ECE 586 BH: Interplay between Control and Machine Learning **Fall 2023**

Lecture 7

A Control Perspective on Certifiably Robust Neural Networks, Part I

Lecturer: Bin Hu, Date:09/12/2023

In this lecture, we will study how to apply robust control tools to tackle the certified robustness issue in deep learning. We will slightly modify the quadratic constraint approach that has been covered previously, and then apply it to design certifiably robust neural networks for classification tasks.

7.1 Background: Neural Networks for Classification

Consider an image classification task. The goal is to predict the label $y \in \mathcal{Y} := \{1, 2, \ldots, k\}$ from the image $x \in \mathbb{R}^d$. Suppose a neural network classifier $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^k$ is designed to achieve this goal. For each image x, the neural network classifier $\mathbf{f}$ will generate k logits values, each of which corresponds to a certain label class. Therefore, we have

$$\mathbf{f}(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_k(x) \end{bmatrix}, \tag{7.1}$$

where $f_i(x)$ characterizes the possibility that the label of the input x is i. (If we further put the output of f through a softmax layer, then we will obtain the probability score for each label class.)

The predicted label for an image input x is $\arg\max_j f_j(x)$. For example, suppose we have a binary classification task, i.e. we have $k = 2$ and the classifier $\mathbf{f}$ is designed to predict whether an image is about a cat (class 1) or a dog (class 2). For an image x, if $f_1(x) > f_2(x)$, then the predicted label is "cat" (label 1). If $f_1(x) < f_2(x)$, the predicted label is "dog" (label 2). Given an image x and its true label y, the classifier $\mathbf{f}$ makes the correct prediction if we have $y = \arg\max_j f_j(x)$.

There are many different network structures that can be used to parameterize the classifier $\mathbf{f}$. Suppose the total layer number is $(h + 1)$. Here are a few standard network structures:

- Feedforward: $x_{k+1} = \sigma(W_k x_k + b_k)$ for $k = 0, 1, \cdots, h$
- Residual network: $x_{k+1} = x_k - \sigma(W_k x_k + b_k)$ for $k = 0, 1, \cdots, h$
- Many other structures: transformers, etc

Given any input x, the neural network is initialized at $x_0 = x$, and the output satisfies $\mathbf{f}(x) = x_{h+1}$ (the i-th entry of x_{h+1} is equal to $f_i(x)$). The nonlinear activation function σ is applied to every layer in an entry-wise manner.

7.2 Adversarial Attacks and Certified Robustness

In the deep learning field, it has been observed that a small adversarial perturbation τ can fool state-of-the-art neural network classifiers to make the wrong predictions [7]. Specifically, given an image x and its true label, let's say that the neural network classifier $\mathbf{f}$ makes the correct prediction, i.e. $y = \arg\max_j f_j(x)$. Then one can apply optimization-based algorithms such as projected gradient descent to successfully find adversarial perturbation τ such that $\|\tau\|$ is very small and $y \neq \arg\max_j f_j(x+\tau)$. This issue leads to the study of **certified robustness**, which is formally defined below.

Definition 1. *A classifier* $\mathbf{f}$ *is* certifiably robust at radius $\varepsilon \geq 0$ *at the data point* x *with label* y *if for all* τ *satisfying* $\|\tau\| \leq \varepsilon$*, we have* $\arg\max_j f_j(x+\tau) = y$.

The fragility of neural networks under small adversarial perturbations can be roughly explained from a sensitivity perspective. State-of-the-art neural networks typically rely on very large weight to boost the expressive power. For example, if a feed-forward network $x_{k+1} = \sigma(W_k x_k + b_k)$ is used, standard training procedures will make $\|W_k\|$ large and cause the network to become very sensitive to input perturbations.[1] Consequently, a small change in the input x can potentially lead to a large change in the output $\mathbf{f}(x)$. Intuitively, one can mitigate this sensitivity issue via controlling the weight norm. For example, the famous spectral normalization technique just normalizes every layer as $x_{k+1} = \sigma\left(\frac{1}{\|W_k\|}W_k x_k + b_k\right)$. The normalized matrix $\frac{1}{\|W_k\|}W_k$ does not have a big spectral norm, and the network layer becomes less sensitive to input changes. As a matter of fact, such a simple idea has led to the development of the more general Lipschitz neural network approach, which provides the state-of-the-art deterministic certified robustness results on image classification benchmarks. Formally, the classifier $\mathbf{f}$ is L-Lipschitz if $\|\mathbf{f}(x') - \mathbf{f}(x)\| \leq L\|x' - x\|$ for any $x, x' \in \mathbb{R}^d$. The following result is well-known.

Proposition 2 ([9]). *Let* $\mathbf{f}$ *be* L*-Lipschitz. Given an input* x *with the true label* y*. Suppose* $y = \arg\max_j f_j(x)$*. If we have*

$$\mathcal{M}_{\mathbf{f}}(x) := \max(0, f_y(x) - \max_{j \neq y} f_j(x)) > \sqrt{2}L\varepsilon,$$

then for every τ *satisfying* $\|\tau\| \leq \varepsilon$*, we must have* $\arg\max_j f_j(x+\tau) = y$*. In other words,* $\mathbf{f}$ *is certifiably robust at radius* ε *at the data point* x*.*

A complete proof for the above result can be found in [9], and hence is skipped here. Let's just try to understand the above result. Here $\mathcal{M}_{\mathbf{f}}(x)$ is the prediction margin of $\mathbf{f}$ at the data point x. A large margin typically means that we are confident in the predicted outcome for that data point. If the margin is quite small, it typically means that the classifier

[1] Let's think about what happens in the first layer. If $\|W_0\|$ is large, a small perturbation τ can lead to the change $W_0(x+\tau) - W_0 x = W_0\tau$, which is going to be significant if τ is chosen carefully. Such an issue exists for every layer, and the compounding effect is huge.

is not that certain about the prediction result. The prediction margin has to be calculated in a point-by-point manner. The above result just states that perturbation smaller than $\mathcal{M}_{\mathbf{f}}(x)/\sqrt{2}L$ cannot deceive $\mathbf{f}$ for datapoint x! If we can control the Lipschitz constant of $\mathbf{f}$ and achieve good prediction margins at the same time, then certified robustness can be ensured. This leads to the following approach for achieving certified robustness.

1. Constructing 1-Lipschitz layers: If each layer of a neural network is 1-Lipchitz, the entire network is 1-Lipschitz (by chain rule). So one can parameterize 1-Lipschitz neural networks via augmenting 1-Lipschitz layers and other 1-Lipschitz operations (such as average pooling set up in some proper way).

2. Training: Use a training procedure to boost the prediction margin as much as possible (typically one can modify the loss function to achieve this [5]). Boosting margins over some data points can potentially reduce the clean accuracy over other points. There is some trade-off here, and hence one needs to carefully tune the hyper-parameters.

3. Certification: For each data point in the testing set, we test whether $\mathcal{M}_{\mathbf{f}}(x) > \sqrt{2}\varepsilon$ (the network is parameterized to be 1-Lipschitz), and then count the percentage of the data points that is guaranteed to be guarded for perturbation smaller than ε. This gives us the so-called certified robust accuracy for that ε on a given data set (such as CIFAR10/100 and TinyImageNet).

7.3 Lipschitz Neural Networks

The above certified robustness approach requires parameterizing 1-Lipschitz network layers. We can clearly see that the spectral normalization method exactly follows this idea (we know ReLU is 1-Lipschitz). Now we give a brief review of existing 1-Lipschitz layers.

- Spectral normalization [4, 2]: $x_{k+1} = \sigma\left(\frac{1}{\|W_k\|_2}W_k x_k + b_k\right)$
- Orthogonal layers [8, 6, 11, 10]: $x_{k+1} = \sigma(W_k x_k + b_k)$ with $W_k^\mathsf{T} W_k = I$ (there are many ways to parameterize W_k to satisfy $W_k^\mathsf{T} W_k = I$, and one popular approach is to use Cayley transformation [8], i.e. setting $W_k = (I - (H_k - H_k^\mathsf{T}))(I + H_k - H_k^\mathsf{T})^{-1}$ with H_k being some unconstrained decision variable to be trained)
- Convex potential layer (CPL) [3]: $x_{k+1} = x_k - \frac{2}{\|W_k\|^2}W_k\sigma(W_k^\mathsf{T} x + b_k)$
- Almost-orthogonal layers (AOL) [5]: $x_{k+1} = \sigma(W_k \mathrm{diag}(\sum_j |W_k^\mathsf{T} W_k|_{ij})^{-\frac{1}{2}} x_k + b_k)$

CPL and AOL have both achieved very competitive results for certified robustness on CIFAR10/100. However, the developments of these layers are done in a case-by-case manner. Their derivations greatly differ, and strongly rely on deep expert insights. Next, we will discuss how to use the robust control theory (more specifically, the quadratic constraint approach) to unify existing 1-Lipschitz network layers and derive new network structures in a routinized manner.

7.4 Deriving Lipschitz Structures via Control Tools

We can think neural network layers as special cases of the Lur'e system that has been covered in the previous lecture:

$$x_{k+1} = A_k x_k + B_k \sigma(C_k x_k + b_k). \tag{7.2}$$

Clearly, (7.2) can cover various neural network structures if we choose (A_k, B_k, C_k) properly. If $A_k = 0$ and $B_k = I$, then (7.2) becomes the standard feed-forward network. If we choose $A_k = I$, then (7.2) reduces to the residual network. Now we can ask the question: How can we choose (A_k, B_k, C_k) to ensure (7.2) is 1-Lipschitz (i.e. $\|x'_{k+1} - x_{k+1}\| \leq \|x'_k - x_k\|$)? This is actually a control problem. To see this, notice that (7.2) can be represented as the feedback interconnection $F_u(G, \Delta)$ as shown in 7.1.

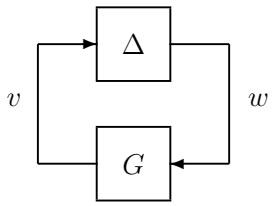

Figure 7.1. The Block-Diagram Representation for Feedback Interconnection $F_u(G, \Delta)$

We can choose G as

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k w_k \\ v_k &= C_k x_k + b_k \end{aligned}$$

In addition, we set Δ as $w_k = \Delta(v_k) = \sigma(v_k)$. We have seen how the quadratic constraint approach can be used to address such feedback interconnections in a unified manner. Now we can just apply the same approach here.

To establish the 1-Lipschitz property, we need to show $\|x'_{k+1} - x_{k+1}\| \leq \|x'_k - x_k\|$, where $\{x_k\}$ and $\{x'_k\}$ are two arbitrary trajectories of (7.2). Notice that the following relation holds

$$x'_{k+1} - x_{k+1} = A_k(x'_k - x_k) + B_k \left(\sigma(C_k x'_k + b_k) - \sigma(C_k x_k + b_k)\right) \tag{7.3}$$

If we treat $x'_k - x_k$ as the state variable ξ_k from our previous lectures, then we can immediately obtain the following result.

Theorem 7.1. *Consider the Lur'e system* (7.2)*. Suppose the following (incremental) quadratic constraint holds for any* (v, v')*:*

$$\begin{bmatrix} v' - v \\ \sigma(v') - \sigma(v) \end{bmatrix}^{\mathsf{T}} M_k \begin{bmatrix} v' - v \\ \sigma(v') - \sigma(v) \end{bmatrix} \geq 0. \tag{7.4}$$

If there exists $P \geq 0$ such that

$$\begin{bmatrix} A_k^\mathsf{T} P A_k - P & A_k^\mathsf{T} P B_k \\ B_k^\mathsf{T} P A_k & B_k^\mathsf{T} P B_k \end{bmatrix} + \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix}^\mathsf{T} M_k \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix} \leq 0, \tag{7.5}$$

then we must have $(x'_{k+1} - x_{k+1})^\mathsf{T} P (x'_{k+1} - x_{k+1}) \leq (x'_k - x_k)^\mathsf{T} P (x'_k - x_k)$, where $\{x_k\}$ and $\{x'_k\}$ are two arbitrary trajectories of (7.2).

From the above theorem, if (7.5) is feasible with $P = I$, then we immediately have $\|x'_{k+1} - x_{k+1}\| \leq \|x'_k - x_k\|$. Hence (7.5) provides a general matrix condition for designing 1-Lipschitz network layers (such a control-theoretic network design approach was originally proposed in [1]). The proof of the above theorem is straightforward. If (7.5) is feasible, we must have

$$\begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix}^\mathsf{T} \begin{bmatrix} A_k^\mathsf{T} P A_k - P & A_k^\mathsf{T} P B_k \\ B_k^\mathsf{T} P A_k & B_k^\mathsf{T} P B_k \end{bmatrix} \begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix} + \begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix}^\mathsf{T} \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix}^\mathsf{T} M_k \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix} \leq 0.$$

From previous lectures, we know that the first term can be simplified and eventually rewritten as $(x'_{k+1} - x_{k+1})^\mathsf{T} P (x'_{k+1} - x_{k+1}) - (x'_k - x_k)^\mathsf{T} P (x'_k - x_k)$. In addition, the second term can be simplified as

$$\begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix}^\mathsf{T} \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix}^\mathsf{T} M_k \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x'_k - x_k \\ w'_k - w_k \end{bmatrix} = \begin{bmatrix} v'_k - v_k \\ \sigma(v'_k) - \sigma(v_k) \end{bmatrix}^\mathsf{T} M_k \begin{bmatrix} v'_k - v_k \\ \sigma(v'_k) - \sigma(v_k) \end{bmatrix}$$

which has to be non-negative based on (7.4). Therefore, we must have $(x'_{k+1} - x_{k+1})^\mathsf{T} P (x'_{k+1} - x_{k+1}) - (x'_k - x_k) P (x'_k - x_k) \leq 0$. This leads to the desired conclusion.

How to choose M_k? It is well-known that ReLU is slope-restricted on $[0, 1]$. This is similar to choosing $m = 0$ and $L = 1$ for the quadratic constraint covered in the previous lecture. Since ReLU is applied in an entry-wise manner, we can choose any positive definite diagonal matrix Λ_k and parameterize M_k as

$$M_k = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} \otimes \Lambda_k \tag{7.6}$$

Substituting the above choice of M_K and $P_k = I$ to (7.5) leads to

$$\begin{bmatrix} A_k^\mathsf{T} A_k - I & A_k^\mathsf{T} B_k \\ B_k^\mathsf{T} A_k & B_k^\mathsf{T} B_k \end{bmatrix} + \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix}^\mathsf{T} \begin{bmatrix} 0 & \Lambda_k \\ \Lambda_k & -2\Lambda_k \end{bmatrix} \begin{bmatrix} C_k & 0 \\ 0 & I \end{bmatrix} \leq 0. \tag{7.7}$$

We can easily recover existing 1-Lipschitz layers using the above condition. If we choose $A_k = 0$, $B_k = I$, and $C_k = W_k$, then (7.7) is feasible with $\Lambda_k = I$ and W_k satisfying $\|W_k\| \leq 1$. This recovers spectral normalization, orthogonal layers, and AOL. If we choose $A_k = I$, $B_k = -\frac{2}{\|W_k\|^2} W_k$, and $C_k = W_k^\mathsf{T}$, then (7.7) is feasible with $\Lambda_k = \frac{2}{\|W_k\|^2} I$ (verify this yourself!). In the next lecture, we will apply (7.7) to derive more general layers. We will also discuss various generalizations of the condition (7.7).

Bibliography

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

[2] F. Farnia, J. Zhang, and D. Tse. Generalizable adversarial training via spectral normalization. In *International Conference on Learning Representations*, 2019.

[3] L. Meunier, B. Delattre, A. Araujo, and A. Allauzen. A dynamical system perspective for lipschitz neural networks. In *International Conference on Machine Learning*, 2022.

[4] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[5] B. Prach and C. H. Lampert. Almost-orthogonal layers for efficient general-purpose lipschitz networks. In *Computer Vision–ECCV 2022: 17th European Conference*, 2022.

[6] S. Singla and S. Feizi. Skew orthogonal convolutions. In *International Conference on Machine Learning*, 2021.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[8] A. Trockman and J. Z. Kolter. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations*, 2021.

[9] Y. Tsuzuku, I. Sato, and M. Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, 2018.

[10] X. Xu, L. Li, and B. Li. Lot: Layer-wise orthogonal training on improving l2 certified robustness. In *Advances in Neural Information Processing Systems*, 2022.

[11] T. Yu, J. Li, Y. Cai, and P. Li. Constructing orthogonal convolutions in an explicit manner. In *International Conference on Learning Representations*, 2022.