Comparative Analysis of Frank-Wolfe and Pairwise Frank-Wolfe Algorithms for Adversarial Attacks on Image and Audio Data

Maxim Nitsenko

Università degli studi di Padova, Dipartimento di Matematica, Padova, Italia
E-mail: maximnikolaevich.nitsenko@student.unipd.it
Matricola: 2129684

1. Introduction

Machine learning models, despite their remarkable performance across various domains, have been shown to be vulnerable to adversarial attacks, carefully crafted perturbations that can cause models to make incorrect predictions. These vulnerabilities pose significant challenges to the reliability and security of artificial intelligence systems, spanning critical applications from autonomous driving to medical diagnostics.

The Frank-Wolfe (FW) algorithm, a classical optimization technique, has gained renewed attention in recent years due to its effectiveness in solving constrained optimization problems, including adversarial attack generation. Unlike gradient-based methods that require costly projection steps, the FW algorithm operates by iteratively solving linear approximations of the objective function over a feasible set, making it computationally efficient for large-scale problems.

This project reportr focuses on the application of the FW and PWFW algorithms to generate adversarial attacks on two distinct datasets: MNIST, a benchmark dataset of handwritten digits, and on an audio dataset of spoken digits. These datasets represent different input modalities: images and audio, and provide a comprehensive testbed for evaluating the performance of optimization algorithms in adversarial settings. The goal of the project is to compare the effectiveness, computational efficiency, and convergence behavior of FW and PWFW in generating targeted adversarial attacks under varying perturbation budgets and target classes.

The key objectives of this project are:

- **Implementation and Evaluation**: Implement the FW and PWFW algorithms for adversarial attack generation and evaluate their performance on both image and audio datasets.
- **Comparative Analysis**: Compare the two algorithms in terms of success rates, computational efficiency, and convergence properties across different perturbation budgets and target classes.
- **Practical Insights**: Provide insights into the strengths and limitations of FW and PWFW in adversarial settings, identifying scenarios where one algorithm may be more suitable than the other.

This report is structured as follows: Section 2 provides background on adversarial attacks, Section 3 introduces the Frank-Wolfe algorithms, Section 4 describes the experimental methodology, Section 5 presents the results, and Section 6 discusses the implications and potential future research directions.

2. Adversarial Attacks

2.1 Introduction

Machine learning models, particularly deep neural networks, have achieved remarkable success across a wide range of domains, from image recognition to natural language processing. However, these models are vulnerable to adversarial attacks: carefully crafted perturbations to input data that can cause the model to make incorrect predictions. These vulnerabilities pose significant challenges to the reliability and security of AI systems, especially in critical applications such as autonomous driving, medical diagnostics, and cybersecurity.

Adversarial attacks exploit the sensitivity of machine learning models to small changes in input data. Despite their high accuracy, these models often learn decision boundaries that can be easily manipulated. By introducing subtle, often imperceptible perturbations to the input, attackers can alter the model's predictions, sometimes with potentially dangerous consequences. Understanding and mitigating these vulnerabilities has therefore become a critical area of research in machine learning.

A particularly severe concern with adversarial examples is their high transferability. Adversarial examples crafted for one model often remain effective when used against other models, raising significant security concerns. This transferability occurs because different machine learning models tend to learn similar decision boundaries around the same data points. As a result, adversarial examples generated for one model can generalize to others, even if the models have different architectures or training procedures. [3]

While adversarial attacks are often associated with deep neural networks due to their complexity and susceptibility to overfitting, it is important to note that adversarial vulnerability is an intrinsic property of all classifiers. This means that even simpler machine learning models, such as logistic regression and decision trees, are not immune to adversarial manipulation, further emphasizing the importance of robust defenses. [6].

2.2 Types of Adversarial Attacks

Adversarial attacks can be categorized along two orthogonal dimensions: targeted vs. untargeted and white-box vs. black-box

2.2.1 Targeted vs. Untargeted Attacks

- Targeted Attacks: The attacker aims to misclassify the input into a specific target class. For example, an image of a cat might be perturbed to be classified as a dog.
- Untargeted Attacks: The attacker aims to misclassify the input into any incorrect class, regardless of the specific class.

2.2.2 White-Box vs. Black-Box Attacks

- White-Box Attacks: The attacker has full knowledge of the model, including its architecture, parameters, and gradients. This allows the attacker to craft highly

effective adversarial examples using techniques like gradient-based optimization.

- Black-Box Attacks: The attacker has no knowledge of the model's internal workings and relies on querying the model to craft adversarial examples. Despite the lack of internal knowledge, black-box attacks can still be effective due to the transferability of adversarial examples: perturbations crafted for one model often fool other models.

The project focuses on targeted white-box attacks, where the goal is to modify an input (e.g., an image or audio clip) such that the model predicts a specific target class, leveraging full knowledge of the model.

2.3 Formal Description of a Targeted Attack

While there are several formulations of a targeted adversarial attack, below is the **maximum allowable attack** formulation for neural networks. [6]

- Let $f : \Omega \subseteq \mathbb{R}^d \to \mathbb{R}^K$ represent a neural network classifier that outputs log probabilities for K classes. For an input $x \in \Omega$, the classifier produces $f(x)$, where $f(x)_k$ denotes the log probability assigned to class k.
- Let $x_{\text{att}} \in \Omega$ be the original input example to be perturbed by the attack.
- Let $y_{\text{tar}} \in \{1, 2, \ldots, K\}$ be the target class, representing the class to which the adversarial example should be misclassified.
- Let $\epsilon > 0$ define the perturbation budget, which constrains the magnitude of the perturbation, ensuring that the adversarial example remains close to the original input in a specified norm.

The objective of the targeted attack is to construct an adversarial example $x_{\text{adv}} \in \Omega$ such that:

1. The classifier assigns the highest log probability to the target class y_{tar}, i.e., $f(x_{\text{adv}})_{y_{\text{tar}}}$ is maximized.
2. The perturbation magnitude is constrained by ϵ, ensuring that $\|x_{\text{adv}} - x_{\text{att}}\| \leq \epsilon$ in a specified norm (e.g., L_2 or L_∞).
3. The perturbed input remains valid, i.e., $x_{\text{adv}} \in \Omega$.

Mathematically, the optimization problem can be expressed as:

$$x_{\text{adv}} = \arg\max_{x \in \Omega} f(x)_{y_{\text{tar}}} \quad \text{subject to} \quad \|x - x_{\text{att}}\| \leq \epsilon$$

This is generally a non-convex, constrained optimization problem, which is particularly well-suited for solving within the Frank-Wolfe optimization framework.
An attack is considered successful if the classifier assigns the highest log probability to the target class y_{tar} for the adversarial example x_{adv}. Formally, this is written as:

$$f(x_{\text{adv}})_{y_{\text{tar}}} > f(x_{\text{adv}})_i, \quad \forall i \in \{1, \ldots, K\},\ i \neq y_{\text{tar}}$$

or, equivalently:

$$\arg\max_i f(x_{\text{adv}})_i = y_{\text{tar}}$$

2.4 Visual Example of Adversarial Attacks

To demonstrate the concept of adversarial attacks, an illustrative example is provided in Figure 1. The figure depicts an original image of a handwritten digit "3" from the MNIST dataset. A small, carefully crafted perturbation is applied to the image, resulting in an adversarial example that is almost indistinguishable from the original.

Despite the minimal visual difference, the model misclassifies the perturbed image as a "7", which is the target class for the attack. This demonstrates how adversarial examples exploit model vulnerabilities by shifting decision boundaries, causing incorrect classifications even when changes are small or imperceptible to humans.

The accompanying probability distributions further highlight the effectiveness of the attack. Before the perturbation, the model assigns a high probability to the correct class, with $p(3|x_{\text{att}}) = 0.89$, and a negligible probability to the target class, with $p(7|x_{\text{att}}) = 0.017$. After the attack, the probability distribution shifts significantly, with $p(7|x_{\text{adv}}) = 0.44$, making it the highest probability across all classes and successfully misleading the model to the target class.

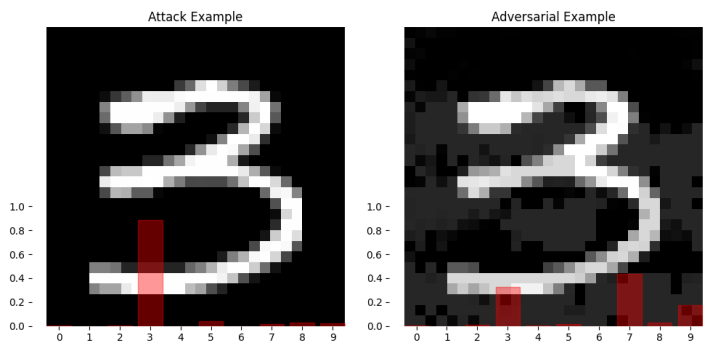

Figure 1: Comparison of the original input (left) and the adversarially perturbed input (right). The attack was performed using the L_∞ norm with a perturbation budget of $\epsilon = 0.15$.

3. Frank-Wolfe Algorithms

3.1 Vanilla Frank-Wolfe (FW) Algorithm

The Frank-Wolfe (FW) algorithm (Algorithm 1), also known as the conditional gradient method, is a classical first-order optimization technique introduced in 1956. It is designed to solve constrained convex optimization problems, a class of problems that includes machine learning adversarial attacks. At each iteration, it finds a search direction by minimizing a linear approximation of the objective function over the feasible set. Once the search direction is determined, the new iterate is computed as a convex combination of the current point and the direction found, with the step size controlled by a predefined schedule or line search. This process repeats, progressively refining the solution until convergence. Unlike gradient projection methods, which require costly projection steps to map iterates back onto the feasible set, the FW algorithm avoids projections entirely. Instead, The minimization is done using a linear oracle, which efficiently solves the linear minimization problem.

An important measure of progress in the FW algorithm is the duality gap, which provides a bound on the suboptimality of the current iterate. The duality gap at iteration k is defined as:

$$g(x_k) = -\nabla f(x_k)^T(s_k - x_k),$$

where s_k is the solution of the linear minimization problem. This gap satisfies two key properties: it is always non-negative ($g(x_k) \geq 0$), and a point x_k is stationary (i.e., a first-order optimal solution) if and only if $g(x_k) = 0$. Importantly, computing the duality gap is essentially free in the FW algorithm, as it relies on quantities that are already computed during the iteration.

For convex functions, the convergence of the Frank-Wolfe algorithm can be proven to be sublinear. This occurs because, when the solution lies on the boundary of the constraint set, the algorithm tends to zigzag, slowing down the rate of convergence. For non-convex objectives Frank-Wolfe algorithm achieves a convergence rate of $O(1/\sqrt{k})$ [1].

3.1.1 Away-Step Frank-Wolfe (ASFW)

To address the zigzagging issue, the away-step variant of the Frank-Wolfe algorithm (ASFW) was proposed. In ASFW, the algorithm not only computes the standard Frank-Wolfe direction, which moves toward a new vertex in the constraint set that minimizes the linear approximation of the objective function, but also computes an away direction. The away direction moves away from a vertex in the current active set that is no longer beneficial, specifically the one that has the largest correlation with the gradient. This variant ensures linear convergence for strongly convex objectives on polytopes, even when the optimal solution lies on the boundary of the feasible set [1].

3.1.2 Pairwise Frank-Wolfe (PWFW)

The Pairwise Frank-Wolfe (PWFW) algorithm (Algorithm 2) is another variant designed to mitigate the zigzagging behavior observed in the standard Frank-Wolfe algorithm. PWFW performs a pairwise exchange of weights between two vertices in the active set. Specifically, the algorithm computes the sum of the standard Frank-Wolfe direction and the away direction, combining them into a single update. This combined direction reduces the weight of less relevant vertices while promoting the weight of more promising ones. By integrating both directions, PWFW achieves a more balanced exploration of the feasible set and often converges faster than the standard Frank-Wolfe algorithm, especially when the optimal solution lies on the boundary of the feasible set. This variant also ensures linear convergence for strongly convex objectives on polytopes [1].

Algorithm 1 Frank-Wolfe Algorithm

1: Choose a point $x_0 \in C$
2: **for** $k := 0$ to `maxiter` **do**
3: **if** x_k satisfies some specific condition **then**
4: **STOP**
5: **end if**
6: $s_k := \text{argmin}_{x \in C} \langle \nabla f(x_k), x - x_k \rangle$
7: $x_{k+1} := x_k + \alpha_k(s_k - x_k)$, with $\alpha_k \in [0, 1]$
8: **end for**

Algorithm 2 Pairwise Frank-Wolfe Algorithm

Choose a point $x_0 \in C$
$S_0 := \{(x_0, w_{x_0} = 1)\}$
for $k := 0$ to `maxiter` **do**
    if x_k satisfies some specific condition **then**
        STOP
    end if
    $s_k := \text{argmin}_{x \in C} \langle \nabla f(x_k), x - x_k \rangle$
    $v_k := \text{argmax}_{v \in S_k} \langle \nabla f(x_k), v \rangle$
    $d_k := s_k - v_k$
    $x_{k+1} := x_k + \alpha_k d_k$, with $\alpha_k \in [0, w_{v_k}]$
    $S_{k+1} := S_k \setminus \{(v_t, w_{v_t}) \cup (s_t, w_{s_t})\} \cup \{(v_t, w_{v_t} - \alpha_k) \cup (s_t, w_{s_t} + \alpha_k)\}$
end for

3.2 Implementation Details

3.2.1 Constraint Set

The constraint set is managed through the `FeasibleSet` class, which is an abstract base class for defining feasible sets in optimization. It includes methods to check set membership, measure constraint violations, and find the linear minimization oracle (LMO) solution. Subclasses like `Box`, `LInf_Ball`, `L2_Ball`, and `Unit_Simplex` implement these methods for specific types of sets (e.g., boxes, L_∞ balls, L_2 balls, and simplices).

Additionally, the `Box` class supports the intersection of two box constraints, allowing the creation of a new box that represents the overlapping feasible region of the two original sets. This feature is particularly useful in adversarial attacks, where the original domain of the input Ω needs to be intersected with an epsilon ball.

However, if a norm other than the L_∞ norm is used, such as the L_2 norm, computing the intersection becomes more challenging. In these cases, instead of directly computing the intersection, the solution must be projected onto the feasible set Ω to ensure that it lies within the desired domain.

3.2.2 Active Set

The `ActiveSet` class manages active atoms (vertices) and their weights in optimization algorithms like Frank-Wolfe. It uses a custom hashing function based on the binary representation of PyTorch tensors to store atoms in a dictionary, which speeds up lookups and updates. Key features of the `ActiveSet` class include:

- Adding and removing atoms.
- Computing the current iterate as a weighted sum of the active atoms.
- Finding the atom with the maximum dot product with the gradient, which is used in the Frank-Wolfe algorithm.

This `ActiveSet` structure enables efficient handling of FW variants. By dynamically managing the set of active atoms, the implementation mitigates the zig-zagging phenomenon associated with standard FW and ensures a more stable convergence behavior.

3.2.3 Network Architecture

The code includes two neural network models for different tasks:

1. **Video Model**:
   - This model is designed for classifying handwritten digits from the MNIST dataset. It consists of:
     - Two convolutional layers with ReLU activation and max-pooling.
     - A dropout layer after the second convolutional layer to prevent overfitting.
     - Two fully connected layers that map the features to the 10 output classes.
     - A log-softmax layer to produce class probabilities.
   - The input shape is `[1, 1, 28, 28]`, corresponding to grayscale images of size 28x28.
   - The model achieves an accuracy of 97.5% on the MNIST test set.
2. **Audio Model**:
   - The model is based on [2] developed specifically for audio data. It consists of:
     - Four convolutional layers with batch normalization and max-pooling.
     - A fully connected layer that maps the features to the 10 output classes (digits 0-9).
     - A log-softmax layer to produce class probabilities.
   - The input shape is `[1, 1, 8000]`, corresponding to 1D audio waveforms resampled to 8000 Hz.
   - The model achieves an accuracy of 92.6%

Both models are trained using gradient-based optimization and are used as the target networks for adversarial attacks.

3.2.4 Optimization Algorithms

The code implements several optimization algorithms for constrained optimization, including:

1. **Frank-Wolfe (FW)**:
2. **Pairwise Frank-Wolfe (PWFW)**:
3. **Away Step Frank-Wolfe (ASFW)**:

All these methods are implemented by a single function, `UFW` (Universal Frank Wolfe), which uses the specified `method` parameter to execute the corresponding algorithm.

3.2.5 Line Search

The code uses backtracking to determine the step size in the optimization algorithms. The Armijo condition ensures that the objective function decreases sufficiently by iteratively reducing the step size until the following condition is met:

$$f(x_k + \alpha d_k) \leq f(x_k) + \gamma\alpha\langle\nabla f(x_k), d_k\rangle$$

where:

- α is the step size,
- d_k is the search direction, and
- γ is a small constant (e.g., 10^{-4}).

This line search method ensures that the optimization algorithms converge efficiently while maintaining feasibility. Importantly, the Armijo line search guarantees convergence for non-convex functions [4], as it ensures sufficient decrease in the objective function at each step, preventing stagnation or divergence.

4. Experiments

4.1 Datasets Description

In this study, two datasets were utilized: MNIST and a spoken digit dataset derived from the Speech Commands dataset. Each dataset is described below. These two datasets provide diverse input domains for evaluating the proposed methods. The MNIST dataset represents a relatively low-dimensional input space ($[0,1]^{784}$), while the filtered Speech Commands dataset offers a significantly higher-dimensional input space ($[-1,1]^{8000}$). This contrast allows for a comprehensive study of the algorithms under different settings, highlighting their performance and behavior across varying dimensionalities and data modalities.

4.1.1 MNIST

The MNIST dataset [5] is a widely used benchmark in machine learning, consisting of grayscale images of handwritten digits ranging from 0 to 9. It is primarily used for tasks such as image classification and deep learning research. The key characteristics of the dataset are as follows:

- **Size**: 60,000 training images and 10,000 test images.
- **Format**: Each image is 28x28 pixels, flattened into a 784-dimensional vector.
- **Range**: Each pixel intensity is normalized to lie within the range $[0,1]$, resulting in the feasible input domain $\Omega = [0,1]^{784}$.

4.1.2 Speech Commands Dataset (Filtered for Digits)

The Speech Commands dataset [7] is a collection of short audio recordings, each containing a single spoken word. For this study, the dataset was filtered to include only recordings of spoken digits. The key characteristics of the filtered dataset are as follows:

- **Content**: WAV audio files, each containing a single spoken digit.
- **Classes**: 10 classes corresponding to the digits 0 through 9.
- **Size**: After filtering, the dataset contains 38908 audio samples.
- **Sampling Rate**: Each audio file is sampled at 16 kHz.
- **Preprocessing Steps**:
  - Audio recordings were resampled from 16 kHz to 8 kHz.

  - Waveforms were padded to a fixed length of 8,000 samples.

- **Range**: Each audio sample lies within the range $[-1, 1]$, as provided by the dataset, resulting in the feasible input domain $\Omega = [-1, 1]^{8000}$.

4.2 Experimental Setup

A comprehensive evaluation was conducted to assess the performance of the Frank-Wolfe (FW) and Pairwise Frank-Wolfe (PWFW) algorithms in the context of adversarial attacks on neural network classifiers. The experiments were designed to compare the effectiveness of these algorithms under varying perturbation budgets and target classes.

- **Dataset Selection**: The evaluation considers only examples that are correctly classified by the model and excludes instances where the true class coincides with the target class of the attack.
- **Target Class**: The target class for all experiments is fixed to 7.
- **Stopping Criteria**: The algorithms run for a maximum of 500 iterations and terminate early if:
  - The duality gap falls below 0.0005, or
  - The objective function stops decreasing.
- **Attack Norms**: Only L_∞ norm is considered for adversarial perturbations.
- **Performance Metrics**: The following metrics are used to assess algorithm performance:
  1. **Success Rate** –The proportion of successful adversarial attacks.
  2. **Elapsed Time** –The time required to generate adversarial examples.
  3. **Average Iterations** –The mean number of iterations before convergence.
  4. **Proportion FW > PWFW** –The proportion of instances where the PWFW algorithm achieves a lower (better) objective function value than PW is computed, considering only the cases where the values differ.
- **Number of Attacks per Test**: For each test configuration, 50 attacks were performed to ensure statistical significance

4.2.1 Experiment Types

1. Perturbation Budget Analysis

- The behavior of both algorithms is analyzed across a range of perturbation budgets (ϵ), ensuring coverage of accuracy values from 0 to 1.
- The tested ϵ values for different data types are:
  - **Images**: $[0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.325, 0.35]$
  - **Audio**: $[0.00001, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008]$

2. Cross-Class Performance Analysis

- The performance of FW and PWFW is evaluated when attacking different digit classes (0–9, excluding 7).
- A fixed perturbation budget is used:
  - $\epsilon = 0.15$ for images
  - $\epsilon = 0.001$ for audio

5. Results

5.1 Image Domain

5.1.1 Epsilon-Based Attack

Both FW and PWFW achieve the same accuracy across different values of ϵ. As expected, increasing ϵ leads to a higher attack success rate, as subtler perturbations are less effective. A compromise between accuracy and visual quality is found at $\epsilon = 0.15$. The smallest radius that allows a nonzero probability of a successful attack is $\epsilon > 0.07$. Beyond $\epsilon = 0.35$, accuracy saturates, but distortions become excessive and unacceptable.

Regarding computational efficiency, both methods exhibit an almost linear increase in the number of iterations as ϵ grows. PWFW consistently requires fewer iterations, completing attacks with an average of 93 fewer steps per instance. This translates into a significant reduction in execution time: 6.8s for FW versus 3.93s for PWFW, with the largest time difference of 4.67s occurring at $\epsilon = 0.15$. Despite FW producing better adversarial examples in 84% of cases, this does not impact accuracy.

The results of the epsilon-based attack are visualized in Figure 2, which shows the accuracy, average time, average iterations, and the proportion of instances where PWFW outperforms FW across different values of ϵ. Figure 3 provides a grid of adversarial examples generated for different digits and ϵ values.

5.1.2 Single-Digit Attack

For $\epsilon = 0.15$, both methods achieve similar accuracy across most digits, except for digit 1 (where FW performs slightly better) and digit 8 (where PWFW performs slightly better). The overall accuracy remains at 0.28.

The easiest digit to attack is 9 (success rate = 0.78), while the most challenging is 6 (success rate = 0.08). The methods perform significantly better on digit 9, likely due to its visual similarity to 7. PWFW requires fewer iterations per attack (246 vs. 334 for FW), and digit 1 stands out as requiring the highest number of iterations.

FW achieves lower objective function values in 87% of the cases where the two methods produce different results. However, this does not translate into a meaningful accuracy difference.

The results of the single-digit attack are visualized in Figure 4, which shows the accuracy, average time, average iterations, and the proportion of instances where PWFW outperforms FW across different digits.

Figure 2: Image Domain - Epsilon-Based Attack

Figure 3: Grid of adversarial examples for different digits and ϵ values. The border color indicates the success of the attack (green for successful, red for unsuccessful). The digits corresponding to each ϵ were selected randomly.

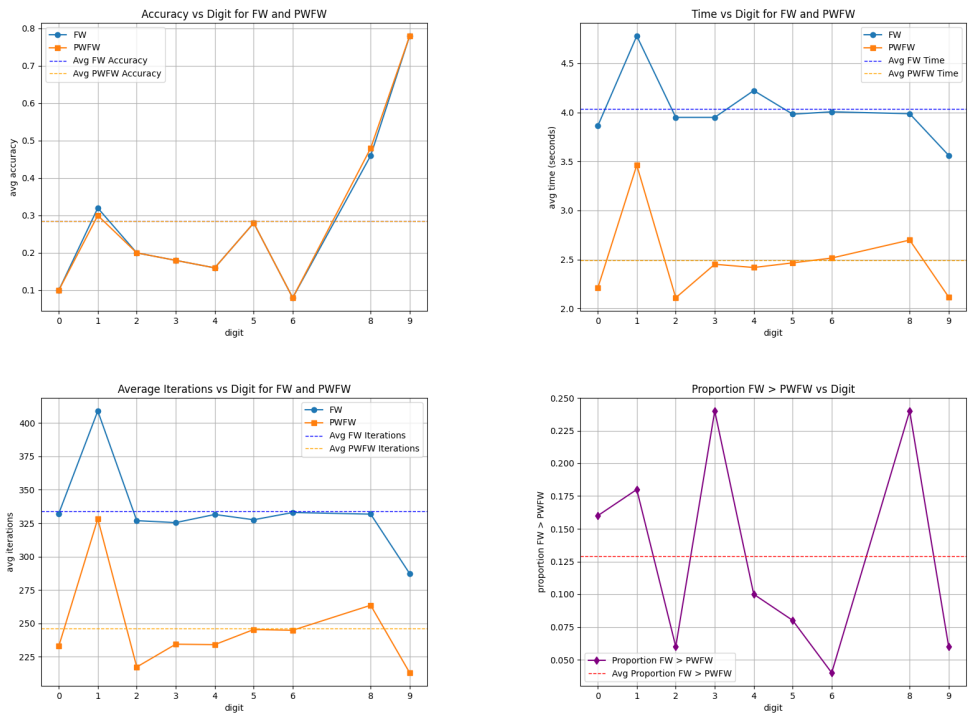

Figure 4: Image Domain - Single-Digit Attack

5.2 Audio Domain

5.2.1 Epsilon-Based Attack

For audio adversarial attacks, FW and PWFW exhibit comparable performance across various values of ϵ, though FW generally achieves slightly higher accuracy (0.767) compared to PWFW (0.747).

The average iterations and execution time per attack exhibit a similar trend across different ϵ values: PWFW performs better for smaller ϵ values but tends to underperform for larger ones. On average, PWFW is slightly slower, with an average execution time of 6.94s compared to FW's 7.76s. In terms of iterations, the two methods are nearly identical, with both averaging around 192 iterations per attack.

The proportion of cases where PWFW performs better than FW averages around 0.19, indicating unbalanced performance in favor of FW in most scenarios.

The results of the epsilon-based attack in the audio domain are visualized in Figure 5.

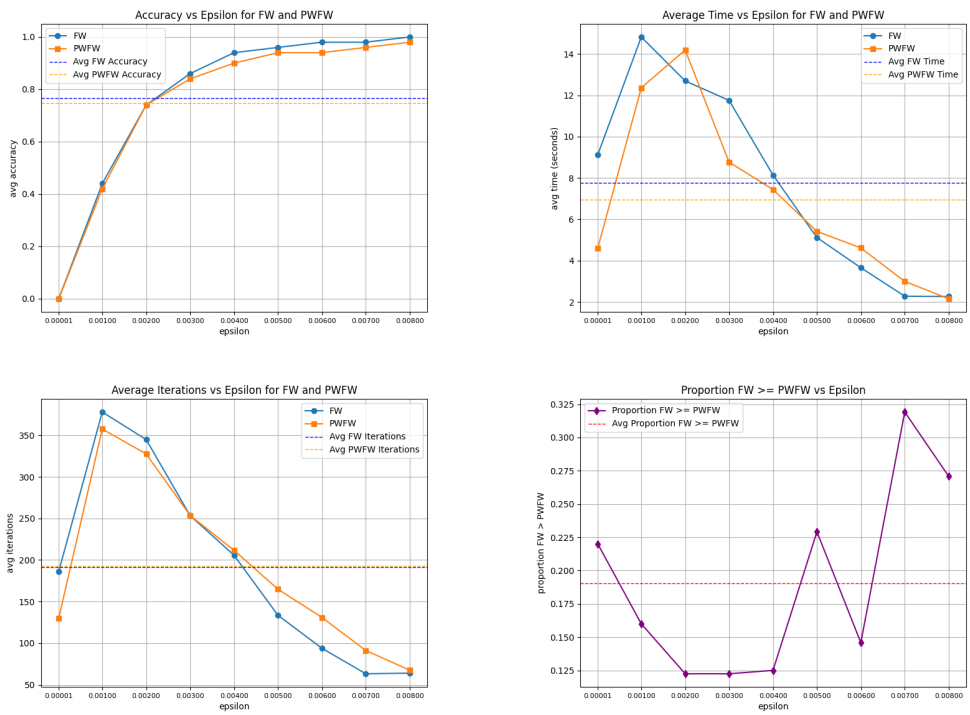

Figure 5: Audio Domain - Epsilon-Based Attack

5.2.2 Single-Digit Attack

PWFW underperforms on all digits except 0, 1 and 5, and is particularly ineffective against digit 2 (0.08 success rate difference). Both methods achieve their highest success rates on digits 2 and 9 (above 0.58), while the most challenging digit to attack is 0, with a success rate of only 0.16. Overall, the global success rate is slightly higher for FW (0.45) compared to PWFW (0.42).

In terms of efficiency, PWFW consistently requires fewer iterations, averaging 30 fewer iterations per attack compared to FW. Additionally, PWFW is faster, concluding attacks 1.91 seconds earlier on average. However, digits 0 and 1 require significantly more time to attack relative to their respective global averages for both methods.

Despite PWFW's efficiency advantages, it finds lower objective function values in only 9.3% of the attacks.

The results of the single-digit attack in the audio domain are visualized in Figure 6.

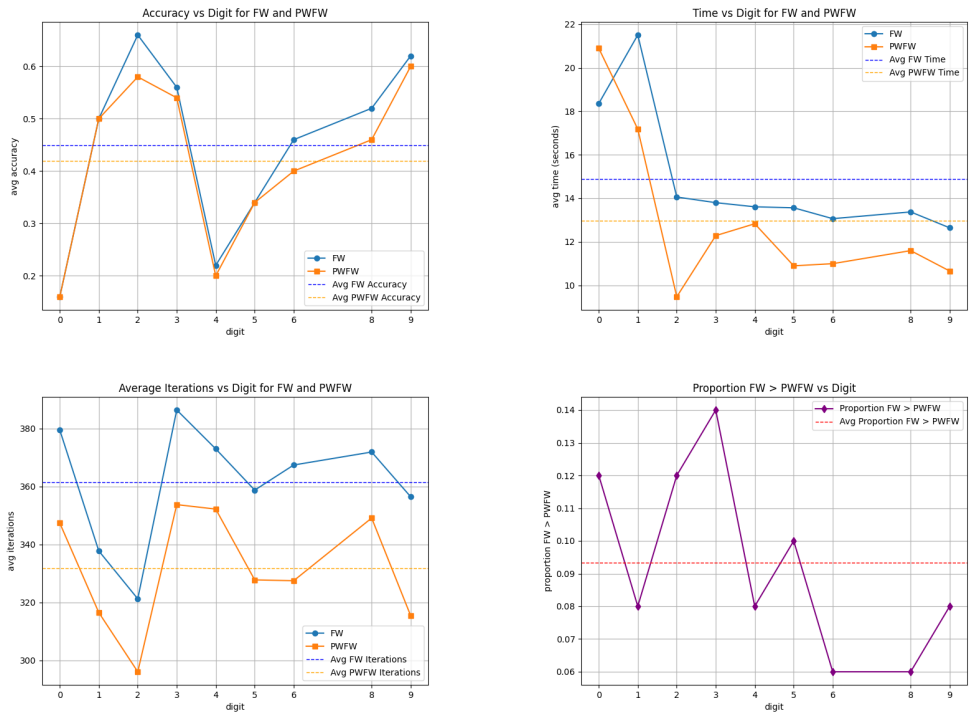

Figure 6: Audio Domain - Single-Digit Attack

6. Discussion

The choice between FW and PWFW depends on the specific task requirements. FW is preferable when achieving high-quality adversarial examples is critical, whereas PWFW is better suited for scenarios where computational efficiency is paramount.

In the image domain, PWFW consistently requires fewer iterations and executes faster while achieving comparable accuracy to FW. This makes it the preferred method unless generating high-quality adversarial examples is a priority. However, the performance difference varies across digits. For instance, FW performs significantly better for digit 1, suggesting that a hybrid approach, where the choice of method is digit-dependent, could be beneficial.

In the audio domain, FW generally produces superior adversarial examples and achieves a higher success rate. However, PWFW is computationally more efficient, particularly for smaller ϵ values. This suggests that PWFW might be preferable in real-time or resource-

constrained scenarios, while FW remains the best option when attack quality is the primary objective.

Future work could explore adaptive hybrid approaches that dynamically select FW or PWFW based on the target instance. Additionally, further investigations into why FW produces superior adversarial examples and why PWFW is computationally more efficient could provide insights into improving both methods.

7. Conclusion

This project was performed a comparative analysis of Frank-Wolfe (FW) and Pairwise Frank-Wolfe (PWFW) algorithms for generating adversarial attacks across two distinct domains: image (MNIST) and audio (Speech Commands) datasets. This research provides several key insights into the performance characteristics of these optimization techniques:
Domain-Specific Performance: The algorithms demonstrated markedly different behaviors across image and audio domains. In the image domain, PWFW showed superior computational efficiency, while in the audio domain, FW produced higher-quality adversarial examples.
Computational Efficiency: PWFW consistently required fewer iterations and shorter execution times, particularly for smaller perturbation budgets. This makes PWFW an attractive option for resource-constrained environments.
Attack Quality: FW generally produced more effective adversarial examples, especially in the audio domain, with higher success rates and lower objective function values.
The research demonstrates that no single algorithm is universally superior, and the choice between FW and PWFW depends on specific task requirements and performance priorities.

References

[1] Immanuel. M. Bomze, Francesco Rinaldi, and Damiano Zeffiro. Frank-wolfe and friends: a journey into projection-free first-order optimization methods, 2021.

[2] Wei Dai, Chia Dai, Shuhui Qu, Juncheng Li, and Samarjit Das. Very deep convolutional neural networks for raw waveforms, 2016.

[3] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum, 2018.

[4] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives, 2016.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Purdue University. Chapter 3 - adversarial attack. https://engineering.purdue.edu/ChanGroup/ECE595/files/chapter3.pdf.

[7] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018.