# Shrink & Cert: Bi-level Optimization for Certified Robustness

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

## Abstract

In this paper, we advance the concept of shrinking weights to train certifiably robust models from the fresh perspective of gradient-based bi-level optimization. Lack of robustness against adversarial attacks remains a challenge in safety-critical applications. Many attempts have been made in literature which only provide empirical verification of the defenses to certain attacks and can be easily broken. Methods in other lines of work can only develop certified guarantees of the model robustness in limited scenarios and are computationally expensive. We present a weight shrinkage formulation that is computationally inexpensive and can be solved as a simple first-order optimization problem. We show that model trained with our method has lower Lipschitz bounds in each layer, which directly provides formal guarantees on the certified robustness. We demonstrate that our approach, Shrink & Cert (**SaC**) achieves provably robust networks which simultaneously give excellent standard and robust accuracy. We demonstrate the success of our approach on CIFAR-10 and ImageNet datasets and compare them with existing robust training techniques. Code : https://github.com/sagarverma/Shrink-and-Cert

## 1. Introduction

Machine learning models are used as solutions to many problems, but they can be easily fooled by perturbations such as adversarial attacks on the input (Szegedy et al., 2013; Papernot et al., 2016), rendering them impractical for safety and security-critical application. Hence, evaluating the robustness and training of robust deep neural networks is a central point for researchers. Heuristic adversarial defenses and verification methods are often broken by more carefully crafted stronger attacks (Athalye et al., 2018). Hence these methods only provide empirical robustness without any formal guarantees. On the other hand, formal robustness methods verify the neural network behaviors by mathematically deriving the prediction bound of a neural network with respect to certain inputs and thus evaluate the certified robustness of the neural network. Formal guarantees are of much importance in critical applications, where we need guarantees that any norm-bounded adversary will not be able to alter the prediction of the network and is thus certifiably robust.

In this paper, we are concerned about the certified robustness of neural networks under $\ell_p$-norm perturbations on the input. The vulnerability of neural networks is linked to their Lipschitz properties, and the models with small Lipschitz constants are more robust (Cisse et al., 2017; Fazlyab et al., 2019; Virmaux & Scaman, 2018) and provide deterministic certificates of robustness (Tsuzuku et al., 2018). Computing the exact Lipschitz constants of CNNs is NP-hard owing to the non-linear activation functions. Existing approaches look to regularize the norms of individual weight layers. These approaches of norm regularization are motivated by the idea that reducing norms of individual layers can reduce the global Lipschitz constant, and reducing the global Lipschitz constant can ensure smaller local Lipschitz constants and thus improve robustness. In (Cisse et al., 2017), authors proposed Parseval network where the $\ell_2$ norms of linear and convolutional layers are constrained to be orthogonal. Parseval network only slightly improves adversarial robustness in most cases and even reduces robustness in some cases. (Miyato et al., 2018) showed control on the Lipschitz constant using spectral normalization for GANs. In (Anil et al., 2019), authors combined GroupSort, which is a gradient norm-preserving activation function, with norm-constrained weight regularization to enforce tight Lipschitz bounds in dense networks while maintaining the expressiveness of the models. In (Li et al., 2019), authors further extended GroupSort to convolution layers by proposing Block Convolution Orthogonal Parameterization (BCOP), which restricts the linear transformation matrix of a convolutional kernel to be orthogonal and thus its $\ell_2$ norm is bounded by one. However, the robustness performances are still not as good as other certification methods, and none of these methods can provide good certified results for $\ell_\infty$ robustness. Contrary to the existing methods, authors of (Liang & Huang, 2021)

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

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

argue that large $\ell_2$ and $\ell_\infty$ norms of convolutional layers are bad for adversarial robustness. In this work, we show the advantages of shrinking the norms of weight layers not only helps the adversarial robustness but also provides certified robustness with guarantees. To this effect we formulate our training strategy as a bi-level optimization problem. To the best of our knowledge, this is the first work linking robustness guarantees with bi-level optimization.

Our contributions are summarized as follows :

- We rethink the normalization of weight layers as a bi-level optimization problem. To train certifiably robust models, we formulate the upper-level objective as training loss minimization and the lower-level objective as regularization of the norms of the weight layers.
- Our method, Shrink & Cert (SaC), can be solved as a first-order optimization problem, which is computationally inexpensive and can be used for larger models. Therefore overcoming a bottleneck in existing state-of-the art.
- We show our method successfully reduces the Lipschitz bounds of each of the individual layers and hence provides deterministic guarantees on the robustness.
- We show results on CIFAR-10 and Imagenet datasets. Compared to state-of-the-art methods, our method gives better certified accuracy under both $\ell_2$ and $\ell_\infty$ perturbations.

## 2. Related work

### 2.1. Lipschitz bounding approaches

In (Miyato et al., 2018), authors reshape the 4D convolutional kernel into a 2D matrix and use power iterations to compute the $\ell_2$ norm of the weight matrix. Although this method can improve the image quality produced by WGAN, the norm of the reshaped convolutional kernel does not reflect the true norm of the kernel. Based on the observation that the result of power iterations can be computed through gradient back-propagation, (Virmaux & Scaman, 2018) proposed AutoGrad to compute the $\ell_2$ norm. (Sedghi et al., 2018) theoretically analyzed the circulant patterns in the unrolled convolutional kernel, based on which they discovered a new approach to compute the singular values of the kernels. Using the computed spectrum of convolution, they proposed singular value clipping, a regularization method that projects a convolution onto the set of convolutions with bounded $\ell_2$ norms.

Lipschitz based methods (Szegedy et al., 2013; Leino et al., 2021) compute a global Lipschitz constant with respect to $\ell_2$ norm by multiplying the spectral norm of all weight matrices where the spectral norm can be computed by power iteration algorithm to obtain tight Lipschitz bounds. The global Lipschitz constant is usually loose but can be efficiently regularized during training. Global Lipschitz constant computation is efficient and scalable to models on bigger datasets, (Leino et al., 2021). Global Lipschitz bound can be improved by finer-grained analysis on convolutional layers (Lee et al., 2020), by computing local Lipschitz bound (Fazlyab et al., 2019) or by combining with IBP (Lee et al., 2020). Recently, (Gupta & Verma, 2023) proposed reducing the global Lipschitz certificates in case of transformer layers. Currently, such Lipschitz based certification methods can certify robustness against only $\ell_2$ adversary. A multivariate aspect of understanding Lipschitz properties is discussed in (Gupta et al., 2022b) to analyze the sensitivity of individual inputs.

Another thread of research proposes specific layer structures in the network and regulates the Lipschitz constant for these layer structures. There are different designs of orthogonal convolutional layers (Trockman & Kolter, 2021; Li et al., 2019). They usually use parameterization or transformation to explicitly construct trainable convolutional layers, which are orthogonal and thus Lipschitz constant of 1. To maintain a good trade-off between robustness and performance (Gupta et al., 2022a) proposes a control loop with a known Lipschitz target. When the Lipschitz constant of the network is small, Lipschitz based certification can provide efficient robustness guarantees. However, again these approaches are just restricted to $\ell_2$ adversary. Recently, (Zhang et al., 2021a) proposed a novel activation function called $\ell_\infty$ neuron and is 1- Lipschitz with respect to $\ell_\infty$ norm. This design enables general Lipschitz based verification to certify robustness against $\ell_\infty$ adversary. Combined with effective training (Zhang et al., 2021b), this approach can certify state-of-the-art $\ell_\infty$ certified robustness.

### 2.2. Smoothed Classifiers

(Cohen et al., 2019) authors introduced randomized smoothing, which considers Gaussian smoothing and derives a tight $\ell_2$ robustness radius based on the Neyman-Pearson lemma. Against $\ell_1$ adversary, (Levine & Feizi, 2021) and (Teng et al., 2020) consider Laplacian smoothing and derive $\ell_1$ robustness radius. Against $\ell_\infty$ adversary, (Yang et al., 2020) empirically show and theoretically justify that it yields the highest $\ell_\infty$ certified radius by using Gaussian smoothing and transforming Neyman-Pearson-based $\ell_2$ robustness radius to $\ell_\infty$ radius: $r \to r/\sqrt{d}$ where d is the input dimension. Certifying robustness against $\ell_\infty$ for high-dimensional input is proven to be intrinsically challenging (Yang et al., 2020; Blum et al., 2020).

### 2.3. Gradient based bi-level optimization

Bi-level Optimization (BLO) is a unified hierarchical learning framework where the objective and variables of the

upper-level problem depend on the lower-level problems. The nested structure of the BLO problem in its generic form is difficult to solve since it is hard to characterize the influence of the lower-level optimization on the upper-level problem. The efficient gradient descent methods provide a promising solution to the complicated bi-level optimization problem and thus are widely adopted in many deep learning research work. In practice, BLO has been successfully applied to optimize hyper-parameters (Chen et al., 2019a), meta-learning (Rajeswaran et al., 2019; Zhong et al., 2022), data poisoning attack design (Huang et al., 2020), and reinforcement learning (Chen et al., 2019b). In (Zuo et al., 2021), authors formulated the adversarial training problem using a Stackelberg Game, a special bi-level optimization case. In (Zhang et al., 2022) proposed Fast-BAT, authors formulated min-max optimization (MMO) problem of adversarial training by using bi-level optimization. Although effective in robust training, the method provides defenses against attacks such as PGD and does not provide any theoretical guarantees on the robustness. In our work, we utilize BLO for training certifiable robust models with guarantees.

## 3. Training with Shrink & Cert (SaC)

Bi-level optimization is an optimization problem involving two optimization tasks, where the lower-level task is nested inside the upper-level task. More precisely, it has the following generic form:

$$\begin{aligned} \underset{u \in U}{\text{minimize}} \quad & f(u, v^*(u)) \\ s.t. \quad & v^*(u) = \underset{v \in V}{\text{argmin}} \; g(v, u) \end{aligned} \tag{1}$$

where $U$ and $V$ are the feasible sets for the variables $u$ and $v$, respectively; $f()$ and $g()$ are the upper and the lower-level objective functions, respectively. The lower-level task in equation 3 builds the relationship between the lower and upper-level variables.

We now formulate the problem of shrinking model weights through the lens of BLO and develop **SaC**. Our algorithm can be visualized as having two objectives. The first objective is determining a shrinking matrix ($s$), and the second is training the shrinking weight matrix to achieve good accuracy. The shrinking matrix $s$ is controlled by hyperparameter $\alpha$, determining the maximum extent to which the weight can be shrunk. This leads to the shrinking matrix $s \in \mathcal{M}$ , where $\mathcal{M} = \{s | s \in [0, \alpha]^n\}$, where $\alpha \in (0, 1)$ and the model weight variable $\theta \in R^n$, where n denotes the total number of model parameters. The shrunk model $s \odot \theta$ will have smaller values of the weights, leading to lower Lipschitz bounds of the network and providing theoretical guarantees on the robustness. We also allow the shrinking matrix $s$ to have zeros, leading to sparsity in the matrix. This will further ensure lowered norms of the weight matrices. To this end, we interpret the shrinking task and the model retraining task as two optimization levels, where the former is formulated as an upper-level optimization problem and relies on optimizing the lower-level retraining task.

To make the above intuition precise, we use the upper-level problem to model the training loss minimization, while the lower-level problem to shrink the weight values in each layer and consider the following BLO problem:

$$\begin{aligned} \underset{s \in \mathcal{M}}{\text{minimize}} \quad & \mathcal{L}(s \odot \theta^*(s), \mathcal{D}^{val}) \\ s.t. \quad & \theta^*(s) = \underset{\theta \in \mathbb{R}^n}{\text{argmin}} \; \mathcal{L}(s \odot \theta, \mathcal{D}^{train}) \end{aligned} \tag{2}$$

where $\mathcal{L}$ denotes the training loss of the network, $s$ and $\theta$ are the upper-level and lower-level optimization variables respectively, $\theta^*(s)$ signifies the lower-level solution obtained by minimizing the objective function given the shrinking matrix $s$. We can easily solve SaC as the first-order alternating optimization. $\mathcal{D}^{train}$ and $\mathcal{D}^{train}$ denote different data batches to train the upper-level and lower-level tasks mimicking meta-learning formulations.

This bi-level optimization formulation optimizes the coupling between the two tasks through the implicit gradient (IG)-based optimization module. Implicit gradient refers to the gradient of the lower-level solution $\theta^*(s)$ with respect to (w.r.t.) the upper-level variable $s$, and its derivation calls the implicit function theory (Gould et al., 2016). Also, following the Hessian-free assumption (Zhang et al., 2022; Finn et al., 2017), the implicit gradient can be obtained using the first-order derivatives. Upper-level and lower-level tasks are solved alternatively, using stochastic project gradient descent in the implicit gradient descent direction for the upper-level task to obtain $s$ for the shrinking task, interlaced with stochastic gradient descent for the lower-level task to obtain model parameters $\theta$. We summarize the iterative SaC algorithm as follows:

**Lower-level** solution to update $\theta^{(t)}$ : Applying SGD to the lower-level problem at iteration $t$, given $s^{(t-1)}$ and $\theta^{(t-1)}$ with randomly selected batch.

$$\theta^{(t)} := \theta^{(t-1)} - \gamma_1 [s^{(t-1)} \odot \nabla \mathcal{L}(s^{(t-1)} \odot \theta^{(t-1)}, \mathcal{D}^{train})] \tag{3}$$

**Upper-level** solution to update $s^{(t)}$: Applying Stochastic projected gradient descent to the upper-level problem along implicit gradient descent direction at iteration $t$, given $s^{(t-1)}$

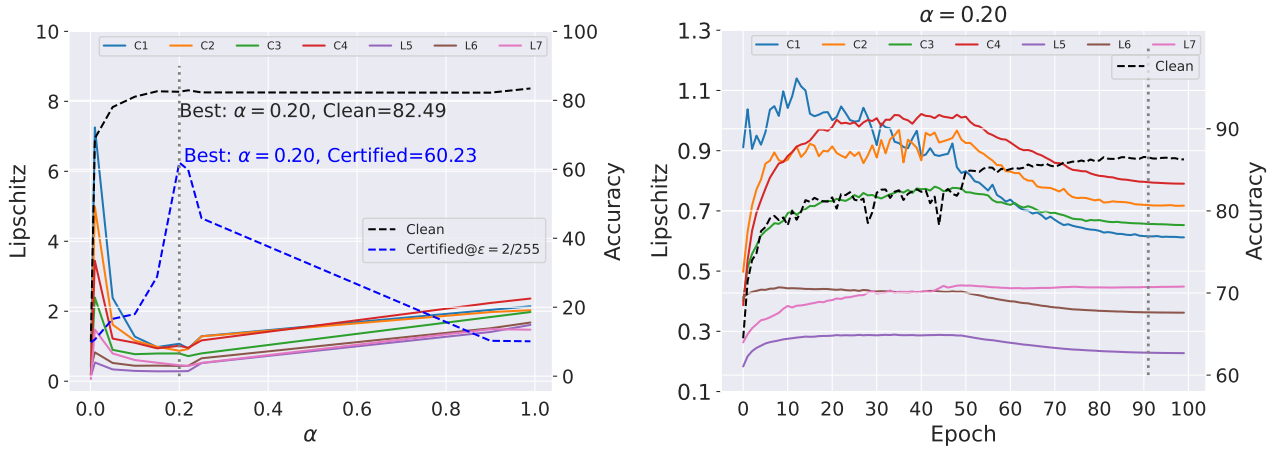


*Figure 1.* 4C3F model trained on CIFAR-10. **Left:** Effect of hyperparameter $\alpha$ on Lipschitz bounds for different layers in CNN. We obtain an optimal value of $\alpha = 0.20$ with clean accuracy = 82.5% and certified accuracy = 60.23%. We can verify that each layer's Lipschitz bounds are close to 1. **Right:** Variation of Lipschitz bounds with each epoch in SaC training. We see an overall decreasing trend of lowered Lipschitz bounds for each layer. Also, note that the Lipschitz bound of each layer is less than 1. **Note:** C1-C4 represents the first four convolutional layers, and L5-L7 represent the last three linear layer of 4C3F.

and updated $\theta^{(t)}$ with randomly selected batch different from the lower-level step.

$$s^{(t)} := \mathcal{P}_s[s^{(t-1)} - \gamma_2 \theta^{(t)} \odot \nabla \mathcal{L}(s^{(t-1)} \odot \theta^{(t)}, \mathcal{D}^{val})] \tag{4}$$

Where $\mathcal{P}_s$ denotes the Euclidean projection onto the feasible set $\mathcal{M}$ and $\gamma_1$, $\gamma_2$ denotes the learning rates for lower and upper-level tasks, respectively.

**Train from scratch** by starting with a random initialization of $\theta$ and $s$.

## 4. Experiments

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. We perform experiments on ResNet-110 trained on CIFAR-10 and ResNet-50 trained on ImageNet for $\ell_2$ perturbations, and we use the 4C3F network from the prior works for comparisons for $\ell_\infty$ perturbations. We use a single node with 8 A100 GPUs for all our experiments. Although more experiments can be performed with larger networks, this set of experiments is sufficient to demonstrate our method's effectiveness and was approachable within our computing budget.

**Training details:** For a fair comparison, we follow the same training details used in (Cohen et al., 2019) and (Salman et al., 2019). We consider three different networks for each model configuration obtained by varying the noise level $\sigma \in 0.25, 0.5, 1.0$. During inference, we apply randomized smoothing with the same $\sigma$ used in training. To train 4C3F and ResNet-110 on CIFAR-10 using SaC, we use a batch size of 128, SGD optimizer, and a single A100 GPU. For the lower-level of SaC, we use a learning rate of 0.1 and *step* schedule for the shrinkage matrix and a learning rate of $1e-4$ for the implicit gradient learning. For the upper-level of SaC, we use a learning rate of 0.1 and *cosine* schedule with weight decay of $1e-4$ and momentum of 0.9. We train both networks for 100 epochs. We tried the following values for $\alpha$ in the given order (0.99, 0.01, 0.90, 0.05, 0.50, 0.1, 0.25, 0.15, 0.22, 0.20) and found $\alpha = 0.20$ to be giving best certified network. Training details for ResNet-110 are the same as 4C3F. In this case, we search for the best $\alpha$ in the neighborhood of 0.20; specifically, we tried following in the given order (0.30, 0.10, 0.25, 0.15, 0.23, 0.16, 0.21, 0.18) and found $\alpha = 0.16$ giving the best result. We train ResNet-50 on ImageNet using SaC for 100 epochs with a batch size of 200 per GPU, SGD optimizer, and 8 A100 GPUs. For the lower-level of SaC, we use a learning rate of 0.1, a step schedule for the shrinkage matrix, and a learning rate of $1e-4$ for the implicit gradient learning. For the upper-level of SaC, we use a learning rate of 0.01 and a cosine schedule with weight decay of $1e-4$ and momentum of 0.9. We try the following values for $\alpha$ in the given order (0.50, 0.10, 0.45, 0.15, 0.35, 0.20, 0.30, 0.25, 0.16, 0.19, 0.17, 0.18) and found $\alpha = 0.17$ to be giving best certified network.

**Evaluation of $\ell_2$:** To evaluate certified robustness for a given classifier $f$, we aim to compute the certified test accuracy at radius $r$, which is defined by the fraction of the test dataset that $f$ can certify the robustness of radius $r$ with respect to the certifiable lower bound. We use (Cohen et al.,

| Dataset | $\epsilon$ | Method | Clean | PGD | Certified |
|---|---|---|---|---|---|
| CIFAR-10 | 2/255 | CAP(Wong et al., 2018) | 0.68 | - | 0.54 |
| | | CROWN-IBP (Zhang et al., 2019) | 0.71 | 0.60 | 0.54 |
| | | IBP (Shi et al., 2021) | 0.67 | - | 0.53 |
| | | COLT (Balunović & Vechev, 2020) | 0.78 | - | 0.60 |
| | | Randomized Smoothing (Cohen et al., 2019) | 0.79 | - | **0.63** |
| | | $l_{\text{inf}}$-distance Net (Zhang et al., 2021b) | 0.61 | 0.54 | 0.54 |
| | | Ours | **0.83** | **0.60** | 0.61 |
| | 8/255 | IBP*(Gowal et al., 2018) | 0.51 | 0.31 | 0.29 |
| | | CROWN-IBP(Zhang et al., 2019) | 0.48 | 0.38 | 0.35 |
| | | COLT (Balunović & Vechev, 2020) | 0.52 | - | 0.27 |
| | | Randomized Smoothing (Cohen et al., 2019) | 0.53 | - | 0.24 |
| | | $l_{\text{inf}}$-distance Net (Zhang et al., 2021b) | 0.57 | 0.37 | 0.33 |
| | | Ours | **0.59** | **0.38** | **0.36** |
| | 16/255 | IBP (Shi et al., 2021) | 0.37 | - | 0.24 |
| | | $l_{\text{inf}}$-distance Net(Zhang et al., 2021b) | 0.48 | 0.33 | 0.29 |
| | | Ours | **0.54** | **0.37** | **0.34** |

*Table 1.* Comparison of clean, PGD and certified accuracy under $\ell_\infty$ perturbation for 4C3F trained on CIFAR-10.

| $\sigma$ | Method | $\epsilon$ | | | | |
|---|---|---|---|---|---|---|
| | | **0.00** | **0.25** | **0.50** | **1.0** | **1.5** |
| **0.25** | Randomized Smoothing (Cohen et al., 2019) | 0.75 | 0.60 | 0.43 | 0.0 | 0.0 |
| | SmoothAdv(Salman et al., 2019) | 0.74 | 0.67 | 0.57 | 0.0 | 0.0 |
| | MACER (Zhai et al., 2020) | 0.81 | **0.71** | 0.58 | 0.0 | 0.0 |
| | Consistency (Jeong & Shin, 2020) | 0.72 | 0.65 | 0.57 | 0.0 | 0.0 |
| | Ours | **0.83** | 0.69 | **0.59** | **0.02** | 0.0 |
| **0.50** | Randomized Smoothing (Cohen et al., 2019) | 0.65 | 0.54 | 0.41 | 0.23 | 0.09 |
| | SmoothAdv(Salman et al., 2019) | 0.50 | 0.46 | 0.44 | 0.38 | 0.29 |
| | MACER (Zhai et al., 2020) | 0.66 | 0.60 | 0.53 | 0.38 | 0.19 |
| | Consistency(Jeong & Shin, 2020) | 0.52 | 0.48 | 0.45 | 0.38 | 0.30 |
| | Ours | **0.69** | **0.61** | **0.54** | **0.40** | **0.32** |
| **1.00** | Randomized Smoothing (Cohen et al., 2019) | 0.47 | 0.39 | 0.34 | 0.21 | 0.14 |
| | SmoothAdv(Salman et al., 2019) | 0.45 | 0.41 | 0.38 | 0.32 | 0.25 |
| | MACER (Zhai et al., 2020) | 0.45 | 0.41 | 0.38 | 0.32 | 0.25 |
| | Consistency (Jeong & Shin, 2020) | 0.42 | 0.39 | 0.36 | 0.31 | 0.25 |
| | Ours | **0.49** | **0.43** | **0.39** | **0.33** | **0.27** |

*Table 2.* Comparison of Certified test accuracy of ResNet-110 trained on CIFAR-10 with state-of-the art methods. Every model is certified with $\sigma$ used for its training. We set our result bold-faced whenever the value improves the baseline. Each column is an $\ell_2$ radius.

2019) implementation, which returns a safe lower bound of certified radius over random $n$ samples with probability at least $1-\eta$ or abstains the certification. The approximate certified test accuracy is defined as the fraction of the test dataset which classifies correctly with a radius larger than $r$ without abstaining. In our experiments, we use $n = 1e6$, $n_0 = 100$, and $\alpha = 1e-3$ as in the previous works.

**Evaluation metrics $\ell_\infty$:** For each method in the table, we report the clean test accuracy without perturbation (denoted as Clean), the robust test accuracy under PGD attack (denoted as PGD), and the certified robust test accuracy (denoted as Certified). The number of iterations of the PGD attack is set to a large number of 100. We then calculate the certified robust accuracy based on the output margin.

| $\sigma$ | Method | $\epsilon$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **0.00** | **0.50** | **1.0** | **1.5** | **2.0** | **2.5** | **3.0** |
| **0.25** | Randomized Smoothing (Cohen et al., 2019) | 0.67 | 0.49 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | SmoothAdv (Salman et al., 2019) | 0.65 | 0.56 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | MACER (Zhai et al., 2020) | 0.68 | 0.57 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | Consistency (Jeong & Shin, 2020) | 0.68 | 0.58 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | Ours | **0.70** | **0.61** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **0.50** | Randomized Smoothing (Cohen et al., 2019) | 0.57 | 0.46 | 0.37 | 0.29 | 0.0 | 0.0 | 0.0 |
| | SmoothAdv (Salman et al., 2019) | 0.54 | 0.49 | 0.43 | 0.37 | 0.0 | 0.0 | 0.0 |
| | MACER (Zhai et al., 2020) | 0.64 | 0.53 | 0.43 | 0.31 | 0.0 | 0.0 | 0.0 |
| | Consistency (Jeong & Shin, 2020) | 0.55 | 0.40 | 0.44 | 0.34 | 0.0 | 0.0 | 0.0 |
| | Ours | **0.66** | **0.55** | **0.45** | **0.38** | 0.0 | 0.0 | 0.0 |
| **1.00** | Randomized Smoothing (Cohen et al., 2019) | 0.44 | 0.38 | 0.33 | 0.26 | 0.19 | 0.15 | 0.12 |
| | SmoothAdv (Salman et al., 2019) | 0.40 | 0.37 | 0.34 | 0.30 | 0.27 | 0.25 | 0.20 |
| | MACER (Zhai et al., 2020) | 0.48 | 0.43 | 0.36 | 0.27 | 0.25 | 0.18 | 0.14 |
| | Consistency (Jeong & Shin, 2020) | 0.41 | 0.37 | 0.32 | 0.28 | 0.24 | 0.21 | 0.17 |
| | Ours | **0.50** | **0.44** | **0.38** | **0.31** | **0.28** | **0.26** | **0.21** |

*Table 3.* Comparison of Certified test accuracy of ResNet-50 trained on ImageNet with state-of-the-art methods. Every model is certified with $\sigma$ used for its training. We set our result bold-faced whenever the value improves the baseline. Each column is an $\ell_2$ radius.

### 4.1. Observations

**Effect of** $\alpha$: Finding the right shrinking parameters $\alpha$ is very important and it is the key to getting an optimally certified network. Choosing a very small value for $\alpha$ can make it harder for the model even to achieve good clean accuracy. In the left sub-figure of figure 1, we can observe that small values of $\alpha$ between 0.05 to 0.15 lead to high Lipschitz and low clean accuracy. Specifically in case of $\alpha <= 0.05$ network is not able to train. The high Lipschitz value is due to the instability in training caused by the shrinking step trying to be very restrictive and the training step trying to over-compensate the small parameter values. A large $\alpha$ can lead to good clean accuracy but lower certified accuracy, as seen for $\alpha >= 0.9$. This is due to Lipschitz of all the layers being closer to 1. The best way to obtain the optimal value for $\alpha$ is to do a binary search by alternatively trying low and high values of $\alpha$.

The right sub-figure of figure 1 shows the variation of Lipschitz bounds with the number of epochs. For the 4C3F model trained on CIFAR-10, we attain 0.20 as an optimal value of $\alpha$. We observe at this value that the model has lower Lipschitz bounds. In this experiment, SaC finds a good balance at epoch 91 when clean accuracy is high enough and Lipschitz of all layers is low enough to provide a good certified network.

For robustness under $\ell_2$ perturbations, from the table 2 and 3, we see that our training strategy was successful in attaining better certified test accuracy than the state-of-the-art methods for different $\epsilon$ radius. From table 2, we also see that with $\sigma = 0.25$, we achieved approximate certified accuracy of 20% at $\ell_2$ radius of 1 where all the state-of-the-art methods fail. However, for higher values of $\ell_2$ radius, our methods also fail to certify the model.

For robustness under $\ell_\infty$ perturbations, from table 1, we observe better PGD and certified accuracy with our method. We also note an increase in the clean accuracy of the model. This supports the claim that the optimization algorithm trains to achieve better trade-offs with clean accuracy.

## 5. Conclusion

In this work, we make the first connection between building certifiably robust models and bi-level optimization. Bi-level optimization has gained a lot of attention in recent literature. Recent works suggest the concrete importance of BLO in adversarial training strategies. We formulate our algorithm to regularize the weight matrix norms by alternating between learning a shrinking matrix (upper-level) and retraining with shrunk matrix to recover accuracy (lower-level). Our algorithm is easily solvable using gradient descent methods and is computationally inexpensive. We show that the shrunk weight matrix obtained has lower Lipschitz bounds with maintained clean accuracy. We show that models obtained with our strategy have better certified and PGD accuracy than state-of-the-art methods. Our strategy was able to certify against both $\ell_2$ and $\ell_\infty$ perturbations better than state-of-the-art methods.

## References


Anil, C., Lucas, J., and Grosse, R. Sorting out lipschitz function approximation. In *ICML*, 2019.

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

Balunović, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *ICLR*, 2020.

Blum, A., Dick, T., Manoj, N., and Zhang, H. Random smoothing might be unable to certify $\ell_\infty$ robustness for high-dimensional images. *JMLR*, 2020.

Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J.-G., and Wang, Y. $\lambda$opt: Learn to regularize recommender models in finer levels. In *ACM SIGKDD*, 2019a.

Chen, Z., Liu, D., Wu, X., and Xu, X. Research on distributed renewable energy transaction decision-making based on multi-agent bilevel cooperative reinforcement learning. 2019b.

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.

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.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv*, 2016.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv*, 2018.

Gupta, K. and Verma, S. Certvit: Certified robustness of pre-trained vision transformers. *arXiv*, 2023.

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, W. R., Geiping, J., Fowl, L., Taylor, G., and Goldstein, T. Metapoison: Practical general-purpose clean-label data poisoning. *NeurIPS*, 2020.

Jeong, J. and Shin, J. Consistency regularization for certified robustness of smoothed classifiers. *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.

Levine, A. J. and Feizi, S. Improved, deterministic smoothing for l_1 certified robustness. In *ICML*, 2021.

Li, Q., Haque, S., Anil, C., Lucas, J., Grosse, R. B., and Jacobsen, J.-H. Preventing gradient attenuation in lipschitz constrained convolutional networks. *NeurIPS*, 2019.

Liang, Y. and Huang, D. Large norms of cnn layers do not hurt adversarial robustness. In *AAAI*, 2021.

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

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In *IEEE ESSP*, 2016.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. *NeurIPS*, 2019.

Salman, H., Yang, G., Li, J., Zhang, P., Zhang, H., Razenshteyn, I. P., and Bubeck, S. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*, 2019.

Sedghi, H., Gupta, V., and Long, P. M. The singular values of convolutional layers. *arXiv*, 2018.

Shi, Z., Wang, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Fast certified robust training with short warmup. *NeurIPS*, 2021.

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

Teng, J., Lee, G.-H., and Yuan, Y. $ell_1$ adversarial robustness certificates: a randomized smoothing approach. 2020.

Trockman, A. and Kolter, J. Z. Orthogonalizing convolutional layers with the cayley transform. *arXiv*, 2021.

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

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

Wong, E., Schmidt, F. R., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *NeurIPS*, 2018.

Yang, G., Duan, T., Hu, J. E., Salman, H., Razenshteyn, I., and Li, J. Randomized smoothing of all shapes and sizes. In *ICML*, 2020.

Zhai, R., Dan, C., He, D., Zhang, H., Gong, B., Ravikumar, P., Hsieh, C.-J., and Wang, L. Macer: Attack-free and scalable robust training via maximizing certified radius. *arXiv*, 2020.

Zhang, B., Cai, T., Lu, Z., He, D., and Wang, L. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *ICML*, 2021a.

Zhang, B., Jiang, D., He, D., and Wang, L. Boosting the certified robustness of l-infinity distance nets. *arXiv*, 2021b.

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*, 2019.

Zhang, Y., Zhang, G., Khanduri, P., Hong, M., Chang, S., and Liu, S. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *ICML*, 2022.

Zhong, T., Chi, Z., Gu, L., Wang, Y., Yu, Y., and Tang, J. Meta-dmoe: Adapting to domain shift by meta-distillation from mixture-of-experts. *arXiv*, 2022.

Zuo, S., Liang, C., Jiang, H., Liu, X., He, P., Gao, J., Chen, W., and Zhao, T. Adversarial regularization as stackelberg game: An unrolled optimization approach. *arXiv*, 2021.