SODM: Maximize Certified Robustness In Randomized Smoothing By Solvable Optimization Via Differentiable Mapping

Anonymous CVPR 2021 submission

Paper ID 4750

Abstract

Randomized smoothing has been proved efficient to provide certified defense to norm based adversarial perturbations for large-scale neural networks. However, this certified robustness is achieved by sacrificing significant classification accuracies. We find that due to the non-differentiable property of 0-1 mapping, the optimization for maximizing accuracy and robustness is not solvable by gradient based training. In this paper, we propose SODM: a solvable optimization for robustness via differentiable mapping in randomized smoothing. We derive the gradient from the base classifier's prediction to the smoothed classifier's robustness by any differentiable mapping, thus achieving the direct robustness optimization. Additionally, to reduce the estimation bias and smooth the uneven gradient in this optimization, we further propose a dynamic mapping function and maximum threshold constraint. Different from Gaussian augmentation and adversarial training, our method provides the theoretical proof for maxmizing provable robustness. Experiment shows that our method is much more efficient than exiting ones: on Cifar10, when compared to adversarial training, our method accelerate the training process by 91% to reach the state of the art result. When compared with Gaussian augmentation, our method accelerate the training process by 47% to get the similar performance on ImageNet.

1. Introduction

Deep neural network has made great progress in areas such as computer vision and nature language processing. But it is vulnerable to adversarial examples [8, 20], which, though being small and imperceptible, will seriously mislead the neural network prediction result. In response, many researchers start to find methods to improve the robustness for neural networks, and recently two mainstream defense methods have been proposed: heuristic defense, and certified defense. For heuristic defense, the most efficient method is adversarial training, which enhances the model's robustness by repeatedly training with new generated adversarial examples. However, in this technology, many well-performed models are later broken by unseen or more powerful adversaries [2, 1, 22]. Additionally, the iterative update of the adversary generator is usually time consuming, which results in the long run training process.

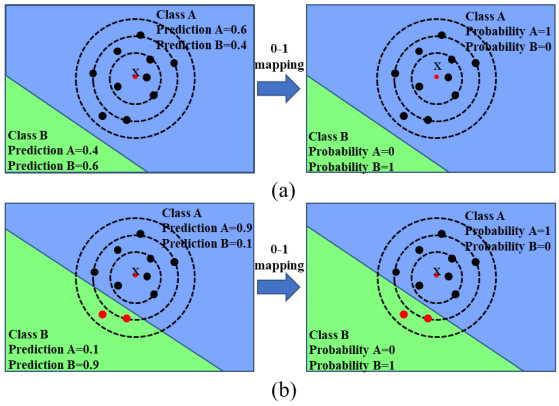


Figure 1. The difference between optimizing prediction and optimizing probability. (a) linear classifier returns class A with average prediction value of 0.6 and average probability value of 1, (b) linear classifier returns class A with average prediction value of 0.74 but average probability value of 0.8. Illustrating that better performance on average prediction cannot guarantee better performance on average probability.

Another line for the robustness is provided by the certified defense methods [17, 25], by which the classifier will give a constant prediction result for any input data x, within certain set around it. Unfortunately, these methods cannot be immediately applied to large scale neural networks that are used in practice. To reduce this limitation, randomized smoothing, developed by [13, 14, 3], has recently been shown effective to provide certified robustness for large-scale neural networks. In randomized smoothing, it turns any classifier $f(x)$ into a new smoothed classifier $g(x)$ which has certifiable l_2 norm robustness guarantee.

Let $f(x)$ be any classifier which maps the input in R^{d} to prediction for classes Y. For any input x, the original classifier $f(x)$ will sample n different examples x' from the distribution of $N(x, \sigma^2 I)$, which is generated by adding the isotropic Gaussian noise $\varepsilon \sim N(0, \sigma^2 I)$ to the input example x and give a prediction result for each sampled input x'. Then, there is a function M mapping the prediction to the probability. The smoothed classifier $g(x)$ returns the class label with highest average probability, and is gauranteed a robustness radius based on the value of highest average probability.

In this technology, the mapping function is a bridge which connects the classification result of $f(x)$ and $g(x)$. In [3, 19], 0-1 mapping as the mapping function, which assigns the class with highest prediction to probability 1 and the others to probability 0. With robustness and accuracy heavily relying on the value of average probability, this mapping function has a significant drawback; the gradient from prediction to probability by this mapping will disappear, meaning that we cannot set robustness even probability as the optimization target in the loss function. So, in those two papers, their optimization is limited in the prediction level. In Figure 1, we give a linear classifier to illustrate the difference between prediction optimization and probability optimization. Though there is a positive correlation between prediction and probability, optimizing the prediction goes in a totally different direction from optimizing probability.

In this paper, we first explain how the accuracy and robustness of $g(x)$ are calculated in randomized smoothing to illustrate why there is a deviation if we only optimize prediction. Then, we prove that, with any differentiable mapping function, there is a solvable optimization expression for robustness and accuracy by gradient descent. To reduce the estimation bias with a lower-cost and smooth the uneven gradient in this optimization, we further propose a dynamic mapping function and maximum threshold constraint. Our method is separately tested in Cifar-10, ImageNet and Mini-ImageNet50, which shows our methods is much more efficient to optimize the robustness than the existing ones. Our contributions can be summarized as follows:

1. We derive the differentiable optimization expression for robustness and accuracy, and propose a dynamic mapping method to modify its gradient and minimize its estimation bias with a relatively low cost.

2. We empirically evaluate our method and compare it with the state-of-the-art methods in Cifar10, ImageNet and Mini-ImageNet50, showing that our method is much more efficient. For example, on Cifar10, to get the state-of-the-art result, our method only consumes around 9% of training time in adversarial training and 34% of training time in MACER.

2. Related work

Recently, many works have proposed different methods to enhance the classifier's robustness for adversarial examples. Generally, those methods can be divided into empirical defenses, which seem robust to known attacks and certified defense, and certified defense which can guarantee a provable robustness to certain kinds of adversarial perturbations.

Empirical defense: Adversarial training [8, 20, 16, 12], is to date the most powerful defense method in empirical defense, which repeatedly generates new adversarial examples during the iteration of classifier and add them to the training set. In this kind of training, the classifier is trained to minimize the worst-case loss over a neighborhood around the input. Though, this defense seems powerful, it cannot guarantee whether the prediction by the empirical robust classifier is truly robust to adversarial examples. In fact, many well performed adversarial training methods are later broken by a stronger or more delicately designed adversaries [2, 1, 22]. To end this arms race between the defenders and attackers, many researchers start to find the defense with formal robustness guarantee.

Certified defense: In certified defenses, any input x is guaranteed with a constant prediction result within a neighborhood of x, often a l_2 or l_∞ ball. Those methods typically are either exact ("complete") or conservative ("sound but incomplete"). Exact methods will report whether there exists or not an adversary near the data point to mislead the classifier, usually by mixed integer linear programming [15, 21] or Satisfiability Modulo Theories [10, 7]. However, those methods take large amount of computational resource thus being hard to transfer to large-scale neural network (more than 100,000 activations) [21]. Conservative methods will also certify there is or not an adversary existing, but they might decline to make a certification when the data x have a safe neighbor. The advantage is that those methods are more flexible and consume less computing resource, thus making them more efficient to build certified defense [25, 23, 24, 18, 26, 6, 4, 10]. However, those methods either require specific network architecture (*e.g.* ReLU activation or layered feedforward structure) or need extensive customization for new architecture. Furthermore, none of those methods are shown to be feasible to provide defense for the modern machine learning tasks such as ImageNet classification.

Randomized smoothing: In randomized smoothing, it does not directly provide certified robustness to the original classifier. Instead, it generates a new smoothed classifier by noise corruption and provide certified robustness for smoothed classifier. Lecuyer *et al.*[13] first prove that by utilizing inequalities from the differential privacy literature, it can provide a strong l_2 and l_1 robustness guarantee for smoothed classifier. Later, Li *et al.* [14], give a stronger

robustness guarantee for l_2 norm adversary based on information theory. In[3], Cohen *et al*. first provides a tight robustness guarantee for l_2 norm adversary and greatly enlarges the average certified robustness radius. Based on this, Salam *et al*. [19] uses adversarial training and Zhai *et al*. [28] uses scalable robustness training to maximize the certified radius and achieve the new state-of-the art result.

3. Existing problem

In this section, we will first perform the analysis about how the certified robustness radius of smoothed classifier is calculated, and then we uncover the problem of 0-1 mapping which conduces to the wrong optimization direction. Correspondingly, we first derive the differentiable expression between the output prediction and the robustness with any differentiable mapping function, by which the robustness can be optimized directly in the training process.

3.1. Robustness and accuracy calculation process

Cohen*et al*. [3] uses Neyman-Pearson lemma to prove a tight certified radius by Gaussian smoothing, and later, Salmam*et al*. [19] comes to the same conclusion by Lipschitz constant. In Gaussian randomized smoothing, it supposed that there is a neural network $f_\theta : x \to U^k$, and a mapping function $M : U^k \to P^k$, where $U^k = \{u_1, ..., u_k\}$ is the output prediction of original classifier $f(x)$ for k classes c, and P^k is the probability for x belonging to c. For the smoothed classifier $g(x)$, the probability P_c for the input x belonging to class c is defined in eq.1, which returns the average of probability P^k for Gaussian input $x + \varepsilon \sim N(x, \sigma^2 I)$, where ε is is the isotropic Gaussian noise with variance of σ and mean of 0. The classifying result of smoothed classifier $g(x)$ is defined in eq.2, which return the class with highest average probability P_c:

$$P_c = \underset{x+\varepsilon\sim N(x,\sigma^2 I)}{E}(M(f_\theta(x+\varepsilon)_c)) \tag{1}$$

$$g(x) = \arg\max_{c\in y} \underset{x+\varepsilon\sim N(x,\sigma^2 I)}{E}(M(f_\theta(x+\varepsilon)_c)) \tag{2}$$

Since the value of $\underset{x+\varepsilon\sim N(x,\sigma^2 I)}{E}(M(f_\theta(x+\varepsilon)_c))$ is not possible to be exactly evaluated, in [3, 19, 28], it is estimated by the Monte Carlo algorithm that succeed with arbitrarily high probability. For each input with label c_A and being correctly classified by $g(x)$, the input is guaranteed with a l_2 certified robustness radius R. For any adversarial perturbation δ with its l_2 norm $\|\delta\|_2 \le R$, the classification result of adversarial example $x + \varepsilon$ will always be c_A. The certified robustness radius R is calculated by eq.3, Where Φ^{-1} is the inverse of standard Gaussian CDF (Cumulative Distribution Function) and $\underline{P_A}$ is the lower bound probability estimation of the most probable class c_A and $\overline{P_B}$ is the higher bound probability estimation of the "runner-up" class c_B.

$$R = \frac{\sigma}{2} * (\Phi^{-1}(\underline{P_A}) - \Phi^{-1}(\overline{P_B})) \tag{3}$$

In randomized smoothing, the mapping function M is the intermediary between the prediction U^k and the probability P_c. However, if we use 0-1 mapping as our mapping function M, the gradient from U^k to P_c will disappear. So, when training the original classifier $f(x)$ to generate a well performed smoothed classifier $g(x)$, the optimization we can achieve is limited in the prediction result U^k but cannot optimize P_c, which finally determines the robustness and the accuracy of $g(x)$.

3.2. Optimization direction and differentiable optimization

While optimizing U^k can also improve the robustness and accuracy, in some sense, it optimizes in the wrong way. Thus they cannot achieve the optimal solution or it takes great effort to achieve the relatively "good solution". We take the optimization objects in two typical papers [3] [19] to explain why optimizing U^k is in the wrong direction. In [3], the optimization is shown in eq.4 and in [19], the optimization is shown in eq.5. We suppose all the labeled data comes from the distribution D and both optimizations take the Cross-Entropy loss (which is selected in their actual training process).

$$\begin{aligned}&\min \underset{\substack{(x,y)\sim D\\ \varepsilon\sim N(0,\sigma^2 I)}}{E}(l(f_\theta, x+\varepsilon, y))\\ &= \min \underset{(x,y)\sim D}{E}[\underset{\varepsilon\sim N(0,\sigma^2 I)}{E}[-\log(M_{soft\max}(f_\theta(x+\varepsilon)_c))]]\end{aligned} \tag{4}$$

$$\begin{aligned}&\min \underset{(x,y)\sim D}{E}[-\log(\underset{\varepsilon\sim N(0,\sigma^2 I)}{E}[M_{soft\max}(f_\theta(x+\varepsilon)_c)])]\\ &\approx \min \underset{(x,y)\sim D}{E}[-\log(\frac{1}{n}\sum_{i=0}^{n}[M_{soft\max}(f_\theta(x+\varepsilon_i)_c)])]\end{aligned} \tag{5}$$

The reason for eq.5 using a "plug-in" estimator to represent the expectation is to calculate the gradient easier. The subtle distinction between those two optimizations of switching the logarithm and the expectation has profound influence on the final result. In eq.4, the optimization attempts to make the classifier perform better on each Gaussian corrupted data point, while the optimization in eq.5 tries to make the classifier perform better on each Gaussian corrupted distribution. From eq.1, the probability P_c, which decides the accuracy and robustness of smoothed classifier is calculated by the average of probability P^k among the Gaussian distribution $N(x, \sigma^2 I)$. Thus, the optimization in eq.5 is much closer to our real goal, but in fact it neither really optimizes robustness, nor accuracy directly. There are two deviations in the optimization direction. First, when generating the

smoothed classifier, both of them take 0-1 mapping to map U^k to P^k, but in their optimization, their use softmax instead. Obviously, with the same U^k, the probability output based on two different mapping function is totally different. Second, if we assign the higher bound estimation of $\overline{P_B}$ as $1-\underline{P_A}$, the calculation of robustness can be written in the eq.6. Assuming the above optimizations take the right mapping function, in this situation, minimizing the value of $-\log(\underline{P_A})$ cannot gaurantee the largest robustness radius, which is defined by $\sigma*(\Phi^{-1}(\underline{P_A}))$.

$$R{=}\sigma*(\Phi^{-1}(\underline{P_A})) \tag{6}$$

So, there are two requirements to make optimization stay in the right direction:

R.1: The mapping function in the optimization process and in the P_c calculation process should be the same.

R.2: The value of the loss function should directly reflect the true value of robustness and accuracy, rather than an approximate value.

To obtain a solvable optimization which meets the above requirements, first we have to create a differentiable function between U^k and P_c, and then make sure the gradient from P_c to robustness and accuracy is computing affordable. Luckily, in [19],it proves that based on Lipschitz-constraint, randomized smoothing also establishs with a soft and differentiable mapping function, where the gradient from U^k to P_c is the derivative of the M. For requirement 2, many previous work propose to decompose the loss of accuracy and robustness[28, 29, 27]. Note that returning a correct classification result is the premise of robustness. Thus, to optimize the robustness, we should first guarantee most points get the correct prediction result, and then, trying to get the largest average robustness radius among those correctly classified samples. So, the optimization should go as eq.7, where β is the parameter to balance the trade-off robustness and accuracy:

$$\max \underset{(x,y\sim D)}{E}(ACC_{\{g_\theta(x)\neq y\}} + \beta * Robustness_{\{g_\theta(x)=y\}}) \tag{7}$$

According to eq.1 and eq.6, we can formulated *ACC* and *Robustness* as shown in eq.8, where $g_\theta(x)$ is the smoothed classifier and, Φ^{-1} is the inverse of standard Gaussian CDF, f_θ is our original classifier:

$$\begin{aligned} ACC_{\{g_\theta(x)\neq y\}} &= \log{(g_\theta(x)_y)}_{\{g_\theta(x)\neq y\}} \\ Robustness_{\{g_\theta(x)=y\}} &= \beta * \Phi^{-1}(g_\theta(x)_y)_{\{g_\theta(x)=y\}} \\ where, g_\theta(x)_y &= \underset{\varepsilon\sim N(0,\sigma^2 I)}{E}[M(f_\theta(x+\varepsilon)_y)] \end{aligned} \tag{8}$$

By choosing a differentiable mapping function M, we can derive the gradient of *ACC* and *Robustness*, which is shown in eq.9 and 10.

$$\nabla ACC_{\{g_\theta(x)\neq y\}} = \frac{\nabla g_\theta(x)}{g_\theta(x)} \approx \frac{\sum_{i=1}^{n}[\nabla M(f_\theta(x+\varepsilon_i)_y)]}{\sum_{i=1}^{n}[M(f_\theta(x+\varepsilon_i)_y)]} \tag{9}$$

$$\begin{aligned} &\nabla Robustness = \nabla g_\theta(x) * \sqrt{2\pi} e^{\frac{\Phi^{-1}(g_\theta(x))^2}{2\sigma^2}} \\ &\approx \tfrac{1}{n}\sum_{i=1}^{n}\nabla[M_{0-1}(f_\theta(x+\varepsilon_i)_y)] * \sqrt{2\pi} e^{\frac{\Phi^{-1}(g_\theta(x))^2}{2\sigma^2}} \\ &where, g_\theta(x) \approx \tfrac{1}{n}\sum_{i=1}^{n}[M_{0-1}(f_\theta(x+\varepsilon_i)_y)] \end{aligned} \tag{10}$$

From the above derivation, we prove it is possible to optimize the robustness and accuracy directly in the model training preocess. However, there still exist some subtle problems in this optimization. In the next section, we will discuss the problems of this optimization by analyzing its gradient and estimation process in detail and then propose the corresponding solutions to mitigate it.

4. Methodology

In this section, we first indicate that if we just optimize the eq.7 and eq.8 directly, there will be an explosion in the gradient of the robustness and an unavoidable estimation bias brought by "plug-in" estimation. Then, we explain the methods to solve those problems: by setting a threshold and choosing a proper mapping function. We summarize two problems as follows:

P.1: The growth rate of the gradient is positively related to P_A, thus making it quite uneven. Futhermore, when P_A is closing to 1, the gradient of robustness will be infinite.

P.2: Using a "plug-in" estimator to represent the expectation will bring the unaviodable bias. However, increasing the number of examples n we sampled to minimize this bias will conversely greatly increase our training time.

4.1. Method to modifiy the gradient

In eq.7 and eq.8, the goal of our optimization is to get a large value of $g_\theta(x)$. However, when $g_\theta(x)$ is close to 1, the value of $\Phi^{-1}(\frac{1}{n}\sum_{i=0}^{n}[M_{soft\max}(f_\theta(x+\varepsilon_i)_c)])$ will grow rapidly and close to infinity. This will bring two problems for model training; the first is that the gradient is easy to explore and the second is that the global convergence speed will be slow if the gradient is too uneven. We give the relationship between gradient of robustness and value of P_A in fig.2 (a), showing that the gradient among the domain ofP_A is extremely uneven and easily explores. To solving this problem, we first limit the value of $g_\theta(x)$ with a threshold, such as 0.99 or 0.98, and the value which exceeds this

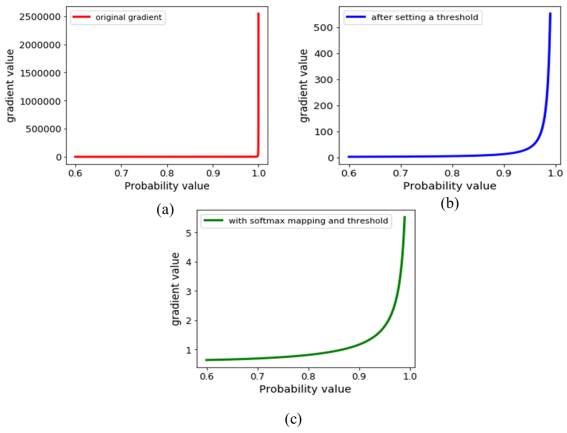


Figure 2. the gradient of the Robustness among the domain of P_A, (a) the gradient without any modification, (b) the gradient with a threshold 0.99, (c) the gradient with softmax mapping and a threshold.

threshold will be compressed or not be counted for gradient. Second, to make this gradient smoother, we use the gradient of $g_\theta(x)$ to moderate its increasing trend. We choose softmax function as our mapping function M for the gradient of this function, shown in eq.11, will be zero when $g_\theta(x)$ closes to 1, thus achieving the moderation.

$$\nabla g_\theta(x) = g_\theta(x) * (1 - g_\theta(x)) \tag{11}$$

With a threshold and softmax mapping, the gradient of Robustness can be written as eq.12, where ρ is the threshold of the largest value of $g_\theta(x)$.

$$\nabla Robustness = g_\theta(x) * (1 - g_\theta(x)) * \sqrt{2\pi} e^{\frac{\Phi^{-1}(g_\theta(x))}{2\sigma^2}}_{g_\theta(x) \le \rho} \tag{12}$$

In fig.2 (b) and (c), we give the modified gradient among the domain of P_A. In (b), we set ρ equals to 0.99 and in (c), we further set the softmax mapping as our mapping function. Obviously, the gradient in fig.2 (c) has a much smoother trend and will not explore.

4.2. Method to reduce the bias

In eq.10, if we use $\frac{1}{n}\sum_{i=0}^{n}[M(f_\theta(x+\varepsilon_i)_c)]$ to represent the value of $\underset{\varepsilon \sim N(0,\sigma^2 I)}{E}[M(f_\theta(x+\varepsilon)_c)]$, there must be an unavoidable bias. In general, enlarging n is the most effective method to dismiss this bias. However, for large scale neural networks, this will great increase the training cost. So, in a different manner, we propose a dynamic and trainable mapping network, which is used to reduce the estimation bias with a relatively low computational cost. We believe that in the training process, the mapping function is like a signpost, guiding the optimization direction of original classifier. Regularly updating the mapping function with a large n' can make the mapping function optimize towards

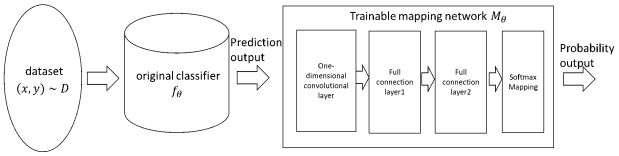


Figure 3. the system flow chart with a dynamically mapping function.

the right direction, thus rectifying the bias caused by a small n during the training process. And because of its smaller scale and slower update rate, it can save a lot of computational costs. In fig.3, we give the flow chart of our dynamic mapping system. In this system, there are two learnable network, original classifier network f_θ and dynamic mapping network M_θ. The classifier network and mapping network have the same optimization object, which combines eq.7 with eq.12. The optimization object is shown as follows:

$$\begin{aligned} &\max \underset{(x,y\sim D)}{E}(ACC_{\{g_\theta(x)\neq y\}} + \beta * Robustness_{\{g_\theta(x)=y, and<\rho\}}) \\ &\approx \min \underset{(x,y\sim D)}{-E}(\log(g_\theta(x)_y)_{\{g_\theta(x)\neq y\}} + \beta * \Phi^{-1}(g_\theta(x)_y)_{\{g_\theta(x)=y, and<\rho\}}) \\ &where, g_\theta(x) \approx \tfrac{1}{n}\sum_{i=1}^{n}[M_\theta(f_\theta(x+\varepsilon_i)_y)] \end{aligned} \tag{13}$$

The difference between those two network is M_θ has a slower update rate and a large n' to estimate the $g_\theta(x)$. The details of robustness training process with dynamic mapping network is describe in Algorithm 1.

Compared to previous work: in [19], Salmam *et al.* tries to use adversarial training to enhance the model's robustness. However, this heuristic method may not take every situation into consideration. Thus it cannot guarantee achieving the optimization situation. Additionally, it takes much time to iteratively update attack module and classification model. Different from adversarial training, our method derives the differentiable expression of robustness and optimize our target directly, thus greatly improve the training efficiency. In [28], Zhai *et al.* first uses a continuous function to approximate 0-1 mapping, thus optimizing robustness directly. However, in his training process, it takes two different mapping functions to measure accuracy and robustness, leading to the deviation in the optimization direction. Secondly, it does not consider the unevenness of the robustness gradient when optimizing, so its training efficiency and result are not as good as ours. In the next section, we will give the comparison experiment result to prove the efficiency and effectiveness of our method.

5. Experiment

We primarily compare our method with Cohen *et al.* [3] and Salman *et al.* [19], which are shown to outperform others. Additionally, we compare our methods with Zhai *et al.* [28], whose training method is similar to us. The experi-

Algorithm 1 Robustness training with dynamic mapping network.

Input: Training set $\hat{p}_{data}$, noise variance σ, number of Gaussian samples n and n', trade-off parameter β, classifier f_θ and mapping network $M_{\theta'}$. update interval m for mapping network;

Output: Optimized parameters θ and θ'

1: **for** each iteration **do**
2: Sample a mini-batch $(x_1, y_1), ..., (x_k, y_k) \sim \hat{p}_{data}$
3: For each input x_i, sample n i.i.d Gaussian examples $x_{i1}, ..., x_{in} \sim N(x, \sigma^2 I)$
4: Calculate the estimation of $g_\theta(x_i) \leftarrow \frac{1}{n} \sum_{j=1}^{n} [M_{\theta'}(f_\theta(x_{ij})_y)]$
5: For each $(x_i, y_i) \sim \hat{p}_{data}$, compute the loss by:

$$l_i = -\log(g_\theta(x_i)_{y_i})_{\{\arg\max(g_\theta(x_i)) \neq y\}} - \beta * \Phi^{-1}(g_\theta(x)_y)_{\{\arg\max(g_\theta(x_i)) = y_i, and g_\theta(x) < \rho\}}$$

6: Update θ with one step of any first-order optimization method to minimize: $\sum_{i=1}^{k} l_i$
7: Initialize b^p as size $n \times 1$ column vector with 0 mean and stardard deviation 0.1
8: **for** $iteration\%m = 0$ **do**
9: Sample a mini-batch $(x_1, y_1), ..., (x_k, y_k) \sim \hat{p}_{data}$
10: For each input x_i, sample n' i.i.d Gaussian examples $x_{i1}, ..., x_{in} \sim N(x, \sigma^2 I)$
11: Repeat the step 5 and 6
12: Update θ' with one step of any first-order optimization method to minimize: $\sum_{i=1}^{k} l_i$
13: **end for**
14: **end for**

ment result demonstrates the efficiency and effectiveness of our proposed method; on Cifar10 we gets much better performance than Cohen's, and, achieve nearly the same result as Salmam's, but using only 9% of training time. For Zhai's, our method significantly outperforms it when the training time is the same. On ImageNet, due to the limitation of computing resource, we only get a similar result to Cohen's, but not as good as Salmam's, which runs the model with adversarial training in a much longer time. To make a further comparison, we randomly choose 50 classes from ImageNet and train our model on this MiniImageNet-50. We compare result with cohen's, showing that our methods can significantly outperform Cohen's.

We use: a ResNet 110 on Cifar10[11] and ResNet 50[9] on ImageNet[5] and Mini-ImageNet50. On cifar10, we run each model for 55 epochs using our proposed algorithm. We employ the same optimizer SGD and set the initialized learning rate to 0.01. We decay the learning rate with 0.1 at 35th/50th epoch. For all models, we set $n = 16$ and $n' = 128$. The trade-off parameters of robustness and accuracy β is set to 6 in the beginning and will increase 10 times at the 35 epoch. The update interval of the mapping network is set to 5. On Mini-Imagenet 50, we set the $n = 4$ and take the Adam optimizer with initialized learning rate 0.01. We run each model for 40 epochs and We decay the learning rate with 0.1 at the 30th/45th epochs. On ImageNet, we run each model for 30 epochs using our proposed algorithm. We use $n = 2$ and take the Adam optimizer with initialized learning rate as 0.01. The learning rate is decayed with 0.1 in 20th/25th epoch. The trade-off parameters of robustness and accuracy β is set to 2.5 in the beginning and will increase 10 times at the 25th epoch. In all datasets, we set the upper threshold ρ as 0.99.

Following the previous work, we report the approximated certified test set accuracy, which is the fraction of the test set that can be certified to be robust at radius r, and the average certified radius, which is the average value of the certified radius among the test set data points. We use the same certified method as Cohen's and set the α=0.001 (similar to pervious work), meaning that, with probability of 0.1%, the smoothed classifier does not return the most probable class or falsely certified a non-robust input.

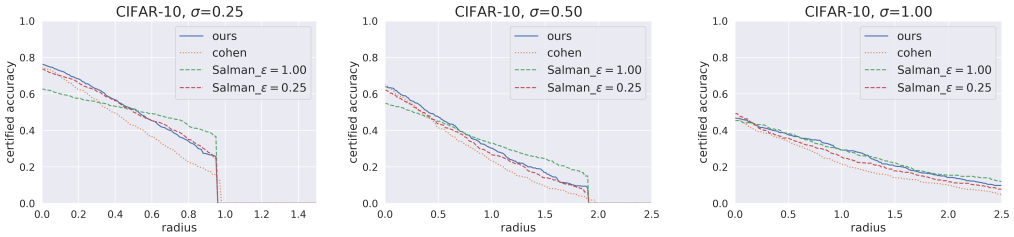
5.1. Result

We report our experiment results on Cifar-10, Mini-ImageNet 50 and Imagenet as follows:

Perfomance: On cifar10, we mainly compared our methods with Cohen's and Salman's. We use the original experiment data provided by Cohen*et al.* and for Salmam's, we take models trained on PGD 10 steps attack with maximum allowed l_2 perturbation being 0.25 and 1.00 for all the models. We give the performance of different models in Table 1 and in Figure 4, we show the certified radius-accuracy curve, under which the area is the ACR. The result shows

Table 1. Approximated certified test accuracy and ACR on Cifar-10: each column is a l_2 radius.

σ	Model	0.00	0.25	0.50	0.75	1.00	1.50	2.00	2.50	ACR
0.25	Cohen	0.75	0.60	0.43	0.27	0.00	0.00	0.00	0.00	0.43
	Salman-PGD10-255	0.62	0.56	0.51	0.46	0.00	0.00	0.00	0.00	**0.49**
	Salman-PGD10-64	0.74	0.63	0.50	**0.39**	0.00	0.00	0.00	0.00	**0.49**
	Ours	**0.76**	**0.66**	**0.52**	0.37	0.00	0.00	0.00	0.00	**0.49**
0.50	Cohen	**0.65**	0.55	0.41	0.32	0.23	0.10	0.00	0.00	0.53
	Salman-PGD10-255	0.55	0.51	0.45	**0.38**	**0.33**	**0.24**	0.00	0.00	**0.67**
	Salman-PGD10-64	0.62	0.54	0.46	0.36	0.26	0.14	0.00	0.00	0.59
	Ours	0.64	**0.56**	**0.48**	**0.38**	0.30	0.17	0.00	0.00	0.64
1.00	Cohen	0.47	0.39	0.34	0.28	0.22	0.14	0.10	0.05	0.54
	Salman-PGD10-255	0.46	**0.43**	0.38	0.33	0.29	**0.22**	**0.15**	**0.12**	**0.75**
	Salman-PGD10-64	**0.49**	0.41	0.36	0.31	0.25	0.18	0.12	0.08	0.63
	Ours	0.47	0.42	**0.38**	**0.35**	**0.29**	0.21	0.14	0.10	0.71

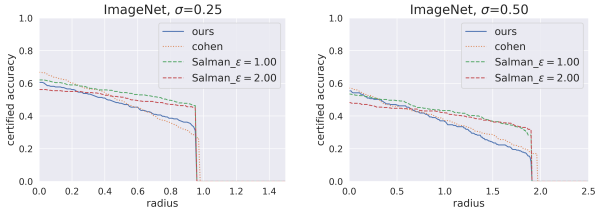


(a) (b) (c)

Figure 4. Certified radius and accuracy curve of different models on Cifar10.

Table 2. Approximated certified test accuracy and ACR on ImageNet: each column is a l_2 radius.

σ	Model	0.00	0.50	1.00	1.50	ACR
0.25	Cohen	**0.67**	0.49	0.00	0.00	0.47
	Salman-PGD1-255	0.62	**0.52**	0.00	0.00	**0.52**
	Salman-PGD1-512	0.56	0.51	0.00	0.00	0.49
	Ours	0.61	0.47	0.00	0.00	0.46
0.50	Cohen	**0.57**	0.46	0.37	0.27	0.73
	Salman-PGD1-255	0.54	0.51	**0.45**	**0.36**	**0.82**
	Salman-PGD1-512	0.48	0.45	0.42	0.37	0.78
	Ours	0.55	0.47	0.37	0.24	0.70



(a) (b)

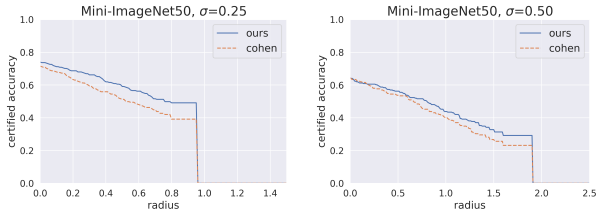
Figure 5. Certified radius and accuracy curve of different models on ImageNet.

that our method get a consistently higher performance than Cohen's and performs better than the model trained with maximum allowed l_2 perturbation limited in 0.25. When compare with the model attack with 1.00 l_2 perturbation, our method get a relatively smaller ACR. However, our model performs better when the image is clean or under a small perturbation.

On ImageNet, we use the same method to get the original experiment data. The difference is that in Salmam's, we choose two well performed models which are trained with PGD 1steps with maximum allowed l_2 perturbation being 1.00 and 2.00. The performance of different models is shown in Table 2 and we show the certified radius-accuracy curve in Figure 5. We only trains our model with 30 epochs, while others have trained for 90 epochs. Our model only gets a similar ACR to cohen's, but being inferior to Salmam's. We guess the reason might be that our model is not fully trained. To further explore the performance of our model, we train our model on MiniImageNet50 with enough epochs. We compare our model with cohen's under σ=0.25and0.50 . The experiment results are given in Tbale 3 and Figure 6, showing that our model outperforms Cohen's in all radius and get a much larger ACR.

Table 3. Approximated certified test accuracy and ACR on MiniImageNet50: each column is a l_2 radius.

σ	Model	0.00	0.25	0.50	0.75	1.00	1.50	ACR
0.25	Cohen	0.72	0.62	0.51	0.43	0.00	0.00	0.51
	Ours	**0.74**	**0.68**	**0.59**	**0.51**	0.00	0.00	**0.57**
0.50	Cohen	**0.64**	0.58	0.54	0.46	0.40	0.27	0.79
	Ours	**0.64**	**0.61**	**0.56**	**0.52**	**0.44**	**0.33**	**0.86**



(a) (b)

Figure 6. Certified radius and accuracy curve of our and cohens' models on mini-ImageNet50.

Table 4. Training time and performance for different models for σ=0.25

Dataset	Model	Sec/epoch	Epochs	total hrs	ACR
Cifar10	Cohen	20.6	90	0.52	0.43
	Salmam-PGD10steps	1305.5	150	54.40	0.49
	MACER-270	183.3	270	13.75	0.48
	Ours	310.3	55	**4.74**	0.49
ImageNet	Coehn	2500.0	90	62.50	0.47
	Salmam-PGD1step	8973.0	90	224.33	0.52
	Ours	3996.1	30	**33.3**	0.46
Mini-ImageNet50	Cohen	128.6	70	2.50	0.51
	Ours	488.2	45	**6.10**	0.57

Training speed: Since our method optimize the model directly by a differentibale optimization function with a smoother gradient, the training speed of our methods are much faster than the others. We first compare our methods with MACER on Cifar10 under σ=0.25, which also achieves direct training by taking a continuous function to estimate the 0-1 mapping. We train two models in MACER: MACER-270, which run 270 epochs with the same policy as its original code, and MACER-90, which run 90 epochs and decay its learning rate with 0.1 at 30th/60th epoch. Our model consumes around the same training as MACER-90 and 34% of training time of MACER-270. The experiment result is shown in Table 5 and Figure 7, showing that our method significantly outperform MACER-90 and get a better performance than MACER-270.

Table 5. Performance of MACER and Ours on Cifar10, σ=0.25

σ	Model	**0.00**	**0.25**	**0.50**	**0.75**	ACR
0.25	MACER-270	0.72	0.61	0.49	**0.37**	0.48
	MACER-90	0.62	0.55	0.45	0.35	0.43
	Ours	**0.76**	**0.66**	**0.52**	**0.37**	**0.49**

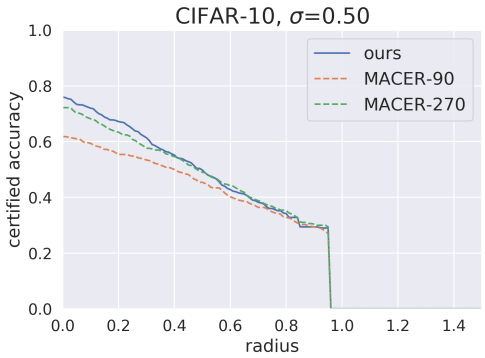


Figure 7. Certified radius and accuracy curve of our model and MACER on Cifar10,σ=0.25.

For all the models mentioned above, we give the estimation running time of them, which is defined by multiplying the time for each epoch and the numbers of epochs. On Cifar10, we use 2 NVIDIA 2080Ti GPUS and in ImageNet and Mini-ImageNet50, we use 4 NVIDIA 2080Ti GPUS to evaluate the training time. The detailed data is shown in Table 4, which demenstrates the effectiveness of our method; On cifar10, we only consume around 9% of training time to reach the same result as adversarial training and around 34% of training time to reach the same result as MACER. In Imagenet, we only use 53% of training time to get the similar result as Cohen's and in Mini-ImageNet, we take 244% time to achieve a much better robustness than Cohen's, which is around 114% less than adversarial training.

6. Conclusion

In this paper, we propose SODM: a solvable optimization for robustness via differentiable mapping in randomized smoothing. Different from previous work, which use Gaussian data augmentation or adversarial training to optimize the robustness in an approximate manner, we optimize the robustness directly by a differentiable mapping function based on gradient descent. Additionally, we further discuss the problems in this optimization: gradient exploration and estimation bias, and propose the corresponding dynamic mapping algorithm to solve it, thus achieving a much more efficient robustness enhancement.

However, there are still some problems in our method: first, our proposed method has not been verified a well performance on ImageNet due to not being fully training. Therefore, it is still questionable whether our method can perform better on large data sets. Second, we do not calculate the excat value of how much estimation bias can be reduced by our dynamic mapping algorithm. In our future work, we will first solve the problem on ImageNet and further explore our dynamic mapping algorithm.

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *International Conference on Machine Learning*, pages 274–283, 2018. 1, 2

[2] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In

Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pages 3–14, 2017. 1, 2

[3] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. pages 1310–1320, 2019. 1, 2, 3, 5

[4] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. Provable robustness of relu networks via maximization of linear regions. In *the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2057–2066, 2019. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 6

[6] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018. 2

[7] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286, 2017. 2

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2

[9] 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

[10] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117, 2017. 2

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

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

[13] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019. 1, 2

[14] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. 2018. 1, 2

[15] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017. 2

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018. 2

[17] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *International Conference on Learning Representations*, 2018. 1

[18] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10877–10887, 2018. 2

[19] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pages 11292–11303, 2019. 2, 3, 4, 5

[20] 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

[21] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *International Conference on Learning Representations*, 2018. 2

[22] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial risk and the dangers of evaluating against weak attacks. *International Conference on Machine Learning*, pages 5025–5034, 2018. 1, 2

[23] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 14, 2018. 2

[24] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems*, pages 6367–6377, 2018. 2

[25] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295, 2018. 1, 2

[26] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *Advances in Neural Information Processing Systems*, pages 8400–8409, 2018. 2

[27] Runtian Zhai, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*, 2019. 4

[28] Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. *International Conference on Learning Representations*, 2019. 3, 4, 5

[29] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. 2019. 4