Toward secure and efficient deep learning inference in dependable IoT systems

Qiu, Han; Zheng, Qinkai; Zhang, Tianwei; Qiu, Meikang; Memmi, Gerard; Lu, Jialiang

2021

Qiu, H., Zheng, Q., Zhang, T., Qiu, M., Memmi, G. & Lu, J. (2021). Toward secure and efficient deep learning inference in dependable IoT systems. IEEE Internet of Things Journal, 8(5), 3180-3188. https://dx.doi.org/10.1109/JIOT.2020.3004498

https://hdl.handle.net/10356/148325

https://doi.org/10.1109/JIOT.2020.3004498

Towards Secure and Efficient Deep Learning Inference in Dependable IoT Systems

Han Qiu, *Member, IEEE,* Qinkai Zheng, Tianwei Zhang, Meikang Qiu, *Senior Member, IEEE,*
Gerard Memmi, *Member, IEEE,* and Jialiang Lu

***Abstract*—The rapid development of Deep Learning (DL) enables resource-constrained systems and devices (e.g., Internet of Things) to perform sophisticated Artificial Intelligence (AI) applications. However, AI models like Deep Neural Networks (DNNs) are known to be vulnerable to Adversarial Examples (AEs). Past works on defending against AEs require heavy computations in the model training or inference processes, making them impractical to be applied in IoT systems. In this paper, we propose a novel method, SUPER-IOT, to enhance the security and efficiency of AI applications in distributed IoT systems. Specifically, SUPER-IOT utilizes a pixel drop operation to eliminate adversarial perturbations from the input and reduce network transmission throughput. Then it adopts a sparse signal recovery method to reconstruct the dropped pixels and wavelet-based denoising method to reduce the artificial noise. SUPER-IOT is a lightweight method with negligible computation cost to IoT devices and little impact on the DNN model performance. Extensive evaluations show that it can outperform three existing AE defensive solutions against most of the AE attacks with better transmission efficiency.**

***Index Terms*—IoT, Deep Learning, Security, Adversarial Examples.**

I. INTRODUCTION

The past decade has witnessed the revolutionary development of Deep Learning (DL) technology with Deep Neural Networks (DNNs). A variety of DL algorithms and models were designed to perform different Artificial Intelligence (AI) tasks. For instance, Convolutional Neural Networks (CNNs) [1] show great capability in handling computer vision tasks; Recurrent Neural Networks (RNNs) [2] power the advance of natural language processing; Deep Reinforcement Learning (DRL) [3] achieves very high performance in robotics and autonomous driving. Those state-of-the-art models have been extensively commercialized in many products, and they are continuously enhanced by experts from academia as well as industry. Nowadays, new techniques have kept emerging at surprising speed to enrich the DL community.

Meanwhile, DL also drives the growth of the Internet of Things (IoT). Equipped with different sensors (e.g., cameras, microphones, gyroscopes), IoT devices become appealing targets for DL applications. They keep sensing data and information from various environmental contexts in a streaming fashion. Then DL models are deployed in centralized servers to process and understand the data. The integration of AI and IoT leads to the era of Artificial Intelligence of Things (AIoT), which have significantly changed our daily life (Figure 1): small-scaled AIoT systems are introduced to build smart homes and increase the comfort and quality of life; medium-scale AIoT systems are deployed in warehouses and factories for higher efficiency and automation; large-scale AIoT systems can contribute to the establishment of smart cities.

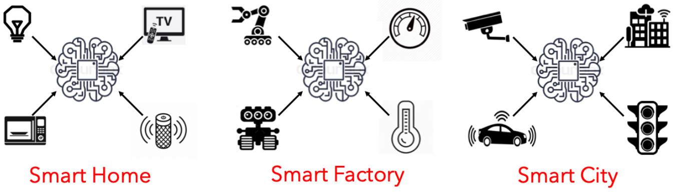

Figure 1. Different scales of Artificial Internet of Things (AIoT) systems.

Two challenges need to be addressed for the deployment of DL models in the AIoT systems. The first one is *efficiency*. An IoT system can consist of a large number of edge devices with high-quality sensors streaming information at a very high rate (e.g. remote sensing [4]). This can result in a large amount of data transferring between the sensor devices and the model server [5]. There could be a performance bottleneck if the network bandwidth of the AIoT system is not high, or an energy bottleneck if the transmission energy budget is low. Thus, it is necessary to have an efficient approach to processing the data at the sensor device before sending them to the model server, in order to reduce the throughput and transmission energy dissipation.

The second challenge we need to consider is *security*. Deep learning models are well known to be vulnerable to Adversarial Examples (AE) [6]. An AE is created by adding imperceptible perturbations to a clean data sample, which can mislead the model to give a wrong decision. Past works have demonstrated that an adversary can generate such AEs of images [7], voices [8], and laser signals [9] to spoof the IoT sensors and cause catastrophic consequences. It is of paramount importance for the sensor devices to detect or prevent such malicious samples for secure model inference.

To the best of our knowledge, currently, there are no existing solutions that can solve both of the two challenges. The most promising direction is to add a preprocessing step on the input samples before feeding them into the model [10], [11],

H. Qiu, Q. Zheng, and G. Memmi are with Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France, 91120. Email: {han.qiu, qinkai.zheng, gerard.memmi}@telecom-paris.fr.

T. Zhang is with Nanyang Technological University, Singapore, 639798. Email: tianwei.zhang@ntu.edu.sg.

M. Qiu (corresponding author) is with Texas A&M University, Texas, USA, 77843. Email: qiumeikang@yahoo.com

J. Lu is with SPEIT, Shanghai Jiao Tong University, Shanghai, China, 200240. Email: jialiang.lu@sjtu.edu.cn.

[12]. Such a step introduces non-differentiable transformations on the inputs to obfuscate the gradients of the models, so the difficulty of AE generation is increased and the impact of the calculated perturbations is mitigated. However, such defense approaches are still vulnerable as the adversary can adaptively and statistically calculate the gradient based on the preprocessing algorithms [13]. Besides, some preprocessing operations can introduce heavy computation (e.g. sophisticated quantization in image compression [12]), which are not applicable to computing resource-constrained IoT devices.

In this paper, we propose SUPER-IOT: a secure and efficient approach to deep learning inference for dependable IoT systems, to overcome the two challenges. (1) The essential component of our methodology is a pixel drop operation on the IoT ends, which randomly selects and drops a certain amount of pixels of the input images. Such operation can reduce the data throughput between the IoT device and server to achieve higher network efficiency. At the same time, it also gets a high chance to invalidate the effects of AEs since it could drop the added perturbations. (2) It is worth noted that the pixel drop operation can affect the model accuracy, especially for the clean samples, as it removes certain information which can be critical for model prediction. To maintain high performance, we adopt a novel pixel reconstruction algorithm, sparse signals recovery, on the model server to recover the dropped pixels. (3) To further enhance the performance of the model on adversarial as well as clean samples, we integrate a wavelet-based denoising operation on the model server to remove the adversarial perturbations and artificial noises.

We conducted extensive evaluations to demonstrate the effectiveness of SUPER-IOT. For security, we measured the defense effects of our solution against 6 state-of-the-art adversarial attacks. We also compared SUPER-IOT with three existing defense methods (Shield [12], Pixel Deflection [10], Feature Distillation [11]): SUPER-IOT can maintain higher model accuracy and lower attack success rate than most defenses. For network efficiency, we measured the size of bitstreams with our preprocessing approach. SUPER-IOT can effectively reduce as high as 25% transmission throughput, while past works can hardly optimize network efficiency.

The major contributions of this paper include: (1) a pixel drop operation to reduce the network throughput and mitigate adversarial examples; (2) a pixel reconstruction algorithm to recover the original input and maintain high model accuracy; (3) a wavelet-based denoising operation to further remove the adversarial perturbations and artificial noises.

This paper is organized as follows. Section II discusses the research background and related works. Section III presents the problem definition and threat model. Section IV describes the design details of SUPER-IOT. Section V presents the evaluation results. We conclude in Section VI.

II. BACKGROUND AND RELATED WORKS

In this section, we briefly present the background and relevant works about AIoT systems, adversarial attacks on DNN models, and preprocessing-based defensive strategies.

A. Artificial Intelligence of Things

Benefiting from the advance of DL technology, IoT systems are becoming more intelligent and multi-functional. A typical IoT network can consist of an enormous amount of IoT devices. They are connected via different communication technologies (e.g., Ethernet, Wi-Fi). They collect sensory data over time and transmits them to one or more centralized hosts. These hosts can be remote cloud servers, local gateways, or powerful edge devices. They run the DNN inference applications, interpret the received sensory data, and make control decisions. Such IoT configuration has been widely adopted in many scenarios, such as face authentication [14], vehicle detection [15], and remote monitoring [4].

The sensory data generated from the IoT devices exhibit some unique features. First, there can be a large quantity of connected IoT devices generating real-time data continuously. This leads to a huge volume of streaming data in the network. Second, various IoT devices can collect different types of sensory data and information, resulting in data heterogeneity. Those data need to be transmitted to the DNN inference engine and processed promptly to extract immediate insights and make fast decisions. These requirements need to be achieved from different perspectives. (1) At the host level, we can utilize powerful cloud servers with high computing capability and execution parallelism, or specialized hardware circuits [16] to accelerate the DNN inference. (2) At the DNN algorithm level, novel algorithms were proposed (e.g., OS-ELM [17], Faster R-CNN [18]) to handle the data streaming for object detection and video analytics. (3) At the network level, one possible method is to preprocess and compress the sensory data to reduce network throughput and communication costs. This is also what we aim to optimize.

B. Adversarial Attacks on DNN models

An adversary can add human-unnoticeable perturbations on the original input to fool a DNN classifier. Formally, as in Eq. 1, the target DNN model is a mapping function F. Given a clean input sample x, the corresponding AE is denoted as $\widetilde{x} = x + \delta$ where δ is the adversarial perturbation. δ is constrained by certain metric (e.g. L_p norm) to make it imperceptible. Then AE generation can be formulated as the optimization problem in Eq. 1a (targeted attack where $l' \neq F(x)$ is the desired label set by the attacker, e.g. a cat image is mis-classified specifically as a dog) or Eq. 1b (untargeted attack, e.g. a cat image is misclassified as an arbitrary class other than a cat.). In this paper, we only evaluate the defense against the targeted attack and the untargeted attack can be mitigated in the same way.

$$min\|\delta\|, s.t.\ F(\widetilde{x}) = l' \tag{1a}$$

$$min\|\delta\|, s.t.\ F(\widetilde{x}) \neq F(x) \tag{1b}$$

Various approaches were proposed to solve the optimization problem and generate AEs. Fast Gradient Sign Method (FGSM) [6] calculates the sign of the gradient of the classification loss with respect to the input sample, which gives the direction to modify input pixel values under $L_{\inf}$ constraints to generate AEs. Later on, variations of FGSM were introduced to iteratively calculate the perturbations with a small step or with momentum, e.g., I-FGSM [7]. Some approaches use a more advanced optimization algorithm to find the minimal adversarial perturbation under L_2 constraint, like LBFGS [19], DeepFool [20], and Carlini & Wagner (CW) [21].

Attack scenarios. Generally, there are three attack scenarios [22], determined by the adversary's knowledge level of the target DNN model. (1) *White-box scenario*: the adversary knows every detail about the model including all the parameters. He can directly adopt the above approaches to generate AEs. (2) *Black-box scenario*: the adversary does not have any knowledge about the target model. He has to use an alternative model of the same task to generate AEs and attack the target one. (3) *Gray-box scenario*: the adversary knows all details of the model (e.g., training algorithms, network topology, hyper-parameters) except the parameters. He can train another similar model with the same configurations for AE generation. The transferability property of AEs [23] can guarantee high success rates for black-box and gray-box scenarios.

C. Preprocessing-based Defenses against AEs

Various defensive strategies have been designed to defeat adversarial attacks. One direction is to train a more robust model from either scratch or an existing model. Those approaches aim to rectify AEs' malicious features by including AEs into the training set [24], processing all the training data [25], or revising the DNN topology [26]. However, training a DNN model is very time and resource-consuming, especially when the model is complicated. Besides, those methods are not applicable when the DNN models are packed as closed-source applications and cannot be modified. Most of all, those methods are not secure: the adversary can still generate adaptive AEs for the new models [27].

A more promising direction is to preprocess the input data to eliminate adversarial influence without touching the DNN model. Typical transformation methods include denoising, compression, drop pixels, etc. These solutions are suitable in AIoT systems, as it is feasible and efficient to preprocess the sensory data on IoT devices. So here, we focus on this preprocessing-based direction. Below we describe some existing works and their limitations. We introduce our novel preprocessing technique in Section IV, and empirically demonstrate its advantages over those works in Section V.

Shield [12]. In this approach, JPEG compression is improved by randomizing the quantization factors to different blocks of image contents. Then the compression process consists of the Discrete Cosine Transform (DCT) and lossy quantization. This non-differentiable and irreversible transformation can obfuscate the gradients of the DNN model with respect to inputs from the adversary. However, this method can also decrease the classification accuracy of clean samples.

Feature Distillation (FD) [11]. This approach uses a revised JPEG compression based mechanism to defeat AEs. The quantization step in the DCT process is modified to optimize the reduction of the adversarial perturbations to improve the robustness of the DNN model. However, FD is inefficient as this revised quantization step can reduce the compression ratio.

Pixel Deflection (PD) [10]. The idea of this approach is to combine the denoising algorithm with the operation of dropping pixels. First, around 0.1% pixels of the input image is dropped and replaced with a random pixel value within a small range. Then, the denoising technique is applied to reduce the adversarial perturbations. PD could provide robustness against the adversarial examples. But the compression ratio (i.e., dropped pixels) has to be very small in order to maintain the model's prediction accuracy.

III. PROBLEM DEFINITION AND THREAT MODEL

In this paper, we aim to design a novel methodology for secure and efficient DNN inference for AIoT systems. Specifically, we consider a distributed IoT system conducting computer vision tasks (e.g., image classification, object detection). The sensor devices in the system keep collecting the visual input at high sampling rates and sending them to a centralized server for DNN inference. We focus on computer vision applications for two reasons. First, vision sensors (e.g., 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 [15], face authentication [14], autonomous driving [28], etc. Second, compared with other sensory data and tasks, vision sensors can produce a larger volume of real-time streaming data with higher throughput. So it is in a more urgent need for an efficient inference solution in an IoT system.

We assume that the DNN model deployed in the IoT system cannot be modified. In reality, the IoT administrator can purchase the DNN model from a model vendor. He may not be allowed to customize the model due to intellectual property protection. He may not be able to alter the model either if it is packed as a closed-source application. Then past approaches to training or retraining models for better robustness cannot be applied in our case. Designing new DNN hardware accelerators for better performance is out of the scope of this paper, as it requires drastic changes to the underlying infrastructure with high cost. The administrator can only implement preprocessing functions on the IoT devices or the server. Those functions must meet the following requirements.

- *Efficiency*: they must be able to reduce the throughput of the transmitted data from the IoT devices to the DNN server to relieve the burden and stress of the network bandwidth.
- *Lightweight*: the preprocessing function on the IoT devices should not be too heavy to impact the devices' performance or operations, considering the limited onboard computing capabilities and resources.
- *Functionality-preserving*: they should not affect the prediction accuracy of the DNN model on clean data samples.
- *Security*: they should be able to effectively remove the adversarial perturbations and preserve the correct prediction results from the model.

Threat Model. We assume the entire IoT system (e.g., sensor devices, central servers, and communication channel) is trusted. So we do not consider the security threats from IoT botnet (e.g., Mirai [29], Hajime [30]) and man-in-the-middle attacks. We also assume the target DNN model is correct without DNN backdoors [31]. The adversary is outside of the IoT system, attempting to spoof the sensors and DNN model by adding malicious perturbations on the physical objects or spots on the lens of the cameras [**?**]. He has white-box access to the DNN model and the preprocessing functions. We aim to show that even the adversary knows all detail of the target model and the defense mechanism, he can still not generate AEs to bypass the defense to compromise the model.

IV. PROPOSED METHODOLOGY

In this section, we present our efficient and secure approach for DNN inference in IoT systems. We give the methodology overview in Section IV-A, following by the descriptions of each operation in Sections IV-B, IV-C and IV-D.

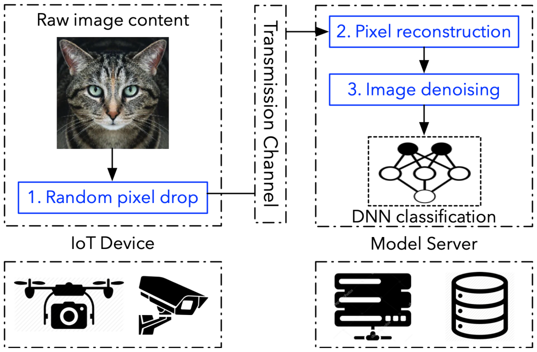

Figure 2. Methodology overview.

A. Design Overview

Figure 2 shows an overview of our proposed methodology. It consists of three steps across the IoT device and model server. The first step is conducted on the IoT device, which randomly drops some pixels from the input image. This operation can remove the potential adversarial perturbations with a high chance if the drop rate is high. Meanwhile, it can also increase the transmission efficiency between the IoT device and the model server as the image size is reduced after dropping certain pixels. This operation is lightweight and incurs very little computing costs on the IoT device. After this operation, the image will be sent out to the model server.

On the server, the received image cannot be directly fed into the DNN model, as a lot of information has been removed. Then, the second step is to reconstruct the dropped pixels. This operation can approximately recover the dropped pixels other than the perturbations. It will increase the model's prediction accuracy on this image.

The last step is image denoising. This operation can remove the malicious perturbations that are not dropped out at the IoT side, and also the artificial noises introduced during the reconstruction process. After that, the image can be sent to the DNN model for classification. Below we detail the mechanism and algorithm of each step.

B. Step 1: Pixel Dropping

In this step, the IoT device randomly selects a fixed ratio r of pixels and remove them out of the image. This can reduce the transmission throughput, and also remove adversarial perturbations. Specifically, we first divide the raw image into multiple blocks of $N \times N$ pixels. We denote one block as $f_0(x, y)$, where (x, y) represents the coordinate of pixels. Then, we randomly select $n = r \times N \times N$ pixel inside each block and set their values as zero. As shown in Eq. 2, n pixels $\{(x_1, y_1), ..., (x_n, y_n)\}$ are set to zero, while the rest remains the same as $f_0(x, y)$. The resulting block is denoted as $f_1(x, y)$. Finally, we concatenate the new blocks into one output image and send it to the model server.

$$f_1(x,y) = \begin{cases} 0, & \text{if } (x,y) \in \{(x_1,y_1),...,(x_n,y_n)\} \\ f_0(x,y), & \text{otherwise} \end{cases} \tag{2}$$

Note that the value of r can determine the efficiency, security, and also the model performance: a large r can reduce more throughput and decrease the success rate of adversarial perturbations. However, it can also decrease the model performance on clean samples. So we must carefully select r to balance such a trade-off. Figure 3 (first row) shows the output images with different drop ratios. We will empirically identify the optimal value in Section V.

C. Step 2: Pixel Reconstruction

When receiving the compressed image, the model server adopts the pixel reconstruction algorithm inspired by sparse signals recovery [32]. The algorithm is processed block by block using 2D-DCT transform (Eq. 3).

$$\begin{aligned} F(u,v) &= \frac{2}{N}\sum_{i=1}^{N}\sum_{j=0}^{N}\alpha_{i,j}(u,v)f(i,j) \\ \alpha_{i,j}(u,v) &= \Lambda(i)\Lambda(j)\cos[\frac{\pi u}{2N}(2i-1)]\cos[\frac{\pi v}{2N}(2j-1)] \\ \Lambda(x) &= \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } x = 0 \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

The reconstruction algorithm is detailed in Algorithm 1: given a block f_1, for each dropped pixel at position (k, l) ($(k, l) \in \{(x, y)|f_1(x, y) = 0\}$), we estimate a gradient to modify its pixel value. We first perturb the pixel value in two directions with a distortion level Δ (lines 4-5). Then, we calculate their 2D-DCT transform and L_1 norm respectively (lines 6-9). The gradient is calculated (line 10) and used to update the pixel in block f_1 with a step size of μ (line 11). During this iterative process, we keep monitoring the changes of reconstructed images using the metric Mean Square Error (MSE), which is defined as the difference of output images in two consecutive iterations (line 13). When MSE is hardly changed, we dynamically reduce Δ and μ to achieve better reconstruction results (lines 14-18). This iterative process will end until MSE is smaller than a threshold ϵ (line 21).

Figure 4 shows the trends of MSE, PSNR, and $[\Delta, \mu]$ during the image reconstruction process. Initially, MSE keeps decreasing while PSNR keeps increasing. When MSE becomes too small, the reconstruction tends to converge and stops the

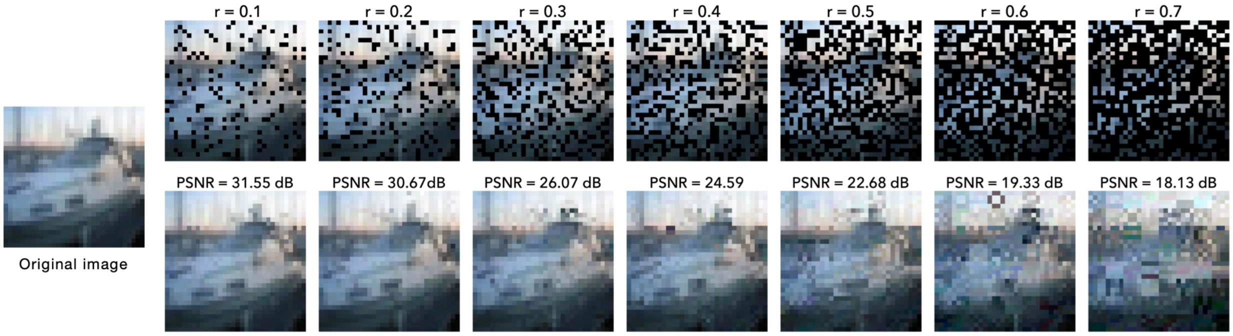

Figure 3. Visual content evaluation of pixel drop with different ratio (r from 0.1 to 0.7) and the reconstruction results measured by PSNR.

ALGORITHM 1: Pixel Reconstruction

Input: a image block $f_1(x, y)$
Output: the reconstructed image block $f_2(x, y)$
Parameters: Δ distortion level; μ step size; ϵ stop criterion.

```
/* Initialization */
$d = 0, f^0(x, y) = f_1(x, y), MSE_{max} = 0$;
do
    for $(k, l)$ in $\{(x, y)|f_1(x, y) = 0\}$ do
        /* Perturb pixel value of missing pixel; $\delta$ Dirac function */
        $f_+^{(k,l)}(x, y) = f_1(x, y) + \Delta\delta(x - k, y - l)$;
        $f_-^{(k,l)}(x, y) = f_1(x, y) - \Delta\delta(x - k, y - l)$;
        /* 2D-DCT transform */
        $F_+^{(k,l)}(u, v) = \frac{2}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{i,j}(u, v)f_+^{(k,l)}(i, j)$;
        $F_-^{(k,l)}(u, v) = \frac{2}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{i,j}(u, v)f_-^{(k,l)}(i, j)$;
        /* $L_1$ norm */
        $\|F_+^{(k,l)}\|_1 = \sum_{u=1}^{N}\sum_{v=1}^{N}\|F_+^{(k,l)}(u, v)\|_1$;
        $\|F_-^{(k,l)}\|_1 = \sum_{u=1}^{N}\sum_{v=1}^{N}\|F_-^{(k,l)}(u, v)\|_1$;
        /* Estimate gradient */
        $grad(k, l) = \frac{\|F_+^{(k,l)}\|_1 - \|F_-^{(k,l)}\|_1}{2\Delta}$;
        /* Update pixel value */
        $f^{d+1}(k, l) = f^d(k, l) - \mu \times grad(k, l)$;
    end
    /* Dynamically update $\Delta$, $\mu$ */
    $MSE = \|f^{d+1}(x, y) - f^d(x, y)\|_2$;
    if $MSE < 0.01 \times MSE_{max}$ then
        $\Delta = \Delta/10$;
        $\mu = \mu/10$;
        $MSE_{max} = 0$;
    end
    $MSE_{max} = max(MSE, MSE_{max})$;
    $d = d + 1$;
while $MSE > \epsilon$;
$f_2(x, y) = f^d(x, y)$;
return $f_2(x, y)$
```

modification of pixel values. To refine the reconstruction, when MSE is smaller than 1% of the maximal MSE previously, Δ and μ is updated dynamically (on the 13th and 17th iterations). At the 20th iteration, MSE already becomes very small and the reconstructed image has a good quality with PSNR bigger than 30. Then, the iterative process stops.

The hyper-parameters used in this pixel reconstruction algorithm can significantly affect the difficulty of the reconstruction process and the quality of the output. So we need to discover the optimal values.

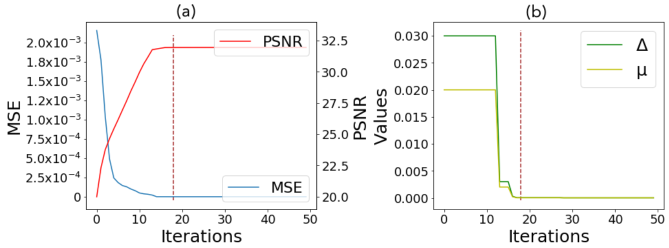

Figure 4. Dynamic process of image reconstruction. (a) (MSE and PSNR) versus iterations (b) $[\Delta, \mu]$ versus iterations.

First, for the initial values of distortion level Δ and step size μ, we tried different values within the range $[0.01, 0.1]$ and measured the quality of reconstructed images using the metric Peak Signal to Noise Ratio (PSNR), which is defined as the visual content deviation from the clean image. Figure 5 shows the average PSNR for each configuration. We can choose $\Delta = 0.03$ and $\mu = 0.02$ that lead to the best image quality.

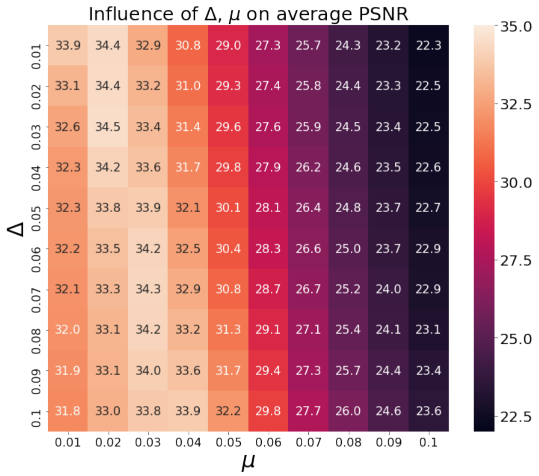

Figure 5. PSNR of reconstructed images under different values of Δ and μ.

Second, the stop criterion ϵ determines the number of iterations during image reconstruction. As shown in Figure 4, the MSE, and PSNR will become saturated after a certain number of rounds. Then, it is not necessary to continue the iteration, as the quality of the reconstructed image will not change. So an appropriate threshold ϵ can guarantee the best quality of output with the minimal number of iterations. We empirically identify the optimal $\epsilon = 10^{-5}$ from Figure 4.

Third, the pixel drop ratio r can also impact the image reconstruction. Figure 3 (second row) shows the reconstructed images and their PSNRs with different drop ratios. We observe that larger r leads to smaller PSNR (i.e., worse quality). Besides, the value of r can also determine the effects of adversarial examples, and network throughput. More evaluation results will be presented in Section V to show the trade-off between those aspects, and discover the ideal drop ratio.

D. Step 3: Image Denoising

After the pixel reconstruction, the model server uses the image denoising algorithm to further improve the image quality. On one hand, the pixel dropping and reconstruction can introduce artificial noises. Then this denoising operation can filter such new introduced noises [10]. On the other hand, this denoising operation is non-differentiable. It can obfuscate the DNN model gradients to further increase the difficulty of AE generations via gradient-based approaches.

In this paper, we adopted the wavelet-based denoising method named BayesShrink [33] from [10]. Other denoising methods can be applied in a similar way. The denoising method is performed in the frequency domain through the wavelet transform. The image noise is always assumed as the small perturbations on values in the high-frequency domain. Therefore, these small values can be removed by setting coefficients below a given threshold to zero (hard threshold) or shrinking different coefficients toward zero by a soft threshold. Firstly, we use the VisuShrink approach to set a hard threshold. For an image X with N pixels, this threshold is given by $\sigma\sqrt{2\log N}$, where σ is normally smaller than the true noise standard deviation. Then, we adopted the BayesShrink algorithm [10] as an additional step to set a soft threshold to further filter the wavelet coefficients. The threshold $T_h * (\sigma_x, \beta)$ is estimated on each wavelet sub-band and the optimal threshold is calculated by minimizing the expected mean square error. We model the threshold for each wavelet coefficient as a Generalized Gaussian Distribution (GGD). It can be approximated as $\frac{\sigma^2}{\sigma_x}$ where σ_x and β are parameters of the GGD for each wavelet sub-band (Eq. 4).

$$T_h * (\sigma_x, \beta) = \underset{T_h}{\operatorname{argmin}}\, E(\widehat{X} - X)^2 \approx \frac{\sigma^2}{\sigma_x} \tag{4}$$

Normally, an approximation of T_h, as shown on the right side of Eq. 4, is used to adapts to the amount of noise in the given image. The parameters for the denoising in this paper are tuned to get the best performance.

V. EVALUATIONS

In this section, we comprehensively evaluate the efficiency and security of our proposed methodology. We measure its resilience against six popular adversarial attacks and compare it with three existing preprocessing-based defense methods. We also measure and compare the network throughput benefits introduced by different approaches.

A. Experimental Configuration

We consider an image classification task on the CIFAR-10 dataset. There are 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 choose ResNet-29 [34] as the target model. It consists of 29 layers for three bottleneck residual blocks with channel sizes of 64, 128, and 256, respectively. We use Keras package with Tensorflow 1.14 [35] backend to implement the model. Weights in all convolutional layers are initialized by a truncated normal distribution proposed in [36]. The training process is done via the Adam optimization algorithm [37] with its hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999$. The model is trained to reach the top-1 accuracy of 92.27% over the testing set after about 150 epochs. Experiments are conducted on a server with a CPU of Intel(R) Core(TM) i9-9900K@3.60GHz and a GPU of NVIDIA GeForce GTX 2080 Ti.

Hyper-parameters. For the pixel drop and reconstruction algorithms, we set $N = 8$ to have blocks with 8×8 pixels, which is a typical configuration of DCT transform applied in image compression. We set $\Delta = 0.03$, $\mu = 0.02$ and $\epsilon = 10^{-5}$, as discussed in Section IV-C.

B. Security Evaluation

First, we check whether our methodology can defeat existing adversarial attacks. We consider six well-known attack techniques: FGSM [19], I-FGSM [7], DeepFool [20], LBFGS [6], CW [21], and PGD [38]. We adopt the CleverHans library (v3.0.1) [39] to generate AEs with those approaches. We set the $|L_2|$ between AEs and original images to be within 0.5, to make the perturbation imperceptible. For FGSM and I-FGSM, the scale of distortion is $\epsilon = 0.005$ under the $L_{\inf}$ constraints. For PGD, the scale of distortion is $\epsilon = 0.01$ and the number of attack iterations is 10. For the rest of the attacks, the optimization process is iterated until the adversary generates AEs of all samples. For all evaluations, we consider the targeted attack, where a random label different from the correct one is selected as the adversary's target. The AEs are generated under a white-box scenario.

Table I shows the prediction accuracy of the clean samples as well as AEs when they are not preprocessed (Baseline column), or preprocessed by our methodology with different drop ratio r. All average accuracy is measured for 100 images. We observe that without any defense, all attacks can significantly compromise the performance of the target model, even making the accuracy drop to 0. With our preprocessing operation, the prediction accuracy is significantly increased. The accuracy of classifying AEs from FGSM, I-FGSM, and PGD will increase first and then decrease which is different from the DeepFool, LBFGS, and CW. This is due to the initial value $r = 0.01$ has been already effective to mitigate the DeepFool, LBFGS, and CW attacks. Even lower r will see the same accuracy trend of DeepFool, LBFGS, and CW compared with FGSM, I-FGSM, and PGD. Then, continuing increase r after 0.25 will lead to the decrease of model accuracy since and is a trade-off between classifying clean samples and mitigating AEs.

Table I
The Top-1 accuracy in the presence of various adversarial attacks on Super-IoT with different pixel drop ratios.

Attack	**Baseline**	r=0.01	r=0.05	r=0.10	r=0.15	r=0.20	r=0.25	r=0.30	r=0.40	r=0.50
Clean	1.00	**1.00**	0.99	0.99	0.93	0.94	0.88	0.86	0.60	0.43
FGSM	0.39	0.60	0.65	0.70	0.72	0.75	**0.81**	0.79	0.59	0.41
I-FGSM	0.21	0.45	0.54	0.65	0.78	0.79	**0.79**	0.71	0.64	0.42
PGD	0.05	0.30	0.39	0.51	0.67	0.69	**0.72**	0.72	0.56	0.51
DeepFool	0.00	**0.99**	0.98	0.97	0.92	0.89	0.85	0.83	0.60	0.44
LBFGS	0.00	0.95	**0.99**	0.97	0.92	0.89	0.85	0.82	0.59	0.37
CW	0.00	**0.98**	0.97	0.96	0.93	0.89	0.88	0.80	0.62	0.43

Next, we compare Super-IoT with existing state-of-the-art solutions: Shield [12], Pixel Deflection (PD) [10], and Feature Distillation (FD) [11]. Those preprocessing-based solutions do not require the modification of DNN models and can be applied to our IoT scenario. Due to the stochastic features in these solutions, we repeat the experiments 10 times and report the average prediction accuracy of each preprocessing method for each attack technique, shown in Table II (We set $r = 0.1$ in our method). We observe that our solution can beat the other methods on the performance of AEs from DeepFool, LBFGS, and CW. For AEs from FGSM, I-FGSM, and PGD, the accuracy is higher than PD but slightly lower than Shield and FD. However, Shield and FD have bad performance on the clean samples, making them less practical.

Table II
The Top-1 accuracy in the presence of various adversarial attacks on baseline model, Shield, FD, PD, and Super-IoT.

Attack	L_{inf}	L_2	**Baseline**	Shield	FD	PD	SUPER
Clean	0.0	0.0	1.00	0.85	0.94	0.98	**0.99**
FGSM	0.005	0.28	0.39	0.78	**0.85**	0.57	0.70
I-FGSM	0.005	0.21	0.21	0.79	**0.83**	0.44	0.65
PGD	0.010	0.39	0.05	**0.79**	0.70	0.29	0.51
DeepFool	0.015	0.12	0.00	0.82	0.91	0.87	**0.97**
LBFGS	0.017	0.15	0.00	0.82	0.92	0.97	**0.97**
CW	0.115	0.09	0.00	0.82	0.93	0.94	**0.96**

C. *Efficiency Analysis*

We measure the efficiency of Super-IoT in terms of reduced transmission throughput. By dropping certain pixels, Super-IoT can effectively reduce the total bitstream for transmission. This can save the energy cost of the IoT devices, and relieve the stress of network bandwidth in IoT systems.

Figure 6 shows the compression ratio evaluated of Super-IoT with different pixel drop ratio ranging from 0.01 to 0.5 (the bars). The evaluation here is made based on the bitstream size before feeding into the compression algorithms. We can see a larger r leads to a larger compression ratio for better efficiency. This is straightforward: when r of the pixels are dropped out, the size of transmitted data will also be reduced by r. However, as we discussed in Section V-B, a larger r can also affect the prediction accuracy of clean samples and AEs (curves in Figure 6). So users need to carefully balance the trade-off between efficiency, security, and functionality, and consider their requirements when configuring the drop ratio.

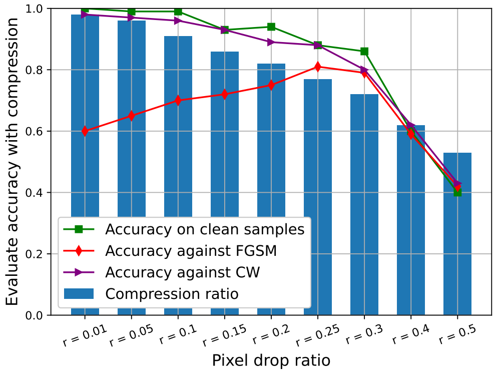

Figure 6. Evaluation of Super-IoT on the DNN model classification accuracy (on both clean sample and AEs generated by two different approaches) and compression ratio with different pixel drop ratio.

In contrast, past works on AE defenses cannot achieve throughput reduction. For PD, the optimal pixel drop ratio is between 0.1% to 1% in order to have a good model performance for both adversarial and clean images. This ratio has very little improvement in transmission efficiency. For Shield and FD, there are no data reduction effects at the bitstream level in their preprocessing operations. Thus, considering the transmission efficiency, Super-IoT has an obvious advantage over PD, Shield, and FD.

For the computation overhead, both PD and our Super-IoT only require pixel drop operation on the IoT devices which will add very limited additional computing overhead. For comparison, Shield and FD have operations of modifying the JPEG compression process with sophisticated randomization or dedicated quantization tables, which are much heavier than the pixel drop operation for IoT devices.

We adopt state-of-the-art configuration, where IoT collects sensory data, and sends them to the server for inference. The only computation cost introduced is the image reconstruction at the server end, which is lightweight. However, we can save the network throughput and transmission power from the IoT end. Since IoT devices are more resource-constrained than the server, such optimization is meaningful.

VI. Future Work and Conclusion

For future work, we plan to explore better data drop and reconstruction methods for higher classification accuracy on both clean samples and AEs. With a larger drop ratio and better reconstruction methods, the network throughput can be further reduced. We will also explore more advanced defense solutions for AIoT systems against more advanced attacks like adaptive adversarial attacks.

In this paper, we proposed a novel approach, SUPER-IOT, to efficiently secure the inference of DNN models in AIoT systems. We employ three techniques (pixel dropping, pixel reconstruction, and image denoising) to defeat adversarial examples and maintain good performance for clean samples. Meanwhile, those operations can also achieve high efficiency for network transmission in the IoT systems by reducing the bitstream of transmitted sensory data. Our approach is lightweight with little impact on the IoT devices' performance or operations. It is generic and can be applied to various computer vision tasks without modifying the DNN models.

VII. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their helpful feedback. This project is supported by the National Research Foundation, Singapore, under its National Cybersecurity R&D Programme (CHFA-GC1-AW03). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 858–862.

[5] H. Qiu, Q. Zheng, G. Memmi, J. Lu, M. Qiu, and B. Thuraisingham, "Deep residual learning based enhanced JPEG compression in the internet of things," *IEEE Transactions on Industrial Informatics*, 2020.

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

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[8] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 513–530.

[9] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2267–2281.

[10] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Deflecting adversarial attacks with pixel deflection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8571–8580.

[11] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen, "Feature distillation: DNN-oriented JPEG compression against adversarial examples," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 860–868.

[12] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, "Shield: Fast, practical defense and vaccination for deep learning using jpeg compression," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 196–204.

[13] A. Athalye and N. Carlini, "On the robustness of the CVPR 2018 white-box adversarial example defenses," *arXiv preprint arXiv:1804.03286*, 2018.

[14] 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)*, 2018, pp. 1–6.

[15] Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system," *Multimedia tools and applications*, vol. 76, no. 4, pp. 5817–5832, 2017.

[16] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[17] Z. Yang, P. Zhang, and L. Chen, "RFID-enabled indoor positioning method for a real-time manufacturing execution system using OS-ELM," *Neurocomputing*, vol. 174, pp. 121–133, 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[22] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 135–147.

[23] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[24] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[25] Y. Yang, G. Zhang, D. Katabi, and Z. Xu, "Me-net: Towards effective adversarial robustness with matrix estimation," in *International Conference on Machine Learning (ICML)*, 2019.

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

[27] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[28] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[29] 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.

[30] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of Hajime, a peer-to-peer IoT botnet." in *NDSS*, 2019.

[31] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[32] L. Stanković, M. Daković, and S. Vujović, "Adaptive variable step algorithm for missing samples recovery in sparse signals," *IET Signal Processing*, vol. 8, no. 3, pp. 246–256, 2014.

[33] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE transactions on image processing*, vol. 9, no. 9, pp. 1532–1546, 2000.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[35] N. Ketkar, "Introduction to keras," in *Deep Learning with Python*. Springer, 2017, pp. 97–111.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[39] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, and R. Long, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv preprint arXiv:1610.00768*, 2018.