Applied Intelligence (2023) 53:9483–9498
https://doi.org/10.1007/s10489-022-03423-5

Adversarial robustness via noise injection in smoothed models

Yaniv Nemcovsky[1] · Evgenii Zheltonozhskii[1] · Chaim Baskin[1] · Brian Chmiel[2,3] · Alex M. Bronstein[1] · Avi Mendelson[1]

Accepted: 22 February 2022 / Published online: 9 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Deep neural networks are known to be vulnerable to malicious perturbations. Current methods for improving adversarial robustness make use of either implicit or explicit regularization, with the latter is usually based on adversarial training. Randomized smoothing, the averaging of the classifier outputs over a random distribution centered in the sample, has been shown to guarantee a classifier's performance subject to bounded perturbations of the input. In this work, we study the application of randomized smoothing to improve performance on unperturbed data and increase robustness to adversarial attacks. We propose to combine smoothing along with adversarial training and randomization approaches, and find that doing so significantly improves the resilience compared to the baseline. We examine our method's performance on common white-box (FGSM, PGD) and black-box (transferable attack and NAttack) attacks on CIFAR-10 and CIFAR-100, and determine that for a low number of iterations, smoothing provides a significant performance boost that persists even for perturbations with a high attack norm, ϵ. For example, under a PGD-10 attack on CIFAR-10 using Wide-ResNet28-4, we achieve 60.3% accuracy for infinity norm $\epsilon_\infty = 8/255$ and 13.1% accuracy for $\epsilon_\infty = 35/255$ – outperforming previous art by 3% and 6%, respectively. We achieve nearly twice the accuracy on $\epsilon_\infty = 35/255$ and even more so for perturbations with higher infinity norm. A reference implementation of the proposed method is provided.

Keywords Neural networks · Computer vision · Adversarial robustness · Adversarial examples · Noise injection · Randomized smoothing

1 Introduction

Deep neural networks (DNNs) are demonstrating spectacular performance in a variety of computer vision tasks, but at the same time are susceptible to adversarial examples – small perturbations that alter the network output [18, 41]. Since the initial discovery of this phenomenon in 2013, increasingly stronger defenses [18, 20, 23, 33, 38, 49, 55, 57] and counterattacks [1, 8, 11, 18, 28, 33, 34, 36] were proposed in the literature. Adversarial attacks have also been shown to occur in tasks beyond the field of image classification where they were first discovered: in real-life object recognition [2, 7, 52], object detection [43], natural language processing [10, 13, 22], reinforcement learning [16], speech-to-text [9], and point cloud classification [47], to mention only a few. Adversarial examples can also be used to improve the performance of the DNNs on unperturbed data [17, 40, 48].

Understanding the root cause of adversarial examples, how they are created, and how we can detect and prevent such attacks, is at the center of many research works. Gilmer et al. [14] argued that adversarial examples are an inevitable property of high-dimensional data manifolds rather than a weakness of specific models. Accordingly, an adversarial defense's real goal is not to get rid of adversarial examples, but rather to make their search difficult.

Current defense methods are based on either implicit or explicit regularization. Explicit regularization methods aim to increase the performance under adversarial attack by directly incorporating a suitable term into the loss of the network during training, usually by incorporating adversarial examples for the dataset used in the training process. In contrast, implicit regularization methods that do not change the objective, such as variational dropout [25], seek to train the network to be robust against any perturbations

✉ Evgenii Zheltonozhskii
evgeniizh@campus.technion.ac.il

[1] Department of Computer Science, Technion, Haifa, Israel

[2] Habana Labs, An Intel Company, Caesarea, Israel

[3] Department of Electrical Engineering, Technion, Haifa, Israel

without taking into account adversarial examples. [15] showed that there is a strong connection between robustness to random noise corruption and adversarial robustness. Implicit regularization seeks to exploit this connection to achieve adversarial robustness. In particular, adding randomness to the network can be an especially successful tactic [5, 20, 31, 42], since information acquired from previous runs cannot be directly applied to a current run. Another way to utilize randomness to improve classifier robustness is through randomized smoothing [12]: averaging the outputs of the classifier over some random distribution centered in the input data point. The effects of these three approaches (explicit regularization, implicit regularization, and smoothing) do not necessarily line up with or contradict each other. Thus, one could use a combination of the three when devising adversarial defenses.

Previous works that discuss randomized smoothing do so exclusively in the context of certified robustness [12, 37]. While certification is a very strong and desired guarantee, researchers were unable to achieve practical ℓ_∞ certification radii, and it is unclear whether this is at all possible [6, 27, 53]. Nevertheless, smoothing should still be considered as a practical method to improve empirical classifier performance. Therefore, we investigate the effect of smoothing for both the clean accuracy and the adversarial robustness of the model. We demonstrate this effect empirically on top of adversarial regularization methods – both implicit [20, 58] and explicit [33]. Below, we outline our main contributions.

Firstly, we study the effect of randomized smoothing on empirical accuracy of adverserially trained classifiers, both on perturbed and clean data. Our experiments show that even for a small number of samples, adversarial accuracy increases with no degradation of the clean accuracy. Also, since the performance grows with the sample size, smoothing introduces a trade-off between inference time complexity and accuracy. While this effect is present even for higher attack radii, it decreases with the number of iterations, suggesting that at least some of the impact may be attributed to random gradient obfuscation.

Secondly, in addition to the combination of smoothing with adversarial training proposed by [37], we use smoothing in tandem with methods that utilize implicit regularization based on noise injection. Combining this regularization with smoothing results in a substantial increase in the smoothed classifier's performance on both perturbed and clean data, achieving state-of-the-art results in various configurations. In addition, we discuss several methods of smoothing during inference and ways to optimize a pretrained adversarial model to improve the accuracy of the smoothed classifier.

Lastly, we show a relation between training the smoothed classifier and training the base model under perturbation. A similar relation was previously demonstrated for the cross-entropy loss by [12]. We, on the other hand, show a relation for the 0-1 loss under weaker assumptions.

The rest of the paper is organized as follows: Section 2 reviews related work, Section 3 describes our proposed method, Section 4 describes the integration of the method with adversarial training, Section 5 provides our experimental results, and Section 6 concludes the paper.

2 Related work

Adversarial attacks were first proposed by [41] who noted that it was possible to use the gradients of a neural network to discover small perturbations of the input that drastically change its output. Moreover, they posited that it is usually possible to change the prediction to a particular class, i.e., perform a *targeted* attack. Adversarial attacks are commonly divided into two classes: *white-box* attacks, which have access to the internals of the model (in particular, its gradients); and *black-box* attacks, which have access only to the output of the model for a given input.

Adversarial defenses. To improve the performance of the model under adversarial attack, [41] generated adversarial examples during training and used them as training samples, optimizing the loss

$$\begin{aligned}\mathscr{L}_{\text{adv}}(\mathbf{x}, y) = &(1-w)\cdot\mathscr{L}_{CE}(f(\mathbf{x}), y)\\ &+w\cdot\mathscr{L}_{CE}(f(\mathbf{x}), y),\end{aligned} \tag{1}$$

where $\mathscr{L}_{CE}$ is the cross-entropy loss, f is the classifier, $\mathbf{x}$ is a training instance with the label y, $\mathbf{x}$ is the corresponding adversarial example, and w is a hyperparameter usually set to $w = 0.5$.

This method is particularly convenient if the generation of adversarial examples is fast [18]. When combined with stronger attacks, it provides a powerful baseline for adversarial defenses [33], and is often utilized as part of defense procedures.

Many works introduced improvements over regular adversarial training by applying stronger attacks during the adversarial training phase, including using Voronoi cells instead of ϵ-balls [23], adding adversarial noise to all the activations [30], using a learning-to-learn framework [21, 50], or increasing the attack strength based on some heuristic [3, 44].

TRADES [55] uses a batch of randomly perturbed inputs to better cover the neighborhood of the point. Together with replacing $\mathscr{L}_{CE}(f(\mathbf{x}), y)$ in (1) by $\mathscr{L}_{CE}(f(\mathbf{x}), f(\mathbf{x}))$, this provided a significant improvement over standard adversarial training. MART [45] further improves defensive performance by treating the misclassified examples separately.

Randomization of the neural network can be a very powerful adversarial defense. [59] improved robustness by reducing the distance between two samples differing by a normally distributed variable with a small variance. [57]

added normal noise to the input, which was shown to reduce the Kullback-Leibler divergence between the clean and the adversarially-perturbed inputs. [35] studied the theoretical bounds of adversarial robustness of randomized classifiers.

Another random-based defense based on adversarial training is parametric noise injection (PNI) developed by [20]. PNI improves network robustness by injecting Gaussian noise into parameters (weights or activations) with learned variance. [58] extended the idea of PNI to colored noise injection (CNI) by injecting low-rank multivariate Gaussian noise instead of white noise:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{2}$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda} + \mathbf{V}\mathbf{V}^{\top}, \tag{3}$$

where $\boldsymbol{\Lambda}$ is an $N \times N$ non-negative diagonal matrix, and $\mathbf{V}$ is an $N \times M$ matrix. Both matrices are learned during the training using the reparametrization trick [24]. At inference time, a noise vector sampled from the learned distribution is added to activations, weights, or both. In our work, we focus on weight noise injection, which usually achieves higher adversarial robustness.

[1] observed that many defense strategies do not improve robustness of the defended network but rather obfuscate gradients, making gradient-based optimization methods less effective. They identified common properties of obfuscated gradients and organized them in a checklist. In addition, they proposed techniques to overcome common instances of obfuscated gradients – in particular, approximating non-differentiable functions with a differentiable substitute and using averaging on the randomized ones.

Randomized smoothing [12], as noted above, is a method for increasing the robustness of a classifier by averaging its outputs over some random distribution in the input space centered at the input sample. [12] showed that randomized smoothing is useful for certification of the classifier [46] – that is, proving its performance under norm-bounded input perturbations. Notably, they demonstrated a tight bound on L_2 certification using randomized smoothing with normal distribution. Consequently, [37] used the smoothed classifier to generate stronger "smooth" adversarial attacks and utilized these for adversarial training. Such training allows the generation of a more accurate base classifier and, as a result, improves certified robustness properties. Certifying against ℓ_∞ perturbations, however, is a significantly harder task [4, 6, 27, 53, 60]. In particular, the state-of-the-art certified accuracy is 30.5% for $\epsilon = 8/255$ [56] and 62.6% for $\epsilon = 2/255$ [6].

3 Randomized smoothing and implicit regularization

In what follows, we start with a definition of inference smoothing, and propose two methods for the aggregation of the samples: prediction smoothing and soft prediction smoothing. Finally, we show a relation between optimization of the smoothed classifier and optimization of the base classifier under implicit regularization. Adversarial perturbations are discussed in Section 4.

3.1 Inference smoothing

Let $F_\theta : \mathbb{R}^n \to \mathbb{R}^{|\mathscr{Y}|}$ be a map assigning, for each input $\mathbf{x} \in \mathbb{R}^n$, logit for each $y \in \mathscr{Y}$. In addition, let $f_\theta : \mathbb{R}^n \to \mathscr{Y}$ be a classifier assigning, for each input $\mathbf{x} \in \mathbb{R}^n$, the most likely class in $y \in \mathscr{Y}$ according to F_θ:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y \in \mathscr{Y}} F_{\boldsymbol{\theta}}(\mathbf{x}), \tag{4}$$

For some random variable $\boldsymbol{\eta} \in \mathbb{R}^n$ with density function $p(\boldsymbol{\eta})$, we define smoothed classifier $\tilde{f}_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathscr{Y}$, a map assigning to an input $\mathbf{x}$ the class that the base classifier $f_{\boldsymbol{\theta}}$ is most likely to return for $\mathbf{x}$ under random perturbation η:

$$\tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y \in \mathscr{Y}} P_{\boldsymbol{\eta}}[f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\eta}) = y] = \tag{5}$$

$$= \arg\max_{y \in \mathscr{Y}} \int_{\mathbb{R}^n} \mathrm{d}\boldsymbol{\eta}\, p(\boldsymbol{\eta}) \mathbb{1}\left[f_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\eta}) = y\right] = \tag{6}$$

$$= \arg\max_{y \in \mathscr{Y}} \int_{\mathbb{R}^n} \mathrm{d}\boldsymbol{\eta}\, p(\boldsymbol{\eta}) \mathbb{1} \left[\arg\max_{y' \in \mathscr{Y}} F_{\boldsymbol{\theta}, y'}(\mathbf{x} + \boldsymbol{\eta}) = y\right], \tag{7}$$

where $F_{\theta, y'} : \mathbb{R}^n \to \mathbb{R}$ is the y' element of logit vector F_θ.

$\tilde{f}_{\boldsymbol{\theta}}$ can be generalized to use an arbitrary voting function of logits $V_y : \mathbb{R}^{|\mathscr{Y}|} \to \mathbb{R}$:

$$\tilde{f}^V_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y \in \mathscr{Y}} \int_{\mathbb{R}^n} \mathrm{d}\boldsymbol{\eta}\, p(\boldsymbol{\eta}) V_y(F_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\eta})), \tag{8}$$

from which (7) can be acquired by taking V_y to be the following indicator function:

$$V_y(F_{\boldsymbol{\theta}}(\mathbf{x} + \boldsymbol{\eta})) = \mathbb{1}\left[\arg\max_{y' \in \mathscr{Y}} F_{\boldsymbol{\theta}, y'}(\mathbf{x} + \boldsymbol{\eta}) = y\right], \tag{9}$$

In the case when the base model's parameters are randomised as in [58], we generalise $\tilde{f}_{\boldsymbol{\theta}}$ further to integrate over all sources of randomness:

$$\tilde{f}^V_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y \in \mathscr{Y}} \int_{\mathbb{R}^{|\boldsymbol{\theta}|}} \int_{\mathbb{R}^n} \mathrm{d}\boldsymbol{\eta}_{\mathbf{x}} \mathrm{d}\boldsymbol{\eta}_{\boldsymbol{\theta}}\, p_{\boldsymbol{\eta}_{\mathbf{x}}}(\boldsymbol{\eta}_{\mathbf{x}}) p_{\boldsymbol{\eta}_{\boldsymbol{\theta}}}(\boldsymbol{\eta}_{\boldsymbol{\theta}}) V_y (F_{\boldsymbol{\theta} + \boldsymbol{\eta}_{\boldsymbol{\theta}}}(\mathbf{x} + \boldsymbol{\eta}_{\mathbf{x}})), \tag{10}$$

where $\boldsymbol{\eta}_{\mathbf{x}} \in \mathbb{R}^n$, $\boldsymbol{\eta}_{\boldsymbol{\theta}} \in \mathbb{R}^{|\boldsymbol{\theta}|}$ are some random variables with density function $p_{\boldsymbol{\eta}_{\mathbf{x}}}(\boldsymbol{\eta}_{\mathbf{x}})$, $p_{\boldsymbol{\eta}_{\boldsymbol{\theta}}}(\boldsymbol{\eta}_{\boldsymbol{\theta}})$.

Similarly to how smoothing ((7)) utilizes the neighborhood of the input point $\mathbf{x}$ to make a more informed prediction,

generalized smoothing (10) can be seen as using the neighborhood of the base classifier $f_{\boldsymbol{\theta}}$ in the parameter space to make an even more informed prediction.

Since the integral (8) is intractable, it is usually approximated using the Monte Carlo method, i.e., by averaging over a number of points sampled independently from each distribution. Note that while the output of the classifier $f_{\boldsymbol{\theta}+\boldsymbol{\eta}_{\boldsymbol{\theta}}}(\mathbf{x}+\boldsymbol{\eta}_{\mathbf{x}})$ is a random variable, the output of $\tilde{f}_{\boldsymbol{\theta}}$ is a deterministic function of $\mathbf{x}$, which depends on the distributions of $\boldsymbol{\eta}_{\mathbf{x}}$ and $\boldsymbol{\eta}_{\boldsymbol{\theta}}$. As the Monte-Carlo approximation is dependent on the sampling, there is some randomness where $\tilde{f}_{\boldsymbol{\theta}}$ is computed using this process. Nonetheless, given enough samples, our approximation should converge to the true value of $\tilde{f}_{\boldsymbol{\theta}}$. We denote by M the number of samples used for Monte Carlo approximation. Under this approximation, the most general form of the smoothed model output is

$$\tilde{f}^{V}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \arg\max_{y\in\mathscr{Y}} \sum_{i=1}^{M} V_y(F^i_{\boldsymbol{\theta}}(\mathbf{x})), \tag{11}$$

where we denote $F^i_{\boldsymbol{\theta}}(\mathbf{x}) = F_{\boldsymbol{\theta}+\boldsymbol{\eta}^i_{\boldsymbol{\theta}}}(\mathbf{x}+\boldsymbol{\eta}^i_{\mathbf{x}})$.

We study two different implementations of smoothing, differing in their voting function V_y: prediction smoothing, and soft prediction smoothing.

Computational complexity Computation of the expression ((11)) consists of several steps for each of the M samples. We first sample noise for $\mathbf{x}$ from a Gaussian distribution. In addition, we sample a low-rank Gaussian noise $\boldsymbol{\epsilon}$ for each tensor in $\boldsymbol{\theta}$, by sampling

$$\boldsymbol{\epsilon}_d \sim \mathscr{N}(0, I_{N\times N}) \tag{12}$$

$$\boldsymbol{\epsilon}_o \sim \mathscr{N}(0, I_{K\times K}) \tag{13}$$

$$\boldsymbol{\epsilon} = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\epsilon}_d + \mathbf{V}\boldsymbol{\epsilon}_o. \tag{14}$$

We then continue to compute the output of the network for the sampled $\mathbf{x}$ and $\boldsymbol{\theta}$. Finally, after calculating predictions for each of M samples, we compute the value of the voting function V_y for each $y \in \mathscr{Y}$. When compared to the randomized smoothing previously discussed by [12] and [37] for certified robustness, the only computational overhead is sampling noise for $\boldsymbol{\theta}$. This process is negligible when compared with network computation, and thus total complexity for all three smoothing approaches can be well approximated as M times the complexity of the single inference.

3.1.1 Prediction smoothing

The simplest possible way to aggregate predictions is to perform prediction voting, i.e., to output the most frequent prediction among the samples, or, equivalently, to approximate (7) directly using the Monte Carlo method. In this case, V_y is the same as in (9):

$$V_y(F^i_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{1}\left[\arg\max_{y'\in\mathscr{Y}} F^i_{\boldsymbol{\theta},y'}(\mathbf{x}) = y\right], \tag{15}$$

Algorithm 1 Inference smoothing prediction (Section 3.2).

Input $\mathbf{x}$: image to be classified
Input $\mathscr{Y}$: the set of possible classifications
Input M: number of Monte Carlo samples
Input V_y: voting function
Input $\boldsymbol{\theta}$: base model parameters
Input $\boldsymbol{\eta}_{\mathbf{x}}$: input noise random variable
Input $\boldsymbol{\eta}_{\boldsymbol{\theta}}$: parameters' random variable
Output y: classification for $\mathbf{x}$

Compute logit vector for Monte Carlo samples:
for $m \leftarrow 1$ to M **do**
  $\boldsymbol{\eta}^m_{\mathbf{x}} \leftarrow \text{Sample}(\boldsymbol{\eta}_{\mathbf{x}})$
  $\boldsymbol{\eta}^m_{\boldsymbol{\theta}} \leftarrow \text{Sample}(\boldsymbol{\eta}_{\boldsymbol{\theta}})$
  $O_m \leftarrow F_{\boldsymbol{\theta}+\boldsymbol{\eta}^m_{\boldsymbol{\theta}}}(\mathbf{x}+\boldsymbol{\eta}^m_{\mathbf{x}})$
end for
Compute classification probabilities:
for $y' \in \mathscr{Y}$ **do**
  $p_{y'} \leftarrow \frac{1}{M}\sum_{m=1}^{M} V_{y'}(O_m)$
end for
return $y = \arg\max_{y'\in\mathscr{Y}} p_{y'}$

yielding the following smoothed classifier

$$\tilde{f}^{V}_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y\in\mathscr{Y}} \sum_{i=1}^{M} \mathbb{1}\left[\arg\max_{y'\in\mathscr{Y}} F^i_{\boldsymbol{\theta},y'}(\mathbf{x}) = y\right], \tag{16}$$

In this form, randomized smoothing was previously discussed by [12] and [37] for certified robustness.

It is important to emphasize that this voting scheme only considers the classification of each of the M samples, discarding the predicted probabilities of each class. From now on, we use this voting method unless otherwise specified. In Section 5 we show that in various configurations, taking M predictions into account is a viable method to increase the classifier's accuracy on both clean and adversarially-perturbed inputs.

3.1.2 Soft prediction smoothing

Alternatively, we can calculate the probability expectation for each class and use the results for prediction generation. In this case, we take V_y to be the y element of the output vector of the softmax function:

$$V_y(F^i_{\boldsymbol{\theta}}(\mathbf{x})) = [\text{softmax}(F^i_{\boldsymbol{\theta}}(\mathbf{x}))]_y, \tag{17}$$

yielding the following smoothed classifier

$$\tilde{f}^V_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_{y\in\mathscr{Y}} \sum_{i=1}^{M} \big[\mathrm{softmax}\big(F^i_{\boldsymbol{\theta}}(\mathbf{x})\big)\big]_y. \tag{18}$$

A similar method was mentioned by [37] as a way to apply adversarial training to a smoothed classifier. We provide a comparison of the two in Section 4.2. Since the probabilities for each class are now differentiable, the classifier can be trained end-to-end using smooth predictions. As opposed to prediction smoothing, we now fully consider the predicted class probabilities of each of the M samples.

Soft smoothing, however, is a weaker adversarial defense than prediction smoothing. Even if the attack is not successful, the probability of competing classes increases, which means that even an unsuccessful attack on a base model impacts the prediction of the smoothed model. Therefore, we do not use this kind of voting as a defense; nevertheless, in Section 4.2 we describe how this method can be used to create stronger attacks for adversarial training.

3.2 Relation to implicit regularization

We now show that optimizing the smoothed classifier is strongly related to optimizing the base classifier under implicit regularization. This relation is our motivation for investigating the effect of smoothing over a CNI base model.

In the case of prediction smoothing (or other cases in which V_y is not differentiable), we cannot train the smoothed model directly. Therefore, we would like to train the base model to optimize the loss of the smoothed model. For this, we estimate the loss function of the smoothed classifier with an expression that can be calculated at training time.

To this end, we use the 0-1 loss of n training samples. Denoting $\tilde{f}_{\boldsymbol{\theta}}$ as in (7), the loss

$$\mathscr{L}_{01} = \sum_{i=1}^{n} \ell_{01}(y_i, \tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)), \tag{19}$$

with the pointwise terms

$$\ell_{01}(y_i, \tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)) = 1 - \mathbb{1}\big[y_i = \arg\max_{y\in\mathscr{Y}} P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y\big]\big], \tag{20}$$

is minimized over the model parameters $\boldsymbol{\theta}$. We can approximate the indicator as

$$\mathbb{1}\big[y_i = \arg\max_{y\in\mathscr{Y}} P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y\big]\big] = \tag{21}$$

$$= \mathbb{1}\big[P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big] \geq \max_{y'\in\mathscr{Y}\setminus y_i} P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]\big] \approx \tag{22}$$

$$\approx \frac{1}{2}\big[1 + P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big] - \max_{y'\in\mathscr{Y}\setminus y_i} P_\eta\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]\big], \tag{23}$$

where we approximate the Heaviside function $\mathbb{1}[x > c]$ on an interval $[0, 1]$ with a better-behaving linear function $(x-c+1)/2$.

We now show a relation to training the base model under perturbation:

$$\begin{aligned}\ell_{01}\big(y_i, \tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)\big) &\approx 1 - \frac{1}{2}\big[1 + \mathbb{E}_\eta[\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big]] \\ &\quad - \max_{y'\in\mathscr{Y}\setminus y_i} \mathbb{E}_\eta[\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]]\big] = \\ &= \frac{1}{2}\big[1 - \mathbb{E}_\eta[\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big]] \\ &\quad + \max_{y'\in\mathscr{Y}\setminus y_i} \mathbb{E}_\eta[\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]]\big].\end{aligned} \tag{24}$$

Written in this form, the 0-1 loss is now amenable to Monte Carlo approximation; however, working with a non-convex loss is still problematic. We, therefore, bound the maximum over the expectation by the expectation of the maximum,

$$\begin{aligned}\ell_{01}(y_i, \tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)) &\leqslant \frac{1}{2}\big[1 - \mathbb{E}_\eta[\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big]] \\ &\quad + \mathbb{E}_\eta\big[\max_{y'\in\mathscr{Y}\setminus y_i} \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]\big]\big] \\ &= \frac{1}{2}\mathbb{E}_\eta\big[1 - \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big] \\ &\quad + \max_{y'\in\mathscr{Y}\setminus y_i} \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]\big].\end{aligned} \tag{25}$$

Since the classification events are disjoint, only one of $\mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big]$ is non-zero and thus we can replace the maximum over the indicators with a sum:

$$\begin{aligned}&\max_{y'\in\mathscr{Y}\setminus y_i} \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big] \\ &= \sum_{y'\in\mathscr{Y}\setminus y_i} \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y'\big] \\ &= 1 - \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big],\end{aligned} \tag{26}$$

acquiring

$$\begin{aligned}\ell_{01}\big(y_i, \tilde{f}_{\boldsymbol{\theta}}(\mathbf{x}_i)\big) &\leqslant \frac{1}{2}\mathbb{E}_\eta\big[1 - \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big] \\ &\quad + 1 - \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big]\big] =\end{aligned} \tag{27}$$

$$\begin{aligned}&= \mathbb{E}_\eta\big[1 - \mathbb{1}\big[f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}) = y_i\big]\big] \\ &= \mathbb{E}_\eta\big[\ell_{01}(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i+\boldsymbol{\eta}))\big],\end{aligned} \tag{28}$$

which is the 0-1 loss of the base classifier under Gaussian perturbation. In addition, for the ℓ-th layer, we can rewrite the network inference as

$$F(\mathbf{x}_i) = F_2 \circ F_1(\mathbf{x}_i) = F_2(\mathbf{x}'_i), \tag{29}$$

where F_1 denotes the first $\ell - 1$ layers and F_2 stands for the rest of the network. Repeating the computation for F_2 with noise injection into $F_1(\mathbf{x}_i) = \mathbf{x}'_i$ instead of F with noise injection into $\mathbf{x}_i$ shows that implicit regularization in the form of injecting Gaussian noise into the activation in any layer should help to minimize the loss of the smoothed classifier. Thus, we study an application of smoothing to CNI [58] – a family of models that inject noise as an adversarial defense, expecting that both the base model and the smoothing process would benefit from noise injection.

4 Adversarial robustness of smoothed classifiers

We now discuss the robustness of a smoothed model under adversarial perturbations, and specifically attacks that target the smoothed classifier, which we denote as "smoothed attacks". We first discuss how adversarial attacks can be produced for smoothed classifiers, and then consider how the soft smoothing scheme described in Section 3.1.2 gives rise to smoothed attacks. Finally, we show a relation between random noise injection and random gradient obfuscation.

4.1 Adversarial attacks on smoothed classifiers

We now examine the robustness of a smoothed model under adversarial perturbations. For an adversarial attack to be successful on a smoothed classifier $\tilde{f}^V_{\boldsymbol{\theta}}(\mathbf{x})$, it needs to achieve high probability of misclassification both for some set of inputs in the neighbourhood of the original one and for some set of models in the neighborhood of the base model $f_{\boldsymbol{\theta}}(\mathbf{x})$ (since it is averaged over both). While due to transferability of the attacks [32] this is still feasible, producing such attacks requires optimizing over multiple models and inputs and, as a result, is more computationally expansive.

In the case of white-box attacks, we propose two methods for attacking smoothed classifiers, and specifically for non-differential voting functions such as in prediction smoothing (16). Our first approach is to make use of PGD attacks on the base classifier while compensating for the noise injection of the smoothed model; i.e, at each step we sample a single input and the smoothed model's network and optimize our attack to fool this specific sample. We consider such attacks an immediate extension of PGD for models that use noise injection in their prediction, and compare their performance with classic PGD attacks on models that do not use noise injection. Secondly, in Section 4.2 we propose to utilize smoothed attacks. We test the performance of smoothed attacks on our model and employ them as a part of fine-tuning adversarial training. When compared to PGD attacks with the same number of steps, the complexity of smoothed attacks is higher as they take into account multiple networks and inputs in each step.

4.2 Smoothed attacks

Since the smoothing of the classifier is utilized as a tool for improving adversarial robustness, the natural idea is to try to target the smoothed classifier rather than the base model in adversarial attacks. In the case of soft smoothing, the output of the smoothed classifier is differentiable and can be used as a source of gradients for white-box attacks. [37] discussed this family of attacks as a way to improve the generalization of a base model and improve the certified accuracy of a smoothed model.

We consider adversarial training with smoothed attacks as a method to optimize the smoothed model directly via training the base model. As such, we expect smoothing to be more effective for a base model trained with these attacks.

Comparison of smoothed attacks In addition to SMOOTHADV$_{\text{PGD}}$ attacks proposed by [37], we consider PGD attacks where the gradients are computed based on several randomized samples, which we denote as expectation projected gradient decent (EPGD) attacks. An important distinction between the SMOOTHADV$_{\text{PGD}}$ attack considered by [37], and EPGD is the order of the averaging and the loss operator when computing the gradients for the attack. For cross-entropy loss and M samples this amounts to:

$$\begin{aligned} &\mathscr{L}_{\text{SmoothAdv}}(\mathbf{x}, y) \\ &= -\log\Big(\frac{1}{M}\sum_{i=1}^{M}\big[\text{softmax}\big(F^i_{\boldsymbol{\theta}}(\mathbf{x})\big)\big]_y\Big) \quad \text{vs.} \end{aligned} \tag{30}$$

$$\begin{aligned} &\mathscr{L}_{\text{EPGD}}(\mathbf{x}, y) \\ &= -\frac{1}{M}\sum_{i=1}^{M}\log\Big(\big[\text{softmax}(F^i_{\boldsymbol{\theta}}(\mathbf{x}))\big]_y\Big). \end{aligned} \tag{31}$$

EPGD attacks can be considered a version of the EOT (Expectation over Transformation) family of attacks suggested by [1]. Since EOT aims to neutralize the effect of random gradient obfuscation in the model's defense, we expect EPGD to be more efficient than SMOOTHADV$_{\text{PGD}}$. In addition, EPGD attacks arise naturally by taking the gradient of the soft prediction smoothing scheme in (18). Since these attacks directly target the smoothed classifier, we both test their performance on our model and in use in adversarial training during fine-tuning.

4.3 Random gradient obfuscation via noise injection

To get some intuition on the effect of noise injection on adversarial attacks, we first perform an analysis of a simple classification model – a support vector machine (SVM) $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ with the parameters $\mathbf{w}$ and b. Specifically, we consider the formulation of an SVM under adversarial attacks with zero-mean random noise injected into the input. We assume that the attacker is aware of this noise, but is limited to observing the effect of a single realization thereof.

Algorithm 2 Soft smoothing adversarial training (Section 4.2).

Input (X, Y): Training dataset
Input ℓ: loss function
Input N: Number of epochs
Input B: Number of Batches in epoch
Input λ: learning rate
Input p: norm number for the attack limit
Input α: step size for the attack
Input ϵ: norm limit for the attack
Input M: number of Monte Carlo samples
Input $\boldsymbol{\eta}_{\mathbf{x}}$: input noise random variable
Input $\boldsymbol{\eta}_{\boldsymbol{\theta}}$: parameters' random variable
Input $\boldsymbol{\theta}$: base model initial parameters
Output $\boldsymbol{\theta}$: base model final parameters

for $epoch = 1$ to N **do**
  for $i = 1$ to B **do**
    $\boldsymbol{\delta}_{\text{EPGD}} \leftarrow 0$
    $\hat{\mathbf{x}}_{\text{EPGD}} \leftarrow x_i$
    for $k = 1$ to K **do**
      Compute gradients for Monte Carlo samples:
      $g \leftarrow 0$
      for $m = 1$ to M **do**
        $\boldsymbol{\eta}_{\mathbf{x}}^m \leftarrow \text{Sample}(\boldsymbol{\eta}_{\mathbf{x}})$
        $\boldsymbol{\eta}_{\boldsymbol{\theta}}^m \leftarrow \text{Sample}(\boldsymbol{\eta}_{\boldsymbol{\theta}})$
        $\hat{y}_i^m \leftarrow f_{\boldsymbol{\theta}+\boldsymbol{\eta}_{\boldsymbol{\theta}}^m}(\hat{\mathbf{x}}_{\text{EPGD}} + \boldsymbol{\eta}_{\mathbf{x}}^m)$
        $g \leftarrow g + \frac{1}{M} \nabla_{\mathbf{x}} \ell(\hat{y}_i^m, y_i)$
      end for
      Update $\boldsymbol{\delta}_{\text{EPGD}}$ and $\hat{\mathbf{x}}_{\text{EPGD}}$:
      $\boldsymbol{\delta}_{\text{EPGD}} \leftarrow \boldsymbol{\delta}_{\text{EPGD}} + \alpha \cdot \frac{g}{\|g\|_p}$
      $\hat{\mathbf{x}}_{\text{EPGD}} \leftarrow x_i + \epsilon \cdot \frac{\boldsymbol{\delta}_{\text{EPGD}}}{\|\boldsymbol{\delta}_{\text{EPGD}}\|_p}$
      $\hat{\mathbf{x}}_{\text{EPGD}} \leftarrow clip(\hat{\mathbf{x}}_{\text{EPGD}}, 0, 1)$
    end for
    $\boldsymbol{\eta}_{\boldsymbol{\theta}}^i \leftarrow \text{Sample}(\boldsymbol{\eta}_{\boldsymbol{\theta}})$
    $\hat{y}_i \leftarrow f_{\boldsymbol{\theta}+\boldsymbol{\eta}_{\boldsymbol{\theta}}^i}(\hat{\mathbf{x}}_{\text{EPGD}})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \cdot \nabla_{\boldsymbol{\theta}} \ell(\hat{y}_i, y_i)$
  end for
end for
return $\boldsymbol{\theta}$

We start from the expectation of the SVM objective over n data points:

$$\mathscr{L}_{SVM}^{\boldsymbol{\eta},\boldsymbol{\delta}} = \mathbb{E}_{\boldsymbol{\eta}} \max_{\boldsymbol{\delta}} \sum_{i=1}^{n} \text{ReLU}\Big[1 - y_i(\mathbf{w} \cdot (\mathbf{x}_i + \boldsymbol{\eta}_i - \boldsymbol{\delta}_i) + b)\Big], \tag{32}$$

where $\boldsymbol{\eta}$ is the injected noise and $\boldsymbol{\delta}$ is the adversarial noise. Applying Proposition 4 from [51] (with $\boldsymbol{\delta} = \boldsymbol{\delta} - \boldsymbol{\eta}$ and $r = 0$), we rewrite (32) as

$$\mathscr{L}_{SVM}^{\boldsymbol{\eta},\boldsymbol{\delta}} = \mathbb{E}_{\boldsymbol{\eta}} \max_{\boldsymbol{\delta}} \mathbf{w} \cdot (\boldsymbol{\delta} - \boldsymbol{\eta}) + \sum_{i=1}^{n} \text{ReLU}\Big[1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\Big] \tag{33}$$

Since the expectation of $\boldsymbol{\eta}$ is 0, the expectation $\mathbb{E}_{\boldsymbol{\eta}}\big[\mathbf{w} \cdot \boldsymbol{\eta}\big] = 0$, leading to

$$\mathscr{L}_{SVM}^{\boldsymbol{\eta},\boldsymbol{\delta}} = \max_{\boldsymbol{\delta}} \mathbf{w} \cdot \boldsymbol{\delta} + \sum_{i=1}^{n} \text{ReLU}\Big[1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\Big], \tag{34}$$

which is nothing but the SVM objective under adversarial attacks, without the injected noise.

Thus, the expectation of an adversarial attack will not change due to the noise injection, and even if the attacker is aware of the injected noise, it is unclear how to devise a better strategy to breach the defense effect provided by the noise injection.

5 Experiments

We now present an empirical evaluation of the proposed method. We start with a comparison of our method to previous art in multiple settings, followed by an extensive ablation study both of the smoothed attacks and the fine-tuning methods described in Section 4.2.

To investigate the effectiveness of the proposed method, we conducted experiments on CIFAR-10 and CIFAR-100

Table 1 Results on CIFAR-10 with ResNet-20 under PGD attack

Method	Accuracy, %		
	Clean	PGD-7	PGD-10
Adversarial training [33]†	83.04	42.44	36.76
PNI [20]	84.89	45.94	43.24†
CNI [58]	78.48	48.84	46.52
IAAT (trained on k=10) [3]	**87.26**	-	43.08
Randomized smoothing [37]+	67.92	45.88	45.32
Smooth adversarial training (no noise)	83.04	42.61	36.85
Smooth adversarial training	81.64	49.63	43.49
Smooth CNI (no noise)	84.46	53.31	49.79
Smooth CNI	81.98	**56.20**	**53.10**

The smoothed models use $M = 512$. $\sigma = 0.24$ for Smooth CNI, $\sigma = 0.3$ for smooth adversarial training, and $\sigma = 0.00$ for the no-noise settings in both cases. For comparison with [37] we applied the smoothing to their best-performing checkpoint provided by the authors, on ResNet-110. PGD-k refers to the PGD attack with k PGD iterations. † denotes our results based on code provided by the authors or our re-implementation. + denotes our results based on the checkpoint provided by the authors

Table 2 Results on CIFAR-100 with Wide-ResNet28-4 (top part), Wide-ResNet28-10 (middle) and Wide-ResNet34-10 (bottom) under FGSM and PGD attacks

Method	Accuracy, %			
	Clean	FGSM	PGD-10	PGD-20
Adversarial training [33]†	56.6	26.87	17.49	15.41
Smooth adversarial training	56.17	27.86	17.9	15.64
PNI [20]†	54.63	36.48	23.86	18.54
CNI [58]†	53.46	38.42	24.71	20.80
Smooth CNI (ours)	61.73	**44.51**	28.57	22.53
Feature Matching [29]	59.51	34.49	**31.68**	**31.11**
IAAT [3]	**68.1**	-	26.17	-
Free [39]	62.13	-	-	25.88

Smoothing uses $M = 64$ with $\sigma = 0.04$ for Smooth CNI, and $\sigma = 0.02$ for smooth adversarial training

[26] under white-box and black-box attacks for ResNet-20 [19], Wide-ResNet28-4, Wide-ResNet28-10 and Wide-ResNet34-10 [54].

Notations In all our experiments, we denote by M the number of samples used for Monte Carlo approximations of the smoothed model, σ the standard deviation for the input noise injection, i.e $\eta_x \sim \mathcal{N}(0, \sigma \cdot I)$, k the number of optimization iterations for the attacks and ϵ the norm limitation for the attacks.

Experimental settings We considered two different base models: adversarially-trained CNNs [33] and CNI [58]. While smoothing applied to adversarially-trained models improved adversarial robustness [37], using CNI as the base model provides significantly larger performance gains.

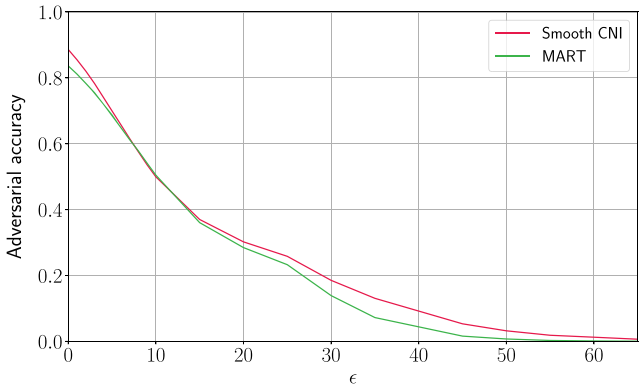


Fig. 1 Accuracy on CIFAR-10 under PGD attack as a function of ℓ_∞ radius of attack, ϵ_∞. The smooth CNI model is WideResNet-28-4, and the model for MART is WideResNet-34-10. The smoothing uses $M = 64$ and $\sigma = 0.25$. Both models were trained using PGD attacks with $k = 10$ and $\epsilon_\infty = 8/255$

Table 3 Comparison of our method to prior art on CIFAR-10 with Wide-ResNet28-4 (top part), Wide-ResNet28-10 (middle) and Wide-ResNet34-10 (bottom) under PGD attack

Method	Accuracy, %		
	Clean	PGD-10	PGD-20
Adversarial training [33]	86.08	40.36	36.10
PNI [20]†	84.54	55.52	47.29
CNI [58]†	84.44	55.32	47.7
Smooth CNI (ours)	85.81	**60.30**	52.08
Feature Matching [29]	85.21	-	52.95
Adversarial training [33]†	87.56	54.2	52.0
PNI [20]†	85.89	56.88	49.86
CNI [58]†	86.44	57.15	50.22
IAAT [3]	**91.3**	48.53	-
CSAT [38]	87.65	54.77	-
TRADES [55]+	84.92	54.8	53.16
RNN-TRADES [50]	84.21	56.35	-
DAT [44]	85.03	48.07	42.07
MART [45]+	83.62	57.3	**57.1**

Smooth CNI uses $M = 64$ with $\sigma = 0.22$ during inference. † denotes our results based on code provided by the authors or our re-implementation. + denotes our results based on the checkpoint provided by the authors

For ResNet-20 on CIFAR-10, we used the CNI as the base model trained adversarially for 400 epochs under PGD attack with $k = 7$. For Wide-ResNet28-4 on both CIFAR-10 and CIFAR-100, we used the CNI model trained adversarially for 100 epochs under PGD attack with $k = 10$. In all cases, we used the adversarial training suggested by [33] and chose the model with the highest adversarial performance on the validation set.

Table 4 Results on CIFAR-10 with Wide-ResNet28-4 under black-box transferable PGD attack with $k = 10$

Method		Accuracy, %		
M	σ	PGD	PGD-trans	PGD-s-trans
1	0	40.36	65.53	65.53
1	0	55.32	65.22	65.22
256	0.09	58.33	**69.04**	68.76
512	0.17	59.59	68.49	**69.61**
512	0.22	**60.30**	68.29	69.01

PGD-s-trans denotes the setting when the attack is generated on a model with smoothing; in this case the σ for both generating and testing the attack is the same. We presents results for adversarial training (top) and for CNI (bottom)

Table 5 Results on CIFAR-10 with Wide-ResNet28-4 under black-box NAttack [28]

Method	Accuracy, %
Adversarial training [33]†	43.75
MART [45]+	47.02
CNI [58]†	42.61
Smooth CNI (ours, $M = 8$)	56.76
Smooth CNI (ours, $M = 64$)	**57.00**

Smooth CNI uses $\sigma = 0.21$ for both models. † denotes our results based on code provided by authors or our re-implementation. + denotes our results based on the checkpoint provided by the authors

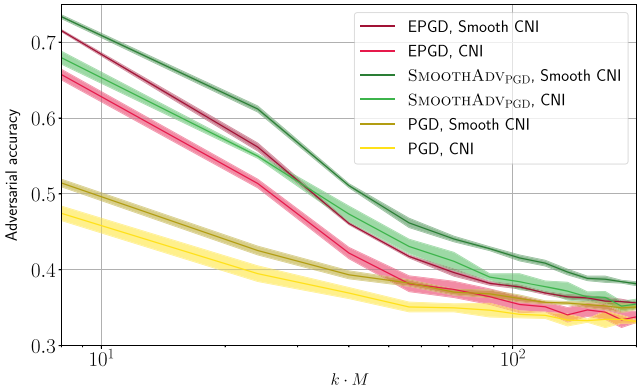


Fig. 3 Accuracy on CIFAR-10 under PGD, EPGD and SmoothAdv$_{PGD}$ attacks as a function of the number of attack steps, $k \cdot M$. The model is WideResNet-28-4 defended with CNI. The smoothing uses $M = 1, 8$ and $\sigma = 0.24$. EPGD and SmoothAdv$_{PGD}$ attacks averaged the gradients over $M_{backward} = 8$ samples in each attack iteration. All the attacks run with 3 random restarts. The shaded area is the standard deviation of the results over three runs

5.1 Comparison to other adversarial defenses

In each case, we compared our best-performing defense instance without fine-tuning (smooth CNI) to the current state-of-the-art. We do not include the fine-tuned models in the main comparison as they were trained on attacks with higher complexity than the current state-of-the-art. For all the results, unless stated otherwise, we use the ℓ_∞ radius of attack $\epsilon = 8/255$.

For ResNet-20 on CIFAR-10, our smooth CNI demonstrated a consistent improvement of ∼7% over the CNI base model, which is the best existing method for this case (Table 1). In addition, our smooth CNI substantially outperformed the randomized smoothing [37] baseline by ∼10.3%. For Wide-ResNet28-4 on CIFAR-10, our smooth CNI showed a consistent improvement of ∼5% over the CNI base model, outperforming the best existing method by ∼3% for $k = 10$ (Table 3). For Wide-ResNet on CIFAR-100, the effect of smoothing diminishes for higher k, with our smooth CNI substantially outperforming prior art for FGSM, but falling behind for PGD with $k = 10, 20$ (Table 2).

Next, we studied the effect of the value of ϵ on the PGD attack. Figure 1 demonstrates resilience of the smoothing effect for high-ϵ attacks with a CNI base model. The degradation of existing models is significantly stronger for $\epsilon > 20/255$. For example, smooth CNI achieves 13.1% accuracy for $\epsilon = 35/255$ and 5.3% accuracy for $\epsilon = 45/255$, outperforming current state-of-the-art by 6% and 3.7%, respectively. Moreover, a similar effect appears when using low values of ϵ. We conjecture that MART [45] is more sensitive to changes in ϵ between training and testing, thus showing similar accuracy for values close $\epsilon = 8/255$, but worse performance compared to smooth CNI otherwise (Table 3).

We tested our defense against black-box attacks: the transferable attack [32] and NAttack [28]. For the transferable attack, we trained another instance of the CNI base model and used it as a source model in two configurations: PGD with and without smoothing. We present the best-performing instances of smooth CNI defense (Tables 4

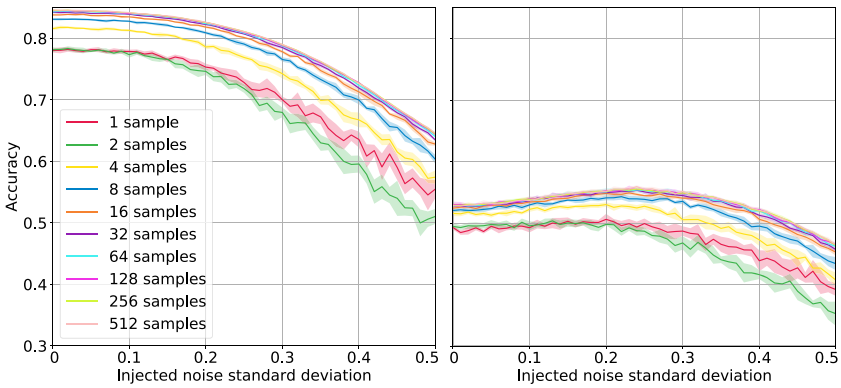


Fig. 2 Accuracy on CIFAR-10 as a function of injected noise strength on clean data (left) and under PGD attack (right). The model is ResNet-20 defended with CNI. The smoothing uses $M = 2^n, n \in \{0, \ldots, 9\}$. The shaded area is the standard deviation of the results over three runs

Table 6 Comparison of our method to prior art on CIFAR-10 (top) and CIFAR-100 (bottom) with Wide-ResNet28-4 under PGD attack, with and without fine-tuning

Method	Accuracy, %		
	Clean	PGD-10	PGD-20
Smooth CNI ($\sigma = 0.22$)	85.81	60.30	52.08
Smooth CNI ($k = 80$) ($\sigma = 0.28$)	87.24	64.30	55.64
Smooth CNI-EPGD ($\sigma = 0.3$)	**87.53**	**64.31**	55.19
MART [45]$^{+}$	83.62	57.3	57.1
MART ($k = 80$) †	83.94	59.4	**57.4**
Smooth CNI ($\sigma = 0.04$)	**61.72**	28.57	22.53
Smooth CNI ($k = 80$) ($\sigma = 0.21$)	55.14	33.34	28.42
Smooth CNI-EPGD ($\sigma = 0.22$)	54.13	**33.45**	28.60
Feature Matching [29]	59.51	31.68	**31.11**

Smooth CNI uses $M = 64$ for all models and the σ value is specified for each model. During the fine-tuning, $k = 10$ and $M = 8$ were used for the EPGD setting, and $k = 80$ for the high k setting. For comparison, we show the results for current state-of-the-art [29, 45] with and without fine-tuning on high k

and 5). For the transferable attack, we conclude that the performance of the model is independent of the smoothing of the source model. Our method showed a consistent improvement of ∼4% over the CNI base model. The NAttack comparison shows that our method substantially outperforms previous art and the baselines by at least 10%.

5.2 Ablation study

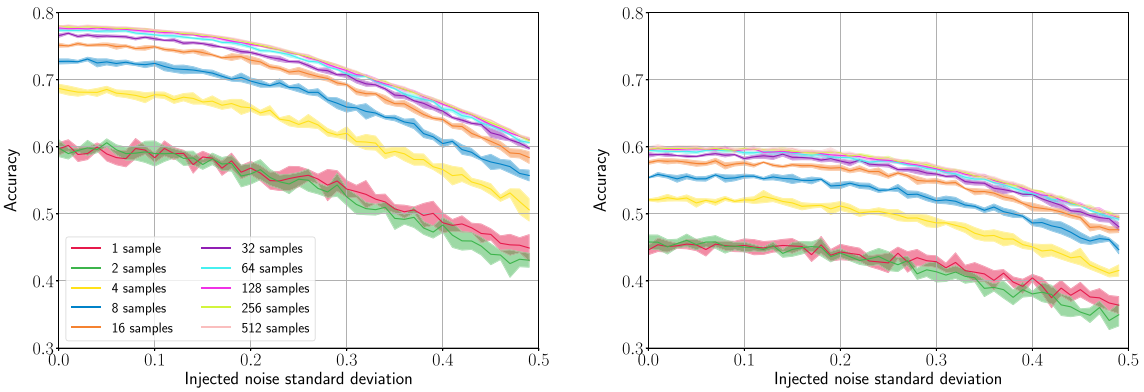
We now turn to an ablation study of our method. In Section 5.2.1 we show the dependence of smoothing on the sampling hyperparameters, namely, the number of samples M and the injected noise standard deviation. In Section 5.2.2 we compare the effectiveness of the PGD, EPGD and SMOOTHADV$_{\text{PGD}}$ [37] attacks on our smooth CNI model. In Section 5.2.3 we conduct a performance comparison of the smoothed models described in Section 4.2. In particular, we compare our fine-tuned models to the current state-of-the-art with similar fine-tuning.

5.2.1 Sampling

In Fig. 2 we compared a different number of Monte Carlo samples: $M = 2^n$, $n \in \{0, \ldots, 9\}$. For each value of M, we considered multiple noise standard deviation values in the range of [0, 0.5]. The selected upper bound corresponds to significant degradation of the model performance. We observe that the accuracy increases for a larger M across all values of noise variance. Accuracy of all models on clean data drops as the noise variance increases, with maximal clean accuracy achieved for almost zero noise variance. In addition, all the models achieve their maximal adversarial accuracy for the same value of noise variance, for which most values of M show improvement in clean accuracy when compared to a single sample ($M = 1$) defense without noise. Hence, smoothing improves both clean and adversarial robustness, with the clean-adversarial trade-off controlled by the noise variance.

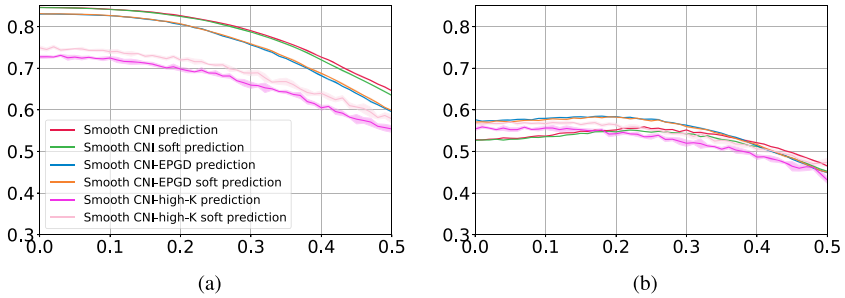
5.2.2 Comparison of attacks

In Fig. 3 we compared the adversarial accuracy of the PGD, EPGD and SMOOTHADV$_{\text{PGD}}$ [37] attacks on the CNI base model and our smooth CNI with $M = 8$ samples. Since EPGD and SMOOTHADV$_{\text{PGD}}$ attacks with k PGD steps and M samples require computing the gradients $k \cdot M$ times, these attacks require an amount of computation similar to a PGD attack with $k \cdot M$ steps. Therefore, we compare the



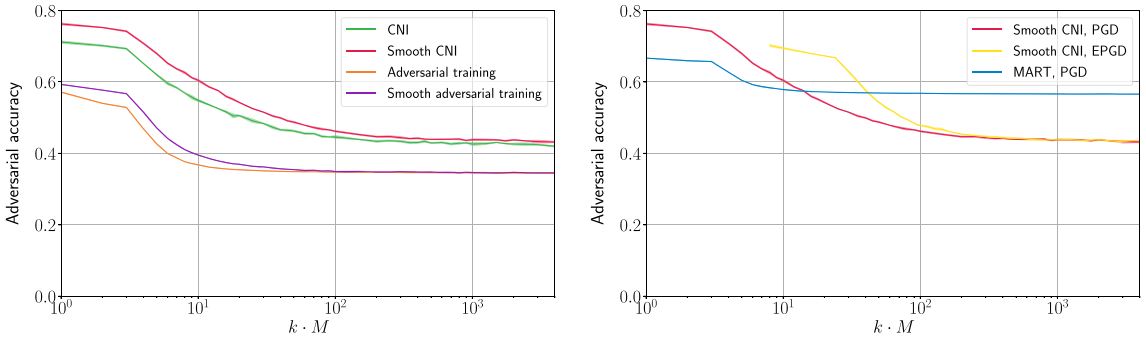
(a) (b)

Fig. 4 Accuracy on CIFAR-10 as a function of injected noise strength on clean data (left) and under PGD attack (right). The model is ResNet-20 defended with CNI and fine-tuned on PGD with $k = 56$. The smoothing uses $M = 2^n$, $n \in \{0, \ldots, 9\}$. The shaded area is the standard deviation of the results over three runs



(a) (b)

Fig. 5 Accuracy on CIFAR-10 as a function of injected noise strength on clean data (left) and under PGD attack (right). The model is ResNet-20 defended with CNI, with different smoothing and fine-tuning methods. The smoothing uses $M = 8$ for all the methods. All the attacks run with 3 random restarts. The shaded area is the standard deviation of the results over three runs



(a) (b)

Fig. 6 Accuracy on CIFAR-10 under PGD or EPGD attacks as a function of the number of attack steps, $k \cdot m$. The model for MART is WideResNet-34-10 and the rest are WideResNet-28-4; all models are fine-tuned on PGD with $k = 80$. EPGD attacks are only used when specified, and the gradients are averaged over $M_{\text{backward}} = 8$ samples in each attack iteration. The smoothing uses $M = 8$ and $\sigma = 0.25$. All the attacks run with 3 random restarts. The shaded area is a standard deviation of the results over three runs

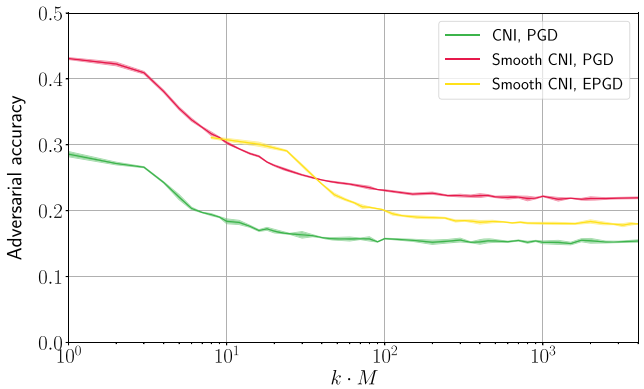


Fig. 7 Accuracy on CIFAR-100 under PGD or EPGD attacks as a function of the number of attack steps, $k \cdot M$. The model is WideResNet-28-4 defended with CNI and fine-tuned on EPGD with $k = 10$ and $M = 8$. EPGD attacks are only used when specified, and the gradients are averaged over $M_{\text{backward}} = 8$ samples in each attack iteration. The smoothing uses $M = 8$ and $\sigma = 0.25$. All the attacks run with 3 random restarts. The shaded area is a standard deviation of the results over three runs

adversarial accuracy as a function of $k \cdot M$ (where for PGD, $M = 1$). We find that EPGD is comparable, but not better than PGD for large $k \cdot M$, and worse than PGD for lower $k \cdot M$. Moreover, we find that SMOOTHADV$_{\text{PGD}}$ is even less effective, resulting in higher accuracy than either PGD or EPGD on both models for all values of $k \cdot M$.

PGD attacks are the most effective for lower $k \cdot M$, but converge to the same value as EPGD attacks without

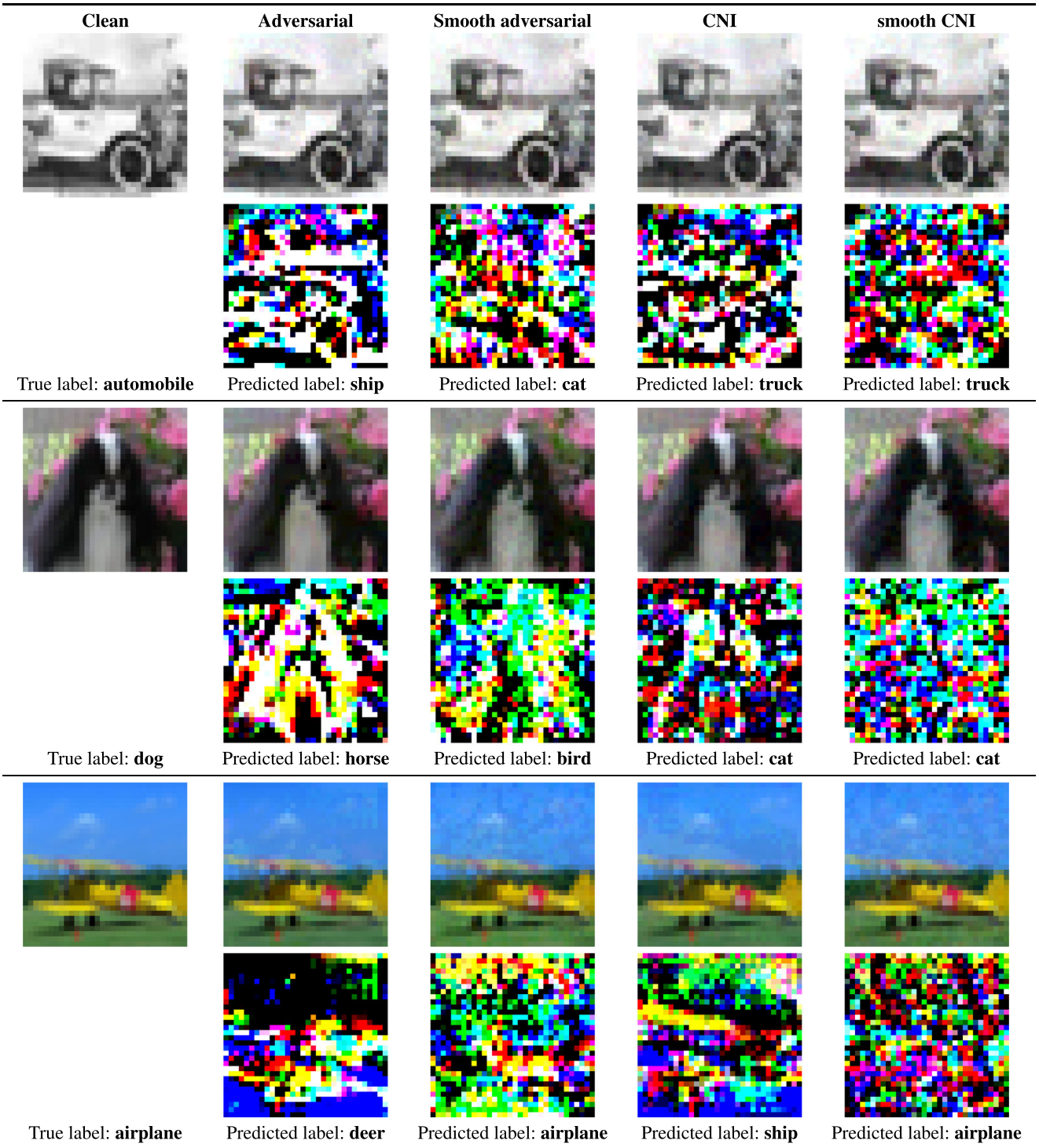


Fig. 8 Visualization of attacks. For each attack we present the adversarial image produced via PGD attack, the normalized perturbation ($\Delta \mathbf{x}/\epsilon$), and the predicted label

smoothing. In contrast, SMOOTHADV$_{\text{PGD}}$ converges to higher accuracy for both models. These results could indicate that EPGD makes better use of the information encoded in the multiple samples, so much so that it produces an attack comparable to PGD with $1/M$ of the required iterations. As such, SMOOTHADV$_{\text{PGD}}$ is less useful than EPGD for empirical evaluation, and we continue to compare EPGD with PGD.

In addition, for all three attacks types, the accuracy on smooth CNI is higher when compared to the base model. The improvement, however, is more substantial for EPGD and SMOOTHADV$_{\text{PGD}}$, suggesting that EOT [1] or similar attacks do not produce a significant decrease in the adversarial accuracy of our smoothed model. Therefore, while some of our method's effect may result from random gradient obfuscation, the obfuscation cannot be easily exploited to undermine our method.

5.2.3 Fine-tuning

In addition to the adversarially-trained CNI [58], we trained two models obtained by fine-tuning the CNI base model, one of which was trained with the soft smoothing training, as described in Section 4.2. For comparison, the second model was similarly fine-tuned with a PGD attack. For the PGD attack, we multiply k by M to get a computational complexity similar to EPGD. We chose to limit the number of samples of the EPGD attack to $M = 8$, since any further improvement of the resulting model is offset by the increase in computational complexity.

We compare the effect of fine-tuning on our models on CIFAR-10 and CIFAR-100. Surprisingly, the results for both types of fine-tuned models were almost identical. As summarized in Table 6, in terms of adversarial accuracy, the smooth fine-tuned models improved by ∼4% on CIFAR-10 and ∼5–6% on CIFAR-100. In contrast, for MART, fine-tuning showed only a ∼2% improvement on CIFAR-10. To conclude, our fine-tuned models significantly outperform the fine-tuned MART [45] and Feature Matching [29] models, achieving state-of-the-art results for $k = 10$ on both CIFAR-10 and CIFAR-100.

In Fig. 4, we evaluated the impact of noise injection variance for different numbers of samples on a ResNet-20 model defended by CNI and fine-tuned on PGD with $k = 56$. We conclude that both clean and adversarial accuracy are maximized for zero or almost zero variance, in contrast to Fig. 2 where the adversarial accuracy is maximized at non-zero variance.

Figure 5 demonstrates that there is a single optimal value of noise strength for each base model, independent of the smoothing method. This indicates that for each such model, there should be an optimal variance of the injected noise.

In Figs. 6 and 7, we show the decay of the smoothing effect for high k attacks both with a CNI and adversarial training as the base model. Due to this decay, the significant performance boost for lower values of k vanishes for larger values of k. Moreover, the same effect appears for the CNI base model, even without the application of smoothing. For CIFAR-100, however, the gap between CNI and smooth CNI is not entirely closed, even with the extreme value of $k = 4000$. In addition, while for CIFAR-10, EPGD shows the same performance as PGD given enough iterations, for CIFAR-100, EPGD manages to break the smooth defense in instances where PGD is incapable of doing so (Fig. 8).

We define a minimal k to break the defense, k_c, as a value for which adversarial accuracy is at most 0.5% higher than the minimum value (as a function of k). For adversarial training on CIFAR-10, this value is $k_c = 40$, while with smoothing, it increases to $k_c = 100$. CNI pushes the minimum further ($k_c = 500$ without smoothing and $k_c = 900$ with smoothing). MART, on the other hand, is barely affected by the increased number of iterations, with $k_c = 25$.

6 Conclusions

This paper proposed a novel inference-only method for improving empirical adversarial robustness by taking into account the spatial information in the area of the data points. We showed that the proposed method could manage changes in ϵ better than previous state-of-the-art approaches, such as MART. As our theoretical results in Section 4.3 suggest, however, the smoothing performance is highly dependent on k, the number of the attack iterations. Applying smoothing over a CNI base model achieved state-of-the-art results for FGSM and PGD with $k \leqslant 10$ on CIFAR-10 with a decline in robustness for large values of k. In addition, smoothing shows high efficiency against black-box attacks, achieving a significant increase over the current state-of-the-art for NAttack on CIFAR-10 as well as improvement over the CNI base model for transferable attacks. For both white-box and black-box attacks, these results require only a small number of samples, M.

We conclude that smoothing offers a practical trade-off between the inference time and model performance for these configurations. In addition, while smoothing can be incorporated into any adversarial defense, we have seen that for the CNI base model [58], is essential for it to be effective, which aligns with our theoretical results in (28).

Nevertheless, as there is a noticeable decline in the performance for attacks with higher k values, we believe that some of our method's effect may indeed result from random gradient obfuscation. The usual ways of overcoming gradient obfuscation, such as estimating the expectation

of the gradient [1], however, did not produce a significant decrease in adversarial accuracy without a substantial increase in computational complexity. Moreover, we note that the same effect appears in the base models we used [20, 58] and, to the best of our knowledge, no efficient attack to exploit this kind of obfuscation has been proposed yet.

Acknowledgements The research was funded by the Hyundai Motor Company through the HYUNDAI-TECHNION-KAIST Consortium, National Cyber Security Authority, and the Hiroshi Fujiwara Technion Cyber Security Research Center.

References

1. Athalye A, Carlini N, Wagner D (2018a) Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: Dy J, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholmsmässan, vol 80. Proceedings of Machine Learning Research, Stockholm Sweden, pp 274–283. http://proceedings.mlr.press/v80/athalye18a.html
2. Athalye A, Engstrom L, Ilyas A, Kwok K (2018b) Synthesizing robust adversarial examples. In: Dy J, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning, PMLR, vol 80. Proceedings of Machine Learning Research, Stockholmsmässan, pp 284–293. http://proceedings.mlr.press/v80/athalye18b.html
3. Balaji Y, Goldstein T, Hoffman J (2019) Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. arXiv:1910.08051
4. Balunovic M, Vechev M (2020) Adversarial training and provable defenses: Bridging the gap. In: International Conference on Learning Representations, https://openreview.net/forum?id=SJxSDxrKDr
5. Bietti A, Mialon G, Chen D, Mairal J (2019) A kernel perspective for regularizing deep neural networks. In: International Conference on Machine Learning. PMLR, pp 664–674
6. Blum A, Dick T, Manoj N, Zhang H (2020) Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. arXiv:2002.03517
7. Brown TB, Mané D, Roy A, Abadi M, Gilmer J (2017) Adversarial patch. arXiv:1712.09665
8. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp 39–57. https://doi.org/10.1109/SP.2017.49
9. Carlini N, Wagner D (2018) Audio adversarial examples: Targeted attacks on speech-to-text. In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE, pp 1–7
10. Chaturvedi A, KP A, Garain U (2019) Exploring the robustness of nmt systems to nonsensical inputs. arXiv:1908.01165
11. Chen PY, Zhang H, Sharma Y, Yi J, Hsieh CJ (2017) ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec '17, ACM, New York, pp 15–26. https://doi.org/10.1145/3128572.3140448
12. Cohen J, Rosenfeld E, Kolter Z (2019) Certified adversarial robustness via randomized smoothing. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th International Conference on Machine Learning, PMLR, vol 97. Proceedings of Machine Learning Research, Long Beach, pp 1310–1320. http://proceedings.mlr.press/v97/cohen19c.html
13. Gao J, Lanchantin J, Soffa ML, Qi Y (2018) Black-box generation of adversarial text sequences to evade deep learning classifiers. In: 2018 IEEE Security and Privacy Workshops (SPW) IEEE, pp 50–56
14. Gilmer J, Metz L, Faghri F, Schoenholz SS, Raghu M, Wattenberg M, Goodfellow I (2018) Adversarial spheres. arXiv:180102774
15. Gilmer J, Ford N, Carlini N, Cubuk E (2019) Adversarial examples are a natural consequence of test error in noise. In: International Conference on Machine Learning, PMLR, pp 2280–2289
16. Gleave A, Dennis M, Wild C, Kant N, Levine S, Russell S (2020) Adversarial policies: Attacking deep reinforcement learning. In: International Conference on Learning Representations, https://openreview.net/forum?id=HJgEMpVFwB
17. Gong C, Ren T, Ye M, Liu Q (2020) Maxup: A simple way to improve generalization of neural network training. arXiv:2002.09024
18. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv:1412.6572
19. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
20. He Z, Rakin AS, Fan D (2019) Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), http://openaccess.thecvf.com/content_CVPR_2019/html/He_Parametric_Noise_Injection_Trainable_Randomness_to_Improve_Deep_Neural_Network_CVPR_2019_paper.html
21. Jiang H, Chen Z, Shi Y, Dai B, Zhao T (2018) Learning to defense by learning to attack. arXiv:1811.01213
22. Jin D, Jin Z, Zhou JT, Szolovits P (2020) Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 8018–8025
23. Khoury M, Hadfield-Menell D (2019) Adversarial training with voronoi constraints. arXiv:1905.01019
24. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv:1312.6114
25. Kingma DP, Salimans T, Welling M (2015) Variational dropout and the local reparameterization trick. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in Neural Information Processing Systems, vol 28. Curran Associates, Inc., pp 2575–2583. http://papers.nips.cc/paper/5666-variational-dropout-and-the-local-reparameterization-trick.pdf
26. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
27. Kumar A, Levine A, Goldstein T, Feizi S (2020) Curse of dimensionality on randomized smoothing for certifiable robustness. arXiv:2002.03239
28. Li Y, Li L, Wang L, Zhang T, Gong B (2019) NATTACK: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th International Conference on Machine Learning, PMLR, vol 97. Proceedings of Machine Learning Research, Long Beach, pp 3866–3876. http://proceedings.mlr.press/v97/li19g.html
29. Li Z, Feng C, Zheng J, Wu M, Yu H (2020) Towards adversarial robustness via feature matching. IEEE Access https://ieeexplore.ieee.org/abstract/document/9089860/

30. Liu A, Liu X, Yu H, Zhang C, Liu Q, Tao D (2021) Training robust deep neural networks via adversarial noise propagation. IEEE Trans Image Process 30:5769–5781
31. Liu X, Cheng M, Zhang H, Hsieh CJ (2018) Towards robust neural networks via random self-ensemble. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 369–385
32. Liu Y, Chen X, Liu C, Song D (2016) Delving into transferable adversarial examples and black-box attacks. arXiv:1611.02770
33. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations, https://openreview.net/forum?id=rJzIBfZAb
34. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, ACM, New York, pp 506–519. https://doi.org/10.1145/3052973.3053009
35. Pinot R, Meunier L, Araujo A, Kashima H, Yger F, Gouy-Pailler C, Atif J (2019) Theoretical evidence for adversarial robustness through randomization. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds), vol 32. Advances in Neural Information Processing Systems. Curran Associates, Inc. https://papers.nips.cc/paper/2019/hash/36ab62655fa81ce8735ce7cfdaf7c9e8-Abstract.html
36. Rony J, Hafemann LG, Oliveira LS, Ayed IB, Sabourin R, Granger E (2019) Decoupling direction and norm for efficient gradient-based L2 adversarial attacks and defenses. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
37. Salman H, Yang G, Li J, Zhang P, Zhang H, Razenshteyn I, Bubeck S (2019) Provably robust deep learning via adversarially trained smoothed classifiers. arXiv:1906.04584
38. Sarkar A, Gupta NK, Iyengar R (2019) Enforcing linearity in dnn succours robustness and adversarial image generation. arXiv:1910.08108
39. Shafahi A, Najibi M, Ghiasi MA, Xu Z, Dickerson J, Studer C, Davis LS, Taylor G, Goldstein T (2019) Adversarial training for free!. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché-Buc F, Fox E, Garnett R (eds) Advances in Neural Information Processing Systems, vol 32. Curran Associates, Inc., pp 3358–3369. http://papers.nips.cc/paper/8597-adversarial-training-for-free
40. Sun H, Wang R, Chen K, Utiyama M, Sumita E, Zhao T (2020) Robust unsupervised neural machine translation with adversarial training. arXiv:2002.12549
41. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. arXiv:1312.6199
42. Wang B, Yuan B, Shi Z, Osher SJ (2020a) Enresnet: Resnets ensemble via the feynman–kac formalism for adversarial defense and beyond. SIAM J Math Data Sci 2(3):559–582
43. Wang D, Li C, Wen S, Nepal S, Xiang Y (2019a) Daedalus: Breaking non-maximum suppression in object detection via adversarial examples. arXiv:1902.02067
44. Wang Y, Ma X, Bailey J, Yi J, Zhou B, Gu Q (2019b) On the convergence and robustness of adversarial training. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th International Conference on Machine Learning, PMLR, vol 97. Proceedings of Machine Learning Research, Long Beach, pp 6586–6595. http://proceedings.mlr.press/v97/wang19i.html
45. Wang Y, Zou D, Yi J, Bailey J, Ma X, Gu Q (2020b) Improving adversarial robustness requires revisiting misclassified examples. In: International Conference on Learning Representations, https://openreview.net/forum?id=rklOg6EFwS
46. Wong E, Kolter Z (2018) Provable defenses against adversarial examples via the convex outer adversarial polytope. In: Dy J, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning, PMLR, vol 80. Proceedings of Machine Learning Research, Stockholmsmässan, pp 5286–5295. http://proceedings.mlr.press/v80/wong18a.html
47. Xiang C, Qi CR, Li B (2019) Generating 3d adversarial point clouds. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
48. Xie C, Tan M, Gong B, Wang J, Yuille A, Le QV (2019a) Adversarial examples improve image recognition. arXiv:1911.09665
49. Xie C, Wu Y, Maaten Lvd, Yuille AL, He K (2019b) Feature denoising for improving adversarial robustness. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 501–509
50. Xiong Y, Hsieh CJ (2020) Improved adversarial training via learned optimizer. arXiv:2004.12227
51. Xu H, Caramanis C, Mannor S (2009) Robustness and regularization of support vector machines. J Mach Learn Res 10(51):1485–1510. http://jmlr.org/papers/v10/xu09b.html
52. Xu K, Zhang G, Liu S, Fan Q, Sun M, Chen H, Chen PY, Wang Y, Lin X (2019) Evading real-time person detectors by adversarial t-shirt. arXiv:1910.11099
53. Yang G, Duan T, Hu E, Salman H, Razenshteyn I, Li J (2020) Randomized smoothing of all shapes and sizes. arXiv:2002.08118
54. Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv:1605.07146
55. Zhang H, Yu Y, Jiao J, Xing E, Ghaoui LE, Jordan M (2019) Theoretically principled trade-off between robustness and accuracy. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of Machine Learning Research, vol 97, Long Beach, pp 7472–7482. http://proceedings.mlr.press/v97/zhang19p.html
56. Zhang H, Chen H, Xiao C, Gowal S, Stanforth R, Li B, Boning D, Hsieh CJ (2020) Towards stable and efficient training of verifiably robust neural networks. In: International Conference on Learning Representations, https://openreview.net/forum?id=Skxuk1rFwB
57. Zhang Y, Liang P (2019) Defending against whitebox adversarial attacks via randomized discretization. In: Chaudhuri K, Sugiyama M (eds) Proceedings of Machine Learning Research, vol 89. PMLR, Proceedings of Machine Learning Research, pp 684–693. http://proceedings.mlr.press/v89/zhang19b.html
58. Zheltonozhskii E, Baskin C, Nemcovsky Y, Chmiel B, Mendelson A, Bronstein AM (2020) Colored noise injection for training adversarially robust neural networks. arXiv:2003.02188
59. Zheng S, Song Y, Leung T, Goodfellow I (2016) Improving the robustness of deep neural networks via stability training. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4480–4488. https://doi.org/10.1109/CVPR.2016.485
60. Zheng T, Wang D, Li B, Xu J (2020) Towards assessment of randomized mechanisms for certifying adversarial robustness. arXiv:2005.07347

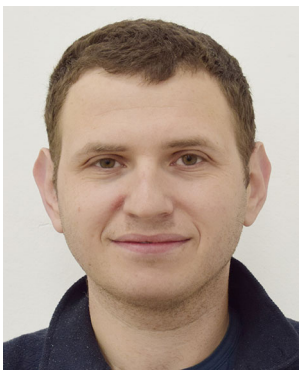
Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Yaniv Nemcovsky a Ph.D. candidate at the CS department, Technion. Previously was a Master's student at the same department, and holds a Master's on the topic of generalization of deep Bayesian networks. His industry experience includes Software engineering positions at Intel. Interested in multiple fields of deep learning, including adversarial attacks, deep learning theory, and computer vision.



Evgenii Zheltonozhskii is a Physics PhD candidate at Technion, working on condensed matter and in particular topological matter. Evgenii holds M.Sc. in Computer Csience from Technion, in which he researched reduced supervision methods for computer vision, including self-supervised, semi-supervised learning and learning with noisy labels.



Dr. Chaim Baskin a Senior Research Associate at CS department. Holds PhD in Machine Learning from CS department, Technion. Previously he was post-doc fellow at the same department. His industry experience includes Deep Learning research positions at Intel, Intel Labs. He Published 22 papers and preprints in multiple fields of deep learning, including neural network compression, self-supervised learning, computer vision, and Graph Neural Networks. Chaim acted as co-PI/Lead researcher at several successful research collaborations with leading companies such Nvidia,Intel, Hyundai, Ceva, Mafat.

Alex M. Bronstein (Fellow, IEEE) is currently a Professor of computer science with the Technion-Israel Institute of Technology and a Principal Engineer with Intel Corporation. His research interests include numerical geometry, computer vision, and machine learning. He has authored over 100 publications in leading journals and conferences, over 30 patents and patent applications, the research monograph Numerical Geometry of Non-Rigid Shapes, and edited several books. Highlights of his research were featured in CNN, SIAM News, and Wired. In addition to his academic activity, he co-founded and served as the Vice President of technology in the Silicon Valley start-up company Novafora, from 2005 to 2009. He was a Co-Founder and one of the main inventors and developers of the 3D sensing technology with the Israeli startup Invision, subsequently acquired by Intel, in 2012. His technology is currently the core of the Intel RealSense 3D camera integrated into a variety of consumer electronic products. He is also a Co-Founder of the Israeli video search startup Videocites and the London-based startup Sibylla, where he serves as the Chief Scientist. He is a Fellow of the IEEE for his contribution to 3D imaging and geometry processing.

Avi Mendelson is a professor in the Computer Science and Electrical Engineering Departments at Technion, Israel Institute of Technology. His research interests include computer architectures, distributed systems, hardware security, and accelerators for machine learning. Mendelson received his Ph.D. in computer engineering from the University of Massachusetts, Amherst. He is a Fellow of the IEEE. Contact him at avi.mendelson@technion.ac.il.