CertViT: Certified Robustness of Pre-Trained Vision Transformers

Kavya Gupta [*1] **Sagar Verma** [*1]

Abstract

Lipschitz bounded neural networks are certifiably robust and have a good trade-off between clean and certified accuracy. Existing Lipschitz bounding methods train from scratch and are limited to moderately sized networks ($<$ 6M parameters). They require a fair amount of hyper-parameter tuning and are computationally prohibitive for large networks like Vision Transformers (5M to 660M parameters). Obtaining certified robustness of transformers is not feasible due to the non-scalability and inflexibility of the current methods. This work presents CertViT, a two-step proximal-projection method to achieve certified robustness from pre-trained weights. The proximal step tries to lower the Lipschitz bound and the projection step tries to maintain the clean accuracy of pre-trained weights. We show that CertViT networks have better certified accuracy than state-of-the-art Lipschitz trained networks. We apply CertViT on several variants of pre-trained vision transformers and show adversarial robustness using standard attacks. Code : https://github.com/sagarverma/transformer-lipschitz

1. Introduction

Deep neural networks (DNNs) are vulnerable to adversarial examples (Szegedy et al., 2013), where the perturbations to the input are constructed deliberately to confuse the classifier. Numerous heuristic solutions for adversarial defenses have been proposed but these solutions are often broken using more carefully crafted stronger attacks (Athalye et al., 2018). This suggests that such methods provide only empirical robustness without any formal guarantees. But for safety and mission-critical scenarios, we need certified defenses with formal robustness guarantees that any norm-bounded adversary will not be able to alter the network predictions. One such line of work providing certified robustness concentrates on bounding the Lipschitz constant of a neural network. Existing methods obtain global and local Lipschitz bounded networks when training from scratch. These methods have been applied to moderately sized networks comprising only convolutional and dense layers. The largest network experimented in (Virmaux & Scaman, 2018) consists of 6M parameters. These methods become untenable for deeper and more complex architectures such as transformers.

Transformers proposed by (Vaswani et al., 2017) for machine translation has become the state-of-the-art method in many NLP tasks. In (Dosovitskiy et al., 2020), the authors have proposed Vision Transformer (ViT) with very few changes to the original transformer. Data-efficient ViT (DeiT) reduces the dependence of ViTs on large-size datasets. Many different Vision Transformer architectures are proposed for various tasks, but the attention layer remains the backbone of most of them. It has to be noted that, for ViT, the number of parameters tends to be very large, so it is computationally expensive to train them from scratch. Instead, researchers use their pre-trained models for fine-tuning desired tasks or datasets. Although there have been many works around the empirical robustness of transformers and several adversarial attacks specifically designed for fooling transformers, the internal properties relating to robustness are not well studied. Moreover, in (Kim et al., 2021), authors have proved that the standard dot product (DP) self-attention is not Lipschitz continuous. They provide a Lipschitz continuous ℓ_2 distance based self-attention as the replacement of DP. They show the efficacy of their transformer-based architecture on language modeling tasks in terms of performance. To the best of our knowledge, the certified robustness of pre-trained transformers has not been investigated. Transformers being large networks efficacy of current certifiable robust training methods is unknown. This paper focuses on obtaining the certified robustness of large pre-trained transformers and is a step in certifying the robustness of parameter heavy models. Our main contributions are summarized as follows:

- We present *CertViT* algorithm utilizing Douglas-Rachford method which consists of two steps: lowering the Lipschitz constant of the network and maintaining good clean accuracy.

[*]Equal contribution [1]Granular AI. Correspondence to: Kavya Gupta <kavya.gupta100@gmail.com>.

2nd AdvML Frontiers workshop at 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.
Copyright 2023 by the author(s).

- We first compare the effectiveness of our proposed method *CertViT* in certifying convolution and linear layer-based models with existing Lipschitz bounding methods.
- For transformers, we adopt a Lipschitz continuous L2 self-attention from pre-trained DP attention networks, making them Lipschitz continuous. We then demonstrate the effectiveness of our method in certifying different transformer architectures. We also discuss the limitations associated with it.

2. On robustness of Transformers

Adversarial robustness of Vision Transformers (ViT) has been studied in (Bhojanapalli et al., 2021; Mahmood et al., 2021; Benz et al., 2021) and they compare different ViTs to their ResNet counterparts. (Mahmood et al., 2021) studied commonly used adversarial attacks and concluded attacks do not transfer readily between transformers and ResNets. They also proposed a new attack SAGA(Self-Attention Gradient attack) for attacking an ensemble of transformers and CNNs. Benz et al. (Benz et al., 2021) did a frequency analysis which suggests that the most robust ViT architectures tend to rely more on low-frequency features compared to CNNs. In (Bhojanapalli et al., 2021) experiments suggest that ViTs are more robust to the removal of any single layer. An interesting observation from the paper suggests a high correlation in the later layers of ViTs, indicating a larger amount of redundancy.

3. Proposed Method

3.1. Lipschitz Continuity

Definition 3.1. *Lipschitz constant of a function f is an upper bound on the ratio between the output and the input variations of a function f. If $L \in [0, +\infty[$ is such that, for every input $x \in \mathbb{R}^{N_0}$ and perturbation $z \in \mathbb{R}^{N_0}$,*

$$\|f(x+z) - f(x)\| \leqslant L\|z\| \tag{1}$$

then L is a Lipschitz constant of f. $\|.\|$ denotes the standard Euclidean norm, but any other norms can be applied. If X is defined as the ϵ-ball at point x, i.e., $X = \{x^{'} | \|x - x^{'}\| \leqslant \epsilon\}$, then L is the local Lipschitz constant of f at x.

The first upper bound on the Lipschitz constant of a neural network was derived by analyzing the effect of each layer independently and considering a product of the resulting spectral norms (Goodfellow et al., 2015). The spectral matrix norm is equal to the maximum singular value of weight matrix. An important assumption is that the activation operators are nonexpansive, i.e., 1-Lipschitz. This assumption is satisfied for all the standard choices of activation operators ReLU, tanh etc.

3.2. Constraining Lipschitz of a layer

Aiming for a tight Lipschitz bound of a network consists of making the Lipschitz constant of individual layers of the network small independently while keeping an overall satisfactory accuracy. We assume that, for a given layer, a training sequence of input/output pairs is available which results from a forward pass performed on the original/pre-trained network for some input dataset of length K. The training sequence is split into J mini-batches of size T so that $K = JT$. The j-th mini-batch with $j \in \{1, \ldots, J\}$ is denoted by $(x_{j,t}, y_{j,t})_{1 \leqslant t \leqslant T}$. To lower the Lipschitz constant of the network, we propose to solve the following constrained optimization problem

Problem 3.2.

$$\underset{W \in C}{\text{minimize}} \;\; L(W) \tag{2}$$

$$C = \Big\{ W \in \mathbb{R}^{N \times M} \mid (\forall j \in \{1, \ldots, J\}) \sum_{t=1}^{T} d^2_{\partial f(y_{j,t})}(W x_{j,t} - y_{j,t}) \leqslant T\eta \Big\}, \tag{3}$$

where L is a Lipschitz measure defined on $\mathbb{R}^{N \times M}$, $\eta \in [0, +\infty[$ is defined accuracy tolerance, and $d_{\partial f(y)}(Wx - y)$ is a suitable accuracy measure for approximated values of the layer parameters W. For more details refer appendix B.

Since, for every $j \in \{1, \ldots, J\}$, the function $W \mapsto \sum_{t=1}^{T} d^2_{\partial f(y_{j,t})}(W x_{j,t} - y_{j,t})$ is continuous and convex, C is a closed and convex subset of $\mathbb{R}^{N \times M}$. In addition, this set is nonempty when there exist $\overline{W} \in \mathbb{R}^{N \times M}$ such that, for every $j \in \{1, \ldots, J\}$ and $t \in \{1, \ldots, T\}$,

$$d^2_{\partial f(y_{j,t})}(\overline{W} x_{j,t} - y_{j,t}) = 0. \tag{4}$$

This condition is satisfied when $\overline{W}$ are the parameters of the pre-trained layer. A standard choice for such a function is the ℓ_1-norm of the matrix elements, $L = \| \cdot \|_1$.

A standard proximal method for solving Problem 3.2 is the Douglas-Rachford algorithm (Lions & Mercier, 1979; Combettes & Pesquet, 2007).This algorithm alternates between a proximal step (relax_β) aiming at lowering the Lipschitz value of the weight matrix and a projection step (proj_C) to maintain original accuracy of the pre-trained network. This algorithm can be written as :

The Douglas-Rachford algorithm uses positive parameters β and $(\lambda_n)_{n \in \mathbb{N}}$. Throughout this article, proj_S denotes the projection onto a nonempty closed convex set S.

Proposition 3.3. *(Combettes & Pesquet, 2007) Assume that Problem 3.2 has a solution and that there exists $\overline{W} \in C$ such $\overline{W}$ is a point in the interior of the domain of h. Assume that $\beta \in \;]0, +\infty[$ and $(\lambda_n)_{n \in \mathbb{N}}$ in $]0, 2[$ is such that*

Algorithm 1 CertViT: Constraining a Layer

Initialize : $\widehat{W}_0 \in \mathbb{R}^{N\times M}$
for $n = 0, 1, \ldots$ **do**
  $W_n = \mathrm{relax}_\beta(\widehat{W}_n)$
  $\widetilde{W}_n = \mathrm{proj}_C(2W_n - \widehat{W}_n, \eta)$
  $\widehat{W}_{n+1} = \widehat{W}_n + \lambda_n(W_n - \widetilde{W}_n)$

$\sum_{n\in\mathbb{N}} \lambda_n(2-\lambda_n) = +\infty$. *Then the sequence* $W_{n\in\mathbb{N}}$ *generated by Algorithm 1 converges to a solution to Problem 3.2.*

The projection step relax_β reduces the magnitude of each parameter or element of the input matrix W_n by β (i.e $(\mathrm{sign}(W_n) * (\mathrm{abs}(W_n) - \beta))$). This is equivalent to making a matrix sparse also known as magnitude pruning (Verma & Pesquet, 2021). In turn, since the convex set C has an intricate form, an explicit expression of proj_C does not exist. Finding an efficient method for computing this projection for large datasets thus constitutes the main challenge in the use of the above Douglas-Rachford strategy, which we will discuss in the next section.

3.3. Maintaining clean accuracy

For every mini-batch index $j \in \{1, \ldots, J\}$, we define the following convex function:

$$(\forall W \in \mathbb{R}^{N\times M}) \quad c_j(W) = \sum_{t=1}^{T} d^2_{\partial f(y_{j,t})}(Wx_{j,t} - y_{j,t}) - T\eta. \quad (5)$$

Note that, for every $t \in \{1, \ldots, T\}$ and $j \in \{1, \ldots, J\}$, function c_j is differentiable and its gradient at $W \in \mathbb{R}^{N\times M}$ is given by

$$\nabla c_j(W) = 2\sum_{t=1}^{T}(Wx_{j,t} - y_{j,t})x_{j,t}^\top. \quad (6)$$

Weight parameters belong to C if and only if it lies in the intersection of the 0-lower level sets of the functions $(c_j)_{1\leqslant j\leqslant J}$. To compute the projection of some $W \in \mathbb{R}^{N\times M}$ onto this intersection, we use Algorithm 2 ($\|\cdot\|_{\mathrm{F}}$ denotes here the Frobenius norm).

This algorithm has the advantage of proceeding in a mini-batch manner. The simplest rule is to use each mini-batch once within J successive iterations of the algorithm so that they correspond to an epoch. The proposed algorithm belongs to the family of block-iterative outer approximation schemes for solving constrained quadratic problems (Combettes, 2003). One of the main features of this algorithm is that it does not require performing any projection onto the 0-lower level sets of the functions c_j, which would be intractable due to their expressions. Instead, these projections are implicitly replaced by subgradient projections, which are much easier to compute in our context.

Algorithm 2 Mini-batch algorithm for computing $\mathrm{proj}_C(W, \eta)$

Initialize : $W_0 = W$
for $n = 0, 1, \ldots$ **do**
  Select a batch of index $j \in \{1, \ldots, J\}$
  if $c_j(W) > 0$ **then**
    $\delta W = \frac{c_j(W)\,\nabla c_j(W)}{\|\nabla c_j(W)\|^2_{\mathrm{F}}}$
    $\pi_n = ((W_0 - W)^\top \delta W)$
    $\mu_n = \|W_0 - W\|^2_{\mathrm{F}}$
    $\nu_n = \|\delta W\|^2_{\mathrm{F}}$
    $\zeta_n = \mu_n\nu_n - \pi_n^2$
    if $\zeta_n = 0$ *and* $\pi_n \geqslant 0$ **then**
      $W = W + \delta W$
    else if $\zeta_n > 0$ *and* $\pi_n\nu_n \geqslant \zeta_n$ **then**
      $W = W_0 + (1 + \frac{\pi_n}{\nu_n})\delta W$
    else
      $W = W + \frac{\nu_n}{\zeta_n}(\pi_n(W_0 - W) - \mu_n\delta W)$
  else
    $W_{n+1} = W_n$

3.4. CertViT for a Network

Algorithm 3 Parallel CertViT for multi-layer network

Input: input sequence $X \in \mathbb{R}^{M\times K}$, β magnitude for soft-thresholding, error tolerance parameter $\eta > 0$, λ is update rate, weight matrices $W^1, \ldots, W^L$
$Y^0 \leftarrow X$
for $l = 1, \ldots, L$ **do**
  $Y^l = R_l(W^l Y^{l-1})$
  $\widehat{W}^l \leftarrow \mathrm{CertViT}(\beta, \eta, \lambda, W^l, Y^l, Y^{l-1})$
$\widehat{W}^1, \ldots, \widehat{W}^L \leftarrow$ fine-tune$(\widehat{W}^1, \ldots, \widehat{W}^L, X, Y)$
Output: $\widehat{W}^1, \ldots, \widehat{W}^L$

Algorithm 3 describes how we make use of CertViT for a transformer. We use a pre-trained transformer and the training sequence to extract layer-wise input-output features. Then we apply CertViT on individual layer l by passing parameter β, η, and λ for magnitude based soft-thresholding, error tolerance, and Douglas-Rachford update rate, respectively. R_l is the activation function just after the layer l. Layer parameters W^l and input-output features (Y^{l-1}, Y^l) are extracted and passed to Algorithm 1. The benefit of

applying CertViT to each layer independently is that we can run CertViT on all the layers of a network in parallel. This reduces the time required to process the whole network and compute resources are optimally utilized. Once all layers have been constrained using CertViT, we finally do a fine-tuning on training set to overcome any accuracy lost during the process. This is similar to retraining after magnitude based pruning.

3.5. Computing Lipschitz constant of Transformers

Transformer model comprises of fully connected, convolutional, and self-attention layers. Though the estimates of the Lipschitz constant of fully connected and convolutional layers are now widely studied in the literature, self-attention layers are still under-explored in terms of Lipschitz bounds and hence their certified robustness is of question. In (Kim et al., 2021) authors investigate the Lipschitz bounds of self-attention layer.They prove the standard DP self-attention used in all the current transformer models is not Lipschitz. They propose an alternative $L2$ Multi headed self-attention ($L2$-MHA).

These estimates are Lipschitz continuous for the condition that query and key weight matrices are shared for each head in multi-headed self attention module. The estimates presented imply transformers have intrinsically high Lipschitz constant due to the self-attention module and naively restricting them to attain lower Lipschitz constant makes them difficult to learn leaving both clean and certified accuracy very low. Since this form of self-attention is Lipschitz continuous we adapt it to learn the ViTs from scratch and also adapt the DP-attention to L_2 attention weights in pre-trained transformers.

The patch embedding layer is used for generating patches and translating the patches to fixed dimension with their positional embeddings. This layer can be constructed using convolutional or dense layers. The output of the patch embedding layer serves as the input to the transformer encoder. The output of this encoder is fed to the MLP layer. The Lipschitz bounds for the weights of both of these types of layers can be estimated using spectral norm via the power iteration method. The MLP layer consists of two dense layers. The first dense layer of MLP is followed by the GeLU activation function. GeLU is not 1-Lipschitz, but Lipschitz continuous with value 1.12 which can be trivially obtained (Appendix C). So we calculate the Lipschitz bound for the MLP layer in the transformer as $1.12\|W\|_s$.

4. Experiment

We perform various experiments to demonstrate our proposed method's effectiveness in making neural network models more robust and provide a good trade-off between clean and certified accuracy. In order to manage our experiments we use Polyaxon[1] on a Kubernetes[2] cluster and use 2 computing nodes with 16 A100 GPUs in each node (40GB VRAM per GPU). Experimental details are available in Appendix E.

4.1. Comparison to Existing Lipschitz Bounding Methods

To concretize the effectiveness of the proposed method in reducing the Lipschitz bounds, we first compare it with existing Lipschitz based methods used in certifying mid-size networks with convolution and dense layers. We compare our results with state-of-the-art of methods such as GloRo (Leino et al., 2021), Local-Lip (Huang et al., 2021), and BCP (Lee et al., 2020). We show these results on limited-sized datasets: MNIST, CIFAR-10, CIFAR-100 and Tiny-ImageNet in Table 1. We used the custom networks used in the certified robustness literature. i.e., 4C3F: MNIST, 4C3F and 6C2F for CIFAR-10, 8C2F for CIFAR-100 and TinyImageNet. We trained our model with the same configurations as provided in Local-Lip (Huang et al., 2021) for fair comparisons. Next, we perform experiments on ViTs on each of the datasets. We train a 6-layer ViT on MNIST and CIFAR-10, 10-layer ViT on CIFAR-100, and 12-layer ViT on TinyImageNet. We replace DP attention with L2 attention to make architecture Lipschitz continuous, as explained in the previous sections. For all experiments of CertViT, we use $\beta = 0.01$, $\eta = 0.1$, and $\lambda = 1.2$. More details about ViT configurations in the Appendix E.

For each experiment, we report accuracy on non-perturbed inputs (clean accuracy), accuracy on adversarial perturbations generated via PGD attack (Madry et al., 2017) (PGD accuracy), and the proportion of inputs that can be correctly classified and certified within ϵ-ball (certified accuracy). Certified accuracy gives a lower bound on the number of correctly classified points that are robust, and PGD accuracy serves as an upper bound on the same quantity. We also report the Lipschitz bounds for the trained models. We test our Lipschitz constrained neural networks to certify robustness against ℓ_2 perturbations within an ϵ-neighborhood of 1.58 for MNIST, $36/255$ for CIFAR-10, CIFAR-100, and TinyImageNet (these are the ℓ_2 norm bounds that have been commonly used in the previous literature.). We also tabulate the computational budget required to train each of these models in terms of FLOPs. More details on the training setup can be found in Appendix D. For different state-of-the-art, we used hyper-parameters mentioned in the respective works detailed in Appendix E for details.

Observations: As mentioned earlier, unconstrained neural network models have very high Lipschitz bounds. We

[1] https://github.com/polyaxon/polyaxon
[2] https://kubernetes.io/

Method (Params)	Model	Clean (%)	PGD (%)	Cert. (%)	Lip.	FLOPs ($\times 10^{13}$)
MNIST ($\epsilon = 1.58$)						
Standard (1973536)	4C3F	**99.3**	45.9	0.0	2.5×10^{3}	6.1
BCP	4C3F	92.4	65.8	44.9	6.9	18.2
GloRo	4C3F	97.0	68.9	50.1	2.3	30.3
Local-Lip	4C3F	96.2	78.2	55.8	**0.7**	17.6
CertViT (Ours)	4C3F	98.2	**82.9**	**61.3**	0.8	**5.5**
Standard (1094528)	ViT	**98.6**	63.4	0.0	1.4×10^{13}	9.3
CertViT (Ours)	ViT	97.8	**64.2**	**54.5**	**1.2**	**8.4**
CIFAR-10 ($\epsilon = 36/255$)						
Standard (2528096)	4C3F	**84.6**	51.1	0.0	2.7×10^{4}	8.2
BCP	4C3F	64.4	59.4	50.0	5.7	16.3
GloRo	4C3F	73.2	66.3	49.0	6.3	49.1
Local-Lip	4C3F	75.7	68.3	67.6	2.5	20.4
CertViT (Ours)	4C3F	81.2	**69.8**	**69.1**	**1.9**	**7.4**
Standard (2360672)	6C2F	**86.4**	50.5	0.0	2.8×10^{5}	17.5
BCP	6C2F	65.7	60.8	51.3	11.35	**35.0**
GloRo	6C2F	70.7	63.8	49.3	9.21	140.1
Local-Lip	6C2F	69.8	**64.3**	54.1	7.89	43.7
CertViT (Ours)	6C2F	82.1	63.2	**57.3**	**6.12**	19.2
Standard (4086912)	ViT	**81.4**	33.7	0.0	8.2×10^{16}	331.5
CertViT (Ours)	ViT	75.1	**42.7**	**33.1**	**9.1**	**358.0**
CIFAR-100 ($\epsilon = 36/255$)						
Standard (2436864)	8C2F	**62.3**	25.3	0.0	8.1×10^{7}	128.6
GloRo	8C2F	29.3	27.7	21.3	10.2	514.3
Local-Lip	8C2F	34.0	31.1	22.9	8.9	360.7
CertViT (Ours)	8C2F	42.4	**35.2**	**25.4**	**7.2**	**77.1**
Standard (4916736)	ViT	**55.3**	15.4	0.0	6.3×10^{14}	397.6
CertViT (Ours)	ViT	46.2	**21.6**	**9.1**	**12.1**	**429.4**
TinyImageNet ($\epsilon = 36/255$)						
Standard (5257984)	8C2F	**39.1**	12.1	0.0	2.9×10^{8}	583.0
GloRo	8C2F	35.5	32.3	22.4	7.7	2331.8
Local-Lip	8C2F	37.4	34.2	**27.4**	**5.9**	1020.2
CertViT (Ours)	8C2F	38.1	**34.9**	26.3	6.1	**349.8**
Standard (9060864)	ViT	**42.4**	10.8	0.0	1.64×10^{28}	2860.1
CertViT (Ours)	ViT	36.3	**13.1**	**2.3**	**13.2**	**3088.9**

Table 1. Comparison of CertViT with state-of-the-art Lipschitz bounding methods. We use Clean, PGD and Certified Accuracy as the performance metrics. We also report the Lipschitz constant values obtained and the FLOPs taken by each method. Note that for CertViT we do not consider the training FLOPs used to obtain the pre-trained network. In case of CIFAR-10 (6C2F) BCP takes less time (FLOPs) compared to standard + CertViT.

Model ($\rightarrow$)	**ViT**							**DeiT**			**Swin**			
	T/16	**S/16**	**S/32**	**B/8**	**B/16**	**B/32**	**L/16**	**T**	**S**	**B**	**T**	**S**	**B**	**L**
DP Params ($\times 10^{6}$)	5.7	22.0	22.9	86.4	80.5	88.2	304.1	5.7	22.0	86.4	28.3	49.5	87.7	196.4
L2 Params ($\times 10^{6}$)	5.2	20.2	21.0	78.8	79.2	81.0	278.4	5.2	20.2	78.8	25.7	44.9	79.9	179.4
L2 FLOPs ($\times 10^{16}$)	6.4	25.0	6.4	392.7	100.6	25.6	350.4	6.4	26.7	396.8	26.3	50.1	90.3	199.9
CertViT FLOPs ($\times 10^{16}$)	7.7	31.2	7.7	471.2	120.7	30.7	420.5	8.2	33.2	473.2	31.5	59.9	108.4	239.8

Table 2. Comparison of parameters of DP-attention and L2-attention transformers and the computation utilized to obtain the L2-attention from DP attention along with FLOPs used to constrain the L2-attention transformer using CertViT. **T/16** means **Tiny** variant with patch size 16. **S**, **B**, and **L** are **Small**, **Base**, and **Large**, respectively.

observe that ViTs with just 1M parameters have very high Lipschitz constant of order 10^{13}, making them unsuitable for certification. The high value can be attributed to the multi-headed self-attention layer in the transformers and residual branches. Our proposed method, CertViT is very successful in lowering the Lipschitz bounds of networks compared to the existing works on networks with convolution and dense layers. CertViT also maintained accuracy close to the pre-trained network. We achieved better PGD and certified accuracy than existing works for all the datasets. CertViT also

Model (↓)	DP Attention		L2 Attention			CertViT on L2 Attention			
	Clean(%)	PGD(%)	Clean(%)	PGD(%)	Lip. ($\times 10^{28}$)	Clean(%)	PGD(%)	Cert.(%)	Lip.
ViT-T/16	63.0	29.1	62.5	28.2	0.2	57.9	32.4	21.7	10.9
ViT-S/16	74.2	48.2	73.7	47.3	1.1	69.2	51.3	0	72.9
ViT-S/32	67.6	40.6	67.5	39.5	2.3	63.1	44.5	0	317.4
ViT-B/8	80.9	59.2	79.9	58.2	13.2	75.5	60.1	0	785.1
ViT-B/16	78.8	57.0	77.8	56.1	13.7	73.2	59.3	0	1720.4
ViT-B/32	75.0	53.7	74.1	52.0	25.8	70.4	56.4	0	8791.7
ViT-L/16	82.6	71.8	81.9	70.2	1.6×10^{10}	77.8	74.8	0	1.7×10^{9}
DeiT-T	72.6	30.4	71.8	29.2	0.5	67.2	31.6	17.3	13.6
DeiT-S	71.5	45.4	70.5	44.8	1.7	64.6	49.3	0	93.7
DeiT-B	75.2	44.6	74.2	43.4	16.1	69.6	48.3	0	4592.0
Swin-T	72.6	21.2	71.2	20.5	7.2	65.8	22.8	0	483.6
Swin-S	75.7	26.0	74.3	25.1	3.8×10^{9}	69.7	28.0	0	2.7×10^{8}
Swin-B	79.8	30.1	78.5	29.1	11.7×10^{10}	73.6	31.1	0	1.9×10^{10}
Swin-L	81.5	28.1	79.8	27.7	23.1×10^{10}	73.9	30.2	0	4.1×10^{10}

Table 3. Comparisons of different ViT variants pre-trained on ImageNet-1K adapted to have L2 attention and constrained using CertViT. Lip. denotes Lipschitz constant of the network.

takes less computational time than other methods, which can be attributed to its applicability to pre-trained weights. Moving to ViTs, CertViT successfully lowered the Lipschitz bounds of pre-trained classifiers by orders of magnitude in all the cases, and we were able to certify the samples. It can also be observed that the performance of existing methods on ViTs is unknown, and the theory usually revolves around non-expansive operators such as ReLU. We still tried these methods on ViTs using a few modifications and found they failed miserably in lowering Lipschitz bounds and hence did not certify any samples. Therefore, we have removed them from our comparisons. In contrast, our method was able to tighten the Lipschitz bound and improve PGD and certified accuracy. In the case of transformers, CertViT requires considerably fewer FLOPs than the existing works implying it requires fewer epochs to converge.

4.2. Constraining Lipschitz of large pre-trained transformers

To show the applicability of our proposed method on large transformers, we apply it on ImageNet-1K pre-trained weights of ViT (tiny, small, base, and large), DeiT (tiny, small, and base), and Swin (tiny, small, base, and large). All weights were obtained from timm[3] library. First, we replace the DP attention in ViT and DeiT networks with L2 attention. Similarly, in Swin, we replace the shifted window and window attention with their L2 variants as described in Appendix F. We then apply our Lipschitz constraining method (CertViT) on these L2-adapted networks. Table 2 shows the number of parameters for DP attention and L2 attention versions of all the large transformers used in our experiments. The Table also reports the computational cycles (FLOPs) used in adapting pre-trained DP attention to L2 attention and the computational cycles used by CertViT when constraining L2 attention pre-trained networks. Table 3 shows results obtained by DP attention, L2 attention, and Lipschitz constrained (CertViT) L2 attention networks. Our proposed method managed to constrain the Lipschitz constant while maintaining acceptable clean and PGD. We were only able to obtain certified accuracy for ViT-T/16 and DeiT-T. This is because we are calculating a very loose upper Lipschitz bound that increases with the depth and parameters of the network. In all other cases, we reduced Lipscthiz, leading to increased PGD accuracy. For PGD accuracy on ImageNet-1K, we randomly select 1000 samples such that each class has one sample from the test set and report the accuracy of the selected sample. The value of ϵ-neighbourhood is kept at $36/255$.

[3] https://github.com/rwightman/pytorch-image-models

5. Conclusion

In this work, we propose an efficient way of providing certified robustness using Lipschitz bounds focusing on vision transformers. With the ever-increasing deployment of neural networks, such formal guarantees are necessary for safety-critical applications. Such Lipschitz constrained training has been missing in the literature for transformers. We have presented a proximal projection step for lowering the Lipschitz bounds while maintaining the accuracy of the classifier. We show the efficacy of CertViT on simple architectures with convolution and fully-connected layers compared to existing techniques. We establish this pipeline for making the transformer layers Lipschitz continuous by using $L2$ attention layers. In the case of pre-trained transformers, we have adapted DP attention to L2 attention layers. We use our proposed method, CertViT, for pre-trained transformers. Results obtained on large transformer variants pre-trained on ImageNet-1K show the efficacy of our proposed method.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Benz, P., Ham, S., Zhang, C., Karjauv, A., and Kweon, I. S. Adversarial robustness comparison of vision transformer and mlp-mixer to cnns. *arXiv*, 2021.

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. Understanding robustness of transformers for image classification. In *ICCV*, 2021.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *ICML*, 2017.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Combettes, P. L. A block-iterative surrogate constraint splitting method for quadratic signal recovery. *IEEE TSP*, 2003.

Combettes, P. L. and Pesquet, J.-C. A Douglas–Rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE JSTSP*, 2007.

Combettes, P. L. and Pesquet, J.-C. Deep neural network structures solving variational inequalities. *SVVA*, 2020a.

Combettes, P. L. and Pesquet, J.-C. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIMODS*, 2020b.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. Efficient and accurate estimation of Lipschitz constants for deep neural networks. In *NeurIPS*, 2019.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 2021.

Gupta, K., Kaakai, F., Pesquet-Popescu, B., and Pesquet, J.-C. Safe design of stable neural networks for fault detection in small uavs. In *SafeComp*, 2022a.

Gupta, K., Kaakai, F., Pesquet-Popescu, B., Pesquet, J.-C., and Malliaros, F. D. Multivariate lipschitz analysis of the stability of neural networks. *Frontiers in Signal Processing*, 2022b.

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. *NeurIPS*, 2021.

Jordan, M. and Dimakis, A. G. Exactly computing the local lipschitz constant of relu networks. *NeurIPS*, 2020.

Kim, H., Papamakarios, G., and Mnih, A. The lipschitz constant of self-attention. In *ICML*, 2021.

Latorre, F., Rolland, P., and Cevher, V. Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv*, 2020.

Lee, S., Lee, J., and Park, S. Lipschitz-certifiable training with a tight outer bound. *NeurIPS*, 2020.

Leino, K., Wang, Z., and Fredrikson, M. Globally-robust neural networks. In *ICML*, 2021.

Lions, P.-L. and Mercier, B. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 1979.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv*, 2017.

Mahmood, K., Mahmood, R., and Van Dijk, M. On the robustness of vision transformers to adversarial examples. In *ICCV*, 2021.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv*, 2018.

Moreau, J.-J. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 1962.

Norouzzadeh, M. S., Lin, W.-Y., Boytsov, L., Rice, L., Zhang, H., Condessa, F., and Kolter, J. Z. Empirical robustification of pre-trained classifiers. In *ICMLW*, 2021.

Salman, H., Sun, M., Yang, G., Kapoor, A., and Kolter, J. Z. Denoised smoothing: A provable defense for pretrained classifiers. *NeurIPS*, 2020.

Serrurier, M., Mamalet, F., González-Sanz, A., Boissin, T., Loubes, J.-M., and del Barrio, E. Achieving robustness in classification using optimal transport with hinge regularization. In *CVPR*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv*, 2013.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *NeurIPS*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *NeurIPS*, 2017.

Verma, S. and Pesquet, J.-C. Sparsifying networks via subdifferential inclusion. In *ICML*, 2021.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *NeurIPS*, 2018.

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for relu networks. In *ICML*, 2018.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. *arXiv*, 2019a.

Zhang, H., Zhang, P., and Hsieh, C.-J. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. In *AAAI*, 2019b.

A. Appendix

A. Related work

In (Virmaux & Scaman, 2018), the problem of computing the exact Lipschitz constant of a differentiable function is shown to be NP-hard; hence most works focus on finding the tightest upper bound. Virmaux et al. (Virmaux & Scaman, 2018) proposed the first generic algorithm (AutoLip) for upper bounding the Lipschitz constant of any differentiable function. RecurJac (Zhang et al., 2019b) is a recursive algorithm analyzing the local Lipschitz constant in a neural network using a bound propagation (Zhang et al., 2019a). FastLip (Weng et al., 2018) is a weaker form of RecurJac. SDP-based Lipschitz Constant estimates were explored in (Fazlyab et al., 2019). Using the non-expansiveness property of the activation function, the optimization problem of estimating a Lipschitz constant is recast as solving a semi-definite positive programming problem, and this estimate is limited to ℓ_2 perturbations. Jordan et al. (Jordan & Dimakis, 2020) compute tight Lipschitz bound using mixed integer programming. In (Latorre et al., 2020), authors proposed a polynomial constrained optimization (LipOpt) based technique for estimating the Lipschitz constant. This estimate can be used for any ℓ_p norm for input and output perturbations, but it is valid for a neural network with a single output. These estimates though tight are only limited to small and pre-trained models since it is difficult to parallelize the optimization solver and make it differentiable for training. A multivariate aspect of understanding Lipschitz properties is discussed in (Gupta et al., 2022b) to analyze the sensitivity of individual inputs.

Utilizing the Lipschitz constant to certify robustness has been studied in several instances in the literature. Neural networks trained without any robustness constraint usually have very large global Lipschitz constant bounds (Szegedy et al., 2013), most existing works train the network with the criterion that promotes small Lipschitz bound. Cisse et al.(Cisse et al., 2017) designed networks with orthogonal weights whose Lipschitz constants are exactly 1. Miyato et al.(Miyato et al., 2018) showed control on the Lipschitz constant using spectral normalization for GANs and in (Serrurier et al., 2021), authors learn 1-Lipschitz networks using hinge regularisation loss for binary classifiers. Lipschitz Margin training (LMT) (Tsuzuku et al., 2018) trains networks that are certifiably robust by constructing a loss on worst logits which are calculated using the global Lipschitz bounds. It adds $\sqrt{2}\epsilon L_{glob}$, where ϵ is the perturbation radius to be certified, to all logits other than that corresponding to the ground-truth class. Box constrained propagation (BCP) (Lee et al., 2020) achieves a tighter outer bound than the global Lipschitz-based outer bound, by considering local information via interval bound (box) propagation. They also compute the worst-case logit based on the intersection of a (global) ball and a (local) box. GloRo (Leino et al., 2021) bounds the upper bounds on the worst margins using the global Lipschitz constant. It constructs a new logit with a newly constructed class called the bottom class and determines if the sample can be certified. Local-Lip (Huang et al., 2021) utilizes the interactions between activation functions (e.g. ReLU, MaxMin) and weight matrices. By eliminating the corresponding rows and columns where the activation function output is constant they guarantee a lower provable local Lipschitz bound than the global Lipschitz bound for the neural network. Gouk et al. (Gouk et al., 2021) proposed a relaxed Lipschitz constrained training method in which neural network optimizers can be adapted with a projection step to lower the Lipschitz constant. This is achieved by normalizing the parameters of a neural network by its spectral norm at every training step. To maintain a good trade-off between robustness and performance (Gupta et al., 2022a) proposes a control loop with a known Lipschitz target.

Adversarial robustness of pre-trained classifiers -(Salman et al., 2020) proposed denoised smoothing to make pre-trained classifiers robust without any retraining or fine-tuning. It prepends a custom-trained Gaussian denoiser to the pre-trained classifier and applies randomized smoothing (Cohen et al., 2019) to the whole network resulting in a certifiably robust classifier. It applies to ℓ_2 bounded perturbations and is not easily extensible to other perturbation models. (Norouzzadeh et al., 2021) offers a higher empirical robust accuracy than denoised smoothing, eliminating the need for multiple queries per sample and reducing the high computational cost of multiple forward passes during inference time. (Norouzzadeh et al., 2021) show the applicability of their approach to ℓ_∞ perturbations as well. Such methods also provide a high degree of empirical robustness but do not provide any formal guarantees of robustness.

B. Linear layers and variational principles

A single layer linear network without any bias can be defined as:

$$y = R(Wx) \tag{7}$$

where $x \in \mathbb{R}^M$ is the input, $y \in \mathbb{R}^N$ the output, $W \in \mathbb{R}^{N \times M}$ is the weight matrix, and R is a nonlinear activation operator from $\mathbb{R}^N$ to $\mathbb{R}^N$. Combettes et al. (Combettes & Pesquet, 2020a;b) recently showed that most neural network activation

functions are proximity operators of convex functions. This shows that there exists a proper lower-semicontinuous convex function f from $\mathbb{R}^N$ to $\mathbb{R} \cup \{+\infty\}$ such that $R = \text{relax}_f$. We recall that f is a proper lower-semicontinuous convex function if the area overs its graph, its epigraph $\{(y, \xi) \in \mathbb{R}^N \times \mathbb{R} \mid f(y) \leqslant \xi\}$, is a nonempty closed convex set. For such a function the proximity operator of f at $z \in \mathbb{R}^N$ (Moreau, 1962) is the unique point defined as

$$\text{relax}_f(z) = \underset{p \in \mathbb{R}^N}{\text{argmin}} \ \frac{1}{2}\|z - p\|^2 + f(p). \tag{8}$$

It follows from standard subdifferential calculus that Eq. (7) can be re-expressed as the following inclusion relation:

$$Wx - y \in \partial f(y), \tag{9}$$

where $\partial f(y)$ is the Moreau subdifferential of f at y defined as

$$\partial f(y) = \{t \in \mathbb{R}^N \mid (\forall z \in \mathbb{R}^N) f(z) \geqslant f(y) + \langle t \mid z - y \rangle\}. \tag{10}$$

The subdifferential constitutes a useful extension of the notion of differential, which is applicable to nonsmooth functions. The set $\partial f(y)$ is closed and convex and, if y satisfies Eq. (7), it is nonempty. The distance to this set of a point $z \in \mathbb{R}^N$ is given by

$$d_{\partial f(y)}(z) = \inf_{t \in \partial f(y)} \|z - t\|. \tag{11}$$

We thus see that the subdifferential inclusion in Eq. (9) is also equivalent to

$$d_{\partial f(y)}(Wx - y) = 0. \tag{12}$$

Therefore, a suitable accuracy measure for approximated values of the layer parameters (W) is $d_{\partial f(y)}(Wx - y)$.

C. GELU activation

The Lipschitz constant of any function $f(x) \colon \mathbb{R} \to \mathbb{R}$, which follows Lipschitz continuity, can be calculated as the absolute maximum value of its derivative $f'(x)$. That is $|f'(x)| \leqslant L \ \forall x$ then L is the Lipschitz constant of f. GELU activation function $(f(x))$ and its derivative $(f'(x))$ is given as

$$f(x) = \frac{x}{2}[1 + erf(x/\sqrt{2})] \tag{13}$$

$$\text{where,} \quad erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2 dt} \tag{14}$$

$$f'(x) = \frac{xe^{-x^2/2}}{\sqrt{2\pi}} + \frac{erf(x/\sqrt{2})}{2} + \frac{1}{2} \tag{15}$$

From the figure 1 we see that the maximum of $|f'(x)|$ is 1.12. So, for the GELU activation function in the transformers, we loosely take the Lipschitz constant as 1.12 and the Lipschitz constant of the MLP layer with the GELU activation function as $1.12\|W\|_s$.

D. Training Setup

D.1. Training Strategies

Figure 2 describes the different training strategies and pipelines we have used in our experiments.

D.2. Experimental Setup

PyTorch is employed to implement all the experiments. We use *timm*[4] implementation of ViT[5] and Swin[6]. We replace Linear and Convolutional layers with our own implementation called LinearX and ConvX, respectively. These layers allow

[4] https://github.com/rwightman/pytorch-image-models

[5] https://github.com/rwightman/pytorch-image-models/blob/0.5.x/timm/models/vision_transformer.py

[6] https://github.com/rwightman/pytorch-image-models/blob/0.5.x/timm/models/swin_transformer.py

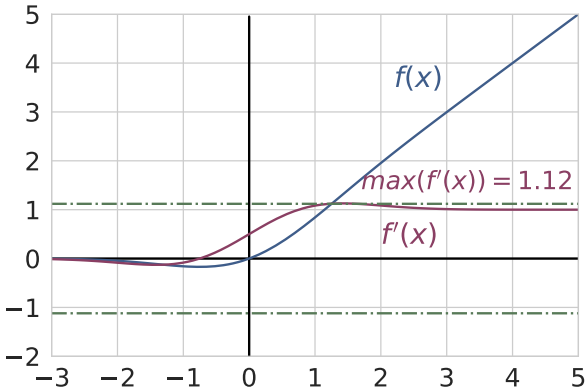

Figure 1. Representation of GELU activation function and its derivative. The maximum absolute value of the derivative is 1.12, as seen from the graph.

us to compute the Lipschitz constant and project weight according to the requirement of CertViT. We also implement L2 attention to replace attention in ViT and Swin. In order to manage our experiments we use Polyaxon[7] on a Kubernetes[8] cluster and use 2 computing nodes with 16 A100 GPUs in each node (40GB VRAM per GPU).

E. Hyper-parameter

E.1. Architectures

Dataset	Input Size	Channels	Patch Size	Layers	Embed Dim	MLP Ratio
MNIST	28	1	7	6	128	4
CIFAR-10	32	3	4	6	192	3
CIFAR-100	32	3	4	12	192	3
TinyImageNet	64	3	4	12	384	3

Table 4. ViT architectures used for different datasets when trained from scratch.

Name	Input Size	Channels	Patch Size	Layers	Embed Dim	MLP Ratio
ViT-T/16	224	3	16	12	192	3
ViT-S/16	224	3	16	12	384	6
ViT-S/32	224	3	32	12	384	6
ViT-B/8	224	3	8	12	768	12
ViT-B/16	224	3	16	12	768	12
ViT-B/32	224	3	32	12	768	12
ViT-L/16	224	3	16	24	1024	16
DeiT-T	224	3	16	12	192	3
DeiT-S	224	3	16	12	384	6
DeiT-B	224	3	16	12	768	12

Table 5. ViT architectures are used for different datasets when ImageNet-1K pre-trained weights are used.

Table 4 shows ViT architecture parameters used to train from scratch on MNIST, CIFAR-10, CIFAR-100 and TinyImageNet datasets. Table 5 shows ViT and DeiT variants parameters for ImageNet-1K pre-trained networks used in the experiment. Swin variants (pre-trained on ImageNet-1k) details is shown in Table 6.

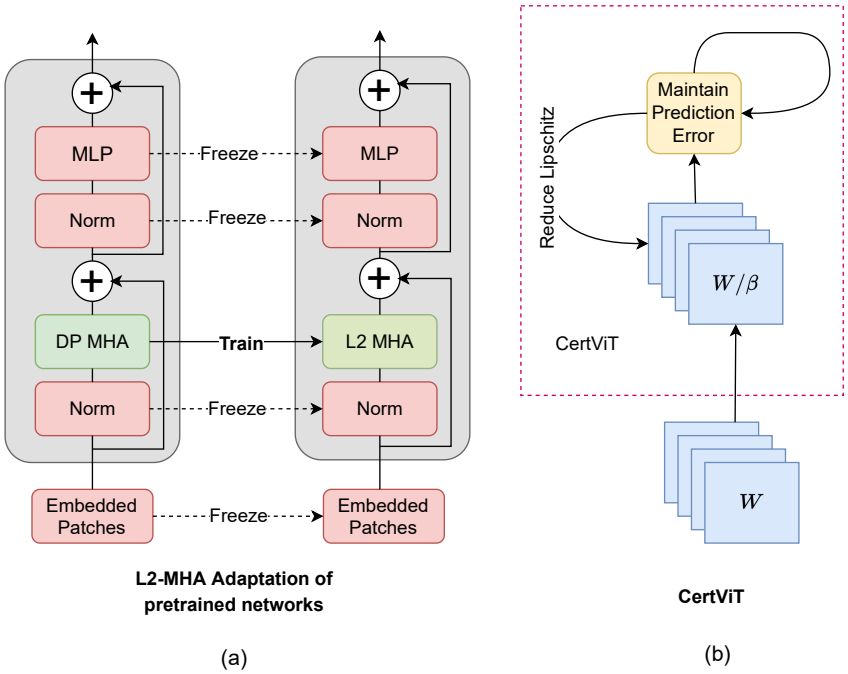

Figure 2. (a) L2 adaptation in a pre-trained network that was trained with DP-MHA. All the layers in the transformer encoder (shown in red) are frozen and the DP-MHA is replaced with L2 MHA and trained for adaptation. (b) Depiction of the proposed approach CertViT, which alternates between maintaining a good accuracy and a small Lipschitz constant.

Name	Input Size	Chnls	Patch Size	Window Size	Layers	Embed Dim	Heads
Swin-T	224	3	4	7	2, 2, 6, 2	96	3, 6, 12, 24
Swin-S	224	3	4	7	2, 2, 18, 2	96	3, 6, 12, 24
Swin-B	224	3	4	7	2, 2, 18, 2	128	4, 8, 16, 32
Swin-L	224	3	4	7	2, 2, 18, 2	192	6, 12, 24, 48

Table 6. Swin architectures used for ImageNet-1K.

Dataset	MNIST	CIFAR-10	CIFAR-100	TinyImageNet
Warm-up	5	15	30	35
Batch Size	512	512	512	512
Epochs	500	800	800	500
ϵ_{train}	1.74	0.16	0.16	0.16
ϵ_{test}	1.58	0.141	0.141	0.141
Optimizer	Adam	Adam	Adam	Adam
Init LR	1e-3	1e-3	1-e3	2.5e-4
LR Decay	5e-6	5e-6	5e-6	5e-7
$\epsilon_{\text{schedule}}$	single	single	single	single
Power Iter	5	5	5	5

Table 7. Hyperparameters used for training GloRo.

E.2. Training from scratch

Table 7 and Table 8.show hyperparameters used in GloRo and LocalLip training of convolutional networks. We use TensorFlow implementation provided here[9]. Table 9 shows the hyperparameters required to train convolutional and ViTs on MNIST, CIFAR-10/100, and TinyImageNet from scratch.

[7] https://github.com/polyaxon/polyaxon
[8] https://kubernetes.io/
[9] https://github.com/klasleino/gloro,https://github.com/yjhuangcd/local-lipschitz

Dataset	MNIST	CIFAR-10	CIFAR-100	TinyImageNet
Warm-up	0	5	20	30
Batch Size	256	256	256	128
Epochs	300	300	800	250
ϵ_{train}	1.58	0.1551	0.1551	0.16
ϵ_{test}	1.58	0.141	0.141	0.141
Init LR	1e-3	1e-3	1-e3	2.5e-4
End LR	5e-6	5e-6	5e-6	5e-7
λ_{sparse}	0.0	0.0	0.0	0.01
λ_{θ}	0.0	0.0	0.0	0.1
LR Decay Epoch	150	200	400	150
$\epsilon_{\text{schedule}}$ **Epochs**	150	200	400	125
Power Iter	5	5	2	1

Table 8. Hyperparameters used for training Local-Lip

Dataset	MNIST	CIFAR-10	CIFAR-100	TinyImage
Batch Size	512	512	512	512
Epochs	100	200	200	200
Optimizer	Adam	Adam	Adam	Adam
Init LR	1e-3	1e-3	1e-3	2.5e-4
LR Decay	5e-6	5e-6	5e-6	5e-7
Scheduler	MultiStep	CosAneal	CosAneal	CosAnneal
Milestones	(50, 60, 70, 80)	NA	NA	NA
$\gamma_{\text{scheduler}}$	0.2	NA	NA	NA
η_{min}	NA	1e-5	1e-5	1e-5
CertViT Epochs	5	5	5	5
$\text{proj}_{\mathbf{C}}$ **Epochs**	2	2	2	2
η	1e-2	1e-2	1e-2	1e-2
β	0.1	0.1	0.1	0.2
λ	1.1	1.2	1.2	1.2

Table 9. Hyperparameters used for training transformers with L2 attention and then using CertViT to constrain the trained network.

E.3. CONSTRAINING PRE-TRAINED NETWORKS

Dataset	Batch Size	Adaptation Epochs	CertViT Epochs	$\text{proj}_{\mathbf{C}}$ Epochs	η	β	λ
ViT-T/16	2048	50	5	2	1e-2	0.1	1.2
ViT-S/16	2048	50	5	2	1e-2	0.1	1.2
ViT-S/32	1024	70	5	2	1e-2	0.1	1.2
ViT-B/8	128	70	6	2	1e-2	0.2	1.3
ViT-B/16	256	50	5	2	1e-2	0.2	1.2
ViT-B/32	256	50	6	2	1e-2	0.2	1.3
ViT-L/16	256	70	6	2	1e-2	0.2	1.3
DeiT-T	2048	50	5	2	1e-2	0.1	1.2
DeiT-S	2048	70	6	2	1e-2	0.2	1.2
DeiT-B	256	70	6	2	1e-2	0.2	1.3
Swin-T	1024	40	5	2	1e-2	0.1	1.2
Swin-S	512	50	6	2	1e-2	0.2	1.2
Swin-B	256	50	6	3	1e-2	0.25	1.3
Swin-L	128	70	7	3	1e-2	0.25	1.3

Table 10. Hyperparameters used for L2 adaptation and CertViT applied on different transformer architectures pre-trained on ImageNet-1K.

Hyper-parameters used in L2 adaptation and constraining using CertViT of transformer variants pre-trained on ImageNet-1K are shown in Table 10.

F. Swin

The Swin transformer builds hierarchical feature maps by merging image patches in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window, unlike in other ViTs where the attention is calculated globally and hence the complexity is quadratic with respect to patch number. Global self-attention in most ViTs variants is generally unaffordable for high-resolution images, while window-based self-attention is scalable. Window-based self-attention lacks connections across windows, limiting its modeling power. They propose a shifted window partitioning that introduces a connection between neighboring non-overlapping windows in consecutive

Swin transformer blocks.

In simple terms, Swin transformers calculate self-attention in non-overlapping local windows and are suitable for applications where the image resolution is high. The windows are arranged to partition the image in a non-overlapping manner evenly. The Swin transformers block is built by replacing the standard multi-head self-attention (MSA) module in the standard transformer block with a module based on shifted windows (W-MSA), with other layers kept the same. This W-MSA layer is still calculated using standard Dot product attention. We replace the Dot product in each W-MSA layer with L2 attention to make the layer Lipschitz continuous and everything else in the architecture remains the same.