Mathematical Problems of Computer Science 58, 42–51, 2022.
doi:10.51408/1963-0091

UDC 519.6, 004.9

# A Brief Comparison Between White Box, Targeted Adversarial Attacks in Deep Neural Networks

Grigor V. Bezirganyan and Henrik T. Sergoyan

Department of Mathematics, Technical University of Munich, Muinch, German
e-mail: grigor.bezirganyan@tum.de, henrik.sergoyan@tum.de

**Abstract**

Today, neural networks are used in various domains, in most of which it is critical to have reliable and correct output. This is why adversarial attacks make deep neural networks less reliable to be used in safety-critical areas. Hence, it is important to study the potential attack methods to be able to develop much more robust networks. In this paper, we review four white box, targeted adversarial attacks, and compare them in terms of their misclassification rate, targeted misclassification rate, attack duration, and imperceptibility. Our goal is to find the attack(s), which would be efficient, generate adversarial samples with small perturbations, and be undetectable to the human eye.
**Keywords:** Adversarial Attacks, Robustness, Machine Learning, Deep Learning.
**Article info:** Received 26 Aprile 2022; received in revised form 4 July 2022; accepted 29 July 2022.

## 1. Introduction

Nowadays, deep neural networks are becoming more and more popular to solve problems in various domains, including safety-critical areas such as medicine, self-driving cars, etc. Unfortunately, techniques to fool deep learning models have recently come out to provide incorrect outputs [1]. Particularly, in the image classification domain, an attacker can create an altered image, which will be misclassified by a model but will be classified correctly by a human. This altered image is often referred to as an *adversarial example*, and this process as an *adversarial attack*. To be protected against such attacks, researchers try to create methods to make the models more robust against such perturbations. Studying adversarial attacks and their potential helps us develop better countermeasures against them.

In this paper, we will discuss some of the adversarial algorithms and test them against an image classification model. We then compare the results of the experiments in terms of their misclassification rate, targeted misclassification rate, attack duration, and imperceptibility.

## 1.1 Definitions and Notations

### 1.1.1 Poisoning Attacks vs Evasion Attacks

In **poisoning attacks**, the attacker tries to insert fake samples (i.e., data samples with wrong labels) into the training dataset, which will make the model learn on those fake samples and output wrong results. This kind of attack is possible when the attacker has the means to import those fake samples into the training set. In contrast, in **evasion attacks**, the attacker does not need access to the dataset. In this case, the attacker creates adversarial samples, which are similar and hard to distinguish by a human from the original samples but are misclassified by the trained model.

### 1.1.2 Attacker's Knowledge of the Model

Based on how much information the attacker has about the model, attacks can be classified into white-box, black-box, and gray-box attacks. In the **white box** scenario, the attacker has full knowledge about the model architecture and uses this knowledge to generate adversarial examples. In contrast, in the **black-box** setup, the attacker does not know the architecture. Instead, the attacker observes the output of the model from the given input. Some of the attacks assume access to the soft labels (i.e., probability or likelihood score of belonging to a class), while others try to generate examples based on only hard labels (i.e., class labels without the score). In the **gray-box** setting, the attacker has an access to the original model and trains a generative model on it. When the generative model is ready, the attacker uses that model to generate adversarial samples. Hence, the original model is no more needed. Recently, in [2] another category was introduced, called **no-box** attacks. In contrast to black-box attacks, the attacker cannot query the model, instead, he has a small number of samples from the same domain as the victim. The authors train an auto-encoder on those samples and then generate the adversarial examples using the features learned from the auto-encoder.

### 1.1.3 Targeted vs Non-Targeted Attacks

In the **targeted attack**, the attacker tries to misclassify the given sample into a specific target label. In contrast, in **non-targeted** attacks, the attacker tries to classify the sample into any other class.

## 1.2 Our Goal and Contribution

In this paper, we try to overview some of the adversarial attack techniques and, running experiments in the same setting, compare them based on:

- **Misclassification:** What percentage of the adversarial samples were misclassified
- **Targeted Misclassification**: What percentage of the adversarial samples were successfully misclassified to the target class
- **Imperceptibility:** How much the adversarial example looks like the original image
- **Duration of the attack**: How long it takes to generate an adversarial example

In this paper, we will concentrate only on white-box and target attacks. In particular, we will discuss and experiment with the Fast Gradient Sign Method [1], Projected Gradient Descent [3], AutoPGD [4], and FW + Dual LMO [5]. We chose these attack methods as FGSM is one of the first and simplest methods, which is still popular today. PGD is the most popular, as even many new state-of-the-art methods are modified versions of the PGD attack. AutoPGD, being one of those variations, achieves state-of-the-art results according to the authors. And while these attacks use $\ell_p$ norms, we also chose FW + Dual LMO as an example of an attack that uses another norm (Wasserstain norm in this case).

## 2. Attack Mechanisms

In this section, we will briefly overview the attacks, which will be used for experimentation further in the paper.

In our attacks, we are given a set of input images $x \in \mathbb{R}^{n\times n}$, and our goal is to craft an adversarial example $x' \in \mathbb{R}^{n\times n}$ that will be misclassified by the deep learning model $F : \mathbb{R}^{n\times n} \to \mathbb{N}$. Since we are discussing targeted attacks, we want to misclassify the adversarial sample into our desired target class $t \in \mathbb{N}$ instead of the original class $y \in \mathbb{N}$. Furthermore, the perturbation we add to the image should be as small as possible, not to be detected by a human. So, we can formulate the problem in the following way: Given a Neural Network $F : \mathbb{R}^{n\times n} \to \mathbb{N}$, input image $x \in \mathbb{R}^{n\times n}$ with a label $y \in \mathbb{N}$, a distance function $||\cdot||$ and a perturbation budget $\epsilon \in \mathbb{R}$ find an $x' \in \mathbb{R}$ such that

$$\begin{aligned} & F(x') = t \neq y \\ \text{s.t.} \quad & ||x' - x|| \leq \epsilon. \end{aligned} \tag{1}$$

In our case, the distance functions will be $l_1, l_2, l_\infty$ distances or the Wasserstein distance.

### 2.1 Fast Gradient Sign Method (FGSM)

Since we can access the gradients of the network in the white-box setting, what most of the gradient-based attacks do, is to fix the network weight and maximize the loss by updating the image. For that, they add a small perturbation $\eta \in \mathbb{R}^{n\times n}$ to the original image:

$$x' = x + \eta$$

The most efficient way to maximize the loss would be to add noise in the same direction as the gradients. [1] introduced an attack method, where they do exactly that: add a perturbation in a direction that will increase the loss function $\mathcal{L}$ between the adversarial example and the original label

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, y)). \tag{2}$$

We can see that in this way the maximum allowed perturbation is added, while still being in the $\epsilon$ ball.

For a targeted setting, the update step will become:

$$x' = x - \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\theta, x, t))$$

in other words, a perturbation is added to minimize the loss between the adversarial sample and the target class $t$.

## 2.2 Projected Gradient Descent

The Projected Gradient Descent attack (PGD) or Basic Iterative Method (BIM) was introduced in [3], where they transformed the FGSM [1] one-step attack into an iterative one by performing the update step (2) multiple times with a small step size $\alpha \in \mathbb{R}^{n\times n}$. This will work better, as the FGSM adds the maximum allowed perturbation, but does not guarantee to maximize the loss within the allowed $\epsilon-$ball. In contrast, in an iterative approach, the algorithm is more likely to find the maxima. To ensure that the adversarial sample remains in the $\epsilon$ neighborhood, PGD projects the sample back to the $\epsilon$ ball after each update step. In other words, it performs projected gradient descent (or ascent) on the input sample. The update steps for targeted and untargeted attacks will be as follows:

$$x^{(i+1)} = \Pi_\epsilon(x^{(i)} + \alpha \cdot \text{sign}(\nabla_{x^{(i)}}\mathcal{L}(\theta, x^{(i)}, y))) \tag{3}$$

$$x^{(i+1)} = \Pi_\epsilon(x^{(i)} - \alpha \cdot \text{sign}(\nabla_{x^{(i)}}\mathcal{L}(\theta, x^{(i)}, t))) \tag{4}$$

So, the attacker tries to find a perturbation that either finds the maximum loss between $x'$ and $y$ (3) (untargeted attack), or the minimum loss between $x'$ and $t$ (4) (targeted attack).

## 2.3 Auto-Projected Gradient Descent

It has recently been suggested [4] that the Cross-Entropy loss and the fixed step size of the PGD attack [3] may be two reasons for its potential failure. They propose an alternative loss function and a new gradient-based method, Auto-PGD, which does not require a fixed step size.

They divide their method into two phases: an exploration phase and an exploitation phase. During the exploration phase, they search for good initial points, while in the exploitation phase, they try to maximize the accumulated knowledge. The step size value depends on the trend of optimization. If the objective function decreases rapidly, then the step size does not need to be changed, otherwise, if it decreases slowly, the step size is reduced.

## 2.4 Wasserstein Attack

The Wasserstein adversarial attack was introduced in [6]. Here they proposed to use the Wasserstein distance instead of the commonly used $\ell_p$ distances. For images, the Wasserstein distance can be seen as the cost of redistributing pixel mass. For example, while rotations change $\ell_p$ norms dramatically, they only slightly change the Wasserstein distance.

So, what their algorithm does, is to do a PGD attack [3], but instead of projecting on an $\ell_p$ norm, they project on the Wasserstein ball. However, since the projection onto the Wasserstein ball is computationally expensive, they make an approximation by performing modified Sinkhorn iterations [7].

[5] improved the algorithm by introducing an exact but still efficient projection operator. They also introduce an adversary generating method based on the Frank-Wolfe [8] method equipped with a suitable linear minimization oracle and show that it works very fast for Wasserstein constraints.

In this paper, we will use that Frank-Wolfe method (FW + Dual LMO) for the experiments.

## 3. Experiments

### 3.1 Goal

In this experiment, our goal is to run FGSM [1], PDG [3], AutoPGD [4], and FW + Dual LMO [5] attacks on the same environment and compare them in terms of misclassification, targeted misclassification, attack duration, and imperceptibility.

### 3.2 Setup

We are performing our experiments on a pre-trained ResNet-18 [9] classifier on the CIFAR-10 dataset [10], with initial 92.4% accuracy on the test set. We generate the adversarial examples on a server with an Nvidia GeForce GTX 1080-Ti GPU.

We use the Adversarial Robustness Toolkit (ART) [11] for FGSM [1] and PGD [3] and AutoPGD [4] attacks, and the original implementation by the authors for FW + Dual LMO [5]. We run each of the adversarial attacks with a set of epsilon values in $\epsilon \in (0, 0.5]$ and for all target classes. We use $\ell_p$ norms for FGSM, PGD, and AutoPGD, and we use the Wasserstein distance for the FW + Dual LMO. All the other hyper-parameters are left to their default values. For the FW + Dual LMO, in the original implementation, there was no option for targeted attacks. Hence, we modified their implementation and added the option for target attacks. For that we converted the problem:

$$\begin{aligned} &\text{maximize} \quad \mathcal{L}(F(x'), y) \\ &\text{subject to } ||x' - x|| \leq \epsilon \end{aligned}$$

to

$$\begin{aligned} &\text{minimize} \quad \mathcal{L}(F(x'), t) \\ &\text{subject to } ||x' - x|| \leq \epsilon \end{aligned}$$

We log the duration of the attack, the misclassification rate, and the targeted misclassification rate for later comparison. The source code for the experiment can be found $https://github.com/bezirganyan/adversarial_arenahere.$

## 4. Results

### 4.1 Targeted Misclassification and Misclassification Rate

We first look at the average misclassification and targeted misclassification scores that each of our models was able to achieve for some $\epsilon \in (0, 0.5]$. In Table 1, we can see average misclassification and targeted misclassification rates for the best epsilon of each attack. As we can see from the $\ell_p$ attacks, the $\ell_\infty$ norm yields the highest scores in our setup. Hence, from now on we will use the $\ell_\infty$ norm for further comparisons. Note that this does not mean that the $\ell_\infty$ norm is better since we could get similar scores and similar perturbations for higher $\epsilon$ values under other norms, as the $\ell_\infty$ attack will add a higher amount of perturbation under the same epsilon.

Furthermore, we can see that from the $\ell_p$ attacks in terms of targeted misclassification rate, the PGD, and AutoPGD attacks yield very high scores leaving the FGSM attack behind with a huge margin. In general, PGD and AutoPGD attacks behave almost identically in our experiments. We hypothesize that this is because we are testing on an undefended model, on which they both reach their maximum potential limit. The developers of the ART framework confirmed that on their tests on defended models in an untargeted setting, AutoPGD behaved slightly better. We, hence, plan to test and compare the models on a defended model in our future work. In Fig. 1, we can see the Misclassification and Targeted misclassification rates of the attacks for different epsilons and under the $\ell_\infty$ norm. We can see that in terms of misclassification and targeted misclassification rates the PGD and AutoPGD attack perform best within the $\ell_p$ attacks by having around 90% misclassification rate even for very small epsilon.

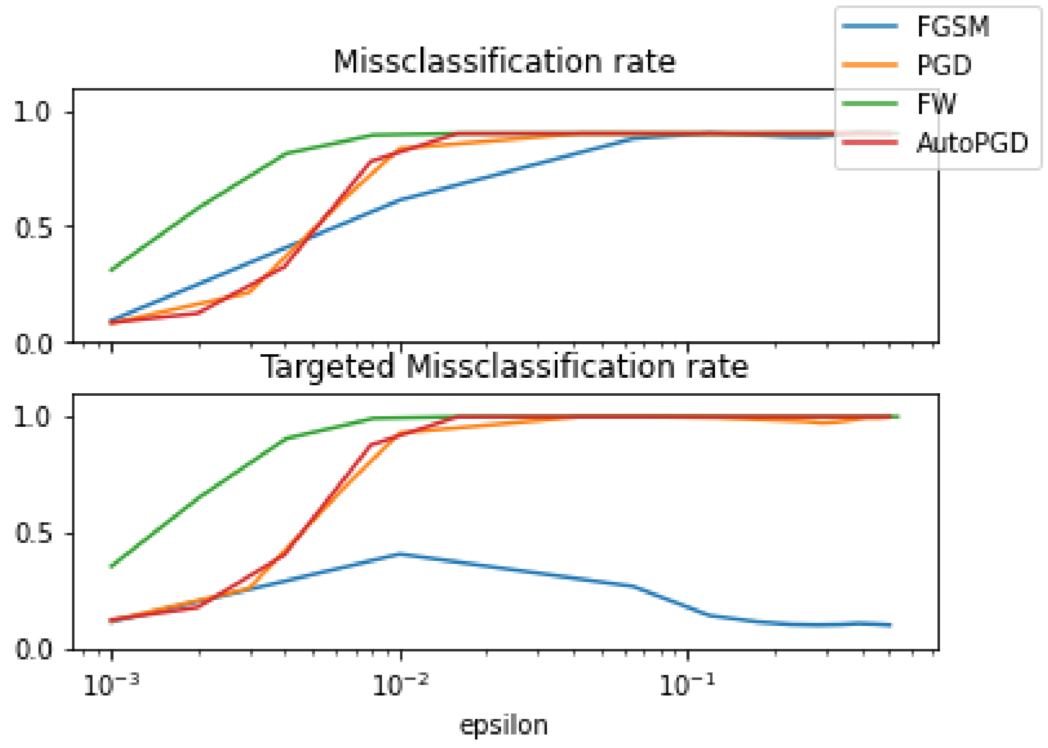

Fig. 1. Average misclassification and targeted misclassification rates for different $\epsilon$ values under $\ell_\infty$ and Wasserstein (FW) norms.

Furthermore, we can see that for the FGSM attack, the targeted misclassification does not increase monotonically. The reason for this can be that since the FGSM is not an iterative algorithm and performs just one step, it overshoots when the epsilon is too big and misses the target class.

The FW+Dual LMO attack performs best in terms of both misclassification and targeted misclassification rates. Nevertheless, we cannot compare the amount of perturbation under $\ell_\infty$ and Wasserstein norms, since they imply different amounts of changes to the image. Hence, we will need to combine these results with the visual ones to be able to make a fair comparison.

## 4.2 Duration

In Table 2, we can see the time duration needed to generate an adversarial example. Being a simple one-step attack, FGSM leads the competition followed by the PGD and AutoPGD

Table 1: Average misclassification and average targeted misclassification rates for different norms

| attack | norm | miscl | targ. miscl. |
|---|---|---|---|
| AutoPGD | $\ell_1$ | 0.0832 | 0.1100 |
| PGD | $\ell_1$ | 0.0810 | 0.1086 |
| FGSM | $\ell_1$ | 0.0879 | 0.1116 |
| AutoPGD | $\ell_2$ | 0.8968 | 0.9977 |
| FGSM | $\ell_2$ | 0.6157 | 0.3914 |
| PGD | $\ell_2$ | 0.8927 | 0.9925 |
| AutoPGD | $\ell_\infty$ | 0.9000 | 1.0000 |
| FGSM | $\ell_\infty$ | 0.9149 | 0.5515 |
| PGD | $\ell_\infty$ | 0.9022 | 1.0000 |
| FW | was | 0.9000 | 1.0000 |

attacks. PGD, which performs much better than FGSM in terms of targeted misclassification rate, is around 71 times slower. The slowest is the FW + Dual LMO attack, which performs around 400 times slower than the FGSM attack.

## 4.3 Duration

In Table 2, we can see the time duration needed to generate an adversarial example. Being a simple one-step attack, FGSM leads the competition followed by the PGD and AutoPGD attacks. PGD, which performs much better than FGSM in terms of targeted misclassification rate, is around 71 times slower. The slowest is the FW + Dual LMO attack, which performs around 400 times slower than the FGSM attack.

Table 2: Duration of generating an adversarial example in seconds.

| FGSM | PGD | AutoPGD | FW+Dual LMO |
|---|---|---|---|
| 0.7 | 50 | 87 | 338 |

## 4.4 Imperceptibility

One of the most important aspects of Adversarial attacks is that they should be undetected by the human eye. Hence, in this section, we study how detectable are the adversarial samples generated by the attacks. To visualize the results, we chose the smallest $\epsilon$ for each of our attacks, under which our model showed at least 80% misclassification. You can see the visualizations in the Figures 2 and 3. We can see that in the examples generated by the FGSM attack, although the original image is still well visible, the perturbation is easily detectable to us. For PGD, AutoPGD, and FW + Dual LMO attacks, however, the perturbations are hardly visible. In fact, from Fig. 3 it is noticeable that PGD and AutoPGD attacks apply small perturbations uniformly over the image. While the FW + Dual LMO attack perturbs only small portions of the image, the perturbations are much more visible.

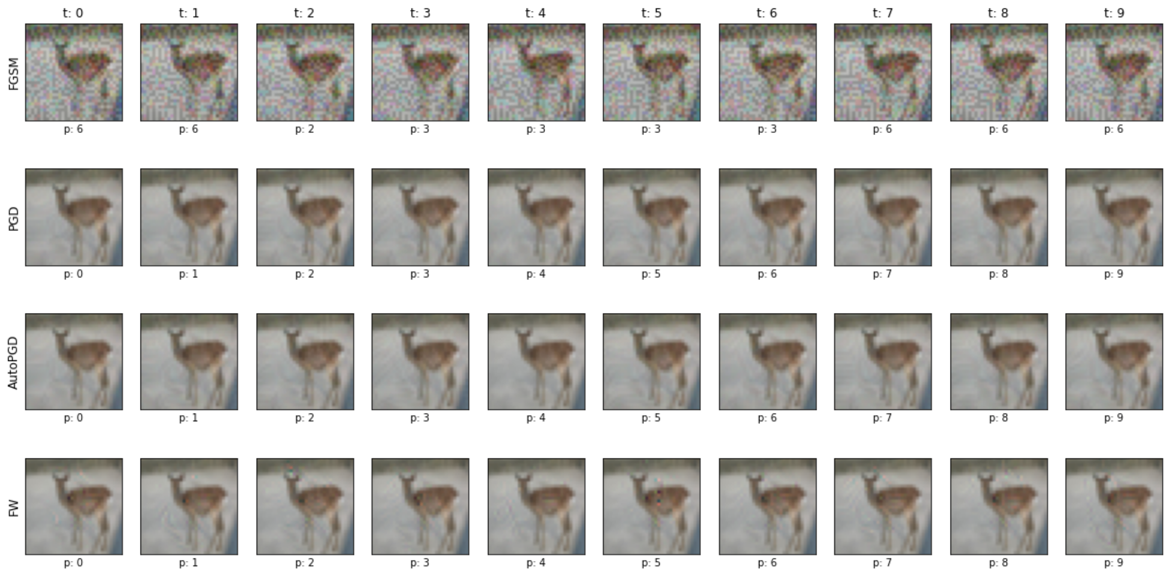
t: 0
t: 1
t: 2
t: 3
t: 4
t: 5
t: 6
t: 7
t: 8
t: 9
FGSM
PGD
AutoPGD
FW
p: 6 p: 6 p: 2 p: 3 p: 3 p: 3 p: 3 p: 6 p: 6 p: 6
p: 0 p: 1 p: 2 p: 3 p: 4 p: 5 p: 6 p: 7 p: 8 p: 9

Fig. 2. Adversarial samples on an image with original label 4 (deer).

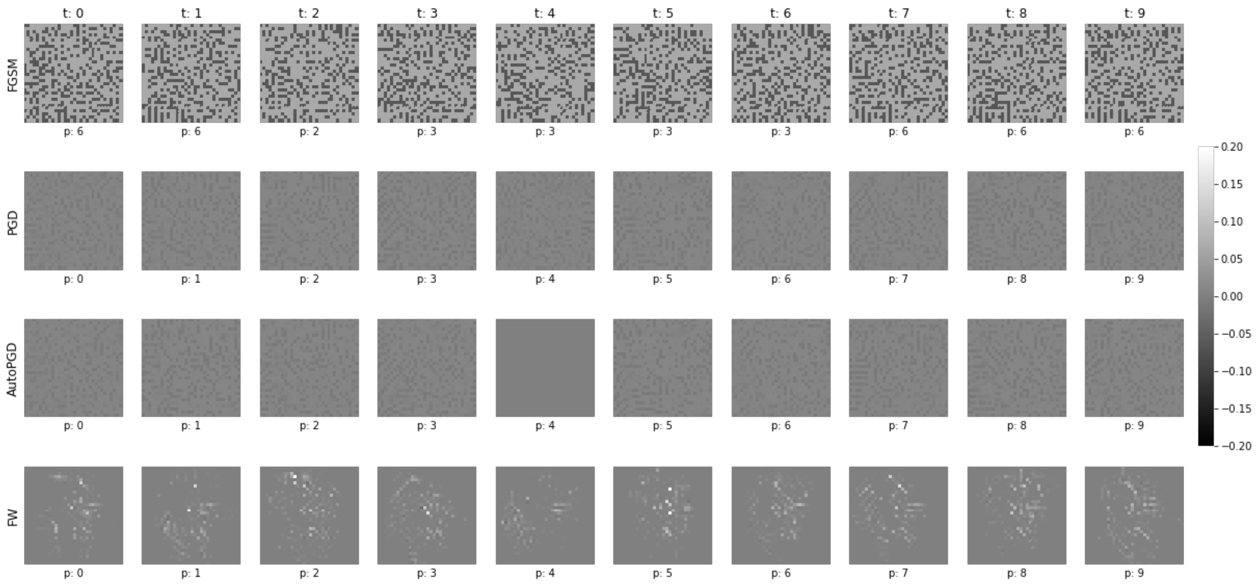
t: 0
t: 1
t: 2
t: 3
t: 4
t: 5
t: 6
t: 7
t: 8
t: 9
FGSM
PGD
AutoPGD
FW
p: 6 p: 6 p: 2 p: 3 p: 3 p: 3 p: 3 p: 6 p: 6 p: 6
p: 0 p: 1 p: 2 p: 3 p: 4 p: 5 p: 6 p: 7 p: 8 p: 9
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20

Fig. 3. Perturbations added to the image with original label 4 (deer).

## 5. Conclusion and Future Work

We compared different attack methods with different metrics. The champion of the comparison is the PGD attack. Although being a very simple attack, it performs very well in terms of misclassification and targeted misclassification rates, is fast, and is almost non-detectable by the human eye in our experiments. AutoPGD, while yielding similar results, is much slower, and hence, comes in second place in our comparison. FW + Dual LMO attack performed very well in terms of duration, misclassification, and targeted misclassification rates, but the perturbations were much more noticeable. The FGSM attack was the fastest with a high misclassification rate but came last in terms of imperceptibility.

Since we've covered only a small portion of attacks, we plan to extend the attack list by adding more well-known or state-of-the-art methods and extend the experiment domain to black-box attacks as well. Furthermore, we plan to test these attacks on a defended model and compare their performances. Particularly, we are interested to see the difference between AutoPGD and PGD attacks on a defended model.

## References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[2] Q. Li, Y. Guo, and H. Chen, "Practical no-box adversarial attacks against dnns," *Pre-proceedings - Advances in Neural Information Processing Systems*, vol. 33, 2020.

[3] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[4] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," *arXiv preprint arXiv:2003.01690*, 2020.

[5] K. Wu, A. Wang, and Y. Yu, "Stronger and faster wasserstein adversarial attacks," in *International Conference on Machine Learning*, pp. 10377–10387, PMLR, 2020.

[6] E. Wong, F. R. Schmidt, and J. Zico Kolter, "Wasserstein adversarial examples via projected sinkhorn iterations," in *36th International Conference on Machine Learning, ICML 2019*, 2019.

[7] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, pp. 2292–2300, 2013.

[8] M. Frank, P. Wolfe, *et al.*, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[10] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[11] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, I. Molloy, and B. Edwards, "Adversarial robustness toolbox v1.2.0," *CoRR*, vol. 1807.01069, 2018.

# Սպիտակ տուփով, թիրախավորված մրցակցային հարձակումների համառոտ համեմատությունը խորը նեյրոնային ցանցերում

Գրիգոր Բեզիրգանյան և Հենրիկ Սերգոյան

Մյունխենի Տեխնիկական Համալսարան
e-mail: grigor.bezirganyan@tum.de, henrik.sergoyan@tum.de

## Ամփոփում

Այսօր նեյրոնային ցանցերն օգտագործվում են տարբեր ասպարեզներում, որոնցից շատերում կարևոր է ունենալ հուսալի և ճշգրիտ արդյունք: Ահա թե ինչու մրցակցային հարձակումները նեյրոնային ցանցերը դարձնում են ավելի քիչ հուսալի՝ բարձր անվտանգության մակարդակ պահանջող տիրույթներում: Հետևաբար, կարևոր է ուսումնասիրել հարձակման հնարավոր մեթոդները՝ ավելի կայուն և անվտանգ ցանցեր մշակելու համար: Այս հոդվածում մենք քննարկում ենք չորս սպիտակ?տուփով, թիրախավորված մրցակցային հարձակումներ և համեմատում դրանք իրենց սխալ դասակարգման աստիճանի, նպատակային սխալ դասակարգման արագության աստիճանի, հարձակման տևողության և աննկատելիության առումով: Մեր նպատակն է գտնել հարձակում(ներ), որոնք արդյունավետ կլինեն և կստեղծեն մրցակցային օրինակներ՝ փոքր շեղումներով և աննկատելի մարդու աչքի համար:

**Բանալի բառեր՝** մրցակցային հարձակումներ, կայունություն, մեքենայական ուսուցում,խորը ուսուցում:

# Краткое сравнение между "Белым ящиком", целевыми состязательными атаками противника в глубоких нейронных сетях

Григор Безирганян и Генрик Сергоян

Технический Университет Мюнхена
e-mail: grigor.bezirganyan@tum.de, henrik.sergoyan@tum.de

## Аннотация

Сегодня нейронные сети используются в различных областях, в большинстве из которых важно иметь надежный и правильный вывод. Вот поэтому состязательные атаки делают глубокие нейронные сети менее надежными для использования в областях, где безопасность имеет решающее значение. Следовательно, важно изучить потенциальные методы атаки, чтобы иметь возможность разрабатывать гораздо более надежные сети. В этой статье мы рассматриваем четыре "белых ящика" - целенаправленные состязательные атаки и сравниваем их с точки зрения частоты ошибочных классификаций, частоты целевых ошибочных классификаций, длительности атаки и незаметности. Наша цель - найти атаки, которые были бы эффективны и генерировали бы состязательные выборки с небольшими возмущениями и не обнаруживались бы человеческим глазом.

**Ключевые слова:** состязательные атаки, надежность, машинное обучение, глубокое обучение.