ON EXPLAINING NEURAL NETWORK ROBUSTNESS WITH ACTIVATION PATH

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite their verified performance, neural networks are prone to be misled by maliciously designed adversarial examples. This work investigates the robustness of neural networks from the activation pattern perspective. We find that despite the complex structure of the deep neural network, most of the neurons provide locally stable contributions to the output, while the minority, which we refer to as float neurons, can greatly affect the prediction. We decompose the computational graph of the neural network into the fixed path and float path and investigate their role in generating adversarial examples. Based on our analysis, we categorize the vulnerable examples into Lipschitz vulnerability and float neuron vulnerability. We show that the boost of robust accuracy from randomized smoothing is the result of correcting the latter. We then propose an SC-RFP (smoothed classifier with repressed float path) to further reduce the instability of the float neurons and show that our result can provide a higher certified radius as well as accuracy.

1 INTRODUCTION

Despite their verified performance, neural networks are prone to be misled by maliciously designed adversarial examples. In response to this issue, many studies focus on defensive algorithms that aim to increase the robustness of deep neural networks. One of the emerging topics in this field is certifiable methods that aim to construct a guaranteed region, within which classifiers are able to provide stable results regardless of the perturbation. The certifiable methods can appear in two different forms: verifiable training and randomized smoothing.

This work introduces an SC-RFP (smoothed classifier with repressed float path) which builds on randomized smoothing algorithms and is able to further improve their robustness accuracy. We decompose the local mapping function into fixed paths and float paths according to the stability of neurons alone on the path. The fixed paths have a stable mapping relationship between input and output, while the float paths can result in a sudden change of the mapping function and alter the result. We categorize the adversarial examples into Lipschitz vulnerable and float neuron vulnerable. With a discussion of the ability of randomized classifiers in correcting misclassified data, we conclude that the essence of the smoothed classifier is to average the contribution of the float path and achieve a locally stable result. Based on this, we further repress the float paths of the network and show that such a classifier can achieve better performance.

The theoretical basis of this work is developed from the analysis of the activation region that was initially proposed for explaining the performance of neural network with a piecewise linear activation function. Intuitively, the input domain of such a neural network N is separated into many regions, within which the mapping of N is piecewise linear. Previous investigation of this field includes the expressivity, sensitivity, and potential issues of the network. However, due to the complexity of neural network, the theoretical investigation only provides insights into the neural network but has yet to be deployed downstream. In this work, we use the theory to explain the model robustness and show introduce a novel way to apply the complex theory to practical. As the theoretical basis is built on piecewise linear functions, in the following of this work, we assume network N has piecewise linear activation without specification.

The contributions of this work are: (1) we introduce a complete framework to describe and decompose the neural network according to the activation status of each neuron; (2) we provide an explanation of adversarial examples and discuss the role of smoothed classifiers as well as their

contribution in correcting misclassified example; (3) we introduce SC-RFP that achieves better performance in certifying the network.

2 Related Works

The adversarial examples are maliciously inputs that are formed by applying an imperceptible perturbation to the original inputs but result in misclassification of a well-trained network (Biggio et al. (2013); Szegedy et al. (2013)). To explain the existence of adversarial example, previous works presented several hypotheses, such as *linearity hypothesis* (Szegedy et al. (2013); Luo et al. (2015)) and *Evolutionary stalling* (Rozsa et al. (2016)). Early works on increasing the robustness of neuron networks focused on adversarial training methods (Goodfellow et al. (2014); Wong et al. (2020); Tramèr et al. (2017); Dong et al. (2018); Kurakin et al. (2016)), while recent investigation shows adversarial training methods can always be broken by more advanced attacks.

Certifiable methods are an emerging solution to the above issue, which aims to provide a certified region, within which the input data are free from attack. It can be categorized as verifiable training and randomized smoothing methods. By viewing the training as a convex optimization problem, dual relaxation approaches apply duality to provide a sound bound for training as well as verify the network (Wong & Kolter (2018); Wong et al. (2018)). An alternative is to estimate the Lipschitz boundary of the network and introduce constraints on either objective loss (Tsuzuku et al. (2018)) or forward propagation (Lee et al. (2020); Weng et al. (2018); Zhang et al. (2019); Huang et al. (2021)). However, verifiable training comes with a compromise of accuracy and scalability.

On the other hand, randomized smoothing introduces a smoothed classifier to the base classifier, therefore has a limited effect on the performance of standard models. Cao & Gong (2017) first propose to ensemble the information around input data to smooth the prediction, but fails to provide a theoretical guarantee on the result. Lecuyer et al. (2019) certify the result of the smoothed classifier with differential privacy. Cohen et al. (2019) provides a theoretical analysis of the certifiable with Monte Carlo, followed by Levine et al. (2019); Li et al. (2019). Based on those works, Jeong & Shin (2020) introduces a regularized to improve the prediction constantly over noise, Jeong et al. (2021) trains the model on a convex combination of samples and Salman et al. (2019) employs PGD attack with randomized smoothing to further increases the robustness accuracy.

Another related topic is the explainability of neuron networks. Lin et al. (2017), Hornik et al. (1989) and Park et al. (2020) investigate how deep models are able to approximate an objective function. An inspiring observation is that the network is a piecewise function when the activation function is piecewise linear Pascanu et al. (2013). The number of linear regions is then adopted as a proxy of network complexity (Montufar et al. (2014); Hanin & Rolnick (2019a;b)). Novak et al. (2018) studies the network sensitivity by countering the transition density of trajectory in the input space. Jiang et al. (2022) compares the similarity of activation patterns globally to study the limitation of deep neural network.

3 Preliminaries

In this section, we introduce the notations adopted in this paper, discuss the robustness training problem and review the definition of activation pattern.

Notation Considering a classification task with c classes drown from data distribution D, where $x \in X$ is the input data and $y \in \{1, 2, \ldots, c\}$ is the ground truth label of x. Let N be a feedforward network with parameter θ and mapping function $f : X \to Y \subseteq R^c$. Assume N is a d layer network, then f can be viewed as composition of functions $f = h_n \circ h_{n-1} \cdots \circ h_1$ where each h_i consists of a linear affine ϕ_i, an optional batch-normalization layer ψ_i, and a non-linear activation layer σ_i, while the last layer h_n omits the activation function.

For data $(x, y) \sim D$, the network computes a probability for each class $f(x) \in R^c$ and predict the label of x by finding the class with the highest probability: $\hat{y} = \arg\max_{m \in Y} f_m(x)$ where $f_m(x)$ is the m-th element of the network output vector. We use $x_i(x; \theta)$, $y_i(x; \theta)$ and $z_i(x; \theta)$ to denote the input, output and pre-activation value of block i for data x. Given R^{n_0} as the size of input space, the output of each layer are deterministic in accordance with the network structure. Let n_i be the

output size of layer i, we use a tuple pair (i, j) to denote the j-th component of layer i and denote $\mathcal{I} = \cup_{i=1}^{d-1}\{(i, j)|j \in \{1, \ldots, n_i\}\}$ as the collection of the indexes of intermediate layers.

3.1 Randomized Certifiable Classifier

The research on *certifiable training* aims to provide a guaranteed region for its input x, within which a classifier always provide stationary result. To be specific, a classifier is regarded as robust for an input x for perturbation of size r if:

$$\arg\max_{m \in Y} f_m(x') = \arg\max_{m} f_m(x), \forall x' \in B_p(x, r) \tag{1}$$

where $B_p(x, r) := \{x' : \|x' - x\|_p \leq r\}$ is the sphere with radius r measured by the metric induced by p-norm. By taking the standard performance of the classifier into account, the robust accuracy of f with radius r is then defined as:

$$R(f) = \mathbb{E}_{(x,y)\sim D}\left[\arg\max_{m \in Y} f_m(x') = y, \forall x' \in B_p(x, r)\right]. \tag{2}$$

However, this robustness of certificated models are achieved with an impairment of expressivity and staleness. As a concession for that, randomized algorithms are proposed to verify the network with a sound theoretical bound at the cost of slight additional computation other than model performance.

Let g be a randomized algorithm constructed based on classifier f. Given a data $(x, y) \sim \mathbb{D}$, the algorithm employs a certain degree of randomness during the induction of f. The output of $g(x)$ is a random vector where each of the element $g_i(x)$ represents the probability of y belongs to class i. For instance, the naive randomized smoothing takes at noised x as input and can be written as:

$$g_i(x) = \mathbb{P}(\arg\max_{m \in \mathcal{Y}} f_m(x + \epsilon) = i), \epsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{3}$$

while the other randomized classifier appears in different forms.

The robustness evaluation of the randomized classifier is slightly different from others in such a way that an expectation of the output random variable is introduced, as the prediction of g appears in probability:

$$\hat{R}(g) = \mathbb{E}_{(x,y)\sim D}\left[\arg\max_{m \in Y} \mathbb{E}\left[g_m(x') = y\right] > y, \forall x \in B_p(x, r)\right] \tag{4}$$

The randomized certifiable algorithm aims to maximize the above equation.

3.2 Activation Pattern and Computational Path

Intuitively, the input space is partitioned into small linear regions by a neural network with piecewise linear activation function. Each of the regions is initially referred to as an activation region. Previous works mostly focused on the general properties of linear regions in investigating the expressivity and limitation of neural network. As our objective is to study the unit-wise reaction towards perturbation, introducing additional notations to describe the neurons is necessary for our analysis.

Definition 1 (Generalized Activation Pattern / Region)**.** *Let N be d block feed forward network with measure zero parameters θ set with respect to Lebesgue measure. For any input $x \in R^{n_0}$, denote the pre-activation of block i as $z_i \in R^{n_i}$. Given an arbitrary activation function $\pi : R^1 \to R^1$, a set of breakpoints $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_q\}$ separate the domain of σ into $q + 1$ intervals $U = \{U_0, U_1, \ldots, U_q\}$, where*

$$U_i = \begin{cases} (-\infty, s_1), & i = 0, \\ (s_i, s_{i+1}), & i = 1, \ldots, q-1, \\ (s_q, +\infty), & i = q. \end{cases}$$

An activation pattern of N is an indexed family: $\mathcal{A} := \{a_{ij} \,|a_{ij} \in \{0, 1, \ldots, q\}, (i, j) \in \mathcal{I}\}$. The activation region given a pattern $\mathcal{A}$ is defined as:

$$\mathcal{R}(\mathcal{A}; \theta, \sigma, \Gamma) := \{x \in R^{in} | z_{ij}(x; \theta) \in U_{a_{ij}}, a_{ij} \in \mathcal{A}\} \tag{5}$$

where $z_{ij}(x;\theta)$ is the j-th element of the pre-activation value in layer i. Conversely, for any $x \in R^{in}$, we denote the pattern of x as:

$$\hat{\mathcal{A}}(x;\theta,\sigma,\Gamma) = \{\hat{a}_{ij}(x), (i,j) \in \mathcal{I}\} \tag{6}$$

where:

$$\hat{a}_{ij}(x;\theta,\sigma,\Gamma) = \begin{cases} i, & z_{ij}(x;\theta) \in U_i, \\ -1, & z_{ij}(x;\theta) \in \{\gamma_1, \gamma_2, \ldots, \gamma_q\} \end{cases} \tag{7}$$

*Moreover, if σ is continues within every interval U_i, the Γ is a **continuous separation** of net N.*

Understanding Activation Region. Given a network N and an input space R^{in}, the generalized activation pattern describes the activation status of each unit at the intermediate layers. Each of the activation pattern defines a unique activation region $\mathbb{R}(\mathcal{A};\theta,\sigma,\Gamma) \subset R^{in}$, such that, for every $x \in \mathbb{R}(\mathcal{A};\theta,\sigma,\Gamma)$, the pre-activation value of each neuron $z_{ij}(x;\theta)$ locates within the interval a_{ij}-th interval of the activation σ domain.

As most of the activation functions have distinguishable activated and deactivated region, the separation on σ is able to provide insights on the neuron level reaction towards the input. When dependencies are fixed, we abbreviate equation 5 and Equation 6 as $\mathcal{R}(A)$ and $\hat{\mathcal{A}}(x)$.

In Equation 7, we assign the pattern of neuron is abstained when it is on the boundary between two regions. This happens only when the data locates on one of the hyperplanes $H_{ijk} := \{x \in U | z_{ij}(x;\theta) = \gamma_k\}$. Assuming the parameter set has measure zero with respect to Lebesgue measure, this happens with probability 0 for $x \in R^{n_0}$.

The essence of operator $\mathbb{R}(\cdot)$ is to find all the x that satisfies a certain constraint determined by the activation pattern. It can be expressed as the union of a sequence of subspaces:

$$\mathcal{R}(\mathcal{A}) = \bigcap_{\forall i,j} \{x \in R^{in} | z_{ij}(x) \in U_{a_{ij}}\} \tag{8}$$

Given a subset of indexes $\mathcal{I}^c \subset \mathcal{I}$, releasing the constraints of post-activations indexed by $\mathcal{I} \backslash \mathcal{I}^c$ then equals to remove related intersection. Formally, we denote A^c as an incomplete activation pattern:

Definition 2. *Let N be a neural network defined as 1 with continues separation set Γ. Given an activation pattern A and a subset of the index set $\mathcal{I}^c \in \mathcal{I}$, we denote $\mathcal{A}_{\mathcal{I}^c} \subset \mathcal{A}$ is an **incomplete activation pattern** of A: $\mathcal{A}_{\mathcal{I}^c} := \{a_{ij} \,|a_{ij} \in \mathcal{A}, (i,j) \in \mathcal{I}^C\}$. The activation region of $\mathcal{A}_{\mathcal{I}^c}$ is denoted as: $\mathcal{R}(\mathcal{A}^c) = \bigcap_{(i,j)\in\mathcal{I}^c} \{x \in R^{in} | z_{ij}(x) \in U_{a_{ij}}\}$.*

In Section 4 we show that any sphere $B_p(x,r)$ can be covered an incomplete activation region $\mathcal{R}(\mathcal{A}^c)$. This suggests that, every neuron $(i,j) \in \mathcal{I}^c$ has same pattern in the region $B_p(x,r)$, therefore are computational stable locally.

4 From Float Path to Model Robustness

In this section, we introduce our definition in describing the local status of neurons and the computational graph. We then provide a theoretical analysis on the relationship between model robustness and the float path, and illustrate the results.

4.1 Decomposing the Computational Graph

Motivation. The investigation of this work is motivated by a simple observation. Given a neural network N with piece-wise linear activation function, the mapping function $f : R^{n_0} \to R^c$ is a piece-wise function, where each piece is linear and can be decomposed as summation of paths. Each of the path is defined as:

Definition 3. *Let N be a neural network defined as Definition 2. Denote $\mathcal{R}(\mathcal{A}_i;\theta,\sigma,\Gamma)$ as the activation region define by an activation pattern $\mathcal{A}$. A path of N is a set of neurons that:*

$$\zeta := \{\zeta_i | i = 0, 1, \ldots, d\} \in [n_0] \times [n_1] \cdots \times [n_d] \tag{9}$$

where $[n_i] = \{1, 2, \ldots, n_i\}$ is neuron index for layer i.

In the above definition, we use ζ to represent a path from input to output. As N is a feedforward network, there is only one unique neuron passed by path ζ during the forward pass which is denoted as ζ_j for layer j. The computational graph of the network then can be decomposed as:

Lemma 1 (Decomposition of Neural Network)**.** *Let N be a neural network defined as Definition 2 and piecewise linear activation function. Given activation pattern $\mathcal{A}$, for any $x \in \mathbb{R}(\mathcal{A}; \theta, \sigma, \Gamma)$. The k-th component of output vector of layer i can be represented as:*

$$y_{ik}(x) = \sum_{\forall \zeta_i = k} x_{j,\zeta_j} \prod_{m=j+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} + \sum_{o=j}^{i-1} \sum_{\forall \zeta_i = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} \quad (10)$$

where x_{j,ζ_j} is the ζ_j-th element of input in layer j, $W'^{(i)}$ and $\beta^{(o)}$ are the equivalent matrix of $\psi_i \circ \phi_i$ and shift parameter of ψ_i, $\mathcal{A}_{m,\zeta_m}$ is the activation pattern of ζ_m-th component of layer m and $d_{\mathcal{A}_{m,\zeta_m}}$ is the slope of pattern $\mathcal{A}_{m,\zeta_m}$: $d_{\mathcal{A}_{m,\zeta_m}} := \sigma'(t), t \in U_{\mathcal{A}_{m,\zeta_m}}$.

The above Lemma decomposes the mapping function from $\mathcal{R}(\mathcal{A}_i; \theta, \sigma, \Gamma)$ to R^c in the manner of combination of path. The value of each path is determined by the input value of that path as well as the activation status $\{d_{\mathcal{A}_{m,\zeta_m}}\}$, while the latter depends on the input activation pattern $\mathcal{A}$. We use $\zeta_{(i,j)}(v; x)$ to denote the value of a path ζ given v as the input of that path:

$$\zeta_{(i,j)}(v, \mathcal{A}) := v \prod_{m=j+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})}. \quad (11)$$

We are interested in the computational relationship between a single region and its neighbor. Geometrically, the input domain are separated by a collection of bent-hyperplanes $\{H_{ijk}(\theta)\}$, where $H_{ijk} := \{x \in U | z_{ij}(x; \theta) = \gamma_k\}$ is determined by the critical point for a particular neuron. In other words, the pattern difference between x and x' is close to the number of bent-hyperplanes crossed by the segment connecting to x and x'. This means the majority of the path value is identical for two close points x and x' and the difference between their prediction are caused by the unstable neurons. Therefore, two close regions should have similar properties in the manner of both neuron status and computation graph. Based on this intuition, we introduce the following definition to specify the *unstable* neurons.

Definition 4 (Float and Fixed Neuron)**.** *Let $N(x)$ be a neural network defined on R^{n_0} with monotonous activation σ, measure zero parameters θ with respect to Lebesgue measure and a continuous separation $\mathcal{S}$. For any subset $R \subset R^{n_0}$, if a neuron z has the same pattern for any $x \in R$, we call it a fixed neuron of R, otherwise it is a float neuron. We denote the collection of fixed neurons and float neurons as $\mathcal{I}^X(R)$ and $\mathcal{I}^T(R)$, respectively:*

$$\begin{aligned} \mathcal{I}^X(R) =& \{(i,j) \mid \exists x_1, x_2 \in R, \hat{a}_{ij}(x_1)_{ij} \neq \hat{a}_{ij}(x_2)_{ij}\} \\ \mathcal{I}^T(R) =& \{(i,j) \mid \forall x_1, x_2 \in R, \hat{a}_{ij}(x_1)_{ij} = \hat{a}_{ij}(x_2)_{ij}\} \end{aligned}$$

where $\hat{a}_{ij}(x_1)$ is defined in Definition 1.

In above definition, both $\mathcal{I}^X(R)$ and $\mathcal{I}^T(R)$ are subset of the index set $\mathcal{I}$. This allows us to decompose the computational graph of the network in a region into two separate set, which we referred as *float* path and *fixed* path:

Definition 5 (Float Path and Fixed Path)**.** *Let N be a neural network defined as Definition 4. Given a subspace $R \in R^{in}$, a path ζ of N is a float path in R, if there exist a $\zeta_i \in \zeta$ is a float neuron, otherwise it is a fixed path. The collection of float path and float path are denoted as:*

$$\begin{aligned} \mathcal{Z}^T(R) &= \{\zeta | \exists \zeta_i \in \zeta, (i, \zeta_i) \in \mathcal{I}^T(R), \} \\ \mathcal{Z}^I(R) &= \{\zeta | \forall \zeta_i \in \zeta, (i, \zeta_i) \in \mathcal{I}^I(R), \} \end{aligned} \quad (12)$$

With all the notations introduced, we are able to show the basic properties of the network in the manner of float neurons and path.

Lemma 2. *Let N be a neural network defined as 4. Given $R \subset X$, denote $\mathcal{I}^X$ and $\mathcal{I}^T$ are the set of float neuron and fixed neuron in R. For any x and $x' \in R$ with activation pattern $\mathcal{A}$ and $\mathcal{A}'$, we have :*

1. $\mathcal{I}^X(R) \bigcup \mathcal{I}^T(R) = \mathcal{I}$
2. $R \subset \mathcal{R}(\mathcal{A}_{\mathcal{I}^X})$
3. $f(x) = \mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) + \mathcal{Z}^T(x, \mathcal{A}; \mathcal{R})$
4. $f(x) - f(x') = J(x)(x - x') + \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}) - \mathcal{Z}^T(x', \mathcal{A}'; \mathcal{R})$

where $\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) = \sum_{\zeta \in \mathcal{Z}^I(\mathcal{R})} \zeta(x, \mathcal{A})$, $\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}) = \sum_{\zeta \in \mathcal{Z}^T(\mathcal{R})} \zeta(x, \mathcal{A})$ *are the sum of fixed path and float path given the region* $\mathcal{R}$, $J(x)$ *is the Jacobian matrix of* f *at* x.

The first three statements in Lemma 2 are elementary, which shows that: (1) Each neuron are either float or fixed within a region; (2) Each region can be covered by the incomplete activation region defined by the fixed neurons; (3) $f(x)$ can be decomposed into fixed part and float path. They lead us to the last statement which contributes the difference between $f(x)$ and $f(x)'$ into two parts. The former is a linear function of the sum of all fixed path in the region, while the latter is the local instability resulted from the float neurons. We therefore categorize unsecured data into *Lipschitz vulnerable* and *float neuron vulnerable* accordingly.

4.2 Float Path and Network Robustness

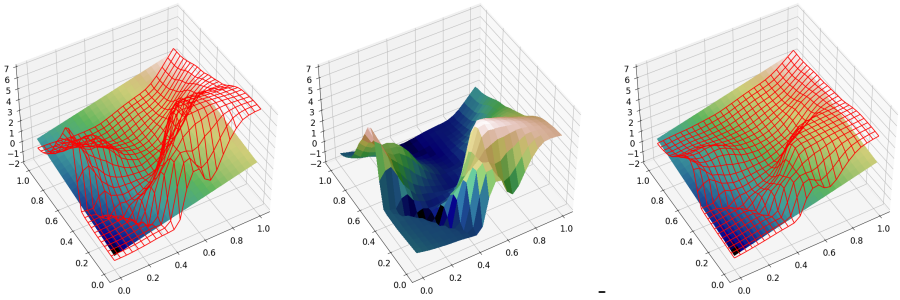
In this section, we further investigate the two types of vulnerability, and discuss the essence of smoothed classifier. Theorem 1 and Theorem 2 discuss the case where smoothed classifier can or cannot fix the misclassified example. We show that the smoothed classifier are able to fix the float neuron vulnerability but not the Lipschitz vulnerable example. This means that the boost of robust accuracy provided by smoothed classifier from averaging the protruding loss from the float path.

Theorem 1. *Let* f *be the base classifier. Given* $(x, y) \sim D$ *with* $f(x) = y$, *denote* $M(f(x), y) := min_{y' \neq y} |f(x)_y - f(x)'_y|$ *as the margin operator of prediction vector. If*

$$\|J(x)\| > \frac{M(f(x), y) + \mathbb{E}[M(Z^T(x, \mathcal{A}; \mathcal{R}), y)]}{r},$$

then for any smoothed classifier g *defined as above, there exist* $x' \in B(x, r)$ *such that* $g(x') \neq y$.

Theorem 1 is achieved by applying the randomized smoother on statement 4 of Lemma 2. It shows that the randomized classifier fails to correct the adversarial examples that caused by the high Lipschitz constant of the network. To understand this, consider the f has locally large Lipschitz constant, then their exists an x' such that, regardless of the sampling of g, the variation between $g(x')$ and $f(x)$ are always larger than the bound.



(a) Prediction Surface (b) Float Computation (c) SC-RFP Surface

Figure 1: The value of first element of prediction vector from VGG16 model trained on CIFAR10 given a 2D slice centered at a random data from test set. (a) The wireframe represents the prediction $f(x)$ while the surface is the sum of fixed path $Z^I(x, \mathcal{A}; \mathcal{R})$, respectively. (b) The sum of float path $Z^T(x, \mathcal{A}; \mathcal{R}) = f(x) - Z^I(x, \mathcal{A}; \mathcal{R})$. (c) The wireframe and surface are prediction of SC-RFP: $Z^I(x, \mathcal{A}; \mathcal{R}) + \eta Z^T(x, \mathcal{A}; \mathcal{R})$ with $\eta < 1$, and the surface is same as (a).

On the other hand, if g is able to fix the misclassified data, the expectation of the float path in the region has averagely lowed loss than that of the data, as stated below:

Theorem 2. *Let f and g be the base and smoothed classifier defined above. Given $(x, y) \sim D$, denote $x' \in B(x, r)$ as an adversarial example that mislead the base classifier but corrected by g without abstain:*

$$\arg\max_{m \in Y} f_m(x) = i, \quad \arg\max_{m \in Y} g_m(x) = y, i \neq y \tag{13}$$

then loss of float path is higher than that of g:

$$CE(\mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}), onehot(y)) > CE(g(\mathcal{Z}^T(x', \mathcal{A}; \mathcal{R})), onehot(y)) \tag{14}$$

where $CE(\cdot, \cdot)$ is the cross entropy loss, onehot(y) is the one hot embedding of label y.

Theorem 1 and Theorem 2 show that a smoothed classifier is not able to boost the performance of fixed path, while it is applied to reduce the sudden change of float path in a region. To be specific, if x' can be corrected by g, it is resulted from the sudden change provided by the float path. A higher confidence score of $g(x')$ can be achieved by reducing the weight of float path during the computation path. On the other hand, if x' is Lipschitz vulnerable, then regardless of the form of smoothed classifier, it cannot be fixed. This lead us to the theoretical basis of SC-RFP algorithm.

Theorem 3. *Denote f and g be the base and smoothed classifier defined as above. Let g' be a smoothed classifier with repressed float path and factor $\eta \in (\eta_{lb}, 1)$*

$$g' = g(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R})) + \eta(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R})), \tag{15}$$

where η_{lb} defined as:$\eta_{lb} = 1 - \mathbb{E}_{x,y \sim D}\left[\left|\frac{\mathcal{M}(f(x),y)}{\mathcal{M}(\mathcal{Z}^T(x,\mathcal{A};\mathcal{R}),y)}\right|\right]$. Then we have: $\hat{R}(g') > \hat{R}(g)$.

We illustrate the intuition with Figure 1 by showing the prediction, fixed path value and float path value. We first present the prediction and the fixed path value in Figure 1(a). The float path value is then computed as $Z^T(x, \mathcal{A}; \mathbb{R}) = f(x) - Z^I(x, \mathcal{A}; \mathbb{R})$ in Figure 1(b). At last, by repressing the float value, we achieve a locally stable prediction from SC-RFP.

4.3 Randomized Classifier with Repressed Activation Path

Algorithm 1 presents the empirical computation of the SC-RFP. At each block, we first compute the pre-activation value of x and $x + \epsilon$. Prior to passes to the activation function, we compare the activation pattern between $z_i(x)$ and $z_i(x + \epsilon)$ according to the separation Γ, and use a repression factor η to reduce the value of float path.

Algorithm 1 Smoothed Classifier with Repressed Float Path
Inputs: Network f with parameter θ, randomized algorithm g, input x, repress ratio η.
Outputs: Predicted label $\hat{y}$

```
while g sample noise ε do
    for Block i in Block 1, 2, ..., d − 1 do
        I_i^T ← A(z_i(x)) ≠ A(z_i(x + ε))
        z_i(x + ε) ← z_i(x + ε) − z_i(x + ε) × I_i^T
        x_{i+1}(x + ε) = σ(z_i(x + ε))
    counts ← φ_d(x_d(x + ε))
ĉ_A, ĉ_B ← top two indices in counts
n̂_A, n̂_B ← counts[ĉ_A], counts[ĉ_B]
if BinomPValue(n_A, n_A + n_B, 0.5) ≤ α then return ĉ_A else return Abstain
```

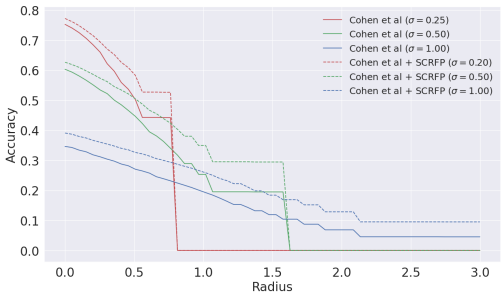
Figure 2 present the proportion of fixed neuron between x and $x + \epsilon$ of VGG16 network trained on CIFAR10. It shows that models trained with noised sample have relatively more stable activation pattern, as the ratio of fixed neurons are lower. In particular, the model trained with clean data has average fixed ratio around 72%, while after adding a minimum scale of noise $\sigma = 0.05$, it increases to around 90%. Intuitively, manipulating can greatly change the prediction result, while we show that there are only a small proportion of path are affected by our method.

5 Experiments

(a) $\epsilon \sim N(0, 0.1)$ (b) $\epsilon \sim N(0, 0.25)$

Figure 2: Proportion of fixed neuron between x and $x + \epsilon$ for VGG16 models trained on CIFAR10 with noised data at different scale. From the left to right models the data are: clean data, $\sigma = 0.05$, $\sigma = 0.10$ and $\sigma = 0.25$. Figure 2(a) and 2(b) show the ratio given $\epsilon \sim N(0, 0.1)$ and $\epsilon \sim N(0.25)$.

As SC-RFP is built on a base smoothed classifier, we measure the method on different model structures and randomized smoothing algorithms. The experiments are performed on CIFAR10 and ImageNet datasets. For the CIFAR10 dataset, we compare our method with the benchmark classifier proposed by Cohen et al. (2019) with VGG16 network to show the effectiveness of our method. For the ImageNet dataset, we choose ResNet50 as model architecture and add the adversarial smoothed classifier Salman et al. (2019) as the base model. We also compare our model with recent work Jeong & Shin (2020); Jeong et al. (2021) to obtain a general evaluation. In line with previous works, we use certifiable accuracy at different radius computed by Cohen et al. (2019) as the metric.



(a) $\epsilon \sim N(0, 0.1)$

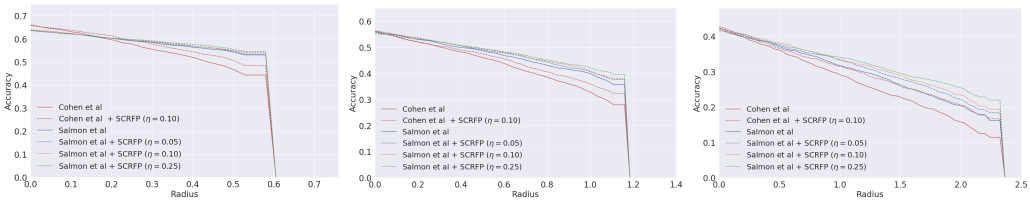
Figure 3: Certified Accuracy of base methods and SC-RFP ($\eta = 0.1$) with different level of noise on CIFAR10. The solid and dashed lines represent benchmark and SC-RFP.

CIFAR10. We train the VGG16 network with different scales of noisy samples. Each of the models is trained for 200 epochs with SGD optimizer and an initial learning rate of 0.1, which decays after 60, 120, and 160 epochs with a rate of 0.2. Table 1 compares the robust accuracy of SC-RFP and benchmark. Generally, our SC-RFP increases have an increasingly robust accuracy between 6% to 10% compared with the benchmark. We notice that by repressing the float path, SC-RFP is able to improve the accuracy of clean data, which supports our analysis that the float neuron vulnerability is one of the causes of misclassification. Figure 3 present the robust accuracy at different noise scale. We find that the performance boost from SC-RFP increases as the radius increases for all the models.

Method	0.25	0.5	0.75	1.0	1.25	1.5	2.0	2.5	Clean
Cohen et al. (2019)	66.0	50.2	44.2	25.4	19.5	19.5	6.9	4.5	75.3
+ SC-RFP	69.4	58.3	52.7	34.9	29.5	29.5	12.9	9.5	77.2

Table 1: Certified robust accuracy for Benchmark and SC-RFP on CIFAR10.

Experiments on ImageNet. We train ResNet50 using the benchmark model (Cohen et al. (2019)) and SmoothAdv (Salman et al. (2019)) on ImageNet-2012 for 80 epochs with same training scheduler and optimizers as the original work. Each of the model are certified with same level of noise as they were trained.



(a) $\sigma = 0.25$ (b) $\sigma = 0.5$ (c) $\sigma = 1.0$

Figure 4: Certified Accuracy of base methods and SC-RFP on ImageNet. Model are trained and tested with (a) $\sigma = 0.25$, (b) $\sigma = 0.5$ and (c) $\sigma = 0.5$. The solid and dashed lines represent benchmark and SC-RFP with different repression factor.

Table 2 compares the certified robust accuracy of different models at different radius on ImageNet. It shows that when deployed on the PGD+noise training (Salman et al. (2019)), the SC-RFP has the highest robust accuracy at all the radius. Moreover, an increasing boost in the robust accuracy is observed as the radius increase, which is consistent with our previous analysis of CIFAR10 result.

Figure 4 provides more details on our experiments. For a noise level of $\sigma = 0.25$, SmoothAdv and SmoothAdv + SC-RFP has neglectable difference, while in 4(c), the SC-RFP ($\sigma = 0.25$) increases the robust accuracy by around 5%. By introducing SC-RFP ($\sigma = 0.1$) to the benchmark classifier, it can achieve comparable results with other state-of-art models when l_2 radius is around 1.5.

l_2 radius	0.25	0.5	0.75	1.0	1.5	2.0	Clean
Jeong & Shin (2020)	59.8	49.8	44.7	39.3	28.1	22.6	-
Jeong et al. (2021)	46.7	38.2	30.3	26.8	15.7	12.1	-
Cohen et al. (2019)	57.9	46.4	41.9	32.6	22.8	15.6	**66.1**
+ SC-RFP (ours)	59.4	50.3	41.5	35.6	26.1	20.2	65.8
Salman et al. (2019)	59.2	54.1	51.5	39.6	26.2	20.1	63.6
+ SC-RFP (ours)	**59.8**	**55.4**	**53.6**	**42.0**	**29.6**	**25.2**	63.9

Table 2: Certified robust accuracy for models with different methods on ImageNet

We find two insightful observations from the above results. First, introducing the repression factor to the float path has a minimum negative effect on the standard accuracy, while can greatly boost the robust accuracy. This supports our discussion that instability vulnerable examples are caused by the float path, and can be cured by reducing their contribution. Second, we find that as the radius grows, the boost from our method increases. This is consistent with the theoretical basis of SC-RFP as well as the result from CIFAR10 dataset. When size of the perturbation grows, the uncertainty of the randomized algorithm increases along with the number of float paths as Figure 2 suggests. This results in a higher abstain rate and a higher possibility of misclassification. By repressing the value of the float path, the deviation of $g(x + \epsilon)$ from $f(x)$ is reduced, therefore a more stable result can be achieved.

6 Conclusion

In this work, we introduce SC-RFP algorithm that can improve the performance of a randomized smoothing classifier. We first introduce a framework for describing the local activation status of neurons and show that most of the neurons are locally stable, while the others can greatly affect the model prediction. By decomposing the computational graph of the network, we find that the boost of robust accuracy provided by the smoothed classifier is averaging the deviation of float paths. Based on this, we suggest further repressing the value of float paths with SC-RFP method. The experiments show that our method can improve the performance of smoothing models.

REFERENCES

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 278–287, 2017.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

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

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pp. 2596–2604, 2019a.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, pp. 361–370, 2019b.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems*, 34:22745–22757, 2021.

Jongheon Jeong and Jinwoo Shin. Consistency regularization for certified robustness of smoothed classifiers. *Advances in Neural Information Processing Systems*, 33:10558–10570, 2020.

Jongheon Jeong, Sejun Park, Minkyu Kim, Heung-Chang Lee, Do-Guk Kim, and Jinwoo Shin. Smoothmix: Training confidence-calibrated smoothed classifiers for certified robustness. *Advances in Neural Information Processing Systems*, 34:30153–30168, 2021.

Ziping Jiang, Yunpeng Wang, Chang-Tsun Li, Plamen Angelov, and Richard Jiang. Delve into activations: Towards understanding dying neuron. *IEEE Transactions on Artificial Intelligence*, 2022.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

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)*, pp. 656–672. IEEE, 2019.

Sungyoon Lee, Jaewook Lee, and Saerom Park. Lipschitz-certifiable training with a tight outer bound. *Advances in Neural Information Processing Systems*, 33:16891–16902, 2020.

Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *arXiv preprint arXiv:1905.12105*, 2019.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. *Advances in neural information processing systems*, 32, 2019.

Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *arXiv preprint arXiv:2006.08859*, 2020.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

Andras Rozsa, Manuel Gunther, and Terrance E Boult. Towards robust deep neural networks with bang. *arXiv preprint arXiv:1612.00138*, 2016.

Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6542–6551, 2018.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pp. 5276–5285. PMLR, 2018.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems*, 31, 2018.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.

A MATHEMATICAL PROOFS.

This section presents the mathematical proofs.

Proof (Lemma 1)**.** *We first show that the pre-activation transformation can be represented by a matrix. As ϕ_i is a linear affine, we denote $\phi_i(x) = W_i x_i$. Combing with batch normalization layer, the mapping from input x_i to pre-activation is:*

$$z_i = \gamma_i \frac{\phi_i(x) - \hat{\mu}}{\hat{\sigma}} + \beta_i = \frac{\gamma}{\hat{\sigma}}\phi_i(x) - \frac{\gamma\hat{\mu}}{\hat{\sigma}} + \beta_i = D_i(\frac{\gamma}{\hat{\sigma}})W_i x - \frac{\gamma\hat{\mu}}{\hat{\sigma}} + \beta_i \tag{16}$$

where $D_i(\cdot)$ is the diagonal operator, $\hat{\mu}$ and $\hat{\sigma}$ are the mean and variance parameter of the ψ_i. Denote $W'^{(i)} = D_i(\frac{\gamma}{\hat{\sigma}})W_i$, $\beta'_i = (\beta_i - \frac{\gamma\hat{\mu}}{\hat{\sigma}})/n_i$. We proof this byh deduction. For $i = 1$:

$$z_i = \sum_{j=0}^{n_0} W'^{(1)}_{ij} d_{\mathcal{A}_{1,i}} x_{0j} + n_1 \times \beta'^{(1)}_i.$$

where $\mathcal{A}_{1,i}$ is the activation pattern of z_{1i}. Since all the path in layer i ends at i are $\{(1,i),(2,i),\ldots,(n_0,i)\}$. Equation 10 holds.

Now we assume equation holds for $i = p$.

$$\begin{aligned}
z_{pi} &= \sum_{j=0}^{n_i} W'^{(i)} x_{ij} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} z_{i-1,j} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} d_{\mathcal{A}_{i,j}} z_{i-1,j} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} d_{\mathcal{A}_{i,j}} (\sum_{\forall \zeta_i = j} x_{j,\zeta_j} \prod_{m=p}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m,\zeta_{m-1})} \\
&+ \sum_{o=p-1}^{i-1} \sum_{\forall \zeta_i = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m,\zeta_{m-1})}) + \beta'^{(i)}_j \\
&= \sum_{\forall \zeta_p = k} x_{j,\zeta_j} \prod_{m=j+1}^{p} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m,\zeta_{m-1})} + \sum_{o=j}^{p-1} \sum_{\forall \zeta_p = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m,\zeta_{m-1})}
\end{aligned} \tag{17}$$

Proof (Lemma 2)**.** *Statement 1 is directly from the definition.*

Now we consider statement 2. For any $x_1, x_2 \in R$, and an index of neuron (i,j). If $\hat{a}_{ij}(x_1) = \hat{a}_{ij}(x_2)$, (i,j) is a fixed neuron in R: $(i,j) \in I^X$. Denote the pattern a_ij. Then for all $x_1 \in R$, $z_{ij;x_1,\theta} = z_{ij;x_2,\theta}$. Therefore, $x_1 \in \mathcal{R}(\mathcal{A}^X)$.

Statement 3 directly comes from Lemma 1. For statement 4, we have:

$$\begin{aligned}
f(x) - f(x') &= \big(\mathcal{Z}^I(x,\mathcal{A};\mathcal{R}) + \mathcal{Z}^T(x,\mathcal{A};\mathcal{R})\big) - \big((\mathcal{Z}^I(x',\mathcal{A}';\mathcal{R}) + \mathcal{Z}^T(x',\mathcal{A}';\mathcal{R})\big) \\
&= \big(\mathcal{Z}^I(x,\mathcal{A};\mathcal{R}) + \mathcal{Z}^T(x,\mathcal{A};\mathcal{R}) - \mathcal{Z}^I(x',\mathcal{A};\mathcal{R}) - \mathcal{Z}^T(x',\mathcal{A};\mathcal{R})\big) + \\
&\quad \mathcal{Z}^T(x',\mathcal{A};\mathcal{R}) - \mathcal{Z}^T(x',\mathcal{A};\mathcal{R})
\end{aligned} \tag{18}$$

Notice that, $\mathcal{Z}^I(x',\mathcal{A};\mathcal{R})$ is the collection of all fixed path, therefore is unrelated with the change of $\mathcal{A}$. We have $\mathcal{Z}^I(x',\mathcal{A};\mathcal{R}) = \mathcal{Z}^I(x',\mathcal{A}';\mathcal{R})$. The former part of above equation is equals to $J(x)(x - x')$, which reaches to statement 4.

Before we move to Theorem 1, we need the following two lemmas. Lemma 4 suggests that, given a fixed path ζ, the expectation for the randomized algorithm $g(\zeta(x))$ is equal to the $\zeta(x)$ when g satisfies certain conditions. In fact, the condition here just assume that the randomly sampled noise has equal probability at each direction, which is a natural assumption for most sampling procedure. Before we start, we additional introduce the definition of a subpath of a path as below:

Definition 6. *A sub-path of a path consists of several consecutive elements of γ:*

$$\zeta_{(i,j)} = \{\zeta_j, \zeta_{j+1} \ldots, \zeta_i\} \subseteq \zeta \tag{19}$$

We use $\zeta_{(i,j}(x)$ to represent the compuataion graph from (j,ζ_j) to (i,ζ_i):

$$\zeta_i(x) = \zeta_{(i,j)} \circ \zeta_{j,0}(x) \tag{20}$$

Lemma 3. *Denote N as a neural network as Definition 4 with mapping function f, ζ is a fixed path in $B_2(x,p)$. Assume g is a randomized algorithm that applies noise ϵ from certain distribution D_{noise} on the computation path of ζ:*

$$g(\zeta(x)) = \zeta_{d,i}(\zeta_{i,0}(x) + \epsilon), i \in \{0, 1, \ldots, d-1\}, \epsilon \sim D_{noise} \tag{21}$$

If the direction of ϵ is uniformly distributed, then the expectation of $g(\zeta(x))$ is equals to $\zeta(x)$, that is:

$$\forall \|\eta_1\| = \|\eta_2\| = 1, P(\frac{\epsilon}{\|\epsilon\|} = \eta_1) = P(\frac{\epsilon}{\|\epsilon\|} = \eta_2) \Rightarrow E[g(\zeta(x))] = \zeta(x), \text{ if} \tag{22}$$

where ϵ is the noise generated for randomized algorithm g.

Proof. *Assume that g samples noise at layer k. Then for any g, $\zeta_(k,0) = g(\zeta_{k,0})$, which we denote as x_k. Since the activation function is piecewise linear, the mapping of $\zeta_{d,k}(\cdot)$ is linear. We have:*

$$\sum_{i}^{n_k} \frac{\partial^2 y_j}{\partial z_{kj}^2} = 0. \tag{23}$$

$\zeta_{d,k}$ is harmonic. Therefore, given any $B(x,r)$ with radius $r > 0$:

$$\zeta_{d,k}(x) = \frac{1}{n\omega_n r^{n-1}} \int_{\partial B(x,r)} \zeta d\sigma, \tag{24}$$

where ω_n is the volume of the unit ball in n dimensions, σ is the $(n-1)$ dimensional surface measure. Now we consider the expectation of $g(x)$:

$$\begin{aligned} \mathbb{E}(g(x)) &= \int p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu \\ &= \int \int_{\|\epsilon\|=r} p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu dr, \end{aligned} \tag{25}$$

where μ is the probability measure of ϵ D_{noise}. As the noise sampling is assumed to be direction irrelevant, the measure of $p(\epsilon | \|\epsilon\|_2 = r)$ is uniformly distributed given radius r. Therefore:

$$\int_{\|\epsilon\|=r} p(\epsilon)\zeta_{d,k} d\mu = P(\|\epsilon\|_2 = r)\zeta_{d,k}. \tag{26}$$

Equation 25 then equals:

$$\begin{aligned} \mathbb{E}(g(x)) &= \int p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu \\ &= \int P(\|\epsilon\|_2 = r)\zeta_{d,k} dr \\ &= \zeta_{d,k}(x) \end{aligned} \tag{27}$$

Lemma 4. *Let f and g be the mapping function and randomized classifier defined as above. Given radius r such that, almost surely, $\mathbb{Z}^T(B(x,r)) = \emptyset, \forall (x,y) \sim D$, then the accuracy of base classifier and the naive smoothed classifier are same, that is:*

$$\mathbb{E}_{(x,y)\sim D}\left[\arg\max_{m \in Y} g_m(x) = y\right] = \mathbb{E}_{(x,y)\sim D}\left[\arg\max_{m \in Y} f_m(x) = y\right]$$

Lemma 4 is the achieved by directly applying Lemma 3 on the all the computational graph of the network. It shows that if all the neurons have locally stable activation pattern with respect to the distribution of dataset, the smoothed classifier provides identical accuracy with the base classifier.

Proof. *Statement 4 of 2 suggests:*

$$f(x) - f(x') = J(x)(x - x') + \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}) - \mathcal{Z}^T(x', \mathcal{A}'; \mathcal{R}) \tag{28}$$

As $\|J(x)\| > \frac{M(f(x),y) + \mathcal{M}(\mathbb{E}[Z^T(x,\mathcal{A};\mathcal{R})],y)}{r}$, there exist x' and i such that:

$$f_i(x') > f_y(x') + \mathbb{E}[Z_y^T(x, \mathcal{A}; \mathcal{R}) - Z_i^T(x, \mathcal{A}; \mathcal{R})] \tag{29}$$

Therefore,

$$\begin{aligned} g_i(x) - g_i(x') &> g_y(x) - g_y(x') + M(f(x), y), \\ g_i(x') &> g_y \end{aligned} \tag{30}$$

Proof (Theorem 2). *Since* $\arg\max_{m\in Y} f_m(x) \neq y$, $\arg\max_{m\in Y} g_m(x) = y$, *there exist* $i \in \{1,\dots,c\}$:

$$\begin{aligned} g_y(x) &> f_y(x) \\ g_i(x) &< f_i(x) \end{aligned} \tag{31}$$

Then we have:

$$g_y(x) - g_i(x) \geq f_y(x) - f_i(x) \tag{32}$$

Introducing Statement 4 of Lemma 2 and Lemma 4:

$$g_i(Z^T(x,\mathcal{A};\mathcal{R})) - g_y(Z^T(x,\mathcal{A};\mathcal{R})) < Z_i^T(x,\mathcal{A};\mathcal{R}) - Z_y^T(x,\mathcal{A};\mathcal{R}). \tag{33}$$

Moreover, since the prediction is not abstained.

$$\mathbb{E}[g_y(x)] - f_y(x) > \mathbb{E}[g_j(x)] > f_j(x), \forall j \neq i, y \tag{34}$$

This directly leads us to the result.

Proof (Theorem 3). *We consider the following cases. Given* $x \sim D$.

If x *is Lipschitz vulnerable. Then their exists* i *and direction* η *such that:* $g(x+\eta)_y < g(x+\eta)_i$. *From Theorem 1, any* g *fails to correct* x.

If x *is misclassified by* g *and float neuron vulnerable. From Theorem 2, we have:*

$$g_i(Z^T(x,\mathcal{A};\mathcal{R})) - g_y(Z^T(x,\mathcal{A};\mathcal{R})) < Z_i^T(x,\mathcal{A};\mathcal{R}) - Z_y^T(x,\mathcal{A};\mathcal{R}). \tag{35}$$

This means the local average of $Z_i^T(x,\mathcal{A};\mathcal{R})$ *is higher than* $Z^T(x,\mathcal{A};\mathcal{R})$. *Then*

$$\begin{aligned} g_i'(x) &= Z_i^I(x;\mathcal{A};\mathcal{R}) + \eta g_i(Z^T(x,\mathcal{A};\mathcal{R})) \leq Z_y^I(x;\mathcal{A};\mathcal{R}) + g_i(Z^T(x,\mathcal{A};\mathcal{R})) = g_i(x) \\ g_i'(x) &= Z_y^I(x;\mathcal{A};\mathcal{R}) + \eta g_y(Z^T(x,\mathcal{A};\mathcal{R})) \leq Z_y^I(x;\mathcal{A};\mathcal{R}) + g_y(Z^T(x,\mathcal{A};\mathcal{R})) = g_i(x). \end{aligned} \tag{36}$$

Therefore, $CE(g'(x), onehot(y)) < CE(g'(x), onehot(y))$. *The upper bound 1 is obtained If* x *is both classified correctly by* g *and* f. *We need a lower bound on* η *to ensure* $g'(x)$ *does not alter the prediction of* g. *For* $\eta > \eta_{lb}$, *assume* i *is the element that closest to* y *we have:*

$$\begin{aligned} \mathcal{M}(f(x)) &= \mathcal{M}(Z^I(x,\mathcal{A};\mathcal{R})) + \mathcal{M}(Z^T(x,\mathcal{A};\mathcal{R})) \\ \mathcal{M}(g'(x)) &= \mathcal{M}(Z^I(x,\mathcal{A};\mathcal{R})) + \eta\mathcal{M}(Z^T(x,\mathcal{A};\mathcal{R})) \end{aligned} \tag{37}$$

Solving above equation, we have $\mathcal{M}(g'(x), y) > 0$ *when* $\eta_{lb} = 1 - \mathbb{E}_{x,y\sim D}\left[\left|\frac{\mathcal{M}(f(x),y)}{\mathcal{M}(Z^T(x,\mathcal{A};\mathcal{R}),y)}\right|\right]$.

B Extra Tables and Figures

This section present the complete experiment results. Table 3 shows the certified accuracy at different l_2 radius level.

Table 4 presents the experiment result of our model certified with different repression rate. Among the different repression rate, we find that when $\eta = 0.25$ SC-RFP provides the best performance on increasing the model robust accuracy for perturbation with large size, while the standard accuracy and are slightly damaged. This is also observed on previous works.

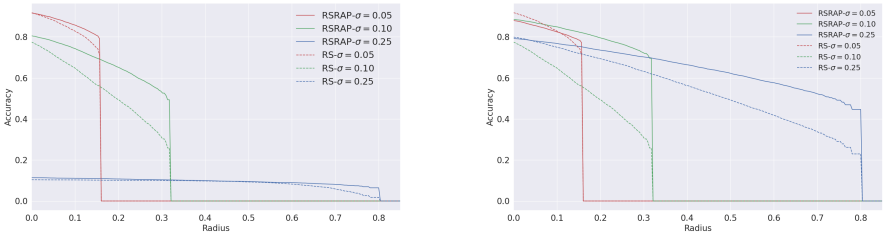
σ	Method	0.25	0.5	0.75	1.0	1.25	1.5	2.0	2.5	Clean
0.25	Cohen et al. (2019)	66.0	50.2	44.2	0.0	0.0	0.0	0.0	0.0	75.3
	+ SC-RFP	69.4	58.3	52.7	0.0	0.0	0.0	0.0	0.0	77.2
0.50	Cohen et al. (2019)	53.4	44.7	34.0	25.4	19.5	19.5	0.0	0.0	60.3
	+ SC-RFP	57.3	50.5	42.4	34.9	29.5	29.5	0.0	0.0	62.7
1.00	Cohen et al. (2019)	31.2	27.1	23.2	19.3	15.3	11.9	6.9	4.5	34.7
	+ SC-RFP	36.0	32.6	29.4	25.9	22.2	18.4	12.9	9.5	39.1

Table 3: Certified robust accuracy for models with different methods on CIFAR10 dataset.

Figure 5 present the model trained with $\sigma = 0.05$ and $\sigma = 0.25$, but under different level of noise. We find the model trained with slightly noised sample can provide effective robustness.

σ	l_2 radius	0.25	0.5	0.75	1.0	1.5	2.0	Clean
0.25	Jeong & Shin (2020)	59.8	49.8				-	
	Jeong et al. (2021)	46.7	38.2					-
	Cohen et al. (2019)	57.9	46.4					**66.1**
	+ SC-RFP (ours)	59.4	50.3					65.8
	Salman et al. (2019)	59.2	54.1					63.6
	+ SC-RFP (ours $\eta = 0.05$)	59.6	54.5					63.8
	+ SC-RFP (ours $\eta = 0.10$)	59.5	54.8					63.9
	+ SC-RFP (ours $\eta = 0.25$)	**59.8**	**55.4**					63.6
0.50	Jeong & Shin (2020)	**53.7**	**49.9**	44.7	39.3			-
	Jeong et al. (2021)	39.1	34.9	30.3	26.8			-
	Cohen et al. (2019)	51.3	46.1	39.9	32.6			**56.5**
	+ SC-RFP (ours)	51.1	47.3	41.5	35.6			55.9
	Salman et al. (2019)	52.6	48.7	44.0	39.6			56.0
	+ SC-RFP (ours $\eta = 0.05$)	52.7	49.1	45.2	40.5			56.4
	+ SC-RFP (ours $\eta = 0.10$)	52.9	49.4	45.4	40.8			56.2
	+ SC-RFP (ours $\eta = 0.25$)	52.8	49.6	**46.0**	**42.0**			55.2
1.00	Jeong & Shin (2020)	40.0	**38.5**	35.4	32.6	28.1	22.6	-
	Jeong et al. (2021)	29.7	26.2	23.0	20.6	15.7	12.1	-
	Cohen et al. (2019)	39.1	36.1	32.5	29.1	22.8	15.6	**42.4**
	+ SC-RFP (ours)	39.9	36.5	33.9	31.5	25.6	19.9	42.9
	Salman et al. (2019)	39.5	37.1	34.7	31.8	26.2	20.1	42.0
	+ SC-RFP (ours $\eta = 0.05$)	**40.2**	37.5	35.2	33.3	28.0	21.8	42.2
	+ SC-RFP (ours $\eta = 0.10$)	39.9	38.1	35.7	33.4	28.9	23.	41.9
	+ SC-RFP (ours $\eta = 0.25$)	39.7	37.8	**35.9**	**34.2**	**29.6**	**25.2**	41.6

Table 4: Certified robust accuracy for models with different methods on ImageNet dataset



(a) $\epsilon \sim N(0, 0.1)$ (b) $\epsilon \sim N(0, 0.1)$

Figure 5: Certified Accuracy of base methods and SCRFP ($\eta = 0.1$) with different level of noise on CIFAR10. The solid and dashed lines represent benchmark and SC-RFP. The models are tested under different level of noise.