A Curious Case of Remarkable Resilience to Gradient Attacks via Fully Convolutional and Differentiable Front End with a Skip Connection

Anonymous authors
Paper under double-blind review

Abstract

We experimented with front-end enhanced neural models where a frozen backbone classifier was prepended by a differentiable and fully convolutional model with a skip connection. By training such composite models using a small learning rate for one epoch (or less), we obtained models that retained the accuracy of the backbone classifier while being unusually resistant to gradient attacks including APGD and FAB-T attacks from the AutoAttack package.

We provided evidence that this was due to gradient masking: Although the gradient masking phenomenon is not new, the degree of masking was quite remarkable for fully differentiable models that did not have gradient-shattering components such as JPEG compression or components that are expected to cause diminishing gradients. The training recipe to produce such models was remarkably stable and reproducible as well: We applied it to three datasets (CIFAR10, CIFAR100, and ImageNet) and several types of models (including recently proposed vision Transformers) without a single failure case.

Although black box attacks such as the SQUARE attack and the zero-order PGD can be partially effective against gradient masking, these attacks are easily defeated by combining gradient-masking models into simple randomized ensembles. We estimate that these ensembles achieve near-SOTA AutoAttack accuracy on CIFAR10, CIFAR100, and ImageNet (while retaining virtually all the clean accuracy of the original classifiers) despite having virtually zero accuracy under adaptive attacks. Quite interesting, adversarial training of the backbone classifier can further increase resistance of the front-end enhanced model to gradient attacks. On CIFAR10, the respective randomized ensemble achieved 90.8±2.5% (99% CI) accuracy under AutoAttack while having only 18.2±3.6% accuracy under the adaptive attack.

We do not aim to establish SOTA in adversarial robustness. Instead, our paper makes methodological contributions and further supports the thesis that adaptive attacks designed with the *complete* knowledge of model architecture are crucial in demonstrating model robustness and that even the so-called white-box gradient attacks can have limited applicability. Although gradient attacks can be complemented with black-box attack such as the SQUARE attack or the zero-order PGD, black-box attacks can be weak against randomized ensembles, e.g., when ensemble models mask gradients.

Code and instructions to reproduce key results is available. `https://anonymous.4open.science/r/curious_case_of_gradient_masking-2D3E`

1 Introduction

In the field of adversarial machine learning, the AutoAttack package has gained quite a bit of popularity as a standard approach to evaluate model robustness. According to Google Scholar, by the end of 2023 core AutoAttack papers (Croce & Hein, 2020a;b; Andriushchenko et al., 2020) were cited about 2500 times.

The AutoAttack is the "engine" of the RobustBench benchmark Croce et al. (2021), which is a popular benchmark tracking progress in adversarial machine learning. However, there has been little to no discussion in the literature about reliability of AutoAttack itself as well as its individual "components". Tramèr et al. (2020) noted in passing that AutoAttack overestimated robust accuracy of several models, which had nearly zero accuracy under adaptive attacks, but they did not discuss the actual failure modes of the AutoAttack.

To partially address this gap, we carry out a case study where the AutoAttack is applied to front-end "enhanced" models. We found such composite models as well as their randomized ensembles to be surprisingly resistant to standard projected gradient (PGD) attacks as well as to gradient attacks from the AutoAttack package (Croce & Hein, 2020a), which we attributed to gradient masking (and provided supporting evidence). This includes the FAB attack that was claimed to be resilient to gradient masking (Croce & Hein, 2020b). Despite the fact that RobustBench does not accept randomized models (Croce et al., 2021), users may apply AutoAttack to randomized defenses unaware that AutoAttack can fail against them (in fact, this was our experience). Moreover, the AutoAttack package was shown to be effective against complicated randomized defences (Croce & Hein, 2020a). Thus, it was reasonable to expect the AutoAttack to be successful against a simple randomized ensemble as well. When AutoAttack detects a randomized defense it only produces an easy-to-miss warning and suggests using the randomized version of the attack. In that, the randomized version of the AutoAttack has only gradient attacks, which—we find—can be *ineffective* in the presence of gradient masking.

Although the gradient masking phenomenon is not new, the degree of this masking is quite remarkable for fully differentiable models that do not have gradient-shattering components such as JPEG compression or components that are expected to cause diminishing gradients (we also carried out sanity checks to ensure there were no numerical anomalies, see § A.1). In particular, the models retain full accuracy on original (non-adversarial) data, but a "basic" 50-step PGD with 5 restarts degrades performance of a front-end enhanced model by only a few percent, despite it is possible to circumvent gradient masking with nearly 100% success rate (using a variant of the backward pass differentiable approximation Athalye et al. 2018). Although black box attacks such as the SQUARE attack and the zero-order PGD can be effective against gradient masking, these attacks are easily defeated by combining gradient-masking models into simple randomized ensembles. We estimate that these ensembles achieve near-SOTA AutoAttack accuracy on CIFAR10, CIFAR100, and ImageNet despite having virtually zero accuracy under adaptive attacks. Quite interesting, adversarial training of the backbone classifier can further increase resistance of the front-end enhanced model to gradient attacks. On CIFAR10, the respective randomized ensemble achieved 90.8±2.5% (99% CI) accuracy under AutoAttack.

We do not aim to establish SOTA in adversarial robustness. Instead, our paper makes methodological contributions and further supports the thesis that adaptive attacks designed with the *complete* knowledge of model architecture are crucial in demonstrating model robustness (Tramèr et al., 2020). We nevertheless believe that automatic evaluation is a useful baseline and we hope that our findings can help improve robustness and reliability of such evaluations. Code and instructions to reproduce key results is available.[1]

2 Related Work

Our paper focuses on a specific sub-field concerned with test-time attacks and defenses for neural-network classifiers on image data, which dates back only to 2013 (Szegedy et al., 2014; Biggio et al., 2013). For a detailed overview we address the reader to recent surveys (Chakraborty et al., 2021; Akhtar et al., 2021). However, a broader field of adversarial machine learning has a longer history with earlier applications such as spam and intrusion detection (Dalvi et al., 2004; Lowd & Meek, 2005; Biggio & Roli, 2018).

Originally, most practical classifiers were linear and despite successes in attacking them there was a belief that attacking complex and non-linear models would be difficult (Biggio & Roli, 2018). This belief was disproved by Biggio et al. (2013) and Szegedy et al. (2014). As Biggio & Roli (2018) noted, adversarial machine learning gained quite a bit of momentum after the publication by Szegedy et al. (2014). In addition

[1] `https://anonymous.4open.science/r/curious_case_of_gradient_masking-2D3E`

to showing that deep learning networks were vulnerable to small adversarial perturbations, Szegedy et al. (2014) found that these perturbations were (to a large degree) *transferable* among models.

Since then, the field has been growing exponentially, with more than eight thousand papers uploaded to arXiv from the beginning of 2018 to the end of 2023 (Carlini, 2019). Despite such an explosive growth, adversarial training is arguably the only reliable defense: which, nevertheless, does *not* provide theoretical guarantees: Some assurances can be provided by provable defenses, but they have limited applicability (Croce & Hein, 2020a; Akhtar et al., 2021).

Quite a few additional empirical (sometimes sophisticated) defenses have been proposed, but they have been routinely broken using strong adaptive attacks (Athalye et al., 2018; Tramèr et al., 2020), which motivated development of automatic evaluation suits, which can reduce manual effort often required to execute adaptive attacks. Among them, AutoAttack (Croce & Hein, 2020b) is arguably the most popular package, which includes several parameter-free gradient attacks (including a newly proposed FAB (Croce & Hein, 2020b) as well as a query-efficient black-box SQUARE attack (Andriushchenko et al., 2020). About one year later Croce et al. (2021) created the RobustBench benchmark to track progress and adversarial machine learning where adversarial robustness is measured using the accuracy under AutoAttack.

Despite Croce & Hein (2020a) advertised AutoAttack as a reliable attack, Tramèr et al. (2020) noted in passing that AutoAttack overestimated robust accuracy of models in several cases, but without providing details. In that, research on weaknesses of AutoAttack appears to be scarce. For example, Lorenz et al. (2021) question the general transferrability of RobustBench findings to more realistic settings, but not reliability of the attack itself. In particular, our finding regarding extreme gradient masking appears to be unknown. Methodologically, the most related paper is a work by Tramèr et al. (2020), who emphasize the importance of adaptive attacks. The main difference is that we focus on documenting one specific use of (extreme) gradient masking and simple randomization.

Finally we note that gradient masking is typically associated with gradient shattering via non-differentiable components and randomization as well as components encouraging vanishing gradients (Athalye et al., 2018), but not with fully differentiable components with skip connections such the denoising front end used in our study (see see Figure 1 in § 4.1). We found only one case when a fully differentiable convolutional network became more resistant to PGD attacks as a result of using a special training procedure, namely, training with differential privacy (Boenisch et al., 2021; Tursynbek et al., 2020). However, the seemingly increased robustness was observed only on MNIST LeCun et al. (2010) and in the low accuracy zone (see Fig. 1a in the paper by Boenisch et al. 2021), whereas in our case gradient-masked models achieve accuracy of about 80-90% under a standard PGD attack while remaining full breakable (see Table 1).

3 Background and Notation

3.1 Preliminaries

We consider a c-class image classification problem with a neural network $h_\theta(x)$, where θ is a vector of network parameters. We will also use $f_\theta(x)$ to denote a special front-end network, which is optionally prepended to a standard vision classification backbone such as ResNet (He et al., 2016). Each input image is a vector x of dimensionality $d = w \cdot h \cdot 3$ (width × height × the number of color channels), which "resides" in a normed vector space. In our work we use exclusively the L^∞ norm and use $\mathcal{B}_\varepsilon(x)$ to denote an L^∞ ball of the radius ε centered at x.

Given input x, the network estimates class-membership probabilities of input x by producing a vector of dimensionality c whose elements are called soft labels:

$$h_\theta(x) : \mathbb{R}^d \to [0, 1]^c$$

The actual prediction (a hard label) is taken to be the class with the highest probability:

$$\arg\max_i h_\theta(x)_i$$

For training purposes the goodness of prediction is measured using a loss function L. The value of the loss function L for input x and true output y is denoted as:

$$L(h_\theta(x), y)$$

In this work—as it is common for classification problems—we use exclusively the cross-entropy loss.

3.2 Adversarial Example and Threat Models

Adversarial examples are *intentionally* (and typically *inconspicuously*) modified inputs designed to fool a machine learning model. In the untargeted attack scenario, the adversary does not care about specific class outcomes as long as the model can be tricked to make an incorrect prediction. In the targeted attack scenario, the attacker objective is to make the model predict a specific class, e.g., to classify all images of pedestrians as motorcycles. The original unmodified inputs are often termed as clean data. The accuracy of the model on clean data is typically termed as *clean* accuracy. In contrast, the accuracy of the model on the adversarial examples is termed as *robust* accuracy.

In this paper, we consider adversarial input perturbations whose L^∞ norm does not exceed a dataset-specific threshold ε: In other words, adversarial examples must be inside the ball $\mathcal{B}_\varepsilon(x)$, where x is the original input. Different types of attacks require different levels of knowledge about the model and its test time behavior. The attacker can query the model (possibly limited number of times) by passing an input x through the model and measuring one or more outcome such as:

- model prediction $\arg\max_i h_\theta(x)_i$;
- soft labels $h_\theta(x)$;
- the value of the loss $L(h_\theta(x), y)$;
- the gradient with respect to input data $\nabla_x L(h_\theta(x), y)$.

In some scenarios, the attacker can additionally know the model architecture and weights, which were produced during training. However, in the case of randomized models, it would not be possible to know test-time values of random parameters. In our study we consider three levels of access:

- *black-box* attacks, which can obtain only the soft labels $h_\theta(x)$ and, consequently, the loss value $L(h_\theta(x), y)$;
- *gradient* attacks, which can *additionally* access gradients $\nabla_x L(h_\theta(x), y)$;[2]
- *full train-time access* attacks, which can *additionally* "know" all the training-time parameters such as model architecture and weights, but not the test-time random parameters.

3.3 Standard and Adversarial Training

Given a classifier $h_\theta(x)$ the standard training procedure aims to minimize the empirical risk on the *clean*, i.e., original training set $\mathcal{D}$:

$$\min_\theta \sum_{x,y\in\mathcal{D}} L(h_\theta(x), y) \tag{1}$$

To train models resistant to adversarial attacks Szegedy et al. (2014) and Goodfellow et al. (2015) proposed to expand or replace the original test with adversarial examples generated by the model. This general idea has several concrete realizations, which are broadly termed as *adversarial training*. As noted by Goodfellow et al. (2015), "the adversarial training procedure can be seen as minimizing the worst case error when the data is perturbed by an adversary." Using a popular formalization of this statement due to Madry et al. (2018), an adversarial training objective can be written as the following minimax optimization problem:

$$\min_\theta \sum_{x,y\in\mathcal{D}} \max_{\hat{x}\in\mathcal{B}_\varepsilon(x)} L(h_\theta(\hat{x}), y) \tag{2}$$

[2] Note that these gradients are computed with respect to input image, rather than with respect to model parameters. Indeed, the objective of attack is to find the input maximizing the loss rather than the model parameters (that minimize the loss).

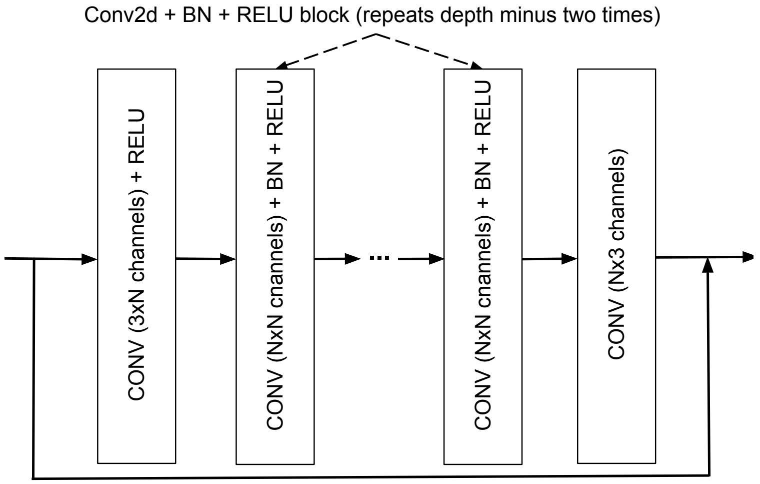
Conv2d + BN + RELU block (repeats depth minus two times)
CONV (3xN channels) + RELU
CONV (NxN cnannels) + BN + RELU
...
CONV (NxN cnannels) + BN + RELU
CONV (Nx3 channels)

Figure 1: Denoising front end (DnCNN) architecture Zhang et al. (2017)): Note the shortcut connection.

The inner maximization problem in Equation 2 corresponds to generation of adversarial example for a given data point and a model. This problem is generally intractable: In practice it is solved approximately using methods such as projected gradient descent (Biggio et al., 2013; Madry et al., 2018). In Section 4.2 we describe several approaches for finding adversarial examples (including PGD), which are used in our experiments.

4 Methods

4.1 Model Architectures and Training

In our experiments we use several types of models:

- Standard, i.e., non-robust models, ,i.e., which are trained using a standard objective (see Equation 1);
- Random ensembles of three standard/non-robust models;
- A standard or robust backbone classifier model combined with a front end, which is further trained using a modified variant of adversarial training (but the backbone is frozen);
- Random ensembles of three models with front ends;
- Models trained using a traditional (multi-epoch) variant of adversarial training (see Equation 2).

In that we use three types of the popular backbone models:

- Standard ResNet models of varying depth (He et al., 2016);
- Wide ResNet with depth 28 and widening factor 10 (Zagoruyko & Komodakis, 2016);
- ViT (Dosovitskiy et al., 2021);

A random ensemble is simply a set of three models, which are selected randomly a uniformly each time we obtain predictions $h_\theta(x)$ or compute gradients $\nabla_x L(h_\theta(x), y)$. It is noteworthy that randomization may unintentionally occur in PyTorch models that are inadvertently left in the training mode. In such as case, the model behaves similarly to a randomized ensemble due to enabled test-time use of dropout (Srivastava et al., 2014).

In the case of front-end enhanced models, an input image is first passed through a pre-processing network $f_{\theta_1}(x)$, aka the front end. Following Norouzzadeh et al. (2021), we use is a popular Gaussian Denoiser model DnCNN (Zhang et al., 2017) to process the image. DnCNN is a fully-convolutional neural network with a skip connection (see Figure 1). The front end is prepended to a standard vision backend such as ResNet (He et al., 2016) or ViT (Dosovitskiy et al., 2021), which we denote as $h_{\theta_2}(x)$. The full network is represented by a composite function $h_{\theta_2}(f_{\theta_1}(x))$.

Unlike many other front-end defenses (Athalye et al., 2018), DnCNN does not explicitly shatter or obfuscate gradients. Moreover, DnCNN is a fully differentiable model with a skip connection: Skip connections are commonly used to improve gradient flow (He et al., 2016).

We train the front-end enhanced model using adversarial training (see Equation 2) with a "frozen" backbone model: This means that only parameters of the front end $f_{\theta_1}(x)$ are being modified (using PGD to generate adversarial examples). In the case of the ResNet (and WideResNet) backends (He et al., 2016; Zagoruyko & Komodakis, 2016), we disable updates of the batch-normalization layers (Ioffe & Szegedy, 2015). Although our training procedure is somewhat similar to the approach of Norouzzadeh et al. (2021), there are two crucial differences:

- We use the original adversarial training objective *without* additional denoiser-specific losses;
- We use an extremely small (in the order of 10^{-6}) learning rate and train for at most one epoch. In contrast, training a strong adversarial model on CIFAR datasets typically requires dozens of epochs.

As mentioned above, several models are trained using a more conventional adversarial training approach. In this case, there is no front end and there is no freezing of parameters. The model is trained for dozens of epochs using learning rates and schedules similar to those used for standard training.

4.2 Attacks

Attacks used in our work require an access to the loss function $L(h_\theta(x), y)$, which is used in a slightly different way for targeted and untargeted attacks. In the case of untargeted attack, the adversary needs to find a point $\hat{x}$ (in the vicinity of the clean input x) with the smallest "likelihood" of the correct class y. This entails solving the following constrained maximization problem:

$$\hat{x} = \underset{x \in \mathcal{B}_\varepsilon(x)}{\arg\max}\, L(h_\theta(x), y) \tag{3}$$

In the case of a targeted attack, the adversary "nudges" the model towards predicting a desired class $\hat{y}$ instead of the correct one y. Thus they minimize the loss of a class $\hat{y}$. This requires solving the following constrained minimization problem:

$$\hat{x} = \underset{x \in \mathcal{B}_\varepsilon(x)}{\arg\min}\, L(h_\theta(x), \hat{y}) \tag{4}$$

There are different approaches to solving Equation 3 or Equation 4. Most commonly, they include first-order gradient-based methods, their zero-order approximations, and random searching. In what follows we briefly survey attacks used in our work. For simplicity of exposition, we focus on the *untargeted* attack case.

Projected Gradient Descent (PGD) Projected Gradient Descent (Biggio et al., 2013; Madry et al., 2018) is an iterative optimization procedure, where in each iteration we make a small step in the direction of the loss gradient and project the result onto $\mathcal{B}_\varepsilon(x)$ to make sure it remains sufficiently close to the initial clean example:

$$x_{n+1} = \text{proj}_{\mathcal{B}_\varepsilon(x)}\left(x_n + \alpha \cdot \text{sign}\, \nabla_{\mathbf{x}} L(h_\theta(x), \hat{y})\right), \tag{5}$$

where sign(x) is an element wise sign operator and α is a step size. PGD with restarts repeats this iterative procedure multiple time starting from different random points inside $\mathcal{B}_\varepsilon(x)$ and produces several examples. Among those, it selects a point with the maximum loss.

Zero-order (finite-difference) PGD. The zero-order PGD is different from the smooth PGD in that it uses a finite-difference approximation for $\nabla_{\mathbf{x}} L(h_\theta(x), \hat{y})$. To simplify the presentation, the below description denotes $g(x) = L(h_\theta(x), \hat{y})$. The classic two-sided finite difference method for estimating gradients of the function $g(x)$ uses the following equation:

$$\nabla_{\mathbf{x}} g(\boldsymbol{x})_i \approx \frac{g(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_i + \alpha, \ldots, \boldsymbol{x}_d) - g(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_i + \alpha, \ldots, \boldsymbol{x}_d)}{2 * \alpha}. \tag{6}$$

However, this computation can be quite expensive as it requires $O(d)$ inferences, where d is the number of pixels in the image. To reduce complexity, we estimate gradients using block-wise differences.

First we organize all variables into blocks. Then, instead of changing a single variable at a time, we add the same α to all variables in a block and additionally divide the estimate in Eq. (6) by the number of variables in a block. This basic procedure works well for low-dimensional CIFAR images, but not for ImageNet. For ImageNet, we first we first divide image into equal-size squares. Then, we combine squares into bigger blocks.

AutoAttack (Croce & Hein, 2020a) is a suit of attacks that has several variants of a gradient (PGD) attack as well as the query-efficient black-box SQUARE attack (Andriushchenko et al., 2020). One of the key contributions of Croce & Hein (2020b) was an Automatic Parameter-free PGD attack with two different loss functions, which are abbreviated as APGD-CE and APGD-DLR.

In addition to this, APGD can be combined with an Expectation Of Transformation (EOT) algorithm (Athalye et al., 2018), which is used to approximate gradients of randomized models. EOT was shown to be effective against randomized defences (Croce & Hein, 2020a). The third algorithm is a targeted FAB attack, which was shown to be effective against gradient masking (Croce & Hein, 2020b).

The `standard` version of AutoAttack includes all gradient attacks *without* EOT and the SQUARE attack. The `rand` version of AuttoAttack has only APGD with EOT where the the number of restarts is set to one, but the number of gradient estimation iterations is set to 20.

Attacks that bypass the front end: To circumvent the gradient-masking front end, we use two similar "removal" approaches. They both exploits the fact that a front end network is close to the identity transformation, i.e., $f_{\theta_1}(x) \approx x$.

The first approach is a variant of the Backward Pass Differentiable Approximation (BPDA) approach with a a straight-through gradient estimator (Athalye et al., 2018). We use a previously described PGD attack and only change the way we compute gradients of the front-end enhanced network $h_{\theta_2}(f_{\theta_1}(x))$. In a general case of a straight-through gradient estimator, one would use $\nabla_{\mathbf{z}} h_{\theta_2}(z)|_{z=f_{\theta_1}(x)}$ instead of $\nabla_{\mathbf{x}} h_{\theta_2}(f_{\theta_1}(x))$. However, due to the front end $f_{\theta_1}(x)$ being approximately equal to the identity transformation, we use $\nabla_{\mathbf{x}} h_{\theta_2}(x)$ directly.

Another approach is a variant of a transfer attack (Szegedy et al., 2014) that simply generates the adversarial example $\hat{x}$ for the network $h_{\theta_2}(x)$ and the clean input x (again we exploit the fact that $f_{\theta_1}(x) \approx x$). To this end, we use APGD gradient attacks from the AutoAttack suit (Croce & Hein, 2020b). Then, the adversarial example $\hat{x}$ is passed to a full composite network $h_{\theta_2}(f_{\theta_1}(x))$.

Table 1: Accuracy for Unprotected/Standard-training Models with a Front End (see table notes)

Single Model						
Model	No Attack	PGD (standard)	APGD+FAB-T (AutoAttack)	SQUARE (AutoAttack)	PGD (0-order)	PGD (BPDA)
CIFAR10						
Resnet-18	91±3%	0%		1±0%	3±2%	
Resnet-18 +front	91±3%	88±4%	75±5%	1±1%	5±2%	0%
VIT-16-224	**97**±2%	0±1%		1±1%	**10**±4%	
VIT-16-224 +front	**97**±2%	**96**±2%	**85**±4%	1±2%	**10**±3%	0±1%
CIFAR100						
Resnet-18	68±6%	0±1%		1±1%	4±3%	
Resnet-18 +front	68±5%	65±5%	48±6%	2±1%	4±2%	0±1%
VIT-16-224	**84**±5%	1±1%		2±2%	8±3%	
VIT-16-224 +front	**84**±5%	**80**±5%	**59**±6%	3±2%	8±4%	1±1%
ImageNet						
Resnet-18	69±9%	0%		2±2%	23±8%	
Resnet-18 +front	70±8%	68±9%	28±8%	2±3%	23±8%	0%
VIT-16-224	**83**±7%	0%		18±8%	60±9%	
VIT-16-224 +front	82±8%	**79**±8%	**48**±9%	**21**±8%	**62**±9%	0%
Randomized Ensemble						
Model	No Attack	PGD (standard)	APGD/EOT+SQUARE (AutoAttack)	SQUARE (AutoAttack)	PGD (0-order)	PGD (BPDA)
CIFAR10						
Resnet-18 3×	92±3%	2±2%		76±5%	66±6%	
Resnet-18 +front 3×	91±4%	89±4%	70±6%	74±5%	64±5%	3±2%
VIT-16-224 3×	**97**±2%	1±1%		78±5%	64±6%	
VIT-16-224 +front 3×	96±2%	**96**±2%	**74**±5%	81±4%	65±6%	1±1%
CIFAR100						
Resnet-18 3×	69±5%	5±3%		51±6%	49±5%	
Resnet-18 +front 3×	66±6%	67±5%	40±6%	54±6%	46±6%	8±3%
VIT-16-224 3×	85±4%	2±2%		**64**±6%	46±5%	
VIT-16-224 +front 3×	**86**±4%	**81**±5%	**54**±6%	63±5%	45±6%	3±1%
ImageNet						
Resnet-18 3×	69±9%	0%		2±2%	23±8%	
Resnet-18 +front 3×	70±8%	68±9%	42±10%	48±9%	33±9%	0±2%
VIT-16-224 3×	**83**±7%	0%		18±7%	58±10%	
VIT-16-224 +front 3×	82±8%	**80**±7%	**63**±9%	**69**±9%	**68**±9%	0%

Test sample has 500 images for CIFAR10/CIFAR100 and 200 images for ImageNet:
We show 99% confidence intervals (CI) when applicable: Note that CIs can not be computed for attacks with 100% success rate.
APGD/EOT+SQUARE is a combination of APGD with EOT (as in `rand` version of AutoAttack) followed by SQUARE attack.
We do not run APGD or BPDA attacks when the accuracy ≈ 0% under the *standard* PGD attack.

Table 2: Summary of Results for Unprotected/Standard-training Models with a Front End

Dataset	ε-attack	Single model (stand. train.)	Best Randomized Ensembles (adv. train. with frozen backbones)			Best RobustBench original dataset		**extra** data	
		No Attack	No Attack	AutoAttack (APGD/EOT+SQUARE)	PGD (BPDA)	No Attack	AutoAttack (standard)	No Attack	AutoAttack (standard)
CIFAR10	8/255	95±3%	96±2%	74±5%	5±2%	92.2%	66.6%	93.3%	71.1%
CIFAR100	8/255	84±5%	86±4%	54±7%	3±1%	69.2%	36.9%	75.2%	42.7%
ImageNet	4/255	85±4%	82±8%	63±9%	0%	78.9%	59.6%		

Attack $\varepsilon = 8/255$. Test sample has 500 images for CIFAR10/CIFAR100 and 200 images for ImageNet:
We show 99% confidence intervals (CI) when applicable: Note that CIs can not be computed for attacks with 100% success rate.

Table 3: Accuracy for Adversarially-Trained Models with a Front End on CIFAR10.

			Single model			
ε-train	No Attack	+APGD-CE	+APGD-T	+FAB-T	+SQUARE	Transfer (AutoAttack)
2/255	95.2%	92.6%	92.4%	92.4%	41.0±4.3%	15.2±3.2%
4/255	92.2%	90.8%	90.6%	90.6%	56.4±4.4%	32.6±4.1%
8/255	84.8%	83.8%	82.6%	82.6%	60.8±4.3%	43.6±4.4%
			Randomized Ensemble			
ε-train	No Attack	+APGD-CE	+APGD-DLR/EOT		+SQUARE	Transfer (AutoAttack)
2/255	94.4%	93.8%	92.4%		90.8±2.5%	18.2±3.6%
4/255	92.2%	91.4%	90.6%		89.0±2.8%	35.6±4.3%
8/255	84.8%	83.2%	82.2%		80.2±3.5%	45.2±4.4%

Attack $\varepsilon = 8/255$. Test sample has 500 images: We show 99% confidence intervals (CI).
To simplify exposition we do not post CIs for intermediate results, but only for the final attacks.

5 Experiments

5.1 Experimental Setup

Evaluation Protocol We use three datasets: CIFAR10, CIFAR100 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009) all of which have RobustBench leaderboards for L^{∞} attacks (Croce et al., 2021). CIFAR10/100 images have a small resolution of 32 by 32 pixels, while ImageNet models use the resolution 224x224. We use attack radii that are the same as in RobustBench, namely, $\varepsilon = 8/255$ for CIFAR10/CIFAR100 and $\varepsilon = 4/255$ for ImageNet.

All our datasets have large test sets. However, running AutoAttack on the full test sets is quite expensive. It is also unnecessary since we do not aim to establish a true AutoAttack SOTA. Thus, we use randomly selected images: 500 for CIFAR10/CIFAR100 and 200 for ImageNet. The uncertainty in accuracy evaluation due to this sampling is estimated using 99% confidence intervals. Note that in some cases an attack is perfectly successful and every image is classified incorrectly. In such cases, the sample variance is zero and the confidence interval cannot be computed.

Models For our main experiments we use four types of the models that can be classified against two dimensions:

- Presence of the front end;

- Single-model vs a three-model randomized ensemble.

A detailed description of these models and their training procedures is given in Section 4.1. Here we only note that for backbone types are the same between CIFAR10/CIFAR100 and ImageNet. The only exception is a WideResnet, which is used only for CIFAR10 and CIFAR100.

Attacks We used a combination of attacks described in Section 4.2:

- Standard 50-step PGD with 5 restarts;
- Zero-order (finite-difference) 10-step PGD with 1 restart. For ImageNet, pixels are first grouped into 8x8 squares. For CIFAR datasets, which have very low resolution and the square size is one pixel, i.e., no such grouping is happening. Furthermore, these blocks are randomly grouped (5 squares in a group).
- Various combinations of attacks from the AutoAttack package (Croce & Hein, 2020a). Unless specified otherwise:
  - When attacking a single (i.e., non-randomized model) we used APGD attacks without EOT (Athalye et al., 2018). Together with the SQUARE attack (Andriushchenko et al., 2020) and FAB-T (Croce & Hein, 2020b) this combination is a `standard` AutoAttack.
  - To attack a randomized ensemble, we use an *expanded* `rand` variant of AutoAttack. AutoAttack `rand` has only APGD attacks with EOT (denoted as APGD/EOT): We also use the SQUARE attack.
- BPDA PGD with 10 steps and 5 restarts.
- Transfer attack using AutoAttack APGD.

5.2 Discussion of Results

Summary of results In this section we briefly review results showing that models with differentiable front ends can, indeed, appear much more robust then they actually are. In particular, when such models are assembled into a randomized ensemble, they appear to achieve near-SOTA robust accuracy virtually without degradation on clean test sets. We will then try to answer the following research questions:

- Is this behavior due to gradient masking?
- Is AutoAttack in principle effective against randomized ensembles?

In our experiments we focused primarily on combining standard models with a front end (except for CIFAR10). This was done to reduce experimentation cost. Indeed, training adversarially robust models for ImageNet is computationally-intensive while high-accuracy models trained using the standard objective and original/clean data are readily available.

The key outcomes of these experiments are presented in Tables 1 and 2. Table 1 includes results only for ResNet-18 (He et al., 2016) and ViT (Dosovitskiy et al., 2021). Expanded variants of Table 1, which include additional models are provided in the Appendix A.2 (Tables 6, 7, 8).

From Table 2 it can be seen that randomized ensembles, which include front-end enhanced models rival the best models on RobustBench (Croce et al., 2021), even in the case when RobustBench models were trained using additional data. At the same time, we do not observe degradation on the clean data, which indicates that a front-end should largely preserve original images. However, all front-end enhanced models have nearly zero accuracy under the BPDA PGD attack. At the same time, both the gradient attacks and the SQUARE Andriushchenko et al. (2020) attack are not particularly effective.

The SQUARE attack is quite effective against a single model without a front end. On CIFAR10 and CIFAR100 datasets (see Table 1 and Tables 6, 7 in Appendix A.2) it degrades accuracy of all models to be nearly zero. For ImageNet and ViT, the accuracy under attack is also quite low (about 20%). However, it is much less successful against randomized ensembles even when individual models are without front ends. We also find that in some cases the zero-order PGD can be more successful than other attacks: However, the attack success rate is still well below the expected 100%. To reiterate, both zero-order PGD and the SQUARE attack are particularly ineffective against randomized ensembles with front-end enhanced models.

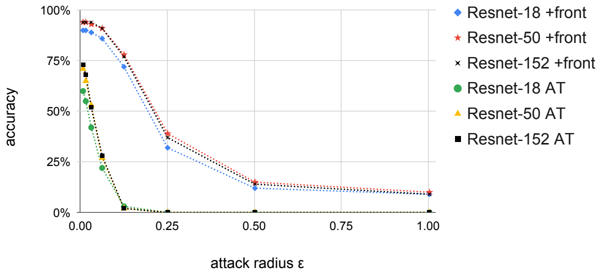

Figure 2: CIFAR10: Accuracy under attack for different attack radii for models with a front-end and models that underwent traditional (multi-epoch) adversarial training without using a front-end (denoted with AT). Accuracy of front-end equipped models is decreasing much slower than the accuracy of AT models, which is indicative of gradient masking.

Table 4: Accuracy under Attack for Adversarially-Trained Models on CIFAR10 for Standard vs Zero-Order PGD

	No Attack	PGD (0-order)	PGD (standard)
Resnet-18	65±2%	53±2%	42±2%
Resnet-50	76±2%	62±2%	53±2%
Resnet-101	78±2%	65±2%	52±2%

All models are trained using a standard adversarial training procedure without a front end (training $\varepsilon = 8/255$).

Test sample has 2000 images: 99% confidence intervals are shown.

What happens when an adversarially trained model is combined with a front end? We carried out a detailed investigation using a sample of CIFAR10 images and presented results in Table 3. The main difference here is that we report accuracy after each attack from the AutoAttack (Croce & Hein, 2020a) packages. Thus, we can see how successful each type of attack is.

From Table 3 we can see that adversarial training even with small $\varepsilon = 2/255$, makes the model nearly "impervious" to gradient attacks when such a model is combined with a front-end. For example, a combination of all gradient attacks degrade the accuracy of a single front-end "protected" model from 95.2% to only 92.4%. At the same time the true robust accuracy of this model does not exceed 15.2%. In that, the SQUARE attack could reduce accuracy to only 41%. In the case of a randomized ensemble, though, it is not effective and the combination of attacks can reduce accuracy to only 90.8%. Curiously enough, adversarial training seems to increase resistance to both gradient and black-box attacks.

Is it gradient masking? It should be now clear, that the AutoAttack package (and/or its inidividual attacks) can greatly exaggerate performance of both single models and (especially) randomized ensembles. There are several indicators that this is primarily due to gradient masking. First, if we plot accuracy of the models for attacks of varying radii ε (as recommended by Tramèr et al. 2020), we can see that the accuracy of adversarially-trained models without front ends quickly decreases as the attack radius ε grows. In contrast, the accuracy of the front-end equipped models declines much slower and does not reach zero when $\varepsilon = 1$, which is the maximum possible distortion (remember that pixel intensity ranges from 0 to 1).

Second, we can see that black-box attacks, namely SQUARE and zero-order PGD, can be substantially more effective than gradient attacks. In that, for a typical adversarially trained model, the zero-order PGD is less effective (see Table 4).

Third, we can see that front-end bypassing attacks, such as BPDA PGD make front-end "protected" models to be much less accurate.

Although the gradient masking phenomenon is not new, the degree of this masking is quite remarkable for fully differentiable models that do not have gradient-shattering components such as JPEG compression. For example, Tramèr et al. (2020) describe a case when gradient masking was caused by an additional "log-softmaxing" which is known to reduce "gradient flows". However, in our case, a gradient-masking culprit is a front end, which is a fully-convolutional network with a skip connection and batch-normalization layers (see § A.1 where we describe checking for numerical abnormalities). Both of these architectural components were instrumental to training deep neural networks, which otherwise would suffer from vanishing or exploding gradients (Szegedy et al., 2014; He et al., 2016).

Is AutoAttack in principle effective against randomized ensembles? We observed that AutoAttack can be ineffective against randomized ensembles. In particular, on a sample of CIFAR10 images we can achieve an accuracy of 90.8±2.5% (99% CI) for a model that has true robust accuracy under 20%. Is AutoAttack, in principle, weak against randomized ensembles or is there something special about our ensemble models?

First of all, we confirmed that even standard PGD can be extremely effective against randomized ensembles where individual models underwent only standard training using clean data. Indeed, from the Table 1 and Tables 6, 7 in Appendix A.2 we see that accuracy of such ensembles is nearly zero under a standard PGD attack. This was an unexpected finding because such attacks do not use averaging techniques such as EOT and each gradient computation uses a mix of models with different random weights.

Table 5: Accuracy under attack for diverse randomized ensembles of varying adversarial robustness on CIFAR10 (500 images).

ε-train	PGD (ensemble) average)	PGD (full ensemble)	AutoAttack
2/255	30±2%	64±6%	29±5%
4/255	47±2%	67±5%	42±6%
8/255	52±3%	64±6%	47±6%

A more careful investigation showed that—despite the lack of EOT—because all models in the ensemble share the same architecture, the standard PGD finds adversarial examples that work for *all* models in the ensemble. To prevent this, we trained ensembles using three different types of backbone models, namely, ResNet-152 (He et al., 2016), wide ResNet-28-10 (Zagoruyko & Komodakis, 2016), and ViT (Dosovitskiy et al., 2021). We did not use any front ends and trained three ensembles using three values of attack ε: 2/255, 4/255, and 8/255. For each model in the ensemble we computed the accuracy under a standard PGD attack (with $\varepsilon = 8/255$) and then obtained ensemble averages, which ranged from 30% to 52% depending on the train-time ε.

As expected, the smaller was ε used during training, the lower was the robust accuracy of the models. However, when we attacked a randomized ensemble using a 50 step PGD with five restarts, we achieved the accuracy close to 65% (see Table 5). Thus, the randomized ensemble of diverse models can fool the standard PGD attack without EOT. In contrast, the accuracy under AutoAttack was slightly lower (as expected) than ensemble-average robust accuracy (see Table 5), thus, confirming that AutoAttack can indeed be very effective against randomized ensembles.

6 Conclusion

We discovered that training a fully differentiable front end with a frozen classification model can result in the model that is unusually resistant to gradient attacks while retaining accuracy on original data. We provided evidence that this behavior was due to gradient masking. Although the gradient masking phenomenon is not new, the degree of this masking is quite remarkable for fully differentiable models that do not have gradient-shattering components such as JPEG compression or components that are expected to cause diminishing gradients. The objective of this paper is to document this phenomenon and discuss potential consequences for automatic robustness evaluation packages such as AutoAttack (Croce & Hein, 2020a) rather than to establish new SOTA in adversarial robustness.

References

Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *CoRR*, abs/2108.00401, 2021.

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *ECCV (23)*, volume 12368 of *Lecture Notes in Computer Science*, pp. 484–501. Springer, 2020.

Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283. PMLR, 2018.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognit.*, 84:317–331, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pp. 387–402. Springer, 2013.

Franziska Boenisch, Philip Sperl, and Konstantin Böttinger. Gradient masking and the underestimated robustness threats of differential privacy in deep learning. *CoRR*, abs/2105.07985, 2021.

Nicholas Carlini. A complete list of all (arxiv) adversarial example papers. `https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html`, 2019. Last accessed in September 2022: The list has been regularly updated at the moment of access.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Trans. Intell. Technol.*, 6(1):25–45, 2021.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216. PMLR, 2020a.

Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2196–2205. PMLR, 2020b.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS Datasets and Benchmarks*, 2021.

Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. Adversarial classification. In *KDD*, pp. 99–108. ACM, 2004.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. IEEE Computer Society, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR (Poster)*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

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

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Peter Lorenz, Dominik Strassel, Margret Keuper, and Janis Keuper. Is robustbench/autoattack a suitable benchmark for adversarial robustness? *CoRR*, abs/2112.01601, 2021.

Daniel Lowd and Christopher Meek. Adversarial learning. In *KDD*, pp. 641–647. ACM, 2005.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR (Poster)*. OpenReview.net, 2018.

Mohammad Sadegh Norouzzadeh, Wan-Yi Lin, Leonid Boytsov, Leslie Rice, Huan Zhang, Filipe Condessa, and J Zico Kolter. Empirical robustification of pre-trained classifiers. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR (Poster)*, 2014.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *NeurIPS*, 2020.

Nurislam Tursynbek, Aleksandr Petiushko, and Ivan V. Oseledets. Robustness threats of differential privacy. *CoRR*, abs/2012.07828, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*. BMVA Press, 2016.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, 26(7):3142–3155, 2017.

A Appendix

A.1 Gradient Tracing: Checking for Numerical Instabilities

To make sure our results were not affected by numerical instabilities (or by exploding and vanishing gradients), we "traced" gradients for ResNet-152 (He et al., 2016) backbone on all three datasets. We used PGD with 50 steps and a single restarted. We did this for all datasets using a sample of 500 images. We compared gradients between two scenarios:

- Attacking a model trained on clean data;
- Attacking a front-end "ehanced" model that was adversarially trained.

We found no apparent anomalies. In particular:

- No gradient was ever NaN or $\pm\infty$;
- In both cases only a modest fraction (10-30%) of absolute gradient values were small (< 0.01);
- In both cases, mean and median gradients (per image) tended to be close to zero;
- 99% of gradients absolute values were under 2000;
- In that, the distribution of gradient values for front-end "enhanced" models tended to be more skewed with bigger extremely values (by 1-2 order of magnitudes).

A.2 Additional Experimental Results

Table 6: Accuracy under attack, single model scenario (CIFAR10)

Single Model

Model	No Attack	PGD	APGD+FAB-T	SQUARE	PGD	PGD
		(standard)	(AutoAttack)	(AutoAttack)	(0-order)	(BPDA)
Resnet-18	91±3%	0%		1±0%	3±2%	
Resnet-18 +front	91±3%	88±4%	75±5%	1±1%	5±2%	0%
Resnet-50	95±3%	0%		0±1%	4±3%	
Resnet-50 +front	95±3%	93±3%	78±5%	**2**±2%	5±3%	0±1%
Resnet-152	94±3%	0%		1±2%	6±3%	
Resnet-152 +front	94±3%	92±3%	81±4%	**2**±2%	7±3%	0%
WideResNet-28-10	96±2%	0%		0%	8±3%	
WideResNet-28-10 +front	95±3%	91±3%	65±5%	1±1%	**10**±4%	1±1%
VIT-16-224	**97**±2%	0±1%		1±1%	**10**±4%	
VIT-16-224 +front	**97**±2%	**96**±2%	**85**±4%	1±2%	**10**±3%	0±1%

Randomized Ensemble

Model	No Attack	PGD	APGD/EOT+SQUARE	SQUARE	PGD	PGD
		(standard)	(AutoAttack)	(AutoAttack)	(0-order)	(BPDA)
Resnet-18 3×	92±3%	2±2%		76±5%	66±6%	
Resnet-18 +front 3×	91±4%	89±4%	70±6%	74±5%	64±5%	3±2%
Resnet-50 3×	95±3%	3±2%		80±4%	69±5%	
Resnet-50 +front 3×	96±2%	94±3%	71±5%	81±4%	69±5%	5±2%
Resnet-152 3×	94±3%	3±2%		**84**±4%	70±5%	
Resnet-152 +front 3×	95±2%	94±3%	71±5%	79±5%	**74**±5%	**6**±3%
WideResNet-28-10 3×	96±2%	3±2%		77±5%	45±6%	
WideResNet-28-10 +front 3×	95±3%	92±3%	60±6%	79±4%	51±5%	3±2%
VIT-16-224 3×	**97**±2%	1±1%		78±5%	64±6%	
VIT-16-224 +front 3×	96±2%	**96**±2%	**74**±5%	81±4%	65±6%	1±1%

Test sample has 500 images: 99% confidence intervals are shown. Each random ensemble has 3 models.

APGD/EOT+SQUARE is a combination of APGD with EOT (as in `rand` version of AutoAttack) followed by SQUARE attack.

Best outcomes are shown in bold. Best results are computed **separately** for single models and ensembles.

We show 99% confidence intervals: They cannot be computed when the attack has 100% success rate.

We do not run other smooth PGD attacks when the accuracy ≈ 0% with the standard PGD attack.

Table 7: Detailed Results for CIFAR100

Single Model

Model	No Attack	PGD (standard)	APGD+FAB-T (AutoAttack)	SQUARE (AutoAttack)	PGD (0-order)	PGD (BPDA)
Resnet-18	68±6%	0±1%		1±1%	4±3%	
Resnet-18 +front	68±5%	65±5%	48±6%	2±1%	4±2%	0±1%
Resnet-50	75±5%	1±0%		1±2%	6±3%	
Resnet-50 +front	77±5%	70±6%	53±6%	2±2%	7±3%	1±1%
Resnet-152	76±5%	1±2%		2±1%	9±3%	
Resnet-152 +front	74±5%	71±5%	51±6%	**5**±2%	**11**±4%	2±2%
WideResNet-28-10	76±5%	2±1%		0±1%	7±3%	
WideResNet-28-10 +front	75±5%	67±5%	35±5%	2±1%	8±3%	**3**±1%
VIT-16-224	**84**±5%	1±1%		2±2%	8±3%	
VIT-16-224 +front	**84**±5%	**80**±5%	**59**±6%	3±2%	8±4%	1±1%

Randomized Ensemble

Model	No Attack	PGD (standard)	APGD/EOT+SQUARE (AutoAttack)	SQUARE (AutoAttack)	PGD (0-order)	PGD (BPDA)
Resnet-18 3×	69±5%	5±3%		51±6%	49±5%	
Resnet-18 +front 3×	66±6%	67±5%	40±6%	54±6%	46±6%	8±3%
Resnet-50 3×	73±5%	6±2%		57±6%	49±6%	
Resnet-50 +front 3×	75±5%	72±6%	46±6%	58±6%	51±6%	9±3%
Resnet-152 3×	76±5%	10±3%		58±6%	49±6%	
Resnet-152 +front 3×	75±5%	71±5%	48±6%	57±6%	**53**±6%	**10**±4%
WideResNet-28-10 3×	78±5%	5±2%		52±5%	31±5%	
WideResNet-28-10 +front 3×	76±5%	70±5%	34±6%	52±6%	33±6%	5±2%
VIT-16-224 3×	85±4%	2±2%		**64**±6%	46±5%	
VIT-16-224 +front 3×	**86**±4%	**81**±5%	**54**±6%	63±5%	45±6%	3±1%

Test sample has 500 images: 99% confidence intervals are shown. Each random ensemble has 3 models.

APGD/EOT+SQUARE is a combination of APGD with EOT (as in `rand` version of AutoAttack) followed by SQUARE attack.

Best outcomes are shown in bold. Best results are computed **separately** for single models and ensembles.

We show 99% confidence intervals: They cannot be computed when the attack has 100% success rate.

We do not run other smooth PGD attacks when the accuracy ≈ 0% with the standard PGD attack.

Table 8: Detailed Results for ImageNet

Single Model

Model	No Attack	PGD	APGD+FAB-T	SQUARE	PGD	PGD
		(standard)	(AutoAttack)	(AutoAttack)	(0-order)	(BPDA)
Resnet-18	69±9%	0%		2±2%	23±8%	
Resnet-18 +front	70±8%	68±9%	28±8%	2±3%	23±8%	0%
Resnet-50	76±8%	0%		7±5%	40±9%	
Resnet-50 +front	76±7%	66±9%	31±9%	8±5%	40±10%	0%
Resnet-152	78±8%	0%		12±5%	46±9%	
Resnet-152 +front	78±8%	74±9%	26±8%	16±6%	47±9%	0%
VIT-16-224	**83**±7%	0%		18±8%	60±9%	
VIT-16-224 +front	82±8%	**79**±8%	**48**±9%	**21**±8%	**62**±9%	0%

Randomized Ensemble

Model	No Attack	PGD	APGD/EOT+SQUARE	SQUARE	PGD	PGD
		(standard)	(AutoAttack)	(AutoAttack)	(0-order)	(BPDA)
Resnet-18 3×	69±9%	0%		2±2%	23±8%	
Resnet-18 +front 3×	70±8%	68±9%	42±10%	48±9%	33±9%	0±2%
Resnet-50 3×	76±8%	0%		8±4%	38±8%	
Resnet-50 +front 3×	76±7%	72±8%	50±9%	55±9%	50±9%	0%
Resnet-152 3×	78±8%	0%		12±5%	44±9%	
Resnet-152 +front 3×	78±7%	74±8%	48±10%	53±9%	53±9%	0%
VIT-16-224 3×	**83**±7%	0%		18±7%	58±10%	
VIT-16-224 +front 3×	82±8%	**80**±7%	**63**±9%	**69**±9%	**68**±9%	0%

Test sample has 200 images: 99% confidence intervals are shown. Each random ensemble has 3 models.
APGD/EOT+SQUARE is a combination of APGD with EOT (as in `rand` version of AutoAttack) followed by SQUARE attack.
Best outcomes are shown in bold. Best results are computed **separately** for single models and ensembles.
We show 99% confidence intervals: They cannot be computed when the attack has 100% success rate.
We do not run other smooth PGD attacks when the accuracy ≈ 0% with the standard PGD attack.