Project for the exam - Optimization for Data Science

Frank-Wolfe for White Box Adversarial Attacks

Eleonora Brasola
Student n° 2007717

Alberto Cocco
Student n° 2020357

Greta Farnea
Student n° 2019052

1. Introduction

Machine learning models, like neural networks, are found to be vulnerable to *adversarial examples*, that are obtained by modifying examples drawn from the data distribution. The models tend to misclassify such examples that are slightly different from the correctly classified examples. The surprising thing is that this behaviour is proper of a wide variety of machine learning models: adversarial examples reveal some blind spots in the state-of-the-art training algorithms.

Many hypothesis have been made to explain such behaviour, a lot of which are related with complexity and depth of the neural networks. In [4], it is shown that it is sufficient to have a simple linear model in order to fool it with adversarial examples. It is interesting to notice that an adversarial example generated for one model is often misclassified by other models, moreover these models generally agree on the output, that is they miscalssify a sample into the same class. This can be explained as a result of 'adversarial directions' being highly related to the weights vectors of the model and the fact that in order to provide an adversarial example what matter the most are these direction and not the specific point in space. Moreover different models learn similar function when trained to perform the same task. We refer to this property and *transferability* of the adversarial example.

In this project, our goal is to implement four different algorithms for Adversarial Attacks. According to Goodfellow [4], Dong [3], Chen [2] and Rinaldi [5], we use the *Projected Gradient Descent*, the *Fast Sign Gradient*, the *Momentum Iterative Fast Sign Gradient* and the *Frank-Wolfe-white* methods.

All of these algorithms belong to the constrained optimization theory, thus their aim is to minimize (or maximize) a function and to comply some restrictions on the domain of the function.

1.1. Constrained optimization

The most general definition of a constrained problem is:

$$\begin{aligned} &\min f(x) \\ &\text{such that } x \in C \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $C \subseteq \mathbb{R}$ is a convex set.

We can define the *set of feasible directions* $F(\bar{x})$ of a point $\bar{x} \in C \neq \emptyset$ as:

$$\begin{aligned} F(\bar{x}) = \{d \in \mathbb{R}^n, d \neq 0 : \exists\, \delta > 0 \text{ s.t. } \bar{x} + \alpha\, d \in C, \\ \forall\, \alpha \in (0, \delta)\}. \end{aligned} \tag{2}$$

We recall this proposition:

Proposition 1.1.1. *Let $x^* \in C$ be local minimum for problem 1 with $C \subseteq \mathbb{R}^n$ convex and $f \in C^1(\mathbb{R}^n)$. Then*

$$\nabla f(x^*)^T (x - x^*) \geq 0 \quad \forall x \in C\,. \tag{3}$$

We can extend this proposition with another one that gives a necessary and sufficient condition for the global minimum:

Proposition 1.1.2. *Let $C \subseteq \mathbb{R}^n$ be a convex set and $f \in C(\mathbb{R}^n)$ be a convex function. A point $x^* \in C$ is a global minimum of problem 1 if and only if*

$$\nabla f(x^*)^T (x - x^*) \geq 0 \quad \forall x \in C\,.$$

All these properties are valid for a minimization constrained problem such as 1.

They can be extended also for a maximization constrained problem, defined as:

$$\begin{aligned} &\max f(x) \\ &\text{such that } x \in C\,. \end{aligned}$$

It is an easy derivation since $\max f(x) \equiv \min -f(x)$.

One finial key ingredient for constrained optimization is:

Weierstrass Theorem A continuous function in $\mathbb{R}$ over a convex set always admits both global maximum and minimum.

1.2. Adversarial attacks

Adversarial examples are used to evaluate the robustness of machine/deep learning models before they effectively used. They are generated by little translations along the gradient direction and, in this way, they add small noise to the examples that are "invisible" to the human eye. However, giving them as input to the machine learning models makes them really confused and they usually generate a wrong output.

One important property of the adversarial examples is their transferability. In fact, an adversarial example created for a single model usually is adversarial also for others. This reveals that different machine learning models learn the same features and capture the same important characteristics of their training data distribution.

There exists a variety of algorithms to create and adversarial attacks and in this work we only see few that are based on gradient directions. The main idea is that, given an image as input, we modify each pixel moving along the gradient direction. Furthermore, we need to satisfy some restrictions in order not to go too far from the original input and to obtain an image that, to the human eye, is not different from the other one.

In order to define the constrained problem for adversarial attacks, we recall that in this paper we consider classifier models.

A classifier learns a function $f(x) : x \in \mathcal{X} \to y \in \mathcal{Y}$, where x is the input image and y is the label of the prediction and $\mathcal{X}$ and $\mathcal{Y}$ are their domain sets.

As an error measure, we define a loss function $\ell(y, y_{\text{true}}) : (y, y_{\text{true}}) \in \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, that usually, for multi-classification purpose, is known as *Categorical Cross Entropy*. The variable y is the predicted label and the variable y_{true} is the ground-truth target label of the classifier's input x.

The main purpose of the adversarial attack algorithms is to make the classifier's output as wrong as possible. Thus, the problem we have to optimize is defined as:

$$\begin{aligned} &\max \ell(x) \\ &\text{such that } \|x - x_{\text{ori}}\| \le \varepsilon \end{aligned} \tag{4}$$

where $\ell(x)$ is the loss function of the classifier, x_{ori} is the input image, x is the adversarial image and the condition is given by a norm $\|\cdot\|$ and a tolerance value ε. We keep in mind that this problem can be transformed into a minimization problem by taking the opposite of $\ell(x)$ as objective function.

The notation $\ell(x)$ can be confusing, as we are supposed to have the labels y and t_{true} as input. For simplicity, we write $\ell(x)$ instead of $\ell(y(x), y_{\text{true}}(x))$, referring to the classifier's loss function for the input x.

The constrain of the problem 4 ensures that the modifications we are applying are small enough, according to the value of ε. The more used norms are the $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

1.3. Targeted and untargeted attacks

As said before, the adversarial attacks' goal is to make the classifier's output wrong. We can decide whether we want to have control of the wrong output or we just care that the classifier mistakes. For these purposes, there are two types of adversarial methods: *untargeted* ones and *targeted* ones.

For the untargeted problems, we just refer to 4. It does not matter which is the output label of the adversarial example, as long as the classifier is wrong and the confidence of the output is high enough. We are satisfied only to maximize its loss function, according to the restriction on the adversarial image.

For the targeted algorithms, the problem is a little different. It is not sufficient to have a high loss value, we want to make the classifier predict a specific target y_{target}. So we want to maximize $\ell(y, y_{\text{true}})$ but also to minimize $\ell(y, y_{\text{target}})$. It has now become a min-max problem and we combine 4 with the following minimization problem:

$$\begin{aligned} &\min \ell(y, y_{\text{target}}) \\ &\text{such that } \|x - x_{\text{ori}}\| \le \varepsilon\,, \end{aligned}$$

in order to define a new constrained optimization problem for targeted adversarial attacks:

$$\begin{aligned} &\max \ell(y(x), y_{\text{true}}(x)) - \ell(y(x), y_{\text{target}}(x)) \\ &\text{such that } \|x - x_{\text{ori}}\| \le \varepsilon\,. \end{aligned} \tag{5}$$

2. Theoretical background

Adversarial studies are relatively recent, since the first study has been published in 2013. In general the definition of an attack depends on how much information about the model an adversary can access to and they can be divided into two categories: *white-box attacks* and *black-box attacks*.

In this project we focus on the fist type of attacks where the adversary have full access to the target model, and thus can compute efficiently the gradient. The optimization-based methods proposed for this setting are several and, as stated before, we will analyze and test 4 different methods: *FGSM, PGD, MI-FGSM, FW-White*.

The *FGSM* method works by linearizing the network loss function and ends in just one step. *PGD* and *MI-FGSM* are iterative methods that achieve better result than simple *FGSM* but they both generate adversarial examples near the boundary of the perturbation set. The more recent *FW-white* achieve both high success rates and good adversarial examples.

2.1. FGSM

The fast gradient sign method generates an adversarial example x with respect to the L_∞ norm following the update rule given by:

$$x = x_{ori} + \varepsilon \operatorname{sign}(\nabla_x \ell(x, y))$$

Differently from the other three algorithms, the *FGSM* is a one-step gradient-based approach. It is not iterative and update only once the input value x_{ori}.

This method linearizes the cost function around the input value. The idea of linearity is explored in Goodfellow [4].

The constraint of our problem, $\|x - x_{ori}\|_\infty < \varepsilon$ is certainly satisfied and it is easy to prove. Each element of the tensor corresponding to the input image x_{ori} is increased or decreased by exactly ε. This is due to the fact that the expression $\operatorname{sign}(\nabla_x \ell(x, y))$ can takes values only in $\{-1, 1\}$.

2.2. PGD

The Projected Gradient method is based on the fact that, considering only the direction of the gradient, without imposing any constraint, we might find a new value for x_{k+1} that is outside the convex set C:

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k) \text{ might be that: } x_{k+1} \notin C \ .$$

In order to overcome this issue, the *PG* method projects back the new point to the nearest point belonging to the set C. The procedure of the Algorithm 1 is shown in the table below:

Algorithm 1 Projected gradient general

```
1: for $k = 1 \ldots$ do
2:     Set $\bar{x}_k = \varrho_C(x_k + s_k \nabla f(x_k))$    ▷ with $s_k > 0$
3:     If $\bar{x}_k$ satisfies some specific condition, then STOP
4:     Set $x_{k+1} = x_k + \gamma_k(\bar{x}_k - x_k)$    ▷ with $\gamma_k \in (0, 1]$
5: end for
```

For consistency in this work, we consider the projection function $\varrho_C(\cdot)$ based on the infinite norm $\|\cdot\|_\infty$. In line 2 of the algorithm 1, the projection $\bar{x}_k$ of a given point x is the solution of the following minimization problem:

$$\min_{\bar{x}_k \in C} \|x - \bar{x}_k\|_\infty \ .$$

The projection function $\varrho_C(\cdot)$ ensures that the final output of the algorithm satisfies the constraint. At the end of each iteration, the new point x_{k+1} is certainly in the convex set. The line 4 can be re-written as:

$$x_{k+1} = x_k + \gamma_k(\bar{x}_k - x_k) = \gamma_k \bar{x}_k + (1 - \gamma_k) x_k \ .$$

Since γ_k is contained in $(0, 1]$ and both x_k and $\bar{x}_k$ are contained in the convex set C, the above combination is convex and this ensures that $x_{k+1} \in C$.

2.3. MI-FGSM

Here we present the *MI-FGSM* method that is a iterative version of the simple *FGSM* combined with momentum. The momentum methods is, in general, an acceleration technique for gradient based algorithms. It consists in memorizing the past gradient direction in order to avoid narrow valleys, poor local minima or maxima and other issues. Since the *FGSM* ends in one step under the assumption of linearity of the decision boundary [4] it may easily be stacked into poor local areas thus it is a good idea to combine it with the momentum method.

One additional advantage of momentum in this setting is that it gives a good trade-off between attack ability and the transferability of the adversarial example.

The algorithm scheme is summarized below for untargeted attacks.

Algorithm 2 MI-FGSM

```
Input: A real example $x$, ground-truth label $y$
Output: An adversarial example $x^*$
Require: Size of perturbation $\varepsilon$, number of iterations $T$,
    decay factor $\mu$
1: Fix $\gamma = \varepsilon/T$, $g_0 = 0$ and $x_0^*$
2: for $t = 0$ to $T - 1$ do
3:     Input $x_t$ and obtain the gradient $\nabla_x J(x_t, y)$
4:     Accumulate velocity
           $g_{t+1} = \beta \cdot g_t + \frac{\nabla_x J(x_t, y)}{\|\nabla_x J(x_t, y)\|_1}$
5:     Update
           $x_{t+1} = x_t + \gamma \cdot \operatorname{sign}(g_{t+1})$
6: end for
```

In line 1 γ is fixed in order to respect the constrain $\|x - x_{ori}\|_\infty \leq \varepsilon$ and the momentum term can be found in line 4 with a decay factor β. An adversarial example x_T^* is perturbed in the direction of g_t with a step size of γ. Moreover at each iteration the current gradient is normalized with its L_1 norm (line 4) in order to correct the gradients scale.

Observation 2.1. If $\beta = 0$ the *MI-FSGM* reduces to *FSGM*

2.4. FW-white

The Frank Wolf method is an iterative first-order optimization algorithm. It is wildly used in data science since it is a projection-free algorithm and has a smaller cost per iteration w.r.t. projection methods.

In this section we present the Frank Wolf method for white-box attacks proposed in [2]. The general idea of this method it to call a Linear Minimization Oracle (LMO): starting from a feasible solution, at each iteration we define a descent direction as the solution of:

$$\min_{x \in C} \nabla f(x_k)^T (x - x_k)$$

The previous is equivalent to the linear approximation of f in x_k:

$$\min_{x \in C} f(x_k) + \nabla f(x_k)^T (x - x_k)$$

By the Wirestrass theorem we have that a solution of such problem exists. In the context of *white-box attacks* the algorithms introduce an additional momentum $\mathbf{m}_t$ term.

The scheme is summarized below.

Algorithm 3 FW-White

```
Input: Number of iterations T, step sizes $\{\gamma_t\}$
Output: An adversarial example $x_T$
1: Set $x = x_0$, $m_{-1} = \nabla_x f(x_0)$
2: for $t = 0$ to $T - 1$ do
3:     $m_t = \beta \cdot m_{-1} + (1 - \beta) \cdot \nabla f(x_t)$
4:     $v_t = \text{argmin}(x, m_t)$
5:     $d_t = v_t - x_t$
6:     $x_{t+1} = x_t + \gamma_t d_t$
7: end for
```

Proposition 2.4.1. *The LMO defined in line 4 has a closed-form solution both for L_2 and L_∞.*

Proof. Let $\mathbf{h} = (x - x_{ori})/\varepsilon$. Than considering line 4 we have

$$\underset{\|x - x_{ori}\|_p \leq \varepsilon}{\text{argmin}} (x, m_t) = \underset{\|h\|_p \leq 1}{\text{argmin}} \, \varepsilon \cdot (h, m_t) = \underset{\|h\|_p \leq 1}{\text{argmax}} \, \varepsilon \cdot (h, -m_t)$$

By Holder's inequality (element wise), the maximum value is reached when:

$$|h_i| = c \cdot |(m_t)_i|^{\frac{1}{p-1}}$$

Since $\|h\|_p \leq 1$ we have:

$$h_i = -\frac{\text{sign}((m_t)_i) \cdot |(m_t)_i|^{\frac{1}{p-1}}}{(\sum_{i=1}^d |(m_t)_i|^{\frac{1}{p-1}})^{\frac{1}{p}}}$$

And,

$$x_i = \varepsilon \cdot -\frac{\text{sign}((m_t)_i) \cdot |(m_t)_i|^{\frac{1}{p-1}}}{(\sum_{i=1}^d |(m_t)_i|^{\frac{1}{p-1}})^{\frac{1}{p}}} + (x_{ori})_i$$

For $p = 2$ we have

$$v_t = -\frac{\varepsilon \cdot m_t}{\|m_t\|_2} + x_{ori}$$

For $p = \infty$ we have

$$v_t = -\varepsilon \cdot \text{sign}(m_t) + x_{ori}$$

□

Observation 2.2. For $T = 1$ Algorithm 3 reduces to the *FGSM*

2.4.1 Convergence Analysis for FW-White

We are in the setting of constrained optimization and in general the loss function for common DNN models are non-convex, thus the gradient norm of f is no longer a proper convergence criterion. We than adapt as a criterion the Frank-Wolfe gap:

$$g(x_t) = \max_{x \in C}(x - x_t, -\nabla f(x_t)) \tag{6}$$

There are 2 essential assumption to make in order to provide the convergence guarantee of the algorithm.

Assumption 2.4.1. Function $f(\cdot)$ is L-smooth with respect to x, i.e. for any x, x' it holds that:

$$f(x') \leq f(x) + \nabla f(x)^T (x' - x) + \frac{L}{2}\|x' - x\|_2^2$$

Assumption 2.4.2. Set C is bounded with diameter D, i.e. $\|x' - x\|_2 \leq D$ for all $x, x' \in C$

Lemma 2.4.1. *Under assumptions 2.4.2 and 2.4.1, for m_t in Algorithm 3, it holds that*

$$\|\nabla f(x_t) - m_t\|_2 \leq \frac{\gamma L D}{1 - \beta}$$

Proof. By definition of m_t in line 3 we have that the fist term is:

$$\begin{aligned}
&\|\nabla f(x_t) - m_t\|_2 = \\
&\quad = \|\nabla f(x_t) - \beta m_{t-1} - (1 - \beta)\nabla f(x_t)\|_2 \\
&\quad = \beta \cdot \|\nabla f(x_t) - m_{t-1}\|_2 \\
&\quad = \beta \cdot \|\nabla f(x_t) - \nabla f(x_{t_1}) + \nabla f(x_{t-1}) - m_{t-1}\|_2 \\
&\quad \leq \beta \cdot \|\nabla f(x_t) - \nabla f(x_{t_1})\|_2 + \beta \cdot \|\nabla f(x_{t_1}) - m_{t-1}\|_2 \\
&\quad \leq \beta L \|x_t - x_{t-1}\|_2 + \beta \cdot \|\nabla f(x_{t_1} - m_{t-1}\|_2
\end{aligned}$$

This holds exploiting the triangle inequality and assumption 2.4.1. Looking at line 6 and using assumption 2.4.2 we also have:

$$\begin{aligned}\|x_t - x_{t-1}\|_2 &= \gamma\|d_{t-1}\|_2 \\ &= \gamma\|v_{t-1} - x_{t-1}\|_2 \leq \gamma D\end{aligned}$$

Combining both these estimates yields:

$$\begin{aligned}&\|\nabla f(x_t) - m_t\|_2 \leq \\ &\quad\leq \gamma(\beta LD + \beta^2 LD + \cdots + \beta^t\|\nabla f(x_0) - m_0\|_2) \\ &\quad= \gamma(\beta LD + \beta^2 LD + \cdots + \beta^{t-1} LD) \\ &\quad\leq \frac{\gamma LD}{1-\beta}\end{aligned}$$

□

Theorem 2.4.1. *Under assumptions 2.4.2 and 2.4.1, let $\gamma_t = \gamma = \sqrt{2(f(x_0) - f(x^*))/(C_\beta LD^2 T)}$, the output of Algorithm 3 satisfies*

$$\tilde{g}_T \leq \sqrt{\frac{2C_\beta LD^2(f(x_0) - f(x^*))}{T}}$$

Therefore the Frank Wolf white-box attack algorithm achieves a $O(1/\sqrt{T})$ convergence rate.

Proof. By assumption 2.4.1, we have:

$$\begin{aligned}f(x_{t+1}) &\leq f(x_t)\nabla f(x_t)^T(x_{t+1} - x_t) + \frac{L}{2}\|x_{t+1} - x_t\|_2^2 \\ &= f(x_t)\gamma\nabla f(x_t)^T(v_t - x_t) + \frac{L\gamma^2}{2}\|v_t - x_t\|_2^2 \\ &\leq f(x_t)\gamma\nabla f(x_t)^T(v_t - x_t) + \frac{LD^2\gamma^2}{2} \\ &= f(x_t) + \gamma m_t^T(v_t - x_t) + \\ &+ \gamma(\nabla f(x_t) - m_t)^T(v_t - x_t) + \frac{LD^2\gamma^2}{2}\end{aligned}$$

Now we use the auxiliary quantity:

$$\tilde{v}_t = \underset{x \in C}{\operatorname{argmin}}(v, \nabla f(x_t)$$

The Frank Wolf gap (6) implies:

$$g(x_t) = -(\tilde{v}_t - x_t, \nabla f(x_t))$$

And on the other hand in line 5 we find:

$$v_t = \underset{v \in C}{\operatorname{argmin}}(v, m_t) \implies (v_t, m_t) \leq (\tilde{v}_t, m_t)$$

Combining both the previous equations:

$$\begin{aligned}&f(x_{t+1}) \leq f(x_t) + \gamma m_t^T(\tilde{v}_t - x_t) + \\ &\quad+ \gamma(\nabla f(x_t) - m_t)^T(v_t - x_t) + \frac{LD^2\gamma^2}{2} \\ &\quad= f(x_t) + \gamma\nabla f(x_t)^T(\tilde{v}_t - x_t) + \\ &\quad+ \gamma(\nabla f(x_t) - m_t)^T(v_t - \tilde{v}_t) + \frac{LD^2\gamma^2}{2} \\ &\quad= f(x_t) - \gamma g(x_t) + \gamma(\nabla f(x_t) - m_t)^T(v_t - \tilde{v}_t) + \frac{LD^2\gamma^2}{2} \\ &\quad\leq f(x_t) - \gamma g(x_t) + \gamma D\|\nabla f(x_t) - m_t\|_2 + \frac{LD^2\gamma^2}{2}\end{aligned}$$

The last inequality holds thanks to Cauchy-Schwarz inequality. Recalling the previous lemma we have:

$$\|\nabla f(x_t) - m_t\|_2 \leq \frac{\gamma LD}{1-\beta}$$

And substituting we obtain:

$$f(x_{t+1} \leq f(x_t) - \gamma g(x_t) + \frac{LD^2\gamma^2}{1-\beta} + \frac{LD^2\gamma^2}{2}$$

Considering all the iterations from $t = 0, \ldots, T-1$:

$$f(x_T) \leq f(x_0) - \sum_{t=0}^{T-1}\gamma g(x_t) + \frac{TLD^2\gamma^2}{1-\beta} + \frac{TLD^2\gamma^2}{2}$$

Isolating $g(\cdot)$:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} g(x_t) &\leq \frac{f(x_0) - f(x_T)}{T\gamma} + \gamma(\frac{LD^2\gamma}{1-\beta} + \frac{LD^2\gamma}{2}) \\ &\leq \frac{f(x_0) - f(x^*)}{T\gamma} + \gamma(\frac{LD^2\gamma}{1-\beta} + \frac{LD^2\gamma}{2})\end{aligned}$$

Where we consider that $f(x_T) \geq f(x^*)$
In conclusion:

$$\tilde{g}_T = \min_{1 \leq k \leq T} g(x_k) \leq \frac{f(x_0) - f(x^*)}{T\gamma} + \gamma(\frac{LD^2\gamma}{1-\beta} + \frac{LD^2\gamma}{2}$$

If we let $\gamma = \sqrt{2(f(x_0) - f(x^*))/(C_\beta LD^2 T)}$ where $C_\beta = (3-\beta)/(1-\beta)$ we have:

$$\tilde{g}_T \leq \sqrt{\frac{2C_\beta LD^2(f(x_0) - f(x^*))}{T}}$$

□

3. Test on ImageNet

Our goal is to compare the attacks of the 4 algorithms presented before. In order to do so we exploited the **ImageNet** dataset, in particular we use 100 different images belonging to the first 0 to 10 classes of database. We choose to run our **Python** code in the *Colab* platform with *runtime* set to *GPU*.

Moreover we decide to attack three different models:

1. **Inception V3**: this model is reported to have a 77.9% top-1 accuracy and a 93.7% top-5 accuracy [1].

2. **Inception ResNet-V2**: this model is reported to have a 80.3% top-1 accuracy and a 95.3% top-5 accuracy [1].

3. **MoblieNet V2**: this model is reported to have a 71.3% top-1 accuracy and a 90.1% top-5 accuracy [1].

3.1. Hyper-parameter selection

In order to be consistent we the related literature we choose to set the hyper-parameters following the baseline given in [2], i.e. we choose to set maximum distortion ε, step-size γ and proportion in momentum β as:

Parameters	PGD	FGSM	MI-FGSM	FW
ε	0.05	0.05	0.05	0.05
γ	0.03	-	0.03	0.1
β	-	-	0.9	0.9

Finally we fix maximum number of iteration at 20.

4. Experiments

We divide our experiment in two sub sections, one for targeted attack and one for untargeted attack. As stated before depending on the type of attacks the algorithms reported above need to be slightly modified in order to perform correctly.

For targeted attacks we need to minimize the objective function and we are forcing the model to return a given label y_{tar}.

For untargeted attacks we want to maximize the loss with respect to the current image and the correct label. Doing so the model will misclassify the image.

As a mean to compare the results we use the *success rate* (%) computed as the precentage of images misclassified by the models with any amount of confidence, the *average number of iterations* computed as a the average among the minimum number of iteration needed to output a misclassification by the model for each image and the *distortion* of the adversarial example x^*, computed as $\|x_{ori} - x^*\|_\infty$

4.1. Untargeted attacks

In the box below are summarized the success rates for all 4 algorithms in both withe box attacks and black box attack, i.e we generate the image on a model and attack the other.

Untargeted Attacks			
	Model	InceptionV3	InResNetV2
InceptionV3	PGD	*cose**	AFG
	FGSM	*cose**	ALA
	MI-FGSM	*cose**	ALB
	FW-white	*cose**	DZA
InResNetV2	PGD	-	*cose**
	FGSM	AX	*cose**
	MI-FGSM	AL	*cose**
	FW-white	DZ	*cose**

4.1.1 Inception V3 metrics

Attack	Success rate	AvgIt	Distortion
PGD	*cose**	AFG	
FGSM	*cose**	ALA	
MI-FGSM	*cose**	ALB	
FW-white	*cose**	DZA	

4.1.2 Inception Res V2 metrics

Attack	Success rate	AvgIt	Distortion
PGD	*cose**	AFG	
FGSM	*cose**	ALA	
MI-FGSM	*cose**	ALB	
FW-white	*cose**	DZA	

4.2. Targeted attacks

Here we report our success rate for targeted attack on both Inception V3 and Inception V4.

Targeted Attacks			
	Attack	InceptionV3	InResNetV2
InceptionV3	PGD	*cose**	AFG
	FGSM	*cose**	ALA
	MI-FGSM	*cose**	ALB
	FW-white	*cose**	DZA
InResNetV2	PGD	-	*cose**
	FGSM	AX	*cose**
	MI-FGSM	AL	*cose**
	FW-white	DZ	*cose**

4.2.1 Inception V3 metrics

Attack	Success rate	AvgIt	Distortion
PGD	*cose**	AFG	
FGSM	*cose**	ALA	
MI-FGSM	*cose**	ALB	
FW-white	*cose**	DZA	

4.2.2 Inception Res V2 metrics

Attack	Success rate	AvgIt	Distortion
PGD	$cose*$	AFG	
FGSM	$cose*$	ALA	
MI-FGSM	$cose*$	ALB	
FW-white	$cose*$	DZA	

5. Result and comparison

We can conclude that the expectation for *Inception V3* are met since our results are comparable with [2].

We can see empirically that the cost per iteration drops with the Frank Wolfe based algorithm as well as the amount of distortion.

The methods works best in *Inception V3* and also reach good results in *Inception ResNet V2*.

References

[1] Keras applications.

[2] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu, editors. *A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks*.

[3] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li, editors. *Boosting Adversarial Attacks with Momentum*.

[4] Ian J. Goodfellow, Jonathon Shelns, and Christian Szegedy, editors. *Explaining and Harnessing Adversarial Examples*.

[5] Francesco Rinaldi. *Optimization for Data Science Notes*. 2021.