Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики факультету інформатики

Курсова робота
За спеціальністю “Прикладна Математика”

Adversarial robustness and attacks in Deep Learning

Керівник курсової роботи

Ст. Викл. Швай Н.О.

_____ (підпис)

“____” _____ 2022 р.

Виконав:

студент МП-1

факультету інформатики

Кузьменко Д. О.

Table of contents

Research Schedule

Number	Coursework writing stages	Date
1	Preliminary acquaintance with the direction and the decision on the topic	2021-09-08 – 2022-09-22
2	Previous works research	2021-10-01 – 2021-12-15
3	Experiment and work plan draft	2022-03-20 – 2022-03-23
4	Additional research on the fresh papers, narrowing down to most suitable options and tools for the experiments.	2022-05-23 – 2022-06-01
5	Conducting planned experiments, making conclusions on the results, and starting a coursework draft.	2022-06-02 – 2022-06-05

6	Pre-final coursework draft	2022-06-06
7	Final coursework draft	2022-06-07

1 Introduction

A problem of adversarial robustness and robustness-accuracy tradeoff [1] has been extensively researched in the previous years. A large amount of gradient-based attacks [2][3], logit-based attacks [4], and even ensembles of both black-box and white-box attacks [5] have emerged. In response, many sophisticated defensive adversarial model training mechanisms focused on perturbing own image classes and weights [6], adding augmentations [7], or incorporating unlabeled data [8] have been benchmarked as very resisting techniques to adversarial attacks' attempts.

Put in plain words, given the initial model *M*, input image *X*, and the corresponding label processed via a model – M(X) = *Y*, an adversarial attack performs certain perturbations on *X*, and receives a new perturbed image *X'*, visually indistinguishable from *X*. The new label assigned by the model *M(X') = Y'*, s.t. *Y!=Y'*, rendering the example successfully attacked.

The theoretical underpinnings for this field involve the notions of robustness and astuteness, local Lipschitzness, r-separability of datasets, robustness-accuracy tradeoff, and *L-inf* distance. This work will cover all the preliminaries, explain the choice of CIFAR-10 with *L-inf* metric space and *eps=8/255* as a main dataset for the task, make use of already well-known attacks and defenses, introduce new ones, and try different ensembles on the 3 most robust models available on the benchmark – Adversarial Weight Perturbation [6], Augmentations and weight averaging [7], and Self-COnsistent Robust Error (SCORE-based model) [20].

The experiments and benchmarking were done with the help of **torchattacks** [9], and **RobustBench** [10] with a single **NVIDIA RTX 3060**.

2 Preliminaries

2.1 Robustness and astuteness

Let $X \subseteq \mathrm{R}^d$ be an instance space equipped with a metric dist : $X \times X \rightarrow \mathrm{R}^+$; this is the metric in which robustness is measured. Let $[C] = \{1, 2, \ldots, C\}$ denote the set of possible labels with $C \geq 2$. For a function $f : X \rightarrow \mathrm{R}^C$, let $f(x)_i$ denote the value of the i-th coordinate.

Let $B(x, \varepsilon)$ denote a ball of radius $\varepsilon > 0$ around x in a metric space. B_∞ denotes the L_∞ ball. A classifier g is ***robust*** at x with radius $\varepsilon > 0$ if for all $x` \in B(x, \varepsilon)$, we have $g(x`) = g(x)$. Additionally, g is ***astute*** at (x, y) if $g(x`) = y$ for all $x` \in B(x, \varepsilon)$. The astuteness of g at radius $\varepsilon > 0$ under a distribution μ is:
$\Pr_{(x,y)\sim\mu}[g(x`) = y \text{ for all } x` \in B(x, \varepsilon)]$.
The goal of robust classification aims to establish a g with the highest astuteness [23]. Natural accuracy is used to refer to standard test accuracy (no adversarial perturbation), in order to differentiate it from robust accuracy, or astuteness (including adversarial perturbation).

2.2 Local Lipschitzness

Definition. Let $(X, dist)$ be a metric space. A function $f : X \rightarrow R^C$ is *L-locally Lipschitz* at radius r if for each $i \in [C]$, we have $|f(x)_i - f(x`)_i| \leq L \cdot dist(x, x`)$ for all $x`$ with $dist(x, x`) \leq r$.

2.3 r-separability

Separation. Let X contain C disjoint classes $X^{(1)}, \ldots, X^{(C)}$, where all points in $X^{(i)}$ have label i for $i \in [C]$.

Definition 2 (*r-separation*). We say that a data distribution over $U_{i \in [C]} X^{(i)}$ is r-separated if $dist(X^{(i)}, X^{(j)}) \geq 2r$ for all $i \mathrel{!}= j$, where $dist(X(i), X(j)) = \min_{x \in X(i), x0 \in X(j)} dist(x, x`)$.

Dataset is **r-separated** if the distance between two samples of different classes is at least *2r*. One of our motivating observations is that many real classification tasks comprise separated classes; for example, if dist is the `∞ norm, then images with different categories (e.g., dog, cat, panda, etc) will be r-separated for some value r > 0 depending on the image space. This property holds for a number of standard image datasets.

2.4 Importance of r-separability

If the data is *r-separated,* it is possible to create a robust classifier, without trading much accuracy for it.

It was shown that it is theoretically possible to achieve both robustness and accuracy for r-separated data. In particular, with a classifier based on a locally Lipschitz, which has astuteness 1 with radius *r*. Working directly in the multiclass case, our proof uses classifiers of the following form. If there are *C* classes, we start with a vector-valued function $f : X \rightarrow R^C$ so that $f(x)$ is a *C*-dimensional real vector. Let $dist(x, X^{(i)}) = \min_{z \in X(i)} dist(x, z)$. The following function was analyzed:

$$f(x) = 1/r \cdot \ dist(x, X^{(1)}), \ldots, dist(x, X^{(C)}) ,$$

In other words, we set $f(x)_i = 1/r \cdot dist(x, X^{(i)})$. Then, a classifier *g* is defined : $X \rightarrow [C]$ as $g(x) = \operatorname{argmin}_{i \in [C]} f(x)_i$.

Authors showed that accuracy and local Lipschitzness together imply astuteness by proving the following lemma:

Lemma 1. Let $f : X \rightarrow R^C$ be a function, and consider $x \in X$ with true label $y \in [C]$. If *f* is *1/r -Locally Lipschitz* in a radius *r* around *x*, and $f(x)_j - f(x)_y \geq 2$ for all $j \mathrel{!}= y$, then $g(x) = argmin_i f(x)_i$ is **astute** at *x* with radius *r*.

2.5 L-infinity distance and *eps* parameter

L-inf distance can be viewed as Max Normalization, i.e. getting the maximum absolute difference between all the pixels from two images.
Eps is a perturbation radius. For L2 distance benchmarks, the *eps* of 128/255 is usually used. For L-inf though, the *eps* value ranges from 4/255 to 16/255 depending on the dataset.

2.6 Adversarial perturbations

Let $x \in R^d$ be an input point and $y \in \{1, \ldots, C\}$ be its correct label. For a classifier f : $R^d \rightarrow R^C$, we define a successful adversarial perturbation with respect to the perturbation set $\Delta \subseteq R^d$ as a vector $\delta \in R^d$ such that $\arg\max_{c \in \{1,\ldots,C\}} f(x + \delta)_c \; != y$ and $\delta \in \Delta$.
The perturbation set Δ is typically chosen such that all points in $x + \delta$ have y as their true label. This motivates a typical robustness measure called robust accuracy, which is the fraction of datapoints on which the classifier f predicts the correct class for all possible perturbations from the set Δ.

In practice, an upper bound on the robust accuracy is computed via some adversarial attacks which are mostly based on optimizing some **differentiable loss** (e.g., cross-entropy) using **local search algorithms** like projected gradient descent (PGD) in order to find a successful adversarial perturbation.

Obfuscated gradients are defined as a kind of gradient masking, as a phenomenon that leads to a false sense of security in defenses against adversarial examples. While defenses that cause obfuscated gradients appear to defeat iterative optimization-based attacks, such defenses can be easily circumvented. [11]

3 Related Works

3.1 Adversarial attacks

3.1.1.1 FGSM

The fast gradient sign method uses the gradients to create an adversarial example. For an input image, the method uses the input image gradients to create a new image maximizing the loss. The resulting image is an adversarial sample.

$$adv\_x = x + \varepsilon * sign\ (\Delta_x J(\Theta, x, y))$$

3.1.1.2 FFGSM

In fast adversarial training, a uniform randomization $U(-\varepsilon, \varepsilon)$ is used instead of $sgn(N(0^n, I^n))$ in R+FGSM. Furthermore, for the first time, a step size α is set to a larger value than ε.

$x'0 = x + U(-\varrho, \varrho)$

$x' = \Pi_{B(x,\varepsilon)}\{x'0 + \alpha \cdot sgn(\nabla x'0\ \ell(f(x'0), y))\}$ where $\Pi_{B(x,\varrho)}$ refers the projection to $B(x, \varepsilon)$. L_∞ is used as the distance measure.

3.1.2 Carlini-Wagner

While PGD intends to maximize the soft-max cross-entropy loss, C&W directly attacks the logits before the softmax layer. In previous works, it shows steadily better performance than the standard attack of PGD. Nevertheless, our works have shown that the adversarial examples generated by C&W can be more vulnerable to perturbations.

$$\textbf{C\&W}: x_{cw} = \text{argmax}_{x` \in B(x,a)}\ (-z_y(x`) + max_{\ i!=y}\ z_i(x`)).$$

3.1.3.1 APGD-CE (improved PGD with cross-entropy loss)

APGD-CE also uses cross-entropy loss for optimization like PGD. The difference is that APGD-CE uses adaptive optimization step size and achieves better performance.

Specifically, on each optimization step of $x`$, the hyper-parameters of η will be updated according to a series of intuitive designs.

3.1.3.2 APGD-DLR, APGD-T

Like C&W, APGD-DLR also attacks the logits. CW loss is not scaling invariant, and thus an extreme re-scaling could in principle be used to induce gradient masking. Thus, they introduce the difference between the largest logit $z_{\pi 1}\ (x`)$ and the third-largest logit $z_{\pi 3}\ (x`)$ to counter the potential scaling.
APGD-T is its targeted version. It iterates among all the false classes as the targeted attacking aim and selects the worst case among them. APGD-T can be more time-consuming but also more effective.
DLR: $x_{DLR} = \text{argmax}_{x` \in B(x,a)} - [z_y(x`) - \max_{i!=y} z_i(x`)\ z_{\pi 1}\ (x`) - z_{\pi 3}\ (x`)]$.

3.1.4 FAB / DeepFool [12]

These two attacks are based on pushing examples to the other side of the decision boundary. The reason they intend to minimize the distance to the natural example $|x` - x|_p$ is to create adversarial examples that are more difficult to be detected by either visual clues or adversarial examples detection techniques.
FAB: $x_{fab} = \text{argmin}\ x`\ |x0 - x|_p$ such that $\text{argmax}_c f_c(x`) != y$

3.1.5 Black-box attacks – Square and RayS [13][14]

The *Square* establishes the landscape of the network by querying it extensively, is thus very computationally expensive. Although slightly worse than the white-box attacks and more time-consuming, its performance on the successful attacking rate is still overwhelming and significantly endangers the deployment of deep networks.

RayS is the best hard-label (using only discrete prediction of the model) black-box attack available at the moment. Its performance on adversarially-trained models is slightly lower than Square's, but the total successful attacking rate is still promising

3.2 Adversarial defenses

3.2.1 AT

Adversarial training is a successful defense by Madry et al. [15] that trains based on adversarial examples:

$$\min_f E\{\max_{X0 \in B(X,\varepsilon)} L(f(X`), Y)\}.$$

3.2.2 Locally-Linear Regularization (LLR)

Qin et al. [16] propose to regularize the local linearity through the motivation that AT with PGD increases the model's local linearity. The authors first formulate the function g to evaluate the local linearity of a model:

$$g(f, \delta, \boldsymbol{X}) = |L(f(\boldsymbol{X} + \delta), Y) - L(f(\boldsymbol{X}), Y) - \delta^T \nabla_X L(f(\boldsymbol{X}), Y)|.$$

Define $\gamma(\varepsilon, \boldsymbol{X}) = E\{\max_{\delta \in B(X,\varepsilon)} g(f, \delta, X)\}$,

and $\delta_{LLR} = E\{\operatorname{argmax}_{\delta \in B(X,\varepsilon)} g(f, \delta, X)\}$.

The loss function for the Locally-Linear Regularization (LLR) model is

$$E\{L(f(X), Y) + \lambda\gamma(\varepsilon, X) + \mu||\delta^T_{LLR} \nabla_X L(f(X), Y)||\}.$$

3.2.3 RST

Robust self-training [17] is a defense proposed to improve the tradeoff between standard and robust error that occurs with AT. RST uses the following optimization problem for the loss function:

$$\min_f E\{L(f(X), Y) + \beta \max_{X` \in B(X,\varepsilon)} L(f(X`), Y)\}.$$

3.2.4 TRADES

One of the best methods for robustness via smoothness is TRADES [18], which has been shown to obtain state-of-the-art adversarial accuracy in many cases. TRADES uses the following optimization problem for the loss function, where the second term encourages local Lipschitzness:

$$\min_f E\{L(f(X), Y) + \beta \max_{X` \in B(X,\varepsilon)} L(f(X), f(X`))\}$$

3.2.5 Gradient Regularization

GR is in the form of soft regularization. We use the latest work by Finlay and Oberman [19] for our experiments. In general, GR models can be formulated as adding a regularization term on the norm of the gradient of the loss function:

$$min_{\ f} E\{L(f(X),\ Y) + \beta||\nabla XL(f(X),\ Y)||_2^2\}.$$

3.2.6 Data augmentations and weight averaging for improved adversarially robust training

Rebuffi et al. [7] demonstrated that combining heuristics-based data augmentations with model weight averaging can significantly improve robustness.
Having explored the generative models they came to the conclusion that generated samples provide a greater diversity of augmentations that help densify the image manifold and allow adversarial training to go well beyond the state-of-the-art.
Well-known heuristic-driven augmentations, such as MixUp and Cutout were used. In combination with variations of GANs and VAEs, this approach helps improve robustness greatly.

HEURISTICS-DRIVEN AUGMENTATIONS				
MixUp	95.93%	0.32%	3.75%	0.18
Cutout	94.15%	0.22%	5.63%	0.23
CutMix	84.04%	4.45%	11.51%	**0.53**
DATA-DRIVEN AUGMENTATIONS				
VDVAE (Child, 2021)	6.71 %	5.76%	87.53%	0.46
BigGAN (Brock et al., 2018)	11.53%	10.51%	77.96%	0.68
DDPM (Ho et al., 2021)	21.79%	20.16%	58.05%	**0.97**

Table 1. Data Augmentations used by Rebuffi et al.

3.2.7 Adversarial Weight Perturbation (AWP)

Wu et al. have proposed Adversarial Weight Perturbation (AWP) [6] to explicitly flatten the weight loss landscape via injecting the worst-case weight perturbation into DNNs. In order to improve the test robustness, we need to focus on both the training robustness and the robust generalization gap (delivered by the flatness of the weight loss landscape). Thus, the objective is

$$\min_w \{\rho(w) + \rho(w + v) - \rho(w))\} \rightarrow \min_w \rho(w + v),$$

where $\rho(w)$ is the adversarial loss, $\rho(w + v) - \rho(w)$ is a term to characterize the flatness of weight loss landscape, and v is weight perturbation that needs to be carefully selected.

3.2.8 Self-COnsistent Robust Error [20]

To eliminate the robustness-accuracy trade-off, the definition of robust error is slightly modified. KL-divergence is incorporated in the new score, Self-COnsistent Robust Error (SCORE), which is formulated as

$$R_{SCORE}(\theta)=E_{pd(x)}\,[max_{\,x`\in B(x)}\, KL(p_d(y|x`)\,||\,p_\theta(y|x`))].$$

4 Experiment prerequisites

4.1 Jitter attack

Jitter attack [21], one of the recent powerful black-box attacks, was selected as a main attacking approach for the experiments. Incorporating scaling, introducing Gaussian noise, and replacing cross-entropy loss with Euclidian loss makes it a great candidate for benchmarking. The details of the attack are listed below.

Scaling softmax

Previous work already demonstrated that high output logits can lead to gradient obfuscation and weaken adversarial attacks. We additionally observe that a small value range of the logits can also lead to attack failure. We propose to scale the softmax function by the following rule:

$$z\hat{} = softmax(\alpha \cdot z / (||z||_\infty)) \quad (1)$$

where α is an easy-to-tune scalar value that controls the lowest and largest possible output values of the softmax function ($0 < \alpha <= 83$ due to potential 32-bit float overflow).

Swapping CE loss with Euclidian loss

The authors proposed to exchange the CE loss function with *the Euclidean distance* between the rescaled softmax output $z\hat{}$ and the one-hot encoded ground truth vector Y. The Euclidean distance increases fastest by maximizing the magnitude of any of the output logits where $z_i \; != z_y$, which simultaneously minimizes the distance between Y and z_y. Combining the euclidean loss function and the scaling described in the loss function can be described by the following equation.

$$L_2 = ||z\hat{} - Y||_2. \quad (2)$$

Adding Gaussian noise

To encourage the attack to explore different gradient directions we additionally perturb the logits after each forward pass with Gaussian noise, where the noise magnitude is controlled by the hyperparameter σ. The resulting loss function is given by:

$$L_{Noise} = ||z\hat{} + N(0, \sigma) - Y||_2. \quad (3)$$

Minimizing the norm of the perturbation

Finally, we aim to encourage the attack to find small perturbations. As long as no successful perturbation is found, the loss function presented in (3) is applied. Once the adversarial attack is successful we additionally consider the norm of the adversarial perturbation. Furthermore, the current perturbation is only overridden if the newly found perturbation also leads to a successful attack. This procedure can never decrease the success rate of the attack and effectively minimizes the norm of the adversarial perturbation in our experiments. Moreover, the norm (or other distance measures) can be freely chosen according to the respective problem (e.g., L_1, L_2, L_∞) as long as it is differentiable.

Formulating final jitter loss

The final loss function can be described as follows:

L_{Jitter} = {

$(||z\hat{}-Y+N(0,\sigma)||_2/||\gamma||_p$ if x_{adv} is misclassified,

$||z\hat{} - Y + N(0,\sigma)||_2$ if x_{adv} is not misclassified yet

}

Tuning jitter loss

The authors explored different values for α between 2 and 20 and observed a stable performance. They tuned σ for every model individually on a batch of 100 samples by testing values for $\sigma \in \{0, 0.05, 0.1, 0.15, 0.2\}$. Note that tuning σ on a small batch for each model introduces only a negligible overhead (≈ 1% additional runtime).

4.2 Dataset and threat-model selection

The dataset selected was **CIFAR-10** with **L-inf** distance and **eps=8/255**. While it is the most relevant and consistent benchmark in adversarial robustness, it also has many different pretrained adversarial models, that can be attacked by new approaches. Taking into consideration the convenient API of both defensive models from **RobustBench** and attacks from **torchattacks**, the decision towards using these repositories was made.

The model f is assumed to be fully known to the attacker. The threat model is defined by the set Δ of the allowed perturbations: the most widely studied ones are the L_p-perturbations, i.e. $\Delta_p = \{\delta \in R^d, ||\delta||_p \leq \varepsilon\}$, particularly for $p = \infty$. The thresholds ε established in the literature are chosen such that the true label should stay the same for each in-distribution input within the perturbation set. We note that robustness towards small L_p-perturbations is a necessary but not sufficient notion of robustness which has been criticized in the literature.

Data subsampling

The subset of CIFAR-10 is 300 class-balanced samples. The sampling was done to speed up the experiment process as most attacks are computationally expensive, and the perturbation using one attack took almost an hour on 1000 samples.

5 Experiments

5.1 Defenses selection

The three main defensive models for the experiments are:
Data augmentations & weight averaging [7], *Adversarial Weight Perturbation* [6], *and Self-COnsistent Robust Error model* [20] – they are all-around among the best defenses available for the selected benchmark.

5.2 Attacks and ensembles selection

The main attack, Jitter, is evaluated using 9 different configurations, varying *std* and *scale* parameters.

```
# same balance of std and scale
jitter_1 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=5, std=0.05)  # low ### Best
jitter_2 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=10, std=0.1)  # standard
jitter_3 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=15, std=0.15) # high

# standard std, different scale
jitter_4 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=5, std=0.1)  # low
jitter_5 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=15, std=0.1) # high

# high/low std, standard scale
jitter_6 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=10, std=0.05) # low
jitter_7 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=10, std=0.15) # high

# contrasting std and scale
jitter_8 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=5, std=0.15) # low-high
jitter_9 = torchattacks.Jitter(model, eps=8/255, alpha=2/255, steps=40, scale=15, std=0.05) # high-low
```

Then, two other attacks were tested – default configurations for FFGSM [22] and the strongest extension of PGD (UPGD) mentioned in section 3.1.

```
ffgsm = torchattacks.FFGSM(model, eps=8/255, alpha=10/255)
upgd = torchattacks.UPGD(model, eps=8/255, alpha=1/255, steps=40, random_start=False)
```

```
### ensemble of black-box attacks
ensemble = torchattacks.MultiAttack([jitter_1, ffgsm, upgd])
```

Lastly, the ensemble of 3 attacks – the best variation of Jitter + FFGSM + UPGD – was tested.

6 Results

Defense 1. Fixing Data Augmentation to Improve Adversarial Robustness

#	Model ID	Paper	Clean accuracy	Robust accuracy	Architecture	Venue
1	Rebuffi2021Fixing_70_16_cutmix_extra	*Fixing Data Augmentation to Improve Adversarial Robustness*	92.23%	66.56%	WideResNet-70-16	arXiv, Mar 2021

#	Attack	Std (Jitter)	Scale (Jitter)	Robust accuracy, %	Inference time, s
1	**Jitter**	**0.05**	**5**	**20.67**	863
2	Jitter	0.1	10	31.67	863
3	Jitter	0.15	15	36	853
4	Jitter	0.1	5	22.67	834
5	Jitter	0.1	15	35.67	837
6	Jitter	0.05	10	31.33	838
7	Jitter	0.15	10	33.67	834
8	Jitter	0.15	5	25.67	855
9	Jitter	0.05	15	31.33	851
10	FFGSM	-	-	37.33	**34**

11	UPGD	-	-	23.33	908
12	Ensemble	0.05	5	21.33	1146

Table 2. Attacks against the augmentation defense.

Ensemble robust accuracy: 21.33%.

Fastest attack: **FGSM** with 34 seconds.

Best result: 20.67% – **Jitter** with reduced hyperparameter values.

Defense 2. Adversarial Weight Perturbation (AWP)

#	Attack	Std (Jitter)	Scale (Jitter)	Robust accuracy, %	Inference time, s
1	**Jitter**	**0.05**	**5**	**30**	193
2	Jitter	0.1	10	32	191
3	Jitter	0.15	15	39.67	190
4	Jitter	0.1	5	30.67	190
5	Jitter	0.1	15	36.33	190
6	Jitter	0.05	10	32	190
7	Jitter	0.15	10	34	190
8	Jitter	0.15	5	31.33	191
9	Jitter	0.05	15	36	189
10	FFGSM	-	-	48.67	**6.5**

11	UPGD	-	-	34.33	153
12	**Ensemble**	**0.05**	**5**	**30**	236

Table 3. Attacks against the AWP defense.

Ensemble robust accuracy: 30%.

Fastest attack: **FGSM** with 6.5 seconds.

Best result: 20.67% – **Jitter** with reduced hyperparameter values and ensemble (tied).

Defense 3. Self-COnsistent Robust Error

#	Attack	Std (Jitter)	Scale (Jitter)	Robust accuracy, %	Inference time, s
1	**Jitter**	**0.05**	**5**	**26**	837
2	Jitter	0.1	10	31.33	832
3	Jitter	0.15	15	35.33	831
4	Jitter	0.1	5	26.67	832
5	Jitter	0.1	15	32.33	831
6	Jitter	0.05	10	29.67	832
7	Jitter	0.15	10	31	832
8	Jitter	0.15	5	27	832
9	Jitter	0.05	15	31.67	832
10	FFGSM	-	-	39.33	**31**
11	UPGD	-	-	30.67	808

12	Ensemble	0.05	5	26.33	1074

Table 4. Attacks against SCORE defense.

Ensemble robust accuracy: 26.33%.

Fastest attack: **FFGSM** with 31 seconds.

Best result: 26% – **Jitter** with reduced hyperparameter values.

Notable observations

Ensembles

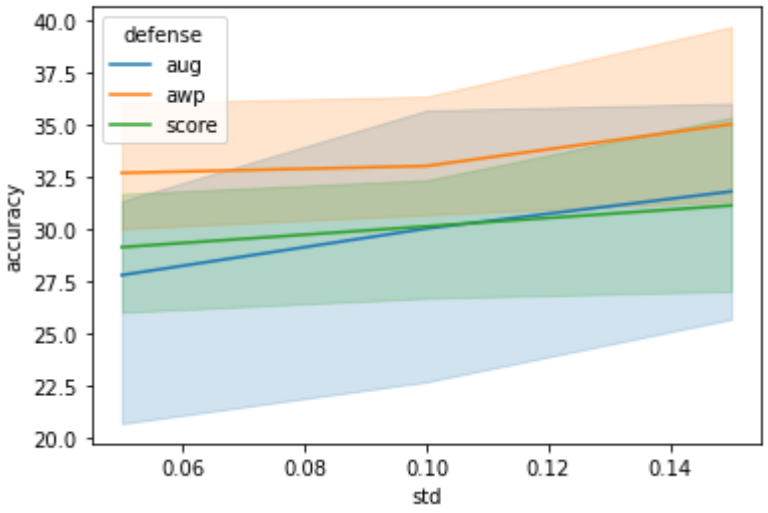
Different ensembling techniques performed slightly worse and took more time than an individual best attack (Jitter in this case). While the longer time is explained by sequential execution of attacks, higher robust accuracy should have improved over a singular attack. However, it appears as though combining weak in terms of score attacks (FFGSM) or powerful, but completely different approach-wise attacks (UPGD using Cross-Entropy loss) may lead to noisier perturbations that counter-align and do not synergize with each other, leading to higher perturbation norm, and lower impact on robust accuracy.

AWP Defense

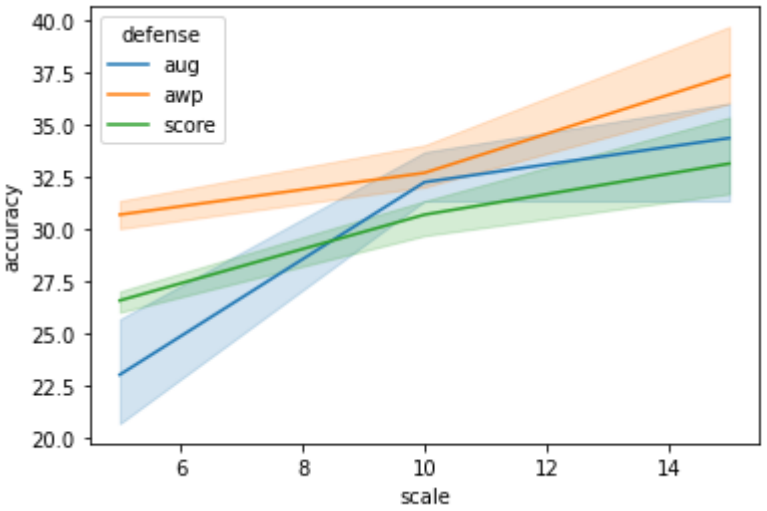
Adversarial weight perturbation defense came to be much more resilient, even though the attacks seemed to infer ~4 times faster. This phenomenon can potentially be explained by certain defensive perturbations that allow the attacks to find suboptimal perturbations in a much shorter time. Hence, the attacking perturbations are exhausted earlier and are prone to have a weaker overall performance – 30% robust accuracy, compared to 20.67% and 26% of augmentation defense and SCORE defense, respectively. It is worth mentioning that both weaker defenses averaged 830 seconds per attack inference, whereas AWP averaged only 190 seconds. This may lead to an assumption that there exists a certain tradeoff between the attack's maximum inference time and lowest resulting robust accuracy.

Scale and *std* hyperparameters in Jitter

9 different configurations were tested in the experiments. All the combinations of higher scale and standard deviation lead to worse results than the lower counterparts. While it is obvious that reducing the amount of noise injected as well as not doing a large scaling leads to more powerful attacks with lower-norm perturbations, it does not guarantee, however, that the attack is consistent and robust itself. It is very possible such an attack makes use of obfuscated gradients and finds weaknesses for the specific defense, which can, in turn, be countered by an improved and fine-tuned adversarial model. Thus, it is assumed that increasing both scale and standard deviation will lead to a more regularized, lower-norm, and overall more robust attack.



Plot 1. *Std* impact on robust accuracy across different defenses.



Plot 2. *Scale* impact on robust accuracy across different defenses.

7 Conclusion and future prospects.

Using ensembles of novel variants of adversarial attacks leads to further reduction of maximum robust accuracy of adversarially trained models. The Jitter attack is capable of undermining the obfuscated gradients of most defenses, yields significantly lower perturbation norm in comparison to most other attacks, and successfully perturbs robustly overestimated defenses. Additionally, it is a universal black-box model that can be applied in different L-p metric spaces, under different threat models. The replacement of Cross-Entropy with Euclidian loss, incorporation of Gaussian noise, and pre-softmax scaling is what allows the attack to stand out, and be further enhanced to combat future newly-emerged defenses.

In the course of this work, certain discoveries were made. They include a confirmation of a known Machine Learning paradigm that mixing considerably weaker predictors (attacks in our case) in an ensemble will lead to performance deterioration. Adversarial perturbation defense appeared the strongest against multiple attacks showing the highest robust accuracy in the end. However, 4 times less time was required to infer an attack. Even though this phenomenon is intuitive, it should be researched more closely. Undoubtedly, the topic of selection of hyperparameters *scale* and *std* in Jitter attack is controversial. Despite lower values for both leading to best results, there is no guarantee that the attacks are just finding the very specific perturbations for the given defense, which is made easier by less scaling and injecting less Gaussian noise. Therefore, this observation should be studied more elaborately as well.

As an extension of this work, a more elaborate analysis of different attacking ensembles will be researched. This includes a thorough choice of Jitter

hyperparameters and testing different variations of PGD (APGD-DLR). For the next more sophisticated set of experiments, the most recent and modern defenses will be selected, and the sample size on the benchmark will also be increased.

References

[1] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, Kamalika Chaudhuri. A Closer Look at Accuracy vs. Robustness. *arXiv preprint arXiv:2003.02460*, 2020.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[3] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.

[4] Nicholas Carlini, David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv preprint arXiv:1608.04644*, 2016.

[5] Francesco Croce, Matthias Hein. Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks. *arXiv preprint arXiv:2003.01690*, 2020.

[6] Dongxian Wu, Shu-Tao Xia, Yisen Wang. Adversarial Weight Perturbation Helps Robust Generalization. *arXiv preprint arXiv:2004.05884*, 2020.

[7] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, Timothy Mann. Fixing Data Augmentation to Improve Adversarial Robustness. *arXiv preprint arXiv:2103.01946*, 2021.

[8] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, John C. Duchi. Unlabeled Data Improves Adversarial Robustness. *arXiv preprint arXiv:1905.13736*, 2022.

[9] Hoki Kim. Torchattacks: A PyTorch Repository for Adversarial Attacks. *arXiv preprint arXiv:2010.01950*, 2021.

[10] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[11] Anish Athalye, Nicholas Carlini, David Wagner. Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[12] Francesco Croce, Matthias Hein. Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack. *arXiv preprint arXiv:1907.02044*, 2019.

[13] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, Matthias Hein. Square Attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2020.

[14] Jinghui Chen, Quanquan Gu. RayS: A Ray Searching Method for Hard-label Adversarial Attack. *arXiv preprint arXiv:2006.12792*, 2020.

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

[16] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pages 13824–13833, 2019.

[17] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. *arXiv preprint arXiv:2002.10716*, 2020.

[18] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *InternationalConference on Machine Learning*, 2019.

[19] Chris Finlay and Adam M Oberman. Scaleable input gradient regularization for adversarial robustness. *arXiv preprint arXiv:1905.11468*, 2019.

[20] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, Shuicheng Yan. Robustness and Accuracy Could Be Reconcilable by (Proper) Definition. *arXiv preprint arXiv:2202.10103*, 2022.

[21] Leo Schwinn, René Raab, An Nguyen, Dario Zanca, Bjoern Eskofier. Exploring misclassifications of robust neural networks to enhance adversarial attacks. *arXiv preprint arXiv:2105.10304*, 2021.

[22] Eric Wong, Leslie Rice, J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[23] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, pages 5133–5142, 2018.