ROBUST RANDOMIZED SMOOTHING VIA TWO COST-EFFECTIVE APPROACHES

Linbo Liu[1] **Nghia T. Hoang**[2] **Lam M. Nguyen**[3] **Tsui-Wei Weng**[1]
[1]UC San Diego, [2]Vietnam National University, [3]IBM Research

ABSTRACT

Randomized smoothing has recently attracted attentions in the field of adversarial robustness to provide provable robustness guarantees on smoothed neural network classifiers. However, existing works show that *vanilla* randomized smoothing usually does not provide good robustness performance and often requires (re)training techniques on the base classifier in order to boost the robustness of the resulting smoothed classifier. In this work, we propose two cost-effective approaches to boost the robustness of randomized smoothing while preserving its standard performance. In the first approach, we propose a new robust training method **AdvMacer** that combines adversarial training and maximizing robustness certificate for randomized smoothing. We show that **AdvMacer** can improve the robustness performance of randomized smoothing classifiers compared to SOTA baselines. The second approach introduces a post-processing method named **EsbRS** which greatly improves the robustness certificate based on model ensembles. We explore different aspects of model ensembles that has not been studied by prior works and propose a mixed design strategy to further improve robustness of the ensemble.

1 INTRODUCTION

The existence of adversarial examples of deep neural networks (DNNs) (Szegedy et al., 2014; Goodfellow et al., 2015) has raised serious concerns to deploy DNNs in real-world systems, especially in the safety critical applications such as self-driving cars and aircraft control systems. Thus, many research efforts have been devoted into developing effective defenses methods to safeguard DNNs. One of the most promising direction is known as *certified defense* via *randomized smoothing*, where the word *certified* means that the defense methods have provable theoretical guarantee as opposed to easily broken heuristic defenses (Athalye et al., 2018), and *randomized smoothing* is a popular technique that allows scalable certified defenses for state-of-the-art DNNs against adversarial examples.

Randomized smoothing is recently proposed by Lecuyer et al. (2019); Li et al. (2018); Cohen et al. (2019) and has achieved state-of-the-art robustness guarantees. Given any classifier f, denoted as a *base classifier*, randomized smoothing predicts the most-likely class on the randomly perturbed input x with Gaussian noises. Following this new prediction rule, randomized smoothing acts like an operator on the original *base classifier* and produce a new *smoothed* classifier which is equipped with provable robustness guarantees under various ℓ_p norm threat models (Lecuyer et al., 2019; Li et al., 2019; Cohen et al., 2019).

Unfortunately, without specially-designed training techniques, the robustness certificate of vanilla smoothed classifiers is usually very weak (Cohen et al., 2019). Thus there are a few recent works (Salman et al., 2019; Zhai et al., 2019) proposed to design specialized robust training methods to improve the robustness certificate of the smoothed classifier. In Salman et al. (2019), the authors propose an adversarial training method called SmoothAdv, which is similar to the PGD training (Madry et al., 2018) but on the *smoothed* classifier. On the other hand, Zhai et al. (2019) propose MACER, whose training objective involves a term to maximize the robustness certificate directly. However, SmoothAdv often requires heavy tuning on a number of hyper-parameters for different noise level σ which could be computationally challenging, meanwhile MACER needs a much larger number of ($3\times$) training epochs to train (and unfortunately the resulting models often have weaker certificate despite higher clean accuracy).

Motivated by the need of cost-effective robust training methods for randomized smoothing, in this work, we propose two approaches to address the limitations of SmoothAdv and MACER. First, we propose a new robust training method called **AdvMacer** , which takes the best of both worlds in SmoothAdv and MACER: **AdvMacer** enjoys computational efficiency, and also gives larger ACR while preserving good accuracy in most settings. Second, we propose to equip our **AdvMacer** models with a training-free ensemble method **EsbRs**, which can further enlarge the smoothed classifier's certified radius (by up to 8% compared with SmoothAdv and 15% compared with MACER), hence establishing the new state-of-the-art result on certified radius. Crucially, we demonstrate the effect of both *intra*-model ensembles and *mixed*-model ensembles from the experimental point of view.

2 Related works and backgrounds

In this section, we give backgrounds on randomized smoothing and review recent literature on applying ensemble methods to randomized smoothing.

Randomized smoothing Consider a neural network classifier $f : \mathbb{R}^d \to \mathcal{Y}$ that maps an input sample $x \in \mathbb{R}^d$ to its predicted label in $\mathcal{Y}$. Cohen et al. (2019) introduced a randomized smoothing (RS) technique that can turn any base classifier $f(x)$ into a smoothed classifier $g(x)$ with provable robustness guarantees. When taking a sample x, the smoothed classifier g returns the class that the base classifier f is most likely to return under isotropic Gaussian noise perturbation of x:

$$g(x) = \arg\max_{c \in \mathcal{Y}} \boldsymbol{P}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)}(f(x + \epsilon) = c),$$

where σ is the noise level that controls the trade-off between clean accuracy and model robustness.

Cohen et al. (2019) further proved the robustness guarantees of such smoothed classifier in Theorem 2.1. Let Φ denote the cumulative density function (CDF) of the standard Gaussian distribution. Suppose that under Gaussian perturbation $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, the most likely class c_A is returned with probability p_A and the second most likely (runner-up) class c_B is returned with probability p_B, i.e. $c_A = \arg\max_{c \in \mathcal{Y}} \boldsymbol{P}(f(x + \epsilon) = c)$, $c_B = \arg\max_{c \neq c_A} \boldsymbol{P}(f(x + \epsilon) = c)$, $p_A = \boldsymbol{P}(f(x + \epsilon) = c_A)$, $p_B = \boldsymbol{P}(f(x+\epsilon) = c_B)$. In practice, Monte Carlo sampling is employed to obtain an estimate of p_A, see Cohen et al. (2019).

Theorem 2.1 (Theorem 1 of Cohen et al. (2019))**.** *Assume p_A attains a lower bound $\underline{p_A}$ and p_B attains an upper bound $\bar{p}_B$ with $\underline{p_A} < \bar{p}_B$, then $g(x + \delta) = c_A$ for all $\|\delta\|_2 < R$, where $R = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\bar{p}_B))$.*

Ensemble Model Ensemble is a popular technique in the machine learning literature to practically improve model performance and reduce generalization errors (Allen-Zhu & Li, 2020). Recently, there are a few works investigating the idea of using model ensemble to improve robustness of a randomized smoothed classifier (Horváth et al., 2021; Yang et al., 2021). However, Horváth et al. (2021) focused on ensemble the same type of models (i.e. models trained from the same process but with different random seeds, which is denoted as intra-model ensemble in the following text) and did not study the effect of using different types of models (i.e. mixed-model ensemble). Although Yang et al. (2021) doesn't have any explicit assumptions on model types, they only experimented using same types of model to ensemble. In contrast, as will be introduced in Section 3.2, our proposed **EsbRs** is a more general and effective ensemble method where we study the effect of *mixed*-model ensembles.

3 Our proposed main methods

In this section, we propose two novel and cost-effective approaches to improve robustness of a randomized smoothed classifier. First, we introduce a new robust training method **AdvMacer** that aim to maximize the certified radius over adversarial examples, and we present the intuitions, formulations as well as the details of our algorithm in section 3.1. Next, in section 3.2, we propose a novel ensemble method called **EsbRs** with motivation and empirical evaluation. Different from the two recent works (Yang et al., 2021; Horváth et al., 2021), we provide a more general ensemble design which does not require individual classifiers to come from the same training method, resulting in richer model diversity and noticeable improvement in the robustness of ensemble models.

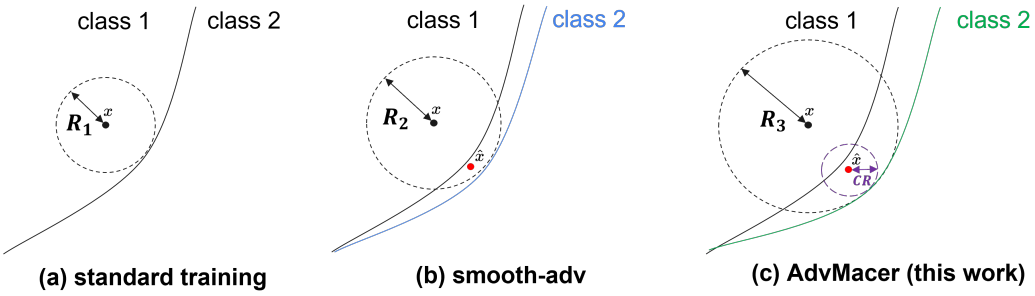

Figure 1: The illustration of the idea behind **AdvMacer** : x (black dot) is the original data sample and $\hat{x}$ (red dot) is an adversarial example of x. The solid black line is the original decision boundary. The blue line in (b) is the decision boundary using SmoothAdv and the green line in (c) is the decision boundary after applying **AdvMacer** . SmoothAdv force the classifier to classify $\hat{x}$ correctly to get the red boundary. **AdvMacer** force $\hat{x}$ to not only have correct prediction but also a large margin. Therefore, **AdvMacer** can obtain larger certified radius $R_3 >$ certified radius of smoothadv $R_2 >$ certified radius of the original classifier R_1.

3.1 Approach 1: AdvMacer

Inspired by the prior work SmoothAdv (Salman et al., 2019) and MACER (Zhai et al., 2019) and to address their limitations, we argue that a smoothed classifier can be trained to have larger certified radius by directly optimizing the certified radius of adversarial examples instead of the clean data points. Notice that this statement requires adversarial example to be predicted correctly (otherwise, the certified radius of original data point may be actually decreased), see the intuition illustrated in Figure 1. Based on the above idea, we propose the following formulation.

Formulation Given a data point x and its label y, our proposed **AdvMacer** loss consists of two terms: $L_{\text{AdvMacer}}(x) = L_{\text{CE}}(\hat{z}(\hat{x}), y) + \lambda L_{\text{R}}(\hat{z}; \hat{x}, y)$, where $\hat{z}$ and L_{R} are given in Eq 3 and Eq 4 in appendix respectively. The 1st term $L_{\text{CE}}(\hat{z}(\hat{x}), y)$ is to encourage adversarial examples $\hat{x}$ to be classified correctly, and the 2nd term $L_{\text{R}}(\hat{z}; \hat{x}, y) = \frac{\sigma}{2} \max\{\gamma - \hat{\xi}_\theta(\hat{x}, y), 0\} \mathbf{1}(\hat{g}(\hat{x}) = y)$ is to maximize the certified radius at the adversarial example $\hat{x}$, where $\hat{x} = \arg\max_{\|x'-x\|_2 \le \epsilon} L_{\text{CE}}(\hat{z}(x'), y)$. To minimize the $L_{\text{AdvMacer}}(x)$, we generate adversarial examples $\hat{x}$ via Eq 1 with T-step PGD using SmoothAdv (Salman et al., 2019), i.e. in the i-th step, we update

$$x_{i+1} = \prod_{\mathcal{B}(x,\epsilon)} \Big(x_i + \nabla_x \Big(-\log \Big(\frac{1}{m} \sum_{k=1}^{m} F(x' + \delta_k)_y \Big) \Big) \Big|_{x=x_i} \Big),$$

where $\prod_{\mathcal{S}}(\cdot)$ is the projection onto set $\mathcal{S}$ and we set $\hat{x} = x_T$. The training objective is to minimize $L_{\text{AdvMacer}}(x)$ by first-order optimization method, and a detailed algorithm is presented in the Appendix due to page constraint.

Discussion and Comparison **(I).** SmoothAdv (Salman et al., 2019) adapted adversarial training to defend against the least favorable samples but did not consider certified radius as another metric. MACER (Zhai et al., 2019) used robust training to directly maximize certified radius on clean samples instead of adversarial examples. In contrast, our proposed **AdvMacer** trains a model on adversarial examples while taking certified radius into consideration. Compared with SmoothAdv, **AdvMacer** doesn't bring any additional computational overhead to calculate robust loss as there exist analytic formula for certified radius; in the meantime, compared with MACER, we require much fewer number of epochs ($3\times$ smaller) to obtain a robust model with much larger certified radius. From the experiments in section 4 (e.g. Table 1), it can be seen that **AdvMacer** outperforms both SmoothAdv and MACER on various dataset. **(II).** SmoothAdv needs to tune a number of hyperparameters for different noise level σ, which becomes a significant challenge when the computing resources are limited. Although MACER also has many tuning parameters, empirical experiments showed that most of these parameters don't change across different σ and datset. However, MACER needs more training epochs (440 epochs per Zhai et al. (2019)) to yield a robust classifier, taking

Table 1: Cifar-10: ACR of different models on the first 500 test images of Cifar-10 with varing σ. Clean accuracy is reported in parenthesis. Reported models include SmoothAdv, MACER, **AdvMacer** , Esb-RS. $N = 100k$ samples are used in certification unless otherwise specified.

	Methods	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1.0$
Baselines	SmoothAdv (Salman et al., 2019)	0.541 (74.2%)	0.735 (56.4%)	0.758 (45.8%)
	MACER (Zhai et al., 2019)	0.518 (79.4%)	0.682 (63.4%)	0.768 (42.4%)
Ours	**AdvMacer**	**0.554** (76.0%)	**0.742** (58.4%)	**0.794** (47.6%)
	EsbRs-AdvMacer $\times 3$	**0.583** (76.4%)	0.772 (58.8%)	0.805 (47.6%)
	EsbRs-SmoothAdv$\times 3$	0.567 (76.6%)	0.777 (58.4%)	0.801 (46.6%)
	EsbRs-AdvMacer $\times 1$+SmoothAdv$\times 2$	0.572 (77.2%)	**0.783** (59.4%)	**0.810** (47.2%)
	EsbRs-AdvMacer $\times 2$+MACER$\times 1$	0.568 (79.8%)	0.728 (63.6%)	0.801 (42.8%)
	EsbRs-AdvMacer $\times 1$+MACER$\times 2$	0.570 (80.4%)	0.723 (65.0%)	0.760 (44.0%)

days to train a ResNet-110 (He et al., 2016) on Cifar-10 (Krizhevsky et al., 2009). Also, MACER usually achieves better clean accuracy but smaller average certified radius (ACR). In contrast, our **AdvMacer** takes the best of both world in Smoothadv and macer: **AdvMacer** enjoys computational efficiency, gives larger ACR while preserves good accuracy in most settings. Besides, **AdvMacer** attains a universal configuration that works well across different σ. Equipped with our proposed ensemble method presented in section 3.2, **AdvMacer** also enriches the diversity of component models, making mixed ensemble more robust. For a thorough comparison by experiments, see section 4 for more details.

3.2 Approach 2: **EsbRs**

Ensemble is a cost-effective post-training technique to enhance model performance and reduce generalization error without spending much additional efforts on re-training the neural networks. By simply averaging the output from several models, ensemble shows remarkable boost in test accuracy and model robustness. Recently, there are a few works investigating the idea of using model ensemble to improve robustness of a randomized smoothed classifier (Horváth et al., 2021; Yang et al., 2021). However, the existing work mainly focused on ensembling similar classifiers with average weights. In contrast, we also consider mixed-model ensemble with component classifiers coming from different training methods, study its performance and compare with intra-model ensemble.

Formulation Suppose we have k trained soft classifiers $F^1, \dots, F^k : \mathbb{R}^d \to P(\mathcal{Y})$ and $\mathcal{Y} = \{1, \dots, c\}$. Consider soft-ensemble model H whose output is a weighted average of the logits from $F^1, \dots, F^k$: $H(x) = \sum_{l=1}^{k} w_l F^l(x)$. Suppose the associated hard classifier is $h(x) = \arg\max_{c \in \mathcal{Y}} \big(H(x)\big)_c$. Then we apply RS to h and get the corresponding smoothed classifier g. Extensive experiments from section 4 show that ensembling the smoothed classifier g outperforms all component classifiers $F^1, \dots, F^k$ in general, no matter F^l comes from the same or different training methods. Specifically, if F^l comes from more than one training methods, we call g a mixed-model ensemble, or mixed ensemble for short.

Discussion Compared with Horváth et al. (2021), we generalize their method to allow mixed ensemble. Based on similar analysis (see appendix C for more details on our analysis), if the logits from different types of models have smaller variance than those from the same type of models, mixed ensemble can outperform intra-model ensemble, suggesting it's not always necessary to ensemble similar classifiers. To our best knowledge, we are the first work to conduct practical experiment with mixed ensemble. We note that the two recent works (Yang et al., 2021; Horváth et al., 2021) did not explore this direction.

4 Experiments

In this section, we present experimental results that empirically evaluate the performance of our proposed methods, **AdvMacer** and **EsbRs**, on Cifar-10 (Krizhevsky et al., 2009). We mainly evaluate model performance on two metrics: clean accuracy and average certified radius (ACR).

Clean accuracy is the classification accuracy when taking the original test images as the input and to evaluate robustness, we use the ACR. We follow the standard evaluation protocol used in (Zhai et al., 2019): for each test data $(x_i, y_i) \in \mathcal{S}_{\text{test}}$, record the radius R_i that can be certified the by the model g. Set $R_i = 0$ if x_i can't be classified correctly by g. Then ACR $= \frac{1}{|\mathcal{S}_{\text{test}}|} \sum_i R_i$. More analysis on evaluation metrics will be introduced in appendix B.

Baseline models Two baseline models are discussed in this section: MACER and SmoothAdv. For MACER, we follow the configurations given by Table 4 in Zhai et al. (2019). For SmoothAdv, we pick the best models under different $\sigma = 0.25, 0.50, 1.00$ from the Github repo of Salman et al. (2019). See Table 2 in appendix B for more details on hyper-parameter selection of SmoothAdv.

AdvMacer We apply Algorithm 1 to train our **AdvMacer** models. On Cifar-10, we choose $\gamma = 8.0$, $\lambda = 12.0$, $\beta = 16.0$ for all $\sigma = 0.25, 0.50, 1.00$. The choice of T, m, ϵ are summarized in Table 2. We follow the same training scheme as Salman et al. (2019). The initial learning rate is 0.1 and decays by a factor of 0.1 every 50 epochs. A batch size of 256 is used in the training. For more details, please refer to Salman et al. (2019). Note that by the choice of hyper-parameters, SmoothAdv and **AdvMacer** have the same training time, which implies the improved performance of **AdvMacer** is not gained from more expensive computation. The experiment results on Cifar-10 are summarized in Table 1.

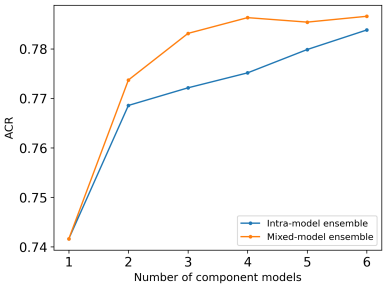

Figure 2: The plot of ACR against different number of component models in **EsbRs** on Cifar-10 with $\sigma = 0.50$. Intra-model ensemble uses N **AdvMacer** models. Mixed ensemble uses m **AdvMacer** models and n SmoothAdv models, where m and n are given in Table 3 of appendix B.

EsbRs We also employ our proposed (both intra- and mixed-) model ensemble techniques introduced in section 3.2 to enhance robustness performance. We use the following naming convention to report our result: **EsbRs-**Model1$\times$n+Model2$\times$m represents the ensemble model obtained by n Model1 and m Model2. Each component model's configuration is the same as that of the non-ensemble individual model for fair comparison. For example, **EsbRs-AdvMacer** $\times$1+SmoothAdv$\times$2 on Cifar-10 with $\sigma = 0.25$ represents the ensemble of one **AdvMacer** model and two SmoothAdv models with configuration from Table 2. Our empirical experiments also verifies the success of mixed ensemble. In ensemble experiments, we independently train all **AdvMacer** and SmoothAdv models on Cifar-10 with $\sigma = 0.50$ and the model configuration is given by Table 2. For intra-model ensemble, we use 1/2/3/4/5/6 **AdvMacer** models. For mixed ensemble, the number of component model from each category is summarized in Table 3 of appendix B. In Figure 2, we observe that ACR improves as the number of component models increases. The same observation holds for both intra-model and mixed-model ensemble. Besides, mixed ensemble gives universally better ACR as shown in Figure 2, which suggests the power of our proposed mixed ensemble.

Discussion Different from (Horváth et al., 2021), we allow mixed ensemble that are a mixture of robust models from various training methods. The introduction of **AdvMacer** also enriches diversity of component models that can be used in model ensemble to further improve the performance.

5 Conclusions

In this work, we have proposed two novel and cost-effective approaches to promote robustness of randomized smoothed classifiers. Our first approach **AdvMacer** improves the robustness by maximizing the certified radius over adversarial examples, and our second approach **EsbRs** can further improve **AdvMacer** on both clean accuracy and robustness certificate. We show that we could improve ACR by 15% compared with MACER and 8% compared with the best models of SmoothAdv. Moreover, we show that mixed-model ensemble works better in certain scenarios, which has not been studied in previous works.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Miklós Z Horváth, Mark Niklas Müller, Marc Fischer, and Martin Vechev. Boosting randomized smoothing with variance reduced classifiers. *arXiv preprint arXiv:2106.06946*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672, 2019.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *NeurIPS*, 2019.

Y. Li, X. Bian, and S. Lyu. Attacking object detectors via imperceptible patches on background. *arXiv preprint arXiv:1809.05966*, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sébastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11292–11303, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Zhuolin Yang, Linyi Li, Xiaojun Xu, Bhavya Kailkhura, Tao Xie, and Bo Li. On the certified robustness for ensemble models and beyond. *arXiv preprint arXiv:2107.10873*, 2021.

Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius. In *International Conference on Learning Representations*, 2019.

A Related work

Now we consider a smoothed soft classifier $G(x) = \boldsymbol{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}F(x+\delta)$.

SmoothAdv Salman et al. (2019) introduced SmoothAdv to find adversarial examples by PGD. Denote L_{CE} as the canonical cross entropy loss. Given a labeled data (x, y), SmoothAdv finds a point $\hat{x}$ that maximizes the cross entropy loss of $G(x)$ in the local neighborhood of x:

$$\hat{x} = \underset{\|x'-x\|_2\le\epsilon}{\arg\max}\ L_{\text{CE}}(G(x'), y) = \underset{\|x'-x\|_2\le\epsilon}{\arg\max}\ -\log \boldsymbol{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}F(x'+\delta)_y. \tag{1}$$

Such optimization problem Eq 1 is solved by projected gradient descent (PGD). To estimate the gradient of Eq 1, Salman et al. (2019) used Monte Carlo simulation to approximate $\nabla_{x'} L_{\text{CE}}(G(x'), y)$ by

$$\nabla_{x'}\Big(-\log\Big(\frac{1}{m}\sum_{k=1}^{m} F(x'+\delta_k)_y\Big)\Big),$$

where $\delta_1, \dots, \delta_m$ are drawn i.i.d. from $\mathcal{N}(0, \sigma^2 I)$.

MACER Since the certified radius is related to the difference between the top probability p_A and the runner-up probability p_B, Zhai et al. (2019) constructed MACER loss L_{MACER}, which consists of classification loss and robustness loss to both minimize classification error and maximize the certified radius of those correctly classified samples. Specifically,

$$L_{\text{MACER}}(x) = L_{\text{CE}}(G(x), y) + \lambda L_{\text{R}}(G; x, y), \tag{2}$$

where $\lambda \ge 0$ is a tuning parameter. The loss in Eq 2 involves the soft smoothed classifier G and Zhai et al. (2019) proposes to approximate $G(x)$ by Monte Carlo sampling:

$$G(x) \approx \hat{z}(x) = \frac{1}{m}\sum_{k=1}^{m} F(x+\delta_k), \quad \hat{g}(x) = \underset{i\in\mathcal{Y}}{\arg\max}\, \hat{z}_i(x). \tag{3}$$

where $\delta_1, \dots, \delta_m$ are drawn i.i.d. from $\mathcal{N}(0, \sigma^2 I)$. Denote by $\widehat{\text{CR}}(x, y)$ the approximated certified radius at x, then $\widehat{\text{CR}}(x, y) = \frac{\sigma}{2}(\Phi^{-1}(\hat{z}_y(x)) - \Phi^{-1}(\max_{y'\neq y}\hat{z}_{y'}(x)))$. Therefore, the robustness loss $L_{\text{R}}(G; x, y)$ can also be approximated by

$$\begin{aligned} L_{\text{R}}(G; x, y) \approx L_{\text{R}}(\hat{z}; x, y) &= \max\{\epsilon + \tilde{\epsilon} - \widehat{\text{CR}}(x, y), 0\}\, \mathbf{1}(\hat{g}(x) = y) \\ &= \frac{\sigma}{2}\max\{\gamma - \hat{\xi}_\theta(x, y), 0\}\, \mathbf{1}(\hat{g}(x) = y), \end{aligned} \tag{4}$$

where $\epsilon, \tilde{\epsilon} > 0$ are hyper-parameters in hinge loss, $\gamma = \frac{2}{\sigma}(\epsilon + \tilde{\epsilon})$, and $\hat{\xi}_\theta(x, y) = \Phi^{-1}(\hat{z}_y(x)) - \Phi^{-1}(\max_{y'\neq y}\hat{z}_{y'}(x))$. Finally, MACER trains a base classifier by minimizing the approximated MACER loss on training dataset. We refer readers to Zhai et al. (2019) for more details.

B Algorithm and Experiment details

The algorithm for **AdvMacer** training in presented in Algorithm 1. We summarize the configuration of the SmoothAdv and **AdvMacer** models on Cifar-10 in Table 2.

To make fair comparisons with previous baseline models, we use the same architectures as in Cohen et al. (2019); Salman et al. (2019); Zhai et al. (2019): ResNet-110 (He et al., 2016). We train our models with $\sigma = 0.25, 0.50, 1.00$ on Cifar-10. We train all models on a single NVIDIA V100 GPU and the training time reported below is all from NVIDIA V100 GPU.

Evaluation We mainly evaluate model performance on two metrics: clean accuracy and average certified radius (ACR). Clean accuracy is the classification accuracy when taking the original test images as the input and to evaluate robustness we use the ACR. We follow the standard evaluation protocol used in Cohen et al. (2019); Salman et al. (2019); Zhai et al. (2019) for fair comparison: for each test data $(x_i, y_i) \in \mathcal{S}_{\text{test}}$, record the radius R_i that can be certified the by the model g. Set $R_i = 0$ if x_i can't be classified correctly by g. Then ACR $= \frac{1}{|\mathcal{S}_{\text{test}}|}\sum_i R_i$. Since the denominator is the size of the full test set, one cannot obtain large ACR without high accuracy. Thus ACR becomes

Algorithm 1 Our **AdvMacer** $(\sigma, m, T, \lambda, \beta, \gamma)$

Input: training set $\hat{p}_{\text{data}}$, noise level σ, number of Gaussian samples m, regularization parameter λ, hinge factor γ, inverse temperature β, number of PGD step T

for each iteration **do**

1) Sample a mini-batch $(x_1, y_1), \ldots, (x_n, y_n) \sim \hat{p}_{\text{data}}$
2) For each (x_i, y_i), use T-step SmoothAdv to generate adversarial example $\hat{x}_i$
3) For each $(\hat{x}_i, y_i)$, draw m i.i.d. Gaussian samples $x_{i1}, \ldots, x_{im}$ from $\mathcal{N}(x_i, \sigma^2 I)$
4) Obtain an estimation of $G_\theta(\hat{x})$ by $\hat{z}_\theta(\hat{x}) \leftarrow \frac{1}{m}\sum_{k=1}^{m} F_\theta(\hat{x}_{ik}),$ for $i = 1, \ldots, n$
5) Collect the set of data with correct prediction: $\mathcal{S}_\theta = \{i : y_i = \arg\max_c \hat{z}_\theta(\hat{x}_i)_c\}$
6) For each $i \in \mathcal{S}_\theta$, compute the second most likely class $\hat{y}_i \leftarrow \arg\max_{c \neq y_i} \hat{z}_\theta(\hat{x}_i)_c$
7) For each $i \in \mathcal{S}_\theta$, compute $\hat{\xi}(\hat{x}_i, y_i) \leftarrow \Phi^{-1}(\hat{z}_\theta(x)_{y_i}) - \Phi^{-1}(\hat{z}_\theta(x)_{\hat{y}_i})$
8) Sample $\delta \sim \mathcal{N}(0, \sigma^2 I)$ and update θ with SGD to minimize

$$-\frac{1}{n}\sum_{i=1}^{n} \log \hat{z}_\theta(\hat{x}_i + \delta)_{y_i} + \frac{\lambda\sigma}{2n}\sum_{i \in \mathcal{S}_\theta} \max\{\gamma - \hat{\xi}_\theta(\hat{x}_i + \delta, y_i), 0\}$$

end for

Output: model parameters θ

Table 2: Model configuration: main hyper-parameters and training time for SmoothAdv and **AdvMacer** on Cifar-10 with varing σ. For the additional parameters in **AdvMacer** , we pick $\lambda = 12.0, \gamma = 8.0, \beta = 16.0$.

Models	T	m	ϵ	Epochs	σ	Time
SmoothAdv	2	8	1.0	150	0.25	15.5h
	2	8	2.0	150	0.50	15.5h
	2	4	2.0	150	1.00	8h
AdvMacer	2	8	1.0	150	0.25	15.5h
	2	8	2.0	150	0.50	15.5h
	2	4	2.0	150	1.00	8h

a popular choice in most of the DL robustness literature. We use CERTIFY algorithm in Cohen et al. (2019) to obtain certified radius and choose $N_0 = 100, N = 100,000, \alpha = 0.001$ in CERTIFY. We report model performance on the first 500 test images on Cifar-10. The details of the component models for mixed ensemble in Figure 2 are given in Table 3.

Table 3: Component models in mixed ensemble experiment in Figure 2. The mixed ensemble with totally N component models uses m **AdvMacer** and n SmoothAdv models. m and n are given as follows.

N	1	2	3	4	5	6
m	1	1	1	2	3	3
n	0	1	2	2	2	3

C Ensemble Analysis

For a fixed query point x with a Gaussian perturbation $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, suppose logits vector $\boldsymbol{y}^l \in P(\mathcal{Y})$ is returned by F^l. Without loss of generality, assume 1 is the majority class in RS for all F^l. For simplicity, we can work with classification margin $z_i^l = y_1^l - y_i^l$, for $i \in \mathcal{Y}$. Let $\bar{\boldsymbol{y}} = H(x + \epsilon)$. Therefore, $\bar{\boldsymbol{y}} = \sum_{l=1}^{k} w_l \boldsymbol{y}^l$. Similarly define $\bar{z}_i = \bar{y}_1 - \bar{y}_i$. Consider $\boldsymbol{E}[\bar{\boldsymbol{z}}] \in \mathbb{R}^c$ and $\text{Var}(\bar{\boldsymbol{z}}) \in \mathbb{R}^{c \times c}$, where the expectation is taken over the randomness in training process, including random

initialization and stochasticity in gradient descent. Then we have

$$\text{Var}(\bar{\boldsymbol{z}}) = \text{Var}\Big(\sum_{l=1}^{k} w_l \boldsymbol{z}^l\Big) = \sum_{l=1}^{k} w_l^2 \text{Var}(\boldsymbol{z}^l) + 2\sum_{l\neq m} w_l w_m \text{Cov}(\boldsymbol{z}^l, \boldsymbol{z}^m) \tag{5}$$

Hence, $\text{Var}(\bar{z}_i) = \text{Var}(\bar{\boldsymbol{z}})_{ii}$. Denote $p_i(w) = \text{Var}(\bar{z}_i)$ as a function of $w = [w_1, \ldots, w_k]^\top$ and

$$\alpha_i = \alpha_i(s) = \max_{1\leq l\leq k} \text{Var}(\boldsymbol{z}^l)_{ii} \quad \beta_i = \beta_i(s) = \max_{l\neq m} \text{Cov}(\boldsymbol{z}^l, \boldsymbol{z}^m)_{ii}$$

Suppose there are a fixed number of training methods and denote this number by s, so the above maximum is in fact taken over s different classes. As a result, $\alpha_i(s), \beta_i(s) = O(1)$ even as $k \to \infty$. As a special case, consider $w_l = \frac{1}{k}$ for all $l = 1, \ldots, k$. By Eq 5, we derive

$$p_i(w) = \text{Var}(\bar{z}_i) \leq \frac{k\alpha_i + k(k-1)\beta_i}{k^2} = \beta_i + \frac{\alpha_i - \beta_i}{k}. \tag{6}$$

These classifiers either come from different training methods, or same training method with different random seeds. Thus, existing work all assumes that the logits from one classifier have larger covariance α_i than the logits from different classifiers β_i. However, as we will see in the following **Discussion** paragraph, ensemble may harm the performance if the above assumption doesn't hold. For now, let's assume $\alpha_i > \beta_i$. By Eq 6, we conclude that the upper bound of $\text{Var}(z_i)$ decreases to a constant β_i as $k \to \infty$.

Next, we explain how $\text{Var}(\bar{z}_i)$ affects certified radius. From Theorem 2.1, we see that $R = \sigma\Phi^{-1}(\underline{p}_A)$ if $\underline{p}_A \geq \frac{1}{2}$, hence we only need to show a lower bound on the top class probability p_A increases as k becomes larger. Since we assume the majority class's number is 1, we see that

$$p_1 = \boldsymbol{P}(\bar{z}_i > 0, \forall i = 2, \ldots, c) \geq 1 - \sum_{i=2}^{c} \boldsymbol{P}(\bar{z}_i \leq 0) \tag{7}$$

By Chebyshev's inequality, $\boldsymbol{P}(\bar{z}_i \leq 0) \leq \boldsymbol{P}\Big(\big|\bar{z}_i - \boldsymbol{E}[\bar{z}_i]\big| \geq \boldsymbol{E}[\bar{z}_i]\Big) \leq \frac{\text{Var}(\bar{z}_i)}{\boldsymbol{E}[\bar{z}_i]^2}$ and let $e_i = e_i(s) = \min_l \boldsymbol{E}[z_i^l]$, thus from Eq 7 we have

$$p_1 \geq 1 - \sum_{i=2}^{c} \frac{\text{Var}(\bar{z}_i)}{e_i^2}. \tag{8}$$

The above equation suggests us to choose the weight w that maximizes the RHS of Eq 8 to have a larger p_1, hence larger certified radius. Since e_i is independent of the choice of w, we can obtain the optimal weight by solving

$$\min_{w\in\mathbb{R}^k} \sum_{i=2}^{c} a_i p_i(w) \quad \text{s.t.} \sum_{l=1}^{k} w_l = 1, \quad w_l \geq 0, \tag{9}$$

where $a_i = e_i^{-2}$ are constants. Note that when $w_l = \frac{1}{k}$ for all $l = 1, \ldots, k$, we have a lower bound on p_1 by Eq 6 and Eq 8:

$$p_1 \geq 1 - \sum_{i=2}^{c} \frac{\beta_i + (\alpha_i - \beta_i)/k}{e_i^2} \to 1 - \sum_{i=2}^{c} \frac{\beta_i}{e_i^2} \text{ as } k \to \infty.$$

This explains why larger k makes p_1 and certified radius larger even in average ensemble.

Discussion Compared with Horváth et al. (2021), we generalize their analysis to allow both weighted and mixed ensemble and hence have several new findings. First, if $\alpha_i < \beta_i$, namely the logits from one model have smaller variance than those from different models, the RHS of Eq 6 becomes an increasing function in k, which implies ensemble does not always work. Second, suppose F^1, F^2 come from model category 1 (for example, SmoothAdv) and F^3 comes from model category 2 (for example, **AdvMacer**). If the logits from different types of models have smaller variance than those from the same type of model, namely $\text{Cov}(F^1, F^3) < \text{Cov}(F^1, F^2)$, β_i will become smaller and makes mixed ensemble work better than single ensemble. This phenomenon is observed in Figure 2.