LOT: Layer-wise Orthogonal Training on Improving ℓ_2 Certified Robustness

Xiaojun Xu **Linyi Li** **Bo Li**
University of Illinois Urbana-Champaign
{xiaojun3,linyi2, lbo}@illinois.edu

Abstract

Recent studies show that training deep neural networks (DNNs) with Lipschitz constraints are able to enhance adversarial robustness and other model properties such as stability. In this paper, we propose a layer-wise orthogonal training method (LOT) to effectively train 1-Lipschitz convolution layers via parametrizing an orthogonal matrix with an unconstrained matrix. We then efficiently compute the inverse square root of a convolution kernel by transforming the input domain to the Fourier frequency domain. On the other hand, as existing works show that semi-supervised training helps improve *empirical* robustness, we aim to bridge the gap and prove that semi-supervised learning also improves the *certified* robustness of Lipschitz-bounded models. We conduct comprehensive evaluations for LOT under different settings. We show that LOT significantly outperforms baselines regarding deterministic ℓ_2 certified robustness, and scales to deeper neural networks. Under the supervised scenario, we improve the state-of-the-art certified robustness for all architectures (e.g. from 59.04% to 63.50% on CIFAR-10 and from 32.57% to 34.59% on CIFAR-100 at radius $\rho = 36/255$ for 40-layer networks). With semi-supervised learning over unlabelled data, we are able to improve state-of-the-art certified robustness on CIFAR-10 at $\rho = 108/255$ from 36.04% to 42.39%. In addition, LOT consistently outperforms baselines on different model architectures with only 1/3 evaluation time.

1 Introduction

Given the wide applications of deep neural networks (DNNs), ensuring their robustness against potential adversarial attacks [7] is of great importance. There has been a line of research providing defense approaches to improve the *empirical* robustness of DNNs [16], and certification methods to *certify* DNN robustness [13]. Existing certification techniques can be categorized as deterministic and probabilistic certifications [13], and in this work we will focus on improving the deterministic ℓ_2 certified robustness by training 1-Lipschitz DNNs.

Although different approaches have been proposed to empirically enforce the Lipschitz constant of the trained model [23], it is still challenging to strictly ensure the 1-Lipschitz, which can lead to tight robustness certification. One recent work SOC [20] proposes to parametrize the orthogonal weight matrices with the exponential of skew-symmetric matrices (i.e. $W = \exp(V - V^\intercal)$). However, such parametrization will be biased when the matrix norm is constrained and the expressiveness is limited, especially when they rescale V to be small to help with convergence. In this work, we propose a layer-wise orthogonal training approach (LOT) by parameterizing the orthogonal weight matrix with an unconstrained matrix $W = (VV^\intercal)^{-\frac{1}{2}}V$. In order to calculate the inverse square root for convolution kernels, we will perform Fourier Transformation and calculate the inverse square root of matrices on the frequency domain using Newton's iteration. In our parametrization, the output is agnostic to the input norm (i.e. scaling $V = \alpha V$ does not change the value of W). We show that such

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

parametrization achieves higher model expressiveness and robustness (Section 6.2), and provides more meaningful representation vectors (Section 6.3).

In addition, several works have shown that semi-supervised learning will help improve the empirical robustness of models under adversarial training and other settings [4]. In this work, we take the first attempt to bridge semi-supervised training with *certified* robustness based on our 1-Lipschitz DNNs. Theoretically, we show that semi-supervised learning can help improve the error bound of Lipschitz-bounded models. We also lower bound the certified radius as a function of the model performance and Lipschitz property. Empirically, we indeed observe that including un-labelled data will help with the certified robustness of 1-Lipschitz models, especially at a larger radius (e.g. from 36.04% to 42.39% at $\rho = 108/255$ on CIFAR-10).

We conduct comprehensive experiments to evaluate our approach, and we show that LOT significantly outperforms the state-of-the-art in terms of the deterministic ℓ_2 certified robustness. We also conduct different ablation studies to show that (1) LOT can produce more meaningful features for visualization; (2) residual connections help to smoothify the training of LOT model.

Technical contributions. In this work, we aim to train a certifiably robust 1-Lipschitz model and also analyze the certified radius of the Lipschitz bounded model under semi-supervised learning.

- We propose a layer-wise orthogonal training method LOT for convolution layers to train 1-Lipschitz models based on Newton's iteration, and thus compute the deterministic certified robustness for the model. We prove the convergence of Newton's iteration used in our algorithm.
- We derive the certified robustness of lipschitz constrained model under semi-supervised setting, and formally show how semi-supervised learning affects the certified radius.
- We evaluate our LOT method under different settings (i.e. supervised and semi-supervised) on different models and datasets. With supervised learning, we show that it significantly outperforms state-of-the-art baselines, and on some deep architectures the performance gain is over 4%. With semi-supervised learning, we further improve certified robust accuracy by over 6% at a large radius.

2 Related Work

Certified Robustness for Lipschitz Constrained Models Several studies have been conducted to explore the Lipschitz-constrained models for certified robustness. [23] first certifies model robustness based on its Lipschitz constant and propose training algorithms to regularize the model Lipschitz. Multiple works [5, 17, 18, 8] have been proposed to achieve 1-Lipschitz during training for linear networks by regularizing or normalizing the spectral norm of the weight matrix. However, when applying these approaches on convolution layers, the spectral norm is bounded by unrolling the convolution into linear operations, which leads to a loose Lipschitz bound [24]. Recently, [2] shows that the 1-Lipschitz requirement is not enough for a good robust model; rather, the gradient-norm-preserving property is important. Besides these training-time techniques, different approaches have been proposed to calculate a tight Lipschitz bound during evaluation. [6] upper bounds the Lipschitz with semi-definite programming while [11] upper bounds the Lipschitz with polynomial optimization. In this work we aim to effectively train 1-Lipschitz convolution models.

Orthogonal Convolution Neural Networks [14] first proposes to directly construct orthogonal convolution operations. Such operations are not only 1-Lipschitz, but also gradient norm preserving, which provides a higher model capability and a smoother training process [2]. BCOP [14] trains orthogonal convolution by iteratively generating 2×2 orthogonal kernels from orthogonal matrices. [22] proposes to parametrize an orthogonal convolution with Cayley transformation $W = (I - V + V^\intercal)(I + V - V^\intercal)^{-1}$ where the convolution inverse is calculated on the Fourier frequency domain. ECO [26] proposes to explicitly normalize all the singular values [19] of convolution operations to be 1. So far, the best-performing orthogonal convolution approach is SOC [20], where they parametrize the orthogonal convolution with $W = \exp(V - V^\intercal)$ where the exponential and transpose are defined with the convolution operation. However, one major weakness of SOC is that it will rescale V to be small so that the approximation of $\exp$ can converge soon, which will impose a bias on the resulting output space. For example, when V is very small W will be close to I. Such norm-dependent property is not desired, and thus we propose a parametrization that is invariant to rescaling. Finally, [21] proposes several techniques for orthogonal CNNs, including a generalized Householder (HH) activation function, a certificate regularizer (CReg) loss, and a last layer normalization (LLN). They can be integrated with our training approach to improve model robustness.

3 Problem Setup

3.1 Lipschitz Constant of Neural Networks and Certified Robustness

Let $f : \mathbb{R}^d \to \mathbb{R}^C$ denote a neural network for classification, where d is the input dimension and C is the number of output classes. The model prediction is given by $\arg\max_c f(x)_c$, where $f(x)_c$ represents the prediction probability for class c. The Lipschitz constant of the model under p-norm is defined as: $\text{Lip}_p(f) = \sup \frac{||f(x_1)-f(x_2)||_p}{||x_1-x_2||_p} \quad \forall x_1, x_2 \in \mathbb{R}^d$. Unless specified, we will focus on ℓ_2-norm and use $\text{Lip}(f)$ to denote $\text{Lip}_2(f)$ in this work. We can observe that the definition of model Lipschitz aligns with its robustness property - both require the model not to change much w.r.t. input changes. Formally speaking, define $\mathcal{M}_f(x) = \max_i f(x)_i - \max_{j \neq \arg\max_i f(x)_i} f(x)_j$ to be the prediction gap of f on the input x, then we can guarantee that $f(x)$ will not change its prediction within $|x' - x| < r$, where

$$r = \mathcal{M}_f(x)/(\sqrt{2}\text{Lip}(f)).$$

Therefore, people have proposed to utilize the model Lipschitz to provide certified robustness and investigated training algorithms to train a model with a small Lipschitz constant.

Note that the Lipschitz constant of a composed function $f = f_1 \circ f_2$ satisfies $\text{Lip}(f) \leq \text{Lip}(f_1) \times \text{Lip}(f_2)$. Since a neural network is usually composed of several layers, we can investigate the Lipschitz of each layer to calculate the final upper bound. If we can restrict each layer to be 1-Lipschitz, then the overall model will be 1-Lipschitz with an arbitrary number of layers.

3.2 Orthogonal Linear and Convolution Operations

Consider a linear operation with equal input and output dimensions $y = Wx$, where $x, y \in \mathbb{R}^n$ and $W \in \mathbb{R}^{n \times n}$. We say W is an orthogonal matrix if $WW^\intercal = W^\intercal W = I$ and call $y = Wx$ an orthogonal linear operation. The orthogonal operation is not only 1-Lipschitz, but also norm-preserving, i.e., $||Wx||_2 = ||x||_2$ for all $x \in \mathbb{R}^n$. If the input and output dimensions do not match, i.e., $W \in \mathbb{R}^{m \times n}$ where $n \neq m$, we say W is semi-orthogonal if either $WW^\intercal = I$ or $W^\intercal W = I$. The semi-orthogonal operation is 1-Lipschitz and non-expansive, i.e., $||Wx||_2 \leq ||x||_2$.

The orthogonal convolution operation is defined in a similar way. Let $y = W \circ X$ be an orthogonal operation, where $x, y \in \mathbb{R}^{c \times w \times w}$ and $W \in \mathbb{R}^{c \times c \times k \times k}$. We say W is an orthogonal convolution kernel if $W \circ W^\intercal = W^\intercal \circ W = I_{conv}$ where the transpose here refers to the convolution transpose and I_{conv} denotes the identity convolution kernel. Such orthogonal convolution is 1-Lipschitz and norm-preserving. When the input and output channel numbers are different, the semi-orthogonal convolution kernel W satisfies either $W \circ W^\intercal = I_{conv}$ or $W^\intercal \circ W = I_{conv}$ and it is 1-Lipschitz and non-expansive.

4 LOT: Layer-wise Orthogonal Training

In this section, we propose our LOT framework to achieve certified robustness.[1] We will first introduce how LOT layer works to achieve 1-Lipschitz. The key idea of our method is to parametrize an orthogonal convolution W with an un-constrained convolution kernel $W = (VV^\intercal)^{-\frac{1}{2}}V$. Next, we will propose several techniques to improve the training and evaluation processes of our model. Finally, we discuss how semi-supervised learning can help with the certified robustness of our model.

4.1 1-Lipschitz Neural Networks via LOT

Our key observation is that we can parametrize an orthogonal matrix $W \in \mathbb{R}^{n \times n}$ with an unconstrained matrix $V \in \mathbb{R}^{n \times n}$ by $W = (VV^\intercal)^{-\frac{1}{2}}V$ [9]. In addition, this equation also holds true in the case of convolution kernel - given any convolution kernel $V \in \mathbb{R}^{c \times c \times k \times k}$, where c denotes the channel number and k denotes the kernel size, we can get an orthogonal convolution kernel by $W = (V \circ V^\intercal)^{-\frac{1}{2}} \circ V$, where transpose and inverse square root are with respect to the convolution operations. The orthogonality of W can be verified by $W \circ W^\intercal = W \circ W^\intercal = I_{conv}$, where I is

[1] The code is available at `https://github.com/AI-secure/Layerwise-Orthogonal-Training`.

the identity convolution kernel. This way, we can parametrize an orthogonal convolution layer by training over the un-constrained parameter V.

Formally speaking, we will parametrize an orthogonal convolution layer by $Y = (V \circ V^\intercal)^{-\frac{1}{2}} \circ V \circ X$, where $X, Y \in \mathbb{R}^{c \times w \times w}$ and $w \geq k$ is the input shape. The key obstacle here is how to calculate the inverse square root of a convolution kernel. Inspired by [22], we can leverage the convolution theorem which states that the convolution in the input domain equals the element-wise multiplication in the Fourier frequency domain. In the case of multi-channel convolution, the convolution corresponds to the matrix multiplication on each pixel location. That is, let $\text{FFT} : \mathbb{R}^{w \times w} \to \mathbb{C}^{w \times w}$ be the 2D Discrete Fourier Transformation and FFT^{-1} be the inverse Fourier Transformation. We will zero-pad the input to $w \times w$ if the original shape is smaller than w. Let $\tilde{X}_i = \text{FFT}(X_i)$ and $\tilde{V}_{j,i} = \text{FFT}(V_{j,i})$ denote the input and kernel on frequency domain, then we have:

$$\text{FFT}(Y)_{:,a,b} = (\tilde{V}_{:,:,a,b}\tilde{V}^*_{:,:,a,b})^{-\frac{1}{2}}\tilde{V}_{:,:,a,b}\tilde{X}_{:,a,b}$$

in which multiplication, transpose and inverse square root operations are matrix-wise. Therefore, we can first calculate $\text{FFT}(Y)$ on the frequency domain and perform the inverse Fourier transformation to get the final output.

We will use Newton's iteration to calculate the inverse square root of the positive semi-definite matrix $A = \tilde{V}_{:,:,a,b}\tilde{V}^*_{:,:,a,b}$ in a differentiable way [15]. If we initialize $Y_0 = A$ and $Z_0 = I$ and perform the following update rule:

$$Y_{k+1} = \frac{1}{2}Y_k(3I - Z_kY_k), \quad Z_{k+1} = \frac{1}{2}(3I - Z_kY_k)Z_k, \tag{1}$$

Z_k will converge to $A^{-\frac{1}{2}}$ when $||I - A||_2 < 1$. The condition can be satisfied by rescaling the parameter $V = \alpha V$, noticing that the scaling factor will not change the resulting W. In practice, we execute a finite number of Newton's iteration steps and we provide a rigorous error bound for this finite iteration scheme in Appendix D to show that the error will decay exponentially. In addition, we show that although the operation is over complex number domain after the FFT, the resulting parameters will still be real, with the theorem below:

Theorem 4.1. *Say $J \in \mathbb{C}^{m \times m}$ is unitary so that $J^*J = I$, and $V = J\tilde{V}J^*$ for $V \in \mathbb{R}^{m \times m}$ and $\tilde{V} \in \mathbb{C}^{m \times m}$. Let $F(V) = (VV^*)^{-\frac{1}{2}}V$ be our transformation. Then $F(V) = JF(\tilde{V})J^*$.*

Proof. First, notice that $V^T = V^* = J\tilde{V}^*J^*$. Second, we have:

$$\begin{aligned}(\tilde{V}\tilde{V}^*)^{-1} &= J^*(J\tilde{V}\tilde{V}^*J^*)^{-1}J \\ &= (J^*(J\tilde{V}\tilde{V}^*J^*)^{-\frac{1}{2}}J)^2,\end{aligned}$$

so that $(\tilde{V}\tilde{V}^*)^{-\frac{1}{2}} = J^*(J\tilde{V}\tilde{V}^*J^*)^{-\frac{1}{2}}J$. Thus, we have

$$\begin{aligned}J^*F(V)J &= J^*(VV^T)^{-\frac{1}{2}}VJ \\ &\overset{\text{by } V = J\tilde{V}J^*}{=} J^*(J\tilde{V}J^*J\tilde{V}^*J^*)^{-\frac{1}{2}}J\tilde{V}J^*J = J^*(J\tilde{V}\tilde{V}^*J^*)^{-\frac{1}{2}}J\tilde{V} \\ &= (\tilde{V}\tilde{V}^*)^{-\frac{1}{2}}\tilde{V}.\end{aligned}$$

□

Remark. This theorem shows that our returned value $JF(\tilde{V})J^*$ equals to the transformed version of the original matrix $F(V) \in \mathbb{R}^{m \times m}$, and thus is guaranteed to be real.

Circular Padding vs. Zero Padding When we apply the convolution theorem to calculate the convolution $Y = W \circ X$ with Fourier Transformation, the result implicitly uses circular padding. However, in neural networks, zero padding is usually a better choice. Therefore, we will first perform zero padding on both sides of input $X^{pad} = \text{zero_pad}(X)$ and calculate the resulting $Y^{pad} = W \circ X^{pad}$ with Fourier Transformation. Thus, the implicit circular padding in this process will actually pad the zeros which we padded beforehand. Finally, we truncate the padded part and get our desired output $Y = \text{truncate}(Y^{pad})$. We empirically observe that this technique helps improve the model robustness as shown in Appendix E.3.

When Input and Output Dimensions Differ In previous discussion, we assume that the convolution kernel V has equal input and output channels. In the case $W \in \mathbb{R}^{c_{out} \times c_{in} \times k \times k}$ where $c_{out} \neq c_{in}$, we aim to get a semi-orthogonal convolution kernel W (i.e. $W \circ W^\intercal = I_{conv}$ if $c_{out} < c_{in}$ or $W^\intercal \circ W = I_{conv}$ if $c_{out} > c_{in}$). As pointed out in [9], calculating W with Newton's iteration will naturally lead to a semi-orthogonal convolution kernel when the input and output dimensions differ.

Emulating a Larger Stride To emulate the case when the convolution stride is 2, we follow previous works [20] and use the invertible downsampling layer [10], in which the input dimension c_{in} will be increased by $\times 4$ times. Strides larger than 2 can be emulated with similar techniques if needed.

Overall Algorithm Taking the techniques we discussed before, we can get our final LOT convolution layer. The detailed algorithm is shown in Appendix B. First, we will pad the input to prevent the implicit circular padding mechanism and pad the kernel so that they are in the same shape. Next, we perform the Fourier transformation and calculate the output on the frequency domain with Newton's iteration. Finally, we perform inverse Fourier transformation and return the desired output.

Comparison with SOC Several works have been proposed on orthogonal convolution layers with re-parametrization[22, 20], among which the SOC approach via $W = \exp(V - V^\intercal)$ has achieved the best performance. Compared with SOC, LOT has the following advantages. <u>First</u>, the parametrization in LOT is norm-independent. Rescaling V to αV will not change the resulting W. By comparison, V with a smaller norm in SOC will lead to a W closer to the identity transformation. Considering that the norm of V will be regularized during training (e.g. SOC will re-scale V to have a small norm; people usually initialize weight to be small and impose l2-regularization during training), the orthogonal weight space in SOC may be biased. <u>Second</u>, we can see that LOT is able to model any orthogonal kernel W by noticing that $(WW^\intercal)^{-\frac{1}{2}}W$ is W itself; by comparison, SOC cannot parametrize all the orthogonal operations. For example, in the case of orthogonal matrices, the exponential of a skew-symmetric matrix only models the special orthogonal group (i.e. the matrices with +1 determinant). <u>Third</u>, we directly handle the case when $c_{in} \neq c_{out}$, while SOC needs to do extra padding so that the channel numbers match. <u>Finally</u>, LOT is more efficient during evaluation, when we only need to perform the Fourier and inverse Fourier transformation as extra overhead, while SOC needs multiple convolution operations to calculate the exponential. We will further show quantitative comparisons with SOC in Section 6.2.

The major limitation of LOT is its large overhead during training, since we need to calculate Newton's iteration in each training step which takes more time and memory. In addition, we sometimes observe that Newton's iteration is not stable when we perform many steps with 32-bit precision. To overcome this problem, we will pre-calculate Newton's iteration with 64-bit precision during evaluation, as we will introduce in Section 4.2.

4.2 Training and Evaluation of LOT

Smoothing the Training Stage In practice, we observe that the LOT layers are highly non-smooth with respect to the parameter V (see Section 6.3). Therefore, the model is difficult to converge during the training process especially when the model is deep. To smooth the training, we propose two techniques. First, we will initialize all except bottleneck layers where $c_{in} = c_{out}$ with identity parameter $V = I$. The bottleneck layers where $c_{in} = c_{out}$ will still be randomly initialized. Second, as pointed out in [12], residual connection helps with model smoothness. Therefore, for the intermediate layers, we will add the 1-Lipschitz residual connection $y = \lambda x + (1 - \lambda) f(x)$. Some work suggests that λ here can be trainable [22], while we observe that setting $\lambda = 0.5$ is enough.

Speeding up the Evaluation Stage Notice that after the model is trained, the orthogonal kernel $\tilde{W}$ will no longer change. Therefore, we can pre-calculate Newton's iteration and use the result for each evaluation step. Thus, the only runtime overhead compared with a standard convolution layer during evaluation is the Fourier transformation part. In addition, we observe that using 64-bit precision instead of the commonly used 32-bit precision in Newton's iteration will help with the numerical stability. Therefore, when we pre-calculate $\tilde{W}$, we will first transform $\tilde{V}$ into float64. After Newton's iterations, we transform the resulting $\tilde{W}$ back to float32 type for efficiency.

4.3 Semi-supervised Learning

Existing works have shown that semi-supervised learning helps improve empirical robustness [4], while the impact of semi-supervised learning on certified robustness has not been explored yet. We aim to bridge this gap and investigate whether applying semi-supervised learning also benefits the certified robustness of Lipschitz constrained models. As we will discuss in Section 5.1, we theoretically show that the certified robustness of a Lipschitz-bounded model can be improved by semi-supervised learning. In practice, suppose we have a (small) labelled dataset $\mathcal{D}_s = \{(x_i, y_i)\}$ and a (large) unlabelled dataset $\mathcal{D}_u = \{x'_i\}$, we will first train a labeller model G_{pl} over the labelled dataset $\mathcal{D}_s$. Then we use G_{pl} to assign pseudo-labels to the unlabelled dataset to get $\mathcal{D'}_u = \{(x'_i, G_{pl}(x'_i))\}$. Finally, we use the overall dataset $\mathcal{D}_s + \mathcal{D'}_u$ to train the student model G as the final model. We assume that both G_{pl} and G will use the LOT model architecture.

5 Theoretical Analysis

In this section, we will introduce our theoretical analysis to show that the certified robustness of Lipschitz-constrained models will benefit from semi-supervised learning. We make similar assumptions with [25] which provides error bound of discrete models in semi-supervised learning, while we will analyze a (Lipschitz-bounded) continuous model and its certified accuracy.

We first introduce *data expansion* proposed in [25], which we will adopt in our analysis. Let $\mathcal{T}$ denote some set of transformations and define $\mathcal{B} \triangleq \{x' | \exists T \in \mathcal{T}$ such that $||x' - T(x)|| < r\}$ to be the set of points within distance r from any of the transformation of x. The neighbourhood of x is defined by $\mathcal{N} = \{x' | \mathcal{B}(x) \cap \mathcal{B}(x') \neq \emptyset\}$. Let P denote the data distribution and P_i denote the distribution of data in the i-th class. The expansion property is defined as:

Definition 5.1 ((a, c)-expansion, as in [25])**.** We say that the class-conditional distribution P_i satisfies (a, c)-expansion if for all $V \subseteq \mathcal{X}$ with $P_i(V) < a$, the following holds:

$$P_i(\mathcal{N}(V)) \geq \min\{cP_i(V), 1\}$$

if P_i satisfies (a, c)-expansion for all classes in P, then we say P satisfies (a, c)-expansion.

Next, we define the classifier and the loss function in the continuous case. For simplicity, we assume a binary classification task here.

Definition 5.2 (Continuous model G and loss functions)**.** We assume a binary classification task, where $G^*(x), G_{pl}(x) \in \{0, 1\}$ are discrete labels, and the trained model is $G(x) \in [0, 1]$. We define the loss function $L_{\mathsf{m}}(G, G^*) \in [0, 1]$ to be:

$$L_{\mathsf{m}}(G, G^*) = \mathbb{E}_{x \in \mathcal{X}}[G^*(x)(1 - G(x)) + (1 - G^*(x))G(x)]$$

and the error of G is defined as $\text{Err}_{\mathsf{m}}(G) = L_{\mathsf{m}}(G, G^*)$.

Remark. Here we provide a general setting here, and if we assume $G(x) \in \{0, 1\}$, $L_{\mathsf{m}}(G, G^*)$ becomes the 0-1 loss in [25].

Then we define the *marginal* consistency set $S_B^{\mathsf{m}}(G; \delta)$ and loss $R_B^{\mathsf{m}}(G; \delta)$ in the continuous case. The consistency set includes cases in which the model prediction will not change a lot within the neighbourhood, and the consistency loss measures the probability that the model is not consistent over the population.

Definition 5.3 (Marginal consistency set and consistency loss)**.** We define the set $S_B^{\mathsf{m}}(G; \delta)$ in which the ratio of marginal prediction change on each input x is no larger than δ within its neighbourhood:

$$S_B^{\mathsf{m}}(G; \delta) = \{x | \frac{G(x)}{G(x')} \geq 1 - \delta \text{ and } \frac{1 - G(x)}{1 - G(x')} \geq 1 - \delta \quad \forall x' \in \mathcal{B}(x)\}.$$

and $R_B^{\mathsf{m}}(G; \delta)$ denotes the marginal consistency loss of the probability that x is not in $S_B^{\mathsf{m}}(G; \delta)$:

$$R_B^{\mathsf{m}}(G; \delta) = \mathbb{P}_{x \in P}[x \notin S_B^{\mathsf{m}}(G; \delta)].$$

Remark. (1) We may use $S_B^{\mathsf{m}}(G)$ and $R_B^{\mathsf{m}}(G)$ for abbreviation when the choice of δ does not lead to confusion; (2) when $\delta = 0$, $S_B^{\mathsf{m}}(G)$ and $R_B^{\mathsf{m}}(G)$ requires that the prediction is exactly the same within the neighbourhood, which are reduced to $S_B(G)$ and $R_B(G)$ defined in [25].

Finally, we define the certified robustness radius $\text{CertR}(G)$ as the average radius over the population in which the model keeps its correct prediction:

$$\text{CertR}(G) = \mathbb{E}_{x \in \mathcal{X}} \left[\sup r \text{ s.t.} \forall ||x' - x||_2 < r, G(x') = G(x) = G^*(x) \right]$$

Note that when $G(x) \neq G^*(x)$, the certified radius r will be 0.

5.1 Error Bound and Certified Radius of Lipschitz-bounded Model

Let $\mathcal{M}(G_{pl}) = \{x : G_{pl}(x) \neq G^*(x)\}$ denote the set in which the pseudolabel is wrong. We will have similar separation and expansion assumptions on the data distribution as in [25].

Assumption 5.1 (Separation)**.** *We assume that P is separable with probability $1 - \mu$ by ground-truth classifier G^*, i.e., $R_B^{\mathsf{m}}(G^*; 0) \leq \mu$.*

Assumption 5.2 (Expansion)**.** *Define $\overline{a} \triangleq \max_i \{P_i(\mathcal{M}(G_{pl}))\}$ to be the maximum fraction of incorrectly pseudolabeled examples in any class. We assume that $\overline{a} < 1/3$ and P satisfies $(\overline{a}, \overline{c})$-expansion for $\overline{c} > 3$. We express our bounds in terms of $c \triangleq \min(1/\overline{a}, \overline{c})$.*

With these assumptions and let $\delta \in [0, \frac{1}{c}]$, we can compare the performance of G and G_{pl} (i.e. the performance with and without semi-supervised learning) with the following theorem:

Theorem 5.1 (Error bound with semi-supervised learning)**.** *Suppose Assumption 5.1 and 5.2 holds and let $\hat{G}$ be a minimizer of the following loss function:*

$$\hat{G} = \arg\min_G L(G) \triangleq \frac{c+3}{c-1} L_{\mathsf{m}}(G, G_{pl}) + \frac{2c+2}{c-1} R_{\mathcal{B}}^{\mathsf{m}}(G; \delta) - \mathit{Err}(G_{pl})$$

Then we can upper bound the error of $\hat{G}$ by:

$$\mathit{Err}_{\mathsf{m}}(\hat{G}) \leq \frac{4}{c-1} \mathit{Err}(G_{pl}) + \frac{2c+2}{c-1} \mu$$

In addition, we can bound the certified robustness radius w.r.t. the Lipschitz constant as below:

Theorem 5.2 (Certified radius with semi-supervised learning)**.** *Suppose Assumption 5.2 holds. Then the certified radius of a model G can be bounded by:*

$$\mathit{CertR}(G) \geq \frac{0.5 - \frac{c+3}{c-1} L_{\mathsf{m}}(G, G_{pl}) - \frac{2c+2}{c-1} R_{\mathcal{B}}^{\mathsf{m}}(G; \delta) + \mathit{Err}(G_{pl})}{\mathit{Lip}(G)}.$$

The proofs of Theorem 5.1 and 5.2 are shown in Appendix C.

Discussion From Theorem 5.1, we observe that the benefits of semi-supervised learning depend on the error of G_{pl} and data property. If the data expands and separates well and G_{pl} is accurate, c is large and μ is small, which means $\text{Err}_{\mathsf{m}}(\hat{G})$ will be smaller than $\text{Err}(G_{pl})$. Note that μ is usually small and we assume $c > 3$, so $\text{Err}_{\mathsf{m}}(\hat{G}) < \text{Err}(G_{pl})$ will hold true in most cases, indicating that semi-supervised learning helps improve the model performance. From Theorem 5.2, we can observe that given the data distribution and G_{pl}, the certified radius of the student model $\text{CertR}(G)$ can be lower-bounded by 1) how well G learns from G_{pl} ($L_{\mathsf{m}}(G, G_{pl})$); 2) how consistent G is ($R_{\mathcal{B}}^{\mathsf{m}}(G; \delta)$). These two factors also correspond to the losses in the actual training process - the cross-entropy loss measures how well G learns from G_{pl} and the Lipschitz constraint measures local consistency for a Lipschitz-bounded model[2].

6 Evaluation

In this section, we will evaluate our method LOT on different datasets and models compared with the state-of-the-art baselines based on the deterministic ℓ_2 certified robustness. We show that on both CIFAR-10 and CIFAR-100, LOT achieves significantly higher certified accuracy under different radii and settings with more efficient evaluation time. In the meantime, we conduct a series of ablation studies to analyze the representation power of our LOT, the error control of Newton's iterations, and the effects of residual connections. We show that under the semi-supervised setting, the certified accuracy of LOT is further improved and outperforms the baselines.

[2]This means that, with smaller Lipschitz, the model is more likely to be consistent within neighbourhood. In addition, the CReg loss will also improve local consistency by maximizing the prediction gap.

Table 1: Certified accuracy of 1-Lipschitz model with CReg loss and HH activation on CIFAR-10/100 in supervised setting. The LLN technique is applied on the CIFAR-100 dataset.

Model	Conv. Type	CIFAR-10				CIFAR-100				Mean Evaluation Time (sec)
		Vanilla Accuracy	Certified Accuracy at $\rho =$ 36/255	72/255	108/255	Vanilla Accuracy	Certified Accuracy at $\rho =$ 36/255	72/255	108/255	
LipConvnet-5	SOC	75.31%	60.37%	45.62%	32.38%	45.82%	32.99%	22.48%	14.79%	2.285
	LOT	**76.34%**	**62.07%**	**47.52%**	**33.99%**	**48.38%**	**34.77%**	**23.38%**	**15.44%**	**1.411**
LipConvnet-10	SOC	76.23%	62.57%	47.70%	34.15%	47.07%	34.53%	23.50%	15.66%	3.342
	LOT	**76.50%**	**63.12%**	**48.59%**	**35.65%**	**48.70%**	**34.82%**	**23.86%**	**15.93%**	**1.563**
LipConvnet-15	SOC	76.39%	62.96%	48.47%	35.47%	47.61%	34.54%	23.16%	15.09%	4.105
	LOT	**76.86%**	**63.84%**	**49.18%**	**36.35%**	**48.99%**	**34.90%**	**24.39%**	**16.37%**	**1.627**
LipConvnet-20	SOC	76.34%	62.63%	48.69%	36.04%	47.84%	34.77%	23.70%	15.84%	5.142
	LOT	**77.12%**	**64.30%**	**49.49%**	**36.34%**	**48.81%**	**35.21%**	**24.37%**	**16.23%**	**1.962**
LipConvnet-25	SOC	75.21%	61.98%	47.93%	34.92%	46.87%	34.09%	23.41%	15.61%	6.087
	LOT	**76.83%**	**64.49%**	**49.80%**	**37.32%**	**48.93%**	**35.23%**	**24.33%**	**16.59%**	**2.387**
LipConvnet-30	SOC	74.23%	60.64%	46.51%	34.08%	46.92%	34.17%	23.21%	15.84%	6.927
	LOT	**77.12%**	**64.36%**	**49.98%**	**37.30%**	**49.18%**	**35.54%**	**24.24%**	**16.48%**	**2.755**
LipConvnet-35	SOC	74.25%	61.30%	47.60%	35.21%	46.88%	33.64%	23.34%	15.73%	7.870
	LOT	**76.91%**	**63.55%**	**49.05%**	**36.19%**	**48.25%**	**34.99%**	**24.13%**	**16.25%**	**3.193**
LipConvnet-40	SOC	72.59%	59.04%	44.92%	32.87%	45.03%	32.57%	22.37%	14.76%	8.668
	LOT	**76.75%**	**63.50%**	**49.07%**	**36.06%**	**48.31%**	**34.59%**	**23.70%**	**15.94%**	**3.595**

6.1 Experiment Setup

Baseline - SOC with CReg loss, HH activation and LLN technique In the evaluation, we will mainly compare with SOC [20], as SOC has been shown to outperform other 1-Lipschitz models in terms of certified robustness. SOC parameterizes an orthogonal layer with $W = \exp(V - V^\intercal)$ where V is the convolution kernel and the transpose and exponential are with respect to the convolution operation. Following their setting, we will focus on the CIFAR-10 and CIFAR-100 datasets and the model architecture varies from LipConvnet-5 to LipConvnet-40.

[21] proposes three techniques to improve SOC networks . First, they observe that adding a Certificate Regularization (CReg) loss can help with the certified performance at larger radius: $\ell_{CReg}(x, y) = -\gamma \text{ReLU}\big((f(x)_y - \max_{i \neq y} f(x)_i)/\sqrt{2}\big)$. Second, they propose a Householder (HH) activation layer, which is a generalization of the standard GroupSort activation. Finally, for tasks with large number of classes (e.g. CIFAR-100), they propose Last Layer Normalization(LLN) to only normalize the final output layer instead of orthogonalize. Since these techniques are agnostic to the type of convolution operation, we will further integrate them with LOT for comparison.

Implementation Details We will train and evaluate the LOT network under supervised and unsupervised scenarios. Following previous works [20], we focus on the CIFAR-10 and CIFAR-100 datasets and the model architecture varies from LipConvnet-5 to LipConvnet-40. In semi-supervised learning, we use the 500K data introduced in [4] as the unlabelled dataset. To train the LOT network, we will train the model for 200 epochs using a momentum SGD optimizer with an initial learning rate 0.1 and decay by 0.1 at the 100-th and 150-th epochs. We use Newton's iteration with 10 steps which we observe is enough for convergence (see Appendix E.4). When CReg loss is applied, we use $\gamma = 0.5$; when HH activation is applied, we use the version of order 1. We add the residual connection with a fixed $\lambda = 0.5$ for LOT; for SOC, we use their original version, as we observe that residual connections even hurt their performance (see discussions in Section 6.3). We show the certified accuracy at radius $\rho \in \{\frac{36}{255}, \frac{72}{255}, \frac{108}{255}\}$. For the evaluation time comparison, we show the runtime taken to do a full pass on the testing set evaluated on an NVIDIA RTX A6000 GPU.

6.2 Supervised Scenario

We show the results of supervised learning on CIFAR-10 and CIFAR-100 in Table 1. We can observe that LOT significantly outperforms baselines on both vanilla accuracy and certified accuracy for different datasets. In particular, we observe that the improvement is more significant for deeper models. We owe it to the reason that our LOT layer has a better expressiveness and unbiased parametrization. "Mean evaluation time" column in Table 1 records the evaluation time per instance averaged between CIFAR-10 and CIFAR-100 models. Since architectures of CIFAR-10 and CIFAR-100 models differ only in the last linear year, the evaluation time is almost the same. We observe that LOT has a better efficiency during the evaluation stage, since LOT pre-calculates Newton's iteration while SOC calculates the exponential in each iteration. A similar conclusion can be observed from results without CReg loss, HH activation and LLN, as shown in Table 6 and 7 in Appendix E.5.

Table 2: Performance comparison of 1-Lipschitz networks with and without residual connections.

Model	Conv. Type	Residual Connection	Vanilla Accuracy	Certified Accuracy at $\rho =$ 36/255	72/255	108/255
LipConvnet-20	SOC	×	76.90%	61.87%	45.79%	31.08%
		✓	76.97%	61.76%	45.46%	30.29%
	LOT	×	77.16%	62.14%	45.78%	30.95%
		✓	**77.86%**	**63.54%**	**47.15%**	**32.12%**

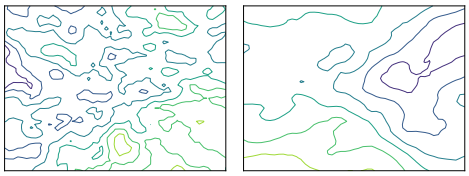
Figure 1: Visualizing the features in the last hidden layer of LipConvnet-20 for SOC (left) and LOT (right). Each image corresponds to one randomly chosen neuron from the last hidden layer and is optimized to maximize the value of the neuron.

6.3 Ablation Studies

In this section, we will perform several ablation studies for LOT. Unless specified, we will use the deep model LipConvnet-20 for evaluation. We use the model without CReg loss and HH activation so that we can see the comparison under the standard CNN setting.

Representation analysis Recent works suggest that adversarially robust models will have a good regularization-free feature visualization [1]. We show the visualization of 4 neurons in the last hidden layer of LipConvNet-20 for both SOC and LOT in Figure 1. More figures are shown in Appendix E.1 The visualization process is as follows: given a chosen neuron, we will start from a random image and take 500 gradient steps to maximize the neuron value with step size 1.0 and decay factor 0.1. We can see that LOT indeed generates more meaningful features. We attribute it to the good expressiveness power of our model. This indicates that our approach indeed leads to better representations.

Error Control of Newton's Iterations To see how our Newton's iteration approximates the inverse square root, we visualize the maximum singular value (σ_{max}) of LOT layers in Appendix E.4. We observe that all the resulting layers have a strictly < 1 Lipschitz. Therefore, we can safely conclude that the overall LOT network is 1-Lipschitz.



(a) LOT, without residual connection.

(b) LOT, with residual connection.

Figure 2: The loss landscape [12] with respect to the parameters of LOT with and without residual connections. The figure is plotted by calculating the loss contour on two randomly chosen directions of parameters.

Effect of residual connections Empirically, we observe that LOT is non-smooth and therefore we need the residual connection to smoothify the model. To verify this phenomenon, we adopt the visualization approach in [12] and show the loss surface in Figure 2. We can see that LOT without residual connection is highly non-smooth, while after adding residual connections the model becomes much smoother. By comparison, we observe that SOC networks with and without residual connection are similar (see Appendix E.2). We hypothesize the reason to be that the exponential parametrization $W = \exp(A)$ enables residual property implicitly, i.e.,

$$\exp(A) \circ X = \underline{X + A \circ X} + \frac{1}{2} A \circ A \circ X + \dots$$

where the first two terms are essentially the format of residual connection. We show the results of models with and without residual connections in Table 2. We can observe that our LOT indeed improves with the residual connection, while the performance remains similar or even worse for SOC. Therefore, in the evaluation, we use the LOT with residual connection and SOC without it.

Robustness against ℓ_∞ Empirical Attack To evaluate the robustness of our model against attacks in other ℓ_p norms, we evaluate the model against standard ℓ_∞ PGD attack[16] with $\epsilon = 8/255$ and 50 steps. We show the results on CIFAR-10 with CReg loss and HH activation in Table 3. We can observe that we still achieve better empirical robustness on the LipConvNet compared with SOC.

Table 3: Empirical robust accuracy of 1-Lipschitz model with CReg loss and HH activation on CIFAR-10 against ℓ_∞-PGD attack at $\epsilon = 8/255$.

Model	Conv.	Vanilla accuracy	Robust accuracy
LipConvnet-5	SOC	75.31%	27.18%
	LOT	**76.34%**	**27.19%**
LipConvnet-10	SOC	76.23%	27.95%
	LOT	**76.50%**	**28.71%**
LipConvnet-15	SOC	76.39%	27.84%
	LOT	**76.86%**	**29.10%**
LipConvnet-20	SOC	76.34%	26.17%
	LOT	**77.12%**	**29.45%**
LipConvnet-25	SOC	75.21%	28.93%
	LOT	**76.83%**	**29.55%**
LipConvnet-30	SOC	74.23%	27.44%
	LOT	**77.12%**	**28.92%**
LipConvnet-35	SOC	74.25%	14.09%
	LOT	**76.91%**	**29.32%**
LipConvnet-40	SOC	72.59%	11.68%
	LOT	**76.75%**	**28.67%**

Table 4: Performance of 1-Lipschitz networks on the CIFAR-10 dataset with semi-supervised training. Results of different architectures are shown in Table 8 in Appendix E.6.

Model	Conv. Type	Setting	Vanilla Accuracy	Certified Accuracy at $\rho =$ 36/255	72/255	108/255
LipConvnet-20	SOC	Supervised	76.34%	62.63%	48.69%	36.04%
		Semi-supervised	70.95%	61.72%	51.78%	42.01%
	LOT	Supervised	**77.12%**	**64.30%**	49.49%	36.34%
		Semi-supervised	71.86%	62.86%	**52.24%**	**42.39%**

6.4 Semi-supervised Learning Scenario

We show the results of semi-supervised learning for one architecture with CReg loss and HH activation in Table 4. The full results are shown in Table 8 and 9 in Appendix E.6. We can observe that, with semi-supervised learning, the vanilla accuracy will drop slightly, which will impact the certified accuracy at a small radius ($\rho = 36/255$). However, the certified accuracy at a large radius will be improved, and the gap is more significant at a larger radius (e.g. $\rho = 108/255$). We improve the previous state-of-the-art certified accuracy by over 6% at $\rho = 108/255$. This indeed shows that semi-supervised learning can help improve certified robustness. We empirically observe that semi-supervised learning does not help much on CIFAR-100. We owe it to the reason that the vanilla accuracy of the teacher model on CIFAR-100 is low (less than 50%). Thus, according to Assumption 5.2 and Theorem 5.2, when c is low, the certified accuracy would be low.

7 Conclusion

In this work, we propose an orthogonal convolution layer LOT and build a 1-Lipschitz convolution network. We show that LOT network outperforms the previous state-of-the-art in certified robustness. We also show that semi-supervised learning can further help with the robustness both theoretically and experimentally.

Acknowledgements

This work is partially supported by the NSF grant No.1910100, NSF CNS No.2046726, C3 AI, and the Alfred P. Sloan Foundation.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 977–988. IEEE, 2022.

[2] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

[3] Dario A Bini, Nicholas J Higham, and Beatrice Meini. Algorithms for the matrix pth root. *Numerical Algorithms*, 39(4):349–378, 2005.

[4] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863. PMLR, 2017.

[6] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[9] Lei Huang, Li Liu, Fan Zhu, Diwen Wan, Zehuan Yuan, Bo Li, and Ling Shao. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6429–6438, 2020.

[10] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[11] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv preprint arXiv:2004.08688*, 2020.

[12] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[13] Linyi Li, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, 22-26 May 2023*. IEEE, 2023.

[14] Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Advances in neural information processing systems*, 32, 2019.

[15] Tsung-Yu Lin and Subhransu Maji. Improved bilinear pooling with cnns. In Gabriel Brostow Tae-Kyun Kim, Stefanos Zafeiriou and Krystian Mikolajczyk, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 117.1–117.12. BMVA Press, September 2017.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[17] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[18] Haifeng Qian and Mark N Wegman. L2-nonexpansive neural networks. *arXiv preprint arXiv:1802.07896*, 2018.

[19] Hanie Sedghi, Vineet Gupta, and Philip M Long. The singular values of convolutional layers. *arXiv preprint arXiv:1805.10408*, 2018.

[20] Sahil Singla and Soheil Feizi. Skew orthogonal convolutions. In *International Conference on Machine Learning*, pages 9756–9766. PMLR, 2021.

[21] Sahil Singla, Surbhi Singla, and Soheil Feizi. Improved deterministic l2 robustness on cifar-10 and cifar-100. In *International Conference on Learning Representations*, 2021.

[22] Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations*, 2021.

[23] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[24] Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11505–11515, 2020.

[25] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *International Conference on Learning Representations*, 2020.

[26] Tan Yu, Jun Li, Yunfeng Cai, and Ping Li. Constructing orthogonal convolutions in an explicit manner. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We point out our limitations in the comparison with previous works in Section 4.1.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss them in Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We state the assumptions in Section 5.
   (b) Did you include complete proofs of all theoretical results? [Yes] We include the complete proof in Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We upload it as supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify them in Section 6.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We tried multiple architectures and showed that the results are consistent across all these different settings in Section 6.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We specify them in Section 6.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the previous state-of-the-art [20].

(b) Did you mention the license of the assets? [N/A]
(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Broader Impact

Adversarial attacks have been investigated a lot and people are worried that the vulnerability of machine learning models may affect their application in real world. This paper, as a potential countermeasure against the adversarial attacks, proposes a type of model architecture that can guarantee the model robustness under certain adversarial attacks. This could help to secure the safe deployment of ML models in real applications, thus promoting the development of various ML systems. In addition, better understanding of the Lipschitz property of machine learning models may help people understand and explain how the models work, which is a key concern when applying these algorithms in practice. On the other hand, the misuse of our technique may also lead to negative impact. For example, if an attacker is fully familiar with this work, he/she may discover some specific attack to fool this model (beyond ℓ_2 attacks which we can certify against). Therefore, in real applications, it is recommended that different defense techniques are applied together to secure the model safety.

B Pseudocode of LOT Layer

We show the detailed pseudocode of our LOT layer in Algorithm 1.

Algorithm 1 LOT layer.

Require: Unconstrained convolution kernel $V \in \mathbb{R}^{c_{out} \times c_{in} \times k \times k}$; Input tensor $X \in \mathbb{R}^{c_{in} \times w \times w}$.

$X^{pad} = \text{zero_pad}(X, (k, k, k, k)) \in \mathbb{R}^{c_{in} \times (w+2k) \times (w+2k)}$.
$V^{pad} = \text{zero_pad}(W, (0, 0, k+w, k+w)) \in \mathbb{R}^{c_{out} \times c_{in} \times (w+2k) \times (w+2k)}$.
// Calculate the Fourier transformation:
for all $i \in \{1, \ldots, c_{in}\}$ **do**
  $\tilde{X}_i = \text{FFT}(X_i^{pad})$.
  for all $j \in \{1, \ldots, c_{out}\}$ **do**
    $\tilde{V}_{j,i} = \text{FFT}(V_{j,i}^{pad})$.
  end for
end for
// Calculate the output on frequency domain:
for all $a, b \in \{1, \ldots, w+2k\}$ **do**
  $\hat{V} = \frac{\tilde{V}_{:,:,a,b}}{\sqrt{||\tilde{V}_{:,:,a,b}\tilde{V}_{:,:,a,b}^{\intercal}||_F}}$ // Rescale $\tilde{V}$.
  Calculate $\tilde{W}_{:,:,a,b} = (\hat{V}\hat{V}^*)^{-\frac{1}{2}}\hat{V}$ with Newton's iteration.
  $\tilde{Y}_{:,a,b} = \tilde{W}_{:,:,a,b}\tilde{X}_{:,a,b}$.
end for
// Get the final output:
for all $i \in \{1, \ldots, c_{out}\}$ **do**
  $Y_i = \text{FFT}^{-1}(\tilde{Y}_i)$.
end for
return $(Y_{:,k:w+k,k:w+k}).real$

C Proof of Theorem 5.1 and Theorem 5.2

C.1 Proof

To prove Theorem 5.1 and Theorem 5.2, we first define the loss on a subset, so that we can divide the loss into different subset losses.

Definition C.1 (Subset loss)**.** Define conditional loss:

$$L_{\mathsf{m}}^{cond}(G, G^*|S) = E_{x \in S}[G^*(x)(1 - G(x)) + (1 - G^*(x))G(x)]$$

to bethe loss function calculated only over the subset $S \subseteq \mathcal{X}$, and define the subset loss:

$$L_{\mathsf{m}}(G, G^*|S) = L_{\mathsf{m}}^{cond}(G, G^*|S) \cdot P(S).$$

A property of the subset loss is that, if $S = S_1 \cup S_2$ where S_1 and S_2 are disjoint, then $L_{\mathsf{m}}(G, G^*|S) = L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G^*|S_2)$.

First, we provide some facts on the relationship between subset loss $L_{\mathsf{m}}(G, G^*|S)$ and $L_{\mathsf{m}}(G, G_{pl}|S)$ when $S \subseteq \mathcal{M}(G_{pl})$ or $S \subseteq \overline{\mathcal{M}(G_{pl})}$.

Fact C.1. *When $S \subseteq \overline{\mathcal{M}(G_{pl})}$, then $L_{\mathsf{m}}(G, G^*|S) = L_{\mathsf{m}}(G, G_{pl}|S)$; when $S \subseteq \mathcal{M}(G_{pl})$, then $L_{\mathsf{m}}(G, G^*|S) + L_{\mathsf{m}}(G, G_{pl}|S) = P(S)$.*

Proof. This is easy to see by noticing that when $x \in \overline{\mathcal{M}(G_{pl})}$, then $G_{pl}(x) = G^*(x)$; when $x \in \mathcal{M}(G_{pl})$, then $G_{pl}(x) \neq G^*(x)$, so $G_{pl}(x) + G^*(x) = 1$. □

Now, we define $S_1 = S_B^{\mathsf{m}}(G) \cap \mathcal{M}(G_{pl})$ and $S_2 = S_B^{\mathsf{m}}(G) \cap \overline{\mathcal{M}(G_{pl})}$. Base on the fact, we will be able to derive the relationship between $L_{\mathsf{m}}(G, G^*|S_1)$ and $L_{\mathsf{m}}(G, G^*|S_2)$:

Lemma C.2. *We have the following relationship between the losses on the sets in which G_{pl} is correct vs. G_{pl} is wrong:*

$$L_{\mathsf{m}}(G, G^*|S_2) + P(S_1) = L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G_{pl}|S_B^{\mathsf{m}}(G))$$

And therefore,

$$L_{\mathsf{m}}(G, G^*|S_2) \leq L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G_{pl}) - \mathit{Err}(G_{pl}) + R_B^{\mathsf{m}}(G)$$

Proof. For the first equation, note that $S_2 \subseteq \overline{\mathcal{M}(G_{pl})}$ and $S_1 \subseteq \mathcal{M}(G_{pl})$, so based on the previous fact, we have:

$$\begin{aligned}
&L_{\mathsf{m}}(G, G^*|S_2) + P(S_1) \\
=&L_{\mathsf{m}}(G, G^*|S_2) + L_{\mathsf{m}}(G, G_{pl}|S_1) + L_{\mathsf{m}}(G, G^*|S_1) \\
=&L_{\mathsf{m}}(G, G_{pl}|S_2) + L_{\mathsf{m}}(G, G_{pl}|S_1) + L_{\mathsf{m}}(G, G^*|S_1) \\
=&L_{\mathsf{m}}(G, G_{pl}|S_B^{\mathsf{m}}(G)) + L_{\mathsf{m}}(G, G^*|S_1)
\end{aligned}$$

For the second inequation, notice $\mathcal{M}(G_{pl}) \backslash \overline{S_B^{\mathsf{m}}(G)} \subseteq S_1$, so $P(S_1) \geq P(\mathcal{M}(G_{pl})) - P(\overline{S_B^{\mathsf{m}}(G)}) = \mathrm{Err}(G_{pl}) - R_B^{\mathsf{m}}(G)$. So:

$$\begin{aligned}
L_{\mathsf{m}}(G, G^*|S_2) =&L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G_{pl}|S_B^{\mathsf{m}}(G)) - P(S_1) \\
\leq&L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G_{pl}) - (\mathrm{Err}(G_{pl}) - R_B^{\mathsf{m}}(G)) \\
=&L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G_{pl}) - \mathrm{Err}(G_{pl}) + R_B(G)
\end{aligned}$$

□

Fact C.3. *Given $\delta \in [0, \frac{1}{c}]$ and $\beta \in (0, \frac{c-1}{2}]$, we can verify that:*

$$\frac{\beta - 1}{c(1-\delta) - 2} \leq \frac{\beta}{c-1}$$

Proof. This can be verified by substituting $\delta = \frac{1}{c}$ into LHS, noticing that LHS is monotonically increasing w.r.t. δ. □

Lemma C.4. *For any $\beta \in (0, \frac{c-1}{2}]$, define $q = \frac{\beta}{c-1}\mathit{Err}(G_{pl})$ and $\alpha = (\beta - 1)\mathit{Err}(G_{pl})$. If G fits the pseudolabels with suffcient accuracy and consistency:*

$$L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G) \leq \mathit{Err}(G_{pl}) + \alpha$$

Then G satisfies the following error bound:

$$\mathit{Err}_{\mathsf{m}}(G) \leq 2(q + R_B^{\mathsf{m}}(G)) + L_{\mathsf{m}}(G, G_{pl}) - \mathit{Err}(G_{pl})$$

The intuition of the proof is as follows. Lemma C.2 provides a relationship between the loss $L_{\mathsf{m}}(G, G^*|S_1)$ and $L_{\mathsf{m}}(G, G^*|S_2)$. On the other hand, the expansion of S_1 is also related with S_2 by $(\mathcal{N}(S_1) \backslash S_1) \cap S_B^{\mathsf{m}}(G) \subseteq S_2$. Note that the expansion $P(\mathcal{N}(S_1)) > c \cdot P(S_1)$, so S_1 cannot be too large or otherwise $\mathcal{N}(S_1) \backslash S_1$ will be too large to be within S_2. We will show that $L_{\mathsf{m}}(G, G^*|S_1) < q$.

Proof. Consider the expansion of S_1, $\mathcal{N}(S_1)$. Since $S_1 \subseteq \mathcal{M}(G_{pl})$ and $P(G_{pl}) < 1/c$, we know that $c \cdot P(S_1) < 1$, so by the assumption of expansion, $P(\mathcal{N}(S_1)) \geq c \cdot P(S_1)$. In addition, notice that $(\mathcal{N}(S_1) \backslash S_1) \cap S_B^{\mathsf{m}}(G) \subseteq S_2$. Therefore, we have:

$$\begin{aligned} L_{\mathsf{m}}(G, G^*|S_2) \geq & L_{\mathsf{m}}(G, G^*|(\mathcal{N}(S_1) \backslash S_1) \cap S_B^{\mathsf{m}}(G)) \\ \geq & L_{\mathsf{m}}(G, G^*|\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G)) - L_{\mathsf{m}}(G, G^*|S_1) \end{aligned}$$

For the first term, we notice that $\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G) \subseteq \mathcal{N}(S_1)$ and $S_1 \subseteq S_B^{\mathsf{m}}(G)$, so the conditional loss satisfies $L(G, G^*|\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G))/P(\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G)) \geq (1-\delta)L(G, G^*|S_1)/P(S_1)$. Therefore,

$$\begin{aligned} L_{\mathsf{m}}(G, G^*|S_2) \geq & L_{\mathsf{m}}(G, G^*|\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G)) - L_{\mathsf{m}}(G, G^*|S_1) \\ \geq & (1-\delta) \cdot \frac{P(\mathcal{N}(S_1) \cap S_B^{\mathsf{m}}(G))}{P(S_1)} \cdot L_{\mathsf{m}}(G, G^*|S_1) - L_{\mathsf{m}}(G, G^*|S_1) \\ \geq & (1-\delta) \cdot \frac{P(\mathcal{N}(S_1)) - P(\overline{S_B^{\mathsf{m}}(G)})}{P(S_1)} \cdot L_{\mathsf{m}}(G, G^*|S_1) - L_{\mathsf{m}}(G, G^*|S_1) \\ \geq & (1-\delta) \cdot \frac{P(\mathcal{N}(S_1))}{P(S_1)} \cdot L_{\mathsf{m}}(G, G^*|S_1) - \frac{L_{\mathsf{m}}(G, G^*|S_1)}{P(S_1)} \cdot P(\overline{S_B^{\mathsf{m}}(G)}) - L_{\mathsf{m}}(G, G^*|S_1) \\ \geq & (1-\delta)c \cdot L(G, G^*|S_1) - P(\overline{S_B^{\mathsf{m}}(G)}) - L_{\mathsf{m}}(G, G^*|S_1) \\ = & (c(1-\delta) - 1)L_{\mathsf{m}}(G, G^*|S_1) - R_B^{\mathsf{m}}(G) \end{aligned}$$

Now, substituting $L(G, G^*|S_2)$ on the LHS with Lemma C.2 and noticing $\mathcal{M}(G_{pl}) \backslash \overline{S_B^{\mathsf{m}}(G)} \subseteq S_1$, with simple transformation we have:

$$\begin{aligned} (c(1-\delta) - 2)L(G, G^*|S_1) \leq & L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) + 2 * R_B^{\mathsf{m}}(G) \\ \leq & \alpha \end{aligned}$$

The last inequality comes from the condition in the lemma. Thus, with Fact C.3, we know $L(G, G^*|S_1) \leq \alpha/((c(1-\delta) - 2)) \leq q$. Now, we can bound the overall error:

$$\begin{aligned} \text{Err}_{\mathsf{m}}(G) = & L_{\mathsf{m}}(G, G^*|S_1) + L_{\mathsf{m}}(G, G^*|S_2) + L_{\mathsf{m}}(G, G^*|\overline{S_B^{\mathsf{m}}(G)}) \\ \leq & q + (q + L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) + R_B^{\mathsf{m}}(G)) + R_B^{\mathsf{m}}(G) \\ \leq & 2(q + R_B^{\mathsf{m}}(G)) + L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) \end{aligned}$$

□

Now, we will prove our main lemma, based on which we will be able to derive Theorem 5.1 and Theorem 5.2.

Lemma C.5. *Suppose Assumption 5.2 holds true. Then we can bound:*

$$Err_{\mathsf{m}}(G) \leq L(G) \triangleq \frac{c+3}{c-1} L_m(G, G_{pl}) + \frac{2c+2}{c-1} R_{\mathcal{B}}^m(G; \delta) - Err(G_{pl}).$$

for any $\delta \in [0, \frac{1}{c}]$.

Proof. First, we consider the case where $L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G) \leq \frac{c-1}{2} \cdot \text{Err}(G_{pl})$. In this case, we can find some $\beta \in (0, \frac{c-1}{2}]$ such that.

$$L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G) = \beta \text{Err}(G_{pl}) = \text{Err}(G_{pl}) + (\beta - 1)\text{Err}(G_{pl})$$

Thus, by lemma C.4, we have:

$$\begin{aligned} \text{Err}_{\mathsf{m}}(G) \leq & 2(\frac{\beta}{c-1}\text{Err}(G_{pl}) + R_B^{\mathsf{m}}(G)) + L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) \\ = & \frac{2}{c-1}\beta \text{Err}(G_{pl}) + 2R_B^{\mathsf{m}}(G) + L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) \\ = & \frac{2}{c-1}(L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G)) + 2R_B^{\mathsf{m}}(G) + L_{\mathsf{m}}(G, G_{pl}) - \text{Err}(G_{pl}) \\ \leq & \frac{c+3}{c-1} L_{\mathsf{m}}(G, G_{pl}) + \frac{2c+2}{c-1} R_B^{\mathsf{m}}(G) - \text{Err}(G_{pl}) \\ = & L(G) \end{aligned}$$

Next, we consider the case where $L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G) > \frac{c-1}{2} \cdot \text{Err}(G_{pl})$. By triangle inequality, we have:

$$\begin{aligned}
\text{Err}_{\mathsf{m}}(G) = L_{\mathsf{m}}(G, G^*) &\leq L_{\mathsf{m}}(G, G_{pl}) + L_{\mathsf{m}}(G_{pl}, G^*) \\
&= L_{\mathsf{m}}(G, G_{pl}) + 2\text{Err}(G_{pl}) - \text{Err}(G_{pl}) \\
&< L_{\mathsf{m}}(G, G_{pl}) + \frac{4}{c-1}(L_{\mathsf{m}}(G, G_{pl}) + 2R_B^{\mathsf{m}}(G)) - \text{Err}(G_{pl}) \\
&= \frac{c+3}{c-1} L_{\mathsf{m}}(G, G_{pl}) + \frac{8}{c-1} R_B^{\mathsf{m}}(G) - \text{Err}(G_{pl}) \\
&\leq \frac{c+3}{c-1} L_{\mathsf{m}}(G, G_{pl}) + \frac{2c+2}{c-1} R_B^{\mathsf{m}}(G) - \text{Err}(G_{pl}) \qquad \text{(using } c > 3\text{)} \\
&= L(G)
\end{aligned}$$

□

Now, we provide the proof of Theorem 5.1 and Theorem 5.2 based on Lemma C.5.

Proof of Theorem 5.1. Since $\hat{G}$ is an optimizer of $L(G)$, we know $\text{Err}_{\mathsf{m}}(\hat{G}) \leq L(\hat{G}) \leq L(G^*)$. Substituting G^* into $L(G)$ gives the bound in the theorem. □

Proof of Theorem 5.2. Note that $\text{CertR}(G) \geq \frac{0.5 - Err(G)}{Lip(G)}$ by its definition. Substituting $Err(G) \leq L(G)$ gives the inequality in the theorem. □

D Error Control of Newton's Method

Recall that given an unconstrained matrix $V \in \mathbb{R}^{n \times n}$, we know that $W = (VV^\intercal)^{-\frac{1}{2}} V$ is orthogonal, i.e., $||W||_2 = 1$, which provides certified robustness for the resulting model. In practice, we use a finite number of Newton's iteration steps to approximate $(VV^\intercal)^{-\frac{1}{2}}$. In this section, we provide the following theorem which rigorously control the spectral norm under finite Newton's iteration steps.

Theorem D.1. *Given a matrix $V \in \mathbb{R}^{n \times n}$ such that $||I - VV^\intercal||_2 < 1$, if we use Newton's iteraton for $k^\star$ steps following Equation* (1) *with initialization $Y_0 = VV^\intercal$ and $Z_0 = I$, then we have*

$$||Z_{k^\star} V||_2 \leq 1 + \frac{||V||_2}{\sqrt{\rho_{\min}(VV^\intercal)}}(1 - \sqrt{1 - ||I - VV^\intercal||_2^{2^{k^\star}}}) \leq 1 + \frac{||V||_2}{\sqrt{\rho_{\min}(VV^\intercal)}} ||I - VV^\intercal||_2^{2^{k^\star}}, \tag{2}$$

where $\rho_{\min}$ is the smallest eigenvalue of the matrix.

Remark. We make sure the condition $||I - VV^\intercal||_2 < 1$ is satisfied by rescaling as discussed in Section 4.1. As the theorem shows, along with the increase of Newton's iteration step $k^\star$, the spectral norm of $Z_{k^\star} V$ approaches 1 where the additional term $||I - VV^\intercal||_2^{2^k}$ decays exponentially. Hence, we can rigorously bound the error in the orthogonalization process caused by finite steps and use the bound to determine how many finite steps are needed. Indeed, in practice, we apply singular value decomposition to the computed matrix $Z_{k^\star} V$, and find its maximum singular value always approaches 1 from the left side, i.e., the actual spectral norm is equal to or smaller than 1. Detail experimental verification is in Appendix E.4.

D.1 Proof of Theorem D.1

For brevity, throughout this section, for $V \in \mathbb{R}^{n \times n}$, we define $A = VV^\intercal$. Note that A is a real symmetric matrix. Then, we recursively define

$$\begin{aligned}
B_0 &= I, \\
B_{k+1} &= \frac{1}{2}\left(3B_k - B_k^3 A\right).
\end{aligned} \tag{3}$$

Before proving the main theorem, we first present the following three lemmas.

Lemma D.1. *For any $k \in \mathbb{N}$, $B_k A = A B_k$ and $B_k A^{\frac{1}{2}} = A^{\frac{1}{2}} B_k$.*

Proof of Lemma D.1. We prove the lemma by induction. Since $B_0 = I$, for $k = 0$ the lemma holds. Suppose that the lemma holds for k, then we have

$$B_{k+1}A = \frac{1}{2}(3B_k - B_k^3 A)A = \frac{1}{2}(3B_k A - B_k^3 A^2) \overset{(*)}{=} \frac{1}{2}(3AB_k - AB_k^3 A) = A \cdot \frac{1}{2}(3B_k - B_k^3 A) = AB_{k+1},$$
$$B_{k+1}A^{\frac{1}{2}} = \frac{1}{2}(3B_k - B_k^3 A)A^{\frac{1}{2}} = \frac{1}{2}(3B_k A^{\frac{1}{2}} - B_k^3 A^{\frac{3}{2}}) \overset{(*)}{=} \frac{1}{2}(3A^{\frac{1}{2}}B_k - A^{\frac{1}{2}}B_k^3 A) = A^{\frac{1}{2}} \cdot \frac{1}{2}(3B_k - B_k^3 A) = A^{\frac{1}{2}}B_{k+1}. \tag{4}$$

In above equations, $(*)$ is due to the induction assumption that $B_k A = AB_k$ or $B_k A^{\frac{1}{2}} = A^{\frac{1}{2}} B_k$. Therefore, the lemma holds with $k+1$ and by induction the lemma holds for any $k \in \mathbb{N}$. □

Lemma D.2. *For any $k \in \mathbb{N}$, $Y_k = B_k A$ and $Z_k = B_k$.*

Proof of Lemma D.2. We prove the lemma by induction. Since $Y_0 = VV^\intercal = A$, $Z_0 = I$, and $B_0 = I$, for $k = 0$ the lemma holds. Suppose that the lemma holds for k, then we have

$$\begin{aligned} Y_{k+1} &= \frac{1}{2}Y_k(3I - Z_k Y_k) = \frac{1}{2}B_k A(3I - B_k^2 A) \overset{(*)}{=} \frac{1}{2}(3B_k - B_k^3 A) \cdot A = B_{k+1}A, \\ Z_{k+1} &= \frac{1}{2}(3I - Z_k Y_k)Z_k = \frac{1}{2}(3I - B_k^2 A)B_k \overset{(*)}{=} \frac{1}{2}(3B_k - B_k^3 A) = B_{k+1}, \end{aligned} \tag{5}$$

where $(*)$ leverages $B_k A = AB_k$ from Lemma D.1. Therefore, by induction the lemma holds for any $k \in \mathbb{N}$. □

Lemma D.3. *When $||I - A||_2 < 1$, for any $k \in \mathbb{N}$, the eigenvalue $\lambda \in \mathbb{R}$ of matrix $A^{\frac{1}{2}}B_k$ is positive.*

Proof of Lemma D.3. We define $C_k = A^{\frac{1}{2}}B_k$, then by leveraging the commutability between B_k and $A/A^{\frac{1}{2}}$ (Lemma D.1) we have the following iteration:

$$\begin{cases} C_0 = A^{\frac{1}{2}}, \\ C_{k+1} = \frac{1}{2}(3C_k - C_k^3). \end{cases} \tag{6}$$

Since $||I - A||_2 < 1$ and A is a real symmetric matrix, any eigenvalue of C_0, denoted by $\lambda_i^{C_0}$, $\in (0, \sqrt{2})$. Denoting the diagonalization of C_0 by $C_0 = P^{-1}D_0 P$ where $D_0 = \mathrm{diag}(\lambda_1^{C_0}, \cdots, \lambda_n^{C_0})$. Then, from the iteration, we have $C_{k+1} = P^{-1}\left(\frac{1}{2}(3D_k - D_k^3)\right)P$ and therefore $\lambda_i^{C_{k+1}} = \lambda_i^{C_k}\left(\frac{3}{2} - \frac{1}{2}(\lambda_i^{C_k})^2\right)$. Define function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) = x(\frac{3}{2} - \frac{1}{2}x^2)$. We find $f'(x) = 0 \Rightarrow x \in \{0, 1\}$. Thus, when $x \in (0, \sqrt{2})$, $f(x) \in (0, 1) \subseteq (0, \sqrt{2})$. Now we apply the induction. When $k = 0$, we have $\lambda_i^{C_k} \in (0, \sqrt{2})$. Suppose $\lambda_i^{C_k} \in (0, \sqrt{2})$, from above result we have $\lambda_i^{C_{k+1}} = f(\lambda_i^{C_k}) \in (0, \sqrt{2})$. Thus, for any $k \in \mathbb{N}$, all eigenvalues of $C_k = A^{\frac{1}{2}}B_k$ are positive. □

Now we are ready to prove the main theorem.

Proof of Theorem D.1. Since $Z_k = B_k$ for any $k \in \mathbb{N}$ acccording to Lemma D.2, we focus on B_k and its expression of iteration (Equation (3)) henceforth. According to [3, Section 6], define $R_k = I - B_k^2 A$, we have $R_{k+1} = \frac{3}{4}R_k^2 + \frac{1}{4}R_k^3$. We have $||I - A||_2 = ||R_0||_2 < 1$. By induction,

$$\begin{aligned} ||R_{k+1}||_2 &\le \frac{3}{4}||R_k||_2^2 + \frac{1}{4}||R_k||_2^3 \\ &\le \frac{3}{4}||R_k||_2^2 + \frac{1}{4}||R_k||_2^2 \quad \text{(by induction condition } ||R_k||_2 \le 1) \\ &= ||R_k||_2^2. \end{aligned} \tag{7}$$

Therefore, $||R_{k^\star}||_2 \le ||I - A||_2^{2^{k^\star}}$, i.e., the eigenvalues of the symmetric matrix $I - B_{k^\star}^2 A$ are in the range $[-||I - A||_2^{2^{k^\star}}, ||I - A||_2^{2^{k^\star}}]$.

Given an eigenvalue $\lambda \in \mathbb{R}$ with eigenvector $x \in \mathbb{R}^n$ of the real symmetric matrix $A^{\frac{1}{2}} B_{k^\star}$, we have

$$\begin{aligned} & A^{\frac{1}{2}} B_{k^\star} x = \lambda x \\ \iff & B_{k^\star}^2 A x = \lambda^2 x \qquad \text{(by Lemma D.1)} \\ \iff & (I - B_{k^\star}^2 A) x = \left(1 - \lambda^2\right) x. \end{aligned} \tag{8}$$

Thus, $(1-\lambda^2)$ is an eigenvalue of matrix $(I - B_{k^\star}^2 A)$. Since $(1-\lambda^2) \in [-||I-A||_2^{2^{k^\star}}, ||I-A||_2^{2^{k^\star}}]$, we get

$$\min\{|\lambda - 1|, |\lambda + 1|\} \leq 1 - \sqrt{1 - ||I - A||_2^{2^{k^\star}}}. \tag{9}$$

Then, according to Lemma D.3, we know that $\lambda > 0$ and hence $\lambda \in \left[\sqrt{1 - ||I-A||_2^{2^{k^\star}}}, 2 - \sqrt{1 - ||I-A||_2^{2^{k^\star}}}\right]$. Now we apply diagonalization to $A^{\frac{1}{2}} B_{k^\star}$:

$$A^{\frac{1}{2}} B_{k^\star} := P^\intercal \Lambda P. \tag{10}$$

As a result,

$$\begin{aligned} & ||B_{k^\star} V||_2 \\ =& ||A^{-\frac{1}{2}} P^\intercal \Lambda P V||_2 \\ =& ||A^{-\frac{1}{2}} P^\intercal (\Lambda - I) P V + A^{-\frac{1}{2}} P^\intercal I P V||_2 \\ \leq& ||A^{-\frac{1}{2}}||_2 \cdot (1 - \sqrt{1 - ||I-A||_2^{2^{k^\star}}}) \cdot ||V||_2 + ||A^{-\frac{1}{2}} V||_2 \\ =& ||A^{-\frac{1}{2}}||_2 \cdot ||V||_2 \cdot (1 - \sqrt{1 - ||I-A||_2^{2^{k^\star}}}) + 1 \end{aligned} \tag{11}$$

where the last equality uses the fact $A^{-\frac{1}{2}} V$ is a orthogonal matrix with spectral norm 1.

Since any eigenvector with eigenvalue λ of $A^{-\frac{1}{2}}$ corresponds to the eigenvector with eigenvalue $(1/\lambda^2)$ of $A = VV^\intercal$, and VV^T as a symmetric real matrix only has real eigenvalues,

$$||A^{-\frac{1}{2}}||_2 = \frac{1}{\sqrt{\rho_{\min}(VV^\intercal)}}. \tag{12}$$

Plug it into Equation (11), we get

$$||Z_{k^\star} V||_2 = ||B_{k^\star} V||_2 = 1 + \frac{||V||_2}{\sqrt{\rho_{\min}(VV^\intercal)}} (1 - \sqrt{1 - ||I-A||_2^{2^{k^\star}}}). \tag{13}$$

Notice that $1 - \sqrt{1-x} \leq x$ for $x = ||I-A||_2^{2^{k^\star}} \in [0, 1]$, we conclude the proof.

□

E Addition Exp Results

E.1 Visualization

We show the representation visualization on 16 neurons for SOC and LOT in Figure 3.

E.2 Residual Connection

We show the loss landscape for both LOT and SOC in Figure 4

E.3 Circular vs. Zero padding

As we discussed in Section 4.1, the default parametrization leads to the convolution result with circular-padding, while we will pre-process the input to get the final result with zero-padding. In Table 5, we show the performance comparison between circular and zero padding. We can see that the performance significantly improves when we do the pre-processing and use the zero-padding instead of default circular-padding.

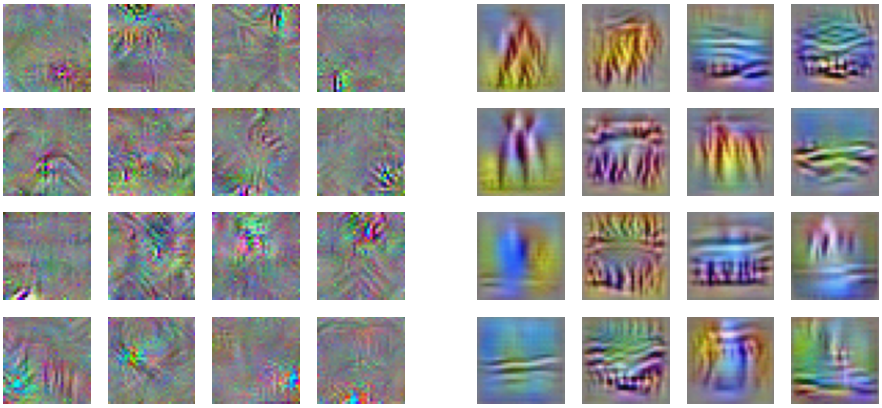
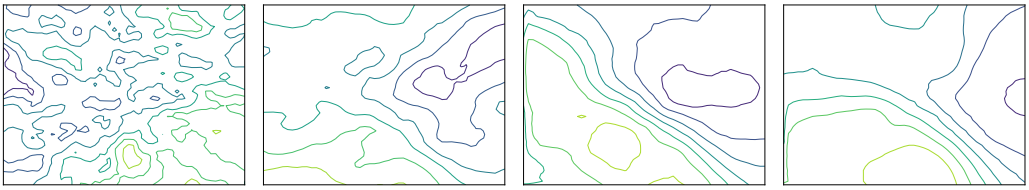


Figure 3: Visualizing the features in the last hidden layer of LipConvnet-20 for SOC (left) and LOT (right). Each image corresponds to one randomly chosen neuron from the last hidden layer and is optimized to maximize the value of the neuron.



(a) LOT, without residual connection. (b) LOT, with residual connection. (c) SOC, without residual connection. (d) SOC, with residual connection.

Figure 4: The loss landscape [12] with respect to the parameters of LOT and SOC network with and without residual connections. The figure is plotted by calculating the loss contour on two randomly chosen directions of parameters. We can observe that the residual connection greatly smoothifies the LOT network, while both SOCs with and without residual connection are smooth.

E.4 Error Control of Newton's Iterations

As we discuss in Section D, we use a finite number of Newton's iteration to approximate the inverse square root of a matrix $Z_{k^*} \approx (VV^\intercal)^{-\frac{1}{2}}$. Theoretically, we have shown that the error of approximated orthogonal matrix $Z_{k^*}V$ will decay exponentially with number of iterations. Furthermore, we observe in practice that during the Newton's iteration, the maximum singular value of Z_kV will always approach 1 from the left side. As an example, we show the maximum singular value (σ_{max}) for all LOT layers in LipConvnet-20 at different steps (k) during the Newton's Iteration in Figure 5. We can see that $\sigma_{max} < 1$ for all the layers, so we can safely assume that the Lipschitz bound is no larger than 1. In addition, $\sigma_{max} > 1 - 10^{-4} = 0.9999$ after $k = 8$ iterations for all the layers, which indicates that the Newton's iteration converges well.

E.5 Supervised Learning without CReg Loss and HH Activation

We show the results of semi-supervised learning under standard setting (without CReg Loss, HH Activation, and LLN on CIFAR-100) in Table 6 and Table 7. We can observe that we still achieve a good performance compared with SOC, and the gap is sometimes larger than with the different optimization techniques.

E.6 Full semi-supervised

We show the results for all architectures with semi-supervised learning in Table 8 (with CReg and HH) and 9 (without CReg and HH). We can see observe that we still achieve a better performance

Table 5: Performance comparison of LOT network with default circular padding and pre-processed zero padding.

Model	**Conv. Type**	**Padding**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
LipConvnet-20	LOT	Circular-pad	76.65%	61.15%	43.83%	28.78%
		Zero-pad	**77.86%**	**63.54%**	**47.15%**	**32.12%**

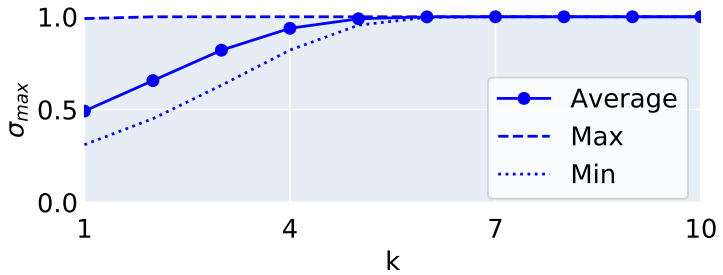


Figure 5: The maximum singular value σ_{max} of $Z_k V$ for each of the LOT layer in LipConvnet-20 during the Newton's iteration. We verify that all the σ_{max}'s are smaller than 1, and after the 8-th iteration, all the σ_{max}'s are larger than 0.9999.

compared with SOC. In addition, these results also have a larger certified accuracy at larger radius compared with the supervised learning setting.

F Results on TinyImageNet

To further validate the performance of LOT, we evaluate it on the TinyImageNet dataset which consists of 100,000 64×64 images in 200 classes. We use the same model architecture and training setting as before and show the comparison between LOT and SOC in Table 10. We can observe that LOT still outperforms the existing SOC approach in most cases. Meanwhile, we observe that all 1-Lipschitz models have a performance drop on the larger dataset compared with vanilla models, and we leave the breakthrough as future work.

G Discussion of Model Architectures

We would like to emphasize that there are two major differences in designing 1-Lipschitz networks and standard networks. First, standard architectures focus a lot on regularizing the network so that it does not overfit (e.g. with dropout), while in 1-Lipschitz networks we do not need strong regularization because the 1-Lipschitz constraint is already strong enough. Second, standard architectures are carefully designed so that the gradient will propagate in a proper manner (e.g. residual connection, BatchNorm), while 1-Lipschitz networks have an intrinsic gradient-norm-preserving property (see [14]). Thus, it may not be appropriate to directly utilize modern architectures in the design of 1-Lipschitz networks. In Table 11, we show the results of a 1-Lipschitz ResNet18 model, where we (1) replace all the conv layers with 1-Lipschitz convolution, (2) replace all the residual connection with average and (3) enforce the variance parameter of BatchNorm layers to be 1 so that the model is 1-Lipschitz. We can observe that LOT still outperforms SOC on both CIFAR-10 and TinyImageNet, while the performance of the ResNet18 architecture is not as good as that of LipConvNet.

Table 6: Certified accuracy of 1-Lipschitz model without CReg loss and HH activation on CIFAR-10 in supervised setting.

Model	**Conv. Type**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255	**Evaluation Time (sec)**
LipConvnet-5	SOC	75.78%	59.18%	42.01%	27.09%	2.117
	LOT	**77.20%**	**61.76%**	**44.45%**	**29.61%**	**1.406**
LipConvnet-10	SOC	76.45%	60.86%	44.15%	29.15%	3.170
	LOT	**77.30%**	**62.54%**	**46.03%**	**30.64%**	**1.420**
LipConvnet-15	SOC	76.68%	61.36%	44.28%	29.66%	3.993
	LOT	**77.34%**	**63.40%**	**46.54%**	**31.75%**	**1.453**
LipConvnet-20	SOC	76.90%	61.87%	45.79%	31.08%	4.752
	LOT	**77.86%**	**63.54%**	**47.15%**	**32.12%**	**1.558**
LipConvnet-25	SOC	75.24%	60.17%	43.55%	28.60%	5.613
	LOT	**77.76%**	**62.77%**	**46.06%**	**31.20%**	**1.834**
LipConvnet-30	SOC	74.51%	59.06%	42.46%	28.05%	6.438
	LOT	**77.34%**	**62.76%**	**46.24%**	**31.07%**	**2.219**
LipConvnet-35	SOC	73.73%	58.50%	41.75%	27.20%	7.400
	LOT	**77.54%**	**62.62%**	**46.28%**	**31.64%**	**2.620**
LipConvnet-40	SOC	71.63%	54.39%	37.92%	24.13%	8.175
	LOT	**77.79%**	**62.69%**	**46.34%**	**31.32%**	**2.910**

Table 7: Certified accuracy of 1-Lipschitz model without CReg loss, HH activation and LLN on CIFAR-100 in supervised setting.

Model	**Conv. Type**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
LipConvnet-5	SOC	42.71%	27.86%	17.45%	9.99%
	LOT	**46.07%**	**31.28%**	**19.86%**	**12.17%**
LipConvnet-10	SOC	43.72%	29.39%	18.56%	11.16%
	LOT	**44.68%**	**30.59%**	**19.69%**	**12.33%**
LipConvnet-15	SOC	42.92%	28.81%	17.93%	10.73%
	LOT	**46.01%**	**32.08%**	**20.72%**	**12.92%**
LipConvnet-20	SOC	43.06%	29.34%	18.66%	11.20%
	LOT	**46.05%**	**32.17%**	**20.81%**	**13.16%**
LipConvnet-25	SOC	43.37%	28.59%	18.18%	10.85%
	LOT	**46.21%**	**31.81%**	**21.01%**	**12.83%**
LipConvnet-30	SOC	42.87%	28.74%	18.47%	11.21%
	LOT	**45.71%**	**32.23%**	**20.87%**	**13.03%**
LipConvnet-35	SOC	42.42%	28.34%	18.10%	10.96%
	LOT	**45.38%**	**31.03%**	**20.02%**	**12.46%**
LipConvnet-40	SOC	41.84%	28.00%	17.40%	10.28%
	LOT	**45.30%**	**30.91%**	**19.97%**	**12.36%**

Table 8: Certified accuracy of 1-Lipschitz model with CReg loss and HH activation on CIFAR-10 in semi-supervised setting.

Model	**Conv. Type**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
LipConvnet-5 + CReg + HH	SOC	69.67%	59.28%	48.02%	38.30%
	LOT	**71.52%**	**61.25%**	**50.66%**	**40.31%**
LipConvnet-10 + CReg + HH	SOC	71.10%	60.81%	50.61%	41.03%
	LOT	**71.82%**	**62.60%**	**51.76%**	**42.31%**
LipConvnet-15 + CReg + HH	SOC	71.15%	61.65%	51.78%	42.53%
	LOT	**71.89%**	**62.37%**	**52.24%**	**42.71%**
LipConvnet-20 + CReg + HH	SOC	70.95%	61.72%	51.78%	42.01%
	LOT	**71.86%**	**62.86%**	**52.24%**	**42.39%**
LipConvnet-25 + CReg + HH	SOC	70.58%	61.40%	51.46%	42.05%
	LOT	**71.64%**	**62.67%**	**52.00%**	**42.34%**
LipConvnet-30 + CReg + HH	SOC	70.37%	60.74%	51.23%	41.81%
	LOT	**72.08%**	**62.84%**	**51.99%**	**41.94%**

Table 9: Certified accuracy of 1-Lipschitz model without CReg loss and HH activation on CIFAR-10 in semi-supervised setting.

Model	**Conv. Type**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
LipConvnet-5	SOC	70.67%	59.14%	45.88%	34.21%
	LOT	**72.86%**	**61.66%**	**48.84%**	**36.67%**
LipConvnet-10	SOC	71.86%	60.15%	48.01%	35.66%
	LOT	**73.57%**	**62.62%**	**50.26%**	**37.82%**
LipConvnet-15	SOC	73.05%	62.11%	50.44%	38.50%
	LOT	**73.74%**	**63.01%**	**50.66%**	**38.73%**
LipConvnet-20	SOC	72.77%	62.51%	50.40%	38.05%
	LOT	**73.64%**	**63.37%**	**51.00%**	**38.52%**
LipConvnet-25	SOC	72.45%	62.03%	50.12%	38.28%
	LOT	**73.62%**	**63.61%**	**51.21%**	**38.77%**
LipConvnet-30	SOC	71.04%	61.06%	49.30%	38.11%
	LOT	**71.71%**	**63.32%**	**51.27%**	**39.42%**

Table 10: Certified accuracy of 1-Lipschitz model without CReg loss and HH activation on TinyImageNet in supervised setting.

Model	**Conv. Type**	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
LipConvnet-5	SOC	30.77%	19.74%	11.60%	6.89%
	LOT	**32.71%**	**21.44%**	**12.96%**	**7.92%**
LipConvnet-10	SOC	31.94%	21.21%	12.80%	**7.79%**
	LOT	**32.31%**	**21.22%**	**12.96%**	7.75%
LipConvnet-15	SOC	32.26%	21.36%	12.94%	7.80%
	LOT	**33.14%**	**22.21%**	**13.34%**	**8.12%**
LipConvnet-20	SOC	32.44%	21.27%	12.90%	7.63%
	LOT	**33.19%**	**22.02%**	**13.42%**	**8.12%**

Table 11: Certified accuracy of 1-Lipschitz ResNet-18 model without CReg loss and HH activation on CIFAR-10/TinyImageNet in supervised setting

Model	**Conv. Type**	CIFAR-10				TinyImageNet			
		Vanilla Accuracy	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255	**Vanilla Accuracy**	**Certified Accuracy at $\rho =$** 36/255	72/255	108/255
ResNet-18	SOC	66.43%	43.00%	23.00%	9.52%	23.26%	11.64%	5.21%	2.17%
	LOT	**68.85%**	**45.46%**	**25.43%**	**11.35%**	**25.09%**	**12.83%**	**5.90%**	**2.62%**