Lecture 14: Certified Defenses III: Randomized Smoothing

November 13, 2019

In this lecture we present the final of the certified defenses we will consider in this class, namely *randomized smoothing* [1, 2, 3]. We will follow the presentation of [4], which simplifies the proofs in the previous papers.

1 Randomized smoothing, a.k.a. the Weierstrauss transform

Randomized smoothing is a transformation of a classifier. It is most naturally stated for soft classifiers. Recall that a soft classifier is a map $F : \mathbb{R}^d \to \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ is the set of probability distributions over $\mathcal{Y}$. That is, a soft classifier, instead of just assigning each data point a class, assigns each point a distribution over classes, which indicates how likely it thinks each class is. Observe that neural networks are naturally soft classifiers, as the mapping from a data point to the output of the last layer (before we read off the most likely class) is a soft classifier. This will become important later on.

Moreover, note that any hard classifier f has an associated soft classifier $F(x) = e_{f(x)}$, where for any $y \in \mathcal{Y}$, we let e_y be the probability distribution which is 1 at y and zero everywhere else. Conversely, we can transform any soft classifier F into a hard classifier f by the mapping $f(x) = \arg\max_y F(x)_y$, although observe that this transformation is lossy.

Given a soft classifier F, we say that its associated *smooth classifier* with parameter $\sigma > 0$ is the soft classifier

$$G(x) = \left(F * \mathcal{N}(0, \sigma^2 I)\right)(x) = \mathop{\mathbb{E}}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} \left[F(x + \delta)\right] \ . \tag{1}$$

Here $*$ denotes the convolution operator between two functions, defined by:

$$f * g(x) = \int_{\mathbb{R}^d} f(t) g(x - t) dt \ .$$

This is also known in mathematics as the *Weierstrauss transform* of F. The reason why this transformation is interesting to us is because of the following theorem. Recally that Φ is the cumulative distribution function of the standard normal Gaussian.

Theorem 1.1 ([3]). *Let $F : \mathbb{R}^d \to \mathcal{P}(\mathcal{Y})$ be any soft classifier, let $\sigma > 0$, and let G be its associated smooth classifier. Let*

$$a = \arg\max_{y \in \mathcal{Y}} G(x)_y \qquad \text{and} \qquad b = \arg\max_{y \in \mathcal{Y} \setminus \{y^*\}} G(x)_y \tag{2}$$

be the two most likely classes for x according to G. Then, we have that $\arg\max_{y \in \mathcal{Y}} G(x')_y = a$ for all x' satisfying

$$\|x' - x\|_2 \leq \frac{\sigma}{2} \left(\Phi^{-1}(p_a) - \Phi^{-1}(p_b) \right) . \tag{3}$$

In other words, the most likely class for any point x' within the ℓ_2 radius as specified by (3) will have the same most likely class as x. Therefore, if we use G's associated hard classifier, it is provably robust within some ℓ_2 radius, which depends on the gap between p_a and p_b. The larger the gap, the larger the associated radius. In the limit, if $p_a = 1$, then observe that the radius is infinite, which makes sense, as this can only happen if the original soft classifier was deterministically a, outside of a set of measure zero.

1.1 Implementing randomized smoothing

Before we prove Theorem 1.1, let's see how to use implement randomized smoothing in practice. The main difficulty is that if F is a soft classifier coming from e.g. a neural network, it is not obvious how to exactly compute its associated soft classifier G. However, by its very nature, it is very easy to approximate via sampling. In particular, we simply need to estimate a, b, p_a, p_b as defined in Theorem 1.1. To do so, we can simply sample $\delta_1, \ldots, \delta_k \sim \mathcal{N}(0, \sigma^2 I)$, and compute $F(x + \delta_1), \ldots, F(x + \delta_k)$, to compute an empirical estimate of $G(x)$, namely, $\widehat{G}(x) = \frac{1}{k} \sum_{i=1}^{k} F(x + \delta_i)$. We then compute $\widehat{p}_a, \widehat{p}_b$ by taking the two largest entries of $\widehat{G}(x)$, and we can use them as proxies for p_a and p_b, to give us a certified radius.

Note that each coordinate of $\widehat{G}(x)$ is a sum of independent $[0, 1]$-valued random variables and thus $\widehat{G}(x)$ converges very quickly to $G(x)$ in ℓ_∞ norm, by Chernoff-style tail bounds. Therefore if we take $k = 10000$, we can be quite confident that our estimates of p_a, p_b are very close to the truth, and we can be very confident that our guarantees are correct with extremely high probability. In this class, for the sake of time, we'll omit some of the details, see [3, 4] for more details.

1.2 Comparison to previous techniques

Randomized smoothing is a bit different than the previous ones we've covered, for a number of reasons, which we'll discuss later.

- It is probabilistic in nature, meaning that it only provides guarantee on the certifiable robustness with high probability.
- It actually changes the classifier, as opposed to simply certifying the original deep network, and it is this altered classifier which will have certifiable guarantees.

- It depends much less on the structure of the actual classifier, as opposed to previous methods which are heavily dependent on the structure of the network as a feedforward ReLU network. Indeed, as we'll see, we can make any classifier into one with certifiable guarantees.
- It is much more scalable than previous certification algorithms, scaling up to CIFAR-10 and ImageNet, and offers very competitive accuracy numbers as well.
- Finally, as opposed to the previous methods, which are naturally suited for ℓ_∞ perturbations, randomized smoothing is most naturally considered for ℓ_2, although one can consider other norms as well.

2 Proof of Theorem 1.1

We now prove Theorem 1.1. We'll first prove a weaker theorem, and then see how to change this proof to get the tight bounds as in Theorem 1.1.

2.1 A warmup

Recall that a function $h : \mathbb{R}^d \to \mathbb{R}$ is L-Lipschitz in a norm $\|\cdot\|$ if

$$|h(x) - h(x')| \leq L \|x - x'\| \ .$$

If h is differentiable, then h is L-Lipschitz if $\|\nabla h(x)\|_* \leq L$ for all x, where $\|\cdot\|_*$ is the dual norm to $\|\cdot\|$.

We start with the following observation:

Lemma 2.1. *Suppose that $F : \mathbb{R}^d \to \mathcal{P}(\mathcal{Y})$ is a soft classifier, and moreover, the function $x \mapsto F(x)_y$ is L-Lipschitz in a norm $\|\cdot\|$, for each $y \in \mathcal{Y}$. Let a, b, p_a, p_b be as (2). Then, we have that $\arg\max_y F(x') = a$ for all x' so that $\|x' - x\| < \frac{1}{2L}(p_a - p_b)$.*

Proof. By the Lipschitz-ness of the map $x \mapsto F(x)_a$, we know that for any for any x' within this ball, we have

$$|F(x')_a - F(x)_a| = |F(x')_a - p_a| \leq L \|x' - x\| < \frac{1}{2}(p_a - p_b) \ .$$

In particular this implies that $F(x')_a > \frac{1}{2}(p_a + p_b)$. However, for any $y \neq a$, by the same logic,

$$F(x')_y < F(x)_y + \frac{1}{2}(p_a - p_b) \leq \frac{1}{2}(p_a + p_b) < F(x')_a \ ,$$

as desired. □

In fact we can trivially generalize this lemma to allow for certain nonlinearities:

Lemma 2.2. *Let $m : \mathbb{R} \to \mathbb{R}$ be a monotone, invertible function. Suppose that $F : \mathbb{R}^d \to \mathcal{P}(\mathcal{Y})$ is a soft classifier, and moreover, the function $x \mapsto m(F(x)_y)$ is L-Lipschitz in a norm $\|\cdot\|$, for each $y \in \mathcal{Y}$. Let a, b, p_a, p_b be as (2). Then, we have that $\arg\max_y F(x') = a$ for all x' so that $\|x' - x\| < \frac{1}{2L}(m(p_a) - m(p_b))$.*

The proof of this lemma is essentially identical to that of Lemma 2.1 and so we omit it.

These lemmata touch upon a general principle for certified defenses that we actually haven't discussed too much: namely, that classifiers which are Lipschitz will be robust. There have been attempts to directly make neural networks Lipschitz to the input, however, typically these don't work very well. As we'll see, randomized smoothing does something else: rather than trying to make the base neural network Lipschitz, which is hard to do while preserving accuracy, it simply takes any neural network classifier, and postprocesses it to make it Lipschitz.

Before we prove our warm-up lemma, we'll also need the following analytic form of the gradient of a smoothed function, a fact known as *Stein's lemma*:

Lemma 2.3 (Stein's lemma, c.f. [5])**.** *Let $\sigma > 0$, let $h : \mathbb{R}^d \to \mathbb{R}$ be measurable, and let $H = h * \mathcal{N}(0, \sigma^2 I)$. Then H is differentiable, and moreover,*

$$\nabla H(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} h(t) \frac{t - x}{\sigma^2} \exp\left(\frac{1}{2\sigma^2} \|x - t\|_2^2\right) dt = \frac{1}{\sigma^2} \mathop{\mathbb{E}}_{X \sim \mathcal{N}(0, \sigma^2 I)} [X \cdot h(x + X)] \ .$$

Proof. The author does not feel like doing the (standard) real analysis that proves that the function H is differentiable and will therefore skip that argument. But assuming this, then we have:

$$\begin{aligned}
\nabla H(x) &= \nabla \left(\frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} h(t) \exp\left(-\frac{1}{2\sigma^2} \|x - t\|_2^2\right) dt \right) \\
&= \frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} h(t) \nabla \left(\exp\left(-\frac{1}{2\sigma^2} \|x - t\|_2^2\right) \right) dt \\
&= \frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} h(t) \frac{t - x}{\sigma^2} \exp\left(-\frac{1}{2\sigma^2} \|x - t\|_2^2\right) dt \ , \qquad (4)
\end{aligned}$$

as claimed. The final equality follows from the definition of expectation and a change of variables. □

Lemma 2.4. *Let $\sigma > 0$, let $h : \mathbb{R}^d \to [0, 1]$ be measurable, and let $H = h * \mathcal{N}(0, \sigma^2 I)$. Then H is $\sqrt{\frac{2}{\pi\sigma^2}}$-Lipschitz in ℓ_2.*

Proof. As ℓ_2 is self-dual, it suffices to show that the gradients of H are bounded in ℓ_2. Now,

for any unit vector $v \in \mathbb{R}^d$, by Stein's lemma (Lemma 2.3), we have

$$\begin{aligned}
|\langle v, \nabla H(x)\rangle| &= \left| \frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} h(t) \left\langle v, \frac{t-x}{\sigma^2} \right\rangle \exp\left(-\frac{1}{2\sigma^2}\|x-t\|_2^2\right) dt \right| \\
&\leq \frac{1}{(2\pi\sigma^2)^{d/2}} \int_{\mathbb{R}^d} \left| \left\langle v, \frac{t-x}{\sigma^2} \right\rangle \right| \exp\left(-\frac{1}{2\sigma^2}\|x-t\|_2^2\right) dt \qquad (5) \\
&= \frac{1}{\sigma^2} \mathop{\mathbb{E}}_{Z\sim\mathcal{N}(0,\sigma^2)}[|Z|] = \sqrt{\frac{2}{\pi\sigma^2}} \,. \qquad (6)
\end{aligned}$$

Here (5) follows from the triangle inequality and since h is bounded by 1, and (6) follows since projections of Gaussians are Gaussians, and from a classical Gaussian integral. Taking a supremum over all unit vectors v immediately yields the claim. □

Lemmata 2.1 and 2.4 immediately imply the following robustness for a smoothed classifier, but which is weaker than what we'll eventually prove:

Corollary 2.5. *Let F be a smooth classifier with parameter $\sigma > 0$, and let a, b, p_a, p_b be as in (2). Then $\arg\max_y F(x') = a$ for all x' satisfying $\|x' - x\|_2 \leq \sqrt{\frac{1}{2\sigma^2}}(p_a - p_b)$.*

This is essentially the sorts of bounds proved in [1, 2], however, it is weaker than the bound that we'll eventually use. The main difference between the two bounds is that this bound lacks the nonlinearity Φ^{-1}. For instance, if $p_a = 1$, this bound still only certifies a constant radius, whereas Theorem 1.1 is able to certify an infinite radius.

2.2 The main course

We now actually prove Theorem 1.1. The key lemma is the following, stronger Lipschitz property of the Weierstrauss transform:

Lemma 2.6. *Let $\sigma > 0$, let $h : \mathbb{R}^d \to [0, 1]$ be arbitrary, and let $H = h * \mathcal{N}(0, \sigma^2 I)$. Then the function $\Phi^{-1}(H(x))$ is σ-Lipschitz.*

Proof. Let's first assume that $\sigma = 1$. Then, we have that

$$\nabla \Phi^{-1}(H(x)) = \frac{\nabla H(x)}{\Phi'(\Phi^{-1}(H(x)))} \,.$$

Thus we need to show that for any unit vector v, we have that

$$\langle v, \nabla H(x)\rangle \leq \Phi'(\Phi^{-1}(H(x))) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\Phi^{-1}(H(x))^2\right) \,.$$

By Stein's lemma (Lemma 2.3), the LHS is simply

$$\mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)} [\langle v, X\rangle \cdot h(x + X)] \,. \qquad (7)$$

and we need to bound the maximum of this quantity subject to the constraint that $h(x) \in [0,1]$ for all x and $\mathbb{E}_{x\sim\mathcal{N}(0,I)}[h(x+X)] = p$. Let $f(z) = h(z+x)$, so that the problem simply becomes

$$\begin{aligned} &\max \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle \cdot f(X)] \\ &\text{s.t.} \qquad f(x) \in [0,1] \qquad \text{and} \qquad \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[f(X)] = p\ . \end{aligned}$$

We claim the solution to this optimization problem is given by the halfspace

$$\ell(z) = \mathbf{1}\left[\langle u, z\rangle > -\Phi^{-1}(p)\right]\ .$$

To see this, first observe that it's easy to show that ℓ is a valid solution. Let f be any other possible solution. Let A be the support of ℓ. Then by assumption we have that $\mathbb{E}_{X\sim\mathcal{N}(0,I)}[\ell(X) - f(X)] = 0$, so in particular we must have that

$$\mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[(\ell(X) - f(X))\mathbf{1}_A] = \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[(f(X) - \ell(X))\mathbf{1}_{A^c}]\ .$$

However, for any $z \in A$ and $z' \in A^c$, we have that $\langle v, z\rangle \geq \langle v, z'\rangle$, and hence

$$\mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle\,(\ell(X) - f(X))\mathbf{1}_A] \geq \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle\,(f(X) - \ell(X))\mathbf{1}_{A^c}]\ ,$$

where this uses that $\ell(z) \geq f(z)$ if $z \in A$ and $f(z) \geq \ell(z)$ otherwise. Rearranging, yields

$$\mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle\,\ell(X)] \geq \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle\,f(X)]\ ,$$

as claimed. Now, we simply observe that

$$\begin{aligned} \mathop{\mathbb{E}}_{X\sim\mathcal{N}(0,I)}[\langle v, X\rangle\,\ell(X)] &= \mathop{\mathbb{E}}_{Z\sim\mathcal{N}(0,1)}[Z\cdot\mathbf{1}_{Z>-\Phi^{-1}(p)}] \\ &= \frac{1}{\sqrt{2\pi}}\int_{-\Phi^{-1}(p)}^{\infty} xe^{-x^2/2}dx \\ &= \exp\left(-\frac{1}{2}\Phi^{-1}(p)^2\right) = \exp\left(-\frac{1}{2}\Phi^{-1}(H(x))^2\right)\ , \end{aligned}$$

as claimed.

To demonstrate the Theorem for general σ, note that we can take the auxiliary function $\widetilde{h}(z) = h(z/\sigma)$, and the corresponding smoothed function $\widetilde{H} = \widetilde{h} * \mathcal{N}(0,1)$. Then $\widetilde{H}(\sigma x) = H(x)$. By the same proof as before, $\Phi^{-1}\circ\widetilde{H}$ is 1-Lipschitz, and this immediately implies that $\Phi^{-1}\circ H$ is σ-Lipschitz. □

Combining this with Lemma 2.2 yields Theorem 1.1.

3 Training for randomized smoothing

We now turn our attention to how to actually train for randomized smoothing. This is crucial for randomized smoothing to be effective. For instance, if I take a standard neural network, trained via vanilla training, its associated smoothed classifier won't in general even get good clean accuracy, much less robust accuracy. Indeed, this holds true even if I take an adversarially trained neural network. So making sure that your base classifier has good performance when smoothed will turn out to be very important. Moreover, we will also want the smoothed classifier to be robust, as after all, if it's not robust, then we cannot certify its robustness.

Attempt #1: Gaussian data augmentation. This is a very natural idea. We want the network to classify well under Gaussian noise. Thus, instead of training your network on your dataset, train it on Gaussian perturbations of the dataset. This is what [3] do, and by doing so were already able to obtain SOTA numbers for certified accuracy, in a number of setting. See Tables 1 and 2 for more details. However, we can improve upon this.

Attempt #2: Directly train the smoothed classifier. While Gaussian data augmentation is going in the right direction, in some sense, it is optimizing the wrong objective. To be concrete, suppose we have a loss function ℓ, and if our neural network is F_θ, parameterized by model weights θ, and our labeled dataset comes from a distribution D then what Gaussian data augmentation is doing is attempting to optimize the following objective:

$$\min \mathop{\mathbb{E}}_{\substack{(X,y)\sim D \\ \delta\sim\mathcal{N}(0,\sigma^2 I)}} [\ell(F_\theta, X+\delta, y)]$$

Let's make this more explicit: suppose ℓ is the cross entropy loss. Then this is exactly

$$\min \mathop{\mathbb{E}}_{(X,y)\sim D} \mathop{\mathbb{E}}_{\delta\sim\mathcal{N}(0,\sigma^2 I)} [-\log(F_\theta(X+\delta))_y] \ . \tag{8}$$

But what we actually care about is whether or not the associated smoothed classifier G_θ itself classifies well. This is the (similar-looking) expression

$$\min \mathop{\mathbb{E}}_{(X,y)\sim D} [-\log(G_\theta(X))_y] = \min \mathop{\mathbb{E}}_{(X,y)\sim D} \left[-\log \left(\mathop{\mathbb{E}}_{\delta\sim\mathcal{N}(0,\sigma^2 I)} [F_\theta(X+\delta)]_y \right) \right] \ . \tag{9}$$

The subtle (but important) distinction between these two expression is simply that the logarithm and the expectation have been switched. However, there is a qualitative distinction: (8) attempts to find a classifier F_θ which has small loss at $X+\delta$, where X is a natural image, and δ is a Gaussian perturbation. On the other hand (9) asks us to directly find a classifier F_θ whose associated smoothed classifier has small loss. Since the latter actually optimizes the smoothed classifier, it should not be shocking that if we train F_θ using (9), then we should expect that its smoothed classifier to perform better.

However, it is not quite clear how to optimize (9). Typically, one needs to do SGD on the objective. Sampling (X, y) is the same as before: simply iterate over your dataset. The main difficulty is then how to compute the gradient at a point (X, y), i.e. we need to estimate

$$\nabla \left(-\log \left(\mathop{\mathbb{E}}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} [F_\theta(X+\delta)]_y \right) \right) . \tag{10}$$

Specifically, because the logarithm and the expectation are switched, it is not obvious how to get good gradient estimators for (10). There are a number of things one can try: for instance, Stein's Lemma actually gives a gradient-free way to try to estimate this gradient. However, [4] found that the following "plug-in" estimator works quite well in practice, although it does not have nice theoretical guarantees: simply take $\delta_1, \ldots, \delta_k \sim \mathcal{N}(0, \sigma^2 I)$, and use the estimator

$$\nabla \left(-\log \left(\mathop{\mathbb{E}}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} [F_\theta(X+\delta)]_y \right) \right) \approx \nabla \left(-\log \left(\frac{1}{k} \sum_{i=1}^{k} [F_\theta(X+\delta_i)]_y \right) \right) .$$

This estimator is asymptotically consistent (under mild assumptions on F_θ), but it lacks essentially any other nice property: for instance, it is not unbiased. But it seems to work.

Attempt #3: Adversarially train the smoothed classifier The final optimization which helps the performance of randomized smoothing considerably is to not just train the smoothed classifier for standard accuracy, but to *adversarially train* it. Intuitively, the smoothed classifier needs to be robust if we have any hope of certifying that it is robust. Hence, we should just use whatever heuristics we can to try to make it as empirically robust as possible, and adversarial training is the strongest technique we know of. This is the algorithm proposed in [4] which significantly boosts the performance of [3]. By directly doing so, [4] improves upon the robust accuracy on CIFAR-10 by up to 15%, and by combining it with other techniques which encourage empirical robustness, such as pretraining and semi-supervision, they achieve accuracy which is better by up to 20%. They also get improvements of $5-6\%$ on Imagenet. See Tables 1 and 2 for more details.

Table 1: Certified top-1 accuracy of randomized smoothing for ImageNet classifiers at various ℓ_2 radii. Table reproduced with consent from at least one author from [4].

ℓ_2 radius (ImageNet)	0.5	1.0	1.5	2.0	2.5	3.0	3.5
[3] (%)	49	37	29	19	15	12	9
[4] (%)	**56**	**43**	**37**	**27**	**25**	**20**	**16**

Table 2: Certified top-1 accuracy of randomized smoothing for CIFAR-10 classifiers at various ℓ_2 radii. Table reproduced with consent from at least one author from [4].

ℓ_2 Radius (CIFAR-10)	0.25	0.5	0.75	1.0	1.25	1.5	1.75	2.0	2.25
[3] (%)	61	43	32	22	17	13	10	7	4
[4] (%)	73	58	48	38	33	29	24	18	16
+ Pre-training (%)	80	62	**52**	38	**34**	**30**	**25**	**19**	16
+ Semi-supervision (%)	80	**63**	**52**	**40**	**34**	29	**25**	**19**	**17**
+ Both(%)	**81**	**63**	**52**	37	33	29	**25**	18	16

References

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

[2] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2018.

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

[4] Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. *arXiv preprint arXiv:1906.04584*, 2019.

[5] Charles M Stein. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151, 1981.