FaRS: Fast Randomized Smoothing

Sara Ghazanfari*

Electrical and Computer Engineering Department
New York University

Abstract

In this project, the focus is on Randomized Smoothing [2] paper which proposes certified robustness by leveraging Randomized Smoothing. One of the active topics of research in the area of Trustworthy AI is Adversarial Machine Learning, which first emerged from the [4] paper which shows that the state-of-the-art classifiers with notable natural accuracy are vulnerable to adversarial attacks i.e., small perturbations invisible to the human eyes that fool the classifier from the correct prediction with strong confidence. From that time, a community of researchers has aimed to come up with defenses to provide empirical robustness for Deep Neural Networks. In a parallel line of research, more powerful attacks were proposed to break the robustness of proposed empirical defenses. Trying to escape this cat-and-mouse game, a growing body of work has focused on defenses with formal guarantees. The Randomized Smoothing paper [2] is among the pioneers to provide defense by certified guarantees. Although the method achieves high robust accuracy with an impressive certified bound, the computational complexity due to 100K Monte Carlo sampling for each test sample is very high and makes it less desirable. In this project, we leverage the Lipschitz bound of the 1-Lipschitz networks as the backbone of our model and add the linear layer for the classification, and therefore we show that the Monte Carlo sampling is only required for the Linear layer (and not for the backbone). The code is available at `https://github.com/SaraGhazanfari/fars`

1 Background

In this section, we want to build the background required for the project. First, we'll provide an overview of the adversarial attacks and defense mechanisms. In the next paragraphs, we'll concentrate on the two main approaches proposed to provide certified robustness with mathematical guarantees.

Adversarial Attack & Defense. Initially shown by [4], neural networks are vulnerable to adversarial attacks, i.e., carefully crafted small perturbations that can fool the model into predicting wrong answers. Figure 1 shows an instance of original and perturbed images that could fool the model with high confidence. To mathematically formulate the problem, assume classifier f is trained to predict the correct class y for a pair of input (x, y). The adversarial attack is a tiny perturbation $\|\delta\|_p \leq \epsilon$, invisible to the human eye, which is added to the input to mislead the classifier $f(x + \delta) \neq y$. It's noteworthy that ℓ_p norm is the first proxy that was used to ensure that the perturbation is small and imperceptible to the human eye. Recently more metrics have been used for this purpose and perceptual similarity metrics are one of the serious candidates as they consider the semantics of the images while comparing them.

Since then a large body of research has been focused on providing a defense mechanism against adversarial attacks. Most of these methods concentrate on decreasing the classifier's variance (expressivity), increasing stability, and making the decision boundaries smoother. Simultaneously,

* Correspondence to Sara Ghazanfari: `sg7457@nyu.edu`

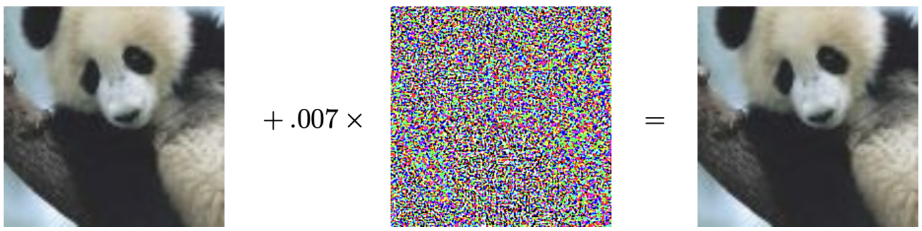


Figure 1: [4] Adversary samples generated using ℓ_∞-PGD attack, with $\epsilon = 0.007$ which fooled the model from predicting panda to predicting the wrong class, gibbon, with very high confidence.

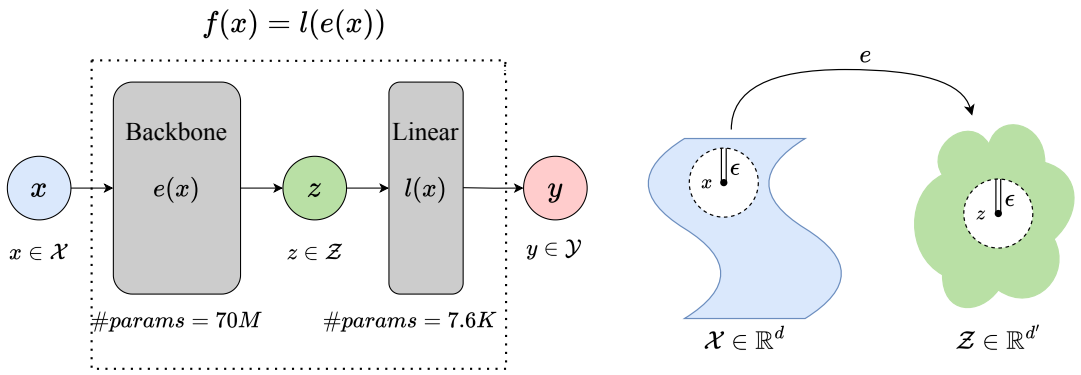
a bunch of other works were trying to generate stronger attacks to break the empirical robustness of defenses. To break this pattern, certified adversarial robustness methods were proposed. By providing mathematical guarantees, the model is theoretically robust against the worst-case attack for perturbations smaller than a specific perturbation budget. Certified defense methods fall into two categories. Randomized Smoothing turns an arbitrary classifier into a smoother classifier, and then based on the Neyman-Pearson lemma, the smooth classifier obtains some theoretical robustness against a specific ℓ_p norm. Despite the impressive results achieved by randomized smoothing in terms of natural and certified accuracy, the high computational cost of inference and the probabilistic nature of the certificate make it difficult to deploy in real-time applications. Another direction of research has been to leverage the Lipschitz property of neural networks to better control the stability of the model and robustness of the model.

Randomized Smoothing A technique proposed by [2] to provide certified robustness in terms of ℓ_2 norm by smoothing the base classifier. Assume f as the vanilla classifier and g as the smoothed version of f assigning the most probable class to the input which is corrupted by a Gaussian noise:

$$g(x) = \arg\max_{c \in \mathcal{Y}} \ \mathbb{P}(f(x + \varepsilon) = c)$$
$$\text{where } \ \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

In other words, g predicts the most probable class for each data point $x + \epsilon$, which is perturbed under the distribution $\mathcal{N}(0, \sigma^2 I)$. Defining the smoothed classifier g, we can prove that smoothed classifier g is certifiably robust to perturbations within radius $R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B}))$ *i.e.*, $g(x + \delta) = g(x)$ for $\|\delta\|_2 \leq R$. Φ^{-1} is the inverse of the standard Gaussian CDF, p_A is probability of the most probable class, and p_B is the probability of the second most probable class.
The randomized smoothing technique has desirable properties; First and foremost, it makes no assumptions about the base classifier's architecture and, therefore the SOTA classifiers can be used as the base classifier. Although the previous statement is true theoretically, in practice, due to the large number of Monte Carlo samples (100,000 as stated in the paper) required for each sample, during inference, this method cannot be applied to recent models with millions to billions of parameters. In this project, we aim to make the inference procedure more efficient by posing one assumption on the backbone of the base classifier.

Lipschitz networks. After the discovery of the vulnerability of neural networks to adversarial attacks, one major direction of research has focused on improving the robustness of neural networks to small input perturbations by leveraging Lipschitz continuity. This goal can be mathematically achieved by using a Lipschitz function. Let f be a Lipschitz function with L_f Lipschitz constant in terms of ℓ_2 norm, then we can bound the output of the function by $\|f(x) - f(x + \delta)\|_2 \leq L_f \|\delta\|_2$. To achieve stability using the Lipschitz property, different approaches have been taken. Initially, some papers tried to calculate the Lipschitz constant of the Neural networks however as calculating the Lipschitz constant of a neural network is computationally expensive, a body of work has focused on designing 1-Lipschitz networks. One efficient way is to design a network with 1-Lipschitz layers which leads to a 1-Lipschitz network. In that case, we have $\|f(x) - f(x + \delta)\|_2 \leq \|\delta\|_2$, which means that ϵ perturbation in the input doesn't cause more than ϵ change in the output.



(a) General architecture of a classifier.

(b) Projection from input space to the embedding space

Figure 2: (a) The general architecture of the classifier is shown in this figure, which is composed of a backbone e that generates the feature embeddings and some Linear layers l that predict the class labels given the feature embeddings. The whole pipeline is showing the classifier f which is $f = l(e(x))$. (b) the projection from input space to the embedding space and given the 1-Lipschitzness of the backbone e, the ϵ perturbation in the input space $\mathcal{X}$ leads to at most ϵ perturbation in embedding space $\mathcal{Z}$. Therefore the certified bound proposed by RS is transferable between spaces.

1-Lipschitz Layers. In order to provide stability by leveraging 1-Lipschitz neural networks different methods have been proposed in the literature. These methods are focused on devising 1-Lipschitz Layers by techniques including spectral normalization or orthogonal parameterization. Let us consider the structure of a simple neural network composed of MLP (Multi-Layer Perception):

$$f(x, \theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(\dots a_1(W^1)\dots)))),$$

Where W^i is the weight of ith layer and a is the activate layers. To have 1-Lipschitz layers we need both the matrix multiplication and activation module to be 1-Lipschitz. The highly-used activations like $ReLU$ are 1-Lipschitz, therefore the concern is focused on the matrix multiplication. In the spectral normalization technique [5] controls the Lipschitz constant of each layer by constraining the spectral norm of each layer i,e. normalizing the spectral norm of the weight matrix W so that it satisfies the Lipschitz constraint $\sigma(W) = 1$:

$$\bar{W}_{\text{SN}}(W) := W/\sigma(W).$$

Although spectral normalization is a simple intuitive method calculating the largest singular value of W in each iteration of training is costly. Later, a body of research replaces the normalized weight matrix with an orthogonal matrix. In the orthogonal-based methods, the 1-Lipschitzness is guaranteed by adding gradient preservation and constraining the parameters by orthogonality during training. In this project, we employ the SDP-base Lipschitz Layers to construct our 1-Lipschitz network which leverages the quadratic constraint approach from control theory to formulate SDPs solving them analytically. the SDP-based Lipschitz Layers (SLL) proposed by [1]:

$$\phi(x) = x - 2W\text{diag}\left(\sum_{j=1}^{n} |W^\top W|_{ij} q_j / q_i\right)^{-1} \sigma(W^\top x + b), \tag{1}$$

where W is a parameter matrix being either dense or a convolution, $\{q_i\}$ forms a diagonal scaling matrix and σ is the ReLU nonlinear activation.

2 Method

In this section, we want to present our contribution to this problem. As stated in the previous section, RS provides good flexibility by not posing any assumption on the base classifier. However, the time complexity of RS due to the Monte Carlo sampling is a concern to the community. On the other hand, another way to provide the certified bound for Neural Networks is to use the Lipschitz property and devise models that are 1-Lipschitz. The 1-Lipschitz networks provide certified robustness deterministically (in contrast with the RS that provides the certified bound with arbitrarily high probability) and the inference is much faster, similar to the inference time of a vanilla Neural Network. In this project, we try to combine the two ideas to have the fast inference time of Lipschitz networks and the flexibility of RS which allows for non-constraint networks to be employed as the base classifier.

2.1 Fast Randomized Smoothing (FaRS)

The general architecture of the classifier is shown in Figure 2a, which is composed of a backbone that generates the feature embeddings and some Linear layers that predict the class labels given the feature embeddings. To formally formulate the problem, we have input space $\mathcal{X} \in \mathbb{R}^d$ and the function $e : x \in \mathcal{X} \to \mathcal{Z}$. Then the embeddings $z \in \mathcal{Z}$ are projected to output space $\mathcal{Y}$ using linear layers $l : z \in \mathcal{Z} \to \mathcal{Y}$. The whole pipeline is showing classifier f which is $f = l(e(x))$. In general, most of the parameters of f correspond to e to generate the embeddings, and quite a small number of parameters correspond to the linear part l (the parameter numbers shown in Figure 2a are specific to the FaRS setting). Therefore limiting the Monte Carlo sampling to the l part of the f classifier can accelerate the sampling process. On the other hand, as the certified bounds should be provided in the input space $\mathcal{X}$, if we aim to do the sampling in the embedding space $\mathcal{Z}$, we need to find the connection between this two spaces so that we can generalize the certified bound in the embedding space to a certified bound in the input space. To achieve this goal, we leverage the 1-Lipschitz property of Neural Networks and prove in the Proposition 1 that given the backbone network e to be 1-Lipschitz, the certified bound found in the embedding space $\mathcal{Z}$ can be applied to the input space $\mathcal{X}$.

Proposition 1. *Defining classifier f to be composed of a backbone module $e : \mathbb{R}^d \to \mathcal{Z}$ and linear layers $l : \mathcal{Z} \to \mathcal{Y}$* i.e. *$f(x) = l \circ e(x)$. Let us assume:*

- *The backbone e is a 1-Lipschitz neural network in ℓ_2 sense: $\|e(x) - e(x + \delta_1)\|_2 \leq \|\delta_1\|_2 = r_1$*
- *Applying Randomized Smoothing on the Linear Layers $\tilde{l}(z) = \arg\max_{c_A \in \mathcal{Y}} \mathbb{P}(l(z + \epsilon) = c_A)$), Then $\tilde{l}(z)$ is certifiably robust for $\|\delta_2\|_2 \leq r_2$ (therefore perturbing z in the r_2 perturbation budget doesn't get the output of $\tilde{l}(z)$).*

The same certified radius as r_2 holds for the f classifier, which means for perturbation $\|\delta\|_1 \leq r_2$ the output of $\tilde{l}(e(x + \delta_1))$ doesn't change: $\tilde{l}(e(x)) = \tilde{l}(e(x + \delta_1))$.

Proof of Proposition 1. For the proof Let's start at the 1-Lipschitz property of e neural network: $\|e(x) - e(x + \delta_1)\|_2 \leq \|\delta_1\|_2 = r_1$. We can substitute the $e(x)$ with z and $e(x + \delta_1) = z + \delta_2$ where δ_2 is a perturbation in $\mathcal{Z}$ space:

$$\begin{aligned} \|e(x) - e(x + \delta_1)\|_2 &\leq r_1 \\ \|z - (z + \delta_2)\|_2 &\leq r_1 \\ \|\delta_2\|_2 &\leq r_1 \end{aligned}$$

From the inequality $\|\delta_2\|_2 \leq r_1$ we conclude that having a perturbation size of $\|\delta_1\|_2$ in the input space means having a perturbation with the same budget in the $\mathcal{Z}$ space. Therefore bounding the perturbation budget in the input space within the same certified budget of the Linear Layers (which is r_2) $r_1 = r_2$ yields having $\|\delta_2\|_2 \leq r_2$ and therefore for perturbation $\|\delta\|_1 \leq r_2$, we have $\tilde{l}(e(x)) = \tilde{l}(e(x + \delta_1))$. □

Table 1: The natural and certified scores of two variants of FaRS. The certified score of **FaRS –** $original$ is slightly better for $\epsilon = \{0.05, 0.1\}$ but **FaRS –** $sparsemax$ has quite better performance for bigger radius $\epsilon = \{0.2, 0.3\}$.

FaRS Version	Natural Score	Certified Score			
		0.05	0.1	0.2	0.3
FaRS – $original$	**80.41**	**78.16**	**53.88**	10.72	0.3
FaRS – $sparsemax$		77.32	50.04	**26.47**	**13.07**

To provide some intuition for Proposition 1, Figure 2b demonstrates the projection from input space to the embedding space and given the 1-Lipschitzness of the backbone e, the ϵ perturbation in the input space $\mathcal{X}$ leads to at most ϵ perturbation in embedding space $\mathcal{Z}$. Therefore the certified bound proposed by RS is transferable between spaces.

2.2 Linear Layer of FaRS

For the Linear part of the FaRS, we can come up with two ideas each of which has its benefits and drawbacks. The first idea is to have exactly one linear layer to map the points from the embedding space to the output space. The second idea is to have multiple layers. Below we'll talk about the two options.

One-Layer Linear Module. This way the RS won't change the l linear layers therefore $\tilde{l}(x) = l(x)$ where $\tilde{l}$ is the smoothed version of l. Using the one-layer linear layer, we have the closed form for the certified bound for the binary classification setting. Let $l(x) = sign(w^T x + b)$ be the linear layer part, and the distance from any input x to the decision boundary is $R = \frac{|w^T x+b|}{\|w\|_2}$, and no perturbation within the range $\|\delta\|_2 \leq R$ can change l's prediction (Please check the proof in Appendix B of [2]). If we want to generalize the linear classifier to the multi-class case, we have to generalize the notion of margin in the multi-class scenario. Multi-class margin is the score multi-class classification difference between the highest-scoring label and the second one. The scores are calculated the same as for the binary case: $\frac{|w^T x+b|}{\|w\|_2}$.

By employing the Lipschitz network the stability of the network is guaranteed given the fact that a small perturbation in the input doesn't lead to big changes in the output. However, to have big certified bounds we need a way to extend the margin between the score of the most probable and the runner-up class. To achieve this goal we leverage two techniques. First, As employed in [3], is to use the $sparsemax$ instead before the $argmax$ that forces a greater margin between the scores. The $sparsemax$ is defined as $sparsemax(z) = \arg\min_{p \in \Delta_r^{c-1}} \|p - z\|_2$, where $\Delta_r^{c-1} = \{p \in \mathbb{R}^c | 1^T p = r, p \geq 0\}$. In other words, we want to project the output to the diamond with distance r from the origin ($\sum_{i=1}^{c} p_i = r$), and this way we can provide more margin between the most and second most probable class. In the experiment section, we'll discuss the results from the experiments performed by $sparsemax$ before $argmax$. The second technique (done by [6]) is adding a "margin" to the score of each class except for the ground truth during training and therefore forcing the model to generate smaller scores for the classes other than the groundtruth.

Multi-Layer Linear Module. In this case, we can devise our Linear module by multiple linear layers and non-linear activations in between. Therefore the closed forms can not be used for the certified bound and the Monte Carlo sampling needs to be done. Looking closer to both ideas, having a one-layer linear that has the close form for the certified bound relaxes the need for multiple times of Monte Carlo sampling, however, the expressivity of one layer is less than the case of having a stack of multiple linear layers. In this project, we'll do the experiments only for the one-layer linear module, and the experiments for the multi-layer case are deferred to future work.

After discussing different options for module l of the architecture, it's noteworthy that the linear layers are applied to the embedding space. As a result, they are more powerful than using the linear layers in the first place.

Table 2: The natural and robust scores of two versions of FaRS. The robust scores are evaluated on the adversary images generated using ℓ_2-PGD attack.

FaRS Version	**Natural Score**	ℓ_2**-PGD**			
		0.05	0.1	0.2	0.3
FaRS – $original$	80.41	79.86	70.22	65.93	59.03
FaRS – $sparsemax$		77.79	68.74	64.57	61.02

3 Experiments

For the experiments, we've selected the ImageNet-10 dataset which has 9469 and 3925 train and val samples respectively. For the backbone e to be 1-Lipschitz, we used the SLL layers proposed by [1]. The SLL architecture comes in different sizes, for the sake of simplicity, the small version which is pre-trained on ImageNet-1k is employed. For the Linear Layer l, the vanilla MLP is used and is trained on the ImageNet-10 dataset while freezing the backbone. In this section, we want to present the certified and empirical robustness of FaRS.

Certified Robustness. We've presented the certified robustness of FaRS in Table 1 for both $original$ and $sparsemax$. The certified score of **FaRS –** $original$ is slightly better for $\epsilon = \{0.05, 0.1\}$ but **FaRS –** $sparsemax$ has quite better performance for bigger radius $\epsilon = \{0.2, 0.3\}$.

Empirical Robustness. In order to do a sanity check on the certified scores reported in Table 1, we aim to perform ℓ_2-PGD attack on our model. Based on the definition of certified radius, no perturbations within the certified bound can decrease the performance. The results of the sanity check are reported in Table 2. The sanity check is passed as the empirical scores are better than the certified scores for all radii. The empirical scores are shown to be quite better than the certified scores and this happens because the chosen attack mechanism (in our case ℓ_2-PGD attack) may not be capable of finding all optimum perturbations to fool the model at a specific radius.

Conclusion & Future work

In this project, we aim to propose a Fast version of Randomized Smoothing, FaRS, by decreasing the time of the Monte Carlo sampling performed to estimate the probabilities and calculate the certified bound. The number of samplings cannot be reduced due to the confidence level required for the estimations. However, by posing the Lipschitz assumption on the model, we could limit the Monte Carlo sampling to the last linear layers performed on the features (in the feature space) to predict classes. More precisely FaRS is composed of a 1-Lipschitz backbone and performs Monte Carlo sampling only on the feature space. In the current project, we focused on a model with a one-layer linear module, which cancels the need for the Monte Carlo sampling. However, in general for the Linear Module to have more expressivity we need more than one linear layer, and the Monte Carlo sampling is needed to generate the certified radius. In order to have bigger margins for each data point we used two techniques that showed its effectiveness in the experiments.

For future work, the multi-layer linear module could be tried by the Monte Carlo sampling to compare the certified accuracies with the results in the single-layer setting of the paper. Another interesting path would be to try the 1-Lipschitz backbone for other downstream tasks like segmentation and apply randomized smoothing on the added layers to the backbone and provide the certified accuracies. Finally, the 1-Lipschitz constraint could be relaxed to L-Lipschitz and try to find the connection between Lipschitzness and the Randomized smoothing technique in general.

References

[1] Alexandre Araujo, Aaron J Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. A unified algebraic perspective on lipschitz neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[2] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[3] Blaise Delattre, Alexandre Araujo, Quentin Barthélemy, and Alexandre Allauzen. The lipschitz-variance-margin tradeoff for enhanced randomized smoothing. *arXiv preprint arXiv:2309.16883*, 2023.

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

[5] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[6] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in neural information processing systems*, 31, 2018.