Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack

Zhezhi He†, Adnan Siraj Rakin† and Deliang Fan
Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816
† These authors contributed equally
{Elliot.he, AdnanRakin}@knights.ucf.edu, dfan@ucf.edu

Abstract

Recent developments in the field of Deep Learning have exposed the underlying vulnerability of Deep Neural Network (DNN) against adversarial examples. In image classification, an adversarial example is a carefully modified image that is visually imperceptible to the original image but can cause DNN model to misclassify it. Training the network with Gaussian noise is an effective technique to perform model regularization, thus improving model robustness against input variation. Inspired by this classical method, we explore to utilize the regularization characteristic of noise injection to improve DNN's robustness against adversarial attack. In this work, we propose Parametric-Noise-Injection (PNI) [1] *which involves trainable Gaussian noise injection at each layer on either activation or weights through solving the Min-Max optimization problem, embedded with adversarial training. These parameters are trained explicitly to achieve improved robustness. The extensive results show that our proposed PNI technique effectively improves the robustness against a variety of powerful white-box and black-box attacks such as PGD, C & W, FGSM, transferable attack, and ZOO attack. Last but not the least, PNI method* ***improves both clean- and perturbed-data accuracy*** *in comparison to the state-of-the-art defense methods, which outperforms current unbroken PGD defense by 1.1 % and 6.8 % on clean- and perturbed- test data respectively, using ResNet-20 architecture.*

1. Introduction

Deep Neural Networks (DNNs) have achieved great success in a variety of applications, including but not limited to image classification [23], speech recognition [17], machine translation [5], and autonomous driving [9]. Despite the remarkable accuracy improvement [15], recent studies [40, 14, 7] have shown that DNNs are vulnerable to adversarial examples. In the image classification task, an adversarial example is a natural image which is intentionally perturbed by visually imperceptible variation but can cause drastic classification accuracy degradation. In addition to image classification, attacks to other DNN-powered tasks have also been actively investigated, such as visual question answering [43, 1], image captioning [10], semantic segmentation [31, 1] and etc [12, 8, 39].

There has been a cohort of works on adversarial example generation (aka., adversarial attack) and developing corresponding defense methods. In general, adversarial attacks can be categorized as *white-box attack* and *black-box attack* based on the information of target model exposed to the attacker. For white-box attack [40, 7], the adversary has full access to the network architecture and parameters. Whereas, only external access to the network (e.g., input and output) are permitted for the black-box attacks [29, 33, 11]. Owing to the richer information, white-box attack can often achieve higher attacking success rates in comparison to the black-box counterpart, for various applications [40, 24, 21, 35, 32, 4, 11, 7].

Recently, different works [6] have viewed the problem of adversarial examples from a unified perspective of model robustness and regularization. Conventional regularization mainly serves the purpose of generalization, thus preventing the model from over-fitting to the training data. Traditional regularization methods, such as dropout [38], Batch Normalization [18], have been demonstrated effective and widely used in practice. Hinton also discusses that adding Gaussian noise into the model (e.g., input, weight, and activation) during training which performs as a regularizer in his lecture note [13] and dropout work [38].

It is evident that a proper model regularization method, which specifically designed for improving DNN robustness, can serve the purpose of defending adversarial example more effectively. Recently, different works [27, 26, 6] have adopted the noise injection method for model regularization but configures the injected noise manually. In contrast to

[1] Our *Pytorch* implementation is publicly available at https://github.com/elliothe/CVPR_2019_PNI

that, we take advantage of regularization property of noise injection while optimizing the magnitude of injected noise through the end-to-end training.

Overview of our approach: In this work, we propose a novel noise injection method called Parametric Noise Injection (PNI) to improve the neural network robustness against adversarial attack. The proposed PNI technique is to apply to inject layer-wise trainable Gaussian noise on various locations, including network input/activation/weights. For each inference, the injected noise is independently sampled from the corresponding Gaussian distribution, where its mean and variance of this distribution is trained by gradient descent method as other parameters of DNN. To achieve the proper objective of the training, PNI is embedded with well-known adversarial training, where the injected noise (i.e., its mean and variance) will be optimized through end-to-end training instead of manual configuration. In general, our proposed PNI technique shows tempting performance improvement on both clean- and perturbed-data accuracy, in comparison to the vanilla adversarial training. The negligible model capacity and computational overhead make PNI a promising solution for practical applications.

2. Related works

2.1. Adversarial Attack

Recently, various powerful adversarial attack methods have been proposed to totally fool a trained DNN through introducing barely visible perturbation upon input data. Several state-of-the-art white-box (i.e., PGD [30], FGSM [14] and C&W [7]) and black-box (i.e., Substitute [34] and ZOO [11]) adversarial attack methods, which will be investigated in this work, are briefly introduced as follows.

FGSM Attack: Fast Gradient Sign Method (FGSM) [40] is an efficient single-step adversarial attack method. Given vectorized input $\boldsymbol{x}$ and corresponding target label $\boldsymbol{t}$, FGSM alters each element x of $\boldsymbol{x}$ along the direction of its gradient w.r.t the inference loss $\partial\mathcal{L}/\partial x$. The generation of adversarial example $\hat{\boldsymbol{x}}$ (i.e., perturbed input) can be described as:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \cdot sgn\big(\nabla_{\boldsymbol{x}}\mathcal{L}(g(\boldsymbol{x};\boldsymbol{\theta}),\boldsymbol{t})\big) \tag{1}$$

where ϵ is the perturbation constraint that determines the attack strength. $g(\boldsymbol{x};\boldsymbol{\theta})$ computes the output of DNN paramterized by $\boldsymbol{\theta}$. $sgn(\cdot)$ is the sign function. Note that, the attack is followed by a clipping operation to ensure the $\hat{x} \in [0,1]$.

PGD Attack: Projected Gradient Descent (PGD) [30] is a multi-step variant of FGSM, which is one of the strongest L^∞ adversarial example generation algorithm. With $\hat{\boldsymbol{x}}^{k=1} = \boldsymbol{x}$ as the initialization, the iterative update of perturbed data $\hat{\boldsymbol{x}}$ in k-th step can be expressed as:

$$\hat{\boldsymbol{x}}^{k} = \Pi_{P_\epsilon(\boldsymbol{x})}\Big(\hat{\boldsymbol{x}}^{k-1} + a \cdot sgn\big(\nabla_{\boldsymbol{x}}\mathcal{L}(g(\hat{\boldsymbol{x}}^{k-1};\boldsymbol{\theta}),t)\big)\Big) \tag{2}$$

where $P_\epsilon(\boldsymbol{x})$ is the projection space which is bounded by $\boldsymbol{x} \pm \epsilon$, and a is the step size. Madry et al. [30] also propose that PGD is a universal adversary among all the first-order adversaries (i.e., attacks only rely on first-order information).

C&W Attack: Recently, Carlini and Wagner propose an attack method called C&W attack [7]. C&W attack considers the generation of adversarial example as a problem of optimizing the L^p-norm of distance metric δ w.r.t the given input data $\boldsymbol{x}$, which can be described as:

$$||\boldsymbol{\delta}||_p = \Big(\sum_{i=1}^{n}|\delta_i|^p\Big)^{1/p}; \quad \delta_i = \hat{x}_i - x_i \tag{3}$$

$$\text{minimize } ||\boldsymbol{\delta}||_p + c\cdot\mathcal{L}(\boldsymbol{x}+\boldsymbol{\delta}) \quad s.t. \quad \boldsymbol{x}+\boldsymbol{\delta} \in [0,1]^n \tag{4}$$

where $\boldsymbol{\delta}$ is taken as the perturbation added upon the input $\boldsymbol{x}$, and a specific loss function $\mathcal{L}$ is chosen in [7] to solve the optimization problem via gradient descent. c is a constant set by the attacker. In this work, we use L^2-norm based C&W attack and take $||\boldsymbol{\delta}||_{p=2}$ as the evaluation metric to measure the robustness of DNN, where a greater value of $||\boldsymbol{\delta}||_{p=2}$ normally indicates a DNN possesses higher robustness against potential adversarial attacks.

Black-box Attacks: The most popular black-box attack is conducted using a substitute model [34], which is trained using the output of target model as the label, to mimic the functionality of the target model. Then, the adversarial example generated from the substitute model is used to perform the attack on the target model. In this work, we specifically investigate the transferable adversarial attack [29], which is a variant of substitute model attack [34]. In the transferable adversarial attack, the adversarial example is generated from one source model to attack another target model. The source model and target can own a totally different structure but trained with the real training data. Beyond that, Zero-th Order Optimization (ZOO) attack [11] is investigated in this work as well. Rather than using the substitute model to approximate the gradients of the target model to perform an attack, ZOO attack directly approximates the gradient based on the input data and output scores using stochastic gradient coordinate.

2.2. Adversarial Defenses:

Improving network robustness via adversarial training [40, 30] is by far the most popular and unbroken defense approach. The key idea of adversarial training is to take

the adversarial example as the training data, which trains the DNN against the adversarial attack. Most of the later works [19, 36] have followed this path to supplement their defense with adversarial training. The initial and important step in adversarial training is to choose an attack model for adversarial example generation. Adopting Projected Gradient Descent (PGD) [30] as an attack model for adversarial training is becoming popular, since it is considered to be capable of generating universal adversarial examples among the first order approaches [30]. Additionally, among many recent defense methods, only PGD based adversarial training can sustain the state-of-the-art accuracy under various other attacks [7, 40, 4].

Recent work [6] has merged the concept of improving model robustness through regularization to defend adversarial examples. A well-known method for model regularization is noise injection, which is a variant of dropout on weights [41] or activations [38]. For further improving the performance of DNN under attack, there are works attempt to introducing randomness into DNN for adversarial defense, such as randomly pruning some activation during the inference [37], randomizing the input layer [42], inserting a noise-layer right before the convolution layers [27, 26]. However, the performance improvement (i.e., perturbed-data accuracy) mainly comes from the stochastic gradient instead of regularizing the DNN for better robustness, which is considered as the broken defense method according to the criterion of gradient obfuscations [4]. An alternative and straight-forward approach to evaluating adversarial defense about the gradient obfuscation is to examine the clean- (attack free) and perturbed-data (under attack) accuracy. If the adopted method mainly performs the model regularization, it is expected to improve the perturbed-data accuracy without sacrificing the clean-data accuracy.

Last but not least, we also notice that recent work Adv-BNN [28] also combines the adversarial training and noise injection on weight (i.e., Bayesian neural network equivalently). In comparison to our proposed PNI, Adv-BNN [28] mainly has the following drawbacks: 1) Significant computational and storage overhead owing to the used weight posterior (double model size) and output ensemble (>10 times), and 2) potential gradient obfuscation (trade the clean-data accuracy with perturbed-data accuracy). The key factor that our PNI outperforms Adv-BNN is the layer-wise noise injection (Eq. (5)) and ensemble loss function (Eq. (10)), which will be explicitly introduced in the following sections.

3. Parametric Noise Injection

In this section, we first introduce the proposed Parametric Noise Injection (PNI) function and will investigate the impact of noise injection on input/weight/activation.

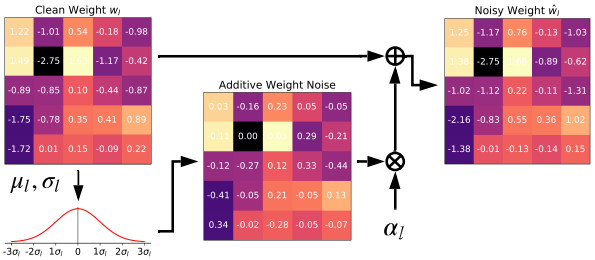

Figure 1. The flowchart of PNI on the weights $\boldsymbol{w}_l$ of a 5×5 fully-connected layer (i.e., PNI-W). For each inference, the process of PNI on $\boldsymbol{w}_l$ can be generally divided into three steps: 1) statistically calculate the standard deviation σ_l of $\boldsymbol{w}_l$; 2) sample the additive weight noise (*i.i.d*) from $\mathcal{N}(0, \sigma_l^2)$; 3) added the scaled weight noise with clean weight, then use the noisy weight $\hat{\boldsymbol{w}}_l$ in the forward path.

Definition. The method that we propose to inject Gaussian noise to different components or locations within DNN can be mathematically described as:

$$\tilde{v}_{l,i} = f_{\text{PNI}}(v_{l,i}) = v_{l,i} + \alpha_l \cdot \eta_{l,i}; \quad \eta_{l,i} \sim \mathcal{N}(0, \sigma_l^2) \quad (5)$$

where $v_{l,i}$ is the element of noise-free tensor $\boldsymbol{v}_l$ in l-th layer of DNN, and such $\boldsymbol{v}_l$ can be input/weight/inter-layer (i.e., activation) tensor in this work. $\eta_{l,i}$ is the noise term which samples from Gaussian distribution with zero mean and variance σ_l^2, for each inference. α_i is the coefficient that scales the magnitude of injected noise η_l. Note that, we adopt the scheme that η_l shares the identical variance with $\boldsymbol{v}$ as in Eq. (5), thus the injected additive noise relies on α_l and the distribution of $\boldsymbol{v}_l$ simultaneously.

In this work, rather than manually configuring α_l to restrict the noise level, we set α_l as a learnable parameter which can be optimized for network robustness improvement. We name such method as *Parametric Noise Injection* (PNI). Assuming we perform the proposed PNI on the weight tensors of convolution/fully-connected layers throughout entire DNN, for each parametric layer there is only one layer-wise noise scaling coefficient (α_l) to be optimized. We take such layer-wise PNI configuration as default in this work. A example of PNI on weight (i.e., PNI-W) is depicted in Fig. 1.

Optimization In this work, we treat the noise scaling coefficient as a model parameter which can be optimized through the back-propagation. For $f_{\text{PNI}}(\cdot)$ that shares the noise scaling coefficient layer-wise, the gradient computation can be described as:

$$\frac{\partial \mathcal{L}}{\partial \alpha_l} = \sum_i \frac{\partial \mathcal{L}}{\partial f_{\text{PNI}}(v_{l,i})} \frac{\partial f_{\text{PNI}}(v_{l,i})}{\partial \alpha_l} \quad (6)$$

where the $\sum_i$ takes the summation over the entire tensor $\boldsymbol{v}_{l,i}$, and $\partial \mathcal{L} / \partial f_{\text{PNI}}(v_{l,i})$ is the gradient back-propagated

from the followed layers. The gradient calculation of the PNI function is:

$$\frac{\partial f_{\text{PNI}}(v_{l,i})}{\partial \alpha_l} = \eta_{l,i} \tag{7}$$

It is noteworthy that the random sampled $\eta_{l,i}$ will be taken as a constant during the back-propagation. Using the gradient descent optimizer with momentum, the optimization of α at step j can be written as:

$$V_l^j = m \cdot V_l^{j-1} + \frac{\partial \mathcal{L}^{j-1}}{\partial \alpha_l}; \quad \alpha_l^j = \alpha_l^{j-1} - \epsilon \cdot V_l^j \tag{8}$$

where m is the momentum, ϵ is the learning rate, and V_l is the updating velocity. Moreover, since weight decay tends to make the learned noise scaling coefficient converge to zero, there is no weight decay term on the α during the parameter updating in this work. We set $\alpha_l = 0.25$ as default initialization.

Robust Optimization. We expect to utilize the aforementioned PNI technique to improve the network robustness. However, directly optimizing the noise scaling coefficient normally leads α_l to converge at a small close-to-zero value (vanilla training in Table 1), owing to the gradient-descent optimizer tends to make the weights to be noise-free thus to over-fit the training data.

In order to succeed in adversarial defense, we jointly use the PNI method with robust optimization (a.k.a. *Adversarial Training*) which can boost the inference accuracy for the perturbed data under attack. Given inputs- $\boldsymbol{x}$ and target labels- t, the adversarial training is to obtain the optimal solution of network parameter $\boldsymbol{\theta}$ for the following min-max problem:

$$\arg\min_{\boldsymbol{\theta}} \Big\{ \arg\max_{\boldsymbol{x}' \in P_\epsilon(\boldsymbol{x})} \mathcal{L}\big(g(\hat{\boldsymbol{x}}; f_{\text{PNI}}(\boldsymbol{\theta})), \boldsymbol{t}\big) \Big\} \tag{9}$$

where the inner maximization tends to acquire the perturbed data $\hat{\boldsymbol{x}}$, and $P_\epsilon(\boldsymbol{x})$ is the input data perturb set constrained by ϵ. While the outer minimization is optimized through gradient descent method as regular network training. L^∞ PGD attack [30] is adopted as the default inner maximization solver (i.e., generating $\hat{\boldsymbol{x}}$).

Moreover, in order to balance the clean data accuracy and perturbed data accuracy for practical application, rather than performing the outer minimization solely on the loss of perturbed data as in Eq. (9), we minimize the ensemble loss $\mathcal{L}'$ which is the weighted sum of losses for clean- and perturbed-data. The ensemble loss $\mathcal{L}_{\text{ens}}$ is described as:

$$\mathcal{L}_{\text{ens}} = w_c \cdot \mathcal{L}(g(\boldsymbol{x}; f_{\text{PNI}}(\boldsymbol{\theta})), \boldsymbol{t}) + w_a \cdot \mathcal{L}(g(\hat{\boldsymbol{x}}; f_{\text{PNI}}(\boldsymbol{\theta})), \boldsymbol{t}) \tag{10}$$

where w_c and w_a are the weights for clean data loss term and adversarial data loss term, respectively. $w_c = w_a = 0.5$ is the default configuration in this work.

Optimizing the ensemble loss $\mathcal{L}_{\text{ens}}$ is the key to successful training of both the model's inherent parameter (e.g. weight, bias) and the add-on noise scaling coefficient α_l from PNI. The intuition behind is that, the gradient-descent optimizer attempt find a equilibrium point of α_l when minimizing $\mathcal{L}_{\text{ens}}$. If α_l is too large, PNI will introduce significant noise into the inference path which will definitely hamper the accuracy for both clean- and perturbed-data. If α_l is too small, PNI will not perform any regularization.

4. Experiments

4.1. Experiment setup

Datasets and network architectures. Two visual datasets for object recognition task is considered in this work, which is MNIST and CIFAR-10. The MNIST [25] dataset is a set of handwritten digit 28×28 gray-scale images with 60K training examples and 10K test examples. The CIFAR-10 [22] dataset is composed of 50K training samples and 10K test samples of 32×32 color image. There is no data augmentation used for MNIST, while CIFAR-10 use the same augmentation method as in [16]. Although we test our method on both CIFAR-10 and MNIST, we mainly present results on CIFAR-10 to validate our method. Since the results on MNIST cannot provide much insight, we put the MNIST results in the appendix.

For MNIST, we test the performance using the variant LetNet5. For CIFAR-10, the classical Residual Networks [16] (ResNet-20/32/44/56) architecture are used, and ResNet-20 is taken as the baseline for most of the comparative experiments and ablation studies. A redundant network ResNet-18 is also used to report the performance for CIFAR-10, since large network capacity is helpful for adversarial defense. Moreover, rather than including the input normalization within the data augmentation, we place a non-trainable data normalization layer in front of the DNN to perform the identical function, thus an attacker can directly add the perturbation on the natural image. Note that, since both PNI and PGD attack [30] include randomness, we report the accuracy in the format of mean±std% with 5 trials to alleviate error.

Adversarial attacks. To evaluate the performance of our proposed PNI technique, we employ multiple powerful white-box and black-box attacks as introduced in Section 2.1. For PGD attack on MNIST and CIFAR-10, ϵ is set to 0.3/1 and 8/255, and N_{step} is set to 40 and 7 respectively. FGSM attack adopts the same ϵ setup as PGD. The attack configurations of PGD and FGSM are identical as the setup in [19, 30]. For C&W attack, we set the constant c as 0.01. ADAM [20] is used to optimize the Eq. (4) with learning rate as $5e^{-4}$. We choose 0 for the confidence coefficient k, which is defined in the loss function used by C&W L^2 at-

Table 1. **Convergence of PNI**: ResNet-20 with Layerwise weight PNI on CIFAR-10 dataset. (Top) The converged layer-wise noise scaling coefficient α under various training scheme. (Bottom) Test accuracy for clean- and perturbed-data under PGD and FGSM attack.

Layer Index	Vanilla Traning	PNI-W+Adv. Train. (without PNI in $\hat{\boldsymbol{x}}$ generation)	PNI-W+Adv. Train. (with PNI in $\hat{\boldsymbol{x}}$ generation)
Conv0	0.003	0.004	**0.146**
Conv1.0	0.002	0.005	**0.081**
Conv1.1	0.004	0.004	**0.049**
Conv1.2	0.002	0.001	**0.097**
Conv1.3	0.004	5.856	**0.771**
Conv1.4	0.005	0.005	0.004
Conv1.5	0.002	0.001	0.006
Conv2.0	0.004	0.000	0.006
Conv2.1	0.006	0.003	0.004
Conv2.2	0.004	0.003	0.030
Conv2.3	0.001	0.006	0.003
Conv2.4	0.003	0.001	0.033
Conv2.5	0.002	0.001	0.023
Conv3.0	0.007	0.001	0.008
Conv3.1	0.003	0.001	0.006
Conv3.2	0.007	0.002	0.001
Conv3.3	0.006	0.001	0.002
Conv3.4	0.009	0.002	0.001
Conv3.5	0.005	0.000	0.001
FC	0.002	0.002	0.001
Clean	92.11%	71.00%	84.89±0.11%
PGD	0.00±0.00%	18.11%	45.94±0.11%
FGSM	14.08%	26.34%	54.48±0.44%

tack in [7]. The binary search steps for the attack is 9, while number of iteration to perform the gradient descent is 10. Moreover, we also conduct the PNI defense against several state-of-the-art black-box attacks (i.e. substitute [34], ZOO [11] and transferable [29] attack) in a Section 4.2.2 to examine the robustness improvement resulted from the proposed PNI technique.

Competing methods for adversarial defense. As far as we know, the adversarial training with PGD [30] is the only unbroken defense method [4], which is labeled as *vanilla adversarial training* and taken as the baseline in this work. Beyond that, several recent works utilizing similar concept as ours in their defense method are discussed as well, including certified robustness [26], random self-ensemble [27], and Adv-BNN [28].

4.2. PNI for adversarial attacks

4.2.1 PNI against white-box attacks

Optimization method of PNI As the discussion at the end of Section 3, the noise scaling coefficient will not be properly trained without utilizing the adversarial training and ensemble loss. We conduct the experiments for training the layer-wise PNI on weight (PNI-W) of ResNet-20, to compare the convergence of trained noise. As tabulated in Table 1, simply performing the vanilla training using momentum SGD optimizer totally fails the adversarial defense, where the noise scaling coefficients α are converged to the negligible values. On the contrary, with the aid of adversarial training (i.e., optimization of Eq. (10)), convolution layers in the network's front-end have obtained relatively large α which are the bold values in Table 1, and the corresponding evolution curve are shown in Fig. 2.

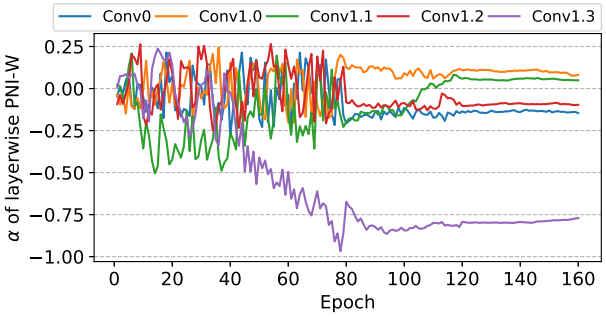

Figure 2. The evolution curve of trainable noise scaling coefficient α for layerwise PNI on weight (PNI-W). Only front 5 layers (bold in Table 1) of ResNet-20 [16] are shown. The learning rate of SGD optimizer is reduced at 80 and 120 epoch.

Since the PGD attack [30] is taken as the inner maximization solver, the generation of adversarial example $\hat{\boldsymbol{x}}$ in Eq. (2) is reformatted as:

$$\hat{\boldsymbol{x}}^{t+1} = \Pi_{P_\epsilon(\boldsymbol{x})}\Big(\hat{\boldsymbol{x}}^t + a \cdot sgn\big(\nabla_{\boldsymbol{x}}\mathcal{L}(g(\hat{\boldsymbol{x}}^t; f_{\text{PNI}}(\boldsymbol{\theta})), \boldsymbol{t})\big)\Big) \quad (11)$$

where the difference between Eq. (2) and Eq. (11) is with/without PNI in $\hat{\boldsymbol{x}}$ generation. It is noteworthy that, keeping the noise term in the model for both adversarial example generation (Eq. (11)) and model parameter update is also the critical factor for the PNI optimization with adversarial training. As listed in Table 1, not incorporating the PNI-W in $\hat{\boldsymbol{x}}$ generation indeed leads to the failure of PNI optimization, and the large value ($\alpha = 5.856$ in Table 1) is not converged due to the probable gradient explosion.

Effect of PNI on weight, activation and input. In this work, even though the scheme of injecting noise on the weight (PNI-W) is taken as the default PNI setup, more results about PNI on activation (PNI-A-a/b), input (PNI-I) and hybrid-mode (e.g. PNI-W+A) are provided in Table 2 for a comprehensive study. PNI-A-a/PNI-A-b denotes injecting noise on the output/input tensor of the convolution/fully-connected layer respectively. Moreover, PNI-A-b scheme intrinsically includes the PNI-I, since PNI-I is applying the noise on the input tensor of the first layer. Note that, all models with PNI variants are jointly trained with PGD-based adversarial training [30] as discussed above.

Table 2. **Effect of PNI location**: The ResNet-20 [16] clean- and perturbed-data (under PGD and FGSM attack) accuracy (mean±std%) on CIFAR-10 test-set, with PNI technique on different network location. Baseline is the ResNet-20 with vanilla adversarial training, and all the PNI combinations are optimized through adversarial training by default.

	Test with PNI			Test without PNI		
	Clean	PGD	FGSM	Clean	PGD	FGSM
Vanilla adv. train [30]	-	-	-	83.84	39.14±0.05	46.55
PNI-W	84.89±0.11	**45.94±0.11**	**54.48±0.44**	85.48	31.45±0.07	42.55
PNI-I	85.10±0.08	43.25±0.16	50.78±0.16	84.82	34.87±0.05	44.07
PNI-A-a	85.22±0.18	43.83±0.10	51.41±0.08	85.20	33.93±0.05	44.32
PNI-A-b	84.66±0.16	43.63±0.20	51.26±0.09	83.97	33.53±0.05	43.37
PNI-W+A-a	85.12±0.10	43.57±0.12	51.15±0.21	84.88	33.23±0.05	43.59
PNI-W+A-b	84.33±0.11	43.80±0.19	51.14±0.07	84.42	33.30±0.05	43.43

Table 3. **Effect of network depth and width**: The clean- and perturbed-data (under PGD and FGSM attack) accuracy (mean±std%) on CIFAR-10 test-set, utilizing different robust optimization configurations. For network depth, the classical ResNet-20/32/44/56 with increasing depth is reported. For network width, the ResNet-20 (1×) is adopted as the baseline, then we compare the wide ResNet-20 with the input and output channel scaled by 1.5×/2×/4×. Capacity denotes the number of trainable parameters in the model.

		No defense			Vanilla adv. train			PNI-W+adv. train (Test with PNI)			PNI-W+adv. train (Test without PNI)		
Model	Capacity	Clean	PGD	FGSM	Clean	PGD	FGSM	Clean	PGD	FGSM	Clean	PGD	FGSM
Net20	269,722	92.1	0.0±0.0	14.1	83.8	39.1±0.1	46.6	84.9±0.1	45.9±0.1	54.5±0.4	85.5	31.6±0.1	42.6
Net32	464,154	92.8	0.0±0.0	17.8	85.6	42.1±0.0	50.3	85.9±0.1	43.5±0.3	51.5±0.1	86.4	35.3±0.1	45.5
Net44	658,586	93.1	0.0±0.0	23.9	85.9	40.8±0.1	48.2	84.7±0.2	48.5±0.2	55.8±0.1	86.0	39.6±0.1	49.9
Net56	853,018	93.3	0.0±0.0	24.2	86.5	40.1±0.1	48.8	86.8±0.2	46.3±0.3	53.9±0.1	87.3	41.6±0.1	51.1
Net20(1.5×)	605,026	93.5	0.0±0.0	15.9	85.8	42.0±0.0	49.6	86.0±0.1	46.7±0.2	54.5±0.2	87.0	38.4±0.1	49.1
Net20(2×)	1,073,962	94.0	0.0±0.0	13.0	86.3	43.1±0.1	52.6	86.2±0.1	46.1±0.2	54.6±0.2	86.8	39.1±0.0	50.3
Net20(4×)	4,286,026	94.0	0.0±0.0	14.2	87.5	46.1±0.1	54.1	87.7±0.1	49.1±0.3	57.0±0.2	88.1	43.8±0.1	54.2

Then, with the same trained model, we report the accuracy with/without the trained noise term (left/right in Table 2) during the test phase. As shown in Table 2, with the noise term enabled during the test phase, PNI-W on ResNet-20 gives the best performance to defend PGD and FGSM attack, in comparison to PNI on other locations. Although it is elusive to fully understand the mechanism that PNI-W outperforms other counterparts, the intuition is that PNI-W is the generalization of PNI-A in each connection instead of each output unit, similar as the relation between the regularization technique DropConnect [41] and Dropout [38].

Furthermore, we also observe that disabling PNI during test phase leads to significant accuracy drop for defending PGD and FGSM attack, while the clean-data accuracy maintains the same level as PNI enabled. Such observation raises two concerns about our PNI techniques: 1) Does the improvement of clean-/perturbed-data accuracy with PNI mainly comes from the attack strength reduction caused by the randomness (potential gradient obfuscation [4])? 2) Is PNI just a negligible trick or it performs the model regularization to construct a more robust model? Our answers to both questions are negative, where the explanations are elaborated under Section 5.

Effect of network capacity. In order to investigate the relation between network capacity (i.e., number of trainable parameters) and robustness improvement by PNI, we examine various network architectures in terms of both depth and width. For different network depths, experiments on ResNet 20/32/44/56 [16] are conducted under vanilla adversarial training [30] and our proposed PNI robust optimization method. For different network widths, we adopt the original ResNet-20 as a baseline and expand its input&output channel of each layer by 1.5×/2×/4× respectively. Same as Table 2, we report clean- and perturbed-data accuracy with/without PNI term during the test phase. The results in Table 3 indicates that increasing the model's capacity indeed improves network robustness against white-box adversarial attacks, and our proposed PNI outperforms vanilla adversary training in terms of both clean-data accuracy and perturbed data accuracy for PGD and FGSM attack. Such observation demonstrates that the perturbed-data accuracy improvement does not come from trading off clean-data accuracy as reported in [19, 2]. Through increasing the network capacity, the robustness improvement results from the proposed PNI becomes less significant. Although both adversarial training and PNI techniques perform regularization, the network structure still needs careful construction to prevent the over-fitting resulted from over-

Table 4. C & W attack L_2 norm comparison

Model	capacity	CW L2-norm		
		No defense	Vanilla adv. train	PNI-W
ResNet-20 (4x)	4,286,026	0.12	1.97	1.95±0.02
ResNet-18	11,173,962	0.12	2.39	2.62±0.04

parameterization.

Robustness evaluation with C&W attack. Improved robustness does not necessarily mean improving the test data accuracy against any particular attack method. Typically L_2 norm based C & W attack [7] should reach 100 % success rate against any defense. Thus average L_2 norm required to fool the network gives more insight about a network's robustness in general [7]. The result presented in Table 4 represents the overall performance of our model against C & W attack. Our method of training the noise parameter becomes more effective for a more redundant network. We demonstrate this phenomenon by performing comparison study between ResNet-20 and ResNet-18 architecture. Clearly, ResNet-18 shows the improvement in robustness from Vanilla adv. training much more than ResNet-20 against C & W attack.

4.2.2 PNI against black-box attack

In this section, we test our proposed PNI technique against transferable adversarial attack [29] and ZOO attack. Following the transferable adversarial attack [29], two trained neural network is taken as the source model (S) and target model (T). The adversarial examples $\hat{\boldsymbol{x}}_s$ is generated from the source model then attack the target model using $\hat{\boldsymbol{x}}_s$, which is denoted as $S \Rightarrow T$. We take ResNet-18 on CIFAR-10 as an example. We train two ResNet-18 model (model-A and B) on CIFAR-10 dataset to attack each other, where model-A is optimized through vanilla adversarial training, while model-B is trained using our proposed PNI variants (i.e., PNI-W/A-a/W+A-a) robust optimization method. Table 5 shows almost equal perturbed-data accuracy for A $\Rightarrow$ B and B $\Rightarrow$ A under various PNI scenarios, which indicates that the presence of PNI during the inference has negligible effect on the attack strength of PGD.

For ZOO attack[11], we test our defense on 200 randomly selected test samples for an untargeted attack. The Attack success rate denotes the percentage of test sample change their classification to a wrong class after the attack. ZOO attack success rate for vanilla Resnet-18 with adversarial training is close to 80 %. The robustness of PNI is more evident from Table 5 as the attack success rate drops significantly for PNI-W+A-a and PNI-W. However, PNI-A-a fails to resist ZOO attack even though it still maintains a lower success rate than the baseline. The failure of PNI-A-a shows that just adding noise in-front of the activation does not necessarily achieves the desired robustness as claimed by some of the previous defenses [26, 27].

Table 5. **PNI against black-box attacks**: On CIFAR-10 test-set, (Left) perturbed-data accuracy under transferable PGD attack, and (Right) the attack success rate for ZOO attack. Model-A is a ResNet-18 trained by vanilla adversarial training, and Model-B is a ResNet-18 trained by PNI-W/A-a/W+A-a with adversarial training.

Train. scheme of B	Transferable attack		ZOO attack
	A $\Rightarrow$ B	B $\Rightarrow$ A	success rate
PNI-W	75.13±0.17	75.23±0.18	57.72
PNI-A-a	74.67±0.11	75.86±0.13	69.61
PNI-W+A-a	75.14±0.10	74.92±0.13	50.00

4.2.3 Comparison to competing methods

As discussed in Section 2.2, a large number of adversarial defense works have been proposed recently, however, most of them are already broken by stronger attacks proposed in [3, 4]. As a result, in this work we choose to compare with the most effective one till date - PGD based adversarial training [30]. Additionally, we compare with other randomness-based works [27, 26, 28] in Table 6 for examining the effectiveness of PNI.

Table 6. Comparison of state-of-the-art adversarial defense methods with clean- and perturbed-data accuracy on CIFAR-10 under PGD attack.

Defense method	model	Clean	PGD
PGD adv. train [30]	ResNet-20 (4×)	87	46.1±0.1
DP [26]	28-10 Wide ResNet (L=0.1)	87.0	25
RSE [27]	ResNext	87.5	40
Adv-BNN [28]	VGG-16	79.7	45.4
PNI-W (this work)	ResNet-20 (4×)	87.7±0.1	49.1±0.3

Previous defense works [2, 19] have shown a trade-off between clean-data accuracy and perturbed-data accuracy, where the perturbed-data accuracy improvement achieved at the cost of lowering the clean-data accuracy. It is worthy to highlight that **our proposed PNI improves both clean- and perturbed data accuracy under white-box attack, in comparison to PGD-based adversarial training** [30]. Differential Privacy (DP) [26] is a similar method of utilizing noise injection at various locations in the network. Although their defense guarantees a certified defense, it does not perform well against L_∞-norm based attacks (e.g., PGD and FGSM). Moreover, in order to achieve a higher level of certified defense, DP significantly sacrifices the clean-data accuracy as well. Another randomness-based approach is Random Self-ensemble (RSE) [27], which inserts noise-layer before all the convolution layer. Even though their defense performs well against C&W attack but poor against

strong PGD attack. Beyond that, both DP and RSE manually configure the noise level which is extremely difficult to find the optimal setup. Whereas, in our proposed PNI method, the noise level is determined by a trainable layer-wise noise scaling coefficient and distribution of weight at noise injected location. For Adv-BNN [28], besides the computational overhead and model size ($>$ 20 times), our PNI also outperforms it in terms of performance on both clean- and perturbed-data.

5. Discussion

The defense performance improvement led by our proposed PNI does not come from the stochastic gradients. The stochastic gradient is considered to incorrectly approximate the true gradient based on a single sample. We try to show that PNI is not relying on the gradient obfuscation from two perspectives: 1) Our proposed PNI method passes each inspection item proposed by [4] to identify gradient obfuscation. 2) Under PGD attack, through increasing the attack steps, our PNI robust optimization method still outperforms vanilla adversarial training (certified as non-obfuscated gradients in [4]).

Table 7. Checklist of examining the characteristic behaviors caused by obfuscated and masked gradient [4] for PNI.

Characteristics to identify gradient obfuscation	Pass	Fail
1. One-step attack performs better than iterative attacks	✓	
2. Black-box attacks are better than white-box attacks	✓	
3. Unbounded attacks do not reach 100% success	✓	
4. Random sampling finds adversarial examples	✓	
5. Increasing distortion bound doesn't increase success	✓	

Inspections of gradient obfuscation. The famous gradient obfuscation work [4] enumerates several characteristic behaviors as listed in Table 7 which can be observed when the defense method owns gradient obfuscation. Our experiments show that PNI passes each inspection item in Table 7.

For item.1, all the experiments in Table 2 and Table 3 report that FGSM attack (one-step) performs worse than PGD attack (iterative). For item.2, our black-box attack experiment in Table 5 shows that the black-box attack strength is worse than a white-box attack. For items.3, as plotted in Fig. 3, we run experiments through increasing the distortion bound-ϵ. The result shows that the unbounded attacks do lead to 0% accuracy under attack. For item.4, the prerequisite is the gradient-based attack (e.g., PGD and FGSM) cannot find the adversarial examples, however the experiments in Fig. 3 reveals that our method still can be broken when increasing the distortion bound. It just increases the resistance against the adversarial attacks, in comparison to the vanilla adversarial training. For item.5, again as shown in Fig. 3, increasing the distortion bound increase the attack success rate.

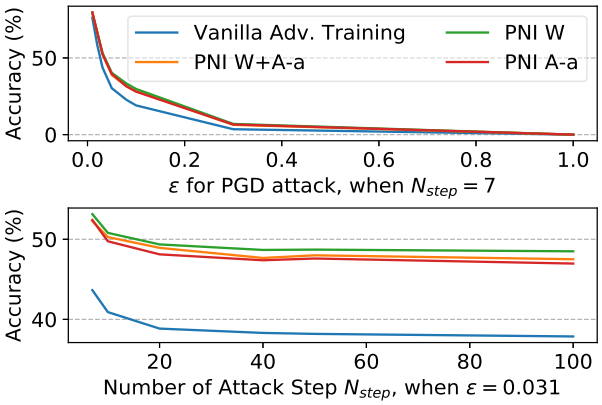

Figure 3. On CIFAR-10 test set, the perturbed-data accuracy of ResNet-18 under PGD attack (Top) versus attack bound ϵ, and (Bottom) versus number of attack steps N_{step}

PNI does not rely on stochastic gradients. As shown in Fig. 3, gradually increasing the PGD attack steps N_{step} raises the attack strength [30], thus leading to perturbed-data accuracy degradation for both vanilla adversary training and our PNI technique. However, for both cases, the perturbed-data accuracy starts saturating and do not degrade any further when $N_{step} = 40$. If our PNI's success comes from the stochastic gradient which gives incorrect gradient owing to the single sample, increasing the attack steps suppose to eventually break the PNI defense which is not observed here. Our PNI method still outperforms vanilla adversarial training even when N_{step} is increased up to 100. Therefore, we can draw the conclusion that, even if PNI does include gradient obfuscation, the stochastic gradient is not the dominant role in PNI for the robustness improvement.

6. Conclusion

In this paper, we present a parametric noise injection technique where the noise intensity can be trained through solving the min-max optimization problem during adversarial training. Through extensive experiments, the proposed PNI method can outperform the state-of-the-art defense method in terms of both clean-data accuracy and perturbed-data accuracy.

Acknowledgement. This work is supported in part by Cyber Florida Collaborative Seed Award Program.

References

[1] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. 1

[2] Anonymous. L2-nonexpansive neural networks. In *Submitted to International Conference on Learning Representations*, 2019. under review. 6, 7

[3] A. Athalye and N. Carlini. On the robustness of the CVPR 2018 white-box adversarial example defenses. *CoRR*, abs/1804.03286, 2018. 7

[4] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. 1, 3, 5, 6, 7, 8

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1

[6] A. Bietti, G. Mialon, and J. Mairal. On regularization and robustness of deep neural networks. *arXiv preprint arXiv:1810.00363*, 2018. 1, 3

[7] 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. 1, 2, 3, 5, 7

[8] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944*, 2018. 1

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2722–2730. IEEE, 2015. 1

[10] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh. Show-and-fool: Crafting adversarial examples for neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017. 1

[11] 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*, pages 15–26. ACM, 2017. 1, 2, 5, 7

[12] M. Cheng, J. Yi, H. Zhang, P.-Y. Chen, and C.-J. Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*, 2018. 1

[13] geoffrey hinton. Neural networks for machine learning: Overview of ways to improve generalization. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec9.pdf, 2014. 1

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[16] 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. 4, 5, 6

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 1

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. 1

[19] C. R. I. G. Jacob Buckman, Aurko Roy. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018. accepted as poster. 3, 4, 6, 7

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[21] J. Kos and D. Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017. 1

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 4

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

[24] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[26] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. *ArXiv e-prints*, 2018. 1, 3, 5, 7

[27] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh. Towards robust neural networks via random self-ensemble. *arXiv preprint arXiv:1712.00673*, 2017. 1, 3, 5, 7

[28] X. Liu, Y. Li, C. Wu, and C.-J. Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *International Conference on Learning Representations (ICLR)*, 2019. 3, 5, 7, 8

[29] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016. 1, 2, 5, 7

[30] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 2, 3, 4, 5, 6, 7, 8

[31] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer. Universal adversarial perturbations against semantic image segmentation. *stat*, 1050:19, 2017. 1

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016. 1

[33] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 1

[34] 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. 2, 5

[35] 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. 1

[36] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. 3

[37] G. K. Santhanam and P. Grnarova. Defending against adversarial attacks by leveraging an entire gan. *CoRR*, abs/1805.10652, 2018. 3

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 1, 3, 6

[39] M. Sun, F. Tang, J. Yi, F. Wang, and J. Zhou. Identify susceptible locations in medical records via adversarial attacks on deep predictive models. *arXiv preprint arXiv:1802.04822*, 2018. 1

[40] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. 1, 2, 3

[41] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013. 3, 6

[42] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018. 3

[43] X. Xu, X. Chen, C. Liu, A. Rohrbach, T. Darell, and D. Song. Can you fool ai with adversarial examples on a visual turing test? *arXiv preprint arXiv:1709.08693*, 2017. 1