Learn2Perturb: an End-to-end Feature Perturbation Learning to Improve Adversarial Robustness

Ahmadreza Jeddi[1], Mohammad Javad Shafiee[1]
Michelle Karg[2], Christian Scharfenberger[2], Alexander Wong[1]
[1]Waterloo AI Institute, University of Waterloo, Waterloo, Ontario, Canada
[2]ADC Automotive Distance Control Systems GmbH, Continental, Germany
[1]{a2jeddi, mjshafiee, a28wong}@uwaterloo.ca
[2]{michelle.karg, christian.scharfenberger}@continental-corporation.com

Abstract

While deep neural networks have been achieving state-of-the-art performance across a wide variety of applications, their vulnerability to adversarial attacks limits their widespread deployment for safety-critical applications. Alongside other adversarial defense approaches being investigated, there has been a very recent interest in improving adversarial robustness in deep neural networks through the introduction of perturbations during the training process. However, such methods leverage fixed, pre-defined perturbations and require significant hyper-parameter tuning that makes them very difficult to leverage in a general fashion. In this study, we introduce Learn2Perturb, an end-to-end feature perturbation learning approach for improving the adversarial robustness of deep neural networks. More specifically, we introduce novel perturbation-injection modules that are incorporated at each layer to perturb the feature space and increase uncertainty in the network. This feature perturbation is performed at both the training and the inference stages. Furthermore, inspired by the Expectation-Maximization, an alternating back-propagation training algorithm is introduced to train the network and noise parameters consecutively. Experimental results on CIFAR-10 and CIFAR-100 datasets show that the proposed Learn2Perturb method can result in deep neural networks which are 4-7% more robust on l_∞ FGSM and PDG adversarial attacks and significantly outperforms the state-of-the-art against l_2 C&W attack and a wide range of well-known black-box attacks[1].

1. Introduction

The vulnerability of DNN models to adversarial examples have raised major concerns [1, 7, 9, 25, 31] on their large-scale adaption in a wide variety of applications.

Adversarial attacks can be divided into two categories of black-box and white-box attacks based on the level of information available to the attacker. Black-box attacks usually perform queries on the model, and they have partial information regarding the data and the structure of the targeted model [16, 28]. On the other hand, white-box attacks have a better understanding of the model that they attack to; therefore, they are more powerful than black-box attacks [14, 34]. This understanding might vary between different white-box attack algorithms; nonetheless, gradients of the model's loss function with respect to the input data is the most common information utilized to modify input samples and generate adversarial examples. First-order white-box adversaries are the most common attacking algorithms which only use the first order of gradients [6, 24, 25, 34] to craft the adversarial perturbation.

In the realm of defense mechanisms, approaches like distillation [26, 29], feature denoising [35], and adversarial training [11, 24] have been proposed to resolve the vulnerability of DNNs on adversarial attacks. Adversarial training is considered as a very intuitive yet very promising solution to improve the robustness of DNN models against adversarial attacks.

Madry *et al.* [24] illustrated that adversarial learning using Projected Gradient Descent (PGD) for generating on-the-fly adversarial samples during training can lead to trained models which provide robustness guarantees against all first-order adversaries. They experimentally showed that the adversarial examples in a l_∞ ball distance around the original sample with many random starts in the ball generated with PGD, all have approximately the same loss value when are fed to the network as input. Due to this fact, they provide the guarantee that as long as the attack algorithm is a first-order adversary, the local maximas of the loss value would not be significantly better than those found by PGD.

[1]The source code is available at https://github.com/Ahmadreza-Jeddi/Learn2Perturb.

Applying regularization techniques is another approach to train more robust network models [5]. To do so, either new loss functions were proposed with added or embedded regularization terms (i.e., adversarial generalization) [10, 15, 33] or the network was augmented with new modules [14, 20, 21, 22] for regularization purposes making the network more robust at the end.

Randomization approaches and specifically random noise injection [14, 21, 22] has been recently proposed as one of the network augmentation methods to address the adversarial robustness in deep neural networks. A random noise generator as an extra module is embedded in the network architecture adding random noise to the input or the output of layers. Although the noise distribution usually follows a Gaussian distribution for its simplicity, it is possible to use different noise distributions. This noise augmentation technique adds more uncertainty in the network and makes the adversarial attack optimization harder which improves the robustness of the model.

While the noise injection technique has shown promising results, determining the parameters of the distribution and how to add the noise values to the network are still challenging. The majority of the methods proposed in literature [20, 21, 22, 36] manually select the parameters of the distribution. However, He *et al.* [14] recently proposed a new algorithm in which the noise distributions are learned in the training step. Their proposed Parametric Noise Injection (PNI) technique injects trainable noise to the activations or the weights of the CNN model. The problem associated to the proposed PNI technique is that the noise parameters tend to converge to zero as the training progresses, making the noise injection progressively less effective over time. This problem is partially compensated through the utilization of PGD adversarial training as suggested by Madry *et al.* [24], but the decreasing trend of noise parameter magnitudes still remains, and thus, limits the overall effect of the PNI.

In this paper, the Learn2Perturb framework, an end-to-end feature perturbation learning approach is proposed to improve the robustness of DNN models. An alternating back-propagation strategy is introduced where the following two steps are performed in an alternating manner: i) the network parameters are updated in the presence of feature perturbation injection to improve adversarial robustness, and ii) the parameters of the perturbation injection modules are updated to strengthen perturbation capabilities against the improved network. Decoupling these two steps helps both sets of parameters (i.e., network parameters and perturbation injection modules) to be trained to their full functionalities and produces a more robust network. To this end, our contributions can be folded as below:

- A highly efficient and stable end-to-end learning mechanism is introduced to learn the perturbation-injection modules to improve the model robustness against adversarial attacks. The proposed alternating back-propagation method inspired by Expectation-Maximization (EM) concept trains the network and noise parameters in a consecutive way gradually without any significant parameter-tuning effort.

- A new effective regularizer is introduced to help the network learning process which smoothly improves the noise distributions. Combining this regularizer and PGD-adversarial training helps the proposed Learn2Perturb algorithm achieve the state-of-the-art performances.

- Exhaustive experiments are conducted for various white-box and black-box adversarial attacks on CIFAR-10 and CIFAR-100 datasets, and new state-of-the-art performances are reported for these algorithms.

The paper is organized as follows: section 2 provides a discussion of related work in terms of different adversarial attacks; the proposed Learn2Perturb approach is presented in Section 3 and experimental results and discussion are presented in Section 4 followed by a conclusion.

2. Related Work

The gradients of the loss function with respect to the input data are very common information used by adversarial attack algorithms. In this type of approaches, the proposed algorithms try to maximize the loss value of the network by crafting the minimum perturbations into input data.

Fast Gradient Sign Method (FGSM) [34] is the simplest yet a very efficient white-box attack. For a DNN parametrized with W (i.e., where the network is encoded as $f_W(x)$) and loss function $\mathcal{L}$, for any input x, the FGSM attack computes the adversarial example x' as:

$$x' = x + \epsilon \cdot \text{sign}\Big(\nabla_x \mathcal{L}\big(f_W(x), x\big)\Big) \tag{1}$$

where ϵ determines the attack strength and $\text{sign}(\cdot)$ returns the sign tensor for a given tensor. Using this gradient ascent step, FGSM tries to locally maximize the loss function $\mathcal{L}$.

The FGSM approach is extended by projected gradient descent (PGD) [19, 24] where for a number of k iterations, PGD produces $x_{t+1} = \text{bound}_{l_p}(FGSM(x_t), x_0)$, in which x_0 is the original input and $0 \leq t \leq k-1$. Using projection, the $\text{bound}_{l_p}(x', x)$ simply ensures that x' is within a specified l_p range of the original input x.

Madry *et al.* [24] illustrated that different PGD attack restarts, each with a random initialization for input within the l_∞–ball around x, find different local maximas with very similar loss values. Based on this finding, they claimed that PGD is a universal first-order adversary.

C&W attack [6] is another strong first-order attack algorithm which finds perturbation δ added to input x by solving the optimization problem formulated as:

$$min\Big[||\delta||_p + c \cdot f(x+\delta)\Big] \quad s.t. \quad x+\delta \in [0,1]^n \qquad (2)$$

where p shows the norm distance. While p can be any arbitrary number, C&W is most effective when $p = 2$; as such, here, we only consider l_2-norm for C&W evaluations. Moreover, $f(\cdot)$ encodes the objective function driving the perturbed sample to be misclassified (ideally $f(\cdot) \leq 0$), and c is a constant balancing the two terms involved in 2. It is worth noting that, all the white-box attacks explained here (i.e. FGSM, PGD, and C&W) are first-order adversaries.

Black-box attacks can only access a model via queries; sending inputs and receiving corresponding outputs to estimate the inner working of the network. To fool a network, the well-known black-box attacks either use surrogate networks [16, 28] or estimate the gradients [8, 32] via multiple queries to the targeted network.

In the surrogate network approach, a new network mimicking the behavior of the target model [28] is trained. Attackers perform queries on the target model and generate a synthetic dataset with the query inputs and associated outputs. Having this dataset, a surrogate network is trained. Recent works [23, 28] showed that adversarial examples fooling the surrogate model can also fool the target model with a high success rate. A simple variant of the surrogate model attack, Transferability adversarial attacks [27], is when the surrogate model has access to the same training data as the interested network. Adversarial examples fooling the substituted network are usually transferred to (and fool) the target model as well. Since substitute networks may not always be successful [8, 16], black-box gradient estimation attacks only deal with the target model itself. Zeroth order optimization (ZOO) [8] and attacks alternating only a few pixels [32] approaches are examples of this kind of black-box attack, to name a few.

3. Methodology

In this work, we propose a new framework called Learn2Perturb for improving the adversarial robustness of a deep neural network through end-to-end feature perturbation learning. Although it has been illustrated both theoretically and practically [2, 30] that randomization techniques can improve the robustness of deep neural networks[2], there is still not an effective way to select the distribution of the noise in the neural networks. In Learn2Perturb, trainable perturbation-injection modules are integrated into a deep neural network with the goal of injecting customized perturbations into the feature space at different parts of the network to increase the uncertainty of its inner workings within an optimal manner. We formulate the joint problem of learning the model parameters and the perturbation distributions of the perturbation-injection modules in an end-to-end learning framework via an alternating back-propagation approach [12]. As shown in Figure 1, the proposed alternating back-propagation strategy for the joint learning of the network parameters and the perturbation-injection modules is inspired from the EM technique; and it comprises of two key alternating steps: i) **Perturbation-injected network training**: the network parameters are trained by gradient descent while the proposed perturbation-injection modules add layer-wise noise to the feature maps (different locations in the network). Noise injection parameters are fixed during this step. ii) **Perturbation-injection module training**: the parameters of the perturbation-injection modules are updated via gradient descent and based on the regularization term added to the network loss function, while network parameters are fixed.

The effect of using such a training strategy is that in step (i), the model minimizes the loss function of the classification problem when noise is being injected into multiple layers, and the model learns how to classify despite the injected perturbations. And in step (ii), the noise parameters are updated with a combination of network gradients and the regularization term applied to these parameters. The goal of this step is to let the network react to the noise injections via gradient descent and pose a bigger challenge to the network via a smooth increase of noise based on the regularizer. The trained perturbation-injection modules perturb the feature layers of the model in the inference phase as well.

3.1. Perturbation-Injection Distribution

Given the observable variables X, W as the input and the set of weights in the neural network, respectively, the goal is to model the neural network as a probabilistic model such that the output of the model, Y, is drawn from a distribution rather than a deterministic function. A probabilistic output is more robust against adversarial perturbation. As such, Y can be formulated as:

$$Y \sim P(X; W, \theta) \qquad (3)$$

where W and θ show the set of network and noise parameters, respectively, and X is the input fed into the network. The output Y is drawn from a distribution driven from W and the set of independent parameters, θ.

For a given layer l of the neural network, the perturbation-injection modules can be used to achieve the following probability model for the layer's final activations:

$$P_l(X_l; W_l, \theta_l) = f_l(X_l, W_l) + Q(\theta_l) \qquad (4)$$

[2]Theoretical background on the effect of randomization algorithm to improve the robustness of a deep neural network model is discussed in the supplementary material.

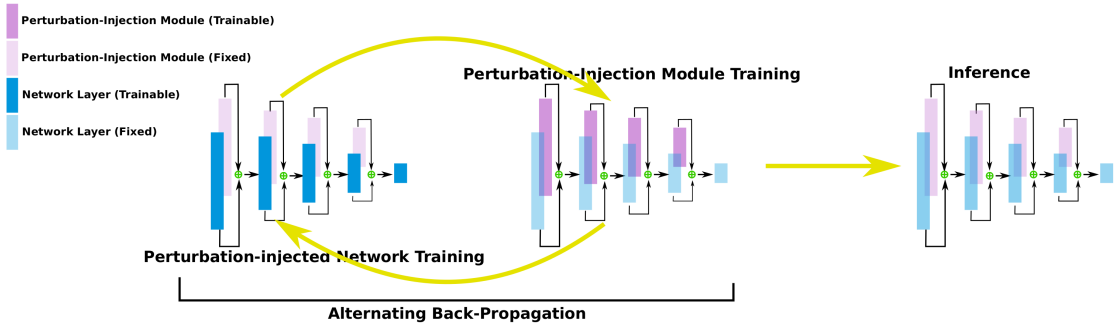

Figure 1. Overview of Learn2Perturb: During training, an alternating back-propagation strategy is introduced where the following two steps are performed in an alternating manner: i) the network parameters are updated in the presence of feature perturbation injection to improve adversarial robustness, and ii) the parameters of the perturbation injection modules are updated to strengthen perturbation capabilities against improved network. The learned perturbation injection modules can be added to some or all tensors in the network to inject perturbations in feature space for two-prong adversarial robustness: i) improve robustness during training when training under perturbation injection, and ii) increase network uncertainty through interference-time perturbation injection to make it difficult to learn an adversarial attack.

where $f_l(X_l, W_l)$ represents the activation of layer l with weights W_l, X_l as its input, and $Q(\theta_l)$ is a noise distribution with parameters θ_l following an exponential distribution function. While $Q(\cdot)$ can be any exponential distribution, we choose Gaussian distribution because of its simplicity and effectiveness, which can be formulated as follow:

$$Q(\theta_l) = \theta_l \cdot \mathcal{N}(0, 1). \tag{5}$$

The parameter θ_l scales the magnitude of the output from the normal distribution encoding the standard deviation of the distribution $Q(\cdot)$. Substituting the right hand-side of $Q(\cdot)$ defined in (5) into (4) enforces $P_l(\cdot)$ to follow a Gaussian distribution:

$$P_l(X_l; W_l, \theta_l) \approx \mathcal{N}\Big(f_l(X_l, W_l), \theta_l\Big). \tag{6}$$

This new probabilistic formulation of layer activations can be extended to the whole network, so instead of a deterministic output Y, network outputs $P(X; W, \theta) \approx \mathcal{N}\Big(f(X, W), \theta\Big)$, with W and θ showing the parameters of all layers.

Having this new formulation for a deep neural network, a proper training process to effectively learn both sets of these parameters is highly desired. To this end, we propose a new training mechanism to learn both network parameters and perturbation-injection modules in an alternating back-propagation approach.

3.2. Alternating Back-Propagation

The proposed neural network structure comprises of two sets of parameters, W and θ, being trained given training samples (X, T) as the input and the ground truth output to the network. However, these two sets of parameters are in conflict with each other and try to push the learning process in two opposite directions. Having the probabilistic representation $P(\cdot)$, W is mapping the input X to output T based on the mean of the distribution $P(\cdot)$, $f(X, W)$; while, the set of θ improves the generalization of the model by including perturbations into the training mechanism.

The proposed alternating back-propagation framework decouples the learning process associated to network parameters W and perturbation-injection distributions θ to effectively update both sets of parameters. To this end, the network parameters and perturbation-injection modules are updated in a consecutive manner.

The training process of the proposed Learn2Perturb is done within two main steps:

- *Perturbation-injected network training*; the parameters of the network, W, are updated via gradient descent to decrease the network loss in the presence of perturbations, caused by the currently fixed perturbation-injection distribution, $Q|_\theta$.

- *Perturbation-injection distribution training*; the parameters of the perturbation-injection distribution, θ, are updated given the set of parameters W are fixed to improve the generalization of the network and as a result, improve its robustness against adversarial perturbation.

These two steps are performed consecutively; however, the number of iterations for each step before moving to the next step can be determined based on the application.

Utilizing a generic loss function in the training of the network when the perturbation-injection modules are embedded forces the noise parameters to converge to zero and eventually removes the effect of the perturbation-injection distributions by making them very small. In other words, the neural network with generic loss tends to learn $P(\cdot)$ as a Dirac distribution where the $Q(\cdot)$ is close to zero. As such, a new regularization term is designed and added to the loss function to prevent the aforementioned problem; the new

Algorithm 1: Alternating back-propagation of the Learn2Perturb framework

Input : Training set D = $\{(x_i, t_i),\ \ i = 1, \ldots, n\}$
Number of training epochs, I
θ_{min}, the lower bound for θ
θ_0, initial values for θ
Learning rate, lr, and constant γ

Output : Learned parameters W
Learned noise distributions $Q(\theta)$

for $t \leftarrow 1$ *to* I **do**

Perturbation-injected training: update W based on the loss function $\mathcal{L}(\cdot)$ Eq. (7) while θ is fixed

$$W^t \leftarrow W^{t-1} - lr \cdot \nabla_W \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big)$$

Perturbation-injection module training: update θ based on Eq. (7) while W is fixed

$$\theta^t \leftarrow \theta^{t-1} - lr \cdot \nabla_\theta \mathcal{L}\Big(P(X; W^{t-1}, \theta^{t-1}), T\Big) - \gamma \cdot \nabla_\theta g(\theta^{t-1})$$

Values of θ^t smaller than θ_{min} are projected to θ_{min}

end

loss function can be formulated as:

$$\arg\min_{W,\theta} \Big[\mathcal{L}\Big(P(X; W, \theta), T\Big) + \gamma \cdot g(\theta)\Big] \tag{7}$$

where $\mathcal{L}(\cdot)$ is the classification loss function (i.e., usually cross entropy) such that the set of parameters W need to be tuned to generate the associated output of the input X. The function $g(\theta)$ is the regularizer enforcing smooth increase in the parameters $\theta = \{\theta_{l,j}\}_{j=1:M_l}^{l=1:K}$, where $\theta_{l,j}$ shows the jth noise parameter in the lth layer, corresponding to an element of the output feature map. K and M_l represent the number of layers and noise parameters per layer, respectively. γ is the hyper-parameter balancing the two terms in the optimization. Independent distributions are learnt for perturbation-injection models in each layer. The regularizer function should be enforced with an annealing characteristic where the perturbation-injection distributions are gradually improved and converged thus the parameters W can be trained effectively. As such the regularization function is formulated as below:

$$g(\theta) = -\frac{\theta^{1/2}}{\tau} \tag{8}$$

where τ is the output of a harmonic series given the current epoch value in the training process. Using a harmonic series to determine τ, gradually decreases the effect of the requlrizer function in the loss and lets the neural network converge. While the squared root of θ makes the equation easier to take the derivative, it also provides a slower rate of change for larger values of θ which helps the network to converge to a steady state smoothly.

As seen in Algorithm 1, first, the perturbation-injection distributions Q and network parameters W are initialized. Then the model parameters W are updated based on the classification loss $\mathcal{L}(\cdot)$, and this loss function is minimized in the presence of perturbation-injection modules. Then, the perturbation-injection distributions Q are updated by performing the "perturbation-injection module training" step.

One of the main advantages of this approach is that since the learning process of these two sets of parameters is decoupled, the training process can be easily performed without a significant manual hyper-parameter tweaking compared to other randomized state-of-the-art approaches. Moreover, the proposed method can help the model to converge faster as the perturbation-injection distributions are continuously improved during the training process.

3.3. Model Setup, Training and Inference

Perturbation-injection distributions are added to the network in different locations and specifically after each convolution operation to create a new network model based on the Learn2Perturb framework. As shown in Figure 1, these modules generate the perturbations with the same size as the feature activation maps of that specific layer. Each perturbation-injection distribution follows independent distribution and therefore, the generated perturbation value for each feature is drawn independently.

In the training phase, the model parameters and the perturbation-injection distributions are trained in an iterative and consecutive manner and based on the proposed alternating back-propagation approach. It is worth to mention that the model parameters are trained for 20 epochs before activating the perturbation distributions to help the network parameters converge to a good initial point. After 20 epochs, the alternating back-propagation is applied to train both model parameters and perturbation-injection distributions. Furthermore, we take advantage of adversarial training technique which adds on-the-fly adversarial examples into the training data, to improve the model's robustness more effectively against perturbations. As such, PGD adversarial technique is incorporated in the training to provide stronger guarantee bounds against all first-order adversaries optimizing in l_∞ space.

The perturbation-injection distributions are applied in the inference step, as well. This will introduce a dynamic nature into the inference process and as a result, it makes it harder for the adversaries to find an optimal adversarial examples to fool the network.

4. Experiments

To illustrate the effectiveness of the proposed Learn2Perturb, we train various models using this framework and evaluate their robustness against different adversarial attack algorithms. Furthermore, the proposed method is compared with different state-of-the-art approaches including PGD adversarial training [24] (also denoted as Vanilla model), Parametric Noise Injection (PNI) [14], Adversarial Bayesian Neural Network (Adv-BNN) [22], Random Self-Ensemble (RSE) [21] and PixelDP (DP) [20].

4.1. Dataset & Adversarial Attacks

For the evaluation purpose, the CIFAR-10 and CIFAR-100 datasets[3] [18] are utilized for training and evaluating the networks. Both of these datasets contain 50,000 training data and 10,000 test data of natural color images of 32×32. While CIFAR-10 has 10 different class with 6000 images per class, CIFAR-100 has 100 classes with 600 images per class.

Different white-box and black-box attacks are utilized to evaluate the proposed Learn2Perturb along with state-of-the-art methods. The competing algorithms are evaluated via white-box attacks including FGSM [34], PGD [19] and C&W attacks [6]. One-Pixel attack [32], and Transferability attack [27] are utilized as the black-box attacks to evaluate the competing method.

4.2. Experimental Setup

We use ResNet based architectures [13] as the baseline for our experiments; The classical ResNet architecture (i.e., ResNet-V1 and its variations) and the new ResNet architecture (i.e., ResNet-V2) are used for evaluation. The main difference between two architectures is the number of stages and the number of blocks in each stage. Moreover, average pooling is utilized for down-sampling in ResNet-V1 architecture while the ResNet-V2 uses 1×1 CNN layers for this purpose. Followed by the experimental setup proposed in [14], data normalization is done via adding a non-trainable layer at the beginning of the network and the adversarial perturbations are directly added to the original input data, before normalization being applied. Both adversarial training and robustness testing setup follow the same configurations as introduced in [24] and [14]. Adversarial training with PGD and testing robustness against PGD, are both done in 7 iterations with the maximum $l_\infty = 8/255$ (i.e., ϵ) and step sizes of 0.01 for each iteration. FGSM attack also uses the same 8/255 limit for perturbation. For C&W attack, we use ADAM [17] optimizer with learning rate $5e^{-4}$. Maximum number of iterations is 1000, and for the constant c in 2 we choose the range $1e^{-3}$ to $1e^{10}$; furthermore to find the value of c, binary search with up to 9 steps is performed. The confidence, κ, parameter of C&W attack, which turns out to have a big effect while evaluating defense approaches involving randomization, takes values ranging from 0 to 5.

In the case of transferability attacks, a PGD adversarially trained network (i.e. a vanilla model) is used as the source network for generating adversarial examples and these adversarial samples are then utilized to attack competing models. For one/few-pixel attacks, we consider the case $\{1, 2, 3\}$-pixel attack in this work. [4]

4.3. Experimental Results

To evaluate the proposed Learn2Perturb framework, the method is compared with PGD adversarial trained model (also denoted as Vanilla). The proposed module is evaluated on three different ResNet architectures. Table 1 shows the effectiveness of the proposed Learn2Perturb method in improving the robustness of different networks architectures. Results demonstrate that the proposed perturbation-injection modules improve the network's robustness. As seen, the proposed perturbation-injection modules can provide robust performance on both 'ResNet-V1' (both with 20 and 56 layers) and 'ResNet-V2' (18 layers) architectures against both FGSM and PGD attacks which illustrates the effectiveness of the proposed module in providing more robust network architectures. Furthermore, the evaluation results for no defense approach (a network without any improvement) are provided as a reference point.

We also evaluate a variation of the proposed Learn2Perturb framework (i.e. Learn2Perturb-R) where we analyze a different approach in performing the two steps of "perturbation-injected network training" and "perturbation-injection module training". In this variation, the perturbation-injection modules are only updated using the regularizer function $g(\theta)$, and network gradients are not used to update θ parameters.

As it can be seen in table 1, taking advantage of both network gradient and the regularizer performs better than when we only take into account the regulizer effect. One reason to justify this outcome is allowing the gradient of loss function $\mathcal{L}(\cdot)$ to update perturbation-injection modules in Learn2Perturb. This would let the loss function to react to perturbations when they cannot tolerate the injected noise and updates the perturbation-injection noise modules more frequently. Nonetheless, the results in table 1 show that Learn2Perturb-R still outperforms other proposed methods in adversarial robustness, though it provides slightly lower accuracy on clean data.

4.4. Robustness Comparison

In this section, to further illustrate the effectiveness of the proposed Learn2Perturb framework, we compare

[3] Experimental results for CIFAR-100 dataset are reported in the supplementary material.

[4] A more detailed experimental setup is provided in the supplementary material.

Table 1. Evaluating the effectiveness of the proposed perturbation-injection modules by comparing against adversarial training algorithm (Vanilla) within the proposed framework and its variation (Learn2Perturb-R).

		No defense			Vanilla[24]			Learn2Perturb-R			Learn2Perturb		
Model	**#Parameter**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**
ResNet-V1(20)	269,722	92.10	0.0±0.0	14.10	83.80	39.10±0.10	46.60	81.15±0.02	50.23±0.14	55.89±0.04	83.62±0.02	51.13±0.08	58.41±0.07
ResNet-V1(56)	853,018	93.30	0.0±0.0	24.20	86.50	40.10±0.10	48.80	82.35±0.03	53.30±0.10	58.71±0.04	84.82±0.04	54.84±0.10	61.53±0.04
ResNet-V2(18)	11.173.962	95.20	0.1±0.0	43.10	85.46	43.90±0.00	52.50	82.46±0.17	53.33±0.12	59.09±0.17	85.30±0.09	56.06±0.16	62.43±0.06

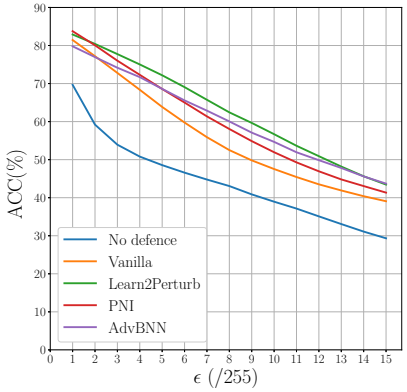

Figure 2. Analyzing the effectiveness of the proposed method compared to state-of-the-art algorithms on different ϵ values for FGSM attack.

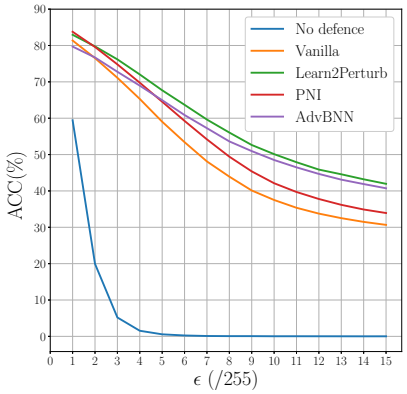

Figure 3. Evaluating the robustness of the proposed Learn2Perturb compared with other state-of-the-art methods through different ϵ based on PGD attack.

Learn2Perturb with PNI [14] and Adv-BNN [22] as two randomization state-of-the-art approaches to improve the robustness of deep neural networks. Table 2 reports these comparison results for different network architectures varying in network depth and capacity. We examine the effect of different network depths including ResNet-V1(20), ResNet-V1(32), ResNet-V1(44) and ResNet-V1(56) along with the effect of network width in Table 2 by increasing the number of filters in ResNet-V1(20) which results to ResNet-V1(20)[1.5×], ResNet-V1(20)[2×] and ResNet-V1(20)[4×]. As seen, while the competing methods do not provide consistent performance by increasing the capacity of the network (increasing depth or width) the proposed framework provides consistent robustness through different network capacities.

The reported results in Table 2 show that while PNI provides minor boosting in network accuracy on clean data, the proposed Learn2Perturb method performs with much higher accuracy when the input data is perturbed with adversarial noise. The main reason for this phenomena is the fact that PNI reach to a very low level of the noise perturbation during the training as the loss function tries to remove the effect of perturbation by making the noise parameters to zero. The results demonstrate that the proposed Learn2Perturb algorithm outperforms the PNI method by 4-7% on both FGSM and PGD adversarial attacks. The proposed method is also compared with Adv-BNN [22]. Results show that while Adv-BNN can provide robust networks in some cases compared to PNI, it is not scalable when the network width is increased and the performance of the networks drop drastically. This is illustrated one of the drawbacks of Bayesian approach which they need to be designed carefully for each network architecture separately.

It has been shown, there is no guarantee that methods robust against l_∞ attacks would provide same level of robustness against l_2 attacks [2]. Araujo *et al.* [2] illustrated experimentally that randomization technique trained with l_∞ can improve the robustness against l_2 attacks as well. In this work we further validate this finding. In order to provide more powerful l_2 attacks challenging the effect of randomization, we apply C&W attacks with different confidence values, κ. The parameter κ enforces the $f(\cdot)$ in 2 to be $\leq -\kappa$ rather than simply ≤ 0. As seen in Table 3, for bigger values of κ the success rate of C&W attack increases; nonetheless, our proposed method outperforms the other competing methods with a big margin for all values of κ.

Table 4 shows the comparison results for the proposed method and state-of-the-art approaches in providing robust network model on CIFAR-10 dataset. The proposed Learn2Perturb method outperforms other state-the-art methods and provides a more robust network model with better performance when dealing with PGD attack.

We also analyze the effectiveness of the proposed method in dealing with different adversarial noise levels. To this end, the ResNet-V2(18) architecture is utilized for all competing methods. The network architectures are designed and trained via four different competing methods; and the trained networks are examined with both FGSM and PGD attacks but with a variation of ϵ values.

Table 2. The effect of network capacity on the performance of the proposed method and other state-of-the-art algorithms. The proposed Learn2Perturb is compared with Parametric Noise Injection (PNI) method [14] and Adv-BNN [22]. Results shows the effectiveness of the proposed Learn2Perturb algorithm in training robust neural network models. To have a fair comparison, we evaluated methods on different network sizes and capacities. Result are reported by standard deviation because of the randomness involved in these methods.

		PNI [14]			Adv-BNN [22]			Learn2Perturb		
Model	**#Parameter**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**	**Clean**	**PGD**	**FGSM**
ResNet-V1(20)	269,722	84.90±0.1	45.90±0.1	54.50±0.4	65.76±5.92	44.95±1.21	51.58±1.49	83.62±0.02	51.13±0.08	58.41±0.07
ResNet-V1(32)	464,154	85.90±0.1	43.50±0.3	51.50±0.1	62.95±5.63	54.62±0.06	50.29±2.70	84.19±0.06	54.62±0.06	59.94±0.11
ResNet-V1(44)	658,586	84.70±0.2	48.50±0.2	55.80±0.1	76.87±0.24	54.62±0.06	58.55±0.49	85.61±0.01	54.62±0.06	61.32±0.13
ResNet-V1(56)	853,018	86.80±0.2	46.30±0.3	53.90±0.1	77.20±0.02	54.62±0.06	57.88±0.02	84.82±0.04	54.62±0.06	61.53±0.04
ResNet-V1(20)[1.5×]	605,026	86.00±0.1	46.70±0.2	54.50±0.2	65.58±0.42	28.07±1.11	36.11±1.29	85.40±0.08	53.32±0.02	61.10±0.06
ResNet-V1(20)[2×]	1,073,962	86.20±0.1	46.10±0.2	54.60±0.2	79.03±0.04	53.46±0.06	58.30±0.14	85.89±0.10	54.29±0.02	61.61±0.05
ResNet-V1(20)[4×]	4,286,026	87.70±0.1	49.10±0.3	57.00±0.2	82.31±0.03	52.61±0.12	59.01±0.04	86.09±0.05	55.75±0.07	61.32±0.02
ResNet-V2(18)	11,173,962	87.21±0.0	49.42±0.01	58.06±0.02	82.15±0.06	53.62±0.06	60.04±0.01	85.30±0.09	56.06±0.08	62.43±0.06

Table 3. Comparison results of the proposed Learn2Perturb and competing methods on C&W [6] attack.

Confidence	**No defense**	**Vanilla**	**PNI**	**Adv-BNN**	**Learn2Perturb**
$\kappa = 0.0$	0.0	0.0	66.9	78.9	**83.6**
$\kappa = 0.1$	0.0	0.0	66.1	78.1	**84.0**
$\kappa = 1.0$	0.0	0.0	34.0	65.1	**76.4**
$\kappa = 2.0$	0.0	0.0	16.0	49.1	**66.5**
$\kappa = 5.0$	0.0	0.0	00.8	16.0	**34.8**

Table 4. Comparison results of the proposed Learn2Perturb and state-of-the-art methods in providing a robust network model. Some of the numbers are extracted from [14]. The reported results are either based on the maximum accuracy achieved in the literature or own results if we achieved higher level of accuracy.

Defense Method	**Model**	**Clean**	**PGD**
RSE [21]	ResNext	87.5	40
DP [20]	28-10 wide ResNet	87	25
Adv-BNN [22]	ResNet-V1(56)	77.20	54.62±0.06
PGD adv. training [24]	ResNet-V1(20) [4×]	87	46.1±0.1
PNI [14]	ResNet-V1(20) [4×]	**87.7±0.1**	49.1±0.3
Learn2Perturb	ResNet-V2(18)	85.3±0.1	**56.3±0.1**

Figure 2 demonstrates the robustness of four competing methods in dealing with FGSM adversarial attack. As seen, while increasing ϵ decreases the robustness of all trained networks, the network designed and trained by the proposed Learn2Perturb approach outperforms other methods through all variations of adversarial noise values (ϵ's).

To confirm the results shown in Figure 2, the same experiment is conducted to examine the robustness of the trained networks on PGD attack. While the PGD attack is more powerful in fooling the networks, results show that the network designed and trained by the proposed Learn2Perturb framework still outperforms other state-of-the-art approaches.

4.5. Expectation over Transformation (EOT)

Athalye *et. al* [3] showed that many of the defense algorithms that take advantage of injecting randomization to network interior layers or applying random transformations on the input before feeding it to the network achieve robustness through false stochastic gradients. They further stated that these methods obfuscate the gradients that attackers utilize to perform iterative attacking optimizations. As such, they proposed the EOT attack (originally introduced in [4]) to evaluate these types of defense mechanisms. They showed that the false gradients cannot protect the network when the attack uses the gradients which are the expectation over a series of transformations.

Since our Learn2Perturb algorithm and other competing methods involve randomization, the tested algorithms in this study are evaluated via the EOT attack method as well. To do so, followed by [30], at every iteration of PGD attack, the gradient is achieved as the expectation calculated from a Monte Carlo method with 80 simulations of different transformations. Results show that the network trained via PNI can provide 48.65% robustness compared to Adv-BNN which provides 51.19% robustness for the CIFAR-10 dataset against this attack. Experimental result illustrates that the proposed Learn2Perturb approach can produce a model which achieves 53.34% robustness and outperforms the other two state-of-the-art algorithms.

It is worth mentioning that the experimental results showed that neither the proposed Learn2Perturb method nor the other competing approaches studied in this work suffer from obfuscated gradients. Furthermore, the proposed Learn2Perturb method successfully passes the five metrics introduced in [3], and thus further illustrates that Learn2Perturb is not subjected to obfuscated gradients.

5. Conclusion

In this paper, we proposed Learn2Perturb, an end-to-end feature perturbation learning approach for improving adversarial robustness of deep neural networks. Learned perturbation injection modules are introduced to increase uncertainty during both training and inference to make it harder to craft successful adversarial attacks. A novel alternating back-propagation approach is also introduced to learn both network parameters and perturbation-injection module parameters in an alternating fashion. Experimental results on both different black-box and white-box attacks demonstrated the efficacy of the proposed Learn2Perturb algorithm, which outperformed the state-of-the-art methods in improving robustness against different adversarial attacks. Future work involves exploring extending the proposed modules to inject a greater perturbation type diversity for greater generalization in terms of adversarial robustness.

References

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

[2] Alexandre Araujo, Rafael Pinot, Benjamin Negrevergne, Laurent Meunier, Yann Chevaleyre, Florian Yger, and Jamal Atif. Robust neural networks using randomized adversarial training. *arXiv preprint arXiv:1903.10219*, 2019. 3, 7

[3] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018. 8

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017. 8

[5] Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pages 664–674, 2019. 2

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017. 1, 3, 6, 8

[7] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018. 1

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui 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. 3

[9] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*, 2018. 1

[10] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In *Advances in neural information processing systems*, pages 842–852, 2018. 2

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[12] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019. 1, 2, 6, 7, 8

[15] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pages 2266–2276, 2017. 2

[16] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018. 1, 3

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 6

[19] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 2, 6

[20] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *arXiv preprint arXiv:1802.03471*, 2018. 2, 6, 8

[21] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018. 2, 6, 8

[22] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. *arXiv preprint arXiv:1810.01279*, 2018. 2, 6, 7, 8

[23] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016. 3

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1, 2, 6, 7, 8

[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal 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

[26] Nicolas Papernot and Patrick McDaniel. On the effectiveness of defensive distillation. *arXiv preprint arXiv:1607.05113*, 2016. 1

[27] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 3, 6

[28] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram 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. 1, 3

[29] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016. 1

[30] Rafael Pinot, Laurent Meunier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler, and Jamal Atif.

Theoretical evidence for adversarial robustness through randomization: the case of the exponential family. *arXiv preprint arXiv:1902.01148*, 2019. 3, 8

[31] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016. 1

[32] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019. 3, 6

[33] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2, 6

[35] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019. 1

[36] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019. 2