# 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 paths and float paths 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 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 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 respect to 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. 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 introduce a novel way to apply the complex theory to practical.

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 malicious 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 be broken by more advanced attacks.

To address the issue, certifiable training and randomized smoothing methods aim to provide a certified region, within which the input data are free from attack. By viewing the training as a convex optimization problem, dual relaxation approaches apply duality to provide a solid 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)). As verifiable training methods often come with a compromise of performance, recent works focus on bridging the gap between adversarial and verifiable training to address the scalability and accuracy issue (Xiao et al. (2018); Balunović & Vechev (2020); De Palma et al. (2022)).

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 fail 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). 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 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. Inspired by the theoretical investigation, Jordan et al. (2019) introduces an algorithm named GeoCert that computes the $l_p$ bound of the network with a piecewise linear activation function. Zhang et al. (2022) proposes an algorithm that systematically searches the adversarial example based on the activation space of ReLU network.

## 3 PRELIMINARIES

### 3.1 NOTATIONS

Let $N$ be a $d$ block feedforward neural network for classification task with measure zero parameter set $\theta$ with respect to Lebesgue measure. Each of the block $h_i$ consists of a linear affine $\phi_i$, an optional batch-normalization layer $\psi_i$, and piecewise linear activation function $\sigma_i$, while the last block $h_d$ omits the activation function. Consider $D$ as the distribution of a classification problem with $c$ classes from $R^{n_0}$ to $Y = \{1, 2, \ldots, c\}$, network $N$ computes a function $f : R^{n_0} \to R^c$, where $f$ is a composition of $d$ blocks $f = h_n \circ h_{d-1} \cdots \circ h_1$.

For every $(x, y) \sim D$, the network computes a probability for each class $f(x) \in R^c$ and predicts the label of $x$ as 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 $x$. We use a tuple pair $(i, j)$ to denote the

$j$-th neuron of layer $i$ and a set $\mathcal{I} = \cup_{i=1}^{d-1}\{(i,j)|j \in \{1,\ldots,n_i\}\}$ to denote the index collection of neurons in intermediate layers, where $n_i$ is the output size of layer $i$.

### 3.2 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, robust models often incur increased stableness and impaired expressivity. 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 $(x,y) \sim D$, $g$ employs a certain degree of randomness during the induction of $f$. For instance, smoothed classifier $g$ computes the probability of $f(x+\epsilon)$ belongs to class $i$ given $\epsilon \in \mathcal{N}(0,\sigma^2 I)$(Cohen et al. (2019)):

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

With certain confidence level $\alpha$, the lower bound of $\underline{p_A}$ of random variable $g_y(x)$ and the upper bound $\overline{p_B}$ of the probability of second possible class $\max m \in Y g_y(x)$ can be computed. This induces a certified radius of classifier $g(x)$:

$$r = \frac{\sigma}{2}\big(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})\big). \tag{4}$$

For every $x' \in B_2(x,r)$, if $\arg\max_{m\in Y} g_m(x) = Y$, then $\arg\max_{m\in Y} g_m(x') = y$. In other words, with confidence level $\alpha$, every $x'$ within the radius can be correctly classifier by the smoothed classifier. The smoothed classifier can also appear in other forms depends on the randomness and training algorithm (Levine et al. (2019); Salman et al. (2019); Jeong & Shin (2020)).

### 3.3 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 a network defined as Section 3.1. Denote $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_q\}$ as a set of breakpoints that separates the domain of activation function into $q+1$ intervals $U = \{U_0, U_1, \ldots, U_q\}$. An activation pattern of $N$ is defined as an indexed family that assigns each neuron a sign to represent the status of its pre-activation in $U$:*

$$\mathcal{A} := \{a_{ij}\,|a_{ij} \in \{0,1,\ldots,q\}, (i,j) \in \mathcal{I}\}.$$

*Given an activation pattern $\mathcal{A}$, the activation region of $\mathcal{A}$ is defined as:*

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

*Conversely, given $x \in R^{in}$, we denote the activation pattern of neuron $(i,j)$ at $x$ as:*

$$\hat{a}_{ij}(x;\theta,\sigma,\Gamma) = \begin{cases} i, & z_{ij}(x;\theta) \in U_i, \\ -1, & z_{ij}(x;\theta) \in \{\gamma_1,\gamma_2,\ldots,\gamma_q\} \end{cases} \tag{6}$$

*The activation pattern of $x$ is the collection of the all the pattern of neurons:*

$$\hat{\mathcal{A}}(x;\theta,\sigma,\Gamma) = \{\hat{a}_{ij}(x)|(i,j) \in \mathcal{I}\} \tag{7}$$

**Activation Region Operator and its Inverse.** The generalized activation pattern describes the activation status of each unit at the intermediate layers. Given an activation pattern, $\mathcal{R}(\cdot)$ finds the region, such that, for every $x \in \mathcal{R}(\mathcal{A}; \theta, \sigma, \Gamma)$, the pre-activation value of each neuron $z_{ij}(x; \theta)$ locates within the interval $U_{a_{ij}}$ that determined by its pattern $a_{ij} \in \{0, 1, \ldots, q\}$.

On the basis of previous works (Hanin & Rolnick (2019b)Raghu et al. (2017)), we further introduce an inverse operator $\hat{\mathcal{A}}(x; \theta, \sigma, \Gamma)$ that computes the activation pattern of an input $x \in R^{n_0}$ in Definition 1. This allows us to study the robustness of network by investigating the stability of activation pattern in the sphere $B(x, r)$. With quantitatively analysis of the pattern, we decompose the computational graph of single input $x$ in Lemma 2 and the prediction difference between inputs $x, x' \sim D$ in Lemma 1. When dependencies are fixed, we abbreviate Equation 5 and 7 as $\mathcal{R}(\mathcal{A})$ and $\hat{\mathcal{A}}(x)$.

**From Single Region to its Neighbor.** With the scale of network grows, the mean volume of single activation region decreases exponentially (Hanin & Rolnick (2019a)). Therefore, it is necessary to generalize our investigation from single region to its neighbor. Equation 5 suggests that 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 intersection 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 neuron set $\mathcal{I} \backslash \mathcal{I}^c$ then equals to remove related intersection. This can be viewed as merging several close activation regions into one.

From a geometric point of view, the input space is partitioned into small activation regions by a set of hyperplanes $\{H_{ijk}(\theta)\}$, where: $H_{ijk}(\theta) := \{x \in U | z_{ij}(x; \theta) = \gamma_k\}$ is determined by neuron $(i, j)$ and breakpoint $\gamma_k$. This means that two adjacent activation regions $\mathcal{R}_1, \mathcal{R}_2$ can be merged into one by removing the bent-hyperplanes that separate them.

Combining with Equation 8, the every data $x$ in the merged region has same activation pattern expect for the neuron $(i, j)$ based on which the removed hyperplane is defined on. We use incomplete activation pattern $A^c$ to denote the stable neurons, and describe the merged region as:

**Definition 2.** *Let $N$ be a network defined as Section 3.1. 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 merged activation region of $\mathcal{A}_{\mathcal{I}^c}$ is denoted as: $\mathcal{R}(\mathcal{A}^c) = \bigcap_{(i,j) \in \mathcal{I}^c} \{x \in R^{in} | z_{ij}(x) \in U_{a_{ij}}\}$.*

# 4 From Float Path to Model Robustness

## 4.1 Float Neuron and Path

Definition 2 proposes a way to describe merged activation regions with incomplete activation patterns. However, the merged regions are irregular spaces determined by bent-hyperplanes that defined by complex function. Analyzing the properties of such space is not feasible. Therefore, we use following definition to describe the neurons in a subset $R \subset R^{n_0}$.

**Definition 3** (Float and Fixed Neuron)**.** *Let $N$ be a network defined as Section 3.1. For any space $R \subset R^{n_0}$, if a neuron $z$ has the same pattern for any $x \in R$, we refer it as 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:*

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

To build a connection between Definition 2 and Definition 3, we propose the following lemma to show: (1) the fixed neurons and float neurons are complementary sets in $\mathcal{I}$ and (2) any subset $R \in R^{n_0}$ can be covered by a merged region defined by the fixed neurons of $R$.

**Lemma 1.** *Let $N$ be a neural network defined as Section 3.1. Given $R \subset R^{n_0}$, denote $\mathcal{I}^X$ and $\mathcal{I}^T$ are the set of float neuron and fixed neuron in $R$. Then:*

*1.* $\mathcal{I}^X(R) \bigcup \mathcal{I}^T(R) = \mathcal{I}$ *2.* $R \subset \mathcal{R}(\mathcal{A}_{\mathcal{I}^X})$

Now we are able to identify the stability of paths in the computational graph of network $N$ with the following two definitions:

**Definition 4.** *Let $N$ be a network defined as Section 3.1. A path of $N$ is a set of neuron indexes:*

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

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

**Definition 5** (Float Path and Fixed Path)**.** *Let $N$ be a neural network defined as Section 3.1. Given a subspace $R \in R^{n_0}$, a path $\zeta$ of $N$ is a float path in $R$ if there exist a neuron $(i, \zeta_i) \in \zeta$ is a float neuron, otherwise it is a fixed path. The collection of float path and float path are denoted as:*

$$\begin{aligned} \mathcal{Z}^T(R) &= \{\zeta | \exists \zeta_i \in \zeta, (i, \zeta_i) \in \mathcal{I}^T(R), \} \\ \mathcal{Z}^I(R) &= \{\zeta | \forall \zeta_i \in \zeta, (i, \zeta_i) \in \mathcal{I}^I(R), \} \end{aligned} \tag{10}$$

To understand the motivation of above definition, consider the simple case that there is only 1 neuron $(i, j)$ is float neuron in region $R \subset R^{n_0}$. The only non-linearity in $R$ is provided by neuron $(i, j)$. The computational graph of $N$ can be decomposed into *float paths* and *fixed paths* according to whether $(i, j)$ it on the path. This is formally described as the first statement in Theorem 1. Moreover, given $x, x' \in R$, the difference between $f(x)$ and $f(x')$ can be viewed the combination of a linear function and an unstable function with high non-linearity as suggested by the second statement in Theorem 1. The remaining problem is how to decompose the computational graph of network $N$, which we discuss in the following section.

### 4.2 Decomposing the Computational Graph

Now that we introduced the framework for describing neuron status, we aim to build a connection between mapping function $f$ and activation pattern.

**Lemma 2** (Decomposition of Neural Network)**.** *Let $N$ be a neural network defined as Section 3.1. Given activation pattern $\mathcal{A}$, for any $x \in \mathcal{R}(\mathcal{A}; \theta, \sigma, \Gamma)$. The $k$-th component of output of layer $i$ can be represented as:*

$$y_{ik}(x) = \sum_{\forall \zeta_i = k} x_{j,\zeta_j} \prod_{m=j+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} + \sum_{o=j}^{i-1} \sum_{\forall \zeta_i = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} \tag{11}$$

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

Lemma 2 decomposes $f$ in the manner of input and activation pattern. Equation 11 shows that $y_{ij}(x)$ can be decomposed into summation of paths, while the value of each path depends on either input $x$ (former part) or bias $\beta$ at each layer (latter part), as well as the pattern of neurons on the path. We use $\zeta_{(i,j)}(v, \mathcal{A})$ to denote the value of a path $\zeta$ given $v$ as the input of that path:

$$\zeta_{(i,j)}(v, \mathcal{A}) := v \prod_{m=j+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})}. \tag{12}$$

In a single activation region $\mathcal{R}$, $\forall x, x' \in \mathcal{R}$ have same activation pattern $\mathcal{A}$. The paths from bias $\zeta_{(i,j)}(\beta^{(j)}_{\zeta_j}, \mathcal{A})$ are constant, while that from input $\zeta_{(i,j)}(x_{j,\zeta_j}, \mathcal{A})$ are linear to $x_{j,\zeta_j}$. On the other hand, for $x, x'$ with different activation patterns $\mathcal{A}, \mathcal{A}'$, each of the path can be categorized as either fixed or float. Since neurons on fixed paths has same pattern at $x$ and $x'$, the above linearity remains unchanged, which means $\zeta_{(i,j)}(v, \mathcal{A}) = \zeta_{(i,j)}(v, \mathcal{A}')$ if $\zeta$ is a fixed path. In other words, given region $R \subset R^{n_0}$, the float paths represent the linear part of $f$ in $R$, while the fixed paths are the aggregation of all the non-linearity from the float neurons, as described bellow:

**Theorem 1.** *Let $N$ be a neural network defined as Section 3. Given $R \subset X$, for any $x, x' \in R$ with activation pattern $\mathcal{A}$ and $\mathcal{A}'$, we have :*

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

*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 second statement of above theorem suggests that the difference between $f(x)$ and $f(x')$ of network $N$ can be decomposed into: (1) a linear part that depends on distance between $x$ and $x'$ as well as the slope at $x$, and (2) a non-linear part that resulted from instability of float neuron. According to the prime contribution of prediction difference, we can categorize the unsecured data into *Lipschitz vulnerable* and *float neuron vulnerable*. In the following section, we discuss the effect of smoothed classifier on the two types of data and introduce the SC-RFP based on our discussion.

### 4.3 Float Path and Network Robustness

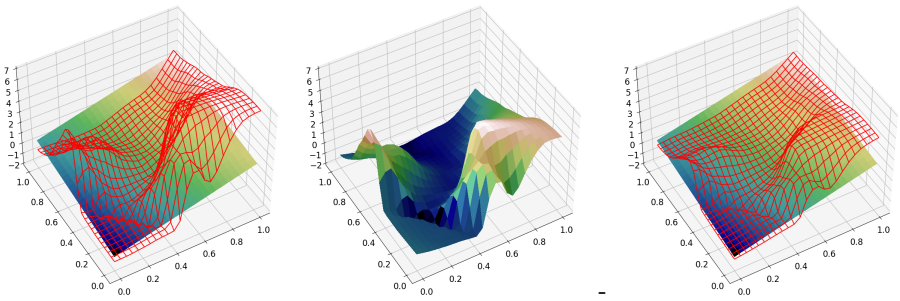
Consider $(x, y) \sim D$ has extreme large Lipschitz constant in $B(x, r)$, then there exists $x' \in B(x, r)$ such that $f(x') - f(x) \gg f_y(x) - f_{m \neq y}(x)$. We refer such an $x$ as a Lipschitz vulnerable data. For example, the Lipschitz constant of a large scale network without any defenses up to hundreds, which means small perturbation $\epsilon$ can alter the prediction of $f(x + \epsilon)$. By applying the randomized smoothing on Theorem 1, we show that the smoothed classifier fails to correct such data.

**Theorem 2.** *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* $g(x') \neq y$.

On the other hand, consider $x' \in B(x, r)$ as an adversarial example that misclassified by $f$ but corrected by $g$, then given $\epsilon \sim D_{noise}$, $g_y(x' + \epsilon) > \arg\max_{m \in Y} g_m(x' + \epsilon)$. This means that the adversarial example $x'$ appears *by change*, while the majority of its neighbor are still voting for the correct label. In other word, the instability of float path causes the misclassification, we refer such data as *float neuron vulnerable*.



(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 sum of fixed path same as (a).

The above discussion suggests that smoothed classifier fails to boost the performance of Lipschitz vulnerable example, but is able to correct float neuron vulnerable data by smoothing the sudden change of float path in a region. Based on this, we introduce the SC-RFP in Algorithm 1.

Figure 1 illustrates the SC-RFP by showing the prediction, fixed path and float path. We first present the prediction and the fixed path value in Figure 1(a). The float path value is then computed as

$Z^T(x, \mathcal{A}; \mathcal{R}) = f(x) - Z^I(x, \mathcal{A}; \mathcal{R})$ in Figure 1(b). At last, by repressing the float value, we achieve a locally stable prediction from 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 $\Gamma$, and use a repression factor $\eta$ to reduce the value of float path.

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

**while** $g$ sample noise $\epsilon$ **do**
  **for** Block $i$ in Block $1, 2, \ldots, d-1$ **do**
    $\mathcal{I}_i^T \leftarrow \mathcal{A}(z_i(x)) \neq \mathcal{A}(z_i(x+\epsilon))$
    $z_i(x+\epsilon) \leftarrow z_i(x+\epsilon) - z_i(x+\epsilon) \times \mathcal{I}_i^T \times \eta$
    $x_{i+1}(x+\epsilon) = \sigma(z_i(x+\epsilon))$
  $counts \leftarrow \phi_d(x_d(x+\epsilon))$
$\hat{c}_A, \hat{c}_B \leftarrow$ top two indices in $counts$
$\hat{n}_A, \hat{n}_B \leftarrow counts[\hat{c}_A], counts[\hat{c}_B]$
**if** $BinomPValue(n_A, n_A + n_B, 0.5) \leq \alpha$ **then** return $\hat{c}_A$ **else** return Abstain

Intuitively, manipulating the computational graph can greatly change the prediction result, while we show that there are only a small proportion of path are affected by our method. 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%.

(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 scales: 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 $\epsilon \sim N(0, 0.25)$.

At the end of this section, we present Theorem 3 to link the certifiable boundary of proposed algorithm with previous works.

**Theorem 3.** *Let $N$ be a network defined as Section 3.1. Let $g$ be a smoothed classifier that samples noise from distribution $D_{noise}$ and $g'$ is the SC-RFP built on $g$. Assume that the direction of $\epsilon$ is uniformly distributed.*

$$\forall \|\eta_1\| = \|\eta_2\| = 1, P(\frac{\epsilon}{\|\epsilon\|} = \eta_1) = P(\frac{\epsilon}{\|\epsilon\|} = \eta_2), \epsilon \sim D_{noise} \tag{13}$$

*If* $\arg\max_{m \in Y} f_y(x) = y$*, then*

$$\underline{p'_A} > \underline{p_A}, \overline{p_B}' < \overline{p_B}, \tag{14}$$

*where* $\underline{p'_A}, \underline{p_A}$ *are the lower bound of* $g'_y(x)$ *and* $g_y(x)$*,* $\overline{p_B}', \overline{p_B}$ *are the upper bound of* $g'_{m\neq y}(x)$*,* $g'_{m\neq y}(x)$*. Moreover,* $\arg\max_{m\in Y} g'(x) = y$ *for all* $\|\epsilon\| \leq R$*,*

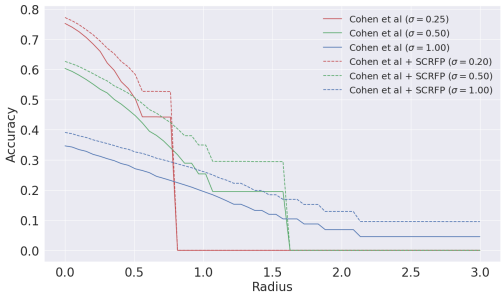
$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \tag{15}$$

Notice that in the above theorem, we do not specify the randomized classifier $g$ but introduce a constraint of the randomness of $g$. This means that the theorem holds for not only the naive smoothed classifier, but also its variants. However, it does not hold for the classifiers that sample directed noise.

The proof of Theorem 3 can be divided into three steps. First, we show that given $\epsilon \sim D_{noise}$, the expectation of fixed paths for the any $\mathbb{E}(Z^X(x+\epsilon)) = Z^X(x)$. This means that the fixed part of $g$ as well as $g'$ equals to that of $f(x)$. Next, we discuss how the float paths affects the prediction $f(x+\epsilon)$ and prove that repressing the float path can increase the expectation of $g'_y(x)$ and its variance. With Chebyshev's Inequality, we conclude the lower bound $p'_A$ is larger than that of $p_A$, and similar to $p'_B$. Since both $g$ and $g'$ are random function that sample noise from same distribution, the certifiable radius holds for both $g$ and $g'$.

## 5 EXPERIMENTS

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

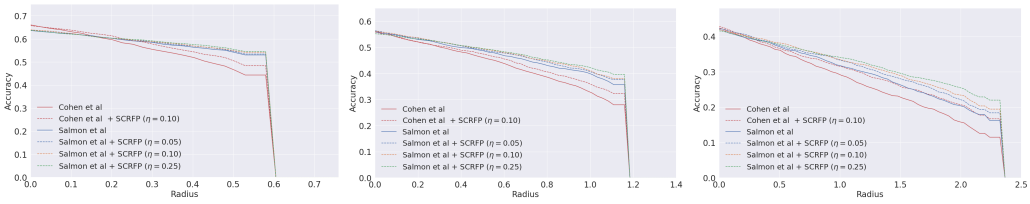
Figure 3: Certified Accuracy of base methods and SC-RFP ($\eta = 0.1$) with different level of noise on CIFAR10. The solid and dashed lines represent benchmark and SC-RFP.

**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 sched-




(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) $\sigma = 1.0$. The solid and dashed lines represent benchmark and SC-RFP with different repression factor.

uler and optimizers as the original work. Each of the model are certified with same level of noise as they were trained.

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

Mislav Balunović and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *8th International Conference on Learning Representations (ICLR 2020)(virtual)*. International Conference on Learning Representations, 2020.

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.

Alessandro De Palma, Rudy Bunel, Krishnamurthy Dvijotham, M Pawan Kumar, and Robert Stanforth. Ibp regularization for verified adversarial robustness via branch-and-bound. *arXiv e-prints*, pp. arXiv–2206, 2022.

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.

Matt Jordan, Justin Lewis, and Alexandros G Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. *Advances in neural information processing systems*, 32, 2019.

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.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

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.

Kai Y Xiao, Vincent Tjeng, Nur Muhammad Mahi Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing relu stability. In *International Conference on Learning Representations*, 2018.

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.

Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of relu networks. In *International Conference on Machine Learning*, pp. 26591–26604. PMLR, 2022.

# A Geometric Illustration

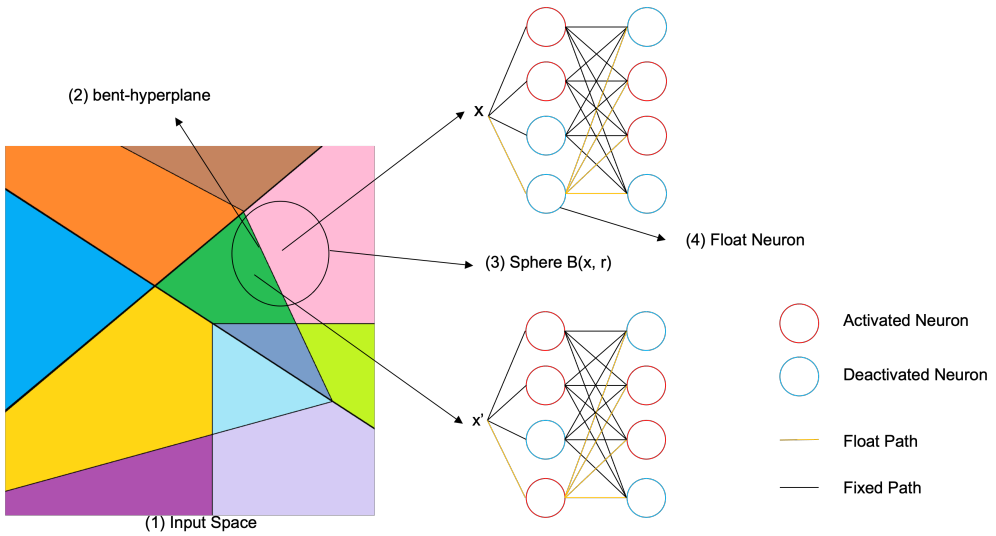



Figure 5: An illustration of the fixed (float) path and neuron of a neural network with 2D input, 4D output, and 1 hidden layer. (1) The 2D input space; (2) A bent-hyperplane defined by $H = x|z_{1,4}(x) = 0$; (3) A sphere centered at $x$; (4) A float neuron with index (1,4) in the network.

Figure 5 presents an illustration of the key concepts of this work. Consider $N$ as a neural network with 2D input, 4D output and 1 hidden layer with 4 neurons. Assume that $N$ has ReLU activation function. The input space is partitioned into several regions by a set of hyperplanes. Each of the region is referred as an activation region that marked in a unique color, within which the mapping function is linear.

For example, the pink and green region are divided by the bent-hpyerplane $H = x|z_{1,4}(x) = 0$, where $z_{1,4}$ is the pre-activation value of 4-th neuron in the 1st layer. Now we consider a sphere $B(x, r)$ centered at $x$ with radius $r$. $B(x, r)$ is covered by the union of pink and green region. Since the two regions can be merged into one by removing the hyperplane, then the incomplete activation pattern of the merged region is $\mathcal{I}\backslash\{(1, 4)\}$. Therefore, $(1, 4)$ is the only float neuron in $B(x, r)$. This means that for every $x \in B(x, r)$, all the neuron has same activation pattern expect for the neuron $(1, 4)$.

Now we consider the computational path of $x \in B(x, r)$. Given a path $\zeta$, if neuron $(1, 4)$ is not on this path, then $\zeta$ is a fixed path, which are the black lines in the figure. This means that for every $x \in B(x, r)$, the value of those fixed path are linear function with respect to $x$. On the other hand, all the non-linearity of function $f(x)$ for $x \in B(x, r)$ are contributed by the float path (Orange lines).

With the illustration above, we proof Lemma 1 as follows.

**Lemma 1** (Restated)**.** *Let $N$ be a neural network defined as Section 3.1. Given $R \subset R^{n_0}$, denote $\mathcal{I}^X$ and $\mathcal{I}^T$ are the set of float neuron and fixed neuron in $R$. Then:*

*1.* $\mathcal{I}^X(R) \bigcup \mathcal{I}^T(R) = \mathcal{I}$ *2.* $R \subset \mathcal{R}(\mathcal{A}_{\mathcal{I}^X})$

**Proof.** *Given a neuron $(i, j)$ and a region $R \in R^{n_0}$. For any $x \in R$, we use $\hat{a}_{ij}(x)$. Since $z_{ij}(x) \in (-\infty, \infty)$, the pattern of $x$ is either an index $k$ of the region when $z_{ij}(x) \in U_k$ or -1 when $z_{ij}(x)$ locate on the bent-hyperplane $H_{ijk} := \{x | z_{ij}(x) = k\}$.*

*This means every neuron has a pattern for $x \in R$. If for every $x \in R^{n_0}$, the pattern of neuron $(i, j)$ remains the same, then $(i, j)$ is a fixed neuron. Otherwise, it is a float neuron. In other words, a neuron is either fixed or float in $\mathbb{R}$. This implies that $\mathcal{I}^X(R) \bigcup \mathcal{I}^T(R) = \mathcal{I}$.*

*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, $x_1 \in \mathcal{R}(\mathcal{A}^X)$.*

## B Decomposing the Network

**Lemma 2.** *(Restate) Let $N$ be a neural network defined as Section 3.1. 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:*

$$y_{ik}(x) = \sum_{\forall \zeta_i = k} x_{j,\zeta_j} \prod_{m=j+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} + \sum_{o=j}^{i-1} \sum_{\forall \zeta_i = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} \quad (16)$$

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

**Proof** (Lemma 2)**.** *We first show that the pre-activation transformation can be represented by a matrix. As $\phi_i$ 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:*

$$z_i = \gamma_i \frac{\phi_i(x) - \hat{\mu}}{\hat{\sigma}} + \beta_i = \frac{\gamma}{\hat{\sigma}} \phi_i(x) - \frac{\gamma \hat{\mu}}{\hat{\sigma}} + \beta_i = D_i(\frac{\gamma}{\hat{\sigma}}) W_i x - \frac{\gamma \hat{\mu}}{\hat{\sigma}} + \beta_i \quad (17)$$

*where $D_i(\cdot)$ is the diagonal operator, $\hat{\mu}$ and $\hat{\sigma}$ are the mean and variance parameter of the $\psi_i$. 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 11 holds.*

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

$$\begin{aligned}
z_{pi} &= \sum_{j=0}^{n_i} W'^{(i)} x_{ij} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} z_{i-1,j} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} d_{\mathcal{A}_{i,j}} z_{i-1,j} + n_i \times \beta'^{(i)}_j \\
&= \sum_{j=0}^{n_i} W'^{(i)} d_{\mathcal{A}_{i,j}} ( \sum_{\forall \zeta_i = j} x_{j,\zeta_j} \prod_{m=p}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} \\
&+ \sum_{o=p-1}^{i-1} \sum_{\forall \zeta_i = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})}) + \beta'^{(i)}_j \\
&= \sum_{\forall \zeta_p = k} x_{j,\zeta_j} \prod_{m=j+1}^{p} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})} + \sum_{o=j}^{p-1} \sum_{\forall \zeta_p = k} \beta^{(o)}_{\zeta_o} \prod_{m=o+1}^{i} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})}
\end{aligned} \quad (18)$$

Lemma 2 decomposes the computational graph of network $N$, while Theorem 1 categorize the paths of $f(x)$ into fixed part and float part. Moreover, it describes the model robustness with by divide the variant of $f$ into a linear part as well as a non-linear part.

**Theorem 1** (Restate)**.** *Let $N$ be a neural network defined as 3. Given $R \subset X$, for any $x, x' \in R$ with activation pattern $\mathcal{A}$ and $\mathcal{A}'$, we have :*

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

*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 $x$.*

Before we proof above Theorem, we use the following Lemma to show that every fixed path in a region is linear regardless of the activation pattern.

**Lemma 3.** *Let $N$ be a network defined as Section 3.3. Let $\zeta$ be a fixed path in $R \subset R_{n_0}$. Then for any $x, x' \in R$ with activation pattern $\mathcal{A}, \mathcal{A}'$, $\zeta(x', \mathcal{A}) = \zeta(x', \mathcal{A}')$. Moreover, $\mathcal{Z}^I(x', \mathcal{A}; \mathcal{R}) = \mathcal{Z}^I(x', \mathcal{A}'; \mathcal{R})$.*

**Proof** (Lemma 3)**.** *Given $x \in R$ with activation pattern $A$,*

$$\zeta(x, \mathcal{A}) := v \prod_{m=1}^{d} d_{\mathcal{A}_{m,\zeta_m}} W'^{(m)}_{(\zeta_m, \zeta_{m-1})}.$$

*Since $\zeta$ is a fixed path in $R$, then every neuron alone $\zeta$ is fixed neuron, the activation pattern of $(m, \zeta_m)$ is same regardless of input $x$. Therefore, $d_{\mathcal{A}_{m,\zeta_m}}$ is constant for any $x \in R$. Then $\zeta(x, \mathcal{A})$ is a linear function of $x : \zeta(x, \mathcal{A}) = \zeta(x)$. In other words, the change of activation pattern does not affect the neurons on $\zeta$, therefore the slope of this path does not change. $\zeta(x)$ is dependent on $x$ in region $R$. This means that $\zeta(x', \mathcal{A}) = \zeta(x', \mathcal{A}')$ for any $x, x' \in R$ with activation pattern $\mathcal{A}, \mathcal{A}'$*

*$\mathcal{Z}^I(x', \mathcal{A}; \mathcal{R})$ is the aggregation of all the fixed path above. Since the summation of linear function is still linear, we have:*

$$\mathcal{Z}^I(x', \mathcal{A}; \mathcal{R}) = \mathcal{Z}^I(x', \mathcal{A}'; \mathcal{R})$$

**Proof** (Theorem 1)**.** *From Lemma 1, every neuron is either fixed neuron or float neuron. For the neurons in path $\zeta$, if there exists a float neuron, then the path is float path. Otherwise, it is a fixed path. The float path and fixed path are complementary set on the set of all paths.*

*Lemma 2 decomposes the $f(x)$ into summation of paths. As each path is either float or fixed, we have*

$$f(x) = \mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) + \mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}).$$

*For statement 2, we have:*

$$\begin{aligned} f(x) - f(x') &= \big(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) + \mathcal{Z}^T(x, \mathcal{A}; \mathcal{R})\big) - \big((\mathcal{Z}^I(x', \mathcal{A}'; \mathcal{R}) + \mathcal{Z}^T(x', \mathcal{A}'; \mathcal{R})\big) \\ &= \big(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) + \mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}) - \mathcal{Z}^I(x', \mathcal{A}; \mathcal{R}) - \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R})\big) + \\ &\quad \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}) - \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}) \end{aligned} \tag{19}$$

*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.*

# C Model Robustness

## C.1 Lipschitz vulnerable and float neuron vulnerable

In Section 4.3 we discuss two kinds of vulnerable data and discuss our motivation of SC-RFP. In this section, we present formally statements of the proposed idea.

Before we move to Theorem 2, we need the following two lemmas. Lemma 5 suggests that, given a fixed path $\zeta$, 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 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 $\gamma$:*

$$\zeta_{(i,j)} = \{\zeta_j, \zeta_{j+1} \ldots, \zeta_i\} \subseteq \zeta \tag{20}$$

*We use $\zeta_{(i,j)}(x)$ to represent the computation graph from $(j, \zeta_j)$ to $(i, \zeta_i)$:*

$$\zeta_i(x) = \zeta_{(i,j)} \circ \zeta_{j,0}(x) \tag{21}$$

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

$$g(\zeta(x)) = \zeta_{d,i}(\zeta_{i,0}(x) + \epsilon), i \in \{0, 1, \ldots, d-1\}, \epsilon \sim D_{noise} \tag{22}$$

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

$$\forall \|\eta_1\| = \|\eta_2\| = 1, P(\frac{\epsilon}{\|\epsilon\|} = \eta_1) = P(\frac{\epsilon}{\|\epsilon\|} = \eta_2) \Rightarrow E[g(\zeta(x))] = \zeta(x), \tag{23}$$

*where $\epsilon$ 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:*

$$\sum_{i}^{n_k} \frac{\partial^2 y_j}{\partial z_{kj}^2} = 0. \tag{24}$$

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

$$\zeta_{d,k}(x) = \frac{1}{n\omega_n r^{n-1}} \int_{\partial B(x,r)} \zeta d\sigma, \tag{25}$$

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

$$\begin{aligned} \mathbb{E}(g(x)) &= \int p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu \\ &= \int \int_{\|\epsilon\|=r} p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu dr, \end{aligned} \tag{26}$$

*where $\mu$ is the probability measure of $\epsilon$ $D_{noise}$. 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:*

$$\int_{\|\epsilon\|=r} p(\epsilon)\zeta_{d,k} d\mu = P(\|\epsilon\|_2 = r)\zeta_{d,k}. \tag{27}$$

*Equation 26 then equals:*

$$\begin{aligned} \mathbb{E}(g(x)) &= \int p(\epsilon)\zeta_{d,k}(x_k + \epsilon) d\mu \\ &= \int P(\|\epsilon\|_2 = r)\zeta_{d,k} dr \\ &= \zeta_{d,k}(x) \end{aligned} \tag{28}$$

**Lemma 5.** *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 5 is the achieved by directly applying Lemma 4 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.

Next, we present the proof of Theorem 2.

**Theorem 2.** *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 $g(x') \neq y$.*

**Proof.** *Statement 4 of 1 suggests:*

$$f(x) - f(x') = J(x)(x - x') + \mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}) - \mathcal{Z}^T(x', \mathcal{A}'; \mathcal{R}) \tag{29}$$

*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:*

$$f_i(x') > f_y(x') + \mathbb{E}[Z_y^T(x, \mathcal{A}; \mathcal{R}) - Z_i^T(x, \mathcal{A}; \mathcal{R})] \tag{30}$$

*Therefore,*

$$\begin{aligned} g_i(x) - g_i(x') &> g_y(x) - g_y(x') + M(f(x), y), \\ g_i(x') &> g_y \end{aligned} \tag{31}$$

Next, we consider the case that $x$ is an adversarial example that mislead $f$ but correctly classified by $g$, which we referred to as a float neuron vulnerable example. The following theorem suggests that, the float path at $x$ are the cause of the altered prediction. In other words, the network is locally correct around $x$, while there are sudden change from the float path that causes the misclassification.

**Theorem 4.** *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:*

$$\arg\max_{m \in Y} f_m(x) = i, \quad \arg\max_{m \in Y} g_m(x) = y, i \neq y \tag{32}$$

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

$$CE(\mathcal{Z}^T(x', \mathcal{A}; \mathcal{R}), onehot(y)) > CE(g(\mathcal{Z}^T(x', \mathcal{A}; \mathcal{R})), onehot(y)) \tag{33}$$

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

**Proof** (Theorem 4)**.** *Since $\arg\max_{m \in Y} f_m(x) \neq y$, $\arg\max_{m \in Y} g_m(x) = y$, there exist $i \in \{1, \ldots, c\}$:*

$$\begin{aligned} g_y(x) &> f_y(x) \\ g_i(x) &< f_i(x) \end{aligned} \tag{34}$$

*Then we have:*

$$g_y(x) - g_i(x) \geq f_y(x) - f_i(x) \tag{35}$$

*Introducing Statement 4 of Lemma 1 and Lemma 5:*

$$g_i(Z^T(x, \mathcal{A}; \mathcal{R})) - g_y(Z^T(x, \mathcal{A}; \mathcal{R})) < Z_i^T(x, \mathcal{A}; \mathcal{R}) - Z_y^T(x, \mathcal{A}; \mathcal{R}). \tag{36}$$

*Moreover, since the prediction is not abstained.*

$$\mathbb{E}[g_y(x)] - f_y(x) > \mathbb{E}[g_j(x)] > f_j(x), \forall j \neq i, y \tag{37}$$

*This directly leads us to the result.*

Theorem 2 and Theorem 4 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.

### C.2 VERIFIABLE RADIUS

At the end of section 4.3, we propose Theorem 3 to describe the upper and lower bound of SC-RFP as well as the certified radius. We present the proof of Theorem 3 below.

**Theorem 3.** *Let $N$ be a network defined as Section 3.1. Let $g$ be a smoothed classifier that samples noise from distribution $D_{noise}$ and $g'$ is the SC-RFP built on $g$. Assume that the direction of $\epsilon$ is uniformly distributed.*

$$\forall \|\eta_1\| = \|\eta_2\| = 1, P(\frac{\epsilon}{\|\epsilon\|} = \eta_1) = P(\frac{\epsilon}{\|\epsilon\|} = \eta_2), \epsilon \sim D \tag{38}$$

*If $\arg\max_{m \in Y} f_y(x) = y$, then*

$$\underline{p'_A} > \underline{p_A}, \overline{p'_B} < \overline{p_B}, \tag{39}$$

*where $\underline{p'_A}, \underline{p_A}$ are the lower bound of $g'_y(x)$ and $g_y(x)$, $\overline{p'_B}, \overline{p_B}$ are the upper bound of $g'_{m \neq y}(x)$, $g'_{m \neq y}(x)$. Moreover, $\arg\max_{m \in Y} g'(x) = y$ for all $\|\epsilon\| \leq R$,*

$$R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})) \tag{40}$$

**Proof.** *The proof of Theorem 3 can be divided into three steps. First, we show that given $\epsilon \sim D_{noise}$, the expectation of fixed paths for the any $\mathbb{E}(Z^X(x+\epsilon)) = Z^X(x)$. Consider $\zeta$ is a fixed path on a region $R$, then from Lemma 4*

$$E[g(\zeta(x))] = \zeta(x). \tag{41}$$

*Notice that, since the SC-RFP does not affect the sample of noise, then above equation holds for both $g$ and $g'$:*

$$E[g'(\zeta(x))] = \zeta(x). \tag{42}$$

*By applying randomized smoothing on statement 1 of Lemma 1, we have:*

$$\begin{aligned} E(g(x)) &= \mathbb{E}(g(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}))) + \mathbb{E}(g(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))), \\ E(g'(x)) &= \mathbb{E}(g'(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}))) + \mathbb{E}(g'(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))), \end{aligned} \tag{43}$$

*where we use $g'(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))$ to denote applying smoothing algorithm $g$ on deterministic function $\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R})$. Since $\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R})$ is the aggregation of all the fixed path, with Equation 41 and 42, we have :*

$$\mathbb{E}(g'(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}))) = \mathbb{E}(g(\mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}))) = \mathcal{Z}^I(x, \mathcal{A}; \mathcal{R}) \tag{44}$$

*This means that the difference between $E(g(x))$ and $E(g(x))$ is same with that of $g(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))$ and $g'(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))$.*

*Next, we discuss how the float path affect the prediction of $g(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))$ and $g'(\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}))$. Given $\epsilon \sim D_{noise}$, we denote $P_1$ as the event that $f_y(x+\epsilon) > f_y(x)$. Since we have excluded the fixed path, $P_1$ holds means that the float path boosts the probability of $f_y(x+\epsilon)$. However, as $\arg\max m \in Y f_y(x) = y$, repressing the float path does not alter the prediction of $f_y(x')$. On the other hand, if $f_y(x+\epsilon) < f_y(x)$, which we denote as $P_2$. This means that the float path $\mathcal{Z}^T(x, \mathcal{A}; \mathcal{R}) < 0$ and negatively contributes to the prediction. Then repressing the float path means that can increase the probability of $y$.*

*In other words, for any $x + \epsilon$, repressing the float path between $x$ and $x'$ can increase the predicted $f_y(x+\epsilon)$. We assume that $P_1$ and $P_2$ happens with probability $p_1$ and $p_2$, then given a certain number of sampling,*

$$E(g'_y(x)) > E(g'_y(x)), Var(g'_y(x)) < Var(g_y(x))$$

*By applying Chebyshev Inequality, we have $\underline{p'_A} > \underline{p_A}$. Similarly, we also have $\overline{p'_B} < \overline{p_B}$ Therefore, we conclude the lower bound $p'_A$ is larger than that of $p_A$, and similar to $p'_B$.*

*Computing the certified radius of $x$ then is same with that of Cohen et al. (2019). Since both $g$ and $g'$ are random function that sample noise from same distribution, the Neyman-Pearson theorem holds for both $g$ and $g'$. Therefore, the certified raidus remains unchanged.*

## D 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.

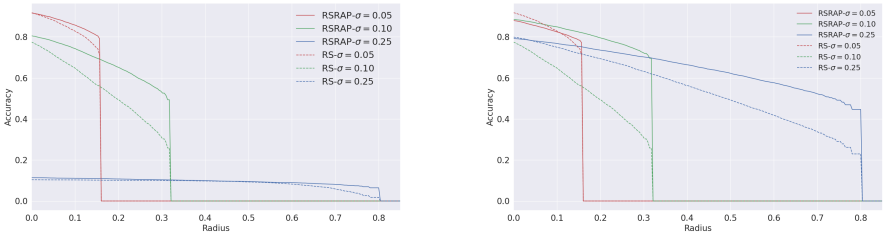
| $\sigma$ | Method | 0.25 | 0.5 | 0.75 | 1.0 | 1.25 | 1.5 | 2.0 | 2.5 | Clean |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.25 | Cohen et al. (2019) | 66.0 | 50.2 | 44.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 75.3 |
| | + SC-RFP | 69.4 | 58.3 | 52.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 77.2 |
| 0.50 | Cohen 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.

| $\sigma$ | $l_2$ 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

Figure 6 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.




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

Figure 6: 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.