CHAPTER 14

Certified robustness training

In the end of the last chapter, we have shown a way to compute the "certified radius" when conducting randomized smoothing, where it is guaranteed (with high probability) that any perturbation within such radius will not change the prediction. In this chapter, we introduce another way to establish the certified robustness guarantees for neural networks without using randomized techniques, and the resulting robustness guarantees are deterministic (with 100% probability).

The main idea of this technique, also known as certified robust training, is based on incomplete neural network verification methods introduced in Part 3. Note that for any given neural network, we can apply any verification techniques introduced in Part 3 to obtain a certified robust radius for each point. Therefore, even without conducting any form of specialized training, we can obtain a certified robust loss or error for any neural network based on the verification methods. However, as complete verification methods are time-consuming to run on larger models (with exponential time complexity), and incomplete verification methods usually give us very loose bounds, applying verification on a standard neural network model can give very low certified robustness. To resolve this issue, we will show that it is possible to define a loss function based on the certified robustness obtained by incomplete verification. When training a model to minimize this loss function, it will achieve reasonably well certified robustness, at least for some standard classification tasks.

14.1 A framework for certified robust training

For training a robust model, we care about the robust error defined as

$$E_{(x,y)} \max_{x' \in B(x)} I(f(x'), y), \tag{14.1}$$

where $I(f(x'), y)$ returns 1 if the (top-1) prediction $f(x')$ does not match the groundtruth label y, and we consider the worst-case perturbation within a perturbation set, so any instance (x, y) is considered correct only if it is predicted correctly and any perturbation within the perturbation set cannot

https://doi.org/10.1016/B978-0-12-824020-5.00025-9

change its prediction. However, it is impossible to compute the exact robust error for general neural network, so there are two ways to evaluate this value. The first way is to conduct an adversarial attack to find adversarial example x', which is equivalent to finding a nonoptimal solution for the inner maximization problem, and the resulting value (error under attack) will be a lower bound of the real robust error. However, this lower bound can be far from the exact robust error, which sometimes gives a false sense of security. On the other hand, in this chapter, we care about "certified robust error", where we use some verification method to obtain upper and lower bounds of the prediction value of $f(x')$ for $x' \in B(x)$. Using this value, we can establish an *upper bound* of robust error, where we consider a sample to be robustly correct only if it is predicted correctly and it can be verified that within $B(x)$ that the (top-1) prediction never changes. For many security-sensitive applications, when we want to prove that the prediction is definitely correct within a certain region, this is much more important than errors under attacks. Further, as we discussed previously, some models are robust against certain attacks but vulnerable to others, so error under attack may not be a reliable measurement to robustness and can deliver biased evaluation of security. Therefore, for certified defense, we will focus on improving the certified robust error instead of error under attacks.

As it is impossible to directly minimize the 0/1 error in (14.1), a commonly used technique is defining a loss function as a surrogate for robust error and learning a model to minimize an upper bound of this robust loss. For example, we can define the robust cross-entropy loss as

$$E_{(x,y)} \max_{x' \in B(x)} \ell_{CE}(f(x'), y). \tag{14.2}$$

Computing the upper bound of the CE loss is based on any neural network verification technique. Note that neural network verification typically bounds $c^T f(x)$ for any $x \in B(x)$, where c is a specification vector. To ensure that the logit of target label i is larger than another label j, we can set $c_i = 1$, $c_j = -1$, and all other elements of c to be zeros. To consider all the labels, we then define a set of L specifications:

$$C_{i,j} = \begin{cases} 1 & \text{if } j = y, i \neq y \text{ (output of ground truth class)} \\ -1 & \text{if } i = j, i \neq y \text{ (output of other classes, negated)} \\ 0 & \text{otherwise (note that the } y\text{th row contains all 0).} \end{cases} \tag{14.3}$$

Each element in vector $m := Cf(x)$ gives us margins between class y and all other classes. We define the lower bound of $Cf(x)$ for all $x \in B(x)$ as $\underline{m}(x, \epsilon)$,

which is a very important quantity: when all elements of $\underline{m}(x, \epsilon) > 0$, x is verifiably robust for any perturbation within $B(x)$. The bound $\underline{m}(x, \epsilon)$ can be obtained by an incomplete neural network verification algorithm, such as convex adversarial polytope, interval bound propagation (IBP), or CROWN, as introduced in Part 3. Based on this formulation, Wong and Kolter (2017) showed that for the cross-entropy (CE) loss,

$$\max_{x \in B(x)} L(f(x); y; \theta) \leq L(f(-\underline{m}(x, \epsilon); y; \theta)). \tag{14.4}$$

Therefore we can solve the upper bound in (14.4) to minimize a tractible upper bound of the inner maximization problem in (14.2). Note that to make this training work, the verification algorithm used to obtain $\underline{m}(x_k, \epsilon)$ has to be *differentiable*. In certified robust training, at each iteration when a batch of samples are selected, we first conduct verification to form $\underline{m}(x_k, \epsilon)$ for every x_k in the batch and then perform back-propagation to compute the gradient $\nabla_\theta L(f(-\underline{m}(x_k, \epsilon); y; \theta))$ to conduct parameter updates. In what follows, we will introduce several effective verification methods that can be used for certified robust training under this framework.

14.2 Existing algorithms and their performances

There are two popular verification algorithms that have been successfully adopted in certified robust training: interval bound propagation (IBP) and linear relaxation-based training. Among them, IBP produces very loose bound but lead to surprisingly good results in certified training; linear relaxation methods provide much tighter bounds than IBP but often lead to unstable training. Therefore some recent methods also tried to combine these two bounds. We will discuss these approaches below.

Interval bound propagation (IBP)

Interval bound propagation uses a simple bound propagation rule. The idea is to obtain an upper and lower bound of each neuron layer-by-layer in forward propagation. For the input layer, we set $x_L \leq x \leq x_U$ elementwise. We then propagate the bound for each neuron to the next layer. When passing through the linear layer $z^{(l)} = Wh^{(l-1)} + b^{(l)}$, assume $\bar{h}^{(l-1)}, \underline{h}^{(l-1)}$ are the (element-wise) upper and lower bounds of $h^{(l-1)}$, we can obtain the

upper and lower bounds for $z^{(l)}$ by

$$\bar{z}^{(l)} = W^{(l)} \frac{\bar{h}^{(l-1)} + \underline{h}^{(l-1)}}{2} + |W^{(l)}| \frac{\bar{h}^{(l-1)} - \underline{h}^{(l-1)}}{2} + b^{(l)},$$
$$\underline{z}^{(l)} = W^{(l)} \frac{\bar{h}^{(l-1)} + \underline{h}^{(l-1)}}{2} - |W^{(l)}| \frac{\bar{h}^{(l-1)} - \underline{h}^{(l-1)}}{2} + b^{(l)},$$

where $|W^{(l)}|$ takes an elementwise absolute value. Note that $\bar{h}^{(0)} = x_U$ and $\underline{h}^{(0)} = x_L$. For an elementwise activation function (such as ReLU), we straightforwardly propagate the upper and lower bound through the activation function:

$$\bar{h}^{(l)} = \sigma(\bar{z}^{(l)}) \qquad \underline{h}^{(l)} = \sigma(\underline{z}^{(l)}).$$

Note that as a verification method, IBP gets much looser bounds than linear relaxation-based methods, since they only maintain the upper and lower bound for each neuron on each layer, without forming a linear approximation to the input. However, surprisingly, Gowal et al. (2018) found that training with IBP can achieve good certified robustness.

When conducting IBP-based training in a naive way, the performance is usually very bad since the bounds computed by IBP are very loose, especially at the initial point, which makes training very unstable. There are thus several important techniques used to stabilize IBP-based certified training. First, Gowal et al. (2018) proposed to use a mixture of robust cross-entropy loss with natural cross-entropy loss as the objective to stabilize training. Further, since training with IBP usually suffers from very bad initial loss, a commonly used technique called ϵ-scheduling has been used in IBP training. The main idea is to initialize ϵ (the perturbation tolerance) as 0 or very small initially, and gradually increase ϵ over iterations. However, the ϵ-scheduling also leads to significant longer training epochs. For example, to achieve state-of-the-art IBP performance, we need thousands of warmup and final training epochs. To resolve this issue, Shi et al. (2021) proposed several regularization methods including batch normalization and a penalty to control the number of inactive neurons to stabilize IBP training. With their training recipes, IBP can achieve competitive results only with around two hundred training epochs on several publich benchmark models and datasets.

Linear relaxation-based training

Based on the formulation of (14.4), it is natural to apply a tighter bound derived by linear relaxation based-methods as $\underline{m}(x, \epsilon)$. As linear relaxation-

based methods obtain linear upper and lower bounds for output neurons with respect to inputs, when plugging-in the function into $\underline{m}$, the whole loss function will be differentiable. Surprisingly, although those bounds are much tighter than IBP, when conducting certified training, it leads to much worse performance than IBP. Zhang et al. (2020a) observed that linear relaxation-based training usually leads to small norms of the model parameters, thus forcing many neurons to be inactive. A conjecture is that since linear relaxation-based methods have much tighter bounds when neurons are either inactive or active, they tend to over-regularize the network to enforce most of neurons to be inactive. However, it remains to be a open question on why this phenomenon happens more common in linear relaxation-based training than in IBP training.

However, linear relaxation-based training is still useful for certified defense. In particular, Zhang et al. (2020a) developed a Crown-IBP training method that combines the linear relaxation-based method with IBP. As mentioned before, the problem of IBP training is that the bounds tend to be very bad near initialization, which makes training extremely unstable. Crown-IBP uses CROWN training (a linear relaxation-based method) at the initial phase and then gradually switches the training objective to IBP. More specifically, Crown-IBP considers the following training objective:

$$\min_{\theta} E_{(x,y)}\big[\kappa L(x; y; \theta) + (1-\kappa)L(-((1-\beta)\underline{\mathrm{m}}_{\mathrm{IBP}}(x, \epsilon) + \beta \underline{m}_{\text{CROWN-IBP}}(x, \epsilon)); y; \theta)\big],$$

where $\underline{m}$ is a combination of two bounds such that $\underline{m}_{\text{CROWN-IBP}}$ conducts IBP in the forward bounding pass and CROWN bound propagation in a backward bounding pass. By setting β to be close to 1 in the beginning and gradually reduced to 0, the Crown-IBP training methods take the advantage of CROWN (linear relaxation-based methods) in the initial training phase but do not over-regularize the neurons. More details can be found in (Zhang et al., 2020a).

14.3 Empirical comparison

We present the comparison of existing certified training approaches on MNIST and CIFAR-10 datasets. For simplicity, we focus on the ℓ_∞ certified robust training with several commonly used settings of ϵ. Experimental results are presented in Table 14.1. Note that for IBP training, we mainly

Table 14.1 A comparison of IBP, Crown-IBP, and Improved IBP on MNIST and CIFAR-10 datasets.

Dataset	ϵ	Training Method	Standard error	Certified error
MNIST	0.4	IBP	1.66	15.01
MNIST	0.4	Crown-IBP	2.17	12.06
MNIST	0.4	Improved IBP	2.2	10.82
CIFAR-10	8/255	IBP	58.72	69.88
CIFAR-10	8/255	CROWN-IBP	53.73	66.62
CIFAR-10	8/255	Improved IBP	51.06	65.05

refer to the training recipe proposed in (Gowal et al., 2018); for Crown-IBP, we used the results from (Zhang et al., 2020a) and (Xu et al., 2020b), and for improved IBP, we consider IBP with the tricks proposed by Shi et al. (2021). For both datasets, we use the seven-layer CNN model used commonly in those papers. We can observe that current methods already achieve reasonably good performance on MNIST datasets; since each pixel value is normalized to be $[0, 1]$, a perturbation of $\epsilon = 0.4$ is already very large, and certified training can obtain models with almost 10% data correctly and robustly classified on MNIST. However, the performance of CIFAR-10 is still not ideal, so many recent works still try to improve the certified robustness accuracy on CIFAR-10 and larger-scale datasets.

14.4 Extended reading

- Fast certified robustness training: (Boopathy et al., 2021).