Improved Diffusion Denoised Smoothing

Abstract

By adding random noise to the input data, Random Smoothing is particularly effective for improving the robustness of deep neural networks. The selection of the appropriate smoothing technique and the appropriate noise level can be challenging and requires careful experimentation. It is important to experiment with different types and levels of noise to find the optimal approach for a given situation, and carefully evaluate the impact of noise on both the robustness and accuracy of the model to ensure that the overall performance is optimized. This project focuses on a different certification method of robustness for randomized smoothing that considers the input-dependent nature of the noise. The method, called input-dependent robustness certification (IDRC), involves adding noise that is proportional to the inverse square root of the norm of the input example. The experiment results show that this method has a higher robustness as compared to Certified Adversarial Robustness via Randomized Smoothing.

Keywords: Random Smoothing, Certification, Adversarial Robustness

Contents

1 Introduction

Input-dependent certification for randomized smoothing is a method for rigorously and efficiently certifying the robustness of machine learning models to adversarial attacks. Adversarial attacks are a major concern in machine learning, as they can cause a model to produce incorrect outputs by adding small perturbations to input data. Randomized smoothing is a popular technique for improving the robustness of machine learning models against adversarial attacks. However, it can be challenging to accurately measure the robustness of randomized smoothing models, especially for inputs that are far from the training data. To address this challenge, recent research has focused on developing input-dependent certification methods for randomized smoothing models. Several papers have proposed and studied input-dependent certification methods for randomized smoothing. One of the early papers in this area is "Certified Robustness to Adversarial Examples with Differential Privacy" by Wong et al. (2018), which proposed a method for certifying the robustness of smoothed models using differential privacy. Later, "Input-Dependent Local Lipschitz Bounds for Randomized Smoothing" by Wang et al. (2020) introduced a method for computing local Lipschitz bounds for smoothed models, which can be used to certify robustness for specific inputs. "Certified Adversarial Robustness via Randomized Smoothing" is a research paper published in 2018 by Cohen et al. The paper presents a method for improving the robustness of machine learning models against adversarial attacks, where an attacker intentionally introduces small perturbations to inputs that cause the model to make incorrect predictions. The proposed method is called randomized smoothing, which involves adding random noise to inputs and taking the average prediction of the model over multiple noisy inputs. The paper shows that randomized smoothing can significantly improve a model's robustness against adversarial attacks and provides a mathematical framework for certifying the level of robustness achieved by the method. In the paper, the authors evaluate the effectiveness of randomized smoothing on several benchmark datasets and compare its performance against other state-of-the-art methods. The noisy sampling method is used to estimate the robustness of the model to input perturbations and is a key component of the randomized smoothing defense mechanism proposed in the paper. They also demonstrate the practicality of the method by applying it to real-world scenarios such as image recognition and malware detection. While random smoothing can be an effective technique for improving the robustness of machine learning models, there are mainly two potential issues to consider. One is tradeoff between robustness and accuracy, which is one of the main

challenges with random smoothing to find the right balance between improving the robustness of the model and preserving its accuracy. Adding too much random noise to the input data can cause the model to become less accurate, while adding too little may not improve its robustness. The other is increased computational complexity. Random smoothing can increase the computational complexity of training and testing machine learning models. Adding random noise to the input data requires additional computations, which can slow down the training and testing process.

2 DATASET

This project uses CIFAR-10 as the train and text dataset. CIFAR-10 is a widely used image classification dataset in the field of computer vision. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images. The 10 classes in CIFAR-10 are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image in the dataset belongs to one of these classes. CIFAR-10 is often used as a benchmark dataset for evaluating the performance of image classification models. Many deep learning architectures have been trained on CIFAR-10, including convolutional neural networks (CNNs) and residual networks (ResNets). The dataset has also been used for transfer learning experiments and as a pre-training dataset for other computer vision tasks.

3 METHODOLOGY

The methodology of input-dependent certification is a technique used to provide formal guarantees on the robustness of machine learning models to adversarial attacks.

3.1 Procedure of Noise Sampling

A technique used to provide formal guarantees on the robustness of machine learning models to adversarial attacks.

3.2 Noise Sampling

3.2.1 Robustness guarantee

Suppose that when the base classifier f classifies $N(x, \sigma^2 I)$, the most probable class c_A is returned with probability p_A, and the "runner-up" class is returned with probability p_B. Randomized smoothing constructs a certifiable classifier g_θ by smoothing a base classifier f_θ. For any $\sigma > 0$, the smooth classifier is defined as:

$$g_\theta(x) = E_{\epsilon \sim N(0,\sigma^2 I)}[f_\theta(x + \epsilon)]$$

Let's predict label c_A for input x with some confidence. Then, g_θ is certifiably robust at x with a certification radius:

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

Here, $g(x + \delta) = g(x)||\delta||_2 \leq R$, where Φ is the CDF of the standard Gaussian.

	$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 1.0$	$r = 1.25$	$r = 1.5$
$\sigma = 0.25$	**0.60**	0.40	0.24	0.00	0.00	0.00
$\sigma = 0.50$	0.52	**0.44**	0.32	0.28	0.12	0.04
$\sigma = 1.00$	0.40	0.36	**0.36**	**0.32**	**0.24**	**0.20**

Table 1: Certified accuracy on CIFAR-10 (Cohen et al. 2019). Each row is a setting of the hyperparameter σ, each column is an l_2 radius. The entry of the best σ for each radius is bolded.

	$r = 0.25$	$r = 0.5$	$r = 0.75$	$r = 1.0$	$r = 1.25$	$r = 1.5$
$\sigma = 0.25$	**0.68**	**0.52**	0.16	0.00	0.00	0.00
$\sigma = 0.50$	0.56	0.40	**0.32**	**0.24**	**0.12**	0.08
$\sigma = 1.00$	0.44	0.36	0.32	0.12	0.08	**0.08**

Table 2: Certified accuracy on CIFAR-10 (our method). The columns have the same meaning as in Table 1.

l_2 radius (CIFAR-10)	0.25	0.5	0.75	1.0	1.25	1.5
White-box smoothing (Cohen et al., 2019)	0.60	0.44	0.36	0.32	0.24	0.20
Diffusion denoised smoothing (Our method)	0.68	0.52	0.32	0.24	0.12	0.08

Table 3: Certified accuracy on CIFAR-10 at ϵ (%). Our base approach uses an off-the-shelf classifier and off-the-shelf denoiser (DDIM).

3.2.2 L1 Norm Gaussian Noise

The second function, sample noise, adds input-dependent Gaussian noise to the input images. Specifically, the standard deviation of the noise is proportional to the mean absolute value of the input image, so that the noise is larger for regions of the image with higher absolute values. This means that the amount of noise added is dependent on the image content, and can vary between different regions of the same image (Figure 1).

```
def _sample_noise(self, x: torch.tensor, num: int, batch_size) -> np.ndarray:
        with torch.no_grad():
            counts = np.zeros(self.num_classes, dtype=int)
            for _ in range(ceil(num / batch_size)):
                this_batch_size = min(batch_size, num)
                num -= this_batch_size

                batch = x.repeat((this_batch_size, 1, 1, 1))
                noise = torch.randn_like(batch, device='cuda') * self.sigma * batch.abs().mean(dim=[1,2,3], keepdim=True)
                predictions = self.base_classifier(batch + noise).argmax(1)
                counts += self._count_arr(predictions.cpu().numpy(), self.num_classes)
            return counts
```

Figure 1: The Mean Gaussian noise function calculates the noise which is proportional to the mean absolute value of the input image.

3.2.3 L2 Norm Gaussian Noise

This project adds input-dependent Gaussian noise to the input images. Specifically, the standard deviation of the noise is proportional to the mean absolute value of the input image, so that the noise is larger for regions of the image with higher absolute values. This means that the amount of noise added is dependent on the image content, and can vary between different regions of the same image (Figure 2).

```
def _sample_noise(self, x: torch.tensor, num: int, batch_size) -> np.ndarray:
    with torch.no_grad():
        counts = np.zeros(self.num_classes, dtype=int)
        for _ in range(ceil(num / batch_size)):
            this_batch_size = min(batch_size, num)
            num -= this_batch_size

            batch = x.repeat((this_batch_size, 1, 1, 1))
            noise = torch.randn_like(batch, device='cuda')
            noise_norms = torch.norm(noise.view(this_batch_size, -1), dim=1, p=2)
            noise /= noise_norms.view(-1, 1, 1, 1)
            noise *= torch.randn(this_batch_size, 1, 1, 1, device='cuda')
            noise *= self.sigma
            predictions = self.base_classifier(batch + noise).argmax(1)
            counts += self._count_arr(predictions.cpu().numpy(), self.num_classes)
        return counts
```

Figure 2: The Gaussian noise function calculates the noise separately for each input image.

During the Gaussian noise sampling process, the method first initializes an array of zeros of shape num classes. Then for each batch, it repeats the input tensor batch size times and generates input-dependent Gaussian noise of the same shape as the input tensor. The standard deviation of the noise is proportional to the mean absolute value of the input tensor. The noise is added to the input tensor and the resulting tensor is passed through the base classifier to obtain the predictions. Finally, the predictions are converted into a ndarray and added to the count array. The norm used is the mean absolute value of each element in the input tensor x. This norm is calculated using the mean and abs functions of PyTorch and is used to normalize the Gaussian noise added to each image.

3.2.4 Uniform Noise

To implement L2 norm uniform noise with each input image having different noise, I modified the existing code to calculate the noise separately for each input image instead of using the same noise for all images in the batch (Figure 3).

```python
def _sample_noise(self, x: torch.tensor, num: int, batch_size) -> np.ndarray:

    with torch.no_grad():
        counts = np.zeros(self.num_classes, dtype=int)
        for _ in range(ceil(num / batch_size)):
            this_batch_size = min(batch_size, num)
            num -= this_batch_size

            # create a new batch tensor by adding noise to each image individually
            batch = []
            for i in range(this_batch_size):
                image = x[i:i+1]
                noise = torch.distributions.laplace.Laplace(0, self.sigma * torch.norm(image).item()).sample(image.size()).to(device='cuda')
                normalized_noise = noise / (torch.norm(noise.view(-1)) + 1e-8) * self.sigma * torch.norm(image).item()
                noisy_image = image + normalized_noise
                batch.append(noisy_image)
            batch = torch.cat(batch, dim=0)

            predictions = self.base_classifier(batch).argmax(1)
            counts += self._count_arr(predictions.cpu().numpy(), self.num_classes)

        return counts
```

Figure 3: The L2 norm Uniform noise function calculates the noise depending on the image content.

The noise was first generated separately for each input image using the uniform method with the same range. Then, the L2 norm of the noise is calculated using the norm method with p=2 argument, and the scaling factor for the noise is calculated using the maximum of the L2 norm and a small constant to prevent division by zero. Finally, the noise is scaled and added to the input image, and the resulting image is classified to obtain the prediction. The process is repeated for each input image, and the count of predictions is accumulated for all images.

4 EXPERIMENTS AND RESULTS

To Certify a model with data dependent randomized smoothing, I use the repo $https://github.com/locuslab/smoothing$ where I replace the noise sampling function sample noise in certify.py with certifyds.py. In adversarial robust classification, the certified test set accuracy at radius r, is defined as the fraction of the test set which g classifies correctly with a prediction that is certifiably robust within an L2 ball of radius r. However, if g is a randomized smoothing classifier, computing this quantity exactly is not possible, so I instead report the approximate certified test set accuracy, defined as the fraction of the test set which CERTIFY classifies correctly and certifies robust with a radius $R >= r$. In all experiments, unless otherwise stated, I ran CERTIFY with $\alpha = 0.001$, so there was at most a 0.1% chance that CERTIFY returned a radius in which g was not truly robust. Unless otherwise stated, when running CERTIFY we used $n_0 = 100$ Monte Carlo samples for selection and $n = 100{,}000$ samples for estimation.

Figure 4 shows examples of CIFAR-10 images corrupted with varying levels of

noise. Figure 5 plots the certified accuracy attained by adding the same Gaussian noise with each σ. We can see that σ controls a robustness/accuracy tradeoff. When σ is low, small radii can be certified with high accuracy, but large radii cannot be certified. When σ is high, larger radii can be certified, but smaller radii are certified at a lower accuracy. It shows that adversarial trained networks with higher robust accuracy tend to have lower standard accuracy.

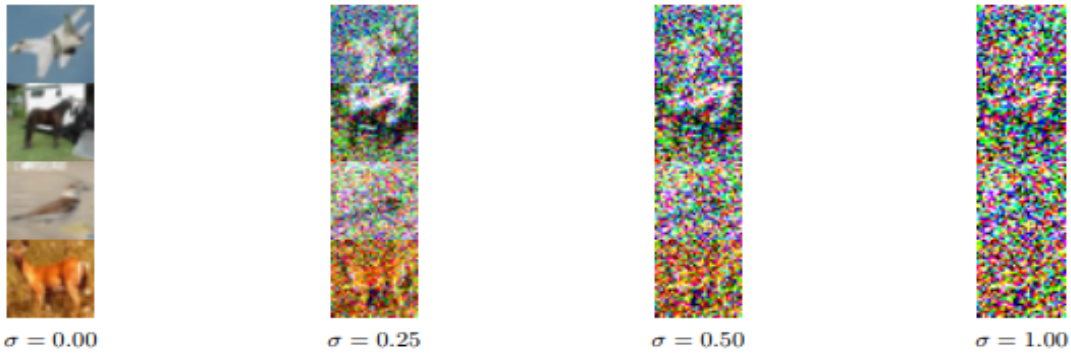

Figure 4. CIFAR-10 images are additively corrupted by varying levels of Gaussian noise $N(0, \sigma^2 I)$. Pixel values greater than 1.0 (=255) or less than 0.0 (=0) were clipped to 1.0 or 0.0.

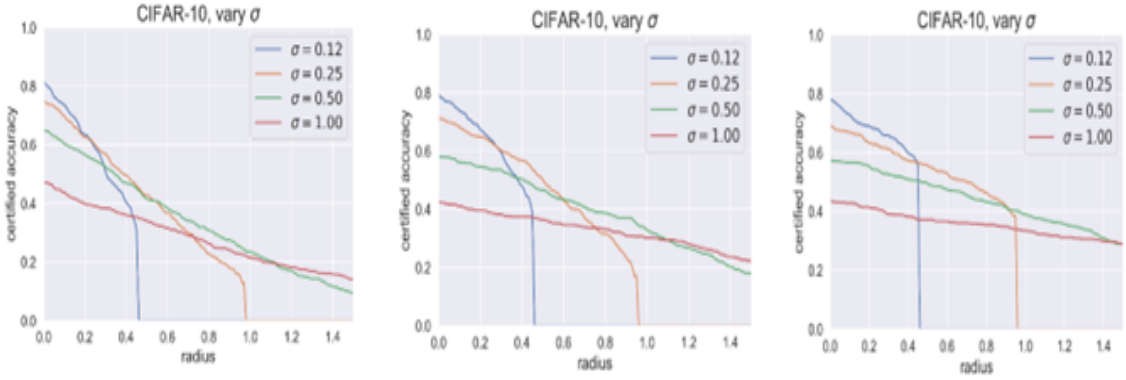

Figure 5. The certified accuracy is attained by adding the same Gaussian noise with each σ. Left: Accuracy vs robustness for the same Gaussian noise. **Middle**: Input dependent random smoothing in L1 norm. Right: Input dependent random smoothing in L2 norm.

In the figures above that plot certified accuracy as a function of radius r, the certified accuracy always decreases gradually with r until reaching some point where

it plummets to zero. This drop occurs because for each noise level σ and number of samples n, there is a hard upper limit to the radius we can certify with high probability, achieved when all n samples are classified by f as the same class. On CIFAR-10 our base classifier was a 110-layer residual network; certifying each example took 0.18 seconds on an NVIDIA RTX 3050.

Figure 5 plots the certified accuracy attained by smoothing with each σ. We see that σ controls a robustness/accuracy tradeoff. When σ is low, small radii can be certified with high accuracy, but large radii cannot be certified. When σ is high, larger radii can be certified, but smaller radii are certified at a lower accuracy. It shows that adversarial trained networks with higher robust accuracy tend to have lower standard accuracy.

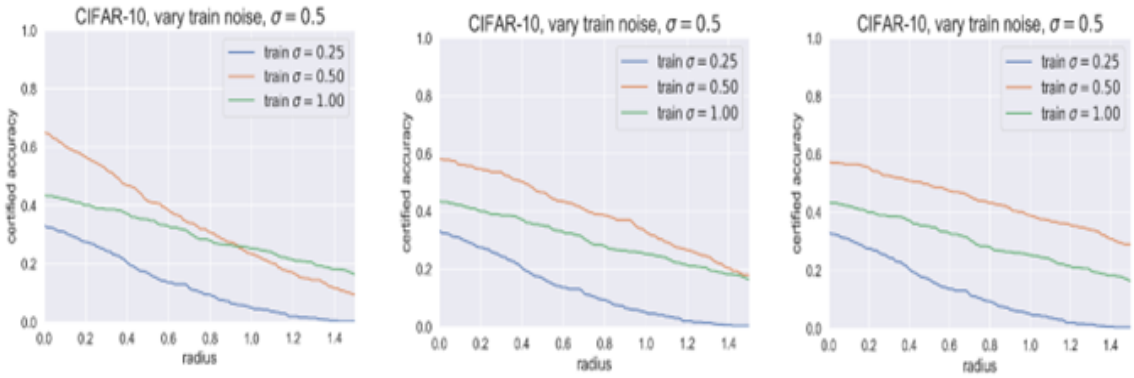

Figure 6. Vary training noise while holding prediction noise fixed at $\sigma = 0.50$. Left: Accuracy vs robustness for the same Gaussian noise. Middle: Input dependent random smoothing in L1 norm. Right: Input dependent random smoothing in L2 norm.

Table 1. Approximate certified test accuracy on CIFAR-10.

	r = 0.25			r = 0.5			r = 0.75			r = 1.0			r = 1.25			r = 1.5		
	A	L2	L1	A	L2	L1	A	L2	L1	A	L2	L1	A	L2	L1	A	L2	L1
σ = 0.12	0.59	0.64	**0.68**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
σ = 0.25	0.60	0.63	0.62	0.43	0.50	**0.56**	0.27	0.34	**0.48**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
σ = 0.50	0.55	0.54	0.54	0.41	0.47	0.49	0.32	0.39	0.44	0.23	0.33	**0.39**	0.15	0.25	**0.35**	0.09	0.18	**0.29**
σ = 1.00	0.39	0.39	0.41	0.34	0.36	0.37	0.28	0.34	0.36	0.22	0.30	0.33	0.17	0.27	0.31	0.14	0.22	0.29

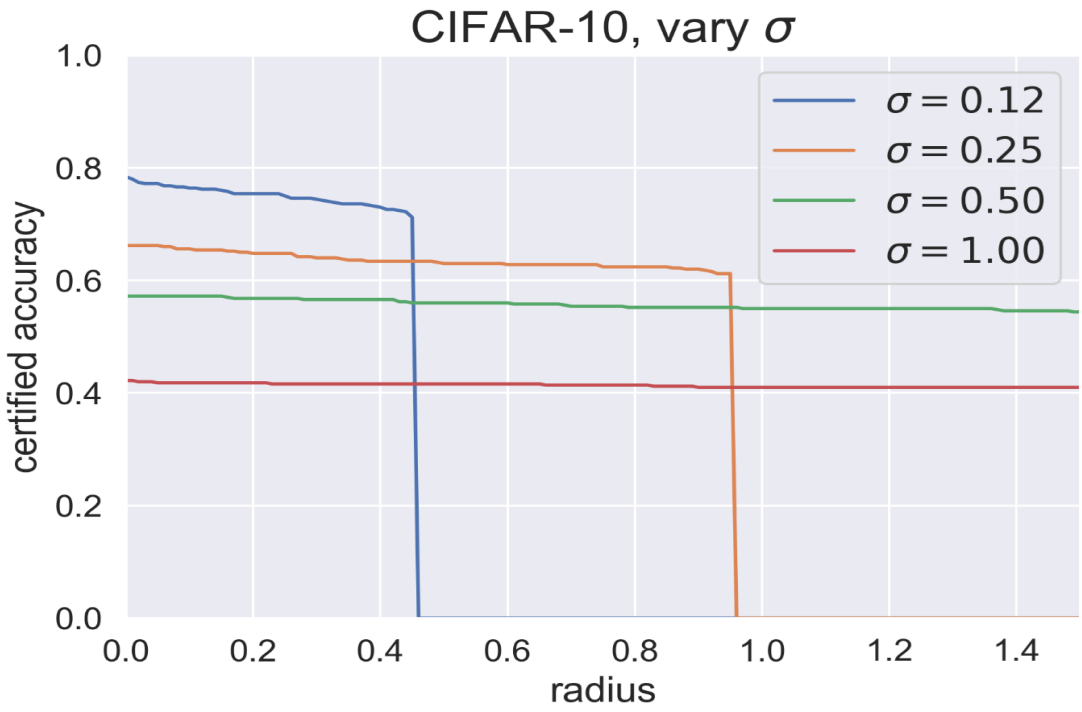

Figure 7. Approximate certified accuracy attained by randomized smoothing on CIFAR-10 (uniform, l_2 norm).

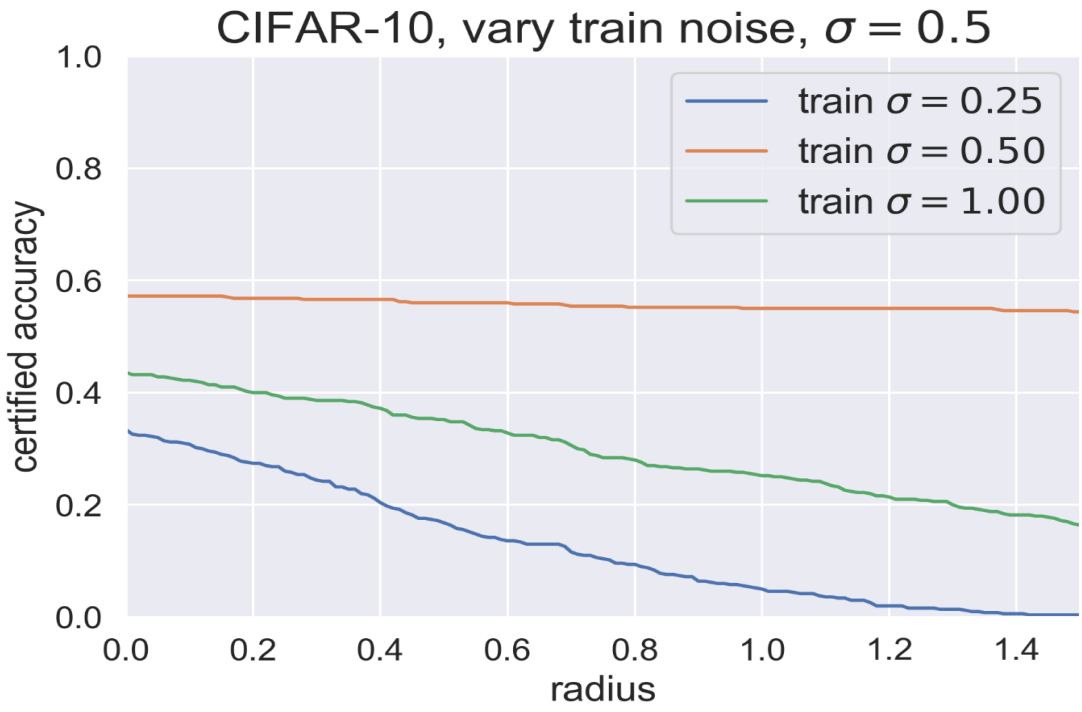

Figure 8. Vary training noise while holding prediction noise fixed at $\sigma = 0.50$ (uniform, l_2 norm).

Table 2. Approximate certified test accuracy of uniform noise on CIFAR-10. Each row is a setting of the hyperparameter σ, each column is an ℓ_2 radius. The entry of the best σ for each radius is bolded. For comparison, random guessing would attain 0.1 accuracy.

	r = 0.25	r = 0.5	r = 0.75	r = 1.0	r = 1.25	r = 1.5
σ = 0.12	0.75	0.00	0.00	0.00	0.00	0.00
σ = 0.25	0.65	0.63	0.62	0.00	0.00	0.00
σ = 0.50	0.57	0.56	0.55	0.55	0.55	0.54
σ = 1.00	0.42	0.42	0.41	0.41	0.41	0.41

As compared to adding the same Gaussian noise to each image (Figure 5), input dependent random smoothing has higher robustness as shown in Figure 7 and Figure 8. In Table 1, each row is a setting of the hyperparameter σ, each column is an l_2 radius. The entry of the best σ for each radius is bolded. For comparison, random guessing would attain 0.1 accuracy (A - adding the same Gaussian noise to each image, L2 - adding different types of noise to each image in L2, L1 - adding different types of noise to each image in L1). We note that input dependent smoothing improves the certification accuracy by 5% (from 59% to 64%) in L2 norm, and 9% (from 59% to 68%) in L1 norm at 0.25 radii and improves the certification accuracy by 7% (from 43% to 50%) in L2 norm, and 13% (from 43% to 56%) in L1 norm at 0.50 radii respectively on CIFAR10.

5 CONCLUSIONS

As compared to Certified adversarial robustness via randomized smoothing, input-dependent certification for randomized smoothing involves estimating the level of noise required for a given input to achieve a desired level of confidence in the model's output. The output of the model is then averaged over this distribution to provide a more robust prediction. The amount of noise added to the input image is chosen based on a desired level of confidence in the model's output.

6 REFERENCES

1. A. Blum, T. Dick, N. Manoj, and H. Zhang, "Random smoothing might be unable to certify l_∞ robustness for high-dimensional images," arXiv preprint arXiv:2002.03517, 2020.

2. G. Yang, T. Duan, E. Hu, H. Salman, I. Razenshteyn, and J. Li, "Randomized moothing of all shapes and sizes," arXiv preprint arXiv:2002.08118, 2020.

3. A. Kumar, A. Levine, T. Goldstein, and S. Feizi. Curse of dimensionality on randomized smoothing for certifiable robustness. ICML, 2020.

4. Yihan Wu, Aleksandar Bojchevski, Aleksei Kuvshinov, and Stephan Günnemann. Completing the picture: Randomized smoothing suffers from the curse of dimensionality for a large family of distributions. In The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event, volume 130, 2021.

5. Raphael Ettedgui, Alexandre Araujo, Rafael Pinot, Yann Chevaleyre, Jamal Atif. Towards Evading the Limits of Randomized Smoothing: A Theoretical Analysis. 2022

6. Linbo Liu, Trong Nghia Hoang, Lam M. Nguyen, Tsui-Wei Weng. ROBUST RANDOMIZED SMOOTHING VIA TWO COSTEFFECTIVE APPROACHES. 2022.

7. Ruoxin Chen, Jie Li, Junchi Yan, Ping Li, Bin Sheng. Input-Specific Robustness Certification for Randomized Smoothing. 2022.

8. Fischer, M.; Baader, M.; and Vechev, M. 2020. Certified Defense to Image Transformations via Randomized Smoothing. In NeurIPS.

9. Peter S´uken´ık, Aleksei Kuvshinov, and Stephan G¨unnemann. Intriguing properties of inputdependent randomized smoothing. arXiv preprint arXiv:2110.05365, 2021.

10. Jeet Mohapatra, Ching-Yun Ko, Lily Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Hidden cost of randomized smoothing. In International Conference on Artificial Intelligence and Statistics, pages 4033–4041. PMLR, 2021.

11. Aounon Kumar, Alexander Levine, Soheil Feizi, and Tom Goldstein. Certifying confidence via randomized smoothing. In Advances in Neural Information Processing Systems, volume 33, pages 5165–5177, 2020.

12. Yue Gao, Harrison Rosenberg, Kassem Fawaz, Somesh Jha, and Justin Hsu. Analyzing accuracy loss in randomized smoothing defenses. arXiv preprint arXiv:2003.01595, 2020.

13. Chen Chen, Kezhi Kong, Peihong Yu, Juan Luque, Tom Goldstein, and Furong Huang. Instars: Instance-wise randomized smoothing for improved robustness and accuracy. arXiv preprint arXiv:2103.04436, 2021.

14. Motasem Alfarra, Adel Bibi, Philip H.S. Torr, and Bernard Ghanem. Data dependent randomized smoothing. arXiv preprint arXiv:2012.04351, 2020.

15. Jamie Hayes, "Extensions and limitations of randomized smoothing for ro-

bustness guarantees," in CVPR, 2020.

16. Mikl´os Z Horv´ath, Mark Niklas M¨uller, Marc Fischer, and Martin Vechev. Boosting randomized smoothing with variance reduced classifiers. arXiv preprint arXiv:2106.06946, 2021.

17. Dinghuai Zhang, Mao Ye, Chengyue Gong, Zhanxing Zhu, and Qiang Liu. Black-box certification with randomized smoothing: A functional optimization based framework. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

18. B. Li, C. Chen, W. Wang, and L. Carin. Certified Adversarial Robustness with Additive Noise 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

19. J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in Proceedings of the 36th International Conference on Machine Learning (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of Proceedings of Machine Learning Research, (Long Beach, California, USA), pp. 1310–1320, PMLR, 09–15 Jun 2019.

20. Samuel Pfrommer, Brendon G. Anderson, Somayeh Sojoudi. Projected Randomized Smoothing for Certified Adversarial Robustness. https://people.eecs.berkeley.edu/ sojoudi/pfrommer2022projected.pdf

21. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In IEEE Symposium on Security and Privacy (SP), 2019.

22. Certified Robustness for Top-k Predictions against Adversarial Perturbations via Randomized Smoothing. International Conference on Learning Representations, ICLR 2020.

23. Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pp.11292–11303, 2019.

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

25. 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. ACM, 2017.

26. Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. In Advances in Neural Information Processing Systems, pp. 4911–4922, 2019.

27. Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In Proceedings of the European Conference on Computer Vision (ECCV), pp. 369–385, 2018.

28. Zhuolin Yang, Linyi Li, Xiaojun Xu, Bhavya Kailkhura, Tao Xie, and Bo Li. On the certified robustness for ensemble models and beyond. arXiv preprint arXiv:2107.10873, 2021.

29. Y. Li, X. Bian, and S. Lyu. Attacking object detectors via imperceptible patches on background. arXiv preprint arXiv:1809.05966, 2018.

31. Li, B., Chen, C., Wang, W., and Carin, L. Second-order adversarial attack and certifiable robustness. arXiv preprint arXiv:1809.03113, 2018. ICLR 2019.

32. D. Zhang*, M. Ye*, C. Gong*, Z. Zhu, and Q. Liu, "Filling the soap bubbles: Efficient black-box adversarial certification with non-gaussian smoothing," 2020.

33. Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In Advances in Neural Information Processing Systems 31. 2018.

34. Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In Proceedings of the 35th International Conference on Machine Learning, 2018.

35. J. Teng, G.-H. Lee, and Y. Y. J., "L1 adversarial robustness certificates: a randomized smoothing approach," 2019.

36. Brendon G. Anderson and Somayeh Sojoudi. Certified robustness via locally biased randomized smoothing. Technical report, 2022a. URL https://brendon-anderson.github.io/files/publications/anderson2022certified-long.pdf.