A Discussions on EMMA Loss

EMMA may bear similarities to Lipschitz-margin training (LMT) [48], which adds $\sqrt{2}\epsilon K$ to all non-groundtruth logits and K is the global Lipschitz constant of F. However, here are two fundamental differences that distinguish EMMA. First, instead of employing the square-root bound, EMMA directly utilizes the Lipschitz constant for each margin, i.e. K_{yi}, which provides a tighter bound. Second, the robust radius used in EMMA is $\tilde{\epsilon}$ instead of the radius ϵ used in testing. By the definition of EMMA, $0 \leq \tilde{\epsilon} \leq \epsilon$ and $\tilde{\epsilon}$ is 0 if the model is not predicting the input correct. $\tilde{\epsilon}$ grows as the model becomes more robust at the corresponding input. As a result, when the model is not sufficiently robust at the input, EMMA uses $\tilde{\epsilon} < \epsilon$ and imposes a milder robust regularization. On the other hand, LMT always adds $\sqrt{2}\epsilon K$ to all non-groundtruth logits even before the model is capable of predicting the label correct.

The dynamic margin used in EMMA loss is important. If using a fixed margin, the loss function turns out to be:

$$\ell_{\text{fixed}} = -\log \frac{f_y(x)}{\sum_i f_i(x) + \epsilon K_{yi}}$$

The fixed margin loss ℓ_{Fixed} penalties the margin Lipschitz between the ground truth class and all other classes. Therefore, this loss function imposes a stronger regularization on the Lipschitz constant of the model than EMMA loss, and limits the model capacity more. We find that models trained with the fixed margin loss require weaker data augmentation or smaller training ϵ to avoid underfitting. However, this will make the model robust overfitting. The gap between validation clean accuracy and validation VRA is increased for the fixed margin loss and the validation VRA is lower than models trained with the dynamic margin loss, i.e., EMMA loss.

B Implementation Details for Table 1

B.1 Training details

Dataset details The input resolution is 32 for CIFAR10/100, 64 for Tiny-ImageNet and 224 for ImageNet respectively. We apply the following data augmentation to CIFAR datasets: random cropping, RandAugment [8], random horizontal flipping. For Tiny-ImageNet, we find this dataset is easy to overfit and add an extra Cutout [10] augmentation. For data augmentation hyper-parameters, we use the default PyTorch setting.

Platform details Our experiments were conducted on an 8-GPU (Nvidia A100) machine with 64 CPUs (Intel Xeon Gold 6248R). Each experiment on CIFAR10/100 and Tiny-ImageNet takes one GPU and each experiment on ImageNet takes 8 GPUs. Our implementation is based on PyTorch [37].

Training details On the first 3 datasets, all models are trained with the NAdam [12] with the Lookahead optimizer wrapper [60] with a batch size of 256 and a learning rate of 10^{-3} for 800 epochs. We use a cosine learning rate decay [33] with linear warmup [18] in the first 20 epochs. On ImageNet, we only change the batch size to 1024 and training epochs to 400.

During training, we schedule the training ϵ to ramp up from small values and slightly overshoot the test epsilon. Let the total number of epochs be T and the test certification radius be ϵ, we use

$$\epsilon_{\text{train}}(t) = \left(\min(\frac{2t}{T}, 1) \times 1.9 + 0.1\right)\epsilon, \quad \epsilon = 36/255.$$

at epoch t. As a result, $\epsilon_{\text{train}}(t)$ begins at 0.1ϵ and increases linearly to 2ϵ before arriving halfway through the training. Later, ϵ_{train} remains 2ϵ to the end.

B.2 Model architecture details

Model stem is used to convert the input images into feature maps. On CIFAR10/100, we use a convolution with kernel size 5, stride 2, and padding 2, followed by a MinMax activation as the stem. On Tiny ImageNet, we use a convolution with kernel size 7, stride 4, and padding 3, followed by a MinMax activation as the stem. On ImageNet, we follow the ViT-like patching [11] and use a

Table 3: Clean accuracy and VRA performance (%) of a ConvNet and a LiResNet on three datasets with different loss functions

loss	**TRADES**		**EMMA**	
	Clean (%)	VRA (%)	Clean (%)	VRA (%)
CIFAR-10 ($\epsilon = 36/255$, 10 classes)				
ConvNet	71.7	58.8	72.5	59.2
LiResNet	79.6	66.2	80.4	66.3
CIFAR-100 ($\epsilon = 36/255$, 100 classes)				
ConvNet	53.4	34.0	50.6	35.0
LiResNet	57.8	37.3	54.2	37.8
Tiny-ImageNet ($\epsilon = 36/255$, 200 classes)				
ConvNet	42.2	26.6	40.0	27.4
LiResNet	45.8	28.8	43.6	30.0

convolution with kernel size 14, stride 14, and padding 0, followed by a MinMax activation as the stem. Thus the output feature map size from the stem layer is 16×16 for all 4 datasets. The number of filters used in the convolution is equal to the model width W.

Model backbone is used to transform the feature maps. It is a stack of L LiResNet blocks followed by the MinMax activation, i.e., (LiResNet block $\rightarrow$ MinMax) $\times L$. We keep the feature map resolutions and the number of channels constant in the model backbone. We find some tricks in normalization-free residual network studies [40, 59] can improve the performance of our LiResNet as our method is also a normalization-free residual network. Specifically, we add an affine layer β that applies channel-wise learnable multipliers to each channel of the feature map (similar to the affine layer of batch normalization) and a scaler of $1/\sqrt{L}$ to the residual branch where L is the number of blocks:

$$y = x + \frac{1}{\sqrt{L}}\beta\text{Conv}(x)$$

Model neck is used to convert the feature maps into a feature vector. In our implementation, the model neck is a 2 layer network. The first layer is a convolution layer with kernel size 4, stride 4, and padding 0, followed by a MinMax activation. The number of input channels is the model width W and the number of output channels is $2W$. Then we reshape the feature map tensor into a vector. The second layer is a dense layer with output dimension d where $d = 2048$ for the three small datasets (CIFAR10/100 and Tiny-ImageNet) and $d = 4096$ for ImageNet.

Model head is used to make classification predictions. We apply the last layer normalization (LLN) proposed by [42] to the head.

B.3 Metric details

We report the clean accuracy, i.e., the accuracy without verification on non-adversarial inputs and the verified-robust accuracy (VRA), i.e., the fraction of points that are both correctly classified and certified as robust. Our results are averaged over 5 runs for CIFAR10/100 and TinyImageNet and 3 runs for ImageNet.

C Details for Table 2a

In Table 2a, we use an L12W256 configuration, i.e., the backbone has 12 blocks and the number of filters is 256. For ConvNet, the only difference is that the LiResNet block is replaced by a convolution of kernel 3, stride 1, and padding 1. All other settings are the same. Table 3 is a more detailed version of Table 2a with the clean accuracy.

Table 4: Clean accuracy and VRA (%) performance on CIFAR-10/100 with different architectures (L is the number of blocks in the model backbone). We use EMMA loss for Gloro training. $\times$ stands for not converging at the end.

Dataset	L	ConvNet		ResNet		LiResNet	
		Clean(%)	VRA(%)	Clean(%)	VRA(%)	Clean(%)	VRA(%)
CIFAR-10	6	77.9	64.0	74.2	60.3	79.9	65.5
	12	72.5	59.2	74.0	60.0	80.4	66.3
	18	$\times$	$\times$	73.9	60.1	81.0	66.6
CIFAR-100	6	51.8	36.5	48.4	33.5	53.6	37.2
	12	50.6	35.0	48.1	33.5	54.2	37.8
	18	$\times$	$\times$	48.2	33.6	54.3	38.0

Table 5: Clean accuracy and VRA (%) performance of LiResNet of different depths (L is the number of blocks in the model backbone).

L	CIFAR10		CIFAR100		Tiny-ImageNet	
	Clean(%)	VRA(%)	Clean(%)	VRA(%)	Clean(%)	VRA(%)
6	79.9	65.5	53.6	37.2	43.1	29.8
12	80.4	66.3	54.2	37.8	43.6	30.3
18	81.0	66.6	54.3	38.0	43.9	30.6
24	81.2	66.8	55.0	38.2	44.2	30.7
30	81.3	66.9	54.9	38.4	44.2	30.6
36	81.2	66.9	55.0	38.3	44.3	30.4

D Details for Table 2b

In Table 2b, we use the configuration of W256, i.e., the number of channels in the backbone is 256. The only difference between conventional ResNet and LiResNet is the block. The block for conventional ResNet is

$$y = x + \beta \text{Conv}(\text{MinMax}(\text{Conv}(x)))$$

where β is the affine layer. We find use zeros to initialize β works the best for conventional ResNet. The number of input and output channels of the two convolution layers are the same as that of the LiResNet block. Table 4 is a more detailed version of Table 2b with clean accuracy.

E Details for Figure 1

We make LiResNet further deeper and study how network depth influences the performance on CIFAR-10/100 and Tiny-ImageNet. Table 5 shows the clean accuracy and VRA of LiResNet (with EMMA loss) on three datasets. All models use a W256 configuration, i.e., the number of convolutional channels is 256. On CIFAR-10/100, the VRA performance of the LiResNet generally improves with depth. On Tiny-ImageNet, the performance remains with the increase of depth.

Figure 1 compares the VRA performance of LiResNet with some existing method for verification robustness on CIFAR-100 (i.e., the 5th of Table 5). The numbers of these methods are taken from their best-reported configurations. The VRA performance of these methods degrades at certain depths, limiting the maximum model capacity of the methods.

F Number of Classes vs. VRA

Despite the fact that EMMA loss improves the ability of GloRo Nets to handle learning problems with many classes, datasets with a large number of classes still stand out as particularly difficult for certified training. In principle, a data distribution with less classes is not guaranteed to have more

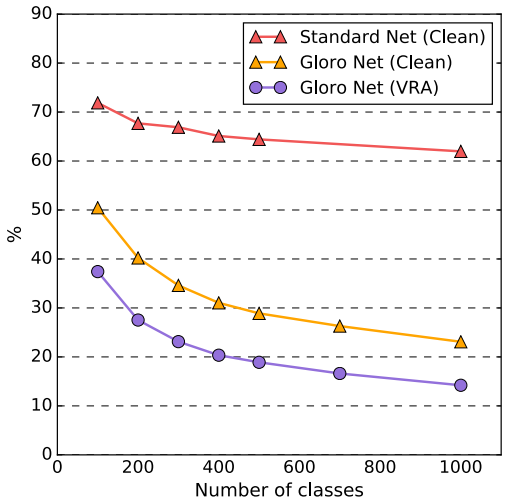

Figure 4: Plot of LiResNet performance on subsets of ImageNet with different number of classes with $\epsilon = 1$

separable features than more classes—indeed, the state-of-the-art clean accuracy for both CIFAR-10 and CIFAR-100 are comfortably in the high 90's despite the large difference in the number of classes. However, training a certifiably robust model with many classes appears more difficult in practice (as observed, e.g., by the large performance gap between CIFAR-10 and CIFAR-100). To test this observation further, we provide an empirical study on various class-subsets of ImageNet to study the relationship between the number of classes and VRA.

We randomly shuffle the 1000 classes of ImageNet and select the first $100 \cdot k$ classes, where $k \in [10]$, to build a series of subsets for training and testing. For each value of k, we train a GloRo LiResNet with EMMA loss ($\epsilon = 1$) and report the clean accuracy and VRA (at $\epsilon = 1$) on the test set. For reference, we also train a standard (i.e., not robust) LiResNet with Cross Entropy and report its clean accuracy on the test set. The final results are shown in Figure 4 with additional details in Appendix F. Compared to the clean accuracy of a standard model, increasing the number of classes leads to a steeper drop in both the VRA and the clean accuracy of the robustly trained models. Specifically, while the performance of the standard model differs only by 10% between a 100-class subset and the full ImageNet, the performance of the GloroNet (both clean accuracy and VRA), drops by 30%.

These results add weight to the observation that, even when mitigated by EMMA loss, large numbers of classes present a particular challenge for certifiably robust learning. This may arise from the need to learn a 2ϵ-margin between all regions with different labels, which becomes progressively more challenging as boundaries between a growing number of classes become increasingly difficult to push off the data manifold.

G Going Wider with LiResNet

We study how network width (i.e., the number of channels in the model backbone) can influence the performance of LiResNet on CIFAR-10, CIFAR-100 and Tiny-ImageNet. Table 6 shows the results. All models use a L12 configuration. Unlike the network depth, increasing the width can stably improve the model performance within a certain range.

H Extra data from DDPM

We use codes from the improved DDPM [35] to train generative models on CIFAR10, CIFAR100 and Tiny-ImageNet. The models are only trained on the training set of each dataset and no external data is used. We use the recommended hyper-parameters from [35] and the models are conditional, i.e., generated samples are with labels. We generate 1 million samples for each dataset.

Table 6: VRA (%) of LiResNet of different widths (W).

W	CIFAR-10	CIFAR-100	Tiny-ImageNet
64	64.6	36.5	28.7
128	65.6	37.5	29.8
256	66.3	37.8	30.0
512	66.9	38.3	30.6

During the training of Gloro Net, we sample 256 samples from the original dataset and 256 samples from the generated data for each batch. Due to the large total number of generated data, we do not need strong data augmentation on the generated data. Compared to the original dataset, we do not use the RandAugment augmentation for the generated data. All other settings are the same for the original dataset and the generated data.

I Extend to Transformers

Transformers are built with self-attention blocks (SA) and feed-forward layers (FFN). Between the blocks, layer normalization layers are applied to make the training of transformers stable.

To make transformers compatible for Lipschitz based robustness certification, layer normalization must be removed from the model since it is not Lipschitz. In fact, no existing Lipschitz based methods for robustness certification use any form of normalization layers. However, normalization is essential for transformers. It will lead to a great performance drop if normalization layers are removed from transformers [56].

The SA operation does not have Lipschitz continuity, thus cannot be applied to Lipschitz based robustness certification. Several studies propose alternatives of SA to make this operation Lipschitz, to name a few: OLSA [53] and L2-MHA [25]. Another idea is to use spatial MLP [46] that performs static "attention" weights.

In terms of FFN, it is a residual block whose residual branch is an MLP: $y = x + W\sigma(Nx)$ where W and N are the weights of two linear layers and σ is the activation. It is Lipschitz and can be applied to Lipschitz based robustness certification. However, according to the theorem of this paper, the Lipschitz estimation of FFN is very loose, thus not a perfect building block for Lipschitz based robustness certification. We propose to use a single layer residual branch, $y = \sigma(x + Wx)$, as the alternative, which is aligned with the motivation of this paper.

We conduct experiments on CIFAR10 and CIFAR100 to see the possibility to use transformers for Lipschitz based robustness certification. For the SA block, we use either OLSA or spatial MLP. For the FFN layer, we use either original FFN, or the single layer FFN, denoted as LiFFN. We use the L12W256 configuration and Table 7 shows the results.

Table 7: VRA (%) of different variants of transformers.

dataset	SA	FFN	VRA(%)
CIFAR10	Spatial MLP	LiFFN	63.3
	Spatial MLP	FFN	62.6
	OLSA	FFN	56.6
CIFAR100	Spatial MLP	LiFFN	36.5
	Spatial MLP	FFN	33.7
	OLSA	FFN	28.4

As shown in Table 7, all variants of transformers performs significantly worse than LiResNet of the same configuration. The combination of "Spatial MLP" and "LiFFN" performs the best among all variants of transformers. This architecture is also most aligned with the motivation of this paper: use linear operations for the weights to obtain tight Lipschitz constant estimation.

To summarise, transformers can be applied to Lipschitz based robustness certification, but the performance is not satisfied. The reasons are that layer normalization layers can not be used and the SA and FFN blocks cannot obtain a tight Lipschitz constant estimation. Some studies apply transformers to non-Lipschitz based methods, such as randomized smoothing [52]. However, stochastic smoothing methods are very slow compared to Lipschitz-based methods.

J Broader Impact

The advancements detailed in this research could have profound societal implications, especially in applications where robust and reliable AI systems are paramount. By introducing the Linear ResNet (LiResNet) architecture and the Efficient Margin MAximization (EMMA) loss function, the authors have significantly increased the robustness of AI models against adversarial attacks. This progress marks a critical step towards ensuring the trustworthiness of AI systems, which is crucial in high-stakes areas such as healthcare, finance, and autonomous vehicles. Furthermore, the ability to scale up fast deterministic robustness guarantees to ImageNet – a dataset more reflective of the complexity and diversity of real-world images –indicates that this approach to robust learning can be applied to practical, real-world applications. This is a significant stride towards making AI systems more secure, reliable, and beneficial for society at large.

Nevertheless, while these advancements are promising, they also emphasize the need for ongoing vigilance and research in the face of increasingly sophisticated adversarial attacks. Ensuring that AI systems are robust and trustworthy will remain a critical task as these technologies continue to permeate society.