Consensus Adversarial Defense Method Based on Augmented Examples

Xintao Ding, Yongqiang Cheng, Yonglong Luo, Qingde Li, and Prosanta Gope, *Senior Member, IEEE*

***Abstract*— Deep learning has been used in many computer-vision-based industrial Internet of Things applications. However, deep neural networks are vulnerable to adversarial examples that have been crafted specifically to fool a system while being imperceptible to humans. In this study, we propose a consensus defense (Cons-Def) method to defend against adversarial attacks. Cons-Def implements classification and detection based on the consensus of the classifications of the augmented examples, which are generated based on an individually implemented intensity exchange on the red, green, and blue components of the input image. We train a convolutional neural network using augmented examples together with their original examples. For the test image to be assigned to a specific class, the class occurrence of the classifications on its augmented images should be the maximum and reach a defined threshold. Otherwise, it is detected as an adversarial example. The comparison experiments are implemented on MNIST, CIFAR-10, and ImageNet. The average defense success rate (DSR) against white-box attacks on the test sets of the three datasets is 80.3%. The average DSR against black-box attacks on CIFAR-10 is 91.4%. The average classification accuracies of Cons-Def on benign examples of the three datasets are 98.0%, 78.3%, and 66.1%. The experimental results show that Cons-Def shows a high classification performance on benign examples and is robust against white-box and black-box adversarial attacks.**[1]

***Index Terms*— Adversarial defense, consensus defense, data augmentation, industrial Internet of Things.**

I. Introduction

Convolutional neural networks (CNNs) have achieved state-of-the-art results in numerous computer vision tasks [1] and have been involved in many industrial Internet of Things (IIoT) topics, such as mobile target tracking [2], intrusion detection [3], and edge computing [4]. However, CNN models are vulnerable to adversarial examples that are usually crafted by injecting small perturbations into benign examples [1], [5]. Although small perturbations are imperceptible to humans, they can fool CNN models and pose a serious threat to critical security applications [5]. Recently, several studies have focused on security topics in IIoT [2], [6], and some studies have crafted adversarial examples to attack IIoT systems [7]. Adversarial defense is a crucial concern in CNN applications.

Several adversarial attack approaches have been designed to fool CNN models in the field of image classification and recognition. Adversarial attacks can be launched either in the digital domain [8]–[14] or in the physical domain [15]. Digital attacks can be launched from four bases: (1) gradient-based attacks, such as fast gradient sign method (FGSM) [8], projected gradient descent (PGD) [9], and DeepFool [10]; (2) optimization search-based attacks, such as Carlini and Wagner (C&W) attacks [11] and Jacobian-based saliency map attacks (JSMAs) [12]; (3) network-based attacks [13]; and, (4) randomness-based attacks [14]. Although digital attacks are complicated, many of them involve gradients. C&W attack employs gradients for optimization. The training of network-based attacks usually depends on the backpropagation of the gradient [13]. For randomness-based attacks, gradients can also be employed to design adversarial attacks [14].

Since adversarial attacks typically craft adversarial examples based on gradients, we intend to design a gradient-based defense method that efficiently utilizes attack results. The gradients are embedded in the images. Since adversarial attacks are inevitable, we do not aim to prevent them but induce them to produce contradictory results. Motivated by this idea, the following three conditions should be addressed. First, we can expand an input image to multiple images with varying gradients. Second, the varying gradients can induce different classifications. Finally, if the second condition is addressed, how do we use heterogeneous classifications on augmented examples to defend against adversarial attacks?

Hence, we propose a consensus defense (Cons-Def) method to address these three conditions in this study. Cons-Def contains two modules: augmentation training and consensus testing. First, we augment the training set to train a CNN model in which every image generates multiple augmented images based on intensity exchange. Subsequently, we imple-

Manuscript received September 5, 2021; revised December 21, 2021 and March 15, 2022; accepted April 15, 2022. This work was supported in part by the Anhui Provincial Natural Science Foundation (1808085MF171), in part by the National Natural Science Foundation of China (61972439, 61976006), and in part by the Designing an all-in-one Vital Sign Point of Care Analysis Tool project funded by the Innovative UK (KTP/12053). (Corresponding author: Yongqiang Cheng; Yonglong Luo.)

X. Ding and Y. Luo are with the School of Computer and Information, Anhui Normal University, Wuhu 241002, China, and also with Anhui Province Key Laboratory of Network and Information Security, Wuhu 241002, China (E-mail: xintaoding@163.com; ylluo@ustc.edu.cn).

Q. Li and Y. Cheng are with Department of Computer Science and Technology, University of Hull, Hull HU67RX, UK (E-mail: Q.Li@hull.ac.uk; Y.Cheng@hull.ac.uk).

P. Gope is with Department of Computer Science, University of Sheffield, Regent Court, Sheffield S1 4DP, UK. (E-mail: prosanta.nitdgp@gmail.com)

[1]The source code for this work is publicly [Online]. Available: https://github.com/xintaoding/Cons-Def

ment consensus decision-making on a group of augmented test examples to defend against adversarial attacks, that is, the test image is classified into a class supported by the classification of the supermajority if the number of supports is not less than a threshold. Otherwise, it is determined to be an adversarial example. Fig. 1 illustrates the defense mechanism used in this study. Fig. 1(a) shows a benign example. Figs. 1(b) and 1(c) show corresponding FGSM perturbation and adversarial example, respectively. The red module in Fig. 1 shows the test results for Cons-Def. The blue module shows FGSM attacks on the augmented examples. To show the details, the perturbations in Fig. 1 are translated to be nonnegative and then magnified by 10. The augmented perturbations in Fig. 1(d) vary to FGSM perturbations in Fig. 1(e). Since FGSM perturbations are aggressive to the CNN model, augmented perturbations may be safe for the model. Otherwise, Figs. 1(d) and 1(e) should be similar to a certain extent. Although a malicious attacker usually fools an artificial intelligence system by submitting adversarial examples, Cons-Def always expands every received example to multiple augmented examples that have varying gradients, which favors immunization against adversarial attacks.

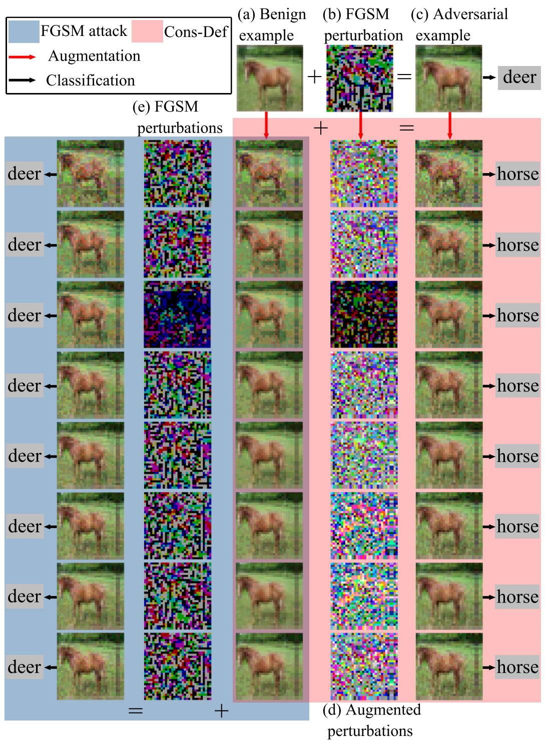


Fig. 1. Attack and defense examples. (a) Benign example. (b) FGSM perturbation. (c) Adversarial example. (d) Augmented perturbations corresponding to (b). (e) FGSM perturbations on the augmented benign examples. The perturbations are translated to be nonnegative and magnified by 10.

The main contributions of this study are summarized as follows:

(1) We propose a consensus decision-making method based on a group of augmented examples to defend against adversarial attacks. We expand the input image to augmented images with varying gradients. Augmented perturbations are usually different from adversarial perturbations; therefore, CNN models cannot easily be fooled on all the augmented examples.

(2) We propose a strategy to implement adversarial classification and detection simultaneously. Classification and detection are two popular adversarial defense tasks. Based on the literature, defense studies usually aim for adversarial classification or detection. Herein, we implement adversarial classification and detection simultaneously to improve defense performance.

The remainder of this paper is organized as follows. Related works are presented in Section II. In Section III, the preliminaries and our proposed framework are introduced. Cons-Def algorithms are presented in Section IV. The comparison experiments are presented in Section V. Finally, the conclusions are presented in Section VI.

II. Related Work

A. Adversarial Attacks

Many adversarial attack techniques have been developed to fool CNNs for computer vision tasks. Generally, adversarial attacks can be categorized as white-box or black-box attacks [1], [5], [14]. In the white-box case, the attacker has comprehensive knowledge of the model and the training data. In a black-box attack, the attacker does not have knowledge of the model. Although attacks can be launched in the physical domain [15], we focus on attacks launched in the digital domain.

1) White-box Attacks: Several white-box attack methods have been established. Since we focus on adversarial defense, the following widely used attacks are employed for testing in this study.

FGSM [8]: FGSM is a classical gradient-based attack. The FGSM generates adversarial examples based on the gradient of the loss function with respect to the input image. FGSM inversely changes the intensities of the pixels in the input image to achieve its purpose. Some pixels with low intensities are perturbed with positive perturbations. Meanwhile, some pixels with high intensities are perturbed by negative perturbations.

C&W attack [11]: Instead of leveraging training loss, Carlini and Wagner designed a loss function and optimized it to craft adversarial examples. C&W attacks are widely regarded as one of the strongest attacks and are usually employed in the defense literature for comparison.

JSMA [12]: JSMA uses an adversarial saliency map to find the input pixels with the greatest impact on the specific output of the target model. It searches several critical pixels with large weights using loop technology. JSMA is usually much slower than the FGSM and C&W attacks.

PGD [9]: PGD iteratively applies FGSM multiple times with a small step size and can be considered an extension of FGSM. PGD attacks are typically much stronger than FGSM attacks.

DeepFool [10]: DeepFool iterates a gradient-based increment to obtain adversarial examples. The adversarial example is linearly iterated by the initial input image.

2) Black-box Attacks: Within the scope of adversarial attacks, black-box attacks are usually produced by transferability between architectures. Adversarial examples generated on one classifier can sometimes cause another classifier to produce misclassifications, even if the classifier has a different architecture or is trained on disjoint datasets [1], [5]. A black-box attack produces adversarial examples on a known classifier (a source model) and transfers them to a target classifier, where the source attack does not know the information of the target model.

B. Adversarial Defenses

Since adversarial attacks are a serious threat to security-critical applications, many studies have focused on adversarial defense. Most defenses are developed in four streams: (1) training-based defense [8], [16]–[18], (2) gradient-based defense [19], [20], (3) input and output-based defense [21]–[25], and (4) knowledge-based defense [26], [27].

1) Training-based Defenses: Goodfellow et al. developed adversarial training by injecting adversarial examples into the training set to enhance the robustness of the CNN model [8]. In [16], the authors proposed an adversarial logit pairing (ALP) method that encourages logits for pairs of examples to be similar. Some studies used learning-based methods to generate adversarial examples and design defense methods, such as defense with conditional generative adversarial networks (CGAN) [17]. By combining adversarial training in shallow layers and an attention weight-based model, Chen et al. proposed an adversarial defense method by refocusing on critical areas and strengthening object contours (RCA-SOC) [1]. Zhu et al. proposed a dual-domain-based adversarial defense (DD-AD) method based on a conditional variational autoencoder and Bayesian network [18]. In [5], the authors proposed a deeply supervised discriminative learning (DSDL) method to defend against adversarial attacks. In this study, we trained models based on augmented examples.

2) Gradient-based Defenses: Since many adversarial attacks are launched based on gradients, several methods defend against adversarial attacks based on gradients. Dabouei et al. proposed a joint gradient phase and magnitude regularization (GPMR) method to explore practical defense [19]. However, GPMR appears to be sensitive to the attack parameters. Anish et al. summarized defense based on obfuscated gradients into three types: shattered gradients, stochastic gradients, and exploding and vanishing gradients [20]. Defenses relying on obfuscated gradients focus on gradient masking, which causes attackers have no useful gradients [20]. Cons-Def launches adversarial defense using augmented images with varying gradients. Heterogeneous gradients are useful for Cons-Def.

3) Input and Output-based Defenses: Contrary to injecting adversarial perturbation for adversarial training, several studies apply image transformation, such as JPEG compression [21], PixelDefend [22], total variance minimization (TVM) [23], and the sparse transformation layer (STL) method [24], in adversarial defense. PixelDefend first purifies input images and then feeds them to the classifier for classification [22]. TVM is a compressed sensing approach that combines pixel dropout with variation minimization [23]. STL first projects input images into a quasi-natural image space and then feeds the projections to the networks [24]. Some studies implemented adversarial defense based on the output, such as feature squeezing (Feat-Squ) [25]. Feat-Squ detects an adversarial example by comparing its prediction on the original sample with that of the sample after squeezing [25]. In our design, the classifications on the augmented adversarial images are usually heterogeneous. We use the consensus of the classifications on the augmented examples to implement adversarial defense.

4) Knowledge-based Defenses: Many studies have implemented adversarial defense based on statistical knowledge. Defensive distillation extracts the knowledge of class probabilities to reduce the success rate of adversarial sample crafting [26]. Liu et al. proposed an enhanced spatial rich model to implement adversarial detection, in which steganalysis was applied to estimate the probability of modifications caused by adversarial attacks [27].

III. Preliminaries and Framework Overview

A. Preliminaries

Generally, CNN is successively made of several convolutional and pooling layers, followed by one or more fully connected (FC) layers and an output layer. Fig. 2 shows a CNN architecture that is suitable for classification tasks. In this study, we denote the CNN-based image classification model as $\mathcal{F}$ with parameters θ.

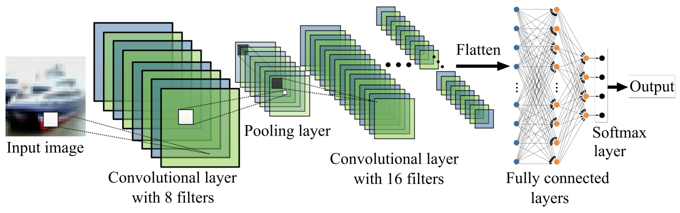


Fig. 2. Overview of a CNN architecture.

Adversarial attacks are typically generated by optimization [5], as shown in (1).

$$\arg\max_{\delta} \mathcal{L}\left(\mathcal{F}(\mathrm{x}+\delta,\theta),\mathcal{F}(\mathrm{x},\theta)\right), s.t.\|\delta\| \leq \varepsilon, \tag{1}$$

where δ is the perturbation of the input image x, $\varepsilon \geq 0$ is a given small constant, $\mathcal{L}$ is a proper loss function, and $\|\bullet\|$ is a norm operator. An attacker explores the adversarial sample $\mathrm{x}^{adv} = \mathrm{x}+\delta$ locally around x but can change the prediction of the classifier as much as possible.

Many defensive techniques against adversarial attacks have been proposed recently [8], [16]–[27]. Based on the evaluation metrics, they can be divided into two categories: one is classification-based defense, which aims to correctly classify adversarial examples. The other is a detection-based defense that aims to distinguish clean and adversarial examples.

Classification and detection defense are usually evaluated on adversarial examples. The classification accuracy (CA) and detection rate (DR) for the classification and detection defenses are shown in (2) and (3), respectively.

$$CA = \frac{n_{adv}(l_p = l_T)}{N_{adv}}, \tag{2}$$

$$DR = \frac{n_{adv}(d = 1)}{N_{adv}}, \tag{3}$$

where N_{adv} is the number of adversarial examples, $n_{adv}(l_p = l_T)$ is the number of adversarial examples correctly classified, and $n_{adv}(d = 1)$ is the number of adversarial examples correctly detected.

Since an input example may also be a benign example, the defense accuracy on benign examples is also evaluated in the literature. Some adversarial detection methods implement evaluations using true-positive rate and false-positive rate (FPR). Generally, FPR is reported on benign examples. The accuracy on benign examples denoted as acc is shown in (4).

$$acc = \frac{n_{ben}(l_p = l_T)}{N_{ben}}, \tag{4}$$

where N_{ben} is the number of benign examples, and $n_{ben}(l_p = l_T)$ is the number of benign examples correctly classified.

B. Framework Overview

The main task of our proposed Cons-Def method is to implement classification and detection based on the consensus of the classifications on the augmented examples. As shown in Fig. 3, the outline of our proposed Cons-Def method comprises two modules. The first is augmentation training, and second is consensus testing.

Let $S = \{\mathrm{x}_0, \mathrm{x}_1, \cdots, \mathrm{x}_{N-1}\}$ be a set composed of N training images, and its label set be $Y = \{y_0, y_1, \cdots, y_{N-1}\}$. Take an RGB image x as an example, let $\mathrm{x}^{(0)}$, $\mathrm{x}^{(1)}$, and $\mathrm{x}^{(2)}$ be the red (R), green (G), and blue (B) component images of x, respectively. The input image $\mathrm{x} \in S$ (Fig. 3(a1)) is first separated into three component images (Fig. 3(a2)). For every component image, we arrange its intensities from low to high and generate an intensity list. We then split every list into 2^k blocks, where $k = k_1, k_1 + 1, \cdots, k_1 + s - 1$, and obtain s intensity exchange lists, as shown by the intensity exchange module in Fig. 3(a3). Taking the split $k = k_1$ as an example, let the block length of the split be l_1. The i-th intensity in the first block is exchanged with the i-th intensity in the second block, $i = 1, 2, \cdots, l_1$. The intensities in the third block are exchanged with those in the fourth block in a similar manner. An intensity exchange list is obtained after all 2^{k_1} blocks are processed. We scan the original component image to generate an augmented component image using the obtained list. If the intensity of a pixel in the original component image is the j-th element in the original intensity list, the intensity of the corresponding pixel in the augmented component image is valued at the j-th element in the obtained exchange list. After all exchange lists are obtained, we generate s augmented component images for every component image (Fig. 3(a4)). The augmented components in Fig.3(a4) are fully connected through the R, G, and B channels to produce s^3 augmented RGB images (Fig. 3(a5)). As shown in Fig. 3(a6), our training set is composed of all augmented examples together with the original training examples. Fig. 3(a7) shows the trained model using a CNN with a structure similar to that shown in Fig. 2.

In the test module (Fig. 3(b)), the input (Fig. 3(b1)) is first padded and cropped (Fig. 3(b2)). The cropped image is then augmented using the aforementioned augmentation scheme (Fig. 3(b3)). Next, classifications are implemented on the s^3 augmented examples using the trained model (Fig. 3(b4)). We then build a histogram of the predicted labels (Fig. 3(b5)). Finally, defense results are obtained, as shown in Fig. 3(b6). If the vertical coordinate of the histogram peak is not less than a given threshold T_c, the input image is classified into the class at the peak. Otherwise, it is detected as an adversarial input.

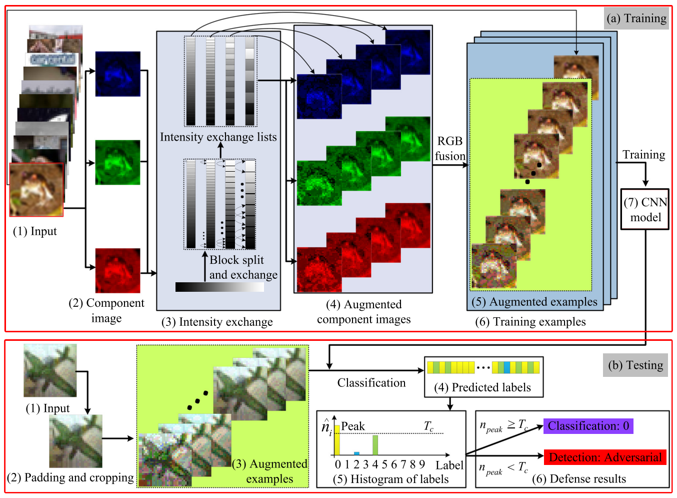


Fig. 3. Framework of our proposed Cons-Def method. (a) and (b) are training and testing modules, respectively.

IV. Defense Algorithms

In this section, we present the details of the implementation of the proposed method. First, the algorithms used for training and testing are presented. We then analyze the computational complexity of our method.

A. Augmentation Training

Adversarial attacks usually craft perturbations based on gradients. The main purpose of data augmentation is to produce new images such that their gradients are opposite to each other. To address this, we use intensity exchange technology to augment the training set.

Let $S = \{\mathrm{x}_i = (\mathrm{x}_i^{(0)}, \mathrm{x}_i^{(1)}, \mathrm{x}_i^{(2)})\}, i = 0, 1, \cdots, N-1$, where $\mathrm{x}_i^{(j)}$ is the j-th component image of the i-th input x_i. Let $\mathrm{x}_i^{(j)}(u, v) \in L_s = \{0, 1, \cdots, L-1\}$ be the intensity of a pixel at coordinates (u, v), $u = 0, 1, \cdots, H-1$, $v = 0, 1, \cdots, W-1$, and H and W are the height and width of the input, respectively. Without loss of generality, let $T_i^{(j)}$ be the intensity list of $\mathrm{x}_i^{(j)}$, i.e.,

$$T_i^{(j)} = \{t_0^{(ij)}, t_1^{(ij)}, \cdots, t_{p_{ij}-1}^{(ij)}\}, \tag{5}$$

where $t_0^{(ij)} < t_1^{(ij)} < \cdots < t_{p_{ij}-1}^{(ij)}$; $t_c^{(ij)}$ is a pixel intensity in $\mathrm{x}_i^{(j)}$, $c = 0, 1, \cdots, p_{ij} - 1$, p_{ij} is the number of intensities of $\mathrm{x}_i^{(j)}$.

The augmentation method on the training set is shown in Algorithm 1, where S_A is the augmented image set and Y_A is the corresponding label set; $\mathbb{Z}^{s^3N\times H\times W\times 3}$ and $\mathbb{Z}^{s^3N}$ are integer spaces in the size of $s^3N\times H\times W\times 3$ and s^3N, respectively; x_A is composed of the augmented images of x_i; $h(x)$ is a binary operator on x, $h(x)=1$ if x is true; otherwise, $h(x)=0$; $f(\bullet)$ is the augmentation module on a component image, as shown in Algorithm 2, and $[\bullet]$ is the least integer function.

In Algorithm 1, the augmentation process is mainly composed of three modules. First, three intensity lists $T_i^{(0)}$, $T_i^{(1)}$, and $T_i^{(2)}$ are obtained. Specifically, we draw all intensities in L_s in descending order. For every intensity $c\in L_s$, we scan the component image $\mathrm{x}_i^{(j)}$ to check whether there is a pixel whose intensity equals to c. If the condition is supported, c is appended to the $T_i^{(j)}$. Otherwise, we check the next intensity in the L_s. Second, we implement Algorithm 2 to augment every component image into s augmented component images. Finally, the s augmented component images are fully connected to generate s^3 augmented images, which compose the augmentation of x_i.

Algorithm 1 Data augmentation method based on intensity exchange.

```
Input: Training image set $S$ and corresponding label set $Y$; $k_1$; $s$
Output: The augmented image set $S_A$ and label set $Y_A$
 1: Initialization: $S_A = 0 \in \mathbb{Z}^{s^3N\times H\times W\times 3}$, $Y_A = 0 \in \mathbb{Z}^{s^3N}$, $\mathrm{x}_A = 0 \in \mathbb{Z}^{s^3\times H\times W\times 3}$
 2: for $i = 0 : N-1$ do
 3:     Data extraction: $\mathrm{x}_i = (\mathrm{x}_i^{(0)}, \mathrm{x}_i^{(1)}, \mathrm{x}_i^{(2)}) \in S$, $y_i \in Y$
 4:     for $j = 0,1,2$ do
 5:         Initialization: $T_i^{(j)} = \phi$
 6:         for $c = 0 : L-1$ do
 7:             if $\sum_{u=0}^{H-1}\sum_{v=0}^{W-1} h(\mathrm{x}_i^{(j)}(u,v) = c) \geq 1$ then
 8:                 $T_i^{(j)} = T_i^{(j)} \cup \{c\}$
 9:             end if
10:         end for
11:     end for
12:     for $j = 0,1,2$ do
13:         for $t = 0 : s-1$ do
14:             $n_t = 2^{k_1+t}$, $l_t = [p_{ij}/n_t]$
15:             $\mathrm{x}_A^{(jt)} = f(\mathrm{x}_i^{(j)}, T_i^{(j)}, H, W, k_1+t, l_t)$
16:         end for
17:     end for
18:     for $r = 0 : s-1$ do
19:         for $g = 0 : s-1$ do
20:             for $b = 0 : s-1$ do
21:                 $\mathrm{x}_A(rs^2+gs+b) = (\mathrm{x}_A^{(0r)}, \mathrm{x}_A^{(1g)}, \mathrm{x}_A^{(2b)})$
22:             end for
23:         end for
24:     end for
25:     $S_A(is^3 : (i+1)s^3-1, :, :, :) = \mathrm{x}_A$
26:     $Y_A(is^3 : (i+1)s^3-1) = y_i$
27: end for
```

For a given block parameter $k=k_1+t, (t=0,1,\cdots,s-1)$, let the augmentation of $\mathrm{x}_i^{(j)}$ be $f_i^{(j)}$. The main task of the augmentation is to build a one-to-one transformation on $T_i^{(j)}$. Algorithm 2 shows the details of the generation of $f_i^{(j)}$, where $l=[p_{ij}/2^k]$ is the block length, and p_{ij} is shown in (5).

In Algorithm 2, we scan $\mathrm{x}_i^{(j)}$ to generate $f_i^{(j)}$. For the current coordinates (u,v), we first search a $t_m^{(ij)}\in T_i^{(j)}$ such that $t_m^{(ij)}=\mathrm{x}_i^{(j)}(u,v)$. Then, with the help of indicator m, the intensity of the corresponding pixel in $f_i^{(j)}$ is assigned the intensity $a_m^{(ij)}$ using (6).

$$a_m^{(ij)} = \begin{cases} t_{m+l}^{(ij)}, m\in[2nl,(2n+1)l) \\ t_{m-l}^{(ij)}, m\in[(2n+1)l,2(n+1)l) \\ t_m^{(ij)}, m\geq 2^k l \end{cases}, \tag{6}$$

where $n=0,1,\cdots,2^{k-1}-1$.

Algorithm 2 Procedure for generating augmented component image.

```
Input: A component image $\mathrm{x}_i^{(j)}$ together with its intensity list $T_i^{(j)}$; the image height $H$ and width $W$; block parameters $k$ and $l$
Output: The augmented component image $f_i^{(j)}$
 1: Initialization: $f_i^{(j)} = 0 \in \mathbb{Z}^{H\times W}$
 2: for $u = 0 : H-1$ do
 3:     for $v = 0 : W-1$ do
 4:         Search a $t_m^{(ij)} \in T_i^{(j)}$ so that $t_m^{(ij)} = \mathrm{x}_i^{(j)}(u,v)$
 5:         if $m < 2^k l$ then
 6:             Let $f_i^{(j)}(u,v) = a_m^{(ij)}$ using (6)
 7:         else
 8:             Let $f_i^{(j)}(u,v) = t_m^{(ij)}$
 9:         end if
10:     end for
11: end for
```

After the training set is augmented using Algorithms 1 and 2, we combine the original and augmented examples to train the CNN model, as shown in (7).

$$\begin{cases} S_T = S_A \cup S \\ Y_T = Y_A \cup Y \end{cases}, \tag{7}$$

where S_T is the set of training images for Cons-Def, and Y_T is the label set corresponding to S_T.

B. Consensus Testing

Our defense scheme leverages the consensus on the predictions of augmented examples. The test image x (Fig. 3(b1)) may be a benign or adversarial example. Let x_{pc} be the padding and cropping image of x (Fig. 3(b2)), and the augmented images of x_{pc} be $\mathrm{x}_A \triangleq \{\mathrm{x}_A^{(0)}, \mathrm{x}_A^{(1)}, \cdots, \mathrm{x}_A^{(s^3-1)}\}$ (Fig. 3(b3)). We implement classifications on x_A using the model obtained in Fig. 3(a7). Let the predicted labels on x_A be $\hat{Y}_A \triangleq \{\hat{y}_0, \hat{y}_1, \cdots, \hat{y}_{s^3-1}\}$ (Fig. 3(b4)). Furthermore, let $\hat{Y}$ be the set of unique elements of $\hat{Y}_A$, i.e.,

$$\hat{Y} = \{\tilde{y}_0, \tilde{y}_1, \cdots, \tilde{y}_q\}, \tag{8}$$

and $\hat{n}_i$ be the number of occurrences of $\tilde{y}_i$ in $\hat{Y}_A$ (Fig. 3(b5)), i.e.,

$$\hat{n}_i = \sum_{k=0}^{s^3-1} h(\tilde{y}_i = \hat{y}_k), \tag{9}$$

where $\hat{y}_k \in \hat{Y}_A$.

Based on (8) and (9), the inference scheme is implemented in (10):

$$\hat{y}_{\mathrm{x}} = \begin{cases} \arg\max\limits_{\tilde{y}_i}(\hat{n}_i), & \max(\hat{n}_i) \geq T_c \\ -1, & \max(\hat{n}_i) < T_c \end{cases}, \tag{10}$$

where $\hat{y}_{\mathrm{x}}$ is the inferred classification, T_c is a given threshold, and $\hat{y}_{\mathrm{x}} = -1$ shows that the test image x is inferred as an adversarial example. In detail, the test procedure is shown in Algorithm 3.

Algorithm 3 Consensus testing on an unknown example.

```
Input: Test image x and threshold $T_c$
Output: Classification label $\hat{y}_{\mathrm{x}}$
1: Generate $\mathrm{x}_A$ using steps 4-24 in Algorithm 1
2: Classify examples in $\mathrm{x}_A$ to produce $\hat{Y}_A$ using the trained model
3: Obtain $\hat{Y}$ using $\hat{Y}_A$
4: Count the occurrences of the predicted labels using (9)
5: Produce classification using (10)
```

C. Complexity Analysis

We analyze the algorithm complexity to show the time efficiency. Since Cons-Def trains models offline, the runtime of the test is analyzed. As shown in Algorithm 3, the test algorithm mainly contains two modules: input augmentation and model classification. In the augmentation stage, three intensity lists are first obtained, and the operations on every list are approximated to WHL. Subsequently, every component image is expanded to s augmented images, and the operations of every augmentation implemented in Algorithm 2 are approximated to WHL. The operations involved in input augmentation are approximated as $3(s+1)WHL$. Since Cons-Def implements classifications on all augmented images, the operations on the classifications are approximated as s^3C, where C is the number of computations of the classification on one image. Overall, the operations of Cons-Def can be approximated as $3(s+1)WHL + s^3C$.

V. Experiments

In this section, experiments are implemented to demonstrate the defense performance of the proposed Cons-Def method. We first train models using augmented datasets and then implement classification and adversarial detection on corresponding test sets. We use a computer with an i5-7500 3.4 GHz CPU, 32 GiB system memory, and a GeForce GTX 1080Ti GPU to conduct the experiments. The experiments are implemented based on the CleverHans package [28] using TensorFlow-gpu-1.12.0.

A. Setup

Datasets: In this study, experiments are conducted on MNIST [29], CIFAR-10 [30], and ImageNet [31]. MNIST consists of 70 k gray images of handwritten digits in classes 0 to 9. The MNIST images, including 60 k training images and 10 k testing images, are 28×28 pixels. The CIFAR-10 dataset consists of 60 k 32×32 pixel RGB images, including 50 k images for training and 10 k images for testing. Since ImageNet is a large-scale dataset, many studies have selected a subset for defense tests [23]–[25]. In this study, experiments on ImageNet are conducted on ImageNet-10, which is extracted from the first ten classes in the dataset, e.g., tench, goldfish, great white shark, and tiger shark. ImageNet-10 consists of 13 k training images and 500 test images.

Networks: To implement the experiments on the datasets, we adopt six models with different convolutional structures. The architecture on MNIST, denoted as CNN-M, is mainly structured in three convolutional layers. This is identical to the basic model in the CleverHans package. For the CIFAR-10 dataset, we train three models: CNN-DT used in defensive distillation [26], ResNet-50 [32], and VGG-16 used in PixelDefend [22]. The CNN-DT network is structured into 4 convolutional layers, 2 pooling layers, and 2 FC layers. For convenience, we denote CNN-DT as 4C+2P+2FC and use similar notations in the following sections. For ImageNet-10, three models are employed in the experiments: ResNet-50 [32], ResNet-101 [32], and Inception-v3 used in RCA-SOC [1]. The training parameters are summarized in Table I, where ResNet-50, ResNet-101, and Inception-v3 are abbreviated to Res50, Res101, and Incept3, and Adadelta and momentum are abbreviated as Adad and Mome, respectively. The experiments on CNN-M and CNN-DT are both implemented in a batch size of 128 using an adaptive moment estimation (Adam) optimizer with a learning rate of 0.001. For ResNet-50, ResNet-101, and VGG-16, the Adadelta optimizer with an initial learning rate of 0.1 is used for training. For ResNet-50 and VGG-16 on CIFAR-10, the height and width of the images fed to the two models are resized to 128×128 and 160×160 pixels, respectively. The batch sizes are set to 128 and 90. For ImageNet-10, the input sizes of ResNet-50 and VGG-16 are both resized to 224×224. The batch sizes of the two models are set to 80 and 48. The Inception-v3 model is trained with a batch size of 64 and an input size of 299×299. The model is trained using a momentum optimizer with a dropout rate of 0.5. The momentum is set to 0.9. The learning rate is initialized at 0.045 and decayed every two epochs at an exponential rate of 0.94.

TABLE I
Training parameters of the models on MNIST, CIFAR-10, and ImageNet-10.

Dataset	MNIST	CIFAR-10			ImageNet-10		
Network	CNN-M	CNN-DT	Res50	VGG16	Res50	Res101	Incept3
Input size	28^2	32^2	128^2	160^2	224^2	224^2	299^2
Optimizer	Adam	Adam	Adad	Adad	Adad	Adad	Mome
Dropout rate	-	-	-	0.5	-	-	0.5
Learning rate	0.001	0.001	0.1	0.1	0.1	0.1	0.045
Batch size	128	128	128	90	80	48	64

Attack methods: To demonstrate the defense ability, both white-box and black-box attacks are employed in the experiments. For white-box attacks, untargeted attacks, including FGSM (l_∞-norm) [8], C&W (l_2-norm) [11], JSMA [12], PGD (l_∞-norm) [9], and DeepFool [10], are implemented on the MNIST, CIFAR-10, and ImageNet-10 datasets. For black-box attacks, the transferability of adversarial examples crafted on CIFAR-10 is studied.

Metrics: Because a robust defense method should show high accuracy on benign and adversarial examples, the CA and acc shown in (2) and (4) are both reported in our comparison experiments. Moreover, our proposed method could classify adversarial examples and detect adversarial attacks, and we use the defense success rate (DSR) to evaluate the defense ability against adversarial attacks, as shown in (11).

$$DSR = \frac{n_{adv}(l_p = l_T) + n_{adv}(d = 1)}{N_{adv}}, \tag{11}$$

where N_{adv}, $n_{adv}(l_p = l_T)$, and $n_{adv}(d = 1)$ are given in (2) and (3), respectively.

B. Parameter Tuning

Our proposed method includes three main parameters. The intensity list on the component image is first divided into $2^k, k = k_1, k_1+1, \cdots, k_1+s-1$, splits to augment examples. The method then uses the threshold T_c to implement classification and discrimination in the test stage (Fig. 3). Clearly, k_1, s, and T_c are the three parameters of our method. For convenience, we fix $k_1 = 3$ and use the one-factor-at-a-time method to tune the parameters s and T_c on CIFAR-10.

For s, 3 and 4 are two selected levels for experiments. For $s = 3$, the intensity list is divided into 2^3, 2^4, and 2^5 splits, i.e., $k = 3, 4, 5$. Every component image produces three augmented component images based on k. The augmented components are fully connected to produce 27 augmented RGB images. Together with the original training examples, all augmented RGB images are employed to train a model. For brevity, we denote the resulting model as $Model_{27}$. As a contrast experiment, $Model_{64}$ is trained with level $s = 4$. Fig. 4 shows the ablation experiments on the parameter s. The black and red lines in Fig. 4 show the classification accuracies on the CIFAR-10 test set using $Model_{27}$ and $Model_{64}$, respectively. Both benign and adversarial examples are used for testing. The results on the benign and adversarial examples are shown as the "clean test" and "Adv test", respectively, in Fig. 4. The adversarial examples are crafted by FGSM with a perturbation of 0.03 on the test examples of CIFAR-10. As shown in Fig. 4, the $Model_{64}$ leverages the stability of accuracy. Therefore, $s = 4$ is employed in the proposed method.

Generally, the defense accuracies on benign and adversarial examples should be as high as possible. To balance the performance on benign examples, we use DSR on benign and adversarial examples (DSR†) to choose T_c. DSR† is defined in (12).

$$DSR^\dagger = \frac{n_{ben}(l_p = l_T) + n_{adv}(l_p = l_T) + n_{adv}(d = 1)}{N_{adv} + N_{ben}}, \tag{12}$$

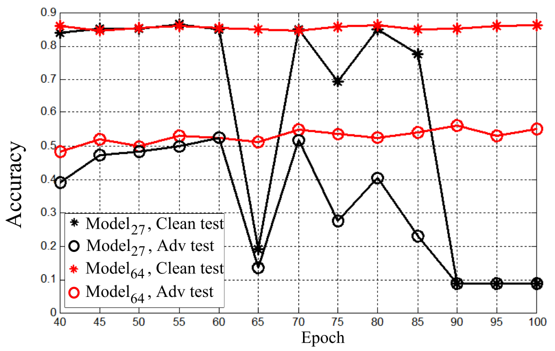


Fig. 4. Ablation experiments of the parameter s on the CIFAR-10 test set. The black and red results show test accuracies using $\boldsymbol{Model_{27}}$ and $\boldsymbol{Model_{64}}$, respectively. The results marked with asterisks and small circles show accuracies on benign and adversarial examples, respectively.

where the notations are shown in (2)-(4).

For parameter T_c, we fix parameter s at 4, and select nine levels $32, 36, \cdots, 64$ for the experiments. Table II lists the ablation experiments on the threshold T_c. The DSR† results in Table II are reported on the CIFAR-10 test set using $Model_{64}$. The perturbation parameters of the FGSM and PGD are both set to 0.03, i.e., $\varepsilon_F = \varepsilon_P = 0.03$. The number of iterations for PGD is set to 10 with a step size of $\varepsilon_P/4$. The number of iteration steps for C&W is set to 1000 with a learning rate of 0.01. The constant parameter c for the C&W attack is set to 10. The parameter of the maximum distortion percentage for JSMA is set as $\gamma = 0.1$. The number of maximum iterations for DeepFool is set to $it_{DF} = 50$, where DeepFool is abbreviated as DFool in Table II.

TABLE II
DSR† OF THE THRESHOLD $\boldsymbol{T_c}$ ON CIFAR-10 (%).

T_c	FGSM $\varepsilon_F = 0.03$	C&W $c = 10$	JSMA $\gamma = 0.1$	PGD $\varepsilon_P = 0.03$	DFool $it_{DF} = 50$	Average
32	75.8	83.9	72.5	69.2	84.6	77.2
36	76.2	84.2	74.2	69.7	84.9	77.8
40	76.9	84.6	75.3	70.2	85.1	78.4
44	77.4	84.8	75.7	70.6	85.2	78.7
48	77.9	84.9	76.9	71.1	85.3	79.2
52	78.4	84.6	78.0	71.7	85.1	79.6
56	78.7	84.4	78.9	72.4	84.8	79.8
60	79.0	83.8	79.8	73.3	84.1	80.0
64	78.1	81.5	78.4	73.7	81.9	78.7

As summarized in Table II, $T_c = 60$ achieves the highest DSR† of 80.0%. In this study, $T_c = 60$ is chosen to defend against adversarial attacks on $Model_{64}$.

C. Defense against White-box Attacks

For MNIST, we train the models on a network with the structure of 3C+1FC. The clean model is trained using 60 k training images. As the examples of MNIST are grayscale images, the gray intensity list is divided into 2, 4, 8, 16, and 32 splits for intensity exchange. The volume of the MNIST dataset is augmented 5-fold for the experiments. The augmented model is trained on 360 k examples after augmentation. The clean and augmented models are both trained for 50 epochs with the corresponding parameters listed in Table I. We report the robustness of the model using a standard perturbation for FGSM and PGD [5], i.e., ε_F and ε_P are both set to 0.3 on the MNIST test set. The attack parameters for C&W, JSMA, and DeepFool are the same as those listed in Table II.

Table III presents the comparison experiments against FGSM, C&W, JSMA, PGD, and DeepFool attacks on the MNIST test set. The accuracies in the "clean" row are obtained on the test set using the clean model without any defense strategy. Since our method can simultaneously classify and detect adversarial examples, both CA and DSR are summarized in Table III, where Res20 is the abbreviation for ResNet-20. The defense threshold T_c is set to 5. The test experiments on JSMA are implemented on the first 1000 images of the test set.

TABLE III
DEFENSE ACCURACY AGAINST WHITE-BOX ATTACKS ON THE MNIST TEST SET (%).

Method	Metrics	Network	FGSM	C&W	JSMA	PGD	DFool
Clean	CA	3C+1FC	24.9	1.0	17.5	2.1	2.1
Feat-Squ [25]	CA	4C+2P+2FC	61.0	35.0	56.0	-	-
ADP$_{2,0.5}$ [33]	CA	3Res20	52.8	23.8	95.0	41.0	-
GPMR [19]	CA	8C+3P+2FC	58.7	28.5	95.0	51.4	-
DSDL [5]	CA	6C+3FC	31.1	29.1	-	19.9	-
RCA-SOC [1]	CA	5C+5P+1FC	67.3	-	71.6	-	44.3
DD-AD [18]	CA	2C+2FC	83.4	97.8	-	-	-
Cons-Def	CA	3C+1FC	89.7	2.3	53.7	39.1	2.1
Cons-Def	DSR	3C+1FC	98.3	99.3	91.6	91.6	99.8

Cons-Def is robust to adversarial attacks. Although the comparison results in Table III result in different networks, most of their structures are similar, except for the adaptive diversity-promoting regularizer (ADP$_{2,0.5}$) network, which is an ensemble suite consisting of three ResNet-20 networks. Although Cons-Def shows classification disability against C&W and DeepFool attacks, the DSRs against both attacks are greater than 95%. Furthermore, Cons-Def achieves the highest defense rate against FGSM, C&W, PGD, and DeepFool. Although Cons-Def did not obtain the best result on JSMA, it showed overall superiority on MNIST.

For CIFAR-10, the clean and augmented models of the CNN-DT are trained for 100 epochs. The clean models of ResNet-50 and VGG-16 are trained for 1000 k and 500 k iterations, respectively. The augmented models of ResNet-50 and VGG-16 are trained for 2500 k iterations. For ImageNet-10, the clean models of ResNet-50 and ResNet-101 are trained for 200 k iterations. The augmented models of ResNet-50 and ResNet-101 on ImageNet-10 are trained for 1700 k iterations. The clean and augmented models of Inception-v3 are trained for 100 epochs. The other training parameters are the same as those in Table I. Tables IV and V list the comparison results on CIFAR-10 and ImageNet-10, respectively. The perturbations for the FGSM and PGD are both set to 0.03 [5], i.e., $\varepsilon_F = \varepsilon_P = 0.03$. The attack parameters for C&W, JSMA, and DeepFool are the same as those listed in Table II. Since JSMA requires more memory than what is available to run, we could not test it on the ResNet, VGG-16, and Inception-v3 networks.

TABLE IV
DEFENSE ACCURACY AGAINST WHITE-BOX ATTACKS ON THE CIFAR-10 TEST SET (%).

Method	Metrics	Network	FGSM	C&W	JSMA	PGD	DFool
Clean-CNN	CA	4C+2P+2FC	19.1	5.8	3.3	9.0	8.8
Clean-Res50	CA	Res50	10.0	0.0	-	2.9	6.1
Clean-Vgg16	CA	VGG16	10.6	0.0	-	6.6	6.5
JPEG-comp [21]	CA	4C+2P+1FC	79.6	-	-	-	82.7
PixelDefend [22]	CA	VGG16	62.0	79.0	-	-	76.0
ADP$_{2,0.5}$ [33]	CA	3Res20	46.2	25.6	37.0	30.4	-
GPMR [19]	CA	8C+3P+2FC	56.0	32.9	48.1	35.8	-
CGAN [17]	CA	ResNet	82.8	79.8	82.1	-	84.2
DSDL [5]	CA	Res110	67.7	37.3	-	27.2	-
RCA-SOC [1]	CA	VGG16	45.6	-	50.2	-	25.0
Cons-Def-CNN	CA	4C+2P+2FC	54.8	66.9	35.6	41.2	64.1
Cons-Def-Res50	CA	Res50	55.2	81.8	-	54.5	82.0
Cons-Def-Vgg16	CA	VGG16	19.7	54.2	-	10.8	46.0
Cons-Def-CNN	DSR	4C+2P+2FC	84.5	94.7	85.8	73.3	95.3
Cons-Def-Res50	DSR	Res50	88.5	95.9	-	84.6	96.1
Cons-Def-Vgg16	DSR	VGG16	54.5	91.8	-	23.8	93.1

From Table IV, our method shows high performance in adversarial defense. The results in the first three lines in Table IV indicate that clean models without a defense strategy are heavily attacked. Compared to clean models, Cons-Def achieves high performance on CA and DSR. This result suggests that Cons-Def is effective against adversarial attacks. Our experiments on the CNN-DT, ResNet-50, and VGG-16 networks show that the same attack with the same parameters could drive different networks to produce significantly different accuracies. The results in Table IV are obtained using three types of structures. The results of Cons-Def-CNN are compared with those of basic convolutional networks, i.e., JPEG compression (JPEG-comp) [21] and GPMR [19]. The average CA of GPMR against the four attacks is 43.2%, which is smaller than the 52.5% for the Cons-Def-CNN. Furthermore, because Cons-Def-CNN could detect adversarial examples, the CAs of JPEG-comp and GPMR are smaller than the DSRs of Cons-Def-CNN. We compare the Cons-Def-Res50 with ADP$_{2,0.5}$ [33], CGAN [17], and DSDL [5]. The average CAs of ADP$_{2,0.5}$ and DSDL are 34.8% and 44.1%, respectively. Since the average CA of Cons-Def-Res50 is 68.4%, it seems that Cons-Def-Res50 is more robust than ADP$_{2,0.5}$ and DSDL. Although the CAs of CGAN are greater than those of Cons-Def-Res50, the DSRs of Cons-Def-Res50 are all greater than the CAs of CGAN. For VGG-16, the CAs of Cons-Def-Vgg16 are inferior to those of PixelDefend [22] and RCA-SOC [1]. The DSR of Cons-Def-Vgg16 against PGD is low. However, Cons-Def-Vgg16 shows its superiority against C&W and DeepFool attacks. Overall, Cons-Def shows its superiority against adversarial attacks on the CIFAR-10 test set.

The results in Table V indicate that Cons-Def is advantageous for adversarial defense on the test set of ImageNet-10. As listed in Table V, the defense performance on the ResNet-50 network is close to that of ResNet-101. For ResNet-50, we compare Cons-Def-Res50 with TVM [23], image quilting [23], and STL [24]. Cons-Def-Res50 shows its superiority to TVM and image quilting in terms of CA and DSR. The DSRs of Cons-Def-Res50 are all greater than the CAs of TVM, image quilting, and STL except for the case of STL on FGSM. For

TABLE V
DEFENSE ACCURACY AGAINST WHITE-BOX ATTACKS ON THE IMAGENET-10 TEST SET (%).

Method	Metrics	Network	FGSM	C&W	PGD	DFool
Clean-V3	CA	Incept3	23.2	0.0	13.2	12.6
Clean-Res50	CA	Res50	23.4	0.0	16.8	16.8
Clean-Res101	CA	Res101	23.6	0.0	16.6	16.6
TVM [23]	CA	Res50	31.4	48.4	-	44.7
Image quilting [23]	CA	Res50	39.6	30.5	-	34.5
ALP [16]	CA	Incept3	-	-	27.9	-
STL [24]	CA	Res50	69.3	67.7	-	68.5
RCA-SOC [1]	CA	Incept3	29.0	-	-	71.6
Cons-Def-V3	CA	Incept3	36.4	58.6	27.2	57.6
Cons-Def-Res50	CA	Res50	41.2	65.8	38.6	64.0
Cons-Def-Res101	CA	Res101	43.6	64.4	38.6	62.6
Cons-Def-V3	DSR	Incept3	64.0	82.6	51.0	85.6
Cons-Def-Res50	DSR	Res50	68.8	84.2	63.0	85.4
Cons-Def-Res101	DSR	Res101	69.4	84.4	66.2	85.4

Inception-v3, the DSRs of Cons-Def-V3 are all higher than the CAs of ALP [16] and RCA-SOC [1].

Overall, the average CA and DSR of Cons-Def on the three datasets are 48.3% and 80.3%, respectively. Although Cons-Def is not sufficiently strong for classification, it is robust to DSR. Comparison experiments on MNIST, CIFAR-10, and ImageNet-10 suggest that Cons-Def shows superiority against white-box attacks.

D. Defense against Black-box Attacks

In this section, we present defense results against black-box attacks on CIFAR-10. We study the transferability of the CNN-DT, ResNet-50, and VGG-16 models. In this study, FGSM, C&W, PGD, and DeepFool are employed for black-box attacks. The parameters of the attacks are the same as those listed in Table IV. Table VI lists the resulting CA and DSR of Cons-Def against black-box attacks on CIFAR-10; for example, the result of 60.6/93.2 in Table VI indicates that the CA and DSR are 60.6% and 93.2%, respectively.

TABLE VI
RESULTING CA AND DSR OF CONS-DEF AGAINST BLACK-BOX ATTACKS ON CIFAR-10 (%).

Target	Source	FGSM	C&W	PGD	DFool
CNN-DT	Res50	60.6/93.2	65.1/93.9	62.2/93.6	65.3/94.2
CNN-DT	VGG16	34.0/84.6	54.7/92.3	32.5/76.9	51.0/91.5
Res50	CNN-DT	72.3/92.9	80.8/95.9	72.7/93.2	80.3/95.7
Res50	VGG16	47.7/84.3	75.6/94.8	44.4/77.1	73.1/95.3
VGG16	CNN-DT	71.7/91.9	77.6/93.6	71.1/91.6	77.2/94.0
VGG16	Res50	75.5/91.8	78.5/93.9	77.5/93.1	78.4/94.1

The results in Table VI indicate that our proposed Cons-Def is robust against black-box attacks. The CAs of the target models CNN-DT, ResNet-50, and VGG-16 on CIFAR-10 without attacks are 86.1%, 91.3%, and 88.1%, respectively. As shown in the first and second lines in Table VI, the average classifications of CNN-DT attacked by ResNet-50 and VGG-16 are 63.3% and 43.1%, and the losses are 22.8% and 43.1%, respectively. The average losses of ResNet-50 attacked by CNN-DT and VGG-16 are 14.8% and 31.1%, respectively. The average losses of VGG-16 attacked by CNN-DT and ResNet50 are 13.7% and 10.6%, respectively. The results show that the structure of the source network has significant variance with respect to the power of black-box attacks. The average CAs of the models attacked by CNN-DT, ResNet-50, and VGG-16 are 70.4%, 75.5%, and 51.6%, respectively. The strongest attacks in our test suite are crafted using the VGG-16. The average CA and DSR of black-box attacks in Table VI are 65.8% and 91.4%, respectively. The DSRs in Table VI indicate that Cons-Def can classify or detect adversarial examples at a high rate. Overall, Cons-Def is robust against black-box attacks on CIFAR-10.

E. Accuracy on Benign Examples

In this section, we show the resulting accuracies on benign examples in Table VII using (4). The accuracies in the "clean" row are obtained on the clean model without any defense strategy. The models in the "augmented" row are trained on our augmented training sets, and the accuracies in the row are tested with original test examples without augmentation. The accuracies in the row of Cons-Def in Table VII result from the augmented models and our defense scheme.

TABLE VII
ACCURACY OF THE MODEL ON BENIGN EXAMPLES (%).

Dataset	MNIST	CIFAR-10			ImageNet-10		
Model	CNN-M	CNN-DT	Res50	VGG16	Incept3	Res50	Res101
Clean	99.3	86.0	90.6	89.7	69.4	68.6	74.0
Augmented	99.0	86.1	91.3	88.1	73.6	73.6	73.0
Cons-Def	98.0	73.4	82.6	79.0	64.0	67.6	66.8

Cons-Def achieves high performance on benign examples. As summarized in Table VII, the average accuracies of the clean, augmented, and Cons-Def models are 82.5%, 83.5%, and 75.9%, respectively. Compared to the clean models, the average improvement in the accuracies of the augmented models is 1.0%. Intensity-based data augmentation is advantageous for classification purposes. The average accuracies of the clean model on the three datasets are 99.3%, 88.8%, and 70.7%, respectively. The average accuracies of Cons-Def on the three datasets are 98.0%, 78.3%, and 66.1%, respectively. Correspondingly, the average losses of Cons-Def on the three datasets are 1.3%, 10.4%, and 4.5%, respectively. Overall, Cons-Def correctly classified most benign examples, and the deficiency of our method on the benign examples is limited in an acceptable range. Cons-Def exhibits high performance on benign examples.

F. Robustness Experiments

Adversarial examples are crafted using attack parameters. Since different parameters produce different attack powers, we test the robustness against these parameters. Fig. 5 shows the defense results against white-box attacks on CIFAR-10. The experimental parameters of FGSM, C&W, PGD, and DeepFool are ε_F, c, ε_P, and it_{DF}, respectively. To test the robustness of Cons-Def, five levels are chosen for the four perturbation parameters: $\varepsilon_F = \varepsilon_P = 0.01, 0.03, 0.1, 0.3, 0.8$, $c = 0.01, 0.1, 1.0, 10, 1000$, and $it_{DF} = 1, 2, 10, 50, 500$. The blue and red bars indicate classification and detection accuracies, respectively.

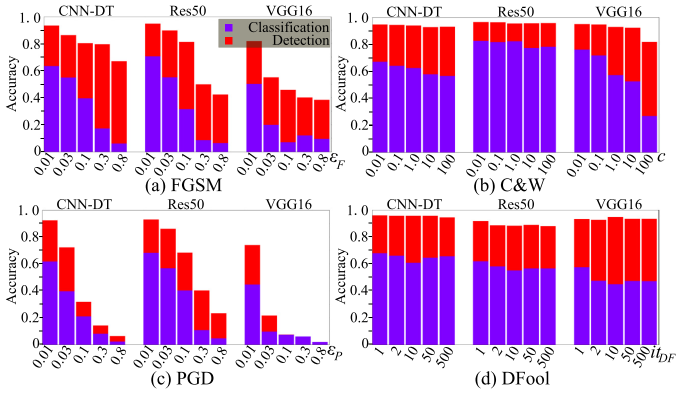


Fig. 5. Robustness experiments against white-box attacks on the CIFAR-10 test set. (a), (b), (c), and (d) are the defense results against FGSM, C&W, PGD, and DeepFool attacks, respectively. The blue and red results indicate classification and detection accuracies, respectively.

As shown in Figs. 5(b) and 5(d), Cons-Def is robust to adversarial examples crafted by C&W and DeepFool. Although the DSRs shown in Fig. 5(a) are distributed over a large range, the red bars in Fig. 5(a) indicate that Cons-Def is robust against adversarial detection. As shown in Fig. 5(c), Cons-Def is not robust to PGD attacks.

G. Time Complexity

In this section, we evaluate the runtime of Cons-Def on the MNIST, CIFAR-10, and ImageNet-10 test sets. Table VIII lists the average defense time of Cons-Def. The results of the CPU and GPU show the augmentation and model test times per image, respectively. As summarized in Table VIII, the time cost for the different models ranges from 0.003 to 0.738 s per image. The runtime is related to the size of the data examples. A larger example often requires more processing time. The average defense speed of ResNet-50 and ResNet-101 against the adversarial input on ImageNet-10 with a size of 224×224 is approximately 2 fps. For Inception-v3, the input is in size of 299×299, and the defense speed is less than 2 fps. Cons-Def is less efficient in terms of the runtime.

TABLE VIII
RUNTIME OF CONS-DEF (SECOND).

Dataset	MNIST	CIFAR-10			ImageNet-10		
Model	CNN-M	CNN-DT	Res50	VGG16	Incept3	Res50	Res101
CPU	0.003	0.007	0.058	0.101	0.452	0.205	0.225
GPU	0.000	0.004	0.067	0.230	0.286	0.206	0.316
Total	0.003	0.011	0.126	0.331	0.738	0.411	0.541

VI. CONCLUSIONS

In this study, a consensus defense method was proposed. Cons-Def uses intensity exchange technology to augment the training examples. The gradients of the augmented examples were opposite to each other. Cons-Def uses the consensus of classifications on augmented examples to defend against adversarial attacks. On the one hand, Cons-Def trains models based on augmented examples. On the other hand, it implements classification and adversarial detection on augmented examples using the consensus of their predictions. The Cons-Def method showed a high classification performance on benign examples and was robust against white-box and black-box adversarial attacks. Notably, Cons-Def could not defend against all adversarial attacks. If an adversarial attack is successful on all augmented examples of an input, Cons-Def encounters a failure defense.

The experimental results showed that the structure of the network plays an important role in an attack. The same white-box attack with the same parameters can drive different networks to produce significantly different accuracies. For black-box attacks, the attack performance varies significantly with respect to the structure of the source network. Therefore, we plan to study the propagation errors of perturbations in our future work.

REFERENCES

[1] J. Chen, H. Zheng, R. Chen, and H. Xiong, "RCA-SOC: A novel adversarial defense by refocusing on critical areas and strengthening object contours," *Computers & Security*, vol. 96, Sept. 2020, Art. no. 101916.

[2] J. Zhang, M. Z. A. Bhuiyan, X. Yang, A. K. Singh, D. F. Hsu, and E. Luo, "Trustworthy target tracking with collaborative deep reinforcement learning in EdgeAI-aided IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1301–1309, Feb. 2021.

[3] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, Sept. 2020.

[4] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, May 2020.

[5] A. Mustafa, S. H. Khan, M. Hayat, R. Goecke, J. Shen, and L. Shao, "Deeply supervised discriminative learning for adversarial defense," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 3154–3166, Sept. 2021.

[6] A. H. Celdrán, M. G. Pérez, I. Mlakar, J. M. A. Calero, F. J. G. Clemente, G. M. Pérez, and Z. A. Bhuiyan, "PROTECTOR: Towards the protection of sensitive data in Europe and the US," *Computer Networks*, vol. 181, Nov. 2020, Art. no. 107448.

[7] X. Xu, J. Zhang, Y. Li, Y. Wang, Y. Yang, and H. T. Shen, "Adversarial attack against urban scene segmentation for autonomous vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4117–4126, Jun. 2021.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, San Diego, CA, USA, 2015.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv: 1706.06083*, 2019.

[10] 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*, Las Vegas, NV, USA, 2016, pp. 2574–2582.

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

[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, Saarbruecken, Germany, 2016, pp. 372–387.

[13] L. Yang, Q. Song, and Y. Wu, "Attacks on state-of-the-art face recognition using attentional adversarial attack generative network," *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 855–875, Sept. 2021.

[14] H. Wang, G. Li, X. Liu, and L. Lin, "A Hamiltonian Monte Carlo method for probabilistic adversarial attack and learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 4, pp. 1725–1737, Apr. 2022.

[15] Y. Wang, H. Lv, X. Kuang, G. Zhao, Y. Tan, Q. Zhang, and J. Hu, "Towards a physical-world adversarial patch for blinding object detection models," *Information Sciences*, vol. 556, pp. 459–471, May 2021.

[16] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv: 1803.06373*, 2018.

[17] F. Yu, L. Wang, X. Fang, and Y. Zhang, "The defense of adversarial example with conditional generative adversarial networks," *Security and Communication Networks*, vol. 2020, 2020, Art. no. 3932584.

[18] J. Zhu, G. Peng, and D. Wang, "Dual-domain-based adversarial defense with conditional VAE and Bayesian network," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 596–605, Jan. 2021.

[19] A. Dabouei, S. Soleymani, F. Taherkhani, J. Dawson, and N. M. Nasrabadi, "Exploiting joint robustness to adversarial perturbations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 2020, pp. 1122–1131.

[20] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 274–283.

[21] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with JPEG compression," *arXiv: 1705.02900*, 2017.

[22] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," in *International Conference on Learning Representations*, Vancouver, BC, CAN, 2018.

[23] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations*, Vancouver, BC, CAN, 2018.

[24] B. Sun, N.-h. Tsai, F. Liu, R. Yu, and H. Su, "Adversarial defense by stratified convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 11 447–11 456.

[25] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed Systems Security Symposium (NDSS)*, San Diego, CA, USA, 2018.

[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)*, San Jose, CA, USA, 2016, pp. 582–597.

[27] J. Liu, W. Zhang, Y. Zhang, D. Hou, Y. Liu, H. Zha, and N. Yu, "Detection based defense against adversarial examples from the steganalysis point of view," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 4825–4834.

[28] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy *et al.*, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv: 1610.00768*, 2018.

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

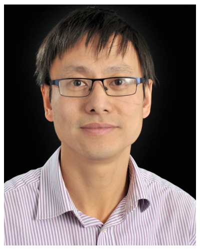
[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Dept. Comp. Sci., Toronto Univ., YTO, ON, CAN, 2009.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.

[33] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 4970–4979.

Xintao Ding received the BSc in mathematics from Anqing Normal University in 2001 and the Ph.D. from the School of Geography and Tourism, Anhui Normal University in 2015. He conducted an Academic Visitor in the Department of Computer Science and Technology, University of Hull, UK, in 2016. He is an Associate Professor with the school of Computer and Information, Anhui Normal University, China. His research interests include computer vision and AI.



Yongqiang Cheng received BEng and MEng degrees in Control Theory and Control Engineering from Tongji University, Shanghai, China in 2001 and 2004, respectively. He obtained his Ph.D. degree in 2010 from School of Engineering, Design and Technology, the University of Bradford, UK. Currently, he is a Reader with the Department of Computer Science and Technology, University of Hull, UK. His research interests include digital healthcare technologies, embedded systems, data mining, and AI.

Yonglong Luo received his Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China in 2005. Currently, he is a Professor with the school of Computer and Information, Anhui Normal University, China. He is also the Director of Anhui Provincial Key Laboratory of Network and Information Security. His research interests include information security and AI.

Qingde Li received the BSc in mathematics from Beijing Normal University in 1982 and the Ph.D. in computer science from the University of Hull in 2002. He has been a Lecture in computer science at the University of Hull since 2001. Previously he was a Professor and the deputy head of Department of Mathematics and Computer Science at Anhui Normal University, China. His recent research interests are in the area of computer graphics and visualization, mixed reality technology, and geometric modelling.

Prosanta Gope is currently working as an Assistant Professor in the Department of Computer Science (Cyber Security) at the University of Sheffield, UK. Dr. Gope served as a Research Fellow in the Department of Computer Science at the National University of Singapore (NUS). He has expertise in lightweight authentication, security of mobile communications, healthcare, Internet of Things, Cloud, RFIDs, WSNs, Smart-Grid and IoT Hardware. He received the Distinguished PhD. Scholar Award in 2014 from the National Cheng Kung University (Taiwan). Several of his papers have been published in high impact journals such as IEEE TIFS, IEEE TDSC, IEEE TIE, IEEE TSG, IEEE JBHI, etc. He currently serves as an Associate Editor of the IEEE Internet of Things Journal, IEEE Systems Journal, IEEE Sensors Journal, and Journal of Information Security and Applications (Elsevier).