# Layer-wise Adversarial Training Approach to Improve Adversarial Robustness

Xiaoyi CHEN
*NEC Laboratories China*
*NEC (China) Co., Ltd.*
Beijing, China
chen_xiaoyi@nec.cn

Ni ZHANG
*NEC Laboratories China*
*NEC (China) Co., Ltd.*
Beijing, China
zhangni_nlc@nec.cn

***Abstract*—Deep neural networks (DNNs) have delivered state-of-the-art performance in many challenging tasks, such as in computer vision, but they are vulnerable to adversarial attacks. Adversarial training is a technique for augmenting training data with adversarial examples and has empirically proven to be the most effective method of defense against adversarial attacks. Motivated by the fact that intermediate layers play a highly important role in maintaining a robust model, we propose to extend conventional adversarial training, which is designed to only manipulate input, such that it becomes layer-wise training. Distinct from previous studies in which robust DNN models were trained in a layer-wise manner, the layer perturbation introduced by our method theoretically proves to be equivalent to the adversarial manipulation of network inputs. This approach guarantees an improvement in the adversarial robustness of DNN models to which the method is applied. We empirically evaluated both shallow and deep CNN models, such as VGG16 and WideResNet28-10, by using MNIST, CIFAR-10, and CIFAR-100 datasets. The results consistently showed that the proposed layer-wise adversarial training approach significantly outperforms conventional adversarial training and that it offers defense against all mainstream attacks including FGSM, IFGSM, PGD, EoT, and C&W. Combining the layer-wise training regime with conventional adversarial training would make it possible to achieve excellent defense performance.**

***Index Terms*—Adversarial training, Adversarial robustness, Adversarial defense, Layer-wise perturbation**

## I. Introduction

Despite their popularity and unprecedented performance on many prediction tasks, deep neural networks (DNNs) are known to be insufficiently robust and vulnerable to adversarial attacks [1]. A well-known adversarial example [2] was used to demonstrate that, by adding carefully crafted perturbation, DNN models over-confidently make incorrect predictions even though the added noise is imperceptible to humans.

Many adversarial attack methods [2]–[5] exist, among which some are sufficiently powerful to deceive multiple classifiers even without knowing the architecture of the target model [6]. Kurakin *et al.* [4] demonstrated that adversarial examples can also be effective in the physical world. This has raised serious concerns when DNNs are employed in safety-critical domains such as for medical diagnosis and in self-driving vehicles.

Research on methods to defend against various adversarial attacks has been attracting considerable interest [1], [7]–[10]. The most effective defense approach is "adversarial training," which generates adversarial samples during the training process and uses them as augmented training data to enhance the robustness of the model [1], [3], [8], [11].

In this paper, we propose a new layer-wise approach for adversarial training to improve the defense performance. Different from conventional adversarial training, which only manipulates the network inputs, we introduce adversarial perturbation in the intermediate layers of neural networks during training. Distinct from previous related work [12], [13], which proposed to train robust DNN models in a layer-wise manner, the layer perturbation introduced by our method theoretically proves to be equivalent to the adversarial manipulation of network inputs, and it guarantees the improvement of the adversarial robustness of DNN models to which our method is applied. We compared our proposed layer-wise method with conventional adversarial training methods on both shallow and deep models, such as VGG16 and WideResNet28-10, and used MNIST, CIFAR-10, and CIFAR-100 datasets to empirically show that the adversarial robustness against all mainstream attacks, including FGSM, IFGSM, PGD, EoT and C&W, is considerably improved. The empirical evaluation also revealed that

- applying our layer-wise adversarial training to all layers (including the input layer) achieves the best adversarial robustness;
- the more the layers that receive adversarial training, the more robust the defense performance.
- the closer the adversarially trained layers are to the network output layer, the more robust the defense performance.
- the addition of increased randomness $\epsilon$ to the adversarial perturbation improves the defense performance of both the proposed layer-wise adversarial training method and conventional adversarial training against adversarial attacks.

## II. Preliminaries and Related Work

This section briefly reviews mainstream adversarial attack methods and adversarial training that augments training data with adversarial examples to defend against these attacks. A few related studies that leverage layer-wise noise to improve the regularization performance of DNNs are also discussed.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

### A. Adversarial Attacks

The objective of adversarial attacks is to deceive models by adding perturbations to inputs. Two attacking scenarios exist: a white-box attack setting where the architecture and weights of targeted models are known, and a black-box attack setting where intrinsic information about the target models and their parameters are unknown. Most of the white-box attack methods depend on gradient information to craft adversarial examples, whereas in the black-box setting, queries are used to estimate the decision boundary [14] or approximate gradients [15]. In practice, the black-box adversarial attacks can be simulated by generating adversarial samples targeting a specific model and then relying on the transferability of those adversarial examples to blindly attack other models. Existing approaches to generate adversarial examples for a specifically targeted model can be categorized into three types.

*1) One-step gradient-based methods:* The fast gradient sign method (FGSM) is a representative method [3]. The adversarial perturbation is crafted by maximizing the loss function $\mathrm{J}\left(\mathrm{x}^{*}, \mathrm{y}\right)$, where J is typically the cross-entropy loss. The FGSM can generate adversarial examples constrained by $\mathrm{L}_{\infty}$ norm $\left\|x^{*}-x\right\|_{\infty} \leq \varepsilon$ as:

$$\boldsymbol{x}^{*}=\boldsymbol{x}+\epsilon \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}, y)\right) \tag{1}$$

where $\nabla_{\mathrm{x}} \mathrm{J}(\mathrm{x}, \mathrm{y})$ is the gradient of the loss function, and $\operatorname{sign}(\cdot)$ denotes the signum function.

The latest gradient-based method was proposed to attack obfuscated gradients [16]. The method computes the gradients of randomized defense by applying expectation over transformation (EoT) [17]. This gradient-based attack is then computed as an expectation of gradients of randomized outputs.

*2) Iterative methods:* Iterative methods employ the fast gradient multiple times with a considerably smaller step size, $\alpha$. The iterative version of FGSM (IFGSM), also known as BIM, can be expressed as follows:

$$\boldsymbol{x}_{0}^{*}=\boldsymbol{x}, \quad \boldsymbol{x}_{t+1}^{*}=\boldsymbol{x}_{t}^{*}+\alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J\left(\boldsymbol{x}_{t}^{*}, y\right)\right). \tag{2}$$

To ensure that the generated adversarial perturbation satisfies the $\mathrm{L}_{\infty}$ limitation, it suffices to simply set $\alpha=\epsilon / \mathrm{T}$, where $\epsilon$ is the maximum value of perturbation and T the number of iterative steps.

Another iterative method is the projected gradient descent (PGD) method, which crafts adversarial examples in a $\gamma$-ball, as

$$\boldsymbol{x}_{t+1}^{*}=\Pi_{\gamma}\left\{\boldsymbol{x}_{t}^{*}+\alpha \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J\left(\boldsymbol{x}_{t}^{*}, y\right)\right)\right\}, \tag{3}$$

where $\Pi_{\gamma}$ is the projection onto the set $\{x|\left\|x-x_{0}\right\|_{\infty} \leq \gamma\}$. Because the number of iterations in a PGD attack is not constrained, it can generate stronger adversarial attacks than IFGSM in the white-box attack setting.

*3) Optimization-based methods:* These methods directly optimize the distance between real and adversarial examples, subject to the misclassification of adversarial examples. Such a method was first introduced as an L-BFGS attack [1] and improved to form the Carlini-Wagner attack (C&W) [5]. The goal of a C&W attack is to determine the value of minimal perturbation $\delta$ such that $\mathrm{D}(\mathrm{x}+\delta)=\mathrm{t}'$, where D is the target model; $\mathbf{x}$ is an input; $t'$ represents either the target class in targeted attacks or a class different from the ground truth class in non-targeted attacks. The total loss function is expressed as follows:

$$\begin{array}{c}\arg \min_{\delta}\|\delta\|_{p}+c \cdot J(x+\delta, y) \\ \text{s.t. } x+\delta \in[0,1]^{n}\end{array} \tag{4}$$

The C&W attack was shown to be extremely strong in the white-box attack setting [5], achieving a misclassification error of over 99.8% on CIFAR-10 dataset, but it is computationally expensive.

### B. Adversarial and Robust Training

Adversarial training augments training data with adversarial examples. It acts as a min-max optimization problem in which adversarial samples are generated to maximize the classification loss (e.g., cross entropy) while the classifier attempts to minimize this loss. Adversarial training is iteratively conducted, and it involves two steps in each iteration. The first step entails generating adversarial examples, and in the second step, a model is trained on these adversarial examples. The generation of adversarial samples is based on obtaining information about the gradient between classification loss and input samples. In other words, in conventional adversarial training [1], [3], [8], [10], adversarial manipulation is conducted only on network inputs.

Although adversarial training aims to improve the robustness of a model against adversarial inputs, in previous studies [1], [3], it was also observed that the use of adversarial examples to train DNN models serves to regularize and marginally improve the performance of the base network on the test data. A recent study by Sankaranarayanan *et al.* [12] extended adversarial training to a layer-wise regularization mechanism to prevent over-fitting. These researchers showed that their layer-wise approach provides much more effective adversarial perturbation and thus stronger regularization, compared to methods that perturb only the input layer. Liu *et al.* [13] extended the aforementioned concept to render the injected perturbation in each layer learnable during the training of DNNs. Unlike the aforementioned method [12] that generates noise for the current mini-batch from the previous mini-batch, adversarial noise propagation(ANP) [13] computes adversarial noise by using the same mini-batch and by utilizing correlations within the mini-batch data.

Herein, we propose to improve the adversarial robustness by introducing adversarial perturbation in the intermediate layers of a DNN. Our aim is to improve the adversarial robustness of conventional adversarial training, which only manipulates network inputs. Distinct from the previous work [12], [13] in which robust DNN models have been proposed to be trained in a layer-wise manner, the adversarial perturbation introduced in intermediate layers by our method theoretically proves to be equivalent to adversarial manipulation on network inputs, which guarantees the improvement of the adversarial

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

robustness of the DNN models to which our method is applied. Unlike the existing method [12], we focus on improving the adversarial robustness instead of the regularization performance on test data to prevent over-fitting. Similar to the work of Liu *et al.* [13], the layer-wise adversarial perturbation in our proposed method is adaptively based on neural networks during training. To the best of our knowledge, existing studies on layer-wise training for adversarial robustness are limited.

## III. Our Approach – Layer-wise Adversarial Training

We propose to introduce adversarial perturbations in the intermediate layers of neural networks during training. Our goal is to attempt to establish an equivalence between conventional adversarial training and our proposed layer-wise adversarial training. In this section, we provide a detailed derivation of the adversarial perturbations introduced in fully connected and convolutional layers, respectively and present the training procedure of our layer-wise adversarial training method.

### A. Adversarial Perturbation for Fully Connected Layers

We first derive adversarial perturbations for fully connected layers. For simplicity, we assume that the network input is $x$, and the adversarial perturbation is $r$ (with $r$ and $x$ having the same dimension). For one fully connected layer with layer weight $w_1$ and bias $b_1$, its layer output ($y$) becomes $y_{\text{adv}}$ when adding adversarial noise $r$ to input $x$:

$$\begin{aligned} y &= w_1 \cdot x + b_1 \\ y_{\text{adv}} &= w_1 \cdot (x + r) + b_1 \\ \text{as a result, } \; y_{\text{adv}} &= y + w_1 \cdot r, \end{aligned} \tag{5}$$

where $\cdot$ denotes the matrix multiplication operation, and $r$ is computed by any of the adversarial attack methods. With the above equality, training on $x + r$ is equivalent to having a perturbation ($w_1 \cdot r$) on the layer output. In a latter part of this subsection, we prove that $w_1 \cdot r$ on layer output can serve as a regularization term, which improves the adversarial robustness.

When multiple fully connected layers are stacked with layer weight $w_i$ and bias $b_i$, the difference between the regular layer output and adversarially trained output is $(\prod_{i=1,\dots,n} w_i) \cdot r$.

Further, if we generate adversarial noise $r$ by $r = \epsilon \cdot \text{sign}\,(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x})$, the perturbation added to the layer output of a fully connected layer is

$$\begin{aligned} w_1 \cdot r &= w_1 \cdot (\epsilon \ \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x})) \\ &= \epsilon \ w_1 \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}\frac{\mathrm{d}y}{\mathrm{d}x}) \\ &= \epsilon \ w_1 \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y} w_1) \\ &= \epsilon \ w_1 \bigodot \text{sign}(w_1) \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}) \\ &= \epsilon \ |w_1| \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}) \\ &= \text{coeff} \ |w_1| \text{ with coeff} = \epsilon \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}), \end{aligned} \tag{6}$$

where $\bigodot$ represents the Hadamard product; $\cdot$ is the matrix multiplication operation; $\text{sign}(\cdot)$ denotes the signum function; $\epsilon$ can either be a fixed value or follow a distribution, e.g., $\epsilon \sim \mathcal{N}(0, 0.1)$. Eq.6 shows that the adversarial perturbation in the fully connected layer is equivalent to $l_0$ regularization. Stacking multiple fully connected layers, the proposed layer-wise adversarial perturbation ($\mathcal{R}_{fc}$) in the last fully connected layer is derived as follows:

$$\begin{aligned} \mathcal{R}_{fc} &= (\prod_{i=1,\dots,n} w_i) \cdot r \\ &= (\prod_{i=1,\dots,n} w_i) \cdot (\epsilon \ \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x})) \\ &= \epsilon \ (\prod_{i=1,\dots,n} w_i) \bigodot (\prod_{i=1,\dots,n} \text{sign}(w_i)) \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}) \\ &= \epsilon \ (\prod_{i=1,\dots,n} |w_i|) \cdot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}) \end{aligned} \tag{7}$$

### B. Adversarial Perturbation for Convolutional Layers

Similar to the fully connected layer, when adversarial noise $r$ is added to input $x$, the layer output of a single convolutional layer becomes $y_{\text{adv}} = y + w_1 * r$, where $*$ denotes the convolution operation. If we generate adversarial noise $r$ by $r = \epsilon \cdot \text{sign}\,(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}x})$, the perturbation added to the layer output of one convolutional layer is

$$w_1 * r = w_1 * \text{sign}(\epsilon \frac{\mathrm{d}y^1}{\mathrm{d}x}) \bigodot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y^1}), \tag{8}$$

where $\bigodot$ represents the Hadamard product, and $w_1 * \text{sign}(\frac{\mathrm{d}y^1}{\mathrm{d}x})$ is defined as the new adversarial regularization term.

When stacking multiple convolutional layers with layer weight $w_i$, the difference between the regular layer output and the adversarially trained output is $(\prod^{\text{conv}}_{i=1,\dots,n} w_i) * r$, where $\prod^{\text{conv}}_{i=1,\dots,n}$ represents successive convolution operations. The proposed layer-wise adversarial perturbation in the last convolutional layer ($\mathcal{R}_{conv}$) is derived as follows:

$$\begin{aligned} \mathcal{R}_{conv} = & w_n * (\dots(w_1 * \text{sign}(\epsilon \frac{\mathrm{d}y}{x})) \bigodot \text{sign}(\frac{\mathrm{d}y^2}{\mathrm{d}y^1}) \\ & \dots \bigodot \text{sign}(\frac{\mathrm{d}y^n}{\mathrm{d}y^{n-1}})) \bigodot \text{sign}(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y^n}), \end{aligned} \tag{9}$$

where $n$ represents the number of convolutional layers.

Note that the aforementioned derivations of layer perturbation use gradient-based adversarial attacks as examples. For optimization-based attack methods such as C&W, the layer-wise perturbation is calculated similarly by $(\prod^{\text{op}}_{i=1,\dots,n} |w_i|)$ *op* $\delta$, once $\delta$ is obtained. Likewise, $\prod^{\text{op}}_{i=1,\dots,n}$ represents successive *op* operations.

### C. Adversarial Perturbation after Nonlinear Activation Function

Nonlinear activation functions are often used in DNN/CNN layers. This section explains why our proposed layer-wise adversarial perturbation still holds after nonlinear activation. Here, we take a fully connected layer with a ReLU activation

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

function as an example. The derivation for a convolutional layer with a ReLU function is similar. ReLU is the most commonly used nonlinear activation function, and we used it in all our experiments.

When ReLU is used after the layer-wise perturbation operation, we have $y$ and $y_{\text{adv}}$ as the original output and adversarial output of the next layer, respectively.

$$\begin{aligned} y &= w_2 \cdot \text{ReLU}(w_1 \cdot x + b_1) + b_2 \\ y_{\text{adv}} &= w_2 \cdot \text{ReLU}(w_1 \cdot x + w_1 \cdot r + b_1) + b_2, \end{aligned} \tag{10}$$

where $w_2$ and $b_2$ represent the weight and bias of the next fully connected layer, respectively. Similar to eq. 5, the difference between $y_{\text{adv}}$ and $y$ can be regarded as a regularization term to achieve adversarially robust training.

In eq.10, we can reposition $w_2$ inside the ReLU function as $|w_2|$ is positive.

$$\begin{aligned} & w_2 \cdot \text{ReLU}(w_1 \cdot x + b_1) \\ =& sign(w_2) \cdot |w_2| \cdot \text{ReLU}(w_1 \cdot x + b_1) \\ =& sign(w_2) \cdot \text{ReLU}(|w_2| \cdot (w_1 \cdot x + b_1)) \end{aligned} \tag{11}$$

As a result, the difference between $y_{\text{adv}}$ and $y$ is derived as follows:

$$\begin{aligned} \text{diff} =& y_{\text{adv}} - y \\ =& sign(w_2) \cdot [\underbrace{\text{ReLU}(|w_2| \cdot w_1 \cdot x + |w_2| \cdot b_1 + |w_2| \cdot w_1 \cdot r)}_{\text{Relu}_{adv}} \\ & - \underbrace{\text{ReLU}(|w_2| \cdot w_1 \cdot x + |w_2| \cdot b_1)}_{\text{Relu}_{ori}}] \end{aligned} \tag{12}$$

Fig.1 presents $\text{Relu}_{ori}$ and $\text{Relu}_{adv}$ in a coordinate system, supposing that $x$ has one dimension [1]. The figure shows that the two ReLU functions are always in parallel in the positive region, whereas both are zero in the negative region. The slope of the positive section of the line is determined by $|w_2| \cdot w_1$, while $sign(w_1)$ determines the direction of the slope as $|w_2|$ is always positive. The relative positions of $\text{Relu}_{adv}$ and $\text{Relu}_{ori}$ are determined by $r$, while $sign(r)$ determines their relative translation directions. Fig.1 indeed represents the case where $|w_2| \cdot w_1 > 0$ and $r > 0$.

Considering all possible cases (determined by $sign(w_1)$ and $sign(r)$), eq.12 can be generalized as follows:

$$\begin{aligned} &\text{when } x > -\frac{b_1}{w_1} + |r|, \\ &\text{diff} = sign(w_2) \bigodot sign(w_1) \bigodot sign(r) \bigodot |w_2| \cdot |w_1| \cdot |r| \\ &\text{when } x < -\frac{b_1}{w_1} - |r|, \\ &\text{diff} = 0, \end{aligned} \tag{13}$$

where $\bigodot$ represents the Hadamard product, and $\cdot$ represents the matrix multiplication operation; $-\frac{b_1}{w_1}$ represents the intersection point of $\text{Relu}_{ori}$ with the horizontal axis. Considering that adversarial perturbation ($r$) has a significantly smaller value compared to the original image pixel value($x$), we assume $-\frac{b_1}{w_1} \pm |r| \approx -\frac{b_1}{w_1}$. As a result, eq. 13 can be further expressed as follows:

$$\begin{aligned} \text{diff} =& =\text{ReLU}(sign(w_1 \cdot x + b_1)) \bigodot sign(w_2) \bigodot sign(w_1) \\ & \bigodot sign(r) \bigodot |w_2| \cdot |w_1| \cdot |r| \\ =& \text{ReLU}(sign(w_1 \cdot x + b_1)) \bigodot w_2 \cdot w_1 \cdot r \end{aligned} \tag{14}$$

In the ReLU function, because only the positive part is activated, $\text{ReLU}(sign(w_1 \cdot x + b_1)) = 1$ in successive layers; therefore, $\text{diff} = w_2 \cdot w_1 \cdot r$. In this way, we demonstrate that the proposed layer-wise adversarial perturbation still holds after nonlinear activation functions by using, for example,ReLU.

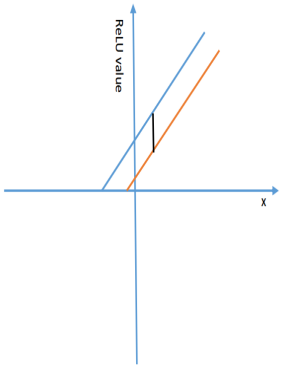


Fig. 1. Simple example of $\text{Relu}_{adv}$ (red line) and $\text{Relu}_{ori}$ (blue line). Their difference is shown as a black line.

### D. Training Procedure of Layer-wise Adversarial Training

During the training procedure, we use the gradients from the previous batch to generate adversarial perturbations to activate the current batch. According to the empirical analysis in [12], it is possible to use random shuffling in a mini-batch setting to ensure that the scheme that generates perturbation for the current mini-batch from the previous mini-batch is a fair approximation to compute adversarial perturbations by using the same mini-batch. In deep CNNs that are composed of both convolutional and fully connected layers, the layer-wise adversarial perturbation in our proposal is added to any intermediate layers based on eq.7 and eq.9, respectively. Assuming that the convolutional layers are stacked over the fully connected layers, the layer perturbation in the last convolutional layer can be passed as an initial value to calculate the perturbation of successive fully connected layers. During testing, $R_{fc}^t$ and $R_{conv}^t$ are not applied. The trained neural networks behave as a standard feed-forward DNN. This training procedure is summarized in algorithm 1.

## IV. Experiments and Evaluation

To understand the effect of the proposed layer-wise perturbation described in the previous section, we compared our method with conventional adversarial training, which manipulates adversarial perturbations only on inputs. The evaluation

[1] for multi-dimensional $x$, the deduction is similar.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

**Algorithm 1** Layer-wise adversarial training procedure

**Input:** $B^t$: batch sampled at iteration t; $\{X^t, Y^t\}$: input-output pairs;
**Output:** adversarially robust DNN model;

Initial $t = 0$;
Sample a batch $\{X^t, Y^t\}$ of size $k$ images from the training data;
Perform regular forward pass while the layer-wise adversarial training is not active for $t = 0$;
Perform backward pass using the classification loss function, and calculate $R^t_{fc}$ (eq.7 ) for fully connected layers and $R^t_{conv}$ (eq.9) for convolutional layers;
**for** $t$ in $1 : |B| - 1$ : **do**
    Sample a batch $\{X^t, Y^t\}$ of size $k$ from the training data;
    Perform a forward pass with perturbation: add $R^t_{fc}$ to the layer output of the fully connected layers, and add $R^t_{conv}$ to the layer output of the convolutional layers;
    Perform a backward pass and update $R^t_{fc}$ and $R^t_{conv}$ to $R^{t+1}_{fc}$ and $R^{t+1}_{conv}$;
**end for**
**Test time usage**: During testing, remove $R^t_{fc}$ and $R^t_{conv}$. The trained neural network behaves as a standard feed-forward DNN.

was conducted by performing image classification tasks using MNIST, CIFAR-10, and CIFAR-100 datasets. All images were normalized to contain pixels in the range [0,1] during training and testing. Both shallow models (4FC and 3Conv2FC) and deep CNNs, such as VGG16 and WideResNet28-10, were employed. For a fair comparison, we do not fine-tuned the models. Training parameters were shared: optimizer=SGD (initial lr=0.1, momentum=0.9), weight decay=0.0001, and epoch=200 with no early stop.

We applied the diverse set of adversarial attack methods that we introduced in Section II, namely FGSM, IFGSM, PGD, EoT, and C&W. All the attacks were conducted in the white-box attack setting, and a value of $0.1$ was assigned to $\epsilon$ for all the attacks. White-box attacks are typically more difficult to defend against than black-box attacks, in that adversaries know all details of the target models and can therefore generate adversarial examples. The evaluation criterion for adversarial robustness is *top-1 worst-case accuracy*.

### A. Shallow DNNs using MNIST dataset

To study the effect of the proposed layer-wise perturbation in detail, we constructed two shallow DNNs: a fully connected network with two hidden layers (4FC) and a CNN with three convolutional layers stacked over two fully connected layers (3Conv2FC).

*1) Experiments on 4FC:* We first studied the choice of $\epsilon$ (in eq.7 and eq.9) using the 4FC model and MNIST dataset. In conventional adversarial training, $\epsilon$ is assigned a small value, which is kept constant, whereas in our experiment, we compared the use of a constant $\epsilon$ $(0.1)$ with a random $\epsilon$ setting, i.e., $\epsilon \sim \mathcal{N}(0, 0.1)$. The results are presented in Table I and indicate that a random $\epsilon$ yields superior results in terms of both the classification accuracy on clean data and adversarial defense performance than a constant $\epsilon$. We conducted additional experiments using MNIST on the constructed 4FC and 3Conv2FC models. The results consistently showed the superiority of using a random $\epsilon$; as a result, $\epsilon$ was assigned a random value in all the following experiments.

TABLE I
PERFORMANCE COMPARISON: USING A CONSTANT $\epsilon$ (-F) VS. A RANDOM $\epsilon$ (-R).

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| one layer-f | 0.9247 | 0.1922 | 0.1649 | 0.1334 | 0.0709 | 0.0532 |
| one layer-r | **0.9256** | **0.3056** | **0.2761** | **0.2411** | **0.1369** | **0.12** |
| two layers-f | 0.9016 | 0.2995 | 0.2601 | 0.2208 | 0.1729 | 0.1297 |
| two layers-r | **0.9297** | **0.4078** | **0.3722** | **0.3236** | **0.2066** | **0.1736** |

TABLE II
DEFENSE PERFORMANCE ON THE 4FC MODEL USING MNIST WITH A SINGLE ADVERSARIAL PERTURBATION LAYER

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| DNN | 0.9429 | 0.1415 | 0.1106 | 0.0819 | 0.0375 | 0.0256 |
| Adv-input | 0.9505 | 0.2151 | 0.1749 | 0.1313 | 0.0787 | 0.0513 |
| FC Layer-2 | 0.9256 | 0.3056 | 0.2761 | 0.2411 | 0.1369 | 0.12 |
| FC Layer-3 | 0.9263 | 0.3552 | 0.323 | 0.2748 | 0.1729 | 0.1426 |
| FC Layer-4 | **0.9693** | **0.5215** | **0.4678** | **0.4055** | **0.3359** | **0.267** |

We studied the effect of applying layer perturbation to only one layer by using the 4FC model on MNIST. Table II compares the results of placing our adversarial perturbation in one of the intermediate layers of a standard DNN without adversarial training (DNN) with those obtained using conventional adversarial training on the input layer (Adv-input). The results show that our proposed layer-wise adversarial training is effective and that it considerably outperforms the conventional adversarial training in terms of the defense performance. An improvement in the adversarial robustness is observed across all the attacks. The results also appear to indicate that the closer the adversarially trained layer is to the network output layer (i.e., FC Layer-4), the more effective the adversarial defense performance.

The results in Table III were obtained by stacking two or more layers and applying layer adversarial perturbation. The

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE III
DEFENSE PERFORMANCE ON THE 4FC MODEL USING MNIST WITH MULTIPLE ADVERSARIAL PERTURBATION LAYERS

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| FC-2&3 | **0.9297** | 0.4078 | 0.3722 | 0.3236 | 0.2066 | 0.1736 |
| FC-3&4 | 0.927 | **0.6415** | **0.6161** | **0.5709** | **0.6243** | **0.5835** |

TABLE IV
DEFENSE PERFORMANCE ON THE 4FC MODEL USING MNIST: OUR LAYER-WISE WITH (MLPW) AND WITHOUT (MLPO) ADVERSARIAL INPUT

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| MLPw | 0.9055 | **0.6492** | **0.6416** | **0.5435** | **0.6255** | **0.5609** |
| MLPo | **0.9125** | 0.6297 | 0.6003 | 0.5237 | 0.5987 | 0.5443 |

results indicate that the defense performance is superior to that when layer perturbation is only applied to one layer, as indicated by Table II. In Table III, FC-2&3 represents the situation in which layer-wise adversarial perturbation was applied to two hidden layers, whereas FC-3&4 represents that in which the perturbation was applied to the output layer and the penultimate hidden layer. Furthermore, as in the aforementioned experiment with a single perturbation layer, the closer the adversarially trained layer is to the network output layer (i.e., FC-3&4), the more effective the adversarial defense performance.

Table IV presents the results we obtained when applying layer-wise perturbation to all layers (incl. the input layer). These results show that the fully connected network 4FC achieves the best adversarial robustness on MNIST dataset.

*2) Experiments on 3Conv2FC:* This subsection details our study on the effect of applying adversarial perturbation to convolutional layers using the constructed 3Conv2FC model on MNIST dataset. Similar to the experiments on the 4FC model, we first investigated the application of layer perturbation to one convolutional layer. This was followed by stacking two and more convolutional layers to which adversarial perturbations were then applied. Next, a combination of convolutional and fully connected layers was subjected to layer-wise adversarial training. Finally, we present the adversarial defense performance resulting from the combination of conventional adversarial training with our layer-wise approach.

Table V compares the results of placing our adversarial perturbations in one of the intermediate convolutional layers with a standard DNN without adversarial training (DNN) and by applying conventional adversarial training to the input layer (Adv-input). Again, it shows that our proposed layer-wise adversarial training is effective and considerably outperforms the conventional adversarial training in terms of the defense performance. Also, in the 3Conv2FC model in which three convolutional layers are stacked over two fully connected layers, the adversarial defense performance does not appear to become more effective when the adversarially trained layers are closer to the network output layer.

Table VI indicates that, when layer adversarial perturbations

TABLE V
DEFENSE PERFORMANCE ON THE 3CONV2FC MODEL USING MNIST WITH A SINGLE ADVERSARIAL PERTURBATION LAYER

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| DNN | 0.9794 | 0.3177 | 0.2652 | 0.2192 | 0.1854 | 0.1156 |
| Adv-input | 0.9792 | **0.4961** | 0.3179 | 0.2492 | 0.2296 | 0.1399 |
| Conv-2 | 0.9775 | 0.4758 | 0.4178 | **0.354** | 0.2594 | **0.1908** |
| Conv-3 | **0.9808** | 0.4367 | **0.3891** | 0.3349 | **0.2808** | 0.1782 |

TABLE VI
DEFENSE PERFORMANCE ON THE 3CONV2FC MODEL USING MNIST WITH ADVERSARIAL PERTURBATION OF MULTIPLE LAYERS

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| Conv-2&3 | 0.98 | 0.5583 | 0.5013 | 0.4277 | 0.3864 | 0.2452 |
| Conv-2&3 +FC-1 | 0.9828 | 0.6866 | 0.6398 | 0.5806 | 0.481 | 0.3766 |
| Conv-2&3 +FC-1&2 | **0.9833** | **0.8096** | **0.7477** | **0.658** | **0.6718** | **0.5097** |

were applied to two or more convolutional layers, the defense performance was superior to that when layer perturbation was applied only to one convolutional layer (Table V). Table VI also compares the situations in which layer-wise adversarial perturbation was applied to the second and third convolutional layers (Conv-2&3), two convolutional layers and the penultimate fully connected layer (Conv-2&3+FC-1), and two convolutional layers and two fully connected layers (Conv-2&3+FC-1&2). These results show that applying adversarial perturbations to both convolutional and fully connected layers significantly improved the adversarial defense performance. With the adversarial perturbations applied on more fully connected layers, the best adversarial robustness can be achieved.

We additionally studied the effect of combining conventional adversarial training with our layer-wise approach on the performance improvement of the 3Conv2FC model. Table VII presents the results of combining conventional adversarial training with our layer-wise perturbation on convolutional layers only (Conv-w) and with all layers (All-w). It also compares these results with those obtained without this combination (Conv-o and All-o). The results are clearly improved when conventional adversarial training was applied to convolutional layers in combination with layer-wise adversarial training. Note that the best defense performance on the 3Conv2FC model was achieved when layer-wise perturbations were applied to all layers except for the input layer, i.e., not in combination with conventional adversarial training.

### *B. Deep CNNs using CIFAR-10 and CIFAR-100 datasets*

To confirm the observations resulting from the empirical analysis of shallow DNNs, we repeated the experiments on deep CNN models, i.e., VGG16 and WideResNet28-10, using CIFAR-10 and CIFAR-100 datasets, respectively. Table VIII compares different placements of our layer-wise adversarial training on VGG16, with standard VGG16 without adversarial

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE VII
DEFENSE PERFORMANCE ON THE 3CONV2FC MODEL USING MNIST: ADVERSARIAL INPUT WITH(-W) AND WITHOUT(-O) LAYER PERTURBATION

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| Conv-o | **0.98** | 0.5583 | 0.5013 | 0.4277 | 0.3864 | 0.2452 |
| Conv-w | 0.9799 | **0.7568** | **0.6325** | **0.4892** | **0.5436** | **0.3237** |
| All-o | **0.9833** | **0.8096** | **0.7477** | **0.658** | **0.6718** | **0.5097** |
| All-w | 0.9823 | 0.7283 | 0.6873 | 0.6297 | 0.5706 | 0.4637 |

TABLE VIII
DEFENSE PERFORMANCE ON VGG16 USING CIFAR-10

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| VGG16 | **0.8655** | 0.213 | 0.175 | 0.097 | 0.113 | 0.083 |
| Adv-input | 0.6127 | 0.2503 | 0.2092 | 0.118 | 0.149 | 0.1013 |
| Conv-all | 0.8515 | 0.3847 | 0.253 | 0.1381 | 0.203 | 0.1221 |
| All-o | 0.8236 | 0.403 | 0.2811 | 0.202 | **0.268** | 0.1934 |
| All-w | 0.8476 | **0.4339** | **0.332** | **0.309** | 0.251 | **0.218** |

training (VGG16) and conventional adversarial training on the input layer (Adv-input). Three variations of the application of our layer adversarial perturbations to VGG16 were compared: application to all convolutional layers (Conv-all), to all layers except for the input layer (All-o), and to all layers including the input layer (All-w). As mentioned in Section III, the adversarial perturbation introduced to each intermediate layer by our method theoretically proves to be equivalent to adversarial manipulation on the network inputs. As a result, applying adversarial perturbations to all layers, including the input layer (i.e., All-w), is equivalent to combining the application of our layer-wise perturbation to all layers except the input layer (i.e., All-o) with conventional adversarial training (i.e., Adv-input). The results in Table VIII show that our proposed layer-wise adversarial training significantly outperforms conventional adversarial training in terms of all the attacks. The best defense performance on the VGG16 model is achieved when applying layer-wise perturbation to all layers including the input layer (All-w), i.e., in combination with conventional adversarial training. In addition, it is noteworthy that the classification accuracy achieved with our proposed layer-wise adversarial training is comparable to that of standard VGG16, whereas conventional adversarial training severely degrades the classification performance. Similar results are presented in Table IX for the WRN28-10 model using CIFAR-100.

TABLE IX
DEFENSE PERFORMANCE ON WIDE-RESNET28-10 USING CIFAR-100

| | Clean | FGSM | IFGSM | PGD | EoT | C&W |
|---|---|---|---|---|---|---|
| WRN28-10 | **0.7615** | 0.1984 | 0.151 | 0.082 | 0.1043 | 0.067 |
| Adv-input | 0.5236 | 0.2006 | 0.1809 | 0.093 | 0.1223 | 0.0732 |
| Conv-all | 0.7115 | 0.3049 | 0.2301 | 0.1107 | 0.1469 | 0.9075 |
| All-o | 0.7235 | 0.3886 | 0.2803 | 0.1901 | 0.2076 | 0.1301 |
| All-w | 0.7523 | **0.4039** | **0.3201** | **0.2343** | **0.2081** | **0.1678** |

## V. CONCLUSION

In this paper, we proposed a layer-wise approach for adversarial training and derived layer adversarial perturbation for both fully connected and convolutional layers. We theoretically proved that the adversarial perturbations introduced in intermediate layers by using our method are equivalent to adversarial manipulation on network inputs, thereby guaranteeing an improvement of the adversarial robustness of DNN models to which the adversarial manipulation is applied. Our proposed method was validated by conducting an empirical evaluation on both shallow and deep CNN models, such as VGG16 and WideResNet28-10, using MNIST, CIFAR-10, and CIFAR-100 datasets. The results consistently showed that our proposed layer-wise adversarial training approach significantly outperforms conventional adversarial training in terms of all mainstream attacks including FGSM, IFGSM, PGD, EoT, and C&W. The best defense performance could be achieved by combining the layer-wise approach with conventional adversarial training. In the future, it would be interesting to investigate the ability of the proposed methods to improve the robustness of a model against generalized corruption noise, in addition to improving the adversarial robustness.

## REFERENCES


[1] 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.

[2] 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.

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

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

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

[6] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[7] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016.

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

[9] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 369–385.

[10] X. Liu, Y. Li, C. Wu, and C. Hsieh, "Adv-bnn: Improved adversarial defense through robust bayesian neural network," *CoRR*, vol. abs/1810.01279, 2018. [Online]. Available: http://arxiv.org/abs/1810.01279

[11] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[12] S. Sankaranarayanan, A. Jain, R. Chellappa, and S. Lim, "Regularizing deep networks using efficient layerwise adversarial training," *CoRR*, vol. abs/1705.07819, 2017. [Online]. Available: http://arxiv.org/abs/1705.07819

[13] A. Liu, X. Liu, C. Zhang, H. Yu, Q. Liu, and J. He, "Training robust deep neural networks via adversarial noise propagation," *ArXiv*, vol. abs/1909.09034, 2019.

[14] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.


978-1-7281-6926-2/20/$31.00 ©2020 IEEE

[15] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 15–26.

[16] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *CoRR*, vol. abs/1802.00420, 2018. [Online]. Available: http://arxiv.org/abs/1802.00420

[17] E. L. I. A. Athalye, A. and K. Kwok, "Synthesizing robust adversarial examples. arxiv preprint," *arXiv:1707.07397*, 2017.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE