Defending against Adversarial Attack towards Deep Neural Networks via Collaborative Multi-task Training

Derek Wang, Chaoran Li, Sheng Wen, Surya Nepal, and Yang Xiang

Abstract—Deep neural networks (DNNs) are known to be vulnerable to adversarial examples which contain imperceptible perturbations. A series of defending methods, either proactive defence or reactive defence, have been proposed in the recent years. However, most of the methods can only handle specific attacks. For example, proactive defending methods are invalid against grey-box or white-box attack, while reactive defending methods are challenged by low-distortion adversarial examples or transferring adversarial examples. This becomes a critical problem since a defender usually do not have the type of the attack as a priori knowledge. Moreover, the two-pronged defence (e.g. MagNet), which takes the advantages of both proactive and reactive methods, has been reported as broken under transferring attacks. To address this problem, this paper proposed a novel defensive framework based on collaborative multi-task training, aiming at providing defence for different types of attacks. The proposed defence first encodes training labels into label pairs and constructs a detector to identify and reject high-confidence adversarial examples when the defending strategy is exposed. In the experiments, we evaluated our defence against four typical attacks on MNIST and CIFAR10 datasets. These four attacks represent the state-of-the-art, which have been widely used in prior works. The results showed that our defending method achieved up to 96.3% classification accuracy on black-box adversarial examples, and detected up to 98.7% of the high confidence adversarial examples. It only decreased the model accuracy on benign example classification by 2.1% for the CIFAR10 dataset. As far as we know, our method is a new two-pronged defence that is resilient to the transferring attack targeting MagNet.

Index Terms—Deep neural network, adversarial example, security.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable performance on various tasks, such as computer vision, natural language processing and data generation. However, DNNs are vulnerable towards adversarial attacks which exploit imperceptibly perturbed examples to fool the neural networks [38]. For instance, deliberately crafted adversarial examples can easily deceive DNN-based systems such as hand-writing recognition [32], face detection [21], and autonomous vehicle [12], making the models generate wrong outputs. The adversarial examples are even possible to trigger catastrophic consequences, such as causing accident originated by faulty object detection in autonomous vehicles [24]. Considering the blooming DNN-based applications in modern electronic systems, as well as it will likely be in the future applications, proposing effective defensive methods to defend DNNs against adversarial examples has never been this urgent and critical.

Adversarial attacks against DNN-based systems can be categorised into three types based on the a priori knowledge that attackers have: 1) black-box attacks, 2) grey-box attacks, and 3) white-box attacks. Currently, there are also a series of defending methods proposed, which can be broadly divided into proactive defence (e.g. [6], [16], [34], [36]) and reactive defence (e.g. [11], [14], [23], [29], [41]). However, they all have limitations. Specifically, current countermeasures can only counter attackers in specific scenarios.

For proactive defending methods, they focus on transferring attacks launched in black-box settings. However, they do not work in grey-box and white-box scenarios since this type of defence relies on model parameter regularisation and robust optimisation to mitigate the effects of adversarial examples. They become invalid once the parameters or the defending strategies are known to attackers. Moreover, this type of defence can be bypassed by using high-confidence adversarial examples (e.g., Carlini-Wagner attack [5]), even in black-box settings. This type of defence is also not resilient to the attacks that exploit input feature sensitivity (e.g., Jacobian matrix based attacks [32]). Instead of passively strengthening models’ robustness, reactive defending methods can capture adversarial examples that have higher attacking confidence and distortion [11], [14], [23], [29], [41]. However, these methods have been demonstrated to be vulnerable to specific transferring attacks [3].

In fact, the type of attack is usually unknown to the defenders, making the selection of a proper defence very challenging. We also cannot simply ensemble the above defending methods together to form a general defence, as attackers can compromise each defending method one by one [17]. MagNet [28] provides a two-pronged defence...
that does not require too much a priori knowledge on the type of the coming attack. However, this method has been reported to be broken by transferring attack directed from substitute autoencoders [4]. So far, there is no secure method that resides with the advantages from both proactive and reactive methods for a general defence.

In this paper, we propose a well-rounded defence that not only increases the robustness of neural networks to low-confidence transferring attacks, but also detects high-confidence black-box/grey-box adversarial examples at a high accuracy. Moreover, our proposed defence can prevent an adversary from finding adversarial examples when the defending strategy is known to the adversary. Our method first introduces adversarial training with robust label pairs to tackle black-box attack. Then it employs a multi-task training technique to construct an adversarial example detector. The proposed method is able to tackle both transferring attack launched in the black-box setting and the adversarial example generation based on the targeted model in the grey-box setting. The main contributions of the paper are summarised as follows:

- We introduced a novel collaborative multi-task training framework as a defence to invalidate transferring adversarial examples;
- This defence uses data manifold information to detect high-confidence adversarial examples crafted in grey-box/black-box settings;
- The proposed defence can prevent an adversary from searching valid adversarial examples using the targeted model in grey-box settings;
- The proposed defence is resilient to the transferring attack which breaks the previous two-pronged defence;
- We carry out both empirical and theoretical studies to evaluate the proposed defence. The experimental results demonstrate that our defence is effective against adversaries with different prior knowledge.

The paper is organised as follows: Section 2 describes the state-of-the-art attacks and clarifies the problem statement and our contributions. Section 3 presents our detailed approach. Section 4 represents the evaluation of our approach. Section 5 provides an analysis on the mechanism of the defence. Section 6 presents a conclusion on the existing remaining unsolved problems of the existing attacks and defences, as well as the possible further improvements of the defence. Section 7 summaries the paper, and proposes the future works.

2 PRIMER

2.1 Adversarial attacks

We first introduce the state-of-the-art attacks in the field. Supposing the DNN model is equal to a non-convex function $F$. In general, given an image $x$ along with the rightful one-hot encoded label $y_{true}$, an attacker searches for the adversarial example $x_{adv}$.

2.1.1 FGSM

Fast gradient sign method (FGSM) is able to generate adversarial examples rapidly [13]. FGSM perturbs an image in the image space towards gradient sign directions. FGSM can be described using the following formula:

$$x_{adv} \leftarrow x + \epsilon \text{sgn}((\nabla_x L(F(x), y_{true}))$$

Herein $L$ is the loss function (a cross-entropy function is typically used to compute the loss). $F(x)$ is the softmax layer output from the model $F$. $\epsilon$ is a hyper-parameter which controls the distortion level on the crafted image. $\text{sgn}$ is the sign function. FGSM only requires gradients to be computed once. Thus, FGSM can craft large batches of adversarial examples in a very short time.

2.1.2 IGS

Iterative gradient sign (IGS) attack perturbs pixels in each iteration instead of a one-off perturbation [20]. In each round, IGS perturbs the pixels towards the gradient sign direction and clip the perturbation using a small value $\epsilon$. The adversarial example in the $i$-th iteration is stated as follows:

$$x_{adv}^i = x_{adv}^{i-1} - \text{clip}(\alpha \cdot \text{sgn}((\nabla_x L(F(x_{adv}^{i-1}), y_{true})))$$

Compared to FGSM, IGS can produce an adversarial example with a higher mis-classification confidence.

2.1.3 JSMA

Jacobian-based saliency matrix attack (JSMA) iteratively perturbs important pixels defined by the Jacobian matrix based on the model output and input features [32]. The method first calculates the forward derivatives of the neural network output with respect to the input example. The adversarial saliency map demonstrates the most influential pixels which should be perturbed. Based on two versions of the saliency map, attacker can increase the value of the influential pixels in each iteration to generate targeted adversarial examples, or decrease pixel values to get non-targeted examples.

2.1.4 Deepfool

Deepfool is able to generate adversarial examples with minimum distortion on original images [30]. The basic idea is to search for the closest decision boundary and then perturb $x$ towards the decision boundary. Deepfool iteratively perturbs $x$ until $x$ is misclassified. The modification on the image in each iteration for binary classifier is calculated as follows:

$$r_i \leftarrow -\frac{F(x)}{\|\nabla F(x)\|_2} \nabla F(x)$$

Deepfool employs the linearity assumption of the neural network to simplify the optimisation process. We use the $L_\infty$ version of Deepfool in our evaluation.

2.1.5 Carlini&Wagner $L_2$

This method has been reported to be able to make defensive distillation invalid [3]. This study explored crafting adversarial examples under three distance metrics (i.e. $L_0$, $L_2$, and $L_\infty$) and seven modified objective functions. We use Carlini&Wagner $L_2$, which is based on the $L_2$ metric, in our experiment. The method first redesigns the optimisation objective $f(x_{adv})$ as follows:
where $Z(x_{adv})$ is the output logits of the neural network, and $\kappa$ is a hyper-parameter for adjusting adversarial example confidence at the cost of enlarging the distortion on the adversarial image. Then, it adapts L-BFGS solver to solve the box-constraint problem:

$$
\min_{\delta} \|\delta\|_2^2 + c \cdot f(x + \delta) \text{s.t. } x + \delta \in [0, 1]^n
$$

Herein $x + \delta = x_{adv}$. The optimisation variable is changed to $\omega : \delta = \frac{1}{2}\tanh(\omega) + 1 - x$. According to the results, this method has achieved 100% attacking success rate on the distilled networks in a white-box setting. By changing the confidence, this method can also have targeted transferable examples to perform a black-box attack.

### 2.2 Threat model

We have three types of models for launching attack (i.e. black-box, grey-box, and white-box). In the real-world cases, the adversary normally do not have the parameters or the architecture of a deep learning model, since the model is well-protected by the service provider. Therefore, in our first threat model, we mainly assume that the adversary is in black-box setting. In prior works [3, 5], it is recommended that the robustness of a model should be evaluated by transferring adversarial examples. Otherwise, attackers can use an easy-to-attack model as a substitute to break the defence on the oracle model. Second, in some cases, the architecture and parameters of the model, as well as the defending mechanism, may be leaked to the attacker. This leads to a grey-box scenario. However, the adversary does not know the parameters of the defence. In an extreme case of white-box scenario, the adversary knows everything about the oracle model and the defence. This is a very strong assumption. Attacks launched in this way are nearly impossible to defend since the attacker can take countermeasure for defence. Therefore, we mainly consider black-box and grey-box threats in our work. We list the mentioned threat types as follows:

- **Black-box threat**: the attacker does not know the parameters and architecture of the target model. However, the attacker can train an easy-to-attack model as an substitute to craft adversarial examples, and transfer the examples onto the target classifier (i.e. the oracle). The attacker also has a training dataset which has the same distribution with the dataset used to train the oracle. To simulate the worst yet practical case that could happen in the real world, the subclass of the oracle and the oracle are trained using the same training dataset. However, the attacker knows neither the defensive mechanism, nor the exact architecture and parameters of the oracle.

- **Grey-box threat**: an attacker knows the parameters and the architecture of the oracle, as well as the adopted defending method. In this case, the attacker is able to craft adversarial examples based on the oracle instead of the substitute. However, because the parameters of the defensive mechanism are hidden from the attacker, the defence might still be effective.

In our work, we assume that the defender has no a priori knowledge pertaining to any of the following questions: 1) What attacking method will be adopted by the attacker; 2) What substitute will be used by the attacker.

### 3 DESIGN

We introduce our multi-task adversarial training method in this section. The intuition behind our defence is that: 1) Our defence framework grows an auxiliary output from the original model. It detects adversarial example by checking the pairwise relationship of the original output and the auxiliary output against the encoded label pairs. 2) The framework learns smooth decision surfaces for the original output, and steep decision surfaces for the auxiliary output. 3) The smooth manifold for the original output mitigates transferred black-box attack, while the steep manifold learned for the auxiliary output ensures that adversarial example can be detected. 4) The collaborative learning between the two outputs increases the difficulty for generating adversarial example when the attacker searches adversarial example based on the attacked model using adversarial gradients. To better encode the label pairs for the original output and the auxiliary output, we first examine the vulnerability of the learnt decision surfaces in a neural network model, such that we can later encode label pairs based on the identified vulnerable decision boundaries. Based on the encoded label pairs, we introduce our collaborative multi-task training framework for both black-box attack and grey-box attack.

#### 3.1 Vulnerable decision surface

In this section, given the original data examples of a class, we identify the corresponding adversarial target class that is usually employed by an adversary. We name the model decision boundary between the original class and the targeted class as the vulnerable decision surface.

A main constraint imposed on adversarial example is the distortion between the perturbed example and the original example. In the case of non-targeted attack, by using gradient descent to search for adversarial example, the attacker aims to maximise the error of the classification with minimal changes on the example in the feature space. Assuming we have a dataset $D$ and a model $F$ trained on $D$, the decision surfaces of $F$ will separate data points belonging to different classes in $D$. According to some previous works, we can find out that, given an example belonging to a certain class, it is easier to target the example to specific classes for the attacker [5]. This implies that the vulnerable extent of the decision surfaces varies. However, due to the high dimensionality of the features, the vulnerability cannot be measured based on simple metrics (e.g. Euclidean distance).

Herein, we adopt empirical analysis to effectively measure the vulnerability of decision boundary. Given the data examples in class $i$, we empirically find out the probability of $i$ being misclassified into another class $j$. Specifically, we use Eq.6 to estimate a confidence vector $p_i$:

$$
p_i = \frac{1}{N_i} \sum_{n=0}^{N_i} F(x_{adv})
$$

where $Z(x_{adv})$ is the output logits of the neural network, and $\kappa$ is a hyper-parameter for adjusting adversarial example confidence at the cost of enlarging the distortion on the adversarial image.
3.2 Encode labels into label pairs

Following the above method of vulnerability estimation, we encode labels of the classes in the training dataset to label pairs. To carry out the empirical analysis, we use non-targeted FGSM to generate a set of adversarial examples based on the training dataset. The generated adversarial example set $X_{adv}$ is then fed into the model to get the observation. In the classification results, for a given output label $l_{true}^{i}$ indicating class $i$, we search the corresponding pairing output label $l_{robust}^{i}$ based on minimising the following likelihood from the observation on:

$$l_{robust}^{i} = \arg \min p_i$$

Based on the observation on $p_i$, we select the least likely class being misclassified into as the robust label. Following this procedure, we encode the labeled-pair for each example in the training dataset. The encoding rules between $l_{robust}^{i}$ and $l_{true}^{i}$ are saved as a table Classmap. In the grey-box setting, this information will be used to access the credibility of an input example.

3.3 Collaborative multi-task training

We propose a collaborative multi-task training framework in this section. We consider both black-box and grey-box attacks. The proposed general training framework is depicted in Fig. 1. The framework is trained under a multi-task objective function, which is designed to maximise the divergence between the outputs of adversarial input and benign input. The training process also integrates adversarial gradients in the objective function to regularise the model to defend against Transferring attack.

3.3.1 Multi-task training for black-box attack

According to the label-pair construction method discussed in Section 3.3 and Section 3.2, we use the robust label-pairs to conduct the multi-task training [10]. Assuming the original model has the logits layer that has outputs $Z$. Our method grows another logits layer that outputs logits $Z'$ from the last hidden layer. While the softmaxed $Z$ is used to calculate the loss of the model output with the true label $y_{true}$ of the input $x$. The softmax output of $Z'$ is employed to calculate the model output loss with the robust label $y_{robust}$ when $y_{true}$ is given. We also use adversarial examples to regularise the model against adversarial inputs during the training session. The overall objective cost function $J_{obj}$ of training takes the following form:

$$J_{obj} = \alpha J(x, y_{true}) + \beta J(x_{adv}, y_{true}) + \gamma \mathcal{J}(x, x_{adv}, y_{robust})$$

wherein, $x$ is the benign example. We use adversarial gradients to regularise the model. $x_{adv}$ is the adversarial example produced by the adversarial gradient in the current step. $y_{true}$ is the ground truth label of $x$, and $y_{robust}$ is the most robust label of the current $y_{true}$. $J$ is the cross-entropy cost. $\alpha$, $\beta$, and $\gamma$ are weights adding up to 1.

The first term of the objective function decides the performance of the original model $F$ on benign examples. The second term is an adversarial term taking in the adversarial gradients to regularise the training. The last term moves the decision boundaries towards the most robust class with respect to the current class. As discussed in [13], to effectively use adversarial gradients to regularise the model training, we set $\alpha = \beta = 0.4$, and set $\gamma = 0.2$. The cost function
\[ J'(x, x_{adv}, y_{robust}) = \frac{1}{2} \{ J(x, y_{robust}) + J(x_{adv}, y_{robust}) \} \]

wherein \( J(x, y_{robust}) \) is the cross-entropy cost, and \( J(x_{adv}, y_{robust}) \) is a negative cross-entropy cost function to maximise the loss of the \( y_{robust} \) output, when the input is adversarial.

Once an example is fed into the model, the output through \( Z \) and \( Z' \) will be checked against the Classmap. The example will be recognised as adversarial if the outputs have no match in Classmap. Otherwise, it is a benign example, and the output through \( Z \) is then accepted as the classification result.

In the black-box setting, if the attacker does not know the existence of the defence, the adversarial objective function will only adjust \( x \) to produce an example that only changes the output through \( Z \) to the adversarial class. However, it cannot guarantee that the output through \( Z' \) has the correct encoding rule to the output through \( Z \) in the Classmap. The grey-box attack is then detected by our architecture.

### 3.3.2 Breaking black-box defence in grey-box settings

In the grey-box setting, the adversary does not know the existence of the defence, the adversarial objective function will only adjust \( x \) to produce an example that only changes the output through \( Z \) to the adversarial class. However, it cannot guarantee that the output through \( Z' \) has the correct mapping relationship to the output through \( Z \) in the Classmap. Therefore, it can be observed that the solver can still find an adversarial example by using a simple linear combination of the adversarial losses in the objective functions, in the grey-box setting. The detection method used for grey-box attack corrupts in this case. To solve the grey-box defence problem, we introduce a collaborative architecture into the framework.

### 3.3.3 Collaborative training for grey-box attack

We develop a framework that not only defends against transferring black-box attacks but also stops generating adversarial example using the oracle, in which the adversary has a priori knowledge of both the model and the defending strategy.

We add a gradient lock unit \( g \), between logits \( Z \) and logits \( Z' \). The \( g \) contains two fully connected layers. This architecture is not necessary, but we added it so that \( Z \) and \( Z' \) retain a non-linear relationship. The last layer of \( g \) is a multiplier, which multiplies \( Z \) with the output of \( g \) in an element-wise manner to form a new logits \( Z^* \). The input of \( g \) is \( Z' \). The architecture is then trained using a benign training dataset and regularised by a FGSM adversarial gradient, in the same training process which is used in Section 3.3.1.

The added extra layers contain no parameter to be trained; however it prolongs the path for computing adversarial gradient. After the gradient lock unit is added, the gradients of the loss function become:

\[
\frac{\partial L(Z^*(x), Z'(x), t, t')}{\partial x} = \eta_1 \frac{\partial L_1(Z^*(x), t)}{\partial x} + \eta_2 \frac{\partial L_2(Z'(x), t')}{\partial x} + \frac{\eta_1 \cdot \partial L_1(Z^*(x), t)}{\partial x} \]

\[
\frac{\partial L_1(Z^*(x), t)}{\partial x} = \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial x} + \frac{\partial L_2(Z'(x), t')}{\partial x} \]

\[
\frac{\partial L_2(Z'(x), t')}{\partial x} = \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial x} + \frac{\partial L_1(Z^*(x), t)}{\partial x} \]

\[
\{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial x} \}
\]

\[
\frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \]

\[
\frac{\partial L_2(Z'(x), t') \partial Z'(x)}{\partial x} \]

\[
\{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \}
\]

\[
\frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \]

\[
\frac{\partial L_2(Z'(x), t') \partial Z'(x)}{\partial x} \]

\[
\{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \}
\]

\[
\frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \]

\[
\frac{\partial L_2(Z'(x), t') \partial Z'(x)}{\partial x} \]

\[
\{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t) \partial Z'(x)}{\partial x} \}
\]

It can be seen in the second term that, the back propagation from \( Z' \) to \( x \) and the back propagation from \( Z^* \) to \( x \) are mutually affected by each other. The gradient update is calculated based on \( Z \) and \( Z' \) in the previous step, but it does not take the updates in the current step into consideration. Therefore, it is difficult for the solver to find a converged solution on \( x \). For the gradient based solver, it is hard to find a valid adversarial example based on this architecture.

When the model is put into use, the outputs through \( Z^* \) and \( Z' \) will then be checked against the Classmap from Section 3.3.2. If the outputs match the encoding relationship in the Classmap, the output is credible. Otherwise, the input example is identified as an adversarial example. Therefore, our defending method is to detect and reject adversarial examples. Furthermore, the regularised model output from \( Z^* \) can complement the detection module once there is a mis-detection.
4 Evaluation

In this section, we present the evaluation on our proposed method on defending against the state-of-the-art attacking methods. We first evaluate the robustness of our defence against the FGSM, IGS, JSMA, and Deepfool, in black-box setting. Then we evaluate the detection performance against high-confidence C&W attack in black-box and grey-box settings. We ran our experiments on a Windows server with CUDA supported 11GB GPU memory, Intel i7 processor, and 32G RAM. In the training of our multi-task model and the evaluation of our defence against the fast gradient based attack, we use our implementation of FGSM. For Carlini&Wagner attack, we adopt the implementation from their paper [5]. For other attacks, we employ the implementations provided in Foolbox [35].

4.1 Model, data, and attack

We have implemented one convolutional neural networks architecture as an oracle. To simplify the evaluation, we used the same architecture for the CIFAR10 oracle, denoted as \(O_{CIFAR10}\) and the MNIST oracle, denoted as \(O_{MNIST}\). The architectures of the oracle model are depicted in Table 1.

| Layer Type       | Oracles | \(S_{CIFAR10}\) | \(S_{MNIST}\) |
|------------------|---------|----------------|--------------|
| Conv+ReLU 3×3×32 | 3×3×32 | 3×3×32         | 3×3×32       |
| Conv+ReLU 3×3×32 | 3×3×32 | 3×3×32         | 3×3×32       |
| Max Pooling      | 2×2    | 2×2            | 2×2          |
| Dropout          | 0.2    | -              | -            |
| Conv+ReLU 3×3×64 | 3×3×64 | 3×3×64         | 3×3×64       |
| Conv+ReLU 3×3×64 | 3×3×64 | 3×3×64         | 3×3×64       |
| Max Pooling      | 2×2    | 2×2            | 2×2          |
| Dropout          | 0.2    | -              | -            |
| Conv+ReLU 3×3×128| 3×3×128| -              | -            |
| Conv+ReLU 3×3×128| 3×3×128| -              | -            |
| Max Pooling      | 2×2    | -              | -            |
| Dropout          | 0.2    | -              | -            |
| Fully Connected  | 512    | 256            | 200          |
| Fully Connected  | -      | 256            | 200          |
| Dropout          | 0.2    | -              | -            |
| Softmax          | 10     | 10             | 10           |

Evaluation Datasets

| Dataset | FGSM | IGS | Deepfool | C&W L2 |
|---------|------|-----|----------|--------|
| MNIST   | 10000 | 1000 | 1000     | 1000   |
| CIFAR10 | 10000 | 1000 | 1000     | 1000   |

4.2 Defending low-confidence adversarial examples

We present the performance of our defence against low-confidence transferring black-box adversarial examples in this section. First, given \(S_{MNIST}\) and \(S_{CIFAR10}\), we use the above four attacks to craft adversarial sets whose sizes are listed in Table 2. Then, we feed the mentioned adversarial test sets into \(O_{MNIST}\) and \(O_{CIFAR10}\) respectively. We adopt a non-targeted version of each of the above attacks since the non-targeted adversarial examples are much better in terms of transferring between models.

Robustness towards adversarial examples is a critical criterion to be assessed for the protected model. For a black-box attack, we measured the robustness by investigating the performance of our defence on tackling the typical low-confidence black-box adversarial examples, which are near the model decision boundaries. We feed adversarial test sets into the protected model, and then we check the classification accuracy of the label output through \(Z^*\). The results of the classification accuracy are demonstrated in Table 3. It shows that the accuracy of CIFAR10 classification is limited by the performance of the oracle \(O_{CIFAR10}\). When we measured the accuracy of the output label of the adversarial examples against the predicted labels of the benign examples, the performances on the CIFAR10 task matches that of the MNIST task (Table 3).

It can be found that, in all cases, our method has improved the classification accuracy of the oracle, except for C&W attack. The reason is that C&W attack can still successfully bypass the black-box defence because the confidence of the generated example is set to a very high black-box distilled network in the original paper [5], producing high-confidence adversarial examples. Later, we also evaluate the performances of our defence under different \(k\) values.
value (i.e. $\kappa = 40$). The nature of the black-box defence is to regularise the position of the decision boundary of the model, such that adversarial examples near the decision boundary will become invalid. However, the defence can be easily bypassed if we can adjust the level of the perturbation or the adversarial confidence to become higher, to which C&W is fully capable. This vulnerability also suggests that we need a more effective defence for C&W attack. Later on, we presented the results of our detection-based defence which tackles the C&W attack.

4.3 Defending high-confidence adversarial examples

We evaluate our defence against high-confidence adversarial examples crafted by C&W attack in this section. The attacking confidence of adversarial examples from C&W attack is changeable through adjusting the hyper-parameter $\kappa$ in the adversarial objective function. Large $\kappa$ value will lead to producing a high-confidence adversarial example. Therefore, C&W attack can even achieve remarkable attacking performance through transferring attack in black-box setting. Therefore, our defence relies on detection mechanism to tackle high-confidence C&W attack. We evaluate the adversarial example detection performance against C&W examples crafted using different $\kappa$ values in the black-box setting.

4.3.1 Defending transferring C&W adversarial examples

In a black-box setting, the high-confidence C&W examples can transfer from the substitute to the oracle. We test the defence performance of proactively invalidating adversarial examples.

To demonstrate the performance, we measure the success rates of the transferring attacks which change the output results from the $Z^*$. The successful transfer rate from the substitutes to the oracles are plotted in Fig. 2. When $\kappa$ is set to a high value, the adversarial example can still successfully break our black-box version defence, since the nature of our black-box defence is similar to the adversarial training. It regularises the decision boundaries of the original oracle, to invalidate the adversarial examples near the decision boundaries. But unfortunately, the high-confidence examples are usually far away from the decision boundaries. Therefore, we defend the high-confidence adversarial examples by reactively detecting them. However, compared to the unprotected oracle, our proactive defence can keep the attacking success rate stays below 20% when $\kappa$ is less than 10.

4.3.2 Detecting high-confidence adversarial examples

In this section, we evaluated the performance of our method on detecting the C&W attack in a normal black-box setting and in the worst case of black-box settings. For each $\kappa$ value, we craft 1,000 adversarial examples. We then mix 1,000 benign examples into each group of 1,000 adversarial examples to form the evaluation datasets for different $\kappa$ values. We measure the precision and the recall of our defence on detecting the examples. The precision is calculated as the percentage of the genuine adversarial examples in the examples detected as adversarial. The recall is the percentage of adversarial examples detected from the set of adversarial examples.

In a normal black-box setting, the substitute model is not exactly the same as the oracle model. In this case, we employed the substitute $S_{MNIST}$ and $S_{CIFAR10}$ to generate C&W examples. We evaluated the precision and the recall on detecting black-box adversarial examples crafted under different $\kappa$ values. The precision values and the recall values are plotted in Fig. 2. Our defence achieved above 95% detection precision on $MNIST$ adversarial examples and above 75% detection precision on $CIFAR10$ adversarial examples. At the same time, the detection recall values on detecting $MNIST$ adversarial examples are above 80%. And the recall values on detecting $CIFAR10$ adversarial examples are above 95%. Moreover, the precision and recall are robust to the changing $\kappa$ value.

To further increase the difficulty for defence, we evaluated our detection defence in the worst case of black-box settings. In the worst case, the attacker has a substitute model whose parameters are exactly the same as that of the oracle model. In this case, the adversarial examples cannot
Fig. 3. The precision and recall of detecting C&W adversarial examples in a normal black-box setting.

Fig. 4. The precision and the recall of detecting C&W adversarial examples in the worst case of black-box settings.

be invalidated by proactive defence. We set the parameters of the substitute to be the same as that of the oracle model with proactive defence (without the $Z'$ output), and $\kappa$ was varied from 0 to 40 to generate adversarial examples based on this substitute.

The precision and recall values are plotted in Fig. 4. It can be observed that our defence method still has achieved high precision and recall in the worst case of black-box settings. The detection performance is robust to the varying attacking confidence.

4.4 Tackling grey-box attacks

We evaluate the performance of our defence against grey-box attacks. We evaluate our defence against the C&W attack in this section since C&W can produce the best attacking result, and is flexible in searching adversarial examples due to the tunable $\kappa$ hyper-parameter. The adversarial examples used in the evaluation are crafted based on the linearly-combined adversarial loss functions mentioned in Section 3.3.2. To measure the grey-box defence, we sweep the value of $\kappa$ from 0 to 40, and then we examine the rate of successful adversarial image generation given 100 MNIST images under each $\kappa$ value. We record the successful generation rate of targeted and non-targeted MNIST adversarial examples based on the defended oracle. For targeted attack, we randomly set the target label during the generation process. The generation rates are recorded in Table 4.

It can be found that the rate of finding valid adversarial examples is kept at a reasonably low level, especially when the values of $\kappa$ are high. Some of the generated C&W adversarial examples with/without our defence are displayed in the appendix in our supplementary file.

4.5 Trade-off on benign examples

We also evaluate the trade-off on normal example classification and the false positive detection rate when the input examples are benign.

First, we evaluate the accuracy of the classification results output through $Z^*$, after our protections are applied on the oracle. We use both CIFAR10 dataset and MNIST dataset in the evaluation. For each dataset, we draw 10,000 benign examples and feed them into the defended oracles. The results are in Table 5. The classification accuracy of the protected model is the accuracy of the output classification label through $Z^*$, given the set of correctly identified normal examples. It can be found that for the MNIST oracle, our defence had no decrease on the classification accuracy. On CIFAR10 task, our defence decreases the accuracy by 2.1%. The trade-offs are within the acceptable range considering the improvements on defending adversarial examples.

Next, we assess the mis-detection rate of our defence. We feed 10,000 benign MNIST examples and 10,000 CIFAR10 benign examples into the corresponding defended oracles to check how many of them are incorrectly recognised as adversarial examples. Our method has achieved 0.09% mis-detection rate on MNIST dataset. For CIFAR10 dataset, our mis-detection rate is 3.92%. Our detector had a very limited mis-detection rate for both datasets. Hence, our detection-based defence can accurately separate adversarial examples from benign examples.

5 JUSTIFICATION OF THE DEFENCE

In this section, we present the justification on the mechanism of our defence on both black-box and C&W attacks.

5.1 Classmap evaluation

We evaluate the generalisation of the Classmap extracted from different attacking examples. We generate Classmaps from FGSM, IGS, Deepfool, and C&W example sets. The heat maps of the Classmaps are displayed in Fig. 5. All the Classmaps share a similar pattern. This means the Classmaps learned from the first-order FGSM examples can generalise to other types of adversarial example. The generalisation of the Classmap ensures that the detection mechanism can generalise across different attacks.
maximised at $l$, the gradient updates $l$ into the robust label example whose original label is $l$, it is unlikely to classify it to the desired label. Therefore, the required total efforts for changing the combined output becomes higher.

From the perspective of a classifier decision boundary, our multi-task training method has also increased the robustness of the model against black-box examples. The robust label regularisation term actually moves the decision boundary towards the robust class (refer to Fig. 6). Compared to the traditional adversarial training, which tunes the decision boundary depending merely on the generated adversarial data points, our regularisation further enhances the robustness of the model towards nearby adversarial examples.

5.3 Defending the C&W attack

We provide a brief analysis on why our method can defend the C&W attack here. A C&W attack mainly relies on the modified objective function to search for adversarial examples, given the logits from the model. By observing the objective function $f(x_{adv}) = \max \{ \max_i (Z(x_{adv})_i : i \neq l) - Z(x_{adv})_l \} - \kappa$, we can find out that the objective is actually to increase the value of the logits corresponding to the desired $l$ class, until the difference between $l$ and the second-to-the-largest class reaches the upper bound defined by $\kappa$. The optimisation process can be interpreted as adjusting the input pixels along the direction of the gradient that maximises the logits difference.

In a black-box setting, when we adopt the collaborative multi-task training as a defence, the model actually modifies the output logits to have high outputs not only on the position corresponding to the ground truth class, but also on the position corresponding to the robust class of the current ground truth. In the black-box setting, the defence is hidden from the attacker. The attacker crafts adversarial examples
solely based on the oracle model without the robust logits branch. Hence, the adversarial objective function is not a linear combination of \( L(Z(x), t) \) and \( L(Z(x), t') \), but a single loss \( L(Z(x), t) \). The crafted adversarial example can only modify the output from \( Z \) to the adversarial target \( t \), but the output through \( Z' \) is not guaranteed to be the corresponding robust label of \( t \) in the Classmap.

In a grey-box setting, the defending strategy is exposed to the adversary. Therefore, the adversary can first query the oracle model by feeding a certain volume of data examples to find the approximate Classmap. Then, the attacker can set the adversarial objective to a linear combination form mentioned in Section 3.3.2 to successfully bypass the defence. However, after the gradient lock unit is added, the path for back-propagating adversarial loss becomes longer. The solver cannot effectively adjust the input pixel values of the original example due to vanishing gradients. Second, the two logits outputs (i.e. \( Z \) and \( Z' \)) are related to each other in the gradient back-propagation. Based on the last step of gradient update, the adjustments made on \( Z \) and \( Z' \) become:

\[
\delta_z = \eta_1 \cdot \frac{\partial L_1(Z^*(x), t) \partial Z^*(x) \partial Z(x)}{\partial Z^*(x)} \cdot \partial x + \\
\{ \eta_1 \frac{\partial L_1(Z^*(x), t) \partial Z^*(x)}{\partial Z^*(x)} + \eta_2 \frac{\partial L_2(Z'(x), t') \partial Z'(x)}{\partial Z'(x)} \} \cdot \partial x
\]

(14)

However, the updates on \( Z \) and \( Z' \) are based on the previous values of \( Z' \) and \( Z \). Eventually, the inputs will actually decay the progress made by the gradient updates. Therefore, it becomes difficult for the optimisation solver to find a satisfactory solution that can generate adversarial label and the correct paired robust label of the adversarial label.

6 Related work

6.1 Attacking methods

Based on the method of generating adversarial example, current attacking methods can be divided into optimisation based attack and forward derivative based attack. Optimisation based attack sets an adversarial objective function and optimises the example feature to achieve the optimum. FGSM uses a single gradient descent step to slightly perturbed the input example \[13\]. Subsequently, an iterative gradient sign based method was proposed in \[20\]. L-BFGS based attack optimises a box-constrained adversarial objective to find adversarial example \[38\]. Furthermore, a method named DeepFool is proposed. DeepFool iteratively finds the minimum perturbations of images \[30\]. Last but not the least, Carlini\&Wagner attack is proposed to optimise a specially designed adversarial loss to craft adversarial example with changeable attacking confidence \[5\]. For the forward derivative based attack, it is based on the Jacobian of the model output with respect to each input feature. JSMA was first proposed by Papernot et al. \[33\]. This method modifies the most critical pixel according to the saliency map defined by the model Jacobian.

To make adversarial examples more imperceptible for human beings, there are methods using different distortion metrics. For example, an image can be perturbed in HSV color space to generate adversarial example \[18\]. Additionally, an image can be rotated to be adversarial \[9\]. Beyond the mentioned attacks, there are attacks modified towards different systems. Adversarial examples have been designed towards applications such as object detection and segmentation \[11, 24, 40\], reading comprehension \[19\], text classification \[8\], and malware detection \[15\].

6.2 Defensive methods

Defensive methods can be broadly categorised into proactive defence and reactive defence. In the category of proactive defence, a model is trained to be robust towards adversarial examples. The first defensive mechanism employs simple adversarial training to enhance the robustness of neural nets \[38\]. Other adversarial training based defensive methods are published subsequently \[25, 39\]. Gu et al. proposed a contractive neural network. The contractive net solves the adversarial example problem by introducing a smooth penalty on the neural network model based on Lipschitz condition \[16\]. Defensive distillation distils training dataset with soft labels from a first neural net under a modified softmax layer to train a second identical neural net \[34\]. The soft labels enable more terms of the loss function to be computed in the back-propagation stage. In the mean time, the modified softmax function ensures amplified output from the logits layer. Parseval network is proposed to constrain the weight matrices of the convolutional layers and dense layers, such that the network becomes insensitive to adversarial perturbation \[36\]. Furthermore, few papers employ robust optimisation technique to train robust model towards adversarial examples \[27, 56\]. Additionally, a de-
fence enhances the model robustness by pruning activations from network [7].

For most of the reactive defence methods, a second model is adopted to detect examples with adversarial perturbation. For instance, Metzen et al. attached detectors on the original model and trained it with adversarial example [29]. Another method employs support vector machine to classify the output from the high-level neural network layer [23]. A statistical method is proposed to detect adversarial example batch [14]. Lately, there is a detection method detect adversarial example based on the local intrinsic dimensionality of layer representations [26]. For other detection methods, they align with the idea of observing the changes of an example after applying a certain transformation on it [28]. For example, MagNet relies on autoencoder to reconstruct adversarial examples to normal example, and detect adversarial example based on the reconstruction error and output probability divergence [28].

7 DISCUSSION

Current attacks and defences are largely considered as threat-model-dependent. For instance, as the state-of-the-art defence, in defensive distillation is working in a grey-box scenario, the attacker knows the parameters of the distilled network, but does not know that the network is trained with defensive distillation. Thus, when the attacker crafts adversarial examples based on the distilled network, the removed temperature from the softmax function will lead to vanishing gradients while solving the adversarial objective function. However, once the attacker understands that the network is distilled, the defence can be easily bypassed by adding a temperature into the softmax function while solving the adversarial objective function. Moreover, defensive distillation is invalid towards a black-box attack. Adversarial examples crafted by an easy-to-attack substitute could still be valid on a distilled network.

As the attacking methods based on gradient, except C&W attack, are unable to attack distilled network in a grey-box setting, a black-box attack launched using these methods is more practical and harmful. This is why we evaluated the performance of our defence in a black-box setting. As a special case, the C&W attack claims that it can break the defensive distillation in the white-box setting, since it searches adversarial examples based on the logits instead of the softmax outputs. Hence, the C&W attack can bypass the vanishing gradient mechanism introduced by defensive distillation on the softmax layer. However, having access to the logits itself is a very strong assumption, which actually defines a white-box attack. However, a substitute can be used together with high confidence C&W attack to bypass distilled network in a black-box setting.

There are many possible ways to enhance our method. First, our method can be further improved by incorporating randomness into the defence architecture. For example, switching some of the model parameters based on a set of pre-trained parameters might further increase the security performance of the defence. Second, attacks employ forward derivatives (e.g. JSMA [32]) in grey-box setting can still effectively find adversarial examples, since our defence essentially tackles gradient-based adversarial example searching. However, our defence is still functional towards black-box examples due to the regularised training process. At last, our grey-box defence is based on gradient masking. This can be improved in our future work.

8 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel defence against black-box attack and grey-box attack on deep neural networks. Importantly, our method is able to protect DNN classifiers from the Carlini & Wagner attack, which is the most advanced and relatively practical attack to date. We also demonstrated through experiments that the performance trade-off on normal example classification brought by our defence is also acceptable.

As the shortcomings of our approach, first, the quality of the Classmap will directly affect the performance of the detection rate for adversarial examples. We use a large volume of non-targeted adversarial examples to approximate the encoding rules between class labels. However, the quality is affected by the employed attacking method. Second, introducing randomness into the defence can further increase the defence performance. Moreover, there could be better option than multiplying the logits in the gradient lock unit. These problems will be addressed in our future work.

DNN has achieved the state-of-the-art performance in various tasks. However, compared to traditional machine learning approaches, DNN also provides a practical strategy for crafting adversarial examples, since the back-propagation algorithm of DNN can be exploited by an adversary as an effective pathway for searching adversarial examples.

Current attacks and defences have not yet been exclusively applied to the real-world systems built on DNN. Previous studies have made attempts to attack online deep learning service providers, such as Clarifi [22], Amazon Machine Learning, MetaMind, and Google cloud prediction API [31]. However, there is no reported instance of attacking classifier embedded inside complex systems, such as Nvidia Drive PX2. Successful attack on those systems might require much more sophisticated pipeline of exploiting vulnerability in system protocols, acquiring data stream, and crafting/injecting adversarial examples. However, once the pipeline is built, the potential damage it can deal with would be fatal. This could be another direction for the future works.

REFERENCES

[1] A. Athalye and I. Sutskever. Synthesizing robust adversarial examples. arXiv preprint arXiv:1707.07397, 2017.
[2] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. arXiv preprint arXiv:1712.09665, 2017.
[3] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. arXiv preprint arXiv:1705.07263, 2017.
[4] N. Carlini and D. Wagner. Magnet and “efficient defenses against adversarial attacks” are not robust to adversarial examples. arXiv preprint arXiv:1711.08478, 2017.
[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP). IEEE, may 2017.
[6] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863, 2017.

[7] C. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaiti, A. Khanna, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*, 2018.

[8] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of ACL*, 2018.

[9] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[10] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[11] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *Computer Science*, 2014.

[14] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[15] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[16] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *Computer Science*, 2015.

[17] W. He, J. Wei, X. Chen, N. Carlini, and D. Song. Adversarial example defenses: ensembles of weak defenses are not strong. In *Proceedings of the 11th USENIX Conference on Offensive Technologies*, pages 15–15. USENIX Association, 2017.

[18] H. Hosseini and R. Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619, 2018.

[19] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

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

[21] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[22] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[23] J. Lu, T. Israaron, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *arXiv preprint arXiv:1704.00103*, 2017.

[24] J. Lu, H. Sibai, and E. Fabry. Adversarial examples that fool detectors. *arXiv preprint arXiv:1712.02494*, 2017.

[25] C. Lyu, K. Huang, and H.-N. Liang. A unified gradient regularization family for adversarial examples. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 301–309. IEEE, 2015.

[26] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Witjewickrema, M. E. Houle, G. Schoenebeck, D. Song, and J. Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[27] 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.

[28] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. *arXiv preprint arXiv:1705.09064*, 2017.

[29] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. In *The 5th IEEE International Conference on Learning Representation*, 2017.

[30] S. M. Moosavidezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[31] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[32] N. Papernot, P. McDaniel, S. Jha, M. Fredriksson, 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)*, pages 372–387. IEEE, 2016.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredriksson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *arXiv preprint arXiv:1704.01431*, 2017.

[34] U. Shaham, Y. Yamada, and S. Nogahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 2018.

[35] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[36] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *Computer Science*, 2013.

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

[38] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision*. IEEE, 2017.

[39] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

**APPENDIX**
Fig. 7. The adversarial MNIST images generated using non-targeted C&W attack when \( \kappa = 0 \). The first row and the third row are the original images. The second row is the generated adversarial image based on the original model (classification results are as the labels below the second row). The fourth row is the failed generation after applying our defence on the model.

Fig. 8. The adversarial MNIST images generated using non-targeted C&W attack when \( \kappa = 40 \). The first row and the third row are the original images. The second row is the generated adversarial image based on the original model (classification results are as the labels below the second row). The fourth row is the failed generation after applying our defence on the model.

Fig. 9. The adversarial MNIST images generated using targeted C&W attack when \( \kappa = 0 \). Images in the leftmost column are the original images. Images in the first row are the targeted adversarial image generated based on the original model. Images in the second row are the generated targeted adversarial image after applying our defence on the model. Classification results are as the labels below.

Fig. 10. The adversarial MNIST images generated using targeted C&W attack when \( \kappa = 40 \). Images in the leftmost column are the original images. Images in the first row are the targeted adversarial image generated based on the original model. Images in the second row are the generated targeted adversarial image after applying our defence on the model. Classification results are as the labels below.