Towards Robust Neural Networks via Random Self-ensemble

Xuanqing Liu[1], Minhao Cheng[1], Huan Zhang[2], and Cho-Jui Hsieh[1,3]

[1] Electrical and Computer Science, UC Davis, Davis CA 95616, USA
{xqliu, mhcheng}@ucdavis.edu
[2] Electrical and Computer Engineering, UC Davis, Davis CA 95616, USA
ecezhang@ucdavis.edu
[3] Department of Statistics, UC Davis, Davis CA 95616, USA
chohsieh@ucdavis.edu

Abstract. Recent studies have revealed the vulnerability of deep neural networks: A small adversarial perturbation that is imperceptible to human can easily make a well-trained deep neural network misclassify. This makes it unsafe to apply neural networks in security-critical applications. In this paper, we propose a new defense algorithm called Random Self-Ensemble (RSE) by combining two important concepts: **randomness** and **ensemble**. To protect a targeted model, RSE adds random noise layers to the neural network to prevent the strong gradient-based attacks, and ensembles the prediction over random noises to stabilize the performance. We show that our algorithm is equivalent to ensemble an infinite number of noisy models f_ϵ without any additional memory overhead, and the proposed training procedure based on noisy stochastic gradient descent can ensure the ensemble model has a good predictive capability. Our algorithm significantly outperforms previous defense techniques on real data sets. For instance, on CIFAR-10 with VGG network (which has 92% accuracy without any attack), under the strong C&W attack within a certain distortion tolerance, the accuracy of unprotected model drops to less than 10%, the best previous defense technique has 48% accuracy, while our method still has 86% prediction accuracy under the same level of attack. Finally, our method is simple and easy to integrate into any neural network.

1 Introduction

Deep neural networks have demonstrated their success in many machine learning and computer vision applications, including image classification [14, 7, 35, 9, 34], object recognition [30] and image captioning [38]. Despite having near-perfect prediction performance, recent studies have revealed the vulnerability of deep neural networks to adversarial examples—given a correctly classified image, a carefully designed perturbation to the image can make a well-trained neural network misclassify. Algorithms crafting these adversarial images, called attack algorithms, are designed to minimize the perturbation, thus making adversarial images hard to be distinguished from natural images. This leads to security concerns, especially when applying deep neural networks to security-sensitive systems such as self-driving cars and medical imaging.

To make deep neural networks more robust to adversarial attacks, several defense algorithms have been proposed recently [23, 40, 17, 16, 39]. However, recent studies showed that these defense algorithms can only marginally improve the accuracy under the adversarial attacks [4, 5].

In this paper, we propose a new defense algorithm: Random Self-Ensemble (RSE). More specifically, we introduce the new "noise layer" that fuses input vector with randomly generated noise, and then we insert this layer before each convolution layer of a deep network. In the training phase, the gradient is still computed by back-propagation but it will be perturbed by random noise when passing through the noise layer. In the inference phase, we perform several forward propagations, each time with different prediction scores due to the noise layers, and then ensemble the results. We show that RSE makes the network more resistant to adversarial attacks, by virtue of the proposed training and testing scheme. Meanwhile, it will only slightly affect test accuracy when no attack is performed on natural images. The algorithm is trivial to implement and can be applied to any deep neural networks for the enhancement.

Intuitively, RSE works well because of two important concepts: **ensemble** and **randomness**. It is known that ensemble of several trained models can improve the robustness [29], but will also increase the model size by k folds. In contrast, without any additional memory overhead, RSE can construct infinite number of models f_ϵ, where ϵ is generated randomly, and then ensemble the results to improve robustness. But how to guarantee that the ensemble of these models can achieve good accuracy? After all, if we train the original model without noise, yet only add noise layers at the inference stage, the algorithm is going to perform poorly. This suggests that adding random noise to an pretrained network will only degrade the performance. Instead, we show that if the noise layers are taken into account in the training phase, then the training procedure can be considered as minimizing the upper bound of the loss of model ensemble, and thus our algorithm can achieve good prediction accuracy.

The contributions of our paper can be summarized below:

- We propose the Random Self-Ensemble (RSE) approach for improving the robustness of deep neural networks. The main idea is to add a "noise layer" before each convolution layer in both training and prediction phases. The algorithm is equivalent to ensemble an infinite number of random models to defense against the attackers.
- We explain why RSE can significantly improve the robustness toward adversarial attacks and show that adding noise layers is equivalent to training the original network with an extra regularization of Lipschitz constant.
- RSE significantly outperforms existing defense algorithms in all our experiments. For example, on CIFAR-10 data and VGG network (which has 92% accuracy without any attack), under C&W attack the accuracy of unprotected model drops to less than 10%; the best previous defense technique has 48% accuracy; while RSE still has 86.1% prediction accuracy under the

same strength of attacks. Moreover, RSE is easy to implement and can be combined with any neural network.

2 Related Work

Security of deep neural networks has been studied recently. Let us denote the neural network as $f(w, x)$ where w is the model parameters and x is the input image. Given a correctly classified image x_0 ($f(w, x_0) = y_0$), an attacking algorithm seeks to find a slightly perturbed image x' such that: (1) the neural network will misclassify this perturbed image; and (2) the distortion $\|x' - x_0\|$ is small so that the perturbation is hard to be noticed by human. A defense algorithm is designed to improve the robustness of neural networks against attackers, usually by slightly changing the loss function or training procedure. In the following, we summarize some recent works along this line.

2.1 White-box attack

In the white-box setting, attackers have all information about the targeted neural network, including network structure and network weights (denoted by w). Using this information, attackers can compute gradient with respect to input data $\nabla_x f(w, x)$ by back-propagation. Note that gradient is very informative for attackers since it characterizes the sensitivity of the prediction with respect to the input image.

To craft an adversarial example, [11] proposed a fast gradient sign method (FGSM), where the adversarial example is constructed by

$$x' = x_0 - \epsilon \cdot \mathrm{sign}(\nabla_x f(w, x_0)) \tag{1}$$

with a small $\epsilon > 0$. Based on that, several followup works were made to improve the efficiency and availability, such as Rand-FGSM [32] and I-FGSM [17]. Recently, Carlini & Wagner [5] showed that constructing an adversarial example can be formulated as solving the following optimization problem:

$$x' = \min_{x \in [0,1]^d} c \cdot g(x) + \|x - x_0\|_2^2, \tag{2}$$

where the first term is the loss function that characterizes the success of the attack and the second term is to enforce a small distortion. The parameter $c > 0$ is used to balance these two requirements. Several variants were proposed recently [6, 20], but most of them can be categorized in the similar framework. The C&W attack has been recognized as a strong attacking algorithm to test defense methods.

For untargeted attack, where the goal is to find an adversarial example that is close to the original example but yields different class prediction, the loss function in (2) can be defined as

$$g(x) = \max\{\max_{i \neq t}(Z(x')_i) - Z(x')_t, -\kappa\}, \tag{3}$$

where t is the correct label, $Z(x)$ is the network's output before softmax layer (logits).

For targeted attack, the loss function can be designed to force the classifier to return the target label. For attackers, targeted attack is strictly harder than untargeted attack (since once the targeted attack succeeds, the same adversarial image can be used to perform untargeted attack without any modification). On the contrary, for defenders, untargeted attacks are strictly harder to defense than targeted attack. Therefore, we focus on defending the untargeted attacks in our experiments.

2.2 Defense Algorithms

Because of the vulnerability of adversarial examples [31], several methods have been proposed to improve the network's robustness against adversarial examples. [24] proposed *defensive distillation*, which uses a modified softmax layer controlled by temperature to train the "teacher" network, and then use the prediction probability (soft-labels) of teacher network to train the student network (it has the same structure as the teacher network). However, as stated in [5], this method does not work properly when dealing with the C&W attack. Moreover, [40] showed that by using a modified ReLU activation layer (called BReLU) and adding noise into origin images to augment the training dataset, the learned model will gain some stability to adversarial images. Another popular defense approach is *adversarial training* [17, 16]. It generates and appends adversarial examples found by an attack algorithm to the training set, which helps the network to learn how to distinguish adversarial examples. Through combining adversarial training with enlarged model capacity, [20] is able to create an MNIST model that is robust to the first order attacks, but this approach does not work very well on more difficult datasets such as CIFAR-10.

It is worth mentioning that there are many defense algorithms (*r.f.* [3, 19, 13, 8, 36, 27, 25]) against white box attacks in literature. Unfortunately, as [2, 1] pointed out, these algorithms are not truly effective to white box attacks. Recall the "white box" means that the attackers know *everything* concerning how models make decisions, these include the potential defense mechanisms. In this condition, the white box attacks can walk around all defense algorithms listed above and the accuracy under attack can still be nearly zero. In addition to changing the network structure, there are other methods [39, 21, 10, 12] "detecting" the adversarial examples, which are beyond the scope of our paper.

There is another highly correlated work (*r.f.* [18]) which also adopts very similar idea, except that they view this problem from the angle of differential privacy, while we believe that the adversarial robustness is more correlated with regularization and ensemble learning. Furthermore, our work is public available earlier than this similar work on Arxiv.

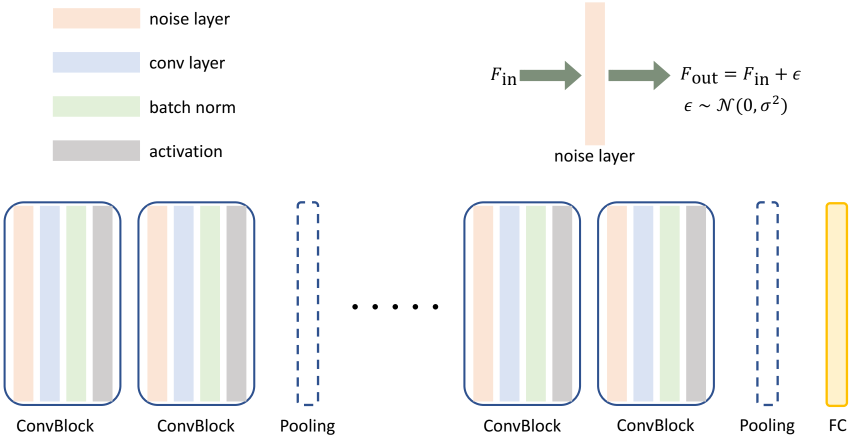

Fig. 1. Our proposed noisy VGG style network, we add a noise layer before each convolution layer. For simplicity, we call the noise layer before the first convolution layer the "init-noise", and all other noise layer "inner-noise". For these two kinds of layers we adopt different variances of Gaussian noise. Note that similar design can be transplanted to other architectures such as ResNet.

3 Proposed Algorithm: Random Self-Ensemble

In this section, we propose our self-ensemble algorithm to improve the robustness of neural networks. We will first motivate and introduce our algorithm and then discuss several theoretical reasons behind it.

It is known that ensemble of several different models can improve the robustness. However, an ensemble of finite k models is not very practical because it will increase the model size by k folds. For example, AlexNet model on ImageNet requires 240MB storage, and storing 100 of them will require 24GB memory. Moreover, it is hard to find many heterogeneous models with similar accuracy. To improve the robustness of practical systems, we propose the following self-ensemble algorithm that can generate an infinite number of models on-the-fly without any additional memory cost.

Our main idea is to add randomness into the network structure. More specifically, we introduce a new "noise layer" that fuses input vector with a randomly generated noise, i.e. $x \to x + \epsilon$ when passing through the noise layer. Then we add this layer before each convolution layer as shown in Fig. 1. Since most attacks require computing or estimating gradient, the noise level in our model will control the success rate of those attacking algorithms. In fact, we can integrate this layer into any other neural network.

If we denote the original neural network as $f(w, x)$ where $w \in \mathbb{R}^{d_w}$ is the weights and $x \in \mathbb{R}^{d_x}$ is the input image, then considering the random noise layer, the network can be denoted as $f_\epsilon(w, x)$ with random $\epsilon \in \mathbb{R}^{d_e}$. Therefore we have an infinite number of models in the pocket (with different ϵ) without

Algorithm 1 Training and Testing of Random Self-Ensemble (RSE)

```
Training phase:
for iter = $1, 2, \ldots$ do
    Randomly sample $(x_i, y_i)$ in dataset
    Randomly generate $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for each noise layer.
    Compute $\Delta w = \nabla_w \ell(f_\epsilon(w, x_i), y_i)$ (Noisy gradient)
    Update weights: $w \leftarrow w - \Delta w$.
end for
Testing phase:
Given testing image $x$, initialize $p = (0, 0, \ldots, 0)$
for $j = 1, 2, \ldots,$ #Ensemble do
    Randomly generate $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for each noise layer.
    Forward propagation to calculate probability output
        $p^j = f_\epsilon(w, x)$
    Update $p$: $p \leftarrow p + p^j$.
end for
Predict the class with maximum score $\hat{y} = \arg\max_k p_k$
```

having any memory overhead. However, adding randomness will also affect the prediction accuracy of the model. How can we make sure that the ensemble of these random models have enough accuracy?

A critical observation is that we need to add this random layer in both training and testing phases. The training and testing algorithms are listed in Algorithm 1. In the training phase, gradient is computed as $\nabla_w f_\epsilon(w, x_i)$ which includes the noise layer, and the noise is generated randomly for each stochastic gradient descent update. In the testing phase, we construct n random noises and ensemble their probability outputs by

$$p = \sum_{j=1}^{n} f_{\epsilon_j}(w, x), \text{ and predict } \hat{y} = \arg\max_k p_k. \tag{4}$$

If we do not care about the prediction time, n can be very large, but in practice we found it saturates at $n \approx 10$ (see Fig. 4).

This approach is different from Gaussian data augmentation in [40]: they only add Gaussian noise to images during the training time, while we add noise before each convolution layer at both training and inference time. When training, the noise helps optimization algorithm to find a stable convolution filter that is robust to perturbed input, while when testing, the roles of noise are two-folded: one is to perturb the gradient to fool gradient-based attacks.The other is it gives different outputs by doing multiple forward operations and a simple ensemble method can improve the testing accuracy.

3.1 Mathematical explanations

Training and testing of RSE. Here we explain our training and testing procedure. In the training phase, our algorithm is solving the following optimization problem:

$$w^* = \arg\min_w \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{train}}} \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} \ell\big(f_\epsilon(w,x_i),y_i\big), \tag{5}$$

where $\ell(\cdot,\cdot)$ is the loss function and $\mathcal{D}_{\text{train}}$ is the training dataset. Note that for simplicity we assume ϵ follows a zero-mean Gaussian, but in general our algorithm can work for a large variety of noise distribution such as Bernoulli-Gaussian: $\epsilon_i = b_i e_i$, where $e_i \overset{\text{iid}}{\sim} \mathcal{N}(0,\sigma^2)$ and $b_i \overset{\text{iid}}{\sim} \mathcal{B}(1,p)$.

At testing time, we ensemble the outputs through several forward propagation, specifically:

$$\hat{y}_i = \arg\max \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} f_\epsilon(w,x_i). \tag{6}$$

Here $\arg\max$ means the index of maximum element in a vector. The reason that our RSE algorithm achieves the similar prediction accuracy with original network is because (5) is minimizing an upper bound of the loss of (6) – Similar to the idea of [22], if we choose negative log-likelihood loss, then $\forall w \in \mathbb{R}^{d_w}$:

$$\begin{aligned}
&\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x_i,y_i)\in\mathcal{D}_{\text{train}}} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} \ell\big(f_\epsilon(w,x_i),y_i\big) \\
&\overset{(a)}{\approx} \mathbb{E}_{(x_i,y_i)\sim\mathcal{P}_{\text{data}}} \Big\{ - \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} \log f_\epsilon(w,x_i)[y_i] \Big\} \\
&\overset{(b)}{\geq} \mathbb{E}_{(x_i,y_i)\sim\mathcal{P}_{\text{data}}} \Big\{ - \log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} f_\epsilon(w,x_i)[y_i] \Big\} \\
&\overset{(c)}{\geq} \mathbb{E}_{(x_i,y_i)\sim\mathcal{P}_{\text{data}}} \Big\{ - \log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} f_\epsilon(w,x_i)[\hat{y}_i] \Big\} \\
&\overset{(a)}{\approx} \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{x_i\in\mathcal{D}_{\text{test}}} - \log \mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2)} f_\epsilon(w,x_i)[\hat{y}_i].
\end{aligned} \tag{7}$$

Where $\mathcal{P}_{\text{data}}$ is the data distribution, $\mathcal{D}_{\text{train/test}}$ is the training set and test set, respectively. And (a) follows from generalization bound (see [28] or appendix for details), (b) comes from Jensen's inequality and (c) is by the inference rule (6). So by minimizing (5) we are actually minimizing the upper bound of inference confidence $-\log f_\epsilon(w,x_i)[\hat{y}_i]$, this validates our ensemble inference procedure.

RSE is equivalent to Lipschitz regularization. Another point of view is that perturbed training is equivalent to Lipschitz regularization, which further helps defensing gradient based attack. If we fix the output label y then the loss function $\ell(f_\epsilon(w,x),y)$ can be simply denoted as $\ell \circ f_\epsilon$. Lipchitz of the function $\ell \circ f_\epsilon$ is a constant $L_{\ell\circ f_\epsilon}$ such that

$$|\ell(f_\epsilon(w,x),y) - \ell(f_\epsilon(w,\tilde{x}),y)| \leq L_{\ell\circ f_\epsilon} \|x - \tilde{x}\| \tag{8}$$

for all $x, \tilde{x}$. In fact, it has been proved recently that Lipschitz constant can be used to measure the robustness of machine learning model [15, 33]. If $L_{\ell \circ f_\epsilon}$ is large enough, even a tiny change of input $x - \tilde{x}$ can significantly change the loss and eventually get an incorrect prediction. On the contrary, by controlling $L_{\ell \circ f}$ to be small, we will have a more robust network.

Next we show that our noisy network indeed controls the Lipschitz constant. Following the notation of (5), we can see that

$$\begin{aligned}\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2)} \ell\big(f_\epsilon(w, x_i), y_i\big) &\overset{(a)}{\approx} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2)} \Big[\ell\big(f_0(w, x_i), y_i\big) + \epsilon^\intercal \nabla_\epsilon \ell\big(f_0(w, x_i), y_i\big) \\ &\qquad + \frac{1}{2}\epsilon^\intercal \nabla^2_\epsilon \ell\big(f_0(w, x_i), y_i\big)\epsilon\Big] \\ &\overset{(b)}{=} \ell\big(f_0(w, x_i), y_i\big) + \frac{\sigma^2}{2} \mathrm{Tr}\Big\{\nabla^2_\epsilon \ell\big(f_0(w, x_i), y_i\big)\Big\}.\end{aligned} \tag{9}$$

For (a), we do Taylor expansion at $\epsilon = 0$. Since we set the variance of noise σ^2 very small, we only keep the second order term. For (b), we notice that the Gaussian vector ϵ is i.i.d. with zero mean. So the linear term of ϵ has zero expectation, and the quadratic term is directly dependent on variance of noise and the trace of Hessian. As a convex relaxation, if we assume $\ell \circ f_0$ is convex, then we have that $d \cdot \|A\|_{\max} \geq \mathrm{Tr}(A) \geq \|A\|_{\max}$ for $A \in \mathbb{S}^{d\times d}_+$, we can rewrite (9) as

$$\mathrm{Loss}(f_\epsilon, \{x_i\}, \{y_i\}) \simeq \mathrm{Loss}(f_0, \{x_i\}, \{y_i\}) + \frac{\sigma^2}{2} L_{\ell \circ f_0}, \tag{10}$$

which means the training of noisy networks is equivalent to training the original model with an extra regularization of Lipschitz constant, and by controlling the variance of noise we can balance the robustness of network with training loss.

3.2 Discussions

Here we show both *randomness* and *ensemble* are important in our algorithm. Indeed, if we remove any component, the performance will significantly drop.

First, as mentioned before, the main idea of our model is to have infinite number of models f_ϵ, each with a different ϵ value, and then ensemble the result. A naive way to achieve this goal is to fix a pre-trained model f_0 and then generate many f_ϵ in the testing phase by adding different small noise to f_0. However, Fig. 2 shows this approach (denoted as Test noise only) will result in much worse performance (20% without any attack). Therefore it is non-trivial to guarantee the model to be good after adding small random noise. In our random self-ensemble algorithm, in addition to adding noise in the testing phase, we also **add noise layer in the training phase**, and this is important for getting good performance.

Second, we found adding noise in the testing phase and then ensemble the predictions is important. In Fig. 2, we compare the performance of RSE with the version that only adds the noise layer in the training phase but not in the testing phase (so the prediction is $f_\epsilon(w, x)$ instead of $\mathbb{E}_\epsilon f_\epsilon(w, x)$). The results clearly show that the performance drop under smaller attacks. This proves **ensemble in the testing phase is crucial**.

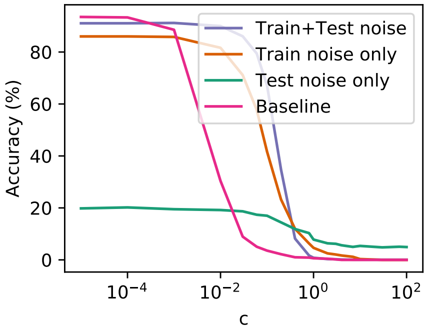

Fig. 2. We test three models on CIFAR10 and VGG16 network: In the first model noise is added both at training and testing time, in the second model noise is added only at training time, in the last model we only add noise at testing time. As a comparison we also plot baseline model which is trained conventionally. For all models that are noisy at testing time, we automatically enable self-ensemble.

4 Experiments

Datasets and network structure We test our method on two datasets—CIFAR10 and STL10. We do not compare the results on MNIST since it is a much easier dataset and existing defense methods such as [23, 40, 17, 16] can effectively increase image distortion under adversarial attacks. On CIFAR10 data, we evaluate the performance on both VGG-16 [26] and ResNeXt [37]; on STL10 data we copy and slightly modify a simple model[4] which we name it as “Model A”.

Defense algorithms. We include the following defense algorithms into comparison (their parameter settings can be found in Tab. 1):

- Random Self-Ensemble (RSE): our proposed method.
- Defensive distillation [24]: first train a teacher network at temperature T, then use the teacher network to train a student network of the same architecture and same temperature. The student network is called the distilled network.
- Robust optimization combined with BReLU activation [40]: first we replace all ReLU activation with BReLU activation. And then at the training phase, we randomly perturb training data by Gaussian noise with $\sigma = 0.05$ as suggested.

[4] Publicly available at `https://github.com/aaron-xichen/pytorch-playground`

– Adversarial retraining by FGSM attacks [17, 16]: we first pre-train a neural network without adversarial retraining. After that, we either select an original data batch or an adversarial data batch with probability 1/2. We continue training it until convergence.

Attack models. We consider the white-box setting and choose the state-of-the-art C&W attack [5] to evaluate the above-mentioned defense methods. Moreover, we test our algorithm under untargeted attack, since untargeted attack is strictly harder to defense than targeted attack. In fact, C&W untargeted attack is the most challenging attack for a defense algorithm.

Moreover, we assume C&W attack knows the randomization procedure of RSE, so the C&W objective function will change accordingly (as proposed in [1] for attacking an ensemble model). The details can be found in the appendix.

Measure. Unlike attacking models that only need to operate on correctly classified images, a competitive defense model not only protects the model when attackers exist, but also keeps a good performance on clean datasets. Based on this thought, we compare the accuracy of guarded models under different strengths of C&W attack, the strength can be measured by L_2-norm of image distortion and further controlled by parameter c in (2). Note that an adversarial image is correctly predicted under C&W attack if and only if the original image is correctly classified and C&W attack cannot find an adversarial example within a certain distortion level.

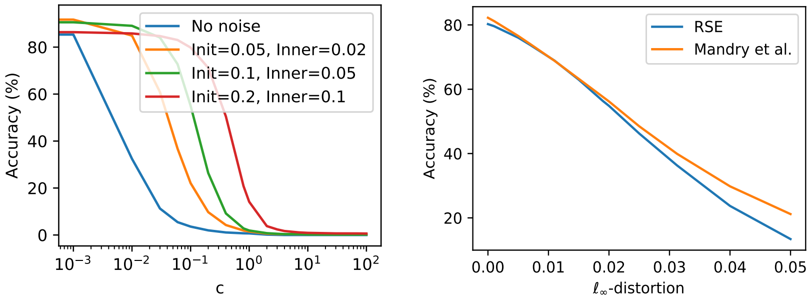

Fig. 3. *Left*: the effect of noise level on robustness and generalization ability. Clearly random noise can improve the robustness of the model. *Right*: comparing RSE with adversarial defense method [20].

4.1 The effect of noise level

We first test the performance of RSE under different noise levels. We use Gaussian noise for all the noise layers in our network and the standard deviation σ of

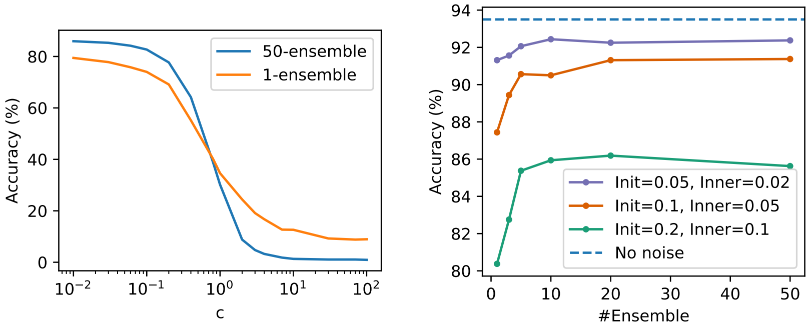

Fig. 4. *Left*: Comparing the accuracy under different levels of attack, here we choose VGG16+CIFAR10 combination. We can see that the ensemble model achieves better accuracy under weak attacks. *Right*: Testing accuracy (without attack) of different n (number of random models used for ensemble).

Table 1. Experiment setting for defense methods

Methods	Settings
No defense	Baseline model
RSE(for CIFAR10 + VGG16)	Initial noise: 0.2, inner noise: 0.1, 50-ensemble
RSE(for CIFAR10 + ResNeXt)	Initial noise: 0.1, inner noise 0.1, 50-ensemble
RSE(for STL10 + Model A)	Initial noise: 0.2, inner noise: 0.1, 50-ensemble
Defensive distill	Temperature = 40
Adversarial training (I)	FGSM adversarial examples, $\epsilon \sim \mathcal{U}(0.1, 0.3)$
Adversarial training (II)	Following [20], PGD adversary with $\epsilon_\infty = \frac{8.0}{256}$
Robust Opt. + BReLU	Following [40]

Gaussian controls the noise level. Note that we call the noise layer before the first convolution layer the "init-noise", and all other noise layers the "inner-noise".

In this experiment, we apply different noise levels in both training and testing phases to see how different variances change the robustness as well as generalization ability of networks. As an example, we choose

$$(\sigma_{\text{init}}, \sigma_{\text{inner}}) = \{(0,0), (0.05, 0.02), (0.1, 0.05), (0.2, 0.1)\} \tag{11}$$

on VGG16+CIFAR10. The result is shown in Fig. 3 (*left*).

As we can see, both "init-noise" and "inner-noise" are beneficial to the robustness of neural network, but at the same time, one can see higher noise reduces the accuracy for weak attacks ($c \lesssim 0.01$). From Fig. 3, we observe that if the input image is normalized to $[0, 1]$, then choosing $\sigma_{\text{init}} = 0.2$ and $\sigma_{\text{inner}} = 0.1$ is good. Thus we fix this parameter for all the experiments.

Table 2. Prediction accuracy of defense methods under C&W attack with different c. We can clearly observe that RSE is the most robust model. Our accuracy level remains at above 75% when other methods are below 30%.

	$c = 0.01$	$c = 0.03$	$c = 0.06$	$c = 0.1$	$c = 0.2$
RSE(ours)	**90.00%**	**86.06%**	**79.44%**	**67.19%**	**34.75%**
Adv retraining	27.00%	9.81%	4.13%	3.69%	1.44%
Robust Opt+BReLU	75.06%	47.93%	30.94%	20.69%	13.50%
Distill	49.88%	17.69%	4.56%	3.13%	1.44%
No defense	30.38%	8.93%	5.06%	3.56%	2.19%

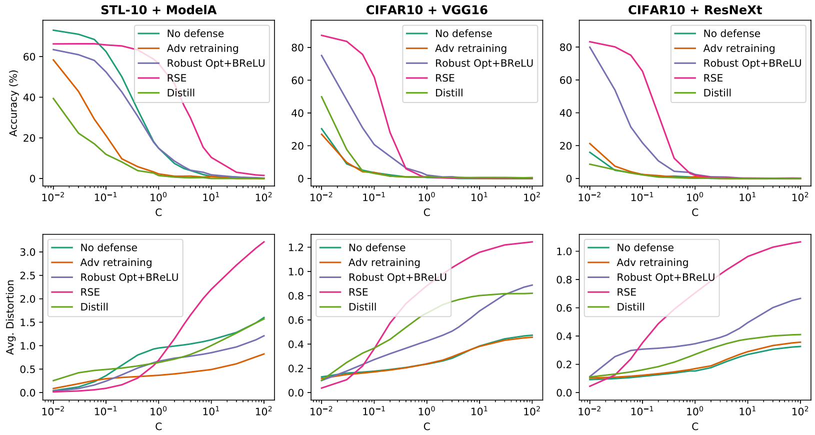

Fig. 5. Comparing the accuracy of CIFAR10+{VGG16, ResNeXt} and STL10+Model A. We show both the change of accuracy and average distortion w.r.t. attacking strength parameter c (the parameter in the C&W attack). Our model (RSE) clearly outperforms all the existing methods under strong attacks in both accuracy and average distortion.

4.2 Self-ensemble

Next we show self-ensemble helps to improve the test accuracy of our noisy mode. As an example, we choose VGG16+CIFAR10 combination and the standard deviation of initial noise layer is $\sigma = 0.2$, other noise layers is $\sigma = 0.1$. We compare 50-ensemble with 1-ensemble (i.e. single model), and the result can be found in Fig. 4.

We find the 50-ensemble method outperform the 1-ensemble method by ∼8% accuracy when $c < 0.4$. This is because when the attack is weak enough, the majority choice of networks has lower variance and higher accuracy. On the other hand, we can see if $c > 1.0$ or equivalently the average distortion greater than 0.93, the ensemble model is worse. We conjecture that this is because when the attack is strong enough then the majority of random sub-models make wrong prediction, but when looking at any individual model, the random effect might

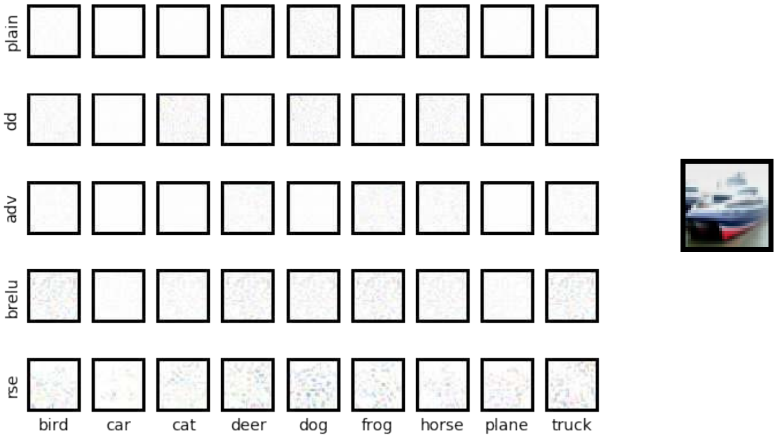

Fig. 6. Targeted adversarial image distortion, each column indicates a defense algorithm and each row is the adversarial target (the original image is in “ship” class, shown in the right side). Here we choose $c = 1$ for targetd C&W attack. Visually, color spot means the distortion of images, thus a successful defending method should lead to more spots.

	bird	car	cat	deer	dog	frog	horse	plane	truck
No defense	1.94	0.31	0.74	4.72	7.99	3.66	9.22	0.75	1.32
Defensive distill	6.55	0.70	**13.78**	2.54	13.90	2.56	**11.36**	0.66	3.54
Adv. retraining	2.58	0.31	0.75	6.08	0.75	9.01	6.06	0.31	4.08
Robust Opt. + BReLU	**17.11**	1.02	4.07	13.50	7.09	15.34	7.15	2.08	17.57
RSE(ours)	12.87	**2.61**	12.47	**21.47**	**31.90**	**19.09**	9.45	**10.21**	**22.15**

Table 3. Image distortion required for targeted attacks.

be superior than group decision. In this situation, self-ensemble may have a negative effect on accuracy.

Practically, if running time is the primary concern, it is not necessary to calculate many ensemble models. In fact, we find the accuracy saturates rapidly with respect to number of models, moreover, if we inject smaller noise then ensemble benefit would be weaker and the accuracy gets saturated earlier. Therefore, we find 10-ensemble is good enough for testing accuracy, see Fig. 4.

4.3 Comparing defense methods

Finally, we compare our RSE method with other existing defense algorithms. Note that we test all of them using C&W untargeted attack, which is the most difficult setting for defenders.

The comparison across different datasets and networks can be found in Tab. 2 and Fig. 5. As we can see, previous defense methods have little effect on C&W attacks. For example, Robust Opt+BReLU [40] is useful for CIFAR10+ResNeXt, but the accuracy is even worse than no defense model for STL10+Model A. In contrast, our RSE method acts as a good defence across all cases. Specifically, RSE method enforces the attacker to find much more distorted adversarial images in order to start a successful attack. As showed in Fig. 5, when we allow an average distortion of 0.21 on CIFAR10+VGG16, C&W attack is able to conduct untargeted attacks with success rate $> 99\%$. On the contrary, by defending the networks via RSE, C&W attack only yields a success rate of $\sim 20\%$. Recently, another version of adversarial training is proposed [20]. Different from "Adversarial training (I)" shown in Tab. 1, it trains the network with adversaries generated by multiple steps of gradient descent (therefore we call it "Adversarial training (II)" in Tab. 1). Compared with our method, the major weakness is that it takes ~ 10 times longer to train a robust network despite that the result is only slightly better than our RSE, see Fig. 3 (*right*).

Apart from the accuracy under C&W attack, we find the distortion of adversarial images also increases significantly, this can be seen in Fig. 2(2nd row), as c is large enough (so that all defense algorithms no longer works) our RSE method achieves the largest distortion.

Although all above experiments are concerning untargeted attack, it does not mean targeted attack is not covered, as we said, targeted attack is harder for attacking methods and easier to defense. As an example, we test all the defense algorithms on CIFAR-10 dataset under targeted attacks. We randomly pick an image from CIFAR10 and plot the perturbation $x_{\mathrm{adv}} - x$ in Fig. 6 (the exact number is in Tab. 3), to make it easier to print out, we subtract RGB channels from 255 (so the majority of pixels are white and distortions can be noticed). One can easily find RSE method makes the adversarial images more distorted.

Lastly, apart from CIFAR-10, we also design an experiment on a much larger data to support the effectiveness of our method even on large data. Due to space limit, the result is postponed to appendix.

5 Conclusion

In this paper, we propose a new defense algorithm called Random Self-Ensemble (RSE) to improve the robustness of deep neural networks against adversarial attacks. We show that our algorithm is equivalent to ensemble a huge amount of noisy models together, and our proposed training process ensures that the ensemble model can generalize well. We further show that the algorithm is equivalent to adding a Lipchitz regularization and thus can improve the robustness of neural networks. Experimental results demonstrate that our method is very robust against strong white-box attacks. Moreover, our method is simple, easy to implement, and can be easily embedded into an existing network.

Acknowledgement. The authors acknowledge the support of NSF via IIS-1719097 and the computing resources provided by Google cloud and Nvidia.

References

1. Athalye, A., Carlini, N.: On the robustness of the cvpr 2018 white-box adversarial example defenses. arXiv preprint arXiv:1804.03286 (2018)
2. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. 35th International Conference on Machine Learning (ICML) (2018)
3. Buckman, J., Roy, A., Raffel, C., Goodfellow, I.: Thermometer encoding: One hot way to resist adversarial examples. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=S18Su--CW
4. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 3–14. AISec '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3128572.3140444, http://doi.acm.org/10.1145/3128572.3140444
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: Security and Privacy (SP), 2017 IEEE Symposium on. pp. 39–57. IEEE (2017)
6. Chen, P.Y., Sharma, Y., Zhang, H., Yi, J., Hsieh, C.J.: Ead: Elastic-net attacks to deep neural networks via adversarial examples. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
7. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q.V., et al.: Large scale distributed deep networks. In: Advances in neural information processing systems. pp. 1223–1231 (2012)
8. Dhillon, G.S., Azizzadenesheli, K., Bernstein, J.D., Kossaifi, J., Khanna, A., Lipton, Z.C., Anandkumar, A.: Stochastic activation pruning for robust adversarial defense. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=H1uR4GZRZ
9. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1625–1634 (2018)
10. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410 (2017)
11. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015), http://arxiv.org/abs/1412.6572
12. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.: On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 (2017)
13. Guo, C., Rana, M., Cisse, M., van der Maaten, L.: Countering adversarial images using input transformations. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=SyJ7ClWCb
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Hein, M., Andriushchenko, M.: Formal guarantees on the robustness of a classifier against adversarial manipulation. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 2263–2273 (2017)
16. Huang, R., Xu, B., Schuurmans, D., Szepesvári, C.: Learning with a strong adversary. arXiv preprint arXiv:1511.03034 (2015)

17. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. In: International Conference on Learning Representations (ICLR) (2017)
18. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., Jana, S.: Certified Robustness to Adversarial Examples with Differential Privacy. ArXiv e-prints (Feb 2018)
19. Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S., Schoenebeck, G., Houle, M.E., Song, D., Bailey, J.: Characterizing adversarial subspaces using local intrinsic dimensionality. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=B1gJ1L2aW
20. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. 6-th International Conference on Learning Representations (ICLR) (2018)
21. Meng, D., Chen, H.: Magnet: A two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 135–147. CCS '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134057, http://doi.acm.org/10.1145/3133956.3134057
22. Noh, H., You, T., Mun, J., Han, B.: Regularizing deep neural networks by noise: Its interpretation and optimization. In: Advances in Neural Information Processing Systems. pp. 5115–5124 (2017)
23. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697 (2016)
24. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Security and Privacy (SP), 2016 IEEE Symposium on. pp. 582–597. IEEE (2016)
25. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=BkJ3ibb0-
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representation (2015)
27. Song, Y., Kim, T., Nowozin, S., Ermon, S., Kushman, N.: Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=rJUYGxbCW
28. Steinhardt, J., Koh, P.W.W., Liang, P.S.: Certified defenses for data poisoning attacks. In: Advances in Neural Information Processing Systems. pp. 3520–3532 (2017)
29. Strauss, T., Hanselmann, M., Junginger, A., Ulmer, H.: Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv:1709.03423 (2017)
30. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
31. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representation (2014)
32. Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P.: Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204 (2017)

33. Weng, T.W., Zhang, H., Chen, P.Y., Yi, J., Su, D., Gao, Y., Hsieh, C.J., Daniel, L.: Evaluating the robustness of neural networks: An extreme value theory approach. 6-th International Conference on Learning Representations (ICLR) (2018)
34. Xiao, C., Li, B., yan Zhu, J., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 3905–3911. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/543, `https://doi.org/10.24963/ijcai.2018/543`
35. Xiao, C., Zhu, J.Y., Li, B., He, W., Liu, M., Song, D.: Spatially transformed adversarial examples. arXiv preprint arXiv:1801.02612 (2018)
36. Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A.: Mitigating adversarial effects through randomization. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=Sk9yuql0Z`
37. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. pp. 5987–5995. IEEE (2017)
38. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning. pp. 2048–2057 (2015)
39. Xu, W., Evans, D., Qi, Y.: Feature squeezing: Detecting adversarial examples in deep neural networks. Network and Distributed System Security Symposium (2018)
40. Zantedeschi, V., Nicolae, M.I., Rawat, A.: Efficient defenses against adversarial attacks. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 39–49. ACM (2017)