Optimization for Data Science Project: Zeroth Order Attacks

Jacopo Guidolin
jacopo.guidolin@studenti.unipd.it

Mattia Miolato
mattia.miolato@studenti.unipd.it

Filippo Santin
filippo.santin@studenti.unipd.it

Giovanni Vedana
giovanni.vedana.1@studenti.unipd.it

Abstract

In this paper we focused on the problem of adversarial attacks and we aimed to find an efficient way to solve this problem by means of several Zeroth Order Optimization methods. In particularly, we studied some Frank-Wolfe based methods and a derivative free method for structured optimization. After the comparison of the algorithms we applied these techniques trying to attacking the MNIST dataset.

1. Adversarial Attacks

The aim of Adversarial Attacks is to add a tiny amount of noise to a well classified signal in such a way that the perturbated signal will result misclassified with respect to an already existing particular classifier. In particular we focused on attacking images.

Starting with a satisfactory classifier, in our case a Deep Neural Network, we tried to find a suitable change on pixels, even an irrelevant one for human understanding, that could lead the net to a misclassification.

Based on the available information about the classifier Adversarial Attacks can be divided in two different categories: we refer to *white-box attacks* when we know the internal structure of the network, and thus we are able to compute the exact gradient through backpropagation. On the other hand we refer to *black-box attacks* when we only know the output of the networks for each provided input. Clearly the latter case is more complex since we are forced to compute an adequate approximation of the gradient.

Usually the dimension of the space we work with can be considerable and so we have to find efficient ways to estimate the gradient. Clearly, in high dimensional environments computing an approximation of the partial derivatives with respect to all the canonical directions is not feasible.

In addition to this the attacks can be further divided into two categories depending on the type of misclassification we aim to achieve.

An attack is considered *untargeted* if we try to mislead the classifier into changing the true target to any other class. On the other hand a *targeted attack* is an attack in which we want to perturbate the signal in such a way that the new image is classified as a particular class we have decided. The untargeted attack is obviously more flexible, and in what follows we will study only this case.

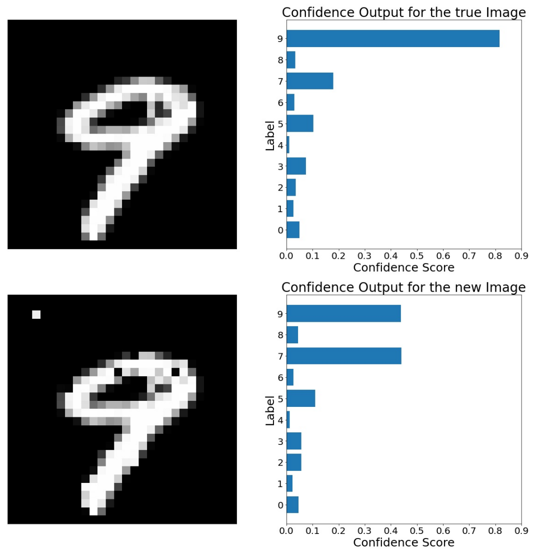

Figure 1: An Adversarial Attack on a digit image

2. Formulation of the problem

Given an image x_0 and the loss function $f(x)$ we can formulate the Adversarial Attack problem in the following way:

$$\min \ f(x_0 + x) \qquad (1)$$
$$\text{s.t} \ \|x\|_p \leq \varepsilon$$

where x is the perturbation, ε is the magnitude of the perturbation and p is an integer. In our experiments we will use the L^1 norm and an objective function that guarantees the misclassification of the instance once it reaches the value 0.

3. Frank-Wolfe based Attacks

The first approach we considered while trying to solve the problem of Adversarial Black-Box Attacks was Frank-Wolfe based methods. In particular we will concentrate on two main methods. We will refer to them with the names proposed in the original papers [1] and [2], respectively: *Deterministic Zeroth Order Frank-Wolfe* and *Frank-Wolfe Black-Box Attack*.

3.1. Gradient Approximations

In the aforementioned setting we also considered different gradient estimation techniques, in particular: *KWSA*, *RDSA* and *I-RDSA*. KWSA consists of an estimate on the gradient considering all the canonical directions. RDSA involves estimating the gradient along a single random direction at each iteration while I-RDSA involves estimating the directional derivatives along a number b of directions randomly sampled from an appropriate probability distribution.

Naturally computing the gradient with respect to all the canonical directions is the most query hungry scheme, while RDSA and I-RDSA are potentially inaccurate but less computationally demanding.

To put things in prospective, in our case, an approximation with respect to the canonical basis would have requested 784 different directions, but by sampling only 20 directions we have obtained satisfying improvements in terms of computational time without sacrificing any accuracy.

3.2. Deterministic Zeroth Order Frank-Wolfe [1]

We will begin by examining our first zeroth-order Frank-Wolfe algorithm.

In this context to estimate the gradient we will implement the *Kiefer-Wolfowitz* approximation (KWSA).

When using KWSA we can estimate the gradient as follows:

$$g(x_t) = \sum_{i=1}^{d} \frac{F(x_t + c_t e_i) - F(x_t)}{c_t} e_i.$$

Since we have that

$$g(x_t) = \nabla F(x_t) + \sum_{i=1}^{d} \frac{c_t}{2} \langle e_i, \nabla^2 F(x_t + \lambda_t c_t e_i) e_i \rangle e_i,$$

with $\lambda \in [0, 1]$, the linear optimization step becomes:

$$\begin{aligned}\langle v, g(x_t)\rangle =& \langle v, \nabla F(x_t)\rangle \\ &+ \frac{c_t}{2} \sum_{i=1}^{d} \langle e_i, \nabla^2 F(x_t + \lambda_t c_t e_i) e_i \rangle \langle v, e_i \rangle\end{aligned}$$

and this means that

$$\min_{v\in\mathcal{C}} \langle v, g(x_t)\rangle \le \min_{s\in\mathcal{C}} \langle s, \nabla F(x_t)\rangle + \frac{c_t L R d}{2}.$$

In this context the proposed algorithm is the following:

Algorithm 1:

```
1 Initialize $x_0 \in \mathcal{C}$;
2 for $t = 0, ..., T-1$ do
3     Compute $g(x_t) = \sum_{i=1}^{d} \frac{f(x_t + c_t e_i) - f(x_t)}{c_t} e_i$
        Compute $v_t = \text{argmin}_{s\in\mathcal{C}} \langle s, g(x_t)\rangle$;
4     Compute $x_{t+1} = (1-\gamma_t) x_t + \gamma_t v_t$
```

Finally for this proposed algorithm we have the following result:

Theorem 3.1 *Given the zeroth order Frank-Wolfe algorithm in Algorithm 1, we obtain the following bound:*

$$F(x_t) - F(x^*) = \frac{Q_{ns}}{t+2}$$

where $Q_{ns} = \max\{2(F(x_0) - F(x^*)), 4LR^2\}$.

3.3. Frank-Wolfe Black-Box Attack [2]

This algorithm includes also a momentum, that works as an accelerator, computed as the weighted sum of the gradients of the previous iterations, as follows:

$$m_t = \rho \cdot m_{t-1} + (1-\rho) \cdot g_t,$$

where $\rho \in (0, 1)$ is a parameter and g_t is the estimation of the gradient at iteration t.

This approach, in line with all of Frank-Wolfe methods, requires a solution to a Linear Minimization Oracle (LMO), represented by the following optimization problem:

$$v_t = \text{argmin}_{x\in X} \langle x, m_t\rangle$$

where m_t is the momentum at iteration t.

Fortunately there exists a closed form for the solution on the feasible set we have considered previously. Indeed, if we consider the L^1 norm a solution for the LMO is given by:

$$v_t = \varepsilon \cdot \text{sign}(-\nabla_{i_t} f(x_t)) \cdot e_{i_t}$$

with $i_k = \text{argmax}_i |\nabla_i f(x_k)|$.

Moreover, since we are considering black-box attacks, we cannot rely on backpropagation to find the exact gradient. Thus we have to find a numerical method that provides an adequate approximation of it.

The proposed algorithm is the following:

```
Algorithm 2:
1 $x_0 = x_{ori}, m_{-1}$ =GRAD_EST$(x_0, b, \delta)$;
2 for $t = 0, ..., T-1$ do
3     Compute $q_t$ =GRAD_EST$(x_t, b, \delta)$;
4     Compute $m_t = \beta \cdot m_{t-1} + (1-\beta) \cdot q_t$;
5     Compute $v_t$ = argmin$_{v \in \mathcal{X}} \langle v, m_t \rangle$;
6     Compute $d_t = v_t - x_t$;
7     Compute $x_{t+1} = x_t + \gamma_t d_t$
```

```
Algorithm 3: GRAD_EST
1 $q = 0$;
2 for $i = 1, ..., b$ do
3     Option 1: sample $u_i$ uniformly from the
        Euclidean unit sphere with $\|u_i\|_2 = 1$:
        $q = q + \frac{d}{2\delta b}(f(x+\delta u_i) - f(x-\delta u_i))u_i$
      Option 2: sample $u_i$ uniformly from the
        standard Gaussian distribution $N(0, I)$:
        $q = q + \frac{d}{2\delta b}(f(x+\delta u_i) - f(x-\delta u_i))u_i$
```

In particular as we can see in Algorithm 3 we provide two options which improve the gradient estimation.

We notice that now the approximation of the gradient is done only on a limited number of directions (given by the parameter b), making this method more effective than the Deterministic when the dimension becomes high.

3.3.1 Convergence Criterion

To prove the convergence of the method described above we use the Frank-Wolfe gap as criterion, given by

$$g(x_t) = \max_{x \in X} \langle x - x_t, -\nabla f(x_t) \rangle$$

that can be considered as a good convergence criterion since a point x_t is stationary for our optimization problem if and only if $g(x_t) = 0$.

3.3.2 Convergence for Frank-Wolfe Black-box Attacks

We can prove the convergence of the previous method under three assumptions:

1. Function $f(\cdot)$ is L-smooth with respect to x, that means that for any x, x' it holds that

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

2. Set X is bounded with diameter D

3. Gradient of $f(\cdot)$ at zero point $\nabla f(0)$ satisfies $\|\nabla f(0)\|_2 \leq G$

Under the previous assumptions, it can be proved that (for suitable choices of the parameters γ_t, b and δ) for the Frank-Wolfe Black-box Attack algorithm we have that $\mathbb{E}[\tilde{g}_T]$ is less or equal than

$$\frac{D}{\sqrt{T}}\left(2\sqrt{C_\beta L(f(x_0) - f(x^*))} + C_\beta(L + G + LD)\right)$$

where $\tilde{g}_T = min_{1 \leq k \leq T} g(x_k)$ and $C_\beta = (3-\beta)/(1-\beta)$.

The previous result suggests that for this Frank-Wolfe based method the convergence rate is $O(1/\sqrt{(T)})$.

4. Optimize, Refine and Drop Algorithm [3]

We start by considering

$$\min_{x \in \mathcal{H}} f(x) \tag{2}$$

where $\mathcal{H}$ is the convex hull of a finite set of points $\mathcal{A} = \{a_1, ..., a_m\} \subset \mathbb{R}^d$ called atoms and $f : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function.

Any point $x \in \mathcal{H}$ can be written as a convex combination of the atoms, so (2) can be reformulated in this way:

$$\min_{\omega \in \Delta_{m-1}} f(Aw) \tag{3}$$

where $A = [a_1, ..., a_m] \in \mathbb{R}^{d \times m}$ and $\Delta_{m-1} = \{w \in \mathbb{R}^m : \sum_{i=1}^m w_i = 1, w_i \geq 0\}$ where w_i is the weight of the ith atom in the convex combination.

The problem (3) can be faced with any linearly constrained derivative-free optimization (DFO) algorithm i.e. direct search schemes or model-based approaches, like LINCOA or NEWUOA. Anyway, these methods do not take advantage of the specific structure of (3) since they could might grid to an halt if the problem dimension increases. Secondly, another way to deal with the original problem is by generating the facet-inducing halfspaces that describe the feasible set $\mathcal{H}$. However, there exists a number of situations where using this technique is not a practicable option.

Moreover, we are focusing on instances whose solutions can be obtained as a proper convex combination of just a few atoms in the set. Thus we are in the case $m \gg d$ or also in the case of polytopes with $\mathcal{O}(d)$ vertices and a particular structure i.e. L_1 ball . Therefore, we want to find a new method that takes advantage of this *vertex description*.

4.1. Derivative Free Simplex

This new method needs an inner solver to minimize the objective function over a subset of atoms. This tailored approach is called **df-simplex** and can be used for problems of the form:

$$\min_{y \in \Delta_{\bar{m}-1}} \varphi(y)$$

where $\varphi : \mathbb{R}^{\bar{m}} \to \mathbb{R}$ is a continuously differentiable function.

df-simplex combines the use of a set of sparse directions containing positive generators of the tangent cone with a specific line search. This method proceeds by selecting a feasible point $y^0 \in \Delta_{\bar{m}-1}$ and some stepsizes $\hat{\alpha}_i^0, i = 1, ..., \bar{m}$. Then it chooses a variable index j_k such that $y_{j_k}^k$ is positive and establishes the directions $d_i^k = \pm(e_i - e_{j_k})$ for all indices $i \neq j_k$, with e_i being the ith vector of the canonical basis. Shuffling these search directions used at each iteration k can improve performances.

Thanks to an additional algorithm, called *Line Search Procedure*, **df-simplex** is able to produce a line search to achieve a good reduction in the objective function and it constantly refreshes the stepsizes we implemented initially.

4.1.1 Stopping condition

Given $\epsilon > 0$, the formula $\hat{\alpha}_i^{k+1} = \max\{\theta\hat{\alpha}_i^k, \epsilon\}$ is used to upgrade each stepsize. The method concludes the algorithm, in a finite number of iterations, at the first iteration k in which

$$\hat{\alpha}_i^k = \epsilon \quad \forall i \in 1, ...\bar{m} \quad and \quad \alpha_i^k = 0 \quad \forall i \neq j_k \tag{4}$$

4.1.2 Additional result

We will now show a bound on the stationary error for the solution obtained by **df-simplex**.

Theorem 4.1 *Assume that $\nabla\varphi$ is Lipschitz continuous with constant L and the stopping condition (4) is used with a given tolerance $\epsilon > 0$. Then the solution $\bar{y}$ returned by df-simplex is such that*

$$\max_{y \in \Delta_{\bar{m}-1}} -\nabla\varphi(\bar{y})^T(y - \bar{y}) \leq C\epsilon$$

where $C = 2\sqrt{2}(\bar{m} - 1)(2L + \gamma)$.

4.2. ORD

When the number of atoms rises, it is not always possible to achieve a reasonable solution. Therefore we will now see how **df-simplex** can be enhanced.

The new scheme consists of three phases. The issue of the dimension is dealt with the research of an improvement by including and removing atoms, according to some rules. This new method is called **ORD**, as in $optimize$, $refine$ and *drop*. The only assumptions that have to be satisfied are smoothness of the objective function and the use of zeroth order information to approximately minimize the reduced problem and to select a new atom.

- $Optimize$: we take A^k, a matrix with columns containing atoms of $\mathcal{A}^k$, a subset of $\mathcal{A}$. We use **df-simplex**, as we have told before, in the problem:

$$\min_{y \in \Delta_{|\mathcal{A}^k|-1}} \varphi^k(y)$$

  where $\varphi^k(y) = f(A^k y)$.

- $Refine$: we randomly choose an atom $a_{i_k} \in \mathcal{A} \setminus \mathcal{A}^k$ so that it guarantees an improvement of the objective function.

- *Drop*: we use the set $\mathcal{D}^k \subseteq \{a \in \mathcal{A}^k$ such that $a = A^k e_h$ and $\bar{y}_h^k = 0\}$, containing elements (atoms) to be removed to keep the dimension of the reduced problem constantly small enough.

4.2.1 Theoretical analysis

We affirm the main convergence result.

Theorem 4.2 *Let x^k be a sequence of points produced by ORD. At least one limit point $\lim_{k\to\infty} x^k = x^*$ is stationary for (2).*

4.2.2 Properties of ORD

- from a certain iteration onwards the set A^k does not contain atoms that are not needed to express the optimal solution.

- there is a further small improvement associated with the third drop phase thanks to the compute of gradient estimates. Indeed by redefining, at every iteration k, the set $\mathcal{D}^k = \{a \in \mathcal{A}^k$ such that $a = A^k e_h$, $\bar{y}_h^k = 0$ and $(g^k)^T(e_k - \bar{y}^k) \geq 0\}$ we can get a correct and more accurate elimination of useless atoms. Let's specify that g^k is precisely an approximation of $\nabla\varphi^k(\bar{y}^k)$ satisfying $\left\|\nabla\varphi^k(\bar{y}^k) - g^k\right\|$ with $\{r^k\}$ a sequence of positive scalars converging to 0.

5. Experiments

In order to verify the performances of the algorithms, we applied our methods in the particular context of Adversarial Attacks. We tried to attack the MNIST Datset, containing 60000 images of hand-written digits of dimension 28×28. To begin with we trained a simple Neural Network on 10000 samples and with all the algorithms we have tried to fool this model. In this process we chose a suitable loss function as described in [3]:

$$f(x) = \max\{\log[F(x)_{t_0}] - \max_{i:i\neq t_0}\{\log[F(x)_i]\}, 0\} \tag{5}$$

where $F(x)$ is the output of the network, that represents the confidence score that the input belongs to a particular class, and t_0 is the true target. It is noticeable that this loss function is strictly positive when the image is still correctly classified, while it is equal to 0 if and only if it is misclassified.

In our setting we tried to solve the optimization problem in (1), where x_0 is the starting image and the goal of the algorithm is to produce a suitable noise x, such that the new image $x_0 + x$ will be misclassified by the network. The space is endowed with the norm L_1, since this choice guarantees a sparse solution.

In order to evaluate the performances of the methods we sampled 100 correctly classified images from the test set and we tried to attack these images with every algorithm.

Moreover, to compare the methods we chose three different criteria:

- *Attack rate* : the rate of successful attacks. In our opinion this is the main comparison measure.
- *CPU time* : the mean CPU time over all successful attacks.
- *Distortion* : the average L_1 norm of the noise over all successful attacks.

For experimentation purposes in addition to the algorithms in paragraph 3 and 4 we also tried to implement the so-called *Algorithm 2* presented in [1]. This algorithm is designed to work in a stochastic optimization environment, but we tried to implement it anyways since for this algorithm the best parameters are provided in [1].

We implemented the Deterministic Zeroth Order Frank-Wolfe algorithm only with KWSA gradient approximation since with the other methods we were not able to find a suitable choice of parameters that led to convergence.

We start by showing the performance measure of the following algorithms:

- **RDSA (a)**: Frank Wolfe Black-box Attacks with RDSA gradient estimation as in **Algorithm 2** (sampling directions from $N(0, I)$)
- **RDSA (b)**: Stochastic Gradient Free Frank Wolfe with RDSA gradient estimation as implemented in [1]
- **I-RDSA (a)** : Frank Wolfe Black-box Attacks with I-RDSA gradient estimation as in **Algorithm 2** (sampling directions from $N(0, I)$)
- **I-RDSA (b)** : Stochastic Gradient Free Frank Wolfe with I-RDSA gradient estimation as implemented in [1]
- **DET** : Deterministic Zeroth Order Frank Wolfe with KWSA
- **ORD** : Optimize, Refine and Drop

	Attack Rate	CPU Time (s)	Distortion
RDSA (a)	94 %	0.52	8.74
RDSA (b)	93 %	0.74	8.92
I-RDSA (a)	93 %	1.46	8.46
I-RDSA (b)	94 %	0.29	8.84
DET	90 %	5.94	7.73
ORD	98 %	1.03	9.93

Table 1: Adversarial Attack Performance comparison

The previous results were found by selecting a maximal distortion $\varepsilon = 50$ for all the experiments and by choosing for each method the optimal parameters in terms of time and performances. We seeked for the best parameters with a grid search and we selected the values that assured the best fit.

For the algorithms that require a particular number of directions to estimate the gradient we decided to use 20 directions: in the figure 2 we reported the results altering the number of directions for the I-RDSA algorithms.

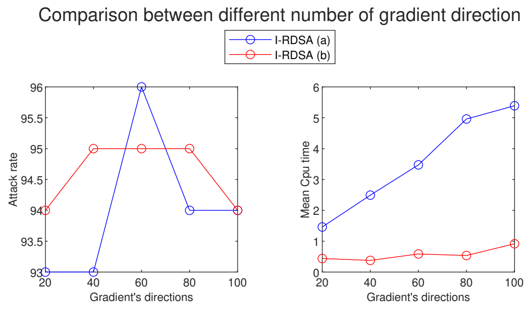

Figure 2

For each algorithm we considered a stopping condition that assures that the method stops once the instance is misclassified: in particular our methods stop once the loss function in (5) reaches the value 0, since in that case the probability that the image belongs to the right class is lower than the probability of belonging to a incorrect class.

We selected a considerable number of iterations: this choice guarantees that if an instance is not misclassified after many epochs probably the instance is unattackable by the method in question.

For the ORD algorithm we chose to take into account all the atoms as starting atoms since this choice gave better results. For what concerns the value of the stopping conditions in (4.1.1) we chose $\epsilon = 10^{-8}$.

Since our images assumes values in the interval $[0, 1]$ we applied the following transformation at each iteration of the methods, as explained in [4]:

$$x_i = (1 + tanh\zeta_i)/2,$$

where ζ_i is the output image.

In order to exploit this transformation we previously processed the images by applying to them the inverse transformation.

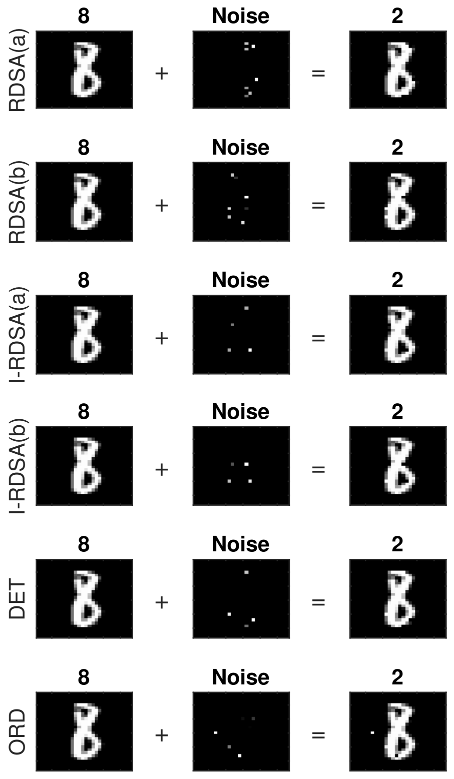

Figure 3: All algorithms mislead the classifier into the same new class.

6. Conclusions

Once we obtained the results of the experiments we compared Frank Wolfe Black-box Attack, Deterministic Zeroth Order Frank-Wolfe and the ORD algorithm.

We first noticed that the Frank Wolfe Black-box Attack algorithm with RDSA and I-RDSA give similar results in terms of attacks rate, but the RDSA approximation is faster. The other RDSA and I-RDSA approaches (Stochastic Gradient Free Frank Wolfe) give comparable results in terms of attack rate but they converge in a smaller amount of time: this behavior could be explained by the particular choice of step size in the update rule and in the approximation of the gradient. This is a surprising result since this algorithm gives satisfying performances in terms of CPU time despite being designed to work in a stochastic optimization setting.

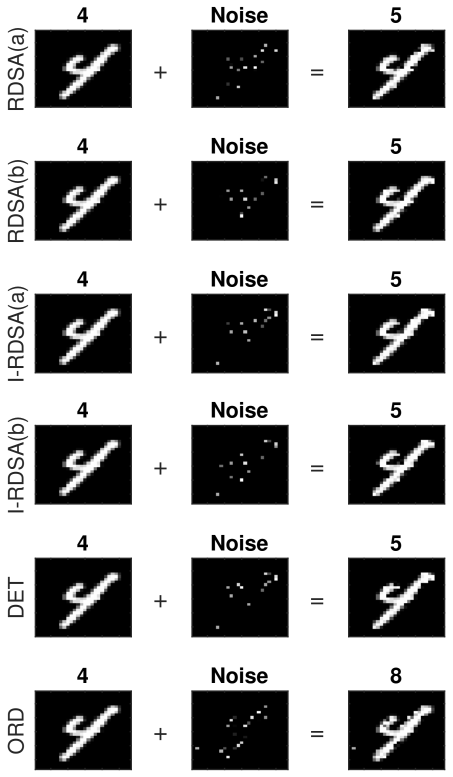

Figure 4: With this instance, we want to show how the different framework implemented by ORD leads to a different misclassification.

The Deterministic algorithm with KWSA approximation turned out to be the worst of the methods, since it does not give uppermost results neither in terms of attack rate, nor for what concerns CPU time. These objectionable performances can be explained by the large computational cost of the estimation of the gradient at each iteration, due to the high dimensional setting. This computation makes the iterations of this algorithm way more expensive than the others.

Finally the ORD algorithm is the one that gives the highest attack rate (98 %) , and it is able to reach this result in a satisfying amount of time.

In terms of distortions all the algorithms are comparable and the modified images are similar (see Figure 3). We noticed that our methods tend to change pixels near the center of the image. Border pixels are rarely changed. This behavior could be explained by the inferior importance of border pixels when considering digit recognition.

We noticed that the Frank-Wolfe methods tend to misclassify the images to the same new class, while ORD (since it solves a different optimization problem) tends to lead to different misclassifications (see Figure 4).

References

[1] Anit Kumar Sahu, Manzil Zaheer, and Soummya Kar. *Towards Gradient Free and Projection Free Stochastic Optimization*. 2019. arXiv: `1810.03233 [math.OC]`.

[2] Jinghui Chen et al. *A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks*. 2019. arXiv: `1811.10828 [cs.LG]`.

[3] Andrea Cristofari and Francesco Rinaldi. *A derivative-free method for structured optimization problems*. 2021. arXiv: `2005.05224 [math.OC]`.

[4] Nicholas Carlini and David Wagner. *Towards Evaluating the Robustness of Neural Networks*. 2017. arXiv: `1608.04644 [cs.CR]`.