

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

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

Abstract—Deep neural network (DNN) has shown an impressive performance on hard perceptual problems. However, researchers found that DNN-based systems are vulnerable to adversarial examples that contain specially crafted humans-imperceptible perturbations. Such perturbations cause DNN-based systems to mis-classify the adversarial examples, with potentially disastrous consequences in applications where the safety or security is crucial.

To address this problem, this paper proposes a novel defensive framework based on a collaborative multi-task training. The proposed defence mechanism first incorporates specific label pairs into adversarial training process to enhance the model robustness in a black-box setting. Then a novel collaborative multi-task training framework is proposed to construct a detector that identifies adversarial examples based on the pairwise relationship of the label pairs. The detector can identify and reject high confidence adversarial examples that bypass the black-box defence. The model, whose robustness has been enhanced, works reciprocally with the detector on the false-negative adversarial examples. Importantly, the proposed collaborative architecture can prevent the adversary from finding valid adversarial examples in a near-white-box setting. We evaluate our defence against four state-of-the-art attacks on *MNIST* and *CIFAR10* datasets. Our defence method increases the classification accuracy of black-box adversarial examples up to 96.3%, and detects up to 98.7% of the high confidence adversarial examples, while only decreases the accuracy of benign example classification by 2.1% on the *CIFAR10* dataset.

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

I. INTRODUCTION

DEEP neural networks (DNNs) have achieved remarkable performances on tasks such as computer vision, natural language processing and data generation. However, DNNs are vulnerable towards adversarial attacks which exploit imperceptibly perturbed examples (refer Fig.1 for details) to fool the neural networks [23].

The adversarial attacks on DNN can be catastrophic and are not ad hoc. For instance, as a prominent type of DNNs, convolutional neural networks (CNN) are widely adopted in applications such as hand-writing recognition [17], face

detection [10], and autonomous vehicle [5]. Therefore, adversarial examples for different CNN-based systems can be crafted under the same methods. Attacks on CNN can be replicated on different CNN-based systems in a costless way. The universal existence of adversarial examples in different scenarios fatally endangers the human users, such as causing accident originated by faulty object detection in autonomous vehicles. 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.

Attacking on DNN-based applications can be performed under different prerequisites for the attacker. The first situation is the white-box attack. Once an attacker has the access to the architecture and parameters of the attacked model, he/she can craft adversarial examples based on the attacked model. Second, a black-box attack can be performed without knowing the exact model architecture, model parameters, and even the training dataset [16]. Crafting adversarial examples in the black-box setting requires using a DNN substitute. The crafted examples are valid for the attacked model due to the transferability of the adversarial examples [11]. Current applications of DNN largely rely on a few representative DNN models (e.g. VGG [21], GoogleNet [22], and ResNet [8] for computer vision missions), which makes it a relatively easy task for the adversary to guess the substitutes.

The adversary can either specify a desired classification result (i.e. targeted attack), or just go for mis-classification without specific target classes (i.e. non-targeted attack). The transferability of adversarial examples is more significantly demonstrated in non-targeted attacks rather than targeted attacks [11]. Therefore, the non-targeted attack is more practical in the black-box setting, which is the most possible situation the attackers fall into. However, high-confidence adversarial examples created by strong attacking methods can also transfer between models [3]. Moreover, these high-confidence adversarial examples are able to bypass defensively distilled network [19], which was considered as the most effective defence in both white-box and black-box settings.

A. Motivation

Adversarial example attack is using a perturbed input example to manipulate the output from a machine learning model. Attacks can be categorised into three categories based

D. Wang, C. Li, S. Wen, and Y. Xiang are with School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia, e-mail: {swen, yxiang}@swin.edu.au.

S. Nepal is with Data 61, CSIRO, Australia, email: Surya.Nepal@data61.csiro.au.

D. Wang is also with Data 61, CSIRO, Australia, email: derek1.wang@data61.csiro.au

Corresponding author: Sheng Wen.

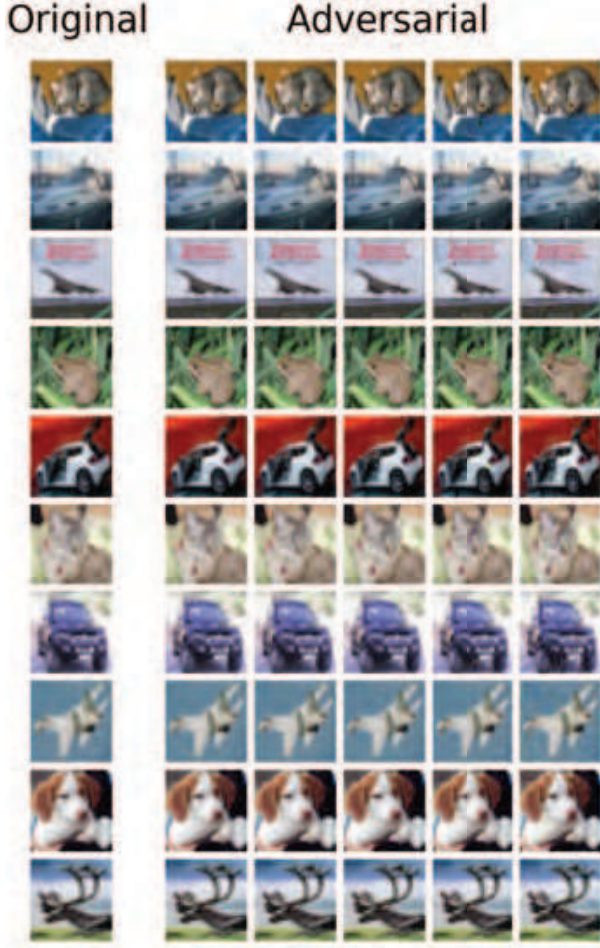


Fig. 1. Adversarial examples of Cifar10 images. These examples are crafted by five attacking methods. First, the adversarial images are all mis-classified by the CNN classifier; Second, the differences between adversarial examples and original examples are imperceptible. From the 2nd column to the 6th column, the attacks are: 1) Carlini&Wagner l_2 [3], 2) Deepfool [15], 3) Iterative Gradient Sign (IGS) [9], 4) Jacobian-based Saliency Map Attack (JSMA) [17], and 5) Fast Gradient Sign method (FGSM) [6].

on the observed information by the attacker: 1) black-box attacks, 2) grey-box attacks, 3) white-box or nearly white-box attacks (refer to Section II-B for details). Currently, there are a series of countermeasures proposed. However, they all have limitations. First, the adversarial training based defensive methods regularise the model parameter to defend black-box adversarial examples. However, it is invalid towards grey-box and white-box attacks. Second, the methods working in grey-box scenario (e.g. defensive distillation) utilise gradient masking [16] to block the searching of adversarial examples by optimising the adversarial objective (i.e. gradient-based). However, this type of defence is invalid towards attacks that exploit input feature sensitivity (e.g. JSMA), and it cannot defend black-box attack. What makes it more frustrating is that, defensive distillation is breakable in both grey-box setting and black-box setting under Carlini&Wagner attack, which is actually gradient-based. The recently proposed MagNet can detect adversarial examples and reform adversarial examples to benign examples [13]. However, it is vulnerable towards transfer attack directed from substitute autoencoders [2]. To

date, there is no valid method which actually increases model that is robustness to both transfer attack and grey-box/white-box attacks.

B. Our Proposed Method & Contributions

In this paper, we propose a well-rounded defence that not only increases the robustness of neural networks towards transferring attack, but also detects high confidence black-box/grey-box adversarial examples at high accuracy. Moreover, our proposed defence can prevent an adversary from finding adversarial example in a near-white-box setting. 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 all the black-box attack, the grey-box attack, and even the near-white-box attack. The main contributions of the paper are summarized as follows:

- We introduced a collaborative multi-task training framework to invalidate/detect adversarial examples. This framework uses data manifold information to build a pair-wised rule to detect adversarial attacks;
- We carry out both empirical and theoretical studies to evaluate the proposed framework. The experimental results demonstrate the capabilities of the proposed defence framework, more specifically: 1) the framework could prevent the adversary from searching valid adversarial examples in near-white-box settings; and 2) it can also detect or invalidate adversarial examples crafted in grey-box/black-box settings.

The paper is organised as follows: Section II describes the state-of-the-art attacks and clarifies the problem statement and our contributions. Section III presents our detailed approach. Section IV presents the evaluation of our approach. Section V provides an analysis on the mechanism of the defence. Section VI presents a conclusion on the existing attacks and the defensive methods. Section VII discusses the remaining unsolved problems of the existing attacks and defences, as well as the possible further improvements of the defence. Section VIII summaries the paper, and proposes the future works.

II. PRIMER

A. Adversarial attacks

We first introduce the state-of-the-art attacks in the field. Suppose 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} .

1) *FGSM*: Fast gradient sign method (FGSM) is able to generate adversarial examples rapidly [6]. FGSM perturbs an image in the image space towards gradient sign directions. FGSM can be described using the following formula:

$$x_{adv} \leftarrow x + \epsilon \text{sgn}(\nabla_x L(F(x), y_{true})) \quad (1)$$

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 . ϵ is a hyper-parameter which

controls the distortion level on the crafted image. 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) *IGS*: Iterative gradient sign (IGS) attack perturbs pixels in each iteration instead of a one-off perturbation [9]. In each round, IGS perturbs the pixels towards the gradient sign direction and clip the perturbation using a small value ϵ . The adversarial example in the i -th iteration is stated as follows:

$$x_{adv}^i = x_{adv}^{i-1} - clip_{\epsilon}(\alpha \cdot sgn(\nabla_x L(F(x_{adv}^{i-1}), y_{true}))) \quad (2)$$

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

3) *Deepfool*: Deepfool is able to generate an adversarial example with a minimum distortion on the original image [15]. 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 achieved follows:

$$r_i \leftarrow -\frac{F(x)}{\|\nabla F(x)\|_2^2} \nabla F(x) \quad (3)$$

Deepfool employs the linearity assumption of the neural network to simplify the optimisation process. We use the L_{∞} version of Deepfool in our evaluation.

4) *Carlini&Wagner L_2* : This method is reportedly able to make defensive distillation invalid [3]. This study has explored crafting adversarial examples under three distance metrics (i.e. L_0 , L_2 , and L_{∞}) 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:

$$f(x_{adv}) = \max(\max\{Z(x_{adv})_i : i \neq l\} - Z(x_{adv})_l, -\kappa) \quad (4)$$

where $Z(x_{adv})$ is the output logits of the neural network, and κ 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:

$$\min_{\delta} \|\delta\|_2^2 + c \cdot f(x + \delta) \text{ s.t. } x + \delta \in [0, 1]^n \quad (5)$$

Herein $x + \delta = x_{adv}$. The optimization 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.

B. Threat model

We have four types of threat from an adversary. In the real-world cases, the adversary normally do not have the parameters or the architecture of the deep learning model, since the model is well-protected by the service provider. Thus, in our

first threat model, we mainly assume that the adversary is in black-box setting. From the previous researches [1], [3], it is recommended that the robustness of a model should be evaluated by transferred 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 model parameters and architecture may be leaked to the attacker. However, the defence mechanism is still hidden from the attacker. This leads to a gray-box attack. Next, if the adversary has both the model and the defensive method, the threat from the adversary becomes a white-box threat. Finally, due to the reason that directly defending a white-box threat is nearly impossible, we define a new threat model named near-white-box threat, to examine the defence in the extreme cases that the adversary has detailed knowledge about the model and the defence, but the parameters and the architecture of the model and the defence is unchangeable. We list the mentioned four threat types in the following:

- **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, the substitute 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**: the adversary knows the parameters and the architecture of the oracle. In this case, the attacker is able to craft adversarial examples based on the oracle instead of the substitute. However, the attacker is blinded from the defensive mechanism.
- **White-box threat**: the adversary knows everything about the oracle 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.
- **Near-white-box threat** In white-box setting, the attacker knows both the model and the defensive method. It is nearly impossible to defend the adversary in such a situation. Thus, we define another threat model, in which the attacker knows the model and the defence, but the attack cannot make change to the architecture or the model parameters or the defence.

We assume that the defender has no prediction about any of the following information: 1) What attacking method will be adopted by an adversary; 2) What substitute will be used by an attacker.

III. 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 defended model, the defence detects adversarial example

by examine the pairwise relationship of the outputs. 2) The framework learns a smooth manifold near the normal output class, and a steep manifold near the auxiliary output class. 3) The smooth manifold for the normal 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 construct the training label for the auxiliary output, we first examine the existence of vulnerable decision boundary given a neural network model and identify robust label pairs of the dataset. Next, we propose the multi-task training framework for both black-box attack and grey-box attack.

A. Vulnerable decision boundary

In this section, we identify vulnerable class pairs, which is the pairs of classes that are usually employed by adversary. A main constraint imposed on adversarial example is the distance 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 is aiming to maximise the loss of the classification to the ground truth with the minimum changes of the inputs in the feature space. Thus, we assume that for a given dataset D and a model F trained on D , the decision boundaries of F will separate data points belonging to different classes in D , in the ideal situation. Therefore, since the distances among different classes of examples are different, non-targeted adversarial examples will be more likely to be classified into another class which is the closest to the original class in the feature space.

Our aim in this step is that, for each class i , we statistically find the probability/confidence of i being misclassified into another class j . Specifically, we use Eq.6 to get the confidence vector as p_i , which is actually used as a second label of class i in the training session:

$$p_i = \frac{1}{N_i} \sum_{n=0}^{N_i} F(x_{adv_i}) \quad (6)$$

Herein, x_{adv_i} is an adversarial examples which originally belong to class i . N_i is the number of examples in the class i . $F(x_{adv_i})$ is an output confidence vector from the model. p_i yields the overall confidence of the examples in i being classified into other classes. Therefore, p_i indicates the manifold information between each pair of classes.

B. Robust class pair identification

In this step, we use a non-targeted attack to examine the vulnerable boundaries of the classifier in a given dataset. Give an input image, we are then able to propose the least possible output distribution that might be exploited by the attacker. First, we produced a set of adversarial examples using FGSM. Then, from the prediction results from the produced adversarial examples, we revealed the vulnerable boundaries and estimate the robust class pairs.

FGSM is able to generate adversarial examples rapidly [6]. FGSM perturbs an image in the image space towards the gradient sign directions. FGSM can be described using the following formula:

$$x_{adv} \leftarrow x + \epsilon \text{sgn}(\nabla_x J(F(x), y_{true})) \quad (7)$$

Herein J is the cost function (cross-entropy function is typically used as the loss function during computing the cost). sgn is the sign function. $F(x)$ is the softmax layer output from model F . ϵ is a hyper-parameter which controls the distortion level on the crafted image. FGSM only requires one back-propagation process to generate examples. Thus, FGSM can efficiently craft large volumes of adversarial examples in a very short time, which copes with the sample-hungry nature of adversarial training.

The generated adversarial example set X_{adv} was then fed through the model. In the classification results, for a given class y_{true} , we search the corresponding robust class based on maximising the following likelihood from observation:

$$\underset{y_{robust}}{\text{argmax}} P(F(x_a) = y_{robust} | \theta, y_{true}, x_a \in X_{adv}) \quad (8)$$

Based on the estimation, we selected the most likely y_{robust} as the robust label-pair of y_{true} . Following this procedure, we build the paired-label for each sample in the training dataset. The mappings between y_{robust} and y_{true} are saved as a table *Classmap*. Later on in the grey-box setting, this information will be used to access the credibility of an input example.

C. 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.2. The framework is trained under a multi-task objective function, which is designed to maximise the difference 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 transferred attack.

1) *Multi-task training for black-box/grey-box attack:* According to the previous analysis on robust class-pair, we use the robust label-pairs to conduct the multi-task training [4]. Assume the original model has the logits layer which has outputs Z . Our method grows another logits layer which 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} , given y_{true} . 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:

$$J_{obj} = \alpha J(x, y_{true}) + \beta J(x_{adv}, y_{true}) + \gamma \mathcal{J}'(x, x_{adv}, y_{robust}) \quad (9)$$

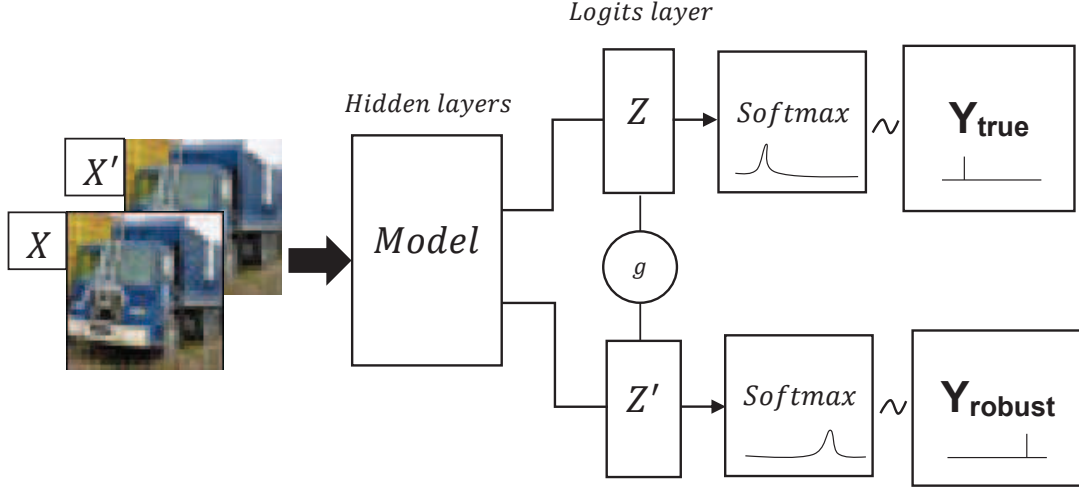


Fig. 2. The multi-task training framework. The training objectives are that, when the incoming input is benign, the outputs from Y_{true} and Y_{robust} will match the pairwise relationship between different classes; When the input is adversarial, the output from Y_{true} stays the same, while the outputs from Y_{robust} will be changed. The collaborative learning by adding a connection between Z and Z' further hardens the model against near-white-box attack.

Herein, x is the benign example. We use adversarial examples to regularise the model, x_{adv} is the adversarial example. 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. α , β , and γ are weights adding up to 1.

The first term of the objective function decides the performance of the original model F on the benign examples. The second term is an adversarial term taking in the adversarial examples 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 [6], to effectively use adversarial example to regularise the model training, we set $\alpha = \beta = 0.4$, and set $\gamma = 0.2$. The cost function \mathcal{J}' is the average over the costs on benign examples and adversarial examples:

$$\mathcal{J}'(x, x_{adv}, y_{robust}) = \frac{1}{2} \{ J(x, y_{robust}) + J(x_{adv}, y_{robust}) \} \quad (10)$$

Herein $J(x, y_{robust})$ is the cross-entropy cost, and $J(x_{adv}, y_{robust})$ is a negative cross-entropy cost function to maximise the difference between the output distribution with y_{robust} , 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 grey-box setting, 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, but cannot guarantee that the output through Z' has the correct mapping relationship to the output through Z in the *Classmap*, grey-box attack is then detected by our architecture.

2) *Breaking grey-box defence*: In the near-white-box setting, the adversary has access to the model weights and the

defensive mechanism. Therefore, adversarial examples can still be crafted against the model under the grey-box defence.

In the grey-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, but cannot guarantee that the output through Z' has the correct mapping relationship to the output through Z , according to the *Classmap*. However, in the near-white-box setting, when the adversary performs adversarial searching for targeted adversarial examples on the model without the gradient lock unit g , the optimisation solver can find a solution by back-propagating a combined loss function $L(Z(x), Z'(x), t, t')$:

$$L(Z(x), Z'(x), t, t') = \eta_1 \cdot L_1(Z(x), t) + \eta_2 \cdot L_2(Z'(x), t') \quad (11)$$

Herein t is the target output from Z , t' is the target output from Z' . t and t' should be a pair in the *Classmap*. Here we assume that the attacker can feed large amount of adversarial examples through the protected model and find out the paired t' for each t . This is a very strong assumption even in the grey-box setting. η_1 and η_2 can be set by the attacker to control the convergence of the solver. The gradients for back-propagating the adversarial loss from logits layer Z and Z' then becomes:

$$\frac{\partial L(Z(x), Z'(x), t, t')}{\partial x} = \eta_1 \cdot \frac{\partial L_1(Z(x), t)}{\partial Z(x)} \frac{\partial Z(x)}{\partial x} + \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial Z'(x)} \frac{\partial Z'(x)}{\partial x} \quad (12)$$

Thus, 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 nearly white-box setting. The detection method used for grey-box attack collapses in this case. To solve the nearly white-box defence problem, we introduce a collaborative architecture into the framework.

3) *Collaborative training for near-white-box attack*: We develop the framework to defend against not only the black-box/grey-box attack, but also the near-white-box attack, in which the adversary has the knowledge of both the model and the defence.

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 to better align with the non-linear relationship between Z and Z' . The last layer of g is a multiplier, which multiplies Z with the output of g in 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 III-C1.

The added extra layers contains 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:

$$\begin{aligned}
\frac{\partial L(Z^*(x), Z'(x), t, t')}{\partial x} &= \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial x} + \\
&\quad \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial x} \\
&= \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \frac{\partial Z^*(x)}{\partial x} + \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial Z'(x)} \frac{\partial Z'(x)}{\partial x} \\
&= \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \left(\frac{\partial Z^*(x)}{\partial Z(x)} \frac{\partial Z(x)}{\partial x} + \frac{\partial Z^*(x)}{\partial Z'(x)} \frac{\partial Z'(x)}{\partial x} \right) + \\
&\quad \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial Z'(x)} \frac{\partial Z'(x)}{\partial x} \\
&= \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \frac{\partial Z^*(x)}{\partial Z(x)} \frac{\partial Z(x)}{\partial x} + \\
&\quad \left\{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \frac{\partial Z^*(x)}{\partial Z'(x)} + \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial Z'(x)} \right\} \frac{\partial Z'(x)}{\partial x} \quad (13)
\end{aligned}$$

It can be seen that in the second term, the back propagation of 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 III-B. If the outputs match the mapping relationship from the *Classmap*, the output is believed to be credible. Otherwise, the input example is identified as an adversarial example. Therefore, our defensive measure is to detect and reject adversarial examples. Furthermore, the regularised model output from Z^* can remedy the detection module once mis-detection occurs.

IV. EVALUATION

In this section, we present the evaluation on our proposed method on defending against the state-of-the-art attacking

TABLE I
MODEL ARCHITECTURES

Layer Type	Oracles	$S_{Cifar10}$	S_{MNIST}
Convo+ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$	$3 \times 3 \times 32$
Convo+ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$	$3 \times 3 \times 32$
Max Pooling	2×2	2×2	2×2
Dropout	0.2	-	-
Convo+ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$	$3 \times 3 \times 64$
Convo+ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2	2×2
Dropout	0.2	-	-
Convo+ReLU	$3 \times 3 \times 128$	-	-
Convo+ReLU	$3 \times 3 \times 128$	-	-
Max Pooling	2×2	-	-
Dropout	0.2	-	-
Fully Connected	512	256	200
Fully Connected	-	256	200
Dropout	0.2	-	-
Softmax	10	10	10

TABLE II
EVALUATION DATASETS

Dataset	FGSM	IGS	Deepfool	C&W L_2
MNIST	10,000	1,000	1,000	1,000
Cifar10	10,000	1,000	1,000	1,000

methods. We first evaluated the performance of our defence against the FGSM, IGS, and Deepfool, in black-box and grey-box settings. Then we evaluated the defence against C&W attack in black-box, grey-box, and near-white-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 L_2 attack, we adopt the implementation from Carlini in their paper [3]. For other attacks, we employ the implementations provided in Foolbox [20].

A. Model, data, and attack

We have implemented one convolutional neural networks architecture as an oracle. To simply the evaluation, we use 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 I.

We first evaluate our method in a black-box setting against four state-of-the-art attacks, namely FGSM, IGS, Deepfool, and Carlini&Wagner L_2 . Second, we evaluate our defence against FGSM, IGS and Deepfool in a grey-box setting. Last but not the least, we evaluate our defence against the C&W attack in both black-box, grey-box and near-white-box settings. For FGSM, in each evaluation session, we craft 10,000 adversarial examples as the adversarial test dataset. For other attacks, considering their inherited nature of heavy computational cost, we craft the adversarial examples of the first 1,000 samples from *MNIST* dataset and *CIFAR10* dataset. We summarise the sizes of all adversarial datasets in Table II. We do not evaluate the detection performance of our defence in black-box setting, since it has been evaluated in the more strict grey-box setting. However, we tested the

robustness of our defence against the transferring black-box attack.

B. Our defence in a black-box setting

We evaluate our defence against the transferring attack in a black-box setting. In the black-box setting, the attacker has neither the defensive method nor the parameters of the oracle model. Instead, the adversary possesses a substitute model whose decision boundaries approximate that of the oracle's. Therefore, attacks launched in a black-box setting are mainly transferring attack.

In our evaluation, we set $\epsilon = 0.1$ in FGSM, in order to transfer attack from the substitute to the oracle. For C&W L_2 attack, we set the parameter κ to 40, as the setting used to break a black-box distilled network in the original paper [3], to produce high-confidence adversarial examples. Later, we also evaluated the performances of our defence under different κ values.

1) *Training the substitute*: To better align the manifold learned by the substitute and the oracle, we feed the whole *MNIST* training dataset into the *MNIST* substitute to train it. We do the same while training the *CIFAR10* substitute. Therefore, we end up having two substitutes for the *CIFAR10* classification task and the *MNIST* classification task. These two substitutes have achieved an equivalent performance with the models used in the previous papers [3], [13], [19]. For the *MNIST* substitute, it achieves 99.4% classification accuracy on 10,000 test samples after it is trained. The trained *CIFAR10* substitute achieves 78.6% accuracy on 10,000 *CIFAR10* test samples. The trained substitutes are named as S_{MNIST} and $S_{CIFAR10}$. The architecture of the substitutes is summarised in Table I.

2) *Defending a black-box attack*: We present the performance of our defence against transferring black-box attack here. First, given S_{MNIST} and $S_{CIFAR10}$, we use the above four attacks to craft adversarial sets whose sizes are listed in Table II. 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 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 III. Herein, 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 III).

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 tackle the black-box defence because the confidence of the generated example is set to a very high 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 C&W attack.

C. Our defence in a grey-box attack

We then measure the success rate on detecting adversarial examples. In a grey-box setting, the attacker has the oracle model but does not know the defensive method. The attacks do not rely on transferring adversarial examples from substitute, which means regularising the decision boundaries of the oracle is no longer an effective way against the attack. Instead, we rely on detection mechanism to tackle the attack.

We assessed the detection rate (i.e. percentage of the detected adversarial examples in the whole adversarial test set) against FGSM, IGS, and Deepfool. The parameter ϵ used to control adversarial perturbation level in FGSM is set to 0.05. For each grey-box adversarial test set, the results are shown in Table IV. It can be found that our detection-based defence can tackle all the attacks very well. The performance on *CIFAR10* task seems worse than on *MNIST* task due to the reason that the *CIFAR10* oracle itself has less accuracy.

D. Detecting C&W attack

We evaluate our defence against the C&W attack here. The attacking confidence of adversarial examples from C&W attack is changeable through adjusting the hyper-parameter κ in the adversarial objective function. Large κ value will lead to producing a high-confidence adversarial example. We evaluated the adversarial example detection performance against C&W examples crafted using different κ values in the black-box, grey-box, and near-white-box settings.

1) *Black-box*: In a black-box setting, the high-confidence C&W examples can better transfer from the substitute to the oracle. We test the defence performance on invalidating adversarial examples and the performance on detecting adversarial examples.

First, we evaluate the performance on invalidating transferred black-box attack. To demonstrate the performance, we measure the success rates of the transferred attacks which change the output results from the Z^* . The successful transfer rate from the substitutes to the oracles are plotted in Fig. 3. When κ is set higher, 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 invalid the adversarial examples near the decision boundaries. But unfortunately, the high-confidence examples are usually far away from the decision boundaries. For defending the

TABLE III
CLASSIFICATION ACCURACY ON NON-TARGETED BLACK-BOX ADVERSARIAL EXAMPLES

	Model		<i>FGSM</i>	<i>IGS</i>	<i>Deepfool</i>	<i>C&WL₂</i>
Black-box defence	O_{MNIST}	Original	34.3%	35.4%	34.1%	12.6%
		Defended	96.2%	95.8%	96.3%	12.6%
	$O_{CIFAR10}$	Original	61.6%	50.2%	63.8%	8.9%
		Defended	68.6%	62.5%	71.3%	10.0%

TABLE IV
THE DETECTION RATE AGAINST ADVERSARIAL EXAMPLES IN GREY-BOX SETTINGS.

	Model	<i>FGSM</i>	<i>IGS</i>	<i>Deepfool</i>
Grey-box	O_{MNIST}	98.7%	94.4%	95.2%
	$O_{CIFAR10}$	75.2%	72.6%	73.8%

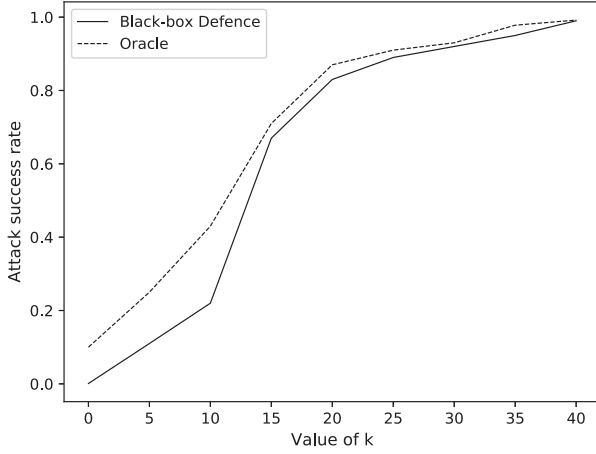


Fig. 3. The rate of successful transferred adversarial examples by C&W attack, in black-box setting. Our black-box defence decreased the success attack rate when κ is under 10. However, when κ became higher, the black-box defence turned out to be invalid.

high-confidence adversarial examples, we mainly rely on the detecting mechanism.

Second, we evaluate the precision and the recall on detecting black-box adversarial examples crafted under different κ . The precision values and the recall values are plotted in Fig.4.

2) *Grey-box*: In a grey-box setting, the attacker knows the model parameters, but not the defence method. To evaluate the performance of our method on defending the grey-box C&W attack, we vary κ from 0 to 40. For each κ value, we craft 1,000 adversarial examples based on the oracle (without the Z' branch). We then mix 1,000 benign examples into each group of 1,000 adversarial examples to form the evaluation datasets for different κ values. We measure the precision and the recall of our defence on detecting the grey-box 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. The precision and recall values are plotted in Fig.5. Our defence method has achieved high precision and recall on detecting the grey-box adversarial examples.

3) *Near-white-box*: We evaluate the performance of our defence against the C&W attack in the near-white-box setting.

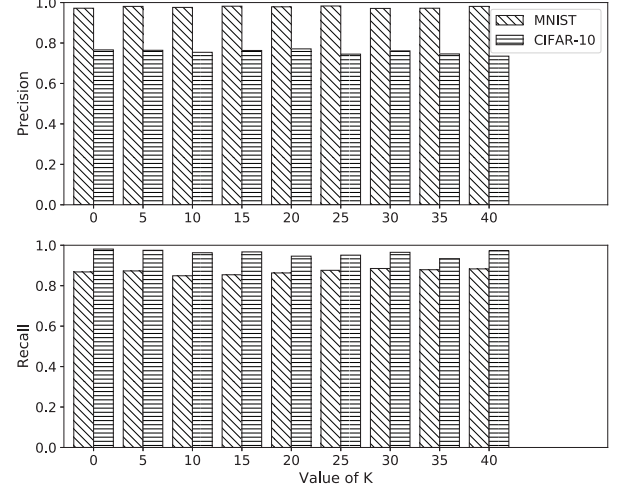


Fig. 4. The precision and recall of detecting black-box C&W adversarial examples.

Adversarial examples used in the evaluation are crafted based on the linearly-combined adversarial loss functions mentioned in Section III-C2. To measure the near-white-box defence, we sweep the value of κ from 0 to 40, and then we examine the rate of successful adversarial image generation given 100 *MNIST* images under each κ value. We record the successful generation rate of targeted and non-targeted attacks based on *MNIST* data. A failed generation should either be no solution found, or generating an invalid image (totally black or totally white). For targeted attack, we randomly set the target label during the generation process. The generation rates are recorded in Table.V.

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

E. 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^* , in both black-box and grey-box settings, after our protections are applied on the oracle. We feed 10,000 *CIFAR10* examples and 10,000 *MNIST*, separately, into the corresponding defended oracles. The results are in Table.VI. Herein the classification accuracy of the protected model is the accuracy of the output classification label through

TABLE V
THE SUCCESSFUL GENERATION RATE OF C&W ATTACK IN NEAR-WHITE-BOX SETTING

Attack	Model	$\kappa = 0$	5	10	15	20	25	30	35	40
Non-targeted	Original	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Defended	8%	0%	0%	0%	0%	0%	0%	0%	0%
Targeted	Original	100%	100%	100%	100%	100%	98%	96%	92%	90%
	Defended	0%	0%	0%	0%	0%	0%	0%	0%	0%

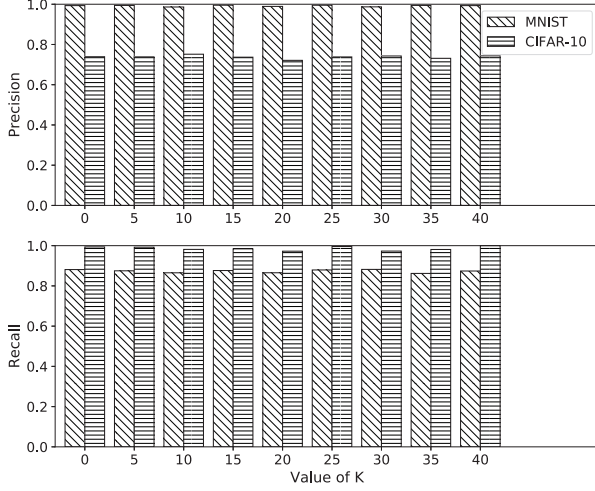


Fig. 5. The precision and the recall of detecting grey-box C&W adversarial examples.

TABLE VI
THE CLASSIFICATION ACCURACY ON BENIGN EXAMPLES

	Oracle	Defended
MNIST	99.4%	99.4%
CIFAR10	78.6%	76.5%

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 the 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.

V. 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.

A. Classmap evaluation

We evaluate the generalisation of the *Classmap* extracted from different attacking examples. We generate *Classmaps*

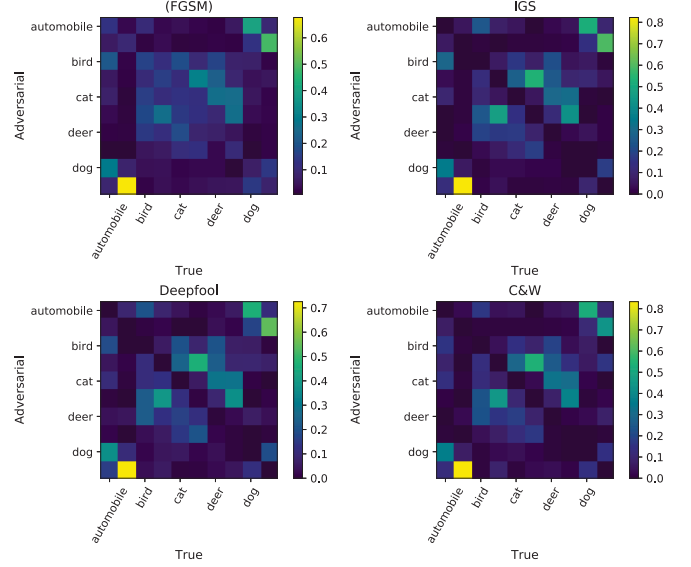


Fig. 6. The misclassification class map of FGSM, IGS, Deepfool, and high confidence C&W examples. It can be found that all the classmaps share a similar pattern.

from FGSM, IGS, Deepfool, and C&W example sets. The heat maps of the *Classmaps* are displayed in Fig.6. 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 generalises across different attacks.

B. Defending a normal black-box attack

For a black-box attack, the adversarial training introduced in this model can effectively regularises the decision boundaries of the model to tackle adversarial examples near the decision boundaries. Compare to vanilla adversarial training [23], our method can further increase the distance required for moving adversarial examples. Crafting adversarial examples based on deep neural net leverage back-propagation of objective loss to adjust the input pixel values, hence, the adversary is actually moving the data point in the feature space according to the gradient direction $\nabla_{x_{adv}} L(x_{adv}, y)$ to maximise the adversarial loss L (a non-targeted attack in a black-box setting). Suppose the adversarial example is found after n steps of gradient decent, for each step of gradient decent, we have a step size α , the total perturbation can be approximately written as:

$$\delta = x_n - x_0 = \alpha \cdot (\nabla_{x_{n-1}} L^{n-1} + \nabla_{x_{n-2}} L^{n-2} + \dots + \nabla_{x_1} L^1 + \nabla_{x_0} L^0) \quad (14)$$

According to Section III-B, the adversary relies on the gradient decent based updates to gradually perturb image until it becomes adversarial. In the non-targeted attack, the search will stop when an adversarial example is found. Given an example whose original label is l , it is unlikely to classify it into the robust label l_r . In other words, within certain steps, the gradient updates $\nabla_{x_i} J^i$ will not converge if the adversarial cost J is calculated based on the output maximised at l and the robust target l_r (or vice versa). Similarly, in the targeted attacks, l_r is more difficult to be used as the adversarial target label. Suppose the total effort for maximising/minimising $L(x : F(x) = l, l_i : i \neq r)$ is δ_i , the total effort for maximising/minimising $L(x : F(x) = l, l_r)$ is δ_r , we have $\delta_r > \delta_i$. When the training objective includes a term containing the robust label y_{robust} , the output of the trained model could be treated as a linear combination of the outputs trained from l and l_r . Therefore, the required total effort for changing the combined output should be 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.7). 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.

C. 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 adversarial examples, given the logits from the model. By observing the objective function $f(x_{adv}) = \max(\max\{Z(x_{adv})_i : i \neq l\} - Z(x_{adv})_l, -\kappa)$, we can find out that the objective is actually increasing the value of the logits corresponding to the desired l class, until the difference between l class and the second-to-the-largest class reaches the upper bound defined by κ . The optimisation process can be interpreted as adjusting the input pixels along the direction of the gradient that maximise the logits difference.

In a grey-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 grey-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'

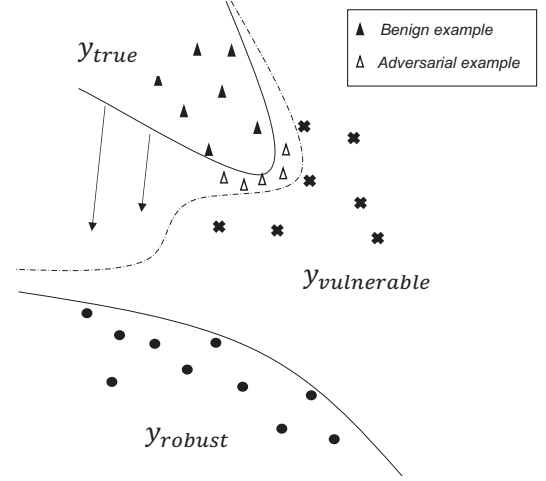


Fig. 7. The regularisation in a 2-dimension sample space. Embedding the robust class y_{robust} in the loss function will further move the decision boundary between y_{true} and y_{robust} towards y_{robust} . The fractions of the decision boundary that were not regularised by injecting adversarial examples will get regularised. Thus, for black-box examples that are near the decision boundary will be invalidated.

is not guaranteed to be the corresponding robust label of t in the *Classmap*.

In a near-white-box setting, the defence is exposed to the adversary. Therefore, the adversary can first feed forward certain volume of adversarial examples to find the approximate *Classmap* of the task dataset. Then, the attacker can set the adversarial objective to a linear combination form mentioned in Section III-C2 to successfully bypass the defence. However, after the gradient lock unit is added, first, 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 previous last step of gradient update, the adjustments made on Z and Z' becomes:

$$\delta_z = \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \frac{\partial Z^*(x)}{\partial Z(x)} \frac{\partial Z(x)}{\partial x} + \{ \eta_1 \cdot \frac{\partial L_1(Z^*(x), t)}{\partial Z^*(x)} \frac{\partial Z^*(x)}{\partial Z'(x)} + \eta_2 \cdot \frac{\partial L_2(Z'(x), t')}{\partial Z'(x)} \} \frac{\partial Z'(x)}{\partial x} \quad (15)$$

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 both generate adversarial label and the correct paired robust label of the adversarial label.

VI. RELATED WORK

A. Defensive methods

In the category of robustness based methods, the first defensive mechanism employs adversarial training to enhance the robustness of neural nets [23]. Later on, there are methods

which change the neural network design to improve the robustness. For example, Gu et al. proposed a contractive neural net. The contractive net solves the adversarial example problem by introducing a smoothness penalty on the neural network model [7]. Techniques used in transferring learning have been adopted to tackle adversarial attacks. 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 [19]. 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. Defensive distillation extends the idea of knowledge distillation, which is originally used to reduce the neural net dimension.

In contrast to making changes on the original DNN model, the detection based methods adopt a second model to detect examples with adversarial perturbation. For instance, Metzen et al. attached detectors on the original model and trained it with adversarial example [14]. Another method employs support vector machine to classify the output from the high-level neural network layer [12]. Lately, MagNet relies on autoencoder to reconstruct adversarial examples to normal example, and detect adversarial example based on the reconstruction error and output probability divergence [13].

However, current defensive methods focusing on improving DNN robustness are not actually making the neural nets more robust. Instead, they just result in more fail attempts for the existing attacking methods. Moreover, current methods usually decrease the performance of the neural nets on non-adversarial data. The detection based methods can detect high-confidence adversarial examples, However, they usually generalise poorly on different type of attacks.

B. Attacking methods

Adversarial examples are crafted based on either maximising the loss between the model output and the ground truth (i.e. non-targeted attack) or minimising the loss between the model output and the desired adversarial results (i.e. targeted attack). The adversary can create hard-to-defend examples if he obtains the architecture and the parameters of the oracle model (i.e. white-box attack). In black-box attack, crafting adversarial examples requires using a DNN substitute. However, current applications of DNN largely rely on few representative DNN models, which makes it a relatively easy task for adversaries to guess the substitutes. Moreover, due to the transferability of adversarial samples [11], the crafted examples can be used in black-box attacks towards other DNNs [16] without knowing the model parameter or the training data.

The representative methods for crafting adversarial examples can be broadly categorised into two types:

Gradient based attack. Gradient based method employs the gradients of the output error with respect to each input pixel to craft an adversarial example. In the fast gradient/sign method, each input pixel will be slightly perturbed along the gradient direction or gradient sign direction [6]. This type of methods only compute gradients using the back-propagation once. Therefore, this type of methods is able

to craft large batch of adversarial examples in a very short amount of time. Additionally, an iterative gradient sign based method was proposed in [9]. The method iteratively perturbs pixels instead of conducting a single step perturbation on all pixels.

Optimisation based attack. Optimisation based methods aim to optimise the output error while keeping the distance between the original example and the adversarial example as close as possible. First, Szegedy et al. proposed a L-BFGS based attack, which solves the adversarial example crafting problem as a box-constraint minimisation problem [23]. Later on, a Jacobian-based saliency map attack was proposed by Papernot et al. [18]. This method modifies the most critical pixel in each saliency map during the iteration to lead to the final adversarial example. Furthermore, a method named DeepFool is proposed. DeepFool iteratively finds the minimum perturbations of images [15]. Last but not the least, Nicolas et al. proposed an optimisation attack based on modified objective function [3]. This method can effectively invalidate defensive distillation, which is a state-of-the-art defensive method.

Both gradient based and optimisation based methods can be used to perform non-targeted attack and targeted attack. Optimisation based attacks can usually achieve higher attacking success rate, and more importantly, higher confidence on the misclassification. Therefore, optimisation methods are usually stronger than the fast gradient based methods.

VII. 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

or near-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 [17]) 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 JSMA examples due to the regularised training process.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel defence against black-box attack, grey-box attack, and near-white-box attack on deep neural networks. Importantly, our method is able to protect CNN 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 mapping relationship 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.

Deep neural networks 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 on the real-world systems built on DNN. Previous researches have made attempts to attack online deep learning service providers, such as Clarifi [11], Amazon Machine Learning, MetaMind, and Google cloud prediction API [16]. 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 would be fatal. This could be another direction for the future works.

REFERENCES

[1] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

[2] N. Carlini and D. Wagner. Magnet and” efficient defenses against adversarial attacks” are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017.

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

[4] 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.

[5] 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.

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

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

[8] 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*, pages 770–778, 2016.

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

[10] 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.

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

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

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

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

[15] 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.

[16] 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.

[17] 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)*, pages 372–387. IEEE, 2016.

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

[19] 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)*, pages 582–597. IEEE, 2016.

[20] J. Rauber, W. Brendel, and M. Bethge. Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

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

APPENDIX

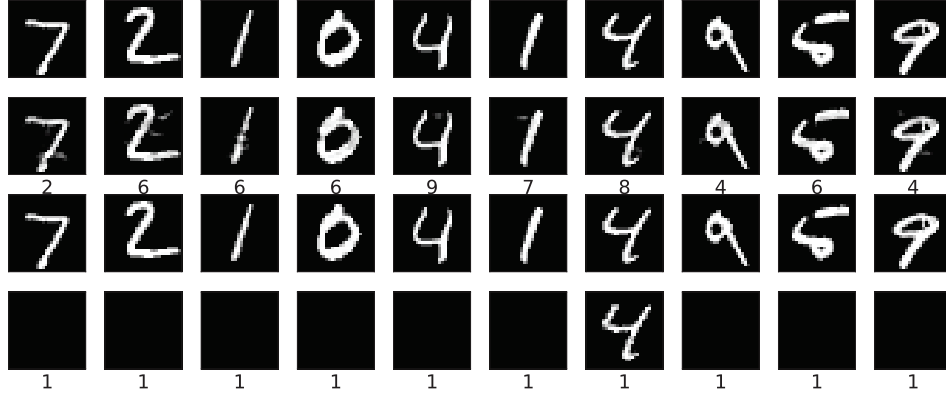


Fig. 8. The adversarial *MNIST* images generated using non-targeted C&W attack when $\kappa = 0$. The first row and the third row are the original images. The second row is the generated adversarial image based on the original model (classification results are as the labels below the second row). The fourth row is the failed generation after applying our defence on the model.

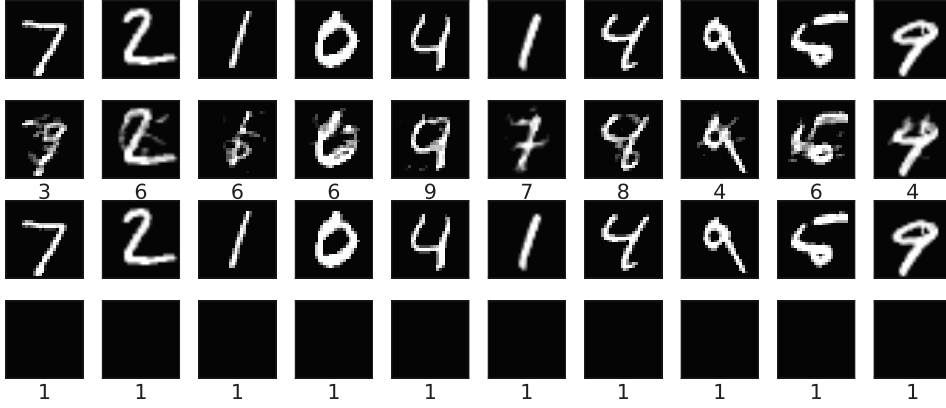


Fig. 9. The adversarial *MNIST* images generated using non-targeted C&W attack when $\kappa = 40$. The first row and the third row are the original images. The second row is the generated adversarial image based on the original model (classification results are as the labels below the second row). The fourth row is the failed generation after applying our defence on the model.

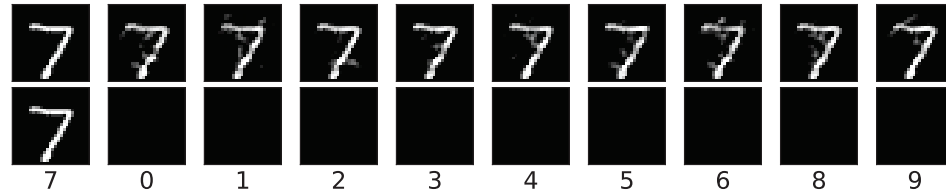


Fig. 10. The adversarial *MNIST* images generated using targeted C&W attack when $\kappa = 0$. Images in the leftmost column are the original images. Images in the first row are the targeted adversarial image generated based on the original model. Images in the second row are the generated targeted adversarial image after applying our defence on the model. Classification results are as the labels below.

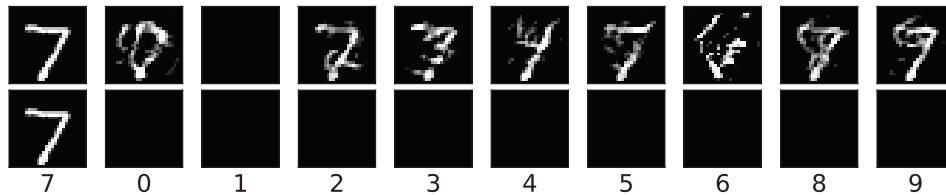


Fig. 11. The adversarial *MNIST* images generated using targeted C&W attack when $\kappa = 40$. Images in the leftmost column are the original images. Images in the first row are the targeted adversarial image generated based on the original model. Images in the second row are the generated targeted adversarial image after applying our defence on the model. Classification results are as the labels below.