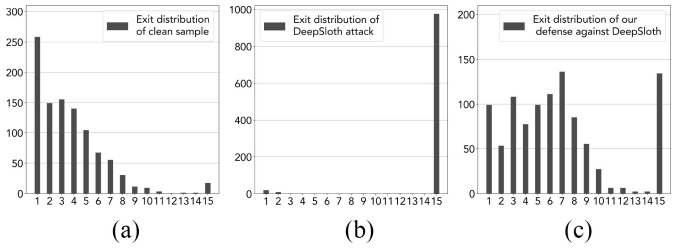


Fig. 6. Exit distribution comparison between (a) 1000 clean samples; (b) clean samples attacked by DeepSloth; and (c) our defense against DeepSloth on SDN-VGG-16 model with RAD = 5.

within the first three exit points under the two thresholds. Most samples must be sent from the edge device to the remote server with higher latency. With DefQ, we see that T3-ER is significantly improved, indicating the successful mitigation of the DeepSloth attack.

Table II shows the EFCY results which are similar to T3-ER. This confirms the effectiveness of DefQ on both benign and malicious samples against different techniques.

We also analyze the distribution of exit points over 1000 samples. We use the SDN-VGG-16 model as an example, which contains 14 exit points and one final prediction point. Fig. 6 shows the results for (a) the clean samples, (b) malicious samples, and (c) transformed malicious samples with our solution. We find that most clean samples can complete the inference process successfully within the first ten exit points. In contrast, the majority of DeepSloth samples will be forced to take the final point. With our defense, the DeepSloth samples become normal again after the transformation, and more than 85% of them can be done within the first ten exit points. This proves that our solution can maintain the functionality of the multiexit mechanism under the inference slow-down attacks.

TABLE III
MODEL ACCURACY OF DIFFERENT SAMPLES

RAD	Sample	SDN-VGG-16		SDN-ResNet-56		SDN-MobileNet	
		original	DefQ	original	DefQ	original	DefQ
5%	Clean	0.84	0.82	0.80	0.77	0.78	0.76
	DeepSloth (L_∞)	0.13	0.72	0.21	0.69	0.21	0.67
	DeepSloth (L_2)	0.67	0.79	0.57	0.73	0.61	0.74
	DeepSloth (L_1)	0.52	0.71	0.45	0.71	0.54	0.70
10%	Clean	0.80	0.78	0.75	0.72	0.73	0.72
	DeepSloth (L_∞)	0.16	0.70	0.21	0.67	0.24	0.66
	DeepSloth (L_2)	0.68	0.75	0.58	0.70	0.62	0.71
	DeepSloth (L_1)	0.57	0.69	0.48	0.70	0.59	0.66

C. Model Accuracy Evaluation and Lightweight Discussion

As discussed in Section III, another defense requirement is to preserve the model accuracy on the inference samples. Table III shows the average accuracy of clean samples and DeepSloth samples generated by different techniques without and with DefQ.

1) We find that the introduced quantization transformation has a small influence on the accuracy of clean samples.
2) The DeepSloth attack can significantly decrease the accuracy of malicious samples, in addition to the exit points.
3) Our DefQ can effectively remove the perturbations and increase the model accuracy of those samples, which is close to that of clean samples. This proves that our defense method achieves the *accuracy preserving* requirement.

DefQ is designed as a special quantization operation after the DCT-2D transformation. Today, most frontend camera sensors have the computing capabilities of compressing images before sending them to edge devices. The most common image compression procedure such as the JPEG standard includes a typical DCT-2D transformation, quantization, and coding operations. Therefore, our defensive quantization function can be easily integrated into this pipeline without any additional computation requirements. Therefore, DefQ meets the *lightweight* requirement.

D. Ablation Study of the Quantization Table

In Section IV-C, we statistically calculate the influence brought by the DeepSloth attack in the frequency domain and remap them into an integer range to generate the quantization table. We compare different ranges of the quantization table during the remapping: [20, 60], [20, 80], and [20, 100]. We adopt the SDN-VGG-16 model and the L_∞ attack, and the evaluation results are shown in Table IV. We observe that the effectiveness and accuracy metrics are similar for different quantization tables. This demonstrates that DefQ is effective based on the statistical understanding of the influence in the frequency domain.

VI. DISCUSSION AND FUTURE WORK

Advanced Attacks and Defenses: In this article, we mainly focus on the mitigation of DeepSloth, the most common inference slow-down attack against computer vision tasks. Even the adversary knows our quantization function, he cannot generate the desired samples using the gradient-based technique,

TABLE IV
ABLATION STUDY ON THE RANGE OF THE QUANTIZATION TABLE

RAD	Sample		SDN-VGG-16		
			T3-ER	EFCY	ACC
5%	Clean		0.55	0.78	0.84
	DeepSloth (L_∞)	Original	0.00	0.02	0.13
		DefQ ([20, 60])	0.26	0.53	0.72
		DefQ ([20, 80])	0.25	0.54	0.72
		DefQ ([20, 100])	0.27	0.55	0.70
10%	Clean		0.70	0.85	0.79
	DeepSloth (L_∞)	Original	0.00	0.06	0.16
		DefQ ([20, 60])	0.41	0.66	0.70
		DefQ ([20, 80])	0.38	0.63	0.72
		DefQ ([20, 100])	0.42	0.65	0.71

since the quantization process is nondifferentiable. One possible attack method is to adopt the backward pass differentiable approximation (BPDA) technique to approximate the gradient of the quantization step. However, BPDA AEs can also be mitigated by advanced gradient obfuscation methods [39], and we believe they can defeat such advanced inference slow-down attacks as well. In the future, we will design and evaluate more sophisticated inference slow-down attacks and countermeasures.

Future Testing in Real-World Systems: We measure the impacts of attacks and our defense on three popular models. We plan to implement our DefQ in real-world edge systems as the future work. Since we have analyzed the discussed the practical computation steps and costs in Section V-C, so our lightweight purpose will hold in real-world edge systems. For the future implementation, we will adopt the edge devices (e.g., raspberry pi, Nvidia Jetson Nano) and remote cloud services to host the computer vision tasks (e.g., face authentication [23], remote monitoring [25]). We will also generate the DeepSloth samples to attack the system and use DefQ to mitigate them. The inference speed, network latency cost, and energy consumption will be measured to demonstrate its practicality and effectiveness.

VII. CONCLUSION

In this article, we proposed DefQ, a novel approach to effectively mitigate the inference slow-down attacks. We designed a quantization-based transformation to preprocess input images, which is able to remove the perturbations on the malicious samples. Then, these samples will follow the correct exit points to improve the inference efficiency and system throughput. Meanwhile, their prediction accuracy is also significantly increased. We also validated that DefQ is able to maintain the efficiency and accuracy of clean samples. DefQ is lightweight and can be deployed in off-the-shelf image sensors to protect the computer vision tasks in artificial intelligent edge systems.

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. 1106–1114.

[2] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, 2013, pp. 6645–6649.

[3] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] K. Gai and M. Qiu, "Optimal resource allocation using reinforcement learning for IoT content-centric services," *Appl. Soft Comput.*, vol. 70, pp. 12–21, Sep. 2018.

[5] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[7] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[8] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 420–436.

[9] 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*.

[10] C. Dai, X. Liu, H. Cheng, L. T. Yang, and M. J. Deen, "Compressing deep model with pruning and tucker decomposition for smart embedded systems," *IEEE Internet Things J.*, early access, Sep. 29, 2021, doi: 10.1109/JIOT.2021.3116316.

[11] S. Xu *et al.*, "Generative low-bitwidth data free quantization," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 1–17.

[12] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5406–5414.

[13] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*.

[14] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[15] J. Hauswald, T. Manville, Q. Zheng, R. G. Dreslinski, C. Chakrabarti, and T. N. Mudge, "A hybrid approach to offloading mobile image classification," in *Proc. IEEE Int. Conf. Acoustic Speech Signal Process. (ICASSP)*, 2014, pp. 8375–8379.

[16] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Architect. News*, vol. 45, no. 1, pp. 615–629, 2017.

[17] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[18] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 148–162.

[19] Z. He, T. Zhang, and R. B. Lee, "Attacking and protecting data privacy in edge-cloud collaborative inference systems," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9706–9716, Jun. 2021.

[20] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. ICLR*, 2018.

[21] S. Hong, Y. Kaya, I.-V. Modoranu, and T. Dumitraş, "A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference," in *Proc. ICLR*, 2021.

[22] M. Qiu, Z. Ming, J. Wang, L. T. Yang, and Y. Xiang, "Enabling cloud computing in emergency management systems," *IEEE Cloud Comput.*, vol. 1, no. 4, pp. 60–67, Nov. 2014.

[23] 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.*, 2018, pp. 1–6.

[24] Z. Shouran, A. Ashari, and T. K. Priyambodo, "Internet of Things (IoT) of smart home: Privacy and security," *Int. J. Comput. Appl.*, vol. 182, no. 39, pp. 3–8, 2019.

[25] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2014, pp. 858–862.

[26] Q. Zhang, T. Huang, Y. Zhu, and M. Qiu, "A case study of sensor data collection and analysis in smart city: Provenance in smart food supply chain," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 11, 2013, Art. no. 382132.

[27] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

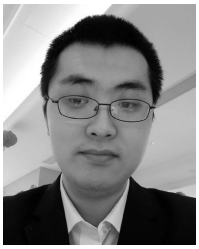
[28] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2019, pp. 3301–3310.
[29] Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system," *Multimedia Tools Appl.*, vol. 76, no. 4, pp. 5817–5832, 2017.
[30] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 446–454.
[31] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[32] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of Hajime, a peer-to-peer IoT botnet," in *Proc. NDSS*, 2019, pp. 1–15.
[33] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, 2019, pp. 707–723.
[34] R. Reininger and J. Gibson, "Distributions of the two-dimensional DCT coefficients for images," *IEEE Trans. Commun.*, vol. 31, no. 6, pp. 835–839, Jun. 1983.
[35] Z. Liu *et al.*, "Feature distillation: DNN-oriented JPEG compression against adversarial examples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 860–868.
[36] H. Qiu, Q. Zheng, T. Zhang, M. Qiu, G. Memmi, and J. Lu, "Toward secure and efficient deep learning inference in dependable IoT systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3180–3188, Mar. 2021.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.
[38] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.
[39] H. Qiu, Y. Zeng, Q. Zheng, S. Guo, T. Zhang, and H. Li, "An efficient preprocessing-based approach to mitigate advanced adversarial attacks," *IEEE Trans. Comput.*, early access, Apr. 30, 2021, doi: 10.1109/TC.2021.3076826.

Han Qiu received the B.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2011, the M.S. degree from Telecom-ParisTech (Institute Eurecom), Biot, France, in 2013, and the Ph.D. degree in computer science from the Department of Networks and Computer Science, Telecom-ParisTech, Paris, France, in 2017.

He worked as a Postdoctoral Fellow and a Research Engineer with Telecom Paris, Paris, and LINCS Lab, Paris, from 2017 to 2020. He is currently an Assistant Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing. His research interests include AI security, big data security, and the security of intelligent transportation systems.

Tianwei Zhang (Member, IEEE) received the bachelor's degree from Peking University, Beijing, China, in 2011, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 2017.

He is an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems.



Tianzhu Zhang received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2012, and the M.S. and Ph.D. degrees from the Politecnico di Torino, Turin, Italy, in 2014 and 2017, respectively.

In August 2020, he joined Nokia Bell Labs, Nozay, France, where he is a Research Engineer. From 2017 to 2019, he was a PostDoctoral Researcher with Telecom ParisTech, Paris, France, and LINCS, Paris, under a research grant from Cisco Systems, San Jose, CA, USA. His current research interests include SDN/NFV, artificial intelligence, edge computing, robotics, big data, and log analysis.

Hongyu Li (Member, IEEE) received the B.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2021, where he is currently pursuing the master's degree with the State Key Laboratory of Networking and Switching Technology.

His research interests include deep learning and AI security.

Meikang Qiu (Senior Member, IEEE) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1992 and 1998, respectively, and the Ph.D. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2007.

He is the Department Head and a tenured Full Professor with Texas A&M University at Commerce, Commerce, TX, USA. He has published more than 20 books and more than 600 peer-reviewed journal and conference papers, including more than 100 IEEE/ACM Transactions papers. His research interests include cybersecurity, big data analysis, cloud computing, smarting computing, intelligent data, and embedded systems.

Dr. Qiu is an Associate Editor of more than ten international journals, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON BIG DATA, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS. He is the Chair of IEEE Smart Computing Technical Committee. He is an ACM Distinguished Member (2019), a Highly Cited Researcher (2020, Web of Science), and an IEEE Distinguished Visitor from 2021 to 2023. He is an ACM Distinguished Member.