CHAPTER 13

Randomization-based defense

Randomized components are expected to make the attacker harder to succeed, so several earlier defensive methods add randomized components to a neural network to make it more robust. However, later on a stronger attack based on expectation over transformations (EoT) (Athalye and Sutskever, 2018; Athalye et al., 2018) is developed for evaluating the robustness of randomized models (see Chapter 4 for details on EoT attacks), and some naive ways for adding randomness may not really improve robustness under the EoT attacks. Nevertheless, later works have demonstrated that some types of randomness can improve robustness, and surprisingly, it has been shown that a randomized component can theoretically improve the robustness of a model, and further, the robustness of such randomized models can be certifiable. In this section, we first discuss some earlier attempts in introducing randomness and how to properly evaluate the robustness of randomized models. Then we discuss two kinds of randomness, adding to either the first layer or all the layers, where they can theoretically improve the robustness.

13.1 Earlier attempts and the EoT attack

Adversarial perturbation can be viewed as a special type of additive noise, and various methods have been proposed to improve the robustness of DNNs by incorporating random components into the model. For instance, Xie et al. (2017) introduced a simple preprocessing method to randomize the input of neural networks, hoping to remove the potential adversarial perturbation. During the testing phase, the input is randomly resized into several different sizes, and then around each of the resized inputs, zeros are randomly padded. The authors demonstrated that this simple method can be applied to large-scale datasets, such as Imagenet. Similarly, Zantedeschi et al. (2017) showed that by using a modified ReLU activation layer (called BReLU) and adding noise to the origin input to augment the training data, the learned model will gain some stability to adversarial examples. On the other hand, Dhillon et al. (2018) introduce randomness into the evaluation of a neural network to defend against adversarial examples. Their method randomly drops some neurons of each layer to 0 with probability proportional to their absolute value. That is, their method essentially applies dropout at each layer, where instead of dropping with uniform probability, nodes are dropped according to a weighted distribution. Values that are retained are scaled up (as is done in dropout) to retain accuracy.

Despite those defense methods demonstrated good performance against regular attack methods (such as FGSM or PGD attacks), it has been shown that under a more carefully designed attack, those simple randomization methods may not work. The main trick is that when conducting an attack, we have to consider the randomness in the model. Let $f_{\theta,\tau}$ be the randomized model with a randomized component τ sampled from some distribution. In the original PGD attack, each step will compute the gradient on a particular random τ, which encounters high variance and may encounter difficulties to converge to a good solution. Therefore, instead of taking $\nabla_x f_{\theta,\tau}(x)$ with one particular τ, we need to take several samples and use the average to estimate the gradient. More specifically, each iteration will take the following form of gradient estimation:

$$\frac{1}{R}\sum_{i=1}^{R}\nabla_x f_{\theta,\tau_i}(x), \tag{13.1}$$

where $\tau_1,\ldots,\tau_R$ are sampled from the distribution of τ. Therefore, even though PGD takes more time at each iteration, it can obtain a much stronger adversarial example against the randomized model $f_{\theta,\tau}$, by performing parameter update using the averaged gradient. It has been shown by Athalye et al. (2018) that under this stronger attack, these previously introduced simple randomized defense models will fail.

13.2 Adding randomness to each layer

Liu et al. (2018c) demonstrated a simple strategy that can make randomized defense achieve similar robustness performance as adversarial training, even under stronger EoT attacks. In this method, they define a "noise layer" to introduce randomness to both the input and the hidden layer output. In this "noise layer", randomly generated Gaussian noise is added to the input:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I),$$

where σ is a hyperparameter. Larger values of σ lead to better robustness but worse prediction accuracy, whereas smaller values of σ result in

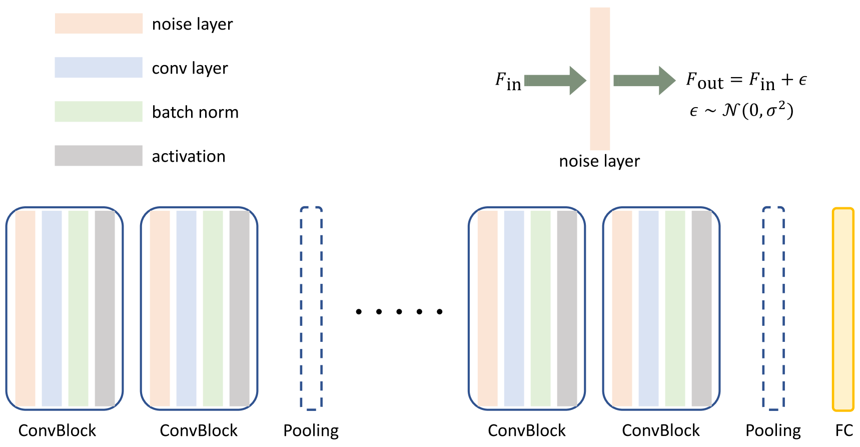

Figure 13.1 In random self-ensemble (RSE) the noise layer is added into the beginning of each convolution block for computer vision applications.

better prediction accuracy but deteriorated robustness. One can then add this noise layer into any place of the neural networks. For example, Liu et al. (2018c) add the noise layer into the beginning of each convolution block. This method, called the random self-ensemble (RSE), is illustrated in Fig. 13.1.

In RSE, a noise layer is added in both training and testing time. In the training phase, gradient for the parameter update is computed as $\nabla_\theta f_{\theta,\tau}(x)$ for each sample x, and $f_{\theta,\tau}$ is a classical convolutional neural network with noise layer added. The noise is generated randomly for each stochastic gradient descent update. In the testing phase, instead of a single prediction, we need to compute the expected prediction value over the random variable τ:

$$E_\tau[f_{\theta,\tau}(x)]. \tag{13.2}$$

As computing the real expected prediction requires huge amount of samples, we can approximate it by sampling only R random noises and compute the average prediction. Usually, the prediction is already stable when $R \approx 10$:

$$p = \sum_{j=1}^{R} f_{\epsilon_j}(w, x), \text{ and predict } \hat{y} = \arg\max_k p_k. \tag{13.3}$$

It has been demonstrated that under a proper EoT attack, this method is only slightly worse than the plain adversarial training method discussed

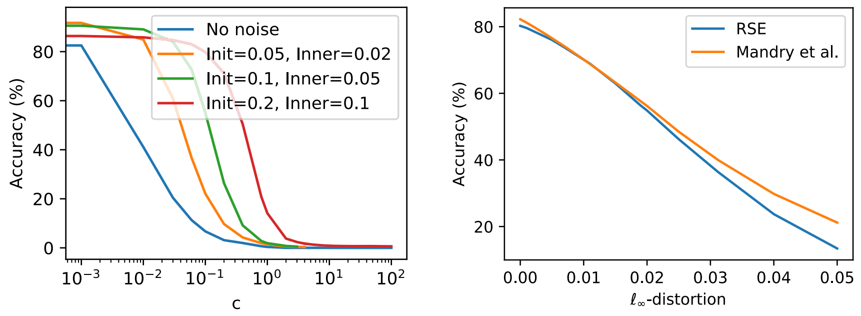

Figure 13.2 *Left*: the effect of noise level on robustness and generalization ability. Clearly, random noise can improve the robustness of the model. *Right*: comparing RSE with adversarial defense method of Madry et al. (2018).

in the previous chapter. Some experimental results are shown in Fig. 13.2. This surprising finding leads to several later investigations on why randomized ensemble can improve adversarial robustness.

But why can the robustness be improved when adding randomness? Wang et al. (2019a) and Liu et al. (2020c) provide a nice interpretation via the neural ordinal differential equation framework. Traditional neural networks are usually stacked with multiple layers; recent work (Chen et al., 2018d) shows that we can model it in the continuous limit. This means that there is no notion of discrete layers, and hidden features are changed smoothly. Mathematically, it has the following form:

$$h_t = h_s + \int_s^t f(h_\tau, \tau; w) d\tau, \tag{13.4}$$

where $t > s$ are two different layers, but unlike the traditional discrete notion of layer, here we assume the index of the layer (t, s) are continuous variables. h_t is the hidden features at layer t. The f function characterizes how to transform from one layer to another layer, which corresponds to the residual block parameterized by w. This formula is exactly the continuous limit of the original Residual Network (ResNet) architecture proposed in (He et al., 2016) structure

$$h_{n+1} = h_n + f(h_n; w_n), \tag{13.5}$$

where the layer index $n = 1, 2, \dots, N$ is discrete.

Notice that the original neural ODE model does not contain any randomness in hidden features h_t. Thus it is not ready to model a variety of random neural networks (such as dropout). To address this limitation, Liu et al. (2020c) proposed to augment the original neural ODE model (13.4) with two kinds of stochastic terms: one is the diffusion term (to model Gaussian noise), and the other is jump term (to model Bernoulli noise); formally,

$$\begin{aligned} h_t = h_s + \underbrace{\int_s^t f(h_\tau, \tau; w) d\tau}_{\text{drift term}} + \underbrace{\int_s^t G(h_\tau, \tau) dB_\tau}_{\text{diffusion term}} \\ + \underbrace{\int_s^t J(h_\tau, \tau) \odot Z_{N_\tau} dN_\tau}_{\text{jump term}} . \end{aligned} \tag{13.6}$$

Compared with neural ODE model in (13.4) that only contains deterministic component (drift term), we add two extra terms in (13.6) to model different nature of randomness, diffusion and jump terms. The diffusion term consists of Brownian motion B_t and its coefficient G (optionally) parameterized by unknown variables v. Inside the jump term, the deterministic function $J(h_\tau, \tau)$ controls the jump size, the random variables $Z_{N_\tau} \sim \text{Bernoulli}(\pm 1, p)$ control the direction; and $N_\tau \sim \text{Poisson}(\lambda\tau)$ is a Poisson counting process controlling the "frequency" of jumps.

To explain the supervior robustness performance of RSE, we just need the diffusion term. The dB_t term is also know as the Brownian motion, which can be viewed as i.i.d. Gaussian random variables with distribution $\mathcal{N}(0, dt)$. This is thus a small Gaussian random variable added to the hidden state, similar to the Gaussian noise in RSE but at the continuous limit.

The behavior of model's prediction, in the Neural ODE case, can be viewed as how the initial value (h_0) affects the final value (h_t) in the the differentiable equation like (13.4). The solution of the ODE system is usually more sensitive to the initial point, however, it has been shown that the diffusion term in (13.6) can lead to the "converged" solution under certain input perturbation. More specifically, it can be shown that under certain condition, the Neural SDE system (13.6) has a converged value of $\lim_{t\to\infty} h_t$, which means the prediction will be stable with respect to input perturbation, when there's infinite depth. More details can be found in (Liu et al., 2020c).

In addition to adding randomness into each convolution block, another method (Liu et al., 2019d) introduced a new min-max formulation to combine adversarial training with Bayesian neural networks (BNNs). All weights in a Bayesian neural network are represented by probability distributions over possible values, rather than having a single fixed value. The proposed framework, called Adv-BNN, combines adversarial training with randomness and is shown to have significant improvement over previous approaches including RSE and Madry's adversarial training.

13.3 Certified defense with randomized smoothing

In the previous section, we have shown that adding randomized components in neural network can improve adversarial robustness, with a theoretical justification by explaining the behavior of neural networks under a neural SDE framework. However, as this analysis often requires taking limits to infinite numbers of layers, it is difficult to actually compute the actual robustness of those networks in practice.

Surprisingly, it has been shown that by properly adding randomness into the input layer of neural networks, the robustness of such randomized network can be certifiable. That is, for those networks, we will be able to obtain a robust radius r for each point x, where it is guaranteed that with high probability, any perturbation within those robust radius cannot change the prediction of neural networks. This technique is called randomized smoothing, first proposed by Lecuyer et al. (2019), and then the bounds are significantly improved in (Cohen et al., 2019; Li et al., 2019a).

In randomized smoothing, we consider a classification model f that maps inputs in $\mathbb{R}^d$ to classes $\mathcal{Y}$. Randomized smoothing is a method for constructing a "smoothed" classifier g from f. For input x, the smoothed classifier g returns whichever class the base classifier f is most likely to return when x is perturbed with a Gaussian random noise. This can be mathematically defined as

$$g(x) = \arg\max_{y \in \mathcal{Y}} P(f(x+\epsilon) = y), \tag{13.7}$$

$$\text{where } \epsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{13.8}$$

where σ is the standard deviation of an isotropic Gaussian distribution controlling the noise added to the classifier. $P(\cdot)$ is the probability for the event. Note that this is equivalent to adding the noise layer introduced in RSE into the input of neural network, whereas all the intermediate layers are unperturbed.

Why adding noise to the input layer can improve robustness? The first paper proposing randomized smoothing established robustness guarantees by connecting randomized smoothing with differential privacy (Lecuyer

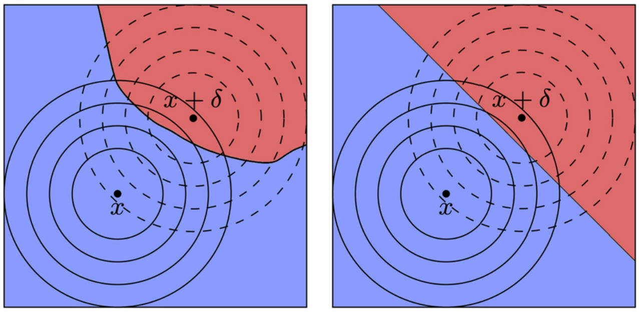

Figure 13.3 Illustration of randomized smoothing in (Cohen et al., 2019). The concentric circles are the density contours of Gaussian random noise added to the input, and the color denotes the binary prediction yielded by the classifier. The ratio of blue (mid gray in print version) and red (dark gray in print version) around x and $x + \delta$ correspond to the predicted probability of blue class versus red class in smoothed classifier, and we can observe that the change of this ratio can be bounded when perturbing x to $x + \delta$.

et al., 2019). Differential privacy adds randomness so that different inputs cannot be distinguished by only looking at the output, which is equivalent to saying "any nearby input yields the same or similar prediction", and this is exactly the definition of robustness here.

However, later on Cohen et al. (2019) established a simpler way to obtain a better certifiable robustness bound. The intuition can be illustrated in Fig. 13.3. Assume that f is the base classifier with binary prediction, each point in the figure corresponds to an input (assuming 2D) of x, and the concentric circles are the density contours of Gaussian random noise added to x or $x+\delta$. We define $p_A(x)$ as the probability that $f(x+\epsilon) = A$ and $p_B(x)$ as the probability that $f(x + \epsilon) = B$. They then correspond to the ratio of blue (mid gray in print version) and red (dark gray in print version) colors in the solid concentric circles around x. After adding the perturbation δ, we have another point $x + \delta$, and we can similarly define $p_A(x + \delta)$ and $p_B(x + \delta)$ based on the dashed concentric circles.

Assume that the smoothed classifier predicts correctly on x, which implies $p_A(x) > p_B(x)$ and there exists a positive gap $p_A(x) - p_B(x) > 0$. We then want to show that when $\|\delta\|$ is small enough, there's still a positive gap between $p_A(x + \delta) - p_B(x + \delta)$. Note that the actual value of $p_A(x + \delta) - p_B(x + \delta)$ depends on the model itself, which corresponds to the placement of the blue and red color in the space. However, it is hard to track the actual placement of red/blue, so to establish a bound on robustness, we are then interested in the "worst" placements of red and blue regions (the "worst" classifier) that will minimize $p_A(x + \delta) - p_B(x + \delta)$. In another words, assuming that $p_A(x)$ and $p_B(x)$ are fixed, we want to find

$$\min_f p_A(x+\delta) - p_B(x+\delta) \quad \text{s.t.} \quad p_A(x) = p_A, \; p_B(x) = p_B. \tag{13.9}$$

Interestingly, if the noise is Gaussian, then there is a very simple way to obtain the worst-case f for (13.9). Intuitively, when we paint a certain point z with red color (corresponding to class B), its contribution to the overall p_B will be enlarged by the ratio $P(N(x+\delta, \sigma^2) = z)/P(N(x, \sigma^2) = z)$ when x is perturbed to $x+\delta$. Therefore the worst-case placements of the colors will be assign red points according to the decreasing order of $P(N(x+\delta, \sigma^2) = z)/P(N(x, \sigma^2) = z)$ until utilized all the mass of red. This is true for any distribution, but for general distribution, it may be hard to calculate an explicit form of this. However, things become simple when we add Gaussian random noise. From Fig. 13.3, $p_B(x)$ is the Gaussian centered at x, and $p_B(x+\delta)$ is another Gaussian centered at $x+\delta$, and if we compute the ratio, then we will see that

$$\frac{P(N(x+\delta, \sigma^2) = z)}{P(N(x, \sigma^2) = z)} = \frac{e^{-\|z-(x+\delta)\|^2/\sigma^2}}{e^{-\|z-x\|^2/\sigma^2}} = Ce^{-((z-x)^T\delta)/\sigma^2}, \tag{13.10}$$

where C is a constant irrelevant to z. Therefore the worst-case f can be illustrated in the right panel of Fig. 13.3, where we place all the red mass to the top-right corner, and the boundary is a hyperplane orthogonal to the perturbation δ. As a Gaussian distribution, when projecting to the 1D space, is still a Gaussian distribution, if we consider the 1-D space pointing from x to $x+\delta$, then the boundary will be the point for which the probability mass of the right-hand side is equal to $P_B(x)$. Therefore the cutting point will be $\Phi^{-1}(p_B)$, where Φ^{-1} is the inverse of standard Gaussian CDF.

By the argument above we can establish the certified robustness bound for randomized smoothed model in the binary case, as follows.

Theorem 2 (Cohen et al., 2019). *For binary classification problem with two classes A, B, suppose $p_A \in (\frac{1}{2}, 1]$. Then $g(x+\delta) = c_A$ for all $\|\delta\|_2 < \sigma\Phi^{-1}(p_A)$.*

In practice, we cannot really compute p_A as it requires infinite amount of samples, but we can sample m points to estimate p_A (ratio of points predicted as class A) and then use some statistical bounds (e.g., Chernoff bounds) to obtain a high probability lower bound of p_A, denoted as $\underline{p_A}$, and then apply Theorem 2 with all p_A replaced by $\underline{p_A}$ to obtain the certified robustness.

It is easy to use a similar way to establish guarantees for multiclass classification models. Assume that there are K classes in total and the smoothed classifier predicts class A. Then for any other class B, we can establish a similar guarantee as follows.

Theorem 3 (Cohen et al., 2019). *For multiclass classification problem with K classes, suppose that for any pair of classes A, B, we have bounds $\underline{p_A}$, p_B satisfying*

$$P(g(x) = A) \geq \underline{p_A} \geq p_B \geq \max_{c \neq A} P(g(x) = C).$$

Then $g(x+\delta) = A$ for any $\|\delta\|_2 < R$, where

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

When applying randomized smoothing to a pretrained model, we need to choose the parameter σ that corresponds to the noise level. Usually, a smaller σ leads to more accurate prediction as the prediction is closer to the original model f, but to a weaker certified robust radius. On the other hand, adding a very large σ will destroy the original prediction while having more robustness.

Note that the ability to be easily applied to any pretrained model is a strength of randomized smoothing. However, as most of the pretrained models will have significantly degraded performance when adding random noise, this often leads to poor robustness guarantees. A better choice is to also add random perturbation in the training stage, so the model is "aware of" the random input noise in the training stage and can perform properly when applying randomized smoothing. This is also very similar to the RSE technique described in the previous subsection, where random noise is added in both training and testing stages. Further, Salman et al. (2019a) showed that conducting adversarial training in the training phase can also improve the certified robustness through randomized smoothing.

The above-mentioned randomized smoothing technique is mostly designed for Gaussian noise. We can see in the analysis that the symmetry of Gaussian noise is very helpful. However, the natural of Gaussian noise only helps to establish the guarantees for ℓ_2 robustness, and if one is interested in bounding the perturbation with another norm (e.g., ℓ_1 or ℓ_∞ norm), then bounding the other ℓ_p norm based on the ℓ_2 robust radius will lead to very loose bounds. Therefore there have been many works trying to extend randomized smoothing to other norms or discuss the difficulty when certifying for some particular norms (Blum et al., 2020; Yang et al., 2020c; Dvijotham et al., 2020).

13.4 Extended reading

- Higher-order randomized smoothing: (Mohapatra et al., 2020).
- Data poisoning attack on randomized smoothing: (Mehra et al., 2021a).
- Randmoized denoizing for certifying pre-trained classifiers: (Salman et al., 2020b).
- Randmoized defense based on hierarchical random switching: (Wang et al., 2019e).