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. 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 Goodfellow [**?**], 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 by misclassifying a single 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 directions and not the specific points in space. Moreover different models learn similar function when trained to perform the same task. We refer to this property as *transferability* of the adversarial example.

In this project, our goal is to implement four different algorithms for Adversarial Attacks. According to Rinaldi [**?**], Goodfellow [**?**], Dong [**?**] and Chen [**?**], we use the *Projected Gradient*, the *Fast Gradient Sign*, the *Momentum Iterative Fast Gradient Sign* 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:

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

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:

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

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:

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

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

One final 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 are effectively run. 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 adversarial examples as input to machine learning models fool them making their output wrong.

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 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 original 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:

$$\begin{aligned} &\max \ell(x) \\ &\text{subject to } \|x - x_{\text{ori}}\| \leq \varepsilon \end{aligned} \tag{4}$$

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 y_{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 constraint of the problem 4 ensures that the modifications we are applying are small enough, according to the value of ε. The most used norms are the $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$. In this work, we consider only the last one, that is defined as:

$$\|x\|_\infty = \max_{1 \leq i \leq d} |x_i| \quad \text{for } x \in \mathbb{R}^d \,.$$

1.3. Untargeted and targeted 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 formally 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:

$$\begin{aligned} &\min \ell(y, y_{\text{target}}) \\ &\text{subject to } \|x - x_{\text{ori}}\| \leq \varepsilon \,, \end{aligned} \tag{5}$$

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

$$\begin{aligned} &\max \ell(y(x), y_{\text{true}}(x)) - \ell(y(x), y_{\text{target}}(x)) \\ &\text{subject to } \|x - x_{\text{ori}}\| \leq \varepsilon \,. \end{aligned}$$

As done in Chen [**?**], we refer only to 5 for simplicity. Therefore, from now on, our aim is to find the optimal solution of the minimization problem when we are dealing with targeted attacks.

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 first 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, PGM, MI-FGSM, FW-White*.

The *FGSM* method works by linearizing the network loss function and ends in just one step. *PGM* and *MI-FGSM* are iterative methods that achieve better results 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

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

This method linearizes the cost function around the input value. It must to be applied a perturbation η to the original point x_{ori} and, in Goodfellow [**?**], they suggest to use:

$$\eta = \varepsilon \operatorname{sign}(\nabla_x \ell(x_{\text{ori}})) .$$

Since in this work we perform both untargeted and targeted attacks, we define the update rule for each of these problem.

$$\begin{aligned} x &= x_{\text{ori}} + \varepsilon \operatorname{sign}(\nabla_x \ell(x_{\text{ori}})) \quad \text{(untargeted)} ; \\ x &= x_{\text{ori}} - \varepsilon \operatorname{sign}(\nabla_x \ell(x_{\text{ori}})) \quad \text{(targeted)} . \end{aligned}$$

It can be noticed that what changes is only the sign before the perturbation. This is due to the fact that:

1. the function $\ell(x_{\text{ori}})$ stays the same for the two problems;

2. the untargeted attack is a maximization problem, thus we have to move along the direction of the gradient in order to find the maximum. For this reason there is the $+$ sign before the perturbation;

3. the targeted attack is a minimization problem, thus we move in the opposite direction of the gradient in order to find the minimum. For this reason there is the $-$ sign before the perturbation.

The constraint of our problem, $\|x - x_{\text{ori}}\|_\infty < \varepsilon$ 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 $\operatorname{sign}(\nabla_x \ell(x_{\text{ori}}))$ can take values only in $\{-1, 1\}$.

We can deduce that the adversarial images will be on the boundary of the convex set, since we always apply the largest possible perturbation.

2.2. PGM

The *Projected Gradient* method is iterative and gradient-based approach. It 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 *PGM* method projects back the new point to the nearest point belonging to the set C. The procedure of the algorithm is shown in the table **Algorithm 1**.

Algorithm 1 PGM

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

For consistency in this work, we consider the projection function $\varrho_C(\cdot)$ based on the infinity norm $\| \cdot \|_\infty$. In line 2 and 3 of the algorithm 1, the projection $\bar{x}_k$ over C is the solution of the following minimization problem:

$$\min_{x \in C} \|x - (x_k \pm s_k \nabla f(x_k))\|_\infty .$$

We define the argument of the projection $\varrho_C(\cdot)$ according to the type of the attack we are considering and, as done for the *FGSM*, we move along or in the opposite direction of the gradient, adding or subtracting the perturbation to x_k.

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 5 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 *Momentum Iterative Fast Sign* method that is a iterative version of the simple *FGSM* combined with momentum. The momentum method 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 [**?**], 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 in the **Algorithm 2**.

Algorithm 2 MI-FGSM

Input: A real example x, ground-truth label y
Output: An adversarial example x^*
Require: Size of perturbation ε, number of iterations T, step size γ, decay factor β

1: Fix $g_0 = 0$ and x_0^*
2: **for** $t = 0$ to $T - 1$ **do**
3: Input x_t and obtain the gradient $\nabla_x f(x_t)$
4: Accumulate velocity

$$g_{t+1} = \beta \cdot g_t + \frac{\nabla_x f(x_t)}{\|\nabla_x f(x_t)\|_1}$$

5: Update

$$x_{t+1} = x_t + \gamma \cdot \text{sign}(g_{t+1}) \quad \text{if untargeted attack}$$
$$x_{t+1} = x_t - \gamma \cdot \text{sign}(g_{t+1}) \quad \text{if targeted attack}$$

6: **end for**

In line 1 we initialize the gradient to zero and fix the starting point. The momentum term can be found in line 4 with a decay factor β and the current gradient is normalized with its L_1 norm in order to correct the gradients scale. As done in the previous methods, in line 5 we choose the update rule according to the type of attack we are considering. An adversarial example x_t^* is perturbed in the direction of g_t (or its opposite) with a step size of γ.

Observation 2.1. If $\beta = 0$, the *MI-FGSM* reduces to the iterative version of *FGSM*, defined as *I-FGSM* in literature.

2.4. FW-white

The *Frank Wolfe* method is an iterative first-order optimization algorithm. It is widely used in data science since it is a projection-free algorithm and has a smaller cost per iteration with respect to projection methods.

In this section, we present the Frank Wolfe method for *white-box* attacks proposed in Chen [**?**]. The general idea of this method is 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 $f(\cdot)$ in x_k:

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

By the Weierstrass 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 in the **Algorithm 3**.

Algorithm 3 FW-White

Input: Number of iterations T, step size γ
Output: An adversarial example x_T

1: Set $x_0 = x_{\text{ori}}$, $m_{-1} = -\nabla_x f(x_0)$ if untargeted attack, $m_{-1} = \nabla_x f(x_0)$ if targeted attack
2: **for** $t = 0$ to $T - 1$ **do**
3: $m_t = \beta \cdot m_{t-1} - (1 - \beta) \cdot \nabla f(x_t)$ ▷ if untargeted
4: $m_t = \beta \cdot m_{t-1} + (1 - \beta) \cdot \nabla f(x_t)$ ▷ if targeted
5: $v_t = \text{argmin}_{x \in C} \langle x, m_t \rangle = -\varepsilon \cdot \text{sign}(m_t) + x_{\text{ori}}$
6: $d_t = v_t - x_t$
7: $x_{t+1} = x_t + \gamma d_t$
8: **end for**

The difference between untargeted and targeted attacks in this case is a little different from the previous methods we have presented. This is because we want not to modify the closed-form solution of the LMO, defined in line 5. For this reason, instead of using two different update rules, we define the momentum term m_t in different ways.

In line 4, we propose what is done in Chen [**?**]; in line 3, since we have to move along the opposite direction of the movement for targeted attacks, we consider $-\nabla f(x_t)$ as the gradient of the cost function.

Thanks to the momentum term m_t, the LMO direction is stabilized and the convergence of the algorithm is found to be empirically accelerated.

In line 5, we exploit the closed-form solution for the LMO for the L_∞ norm case. The proof of this formulation is in the following Section 2.4.1 and here we just write the full update rule of line 7:

$$x_{t+1} = x_t - \gamma\, \varepsilon \cdot \text{sign}(m_t) - \gamma(x_t - x_{\text{ori}})\,.$$

The last term of the equation above enforces x_t to be close to x_{ori} for all of the iterations. We can suppose that the final adversarial image will have smaller distortion with respect to the other algorithms and this is the main advantage of the *FW* method with momentum.

2.4.1 LMO closed-form solution

Proposition 2.4.1. *The LMO defined in line 5 of Algorithm 3 has a closed-form solution both for L_2 and L_∞.*

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

$$\begin{aligned}\underset{\|x-x_{\text{ori}}\|_p \leq \varepsilon}{\operatorname{argmin}} \langle x, m_t \rangle &= \underset{\|h\|_p \leq 1}{\operatorname{argmin}} \varepsilon \cdot \langle h, m_t \rangle \\ &= \underset{\|h\|_p \leq 1}{\operatorname{argmax}} \varepsilon \cdot \langle h, -m_t \rangle \,.\end{aligned}$$

Recalling the Hölder's inequality, in our case we obtain that:

$$|\langle h, -m_t \rangle| \leq \|h\|_p \, \| - m_t\|_q \,, \quad \text{where } p + q = p\, q.$$

For a proper constant k, it is known that the maximum value is obtained if we require that:

$$h_i = k \operatorname{sign}(-m_t)|-(m_t)_i|^{q-1} \iff h_i = k\nabla_i(\|-m_t\|_q).$$

Deriving by component and knowing the condition on $q = p/(p-1)$, we have:

$$\begin{aligned}h_i &= \frac{\partial}{\partial (m_t)_i} \Big(\sum_{i=1}^{d} |(-m_t)_i|^q \Big)^{\frac{1}{q}} \\ &= \Big(\sum_{i=1}^{d} |(m_t)_i|^q \Big)^{(\frac{1}{q})-1} |(m_t)_i|^{q-1} \operatorname{sign}(-(m_t)_i) \\ &= \Big(\sum_{i=1}^{d} |(m_t)_i|^{\frac{p}{p-1}} \Big)^{-\frac{1}{p}} |(m_t)_i|^{\frac{1}{p-1}} \operatorname{sign}(-(m_t)_i) \,.\end{aligned}$$

In the end we find that:

$$h_i = -\frac{\operatorname{sign}((m_t)_i)|(m_t)_i|^{\frac{1}{p-1}}}{\Big(\sum_{i=1}^{d} |(m_t)_i|^{\frac{p}{p-1}} \Big)^{\frac{1}{p}}} \,.$$

From the definition of h, we can isolate x, which a single component is given by:

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

In our case, for $p = \infty$ we have

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

□

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

2.4.2 Convergence Analysis for FW-White

We are in a constrained optimization setting 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 then adapt as a criterion the Frank-Wolfe gap:

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

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

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

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

This condition is also equivalent to the following one:

$$\|\nabla f(x') - \nabla f(x)\|_2 \leq L \|x' - x\|_2 \,.$$

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 2.4.1 and 2.4.2, 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 4 we have that the first 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 7 and using the 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 and iterating the inequality on t, it yields to:

$$\begin{aligned}&\|\nabla f(x_t) - m_t\|_2 \leq \\ &\quad \leq \gamma(\beta L D + \beta^2 L D + \cdots + \beta^t \|\nabla f(x_0) - m_0\|_2) \,.\end{aligned}$$

By the initialization of the momentum in Algorithm 3, we have that $m_0 = \nabla f(x_0)$:

$$\|\nabla f(x_t) - m_t\|_2 \leq \gamma(\beta LD + \beta^2 LD + \cdots + \beta^{t-1} LD)$$
$$\leq \gamma LD \sum_{k=1}^{t-1} \beta^k \leq \gamma LD \sum_{k=0}^{\infty} \beta^k \leq \frac{\gamma LD}{1-\beta}\,.$$

□

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

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

where $\tilde{g}_T = \min_{1\leq k\leq T} g(x_k)$.

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

Proof. Recalling that $x_{t+1} = x_t + \gamma(v_t - x_t)$ and by assumption 2.4.1, we have:

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

Now we use the auxiliary quantity:

$$\tilde{v}_t = \operatorname*{argmin}_{x\in C} \langle v, \nabla f(x_t)\rangle = \operatorname*{argmax}_{x\in C} \langle v, -\nabla f(x_t)\rangle\,.$$

This is one optimal solution for the Frank Wolfe gap problem (6) and it implies:

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

And on the other hand in line 5 we find:

$$v_t = \operatorname*{argmin}_{v\in C} \langle v, m_t\rangle \implies \langle v_t, m_t\rangle \leq \langle \tilde{v}_t, m_t\rangle\,.$$

Combining both the previous inequalities:

$$f(x_{t+1}) \leq f(x_t) + \gamma m_t^T (\tilde{v}_t - x_t) +$$
$$+ \gamma(\nabla f(x_t) - m_t)^T (v_t - x_t) + \frac{LD^2\gamma^2}{2}$$
$$= f(x_t) + \gamma \nabla f(x_t)^T (\tilde{v}_t - x_t) +$$
$$+ \gamma(\nabla f(x_t) - m_t)^T (v_t - \tilde{v}_t) + \frac{LD^2\gamma^2}{2}$$
$$= f(x_t) - \gamma g(x_t) + \gamma(\nabla f(x_t) - m_t)^T (v_t - \tilde{v}_t) +$$
$$+ \frac{LD^2\gamma^2}{2}\,.$$

Thanks to the Cauchy-Schwarz inequality and the Assumption 2.4.2, we can write that:

$$f(x_{t+1}) \leq f(x_t) - \gamma g(x_t) + \gamma D\|\nabla f(x_t) - m_t\|_2 +$$
$$+ \frac{LD^2\gamma^2}{2}\,.$$

Recalling the previous Lemma 2.4.1 and substituting we have:

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

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 the sum of $g(\cdot)$:

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

where the last inequality is obtained by $f(x_T) \geq f(x^*)$, due to the optimality of x^* for the minimization problem.

The first term of the inequality is the average of $g(\cdot)$ among the iteration, so we can state that:

$$\tilde{g}_T = \min_{1\leq k\leq T} g(x_k) \leq \frac{1}{T}\sum_{t=0}^{T-1} g(x_t)\,.$$

In conclusion, we have that:

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

If we let $\gamma = \sqrt{2(f(x_0) - f(x^*))/(C_\beta LD^2 T)}$ and $C_\beta = (3-\beta)/(1-\beta)$, 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 how the 4 algorithms presented before behave in adversarial attacks. In order to do so, we exploit the **ImageNet** dataset, in particular we use 99 different images belonging to 10 different classes of dataset. 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 [**?**].

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

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

3.1. Hyper-parameter selection

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

Parameters	FGSM	PGM	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 the maximum number of iterations at 20. In Section 5, we will explore how ε and the number of maximum iterations affect the algorithms performance in terms of success rate and average distortion.

but we have also set two different stopping criteria: for untargeted attacks we stopped our algorithms when the label assigned to the distorted image from the network become different from the original one, while for targeted attacks we stopped our methods when the label assigned from the network to the distorted image is identical to the one wanted.

3.2. Technicalities

We have set two different stopping criteria. For untargeted attacks, we stop our algorithms when the label assigned to the distorted image from the network becomes different from the original one. While for targeted attacks, we stop our methods when the label assigned from the network to the distorted image is identical to the wanted one.

In order to retrieve the gradient from the models above, **TensorFlow** platform provides an important tool for machine learning called **GradientTape**. GradientTape is an object that you need to declare in your code which is able to record operation executed by TensorFlow. Its main methods, *watch* and *gradient*, are useful since they allow us to observe the processes going on inside the models and, from this information, perform the automatic differentiation with respect to some specific variable.

In our code, we use the function *watch*, that traces the image's tensor we input to the model, and the function *gradient* to obtain the gradient of the loss function with respect the input variable.

Another important fact we have to take into account is the pixels' range of the images. We preprocess the ImageNet figures in order to have the correct size for the models and to scale their pixels' interval to $[0, 1]$. In our algorithms, at each iteration we clip the elements of the image tensors in $[0, 1]$, just to make sure that we do not exit the range.

3.3. Projection for the infinity norm

In the papers we consider for this work, the most common used norm in the infinity one. Hence the projection function we discuss in Algorithm 1 for the *PGM* has to be done with respect to the infinite norm.

We recall that the projection of a point $\bar{x}$ over the set C for a given norm is defined as the optimal solution of the following problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{2}\|x - \bar{x}\|^2 \qquad \text{subject to } x \in C\,.$$

In the case of infinity norm, the condition can be considered equivalent to the following one:

$$\min_{x \in \mathbb{R}^d} \max_i |(x - \bar{x})_i| \qquad \text{subject to } x \in C\,.$$

Practically, as suggested in Dong [**?**], we can clip $\bar{x}$ according to the convex set C. Since in the algorithms our convex set is defined as $C = \{x \mid \|x - x_{\text{ori}}\|_\infty < \varepsilon\}$, at each iteration we clip the image tensor $\bar{x}$ in the ε vicinity of x_{ori}.

We can do this operation by exploiting the function **clip_by_value**, provided by the TensorFlow platform.

4. Experiments

We divide our experiment in two sub sections, one for untargeted attack and one for targeted 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 untargeted attacks we want to maximize the loss with respect to the current image and the correct label. Doing so, the model will hopefully misclassify the image.

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

As a mean to compare the results, we use three different attributes:

- the *success rate* $(\%)$: it is computed as the percentage of adversarial images that are able to successfully attack the model, with any amount of confidence. For the

untargeted attacks, it means that the model misclassifies the perturbed image, while for the targeted ones, the model outputs the target label;

- the *average number of iterations*: it is computed as a the average among the minimum number of iteration needed to output a misclassification by the model for each image. This mean is done without considering the success of the attacks (i.e., for an unsuccessful attack, we consider the maximum number of iterations);

- the *average distortion*: it is computed as the mean of $\|x_{\text{ori}} - x^*\|_\infty$ for all the adversarial examples x^*. This average is done only on the successful adversarial images, since we believe it makes no sense to consider the unsuccessful ones' perturbation.

4.1. Results

4.1.1 Inception V3 (untargeted)

Attack	Success rate	AvgIt	Distortion
FGSM	70.71 %	-	0.05
PGM	80.81 %	8.39	0.0166
MI-FGSM	95.96 %	2.01	0.0344
FW-white	92.03 %	2.68	0.0065

4.1.2 Inception ResNet V2 (untargeted)

Attack	Success rate	AvgIt	Distortion
FGSM	49.49 %	-	0.05
PGM	40.40 %	14.47	0.0162
MI-FGSM	96.97 %	2.48	0.0400
FW-white	92.93 %	4.16	0.0121

4.1.3 MobileNet V2 (untargeted)

Attack	Success rate	AvgIt	Distortion
FGSM	90.91 %	-	0.05
PGM	90.91 %	5.07	0.0200
MI-FGSM	97.98 %	1.46	0.0314
FW-white	95.96 %	1.98	0.0060

4.1.4 Inception V3 (targeted)

Attack	Success rate	AvgIt	Distortion
FGSM	0.00 %	-	-
PGM	87.88 %	7.76	0.0378
MI-FGSM	100.00 %	5.42	0.0500
FW-white	100.00 %	5.86	0.0215

4.1.5 Inception ResNet V2 (targeted)

Attack	Success rate	AvgIt	Distortion
FGSM	0.00 %	-	-
PGM	57.58 %	13.40	0.0411
MI-FGSM	100.00 %	8.66	0.0500
FW-white	98.99 %	8.00	0.0256

4.1.6 MobileNet V2 (targeted)

Attack	Success rate	AvgIt	Distortion
FGSM	0.00 %	-	-
PGM	98.99 %	4.25	0.0380
MI-FGSM	100.00 %	3.24	0.0500
FW-white	100.00 %	3.90	0.0159

In this section, we report tables showing the results of our experiments for both untargeted and targeted attacks (for targeted attacks we choose the *sea lion* class with index 150 in the ImageNet database).

Starting from the *FGSM* we can easily notice that it has no average iterations reported since it is a one step method. Anyway its performance on untargeted attacks is variable: the success rate on MobileNet V2 network is very satisfying while on Inception ResNet V2 it looks very disappointing since it does not even reach a 50 % success rate. It also has the highest possible distortion of all the three models, equal to $\varepsilon = 0.05$ which makes the difference between the original and the distorted image very noticeable with a naked eye.

For targeted attacks instead results are completely different: *FGSM* is completely unsuccessful and it does not fool any proposed model. The distortion value is not computed because no effective adversarial image has been produced. Since it is not an iterative method it is very difficult in only one step to retrieve an image with the wanted label.

Differently from *FGSM*, *PGM* is an iterative methods that performs better both on untargeted and targeted attacks, even though the success rate on ResNet V2 is lower than the other two models. The average number of iterations is the highest with respect to *MI-FGSM* and *FW-white*, perhaps because it does not make use of any acceleration technique.

The average distortion varies a lot between untargeted and targeted attacks. For the first type of attacks, the distortion is really low in all three model. While for the second type of attacks, it is higher, even though all type of attacks seem to produce images having a higher distortion. Probably points produced by *PGM* for targeted attacks are rarely in the convex set, thus, the projection operator outputs points that are near the boundary of the convex set

leading to a relatively large distortion.

The *MI-FGSM*, instead, is on average the fastest method for both targeted and untargeted attacks, thank to the momentum technique that guarantees an accelerated convergence towards the global minimum/maximum of the function. Its success rate is optimal for targeted attacks (100 %) and excellent for untargeted attacks where it does not go below 95 %. However, its fast convergence is compensated by a very high distortion, mainly for targeted attacks where it always equals to the epsilon set as input.

The *FW-white* represents a good balance between the average number of iterations and a low image distortion. In fact, the number of iterations is always greater or equal than *MI-FGSM* but less than *PGM*, while distortion is always the lowest among all methods, making the distorted image not even barely discernible from the original one. The success rate, instead, is usually a bit worse than the *MI-FGSM* (in untargeted attacks) but anyway excellent.

By taking a closer look to the tables, it is also possible to notice that ResNet V2 looks the most robust model: this is probably due to the massive usage of skip connections and non-linear activation functions, making adversarial attacks more difficult.

5. Comparison between the algorithms ***

We now want to compare the four algorithms when changing the distortion ε and the maximum number of iterations. In particular, we are interested in how the success and the distortion rates change.

The values of ε we choose to analyze are: $0.010, 0.042, 0.074, 0.107, 0.139, 0.171, 0.203, 0.236, 0.267, 0.300$. The tested number of iterations are: $2, 4, 6, 8, 10, 12, 14, 16, 18, 20$.

The figures from 1 to 6 are organized in four different graphs. The ones in the first row have the range of ε as x-axis, and the ones in the second row have the range of maximum iterations. In the left column, there is the success rate as y-axis and in the right one there is the distortion rate.

For all the graphs, we keep the same colour legend. The blue line is for the *FGSM*, the red one is for the *PGM*, the green line is for the *MI-FGSM* and the violet one is for the *FW-white*.

6. Conclusion

We have implemented four different algorithms for adversarial attacks, and our results are in line with the literature we have considered. The *FGSM* is the worst performing method since it is a non iterative method. On the other hand we can see that by adding a momentum term and some iteration, i.e. with *MI-FGSM* the adversarial attacks reach good performances. The main disadvantage of the *MI-FGSM* is that it achieves good success rates at the cost of high distortion. Moreover, as we can see in all the tables reported, the *PG* Method has the worst results in the *AvgIt* columns and this is reasonable since the method in question needs to perform a projection outside the convex set at each iteration.

In conclusion we can confirm that the overall best performances are given by the *FW-White* algorithm that it is able to reach an excellent success rate while maintaining a low level of image distortion and a decent number of iterations. We tested all the algorithms on three different pre-trained models, we can see that the majority of methods is able to fool neural networks successfully. However the model architecture attacked has a crucial impact on the performances, indeed the most robust model is *Inception ResNet V2* that has a peculiar architecture compared to the other two models.

Adversarial studies are in continuous development thus we can expect further analysis as well as higher performances and results in different networks in the near future.

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

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 both *Inception ResNet V2* and *MobileNet*. ****

A. Graphs for different ε and number of iteration

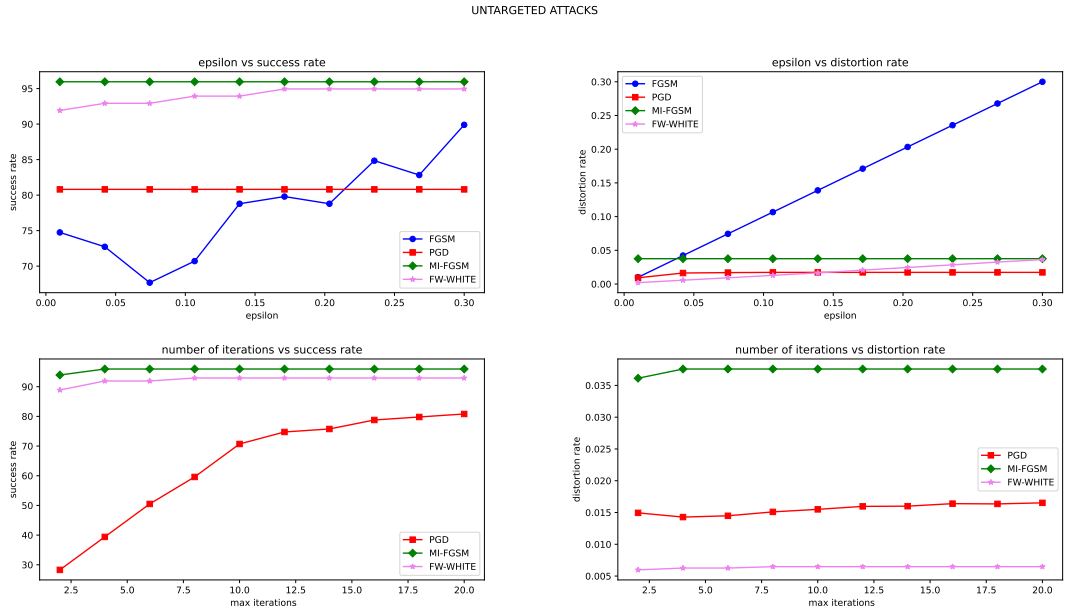

Figure 1. InceptionV3 - untargeted attack

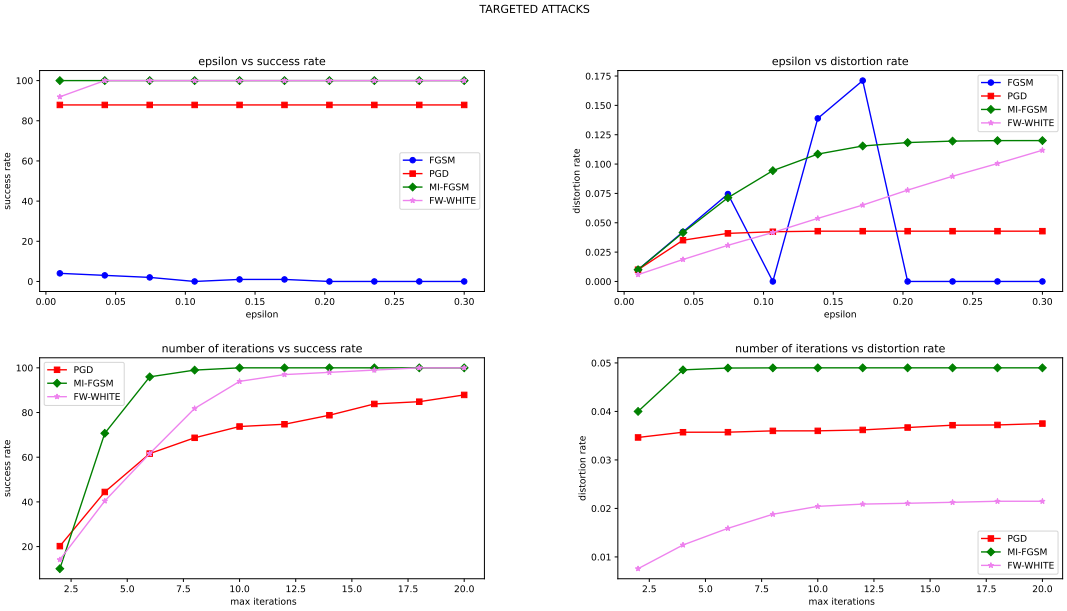

Figure 2. InceptionV3 - targeted attack

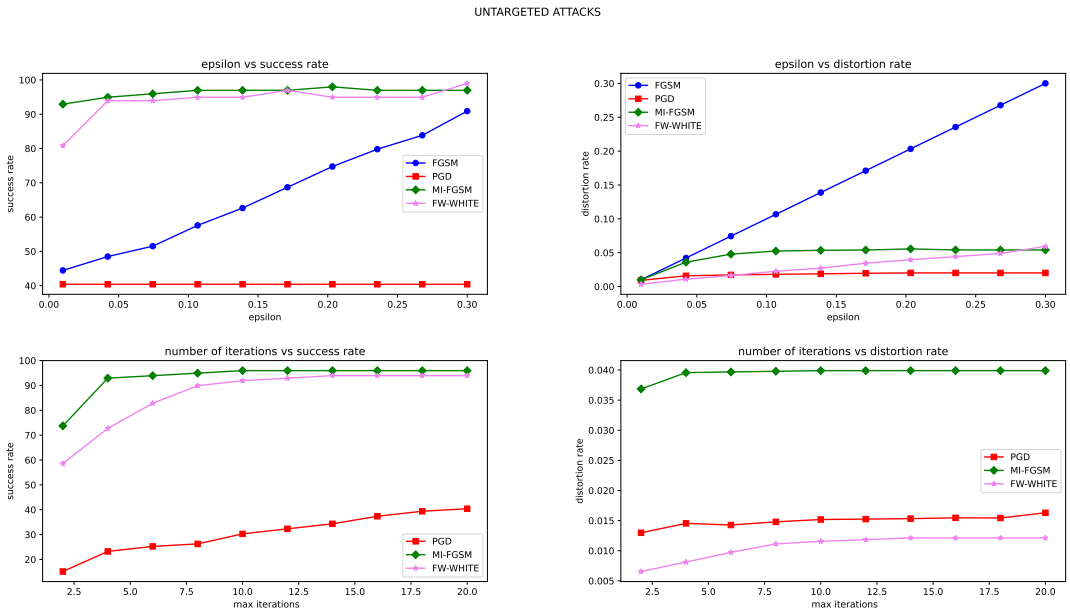

Figure 3. ResNet - untargeted attack

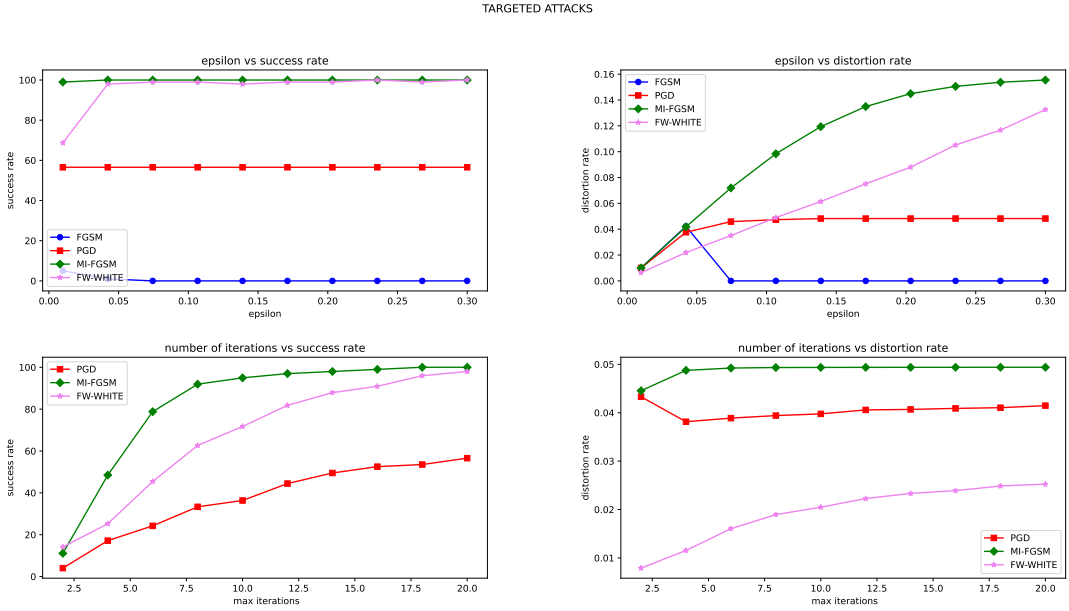

Figure 4. ResNet - targeted attack

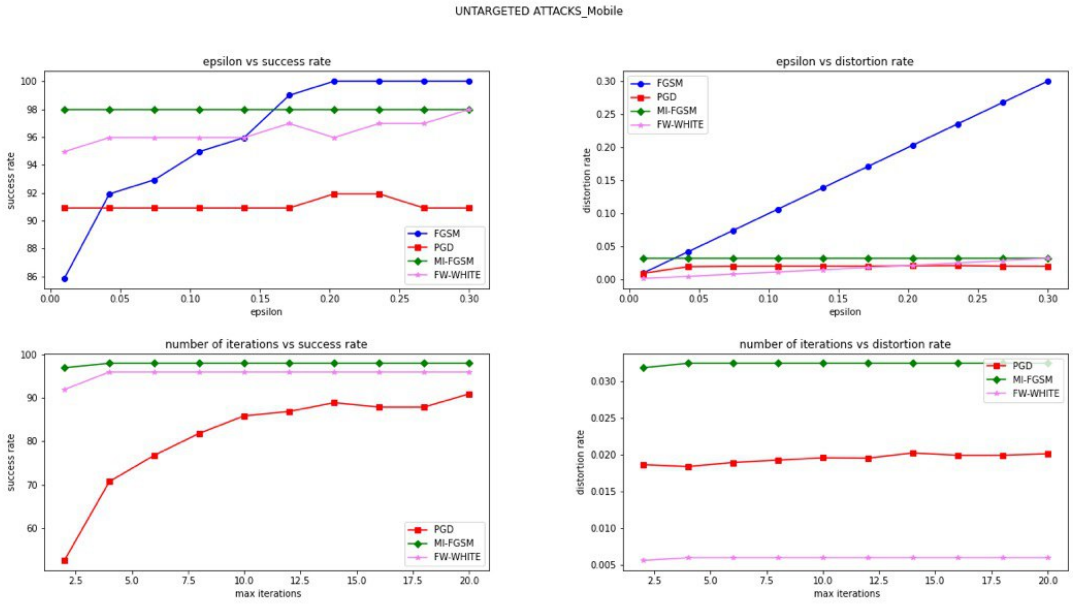

Figure 5. MobileNet - untargeted attack

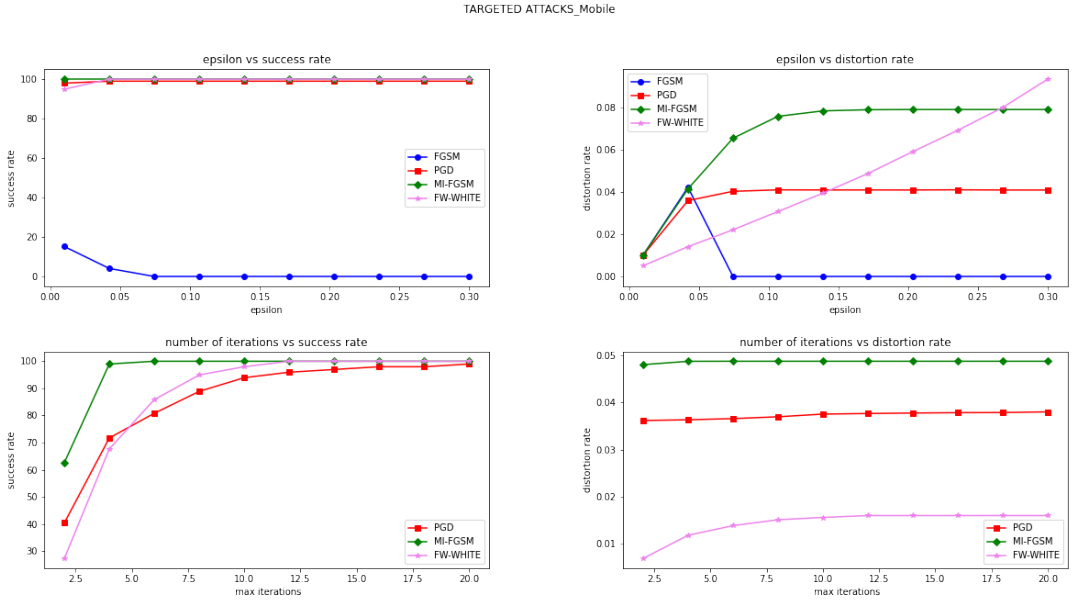

Figure 6. MobileNet - targeted attack