Frank-Wolfe and Projected Gradient Methods for Adversarial Attacks

Manuel D'Alterio Grazioli (2104370), Angelo Guarino (2130763), Laura Martini (2124642)

1 Abstract

The project explores adversarial attacks on neural networks from the perspective of constrained optimization concentrating on white-box scenarios. We will compare the performance of several variants of the projection-free Frank-Wolfe algorithm, including standard, momentum-based, away-step, pairwise versions, and projection-based Projected Gradient Descent algorithm, comprising also the version with momentum. The algorithms will be tested on MNIST and ImageNet datasets under L_2 and L_∞ norm constraints. Our experiments started by finding the most appropriate range ε where attacks were most effective, then proceeded with a more exhaustive set of experiments, where each algorithm was evaluated under every combination of settings, to get a comprehensive analysis of its effectiveness and optimization behavior.

2 Introduction

Adversarial Attacks are deceptive techniques used to manipulate input data with the purpose of deceiving machine learning models into making incorrect predictions or classifications. They exploit vulnerabilities in models, especially deep neural networks, by injecting suitably crafted, often imperceptible, perturbations into input instances. Although visually imperceptible to humans from the original data, the perturbations can greatly alter the model output, leading to incorrect predictions with high confidence. Based on the information and knowledge about the target function that has an adversary, we can categorize them:

- **Black-box attack**: Operates under the assumption that the attacker has no access to the internal structure of the model. Only input-output pairs are available, and the attacker must estimate gradient information, typically requiring a large number of queries, especially in high-dimensional data settings.
- **White-box attack**: Assumes full access to the model's architecture, parameters, and gradients. This knowledge enables the attacker to craft highly effective perturbations by directly leveraging the model's internal information.

Since our project aims to compare and test constrained optimization methods on adversarial attacks, we will focus exclusively on white-box cases that require full access to the model for effective optimization.

3 Methology

3.1 Problem Formulation

For the untargeted attack case we have the following constrained optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x_0 + x) \\ \text{s.t.} \quad & \|x\|_p \leq \varepsilon \end{aligned} \tag{1}$$

where f is a suitably chosen attack loss function, x_0 is a correctly classified data point, x represents the additive noise/perturbation, $\epsilon > 0$ denotes the perturbation budget of the attack. In detail, let's define $\ell(x, \hat{y})$ as the classification loss function of the specific model with an input of $x \in \mathbb{R}^n$, where n is the number of elements in the vector x, and a corresponding label y. While for targeted attacks, we want to maximize $-\ell(x, \hat{y})$ to learn an adversarial examples that will be missclassified to the target class $\hat{y}$. We define the constraint set $X := \{x \mid \|x\|_p \leq \varepsilon\}$ contains all perturbation vectors x such that ℓ_p norm is at most ε. Hence this is the set of allowed perturbations in the adversarial setting. When $p = 1$ or ∞ the constraint set can be characterized as a polytope. For $p = 1$ the perturbation is sparse, meaning only a few components are non-zero, while for $p = \infty$ perturbation is uniformly bounded across all components. In such cases, the feasible region X admits a polyhedral structure and can be described as the convex combination of a finite set of extreme points $X = \text{Conv}(\mathcal{A})$.

3.2 PGD vs Frank-Wolfe Overview

Projected Gradient Descent (PGD) updates converge quickly, but they must project back to the constraint set at each iteration in order to maintain a valid solution. Projection is an operation that tend to drive adversarial examples onto the boundary of the constraint set, which can introduce further distortion. To avoid that, we can use Frank-Wolfe algorithm, which is projection-free unlike PGD. Instead it uses a Linear Minimization Oracle (LMO) to find the direction that minimizes the linear approximation of the loss function over the constraint set X at each iteration:

$$\text{LMO} \in \arg\min_{x \in \mathcal{X}} \langle x, \nabla f(x_t) \rangle$$

By calling LMO, Frank-Wolfe solves a linear problem over the constraint set and moves towards the solution using a weighted average with previous iterate to update the final formula. From the comparative overview presented: PGD is invasive: it moves in the gradient direction, potentially going out from the constraint set, then projects back, which can cause drastic changes or distortions. While Frank-Wolfe is conservative: it never leaves the constraint set and never projects, which can bring

to reduced distortion adversarial examples. In the following sections, we implement, test and compare PGD and Frank-Wolfe optimization approaches in an adversarial setting.

4 Algorithms

4.1 Vanilla Frank-Wolfe for Adversarial Attacks

Algorithm 1 Vanilla Frank-Wolfe Algorithm

input: Number of iterations T, step sizes $\{\gamma_t\}$
$x_0 = x_{\text{ori}}$
for $t = 0, \ldots, T-1$ **do**
    $v_t = \arg\min_{x \in \mathcal{X}} \langle x, (\nabla f(x_t)) \rangle$ ▷ LMO
    $d_t = v_t - x_t$
    $x_{t+1} = x_t + \gamma_t d_t$
end for
output: x_T

The following Frank-Wolfe based white-box attack algorithm is shown in Algorithm 1, which is built upon the standard Frank-Wolfe algorithm. Finding the LMO solution can also be computationally expensive to obtain, but for our constraint set $X := \{x \mid \|x\|_p \leq \varepsilon\}$, the matching LMO has a closed-form solution; for L_∞ norm scenario:

$$v_t = x_{\text{ori}} + \epsilon \cdot \text{sign}(\nabla f(x_t))$$

The update formula for each iteration is as follows:

$$x_{t+1} = x_t - \gamma_t \epsilon \,\text{sign}(\nabla f(x_t)) + \gamma_t (x_t - x_{\text{ori}})$$

with the term $\gamma_t(x_t - x_{\text{ori}})$ that imposes x_t to be near to x_{ori} at each iteration $t = 1, ..., T$ enforcing adversarial sample to have small perturbation. When the optimal point $x*$ lies in the interior of the constraint set X, the LMO picks a steady vertex s direction at each iterate allowing the algorithm to reach quickly $x*$ resulting in a linear convergence rate under some assumptions such as strong convexity and smoothness of the objective function. Nevertheless, in many adversarial issues, the optimal point $x*$ is located on a face of the constraint set X, and at each iterates, the algorithm start to zig- zag between the vertices chosen by LMO to define the face containing $s*$. These oscillations cause a sublinear convergence rate when $x*$ is on the boundary. Zig-zagging is a well-known vanilla Frank-Wolfe algorithm scenario when the optimal point lies on the face of the constraint set X, because the feasible directions chosen by LMO "jump" between vertices rather than aligning steadily towards the solution.

In the next subsections we will discuss some variants of the standard Frank-Wolfe like FW with momentum, Away steps FW and Pairwise FW that are aimed to overcome this problem and recover linear rates even when $x*$ lies on the boundary of X.

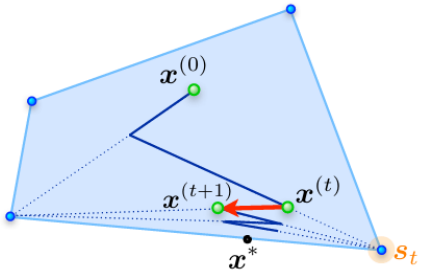

Figure 1: The FW algorithm zig-zags when the solution x* lies on the boundary

4.2 Frank-Wolfe with Momentum for Adversarial Attacks

Algorithm 2 Frank-Wolfe with Momentum Algorithm

input: number of iterations T, step sizes $\{\gamma_t\}$
$x_0 = x_{\text{ori}}, \quad m_{-1} = \nabla f(x_0)$
for $t = 0, \ldots, T-1$ **do**
    $m_t = \beta m_{t-1} + (1-\beta)\nabla f(x_t)$
    $v_t = \arg\min_{x \in \mathcal{X}} \langle x, m_t \rangle$ ▷ LMO
    $d_t = v_t - x_t$
    $x_{t+1} = x_t + \gamma_t d_t$
end for
output: x_T

The main difference between the Algorithm 1 and the following is the introduction of the momentum term $m_t = \beta m_{t-1} + (1-\beta)\nabla f(x_t)$ in place of the gradient used in the LMO and the initialization of $m\_1\nabla f(x_0)$. The method apply a weighted moving average of gradients to smooth the updates, allowing it to give a stronger weight to recent gradient information while gradually reducing the influence of older gradients. This leads to a more stable and faster convergence in practice.

4.3 Away-Steps Frank-Wolfe

In the Away-Steps Wolfe, to address the zig-zagging issue, besides the usual Frank-Wolfe direction $d_t^F W = s_t - x_t$, where $s_t \in \arg\min_{s \in \mathcal{X}} \langle \nabla f(x_t), s \rangle$, we can also move away from an active set $S^(t)$ in the so called away direction $d_t^A = x_t - v_t$, where $v_t \in \arg\max_{v \in \mathcal{S}_t} \langle \nabla f(x_t), v \rangle$. In simpler terms, the away direction d_t^A finds the atom v_t in the active set $S^(t)$ that contributes the least to improve the function. We look at the away gap: $g_A^{(t)} := \langle \nabla f(x^{(t)}), x^{(t)} - v_t \rangle$ to pick the atom that slows down descent the most. This makes AFW to linearly converge on strongly convex function, unlike standard FW, which typically achieves only sublinear convergence.

Away-Steps Frank-Wolfe keeps track of the active set $S^(t)$ and to decide if take a FW direction or Away direction, the algorithm compare the FW gap and Away gap to understand which reduce the most the objected function. The step-size in the away direction must be carefully limited to ensure that all weights in the convex combination remain non-negative and sum to one. The maximum feasible step-size

Algorithm 3 Away-steps Frank-Wolfe Algorithm (AFW)

Input: Initial point $x^{(0)} \in \mathcal{A}$ and active set $S^{(0)} := \{x^{(0)}\}$, with weights $\alpha^{(0)}$ such that $\alpha^{(0)}_{x^{(0)}} = 1$
for $t = 0, \ldots, T$ **do**
  $s_t := \mathrm{LMO}_{\mathcal{A}} \nabla f(x^{(t)}), \quad d_t^{\mathrm{FW}} := s_t - x^{(t)}$ ▷ FW direction
  $v_t := \arg\max_{v \in S^{(t)}} \langle \nabla f(x^{(t)}), v \rangle, \quad d_t^{\mathrm{A}} := x^{(t)} - v_t$ ▷ Away direction
  if $\langle \nabla f(x^{(t)}), d_t^{\mathrm{FW}} \rangle \leq \varepsilon$ **then**
    return $x^{(t)}$ ▷ FW gap is small enough
  end if
  if $\langle \nabla f(x^{(t)}), d_t^{\mathrm{FW}} \rangle \geq \langle \nabla f(x^{(t)}), d_t^{\mathrm{A}} \rangle$ **then**
    $d_t := d_t^{\mathrm{FW}}, \quad \gamma_{\max} := 1$
  else
    $d_t := d_t^{\mathrm{A}}, \quad \gamma_{\max} := \frac{\alpha^{(t)}_{v_t}}{1 - \alpha^{(t)}_{v_t}}$
  end if
  Line search: $\gamma_t := \arg\min_{\gamma \in [0, \gamma_{\max}]} f(x^{(t)} + \gamma d_t)$
  Update $x^{(t+1)} := x^{(t)} + \gamma_t d_t$
  Update weights $\alpha^{(t+1)}$ accordingly (see text)
  Update active set: $S^{(t+1)} := \{v \in \mathcal{A} \mid \alpha_v^{(t+1)} > 0\}$
end for

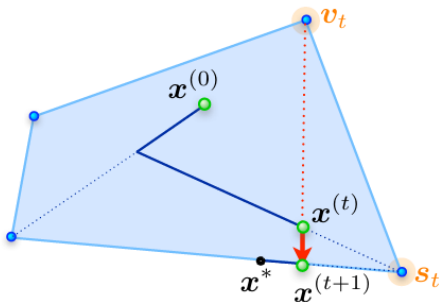

Figure 2: Adding the possibility of an away step attenuates this problem.

γ_{max} is defined to avoid moving outside the convex hull $conv(S^{(t)})$. If the chosen step size reaches γ_{max} , we perform a drop step, completely removing the atom v_t from the active set, i.e., setting its weight to zero. This behavior requires updating the weights $\alpha_v^{(t)}$ of each atom in S^t at each step. When moving in the Frank-Wolfe direction, we increase the weight of $s_t$and scale down the others. When taking an away step, we reduce the weight of v_t and scale up the others. These updates guarantee that $x^{(t)} \in \mathrm{conv}(\mathcal{S}^{(t)})$ remains true, and that feasibility is preserved without needing a projection.

4.4 Pairwise Frank-Wolfe

The key idea is to transfer weight only between two atoms at each iteration. Specifically, weight is moved from the away atom v_t to the Frank-Wolfe atom s_t, while the weights of all other atoms remain the same. So, we don't have to choose between

Algorithm 4 Pairwise Frank-Wolfe Algorithm (PFW)

Input: Initial point $x^{(0)} \in \mathcal{A}$ and active set $S^{(0)} := \{x^{(0)}\}$, with weights $\alpha^{(0)}$ such that $\alpha^{(0)}_{x^{(0)}} = 1$
for $t = 0, \ldots, T$ **do**
  $s_t := \text{LMO}_{\mathcal{A}} \nabla f(x^{(t)}), \quad v_t := \arg\max_{v \in S^{(t)}} \langle \nabla f(x^{(t)}), v \rangle$
  $d_t := s_t - v_t$ ▷ Pairwise FW direction
  $\gamma_{\max} := \alpha^{(t)}_{v_t}$ ▷ Max step-size along away direction
  Line search: $\gamma_t := \arg\min_{\gamma \in [0, \gamma_{\max}]} f(x^{(t)} + \gamma d_t)$
  Update $x^{(t+1)} := x^{(t)} + \gamma_t d_t$
  Update weights $\alpha^{(t+1)}$ accordingly (see text)
  Update active set: $S^{(t+1)} := \{v \in \mathcal{A} \mid \alpha^{(t+1)}_v > 0\}$
end for

Frank-Wolfe direction or Away direction, but combine both: $d_t^{\text{PFW}} = d_t^{\text{FW}} + d_t^{\text{A}} = s_t - v_t$. We call the swap of weights between two atoms a PWF step, i.e. Pairwise Frank-Wolfe, which is: $\alpha^{(t+1)}_{v_t} = \alpha^{(t)}_{v_t} - \gamma$ and $\alpha^{(t+1)}_{s_t} = \alpha^{(t)}_{s_t} + \gamma$ for some step size $\gamma \leq \gamma^{(t)}_{\max} := \alpha^{(t)}_{v_t}$ to keep x x_t feasible, i.e convex combination of atoms in the active set S^t. Pairwise Frank-Wolfe has a weaker theoretical convergence guarantee due to swap steps but often works better in practice, especially for sparse solutions. It efficiently reduces the active set by moving weight directly to a good atom, unlike AFW, which spreads weight more overall and may need more corrections. By actively reducing the influence of ineffective atoms, PFW reduces the oscillations typical of Vanilla FW, and for strongly convex functions, it achieves a linear convergence rate.

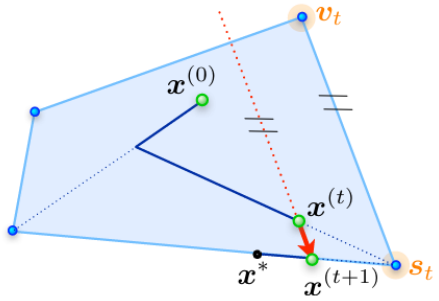

Figure 3: A pairwise FW step reducing oscillations

4.5 Projected Gradient Descent

Since in PGD the negative gradient direction may point outside the feasible region delimited by the constrained set X, we need a projection step after each gradient update. The projection ensures that the updated point remains in the acceptable perturbation set. Without this, the perturbation could overcome the allowed budget ϵ violating the adversarial constraint. Hence, PGD combines taking a step in the direction of the gradient, since we want to increase the adversarial loss, and projecting the result back in the constraint set.

Algorithm 5 Projected Gradient Method

Choose an initial point $x_1 \in C$
for $k = 1, 2, \ldots$ **do**
    Set $\hat{x}_k = \rho_C(x_k - s_k \nabla f(x_k))$, with $s_k > 0$
    if $\hat{x}_k$ satisfies some specific condition **then**
        STOP
    end if
    Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$ suitably chosen
end for

4.6 MI-FGSM

MI-FGSM, i.e. Momentum Iterative Fast Gradient Sign Method, is an extension of the PGD that attach a momentum term that allows to get a stable update direction across iterations escaping from local maxima.

Algorithm 6 MI-FGSM (Momentum Iterative Fast Gradient Sign Method)

Input: Classifier f with loss J, input x, ground-truth label y
Input: Perturbation size ϵ, number of iterations T, decay factor μ
Output: Adversarial example x^* such that $\|x^* - x\|_\infty \leq \epsilon$
$\alpha = \epsilon / T$
$g_0 = 0$, $x_0^* = x$
for $t = 0$ to $T - 1$ **do**
    Compute gradient: $\nabla_x J(x_t^*, y)$
    Update momentum:

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x J(x_t^*, y)}{\|\nabla_x J(x_t^*, y)\|_1}$$

    Update adversarial example:

$$x_{t+1}^* = x_t^* + \alpha \cdot \text{sign}(g_{t+1})$$

end for
Return: $x^* = x_T^*$

Proposed by [7], it keeps an accumulated gradient vector, updating as a weighted moving average of past gradients. At each iteration, the gradient is combined with the momentum from previous steps to form a smoother and more consistent update direction.

5 Experiments and Results

5.1 Stopping Criteria

Two types of stopping criteria were implemented:

- **a simple attack success check**
- **a convergence-based condition more appropriate for optimization analysis.**

For a more principled stopping rule, we adopted different convergence criteria depending on the algorithm.

Duality Gap. For constrained convex problems, the duality gap $g(x)$ measures how much the objective could still be improved by moving in the best feasible direction. It gives an upper bound on how far we are from the optimal value: $g(x) \geq f(x) - f(x^*)$. In the Frank-Wolfe algorithm, it's easy to compute as the inner product between the gradient and the direction from x to the linear minimizer [3]. A small duality gap means there's nosignificantly better direction to go, so we can stop the optimization.

Projected Gradient Norm. For projected gradient descent, we use as a convergence criterion the norm of the difference between the current point and its projected gradient step, given by $\|x - \rho_C(x - s\nabla f(x))\|$. When this quantity falls below a small threshold (that we set to 10^{-5}), we consider the algorithm to have converged. This condition ensures that no feasible descent direction remains and that the current point is approximately stationary.

Success vs. Convergence When using the attack success criterion, the optimization process stops as soon as the model prediction changes, regardless of how confidently the model classifies the adversarial example. As a result, the adversarial point may lie close to the decision boundary, leading to low classification confidence. In contrast, when optimizing until convergence, the algorithm continues minimizing the loss within the feasible set, often pushing the adversarial example deeper into the target class region. This difference can be observed in the visualizations below: in the success-based case, the model incorrectly classifies the adversarial image with only 9.44% confidence, while in the converged case, the confidence reaches 99.68%.

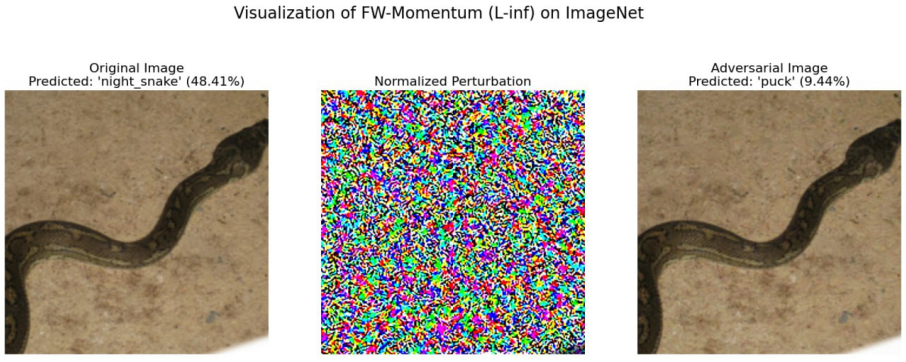

Figure 4: Adversarial example generated using the success-based stopping criterion.

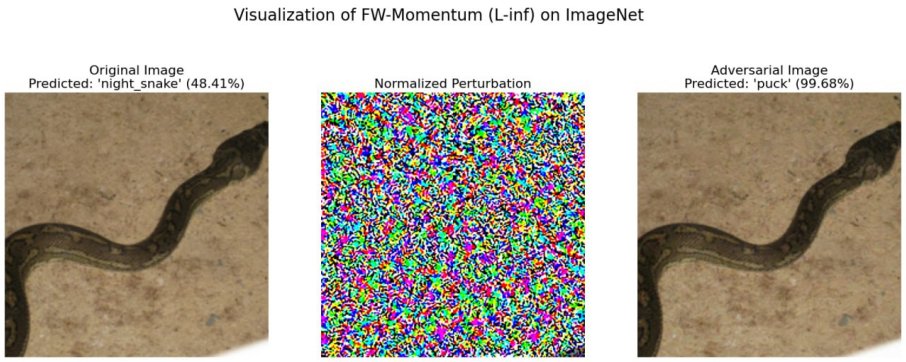

Figure 5: Adversarial example generated using convergence-based stopping criterion.

5.2 Perturbation Analysis

To determine a suitable range of perturbation magnitudes we began by evaluating the performance of each attack method across a set of increasing ε values. These exploratory runs were performed on both the MNIST and ImageNet datasets. In this initial phase, we adopted *line search* as the step-size strategy and used *attack success* as the stopping criterion.

While we acknowledge that the success criterion is not the most principled choice from an optimization standpoint (as explained above), it offered a lightweight and practical solution for efficiently trying a broad range of ε values. This was especially important given the number of algorithm variants and datasets tested. The goal at this stage was not to study convergence behavior but to identify an appropriate ε range where attacks are most effective.

To visualize this, we created plots of *attack success rate* versus ε, from which we could observe which algorithms were more effective at fooling the model, even under very small perturbations.

Once this range was determined, we proceeded with a more exhaustive set of experiments, where each algorithm was evaluated using both L_∞ and L_2 constraints, multiple step-size rules (decreasing and line search for Frank-Wolfe), and both stopping criteria (success and duality gap or projected gradient norm) enabling an analysis of both effectiveness and optimization behavior under different configurations.

Perturbation Range. For the MNIST dataset, we tested perturbation magnitudes in the range $\varepsilon \in [0.05, 0.30]$, while for ImageNet we used a finer range of $\varepsilon \in [0.001, 0.013]$.

This difference reflects the nature of the input domains: MNIST consists of low-resolution grayscale images with simple structures, where relatively large perturbations are often needed to significantly alter the prediction. In contrast, ImageNet consists of high-resolution, color images where even very small perturbations can lead to misclassification due to the higher complexity and sensitivity of the models trained on it.

By observing the results, we selected $\varepsilon = 0.3$ for MNIST and $\varepsilon = 0.005$ for ImageNet.

Comparing Algorithm Effectiveness. For the MNIST dataset, the results shown in Figure 6 indicate that while all methods can eventually succeed with large enough perturbations, certain Frank-Wolfe variants (notably with momentum or pairwise steps) achieve high ASR with smaller perturbations, making them more effective in low ε regimes. PGD performs comparably to the best FW variants, with MI-FGSM being slightly behind.

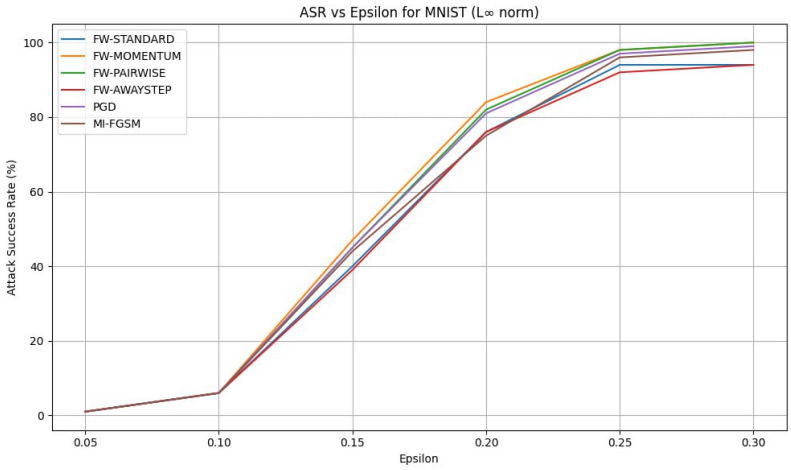

Figure 6: Attack success rate vs. ε on MNIST.

Similarly to the MNIST results, Figure 7 shows that the strongest performers on ImageNet are Frank-Wolfe with momentum and MI-FGSM, which achieve high attack success rates even at lower perturbation levels.

Comparison of Attack Behavior Across Datasets. By comparing the plots for MNIST and ImageNet, we can see that attacks on ImageNet reach high success rates with much smaller perturbations. This confirms that ImageNet models are more complex and sensitive, so even small changes are enough to fool them.

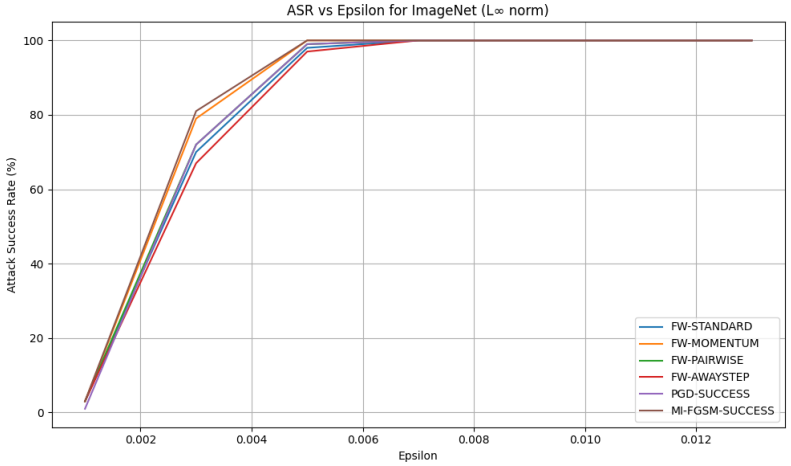

Figure 7: Attack success rate vs. ε on ImageNet.

5.3 Choice of Norm (L_∞ vs. L_2).

Overview. In adversarial attacks, the choice of norm defines the allowed perturbation set: that is, how much we can modify an input while keeping it close to the original. Specifically, we constrain the perturbation δ so that $\|\delta\|_p \leq \varepsilon$, where $p = \infty$ or $p = 2$. This constraint directly affects the feasible set used during optimization.

In L_∞ attacks, each pixel of the input can be changed independently by at most ε. On the other hand, L_2 attacks constrain the overall energy of the perturbation vector. These norms are enforced during key algorithmic steps:

- **Frank-Wolfe:** the norm defines the Linear Minimization Oracle (LMO), which selects a direction of descent within the constraint set;
- **PGD / MI-FGSM:** the norm defines the projection operator used to keep iterates inside the allowed region after each gradient step.

Experiments. Our experimental analysis primarily focused on the L_∞ norm, as it is the most commonly adopted setting in the adversarial attack literature due to its strict per-pixel constraint and alignment with standard benchmarks. However, to ensure a comprehensive comparison, we also ran all attack algorithms under the L_2 constraint. For both norms, we tested each algorithm using the step size strategies and stopping criteria mentioned in the previous paragraphs.

To select a suitable value for ε under the L_2 norm, we performed a sweep over multiple candidates and chose the value that resulted in an attack success rate most comparable to the corresponding L_∞ configuration. We ended up using $\varepsilon = 3.5$ for MNIST and $\varepsilon = 1$ for ImageNet.

Results. Across all attack methods, we observed that L_∞-constrained attacks consistently achieved higher success rates with lower average distortion compared to their L_2 counterparts. This is expected, as the L_∞ norm tightly bounds the maximum perturbation per pixel, often resulting in more effective and less perceptible adversarial examples. In contrast, L_2-based attacks, while allowing more distributed noise, required higher overall distortion to reach similar success, especially under stricter stopping criteria.

Both norms followed consistent trends in response to step size and stopping choices.

For simplicity, in the table and images below we only report the results obtained for Frank-Wolfe with momentum, using line search as the step size strategy and convergence as the stopping criterion. Similar trends were observed across all other algorithms and methods.

5.4 Stepsize

For all Frank-Wolfe variants, we implemented two step-size strategies: a *decreasing stepsize* and a *line search* method. The decreasing stepsize follows the update rule $\gamma_t = \frac{2}{t+2}$, which is a commonly used schedule proposed in the modern analysis of Frank-Wolfe methods [3]. This rule is simple to implement and guarantees convergence for convex problems.

Norm	ASR (%)	# Iterations	Distortion	Time(s)
L_∞	100	40	0.3	37.95
L_2	95	39.61	3.1459	37.85

Table 1: Performance comparison of Frank-Wolfe with momentum on the MNIST dataset under L_∞ and L_2 norm constraints, using line search for step size and duality gap as stopping criterion.

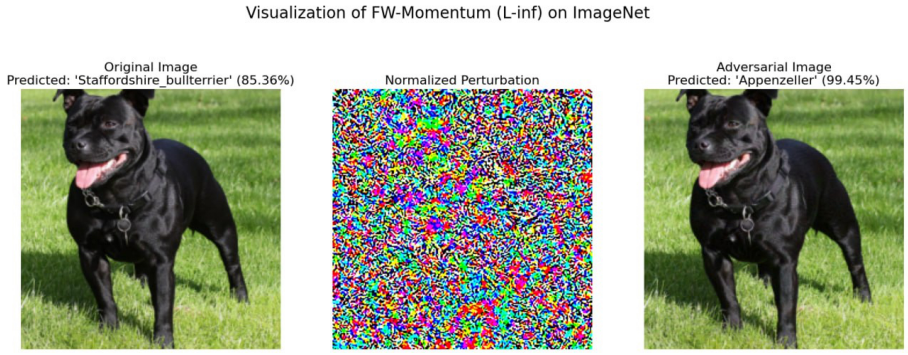

Figure 8: FW-Momentum adversarial example on ImageNet using the L_∞ norm. The perturbation modifies almost all pixels within the allowed range, resulting in dense but small-magnitude changes.

For all the models we implemented a *line search* strategy, as described in Algorithm 3 of the same paper [3], which selects the optimal $\gamma \in [0, 1]$ at each iteration by minimizing the loss along the update direction. This adaptive approach often provides better empirical performance, especially in non-convex settings such as adversarial attacks on neural networks.

In all our experiments, the line search was discretized using 10 equally spaced candidate values for γ.

Results. Across all Frank-Wolfe variants, both line search and decreasing stepsize achieved high attack success rates. Line search occasionally resulted in slightly lower distortion, but at the cost of noticeably higher computational time. Decreasing stepsize remains more efficient overall, especially when the goal is to generate successful adversarial examples quickly.

5.5 Effect of Momentum

In the table 1 we compare targeted L_∞ attack methods using line search as step size and duality gap as convergence criterion on MNIST dataset . FW with momentum achieves the highest Attack Success Rate (100 percent) keeping distorsion very similar to the others algorithms. It needs more iterations and a moderate runtime indicating a good compromise between effectiveness and efficiency. while FW Standard has a faster but slightly less successful alternative. FW standard has the lowest runtime (26.63) and number of iterations (27.91), but have also the lowest ASR (94 percent). Hence, it suggests a trade-off between speed and effectiveness. PGD and MI-FGSM also perform competitively, with ASRs of 99.0 percent and 98.0 percent,

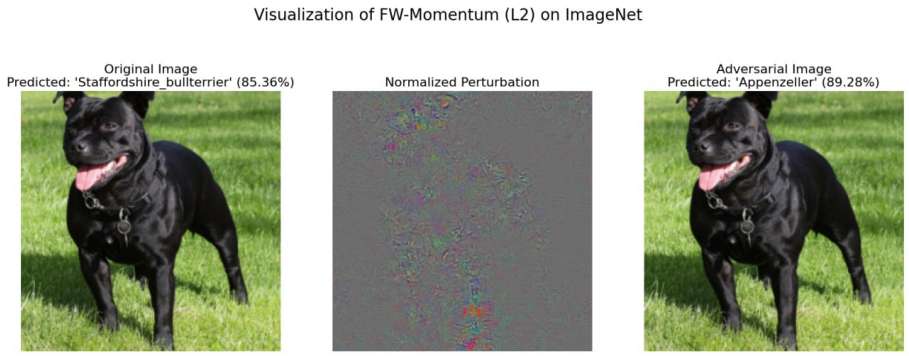

Figure 9: FW-Momentum adversarial example on ImageNet using the L_2 norm. The perturbation tends to concentrate the distortion into fewer pixels, allowing for larger changes or peaks in specific regions.

respectively. However, both have higher runtime costs (43.67s and 45.95s) and more iterations (37.02 and 38.19) compared to FW Momentum, despite achieving slightly lower ASR.

Methods	**ASR (%)**	**# Iterations**	**Distortion**	**Time(s)**
FW Standard	94.0	27.91	0.294	26.63
FW Momentum	100.0	40.0	0.300	37.95
PGD	99.0	37.02	0.2952	43.67
MI-FGSM	98.0	38.19	0.2979	45.95

Table 2: Comparison of targeted L_∞ norm based white-box attacks on MNIST dataset with ε=0.3, step size=line search and convergence criterion= duality gap

Methods	**ASR (%)**	**# Iterations**	**Distortion**	**Time(s)**
FW Standard	83.0	27.69	3.0027	26.12
FW Momentum	95.0	39.61	3.1459	37.85
PGD	75.0	17.59	3.4915	24.07
MI-FGSM	68.0	20.53	3.4899	27.67

Table 3: Comparison of targeted L_2 norm based white-box attacks on MNIST dataset with ε=3.5, step size=line search and convergence criterion= duality gap

In the table 2 we can see that FW with momentum outperforms the other algorithms in terms of Attack Success Rate (95 percent), It also shows relatively moderate distortion (3.1459) and runtime (37.85s), despite the highest iteration count (39.61), indicating a trade-off between effectiveness and computational effort. Hence, the momentum allows to enforce the attack effectiveness, but with a growth of the computational complexity and perturbation size. FW Standard follows with an ASR of 83.0 percent, fewer iterations (27.69), and slightly lower distortion (3.0027), making it a more efficient alternative with acceptable performance. In contrast, PGD and MI-FGSM achieve significantly lower ASRs, 75.0 and 68.0 respectively, despite being faster (24.07s and 27.67s) and need less iterations. However, they also

exhibit higher distortion values (3.4915 and 3.4899), indicating lower precise perturbation. In light of these results, These results indicate that Frank-Wolfe-based methods, especially FW Momentum, are more effective for targeted L_2 norm attacks on MNIST.

5.6 Frank-Wolfe and PGD Comparison

When comparing PGD and FW-Standard under the same settings (line search, L_∞ norm, and convergence-based stopping) for the MNIST dataset, we see that PGD achieves a higher attack success rate and slightly higher distortion. However, it also takes more iterations to converge and has a significantly higher total runtime. If we compute the average time per iteration, PGD takes about 1.18 seconds per step, while FW-Standard only takes around 0.95 seconds. This supports the idea that projections in PGD are more expensive than the linear minimization step used in Frank-Wolfe. So, while FW may not always converge in fewer steps, its projection-free nature makes each iteration faster in practice.

Method	**ASR (%)**	**# Iterations**	**Distortion**	**Time(s)**
FW-Standard	94.0	27.91	0.2940	26.63
PGD	99.0	37.02	0.2952	43.67

Table 4: Comparison between Frank-Wolfe and PGD under L_∞ norm on MNIST.

6 Convergence

We empirically verified the convergence rate of the Frank-Wolfe adversarial attack with momentum proposed by [1]. The paper theoretically proves that their white-box and black-box attack algorithms converge at a rate of $\mathcal{O}(1/\sqrt{T})$ in a non-convex setting. To validate this claim, we conducted an experiment to analyze its performance.

The experiment was executed on ResNet18 over 5000 iterations on a single image using the decreasing stepsize variants, using the same parameters we previously used for attacking the ResNet18: momentum of 0.9, $\varepsilon = 0.005$. The convergence constant C was estimated as $C \approx \text{Gap} \cdot \sqrt{T}$ using the final gap at iteration T.

For more robust results, the experiment should be run on more images, averaging the gaps, but it proved to be a computationally expensive process and we decided to proceed with only one image.

Analysis of Results The results were visualized on a log-log plot, where a rate of $\mathcal{O}(1/\sqrt{T})$ corresponds to a straight line with a slope of -0.5.

During the early iterations, the iterate is far from the optimum, causing the LMO to select vertices that result in large, high-variance update steps, but as we approach the optimal solution, the magnitude of the update vectors decreases and their direction stabilizes. At this point, the convergence dynamics becomes more regular, aligning with the theoretical rate.

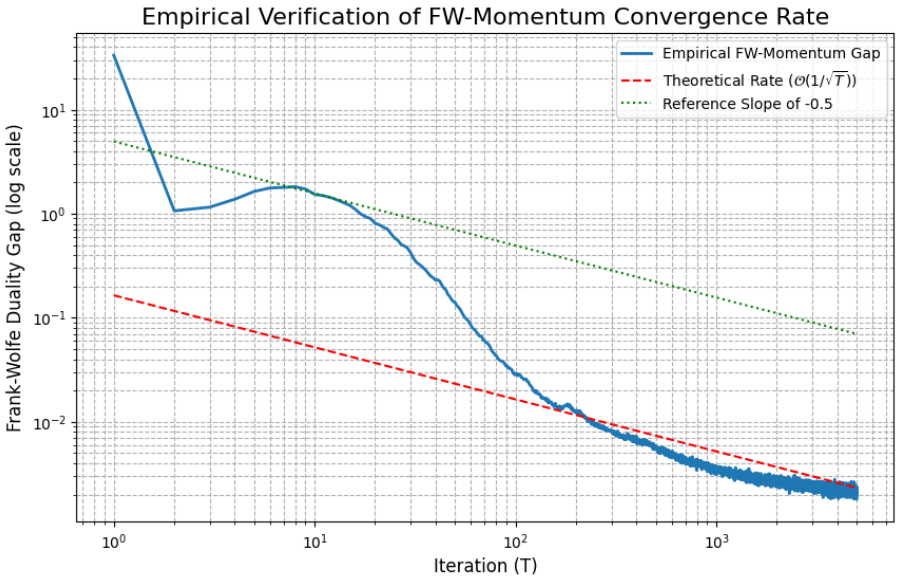

Figure 10: Duality gaps on ImageNet versus the number of iterations (T).

7 Conclusion

We have evaluated projection-free and projection-based optimization methods for working on white-box attacks, focusing on FW and its variants, PGD standard and with momentum, MI-FGSM. The experiments on MNIST and ImageNet datasets under L_2 and L_∞ norms have shown that FW with momentum constantly achieves high Attack Success Rate with reasonably low distorsion and efficient convergence, providing a competitive, and in L_∞ often faster, alternative to projection-based methods by avoiding costly projection steps. The results achieved in this project highlight the overall effectiveness of projection-free approaches in generating successful and precise adversarial examples.

8 References

[1] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, Quanquan Gu. 2019. A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks. `https://arxiv.org/pdf/1811.10828`

[2] Immanuel M. Bomze, Francesco Rinaldi, Damiano Zeffiro. 2021. Frank-Wolfe and friends: a journey into projection-free first-order optimization methods. `https://arxiv.org/pdf/2106.10261`

[3] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. `https://proceedings.mlr.press/v28/jaggi13-supp.pdf`

[4] Simon Lacoste-Julien, Martin Jaggi. 2015. On the Global Linear Convergence of Frank-Wolfe Optimization Variants. `https://arxiv.org/pdf/1511.05932`

[5] Cyrille W. Combettes, Sebastian Pokutta. 2021. Complexity of Linear Minimization and Projection on Some Sets. https://arxiv.org/pdf/2101.10040

[6] Laurent Condat. Fast Projection onto the Simplex and the l1 Ball. Mathematical Programming, Series A, 2016, 158 (1), pp.575-585. https://hal.archives-ouvertes.fr/hal-01056171v2

[7] Dong et al. 2018. Boosting Adversarial Attacks with Momentum. https://arxiv.org/pdf/1710.06081