Adversarial Defense(Robust Training) as MiniMax Optimization

William C. Loe
M.S. in Applied Machine Learning
University of Maryland, College Park
wloe@umd.edu

Sneha Vellelath
M.S. in Applied Machine Learning
University of Maryland, College Park
velsneha@umd.edu

Nithin Skantha Murugan
M.S. in Applied Machine Learning
University of Maryland, College Park
nithin10@umd.edu

Narendra Rajendra Rane
M.S. in Applied Machine Learning
University of Maryland, College Park
nrane@umd.edu

Abhinav Kumar
M.S. in Applied Machine Learning
University of Maryland, College Park
ak395@umd.edu

***Abstract*—Adversarial examples pose critical risks to deep learning systems in domains such as autonomous vehicles, healthcare, and finance. We frame robust training as a saddle-point minimax optimization under ℓ_∞-norm constraints and systematically compare five inner-max solvers—Projected Gradient Descent (PGD), Mirror Descent, Frank–Wolfe, Augmented Lagrangian, and a novel Composite scheme. Each solver is integrated into a unified training pipeline on CIFAR-10 using a ResNet-50 backbone. Our results show that while single-strategy defenses can excel against one attack, they often degrade under others; by contrast, the Composite solver achieves the best overall trade-off, maintaining approximately 80 percentage clean accuracy and ≈60 percentage robust accuracy across all five attack types. Confidence calibration further confirms the Composite model's well-peaked softmax outputs on clean inputs. These findings highlight the value of mixed-strategy adversarial training for broad-spectrum robustness.**

***Index Terms*—Adversarial Defense, Robust Training, Minimax Optimization, Projected Gradient Descent, Neural Networks, Machine Learning Security, Model Robustness, ℓ_p-norm Perturbations**

I. Introduction

Machine learning models, particularly deep neural networks, have achieved significant breakthroughs across multiple domains, including image classification, natural language processing, and autonomous systems. However, these systems remain highly susceptible to adversarial examples—inputs intentionally altered in subtle yet harmful ways to trigger incorrect predictions. Even minor and often imperceptible perturbations can substantially degrade performance, raising considerable concerns regarding the deployment of AI technologies in safety-critical contexts.

To address these vulnerabilities, adversarial defense, commonly known as robust training, modifies the standard training procedure by incorporating adversarially manipulated examples. Instead of training solely on clean data, robust training integrates examples crafted to simulate worst-case perturbations. This approach leads to a robust optimization strategy formulated as a minimax problem, aiming to minimize the model's worst-case loss over a defined set of allowable perturbations.

Specifically, the inner maximization phase identifies the most damaging adversarial perturbation within an ℓ_p-norm bounded region around each input, while the outer minimization phase updates model parameters to correctly classify these adversarially generated examples. Practically, iterative attack techniques such as Projected Gradient Descent (PGD) are employed for solving the inner optimization, with standard optimization algorithms like stochastic gradient descent guiding the outer parameter updates.

This paper introduces an implementation of adversarial training following this minimax approach and evaluates its effectiveness in enhancing model robustness. Experimental results under various adversarial attack conditions confirm that robust training significantly strengthens the model's capacity to maintain accurate predictions despite adversarial perturbations.

II. Related Work

This section reviews key works that inform our approach to adversarial training. We focus on methods that address robustness under ℓ_∞-bounded perturbations. comparisons for standard adversarial training with alternative formulations, optimization strategies, and attack-free defenses. Each work provides unique insights into improving robustness, and collectively, they help contextualize the design choices behind our implementation.

A. Adversarial Training with PGD

Madry *et al.* [1] frame adversarial robustness as a saddle-point optimization problem: the model is trained to minimize the worst-case loss over small, norm-bounded perturbations. This min-max formulation captures the intuition that models should remain accurate even on adversarially perturbed inputs. To approximate the inner maximization, the authors introduce a multi-step *Projected Gradient Descent (PGD)* attack that iteratively perturbs the input in the direction of increasing loss, then projects it back into the ℓ_∞-ball around the original

image. This citation also shown that PGD is a strong first-order adversary and is used during training to generate worst-case examples in each batch. The result is a model that learns to classify robustly even under adaptive, high-strength attacks. In our project, we build on this work by re-implementing their PGD-based adversarial training procedure to defend against ℓ_∞-bounded perturbations.

B. Frank-Wolfe Adversarial Training

Tsiligkaridis and Roberts [2] propose a more efficient approach to adversarial training by using Frank-Wolfe (FW) optimization instead of Projected Gradient Descent (PGD). Their method solves the same ℓ_∞ minimax problem but avoids per-step projection. Each attack step moves in the direction of the signed gradient and forms convex combinations with previous steps. This projection-free strategy introduces a natural metric called *distortion*, which helps measure the flatness of the loss surface. High distortion signals stronger robustness, while low distortion can reveal overfitting—especially in single-step training. The authors extend this idea with FW-AT-Adapt, an adaptive algorithm that increases attack steps when distortion drops. Their method improves robustness over PGD and reduces computation. In our work, we adopt the same minimax objective but use PGD. The FW framework still offers useful insights into the stability of adversarial training.

C. TRADES: Balancing Robustness and Accuracy

Zhang *et al.* [3] introduce TRADES, a defense method that explicitly balances natural accuracy and adversarial robustness. Unlike standard PGD-based training, which minimizes worst-case loss directly, TRADES uses a regularized surrogate loss. This loss combines the standard cross-entropy on clean inputs with a KL divergence term that penalizes changes in model predictions under perturbation. A tunable hyperparameter controls the balance between accuracy and robustness. TRADES comes with theoretical guarantees that this loss upper-bounds the adversarial risk. The method improves robustness while preserving more clean accuracy. Its core idea aligns with our project's objective in P-1: enhancing robustness under ℓ_∞ constraints by refining the training loss.

D. MACER: Attack-Free vs. Attack-Based Robust Training

Zhai *et al.* [4] propose MACER, a training method that improves adversarial robustness *without using adversarial examples*. Unlike PGD-based methods that rely on solving a min–max optimization, MACER trains a smoothed classifier by adding Gaussian noise to inputs. It then maximizes the model's *certified radius*, which defines a region where predictions are provably stable. This approach provides formal robustness guarantees and is faster to train. Results show that MACER performs as well as or better than PGD-based methods in terms of certified robustness. These findings suggest that attack-free training can be both effective and efficient. Elements of MACER, such as its margin-aware loss, could be combined with attack-based methods to further improve robustness.

III. Mathematical Model

The primary goal of adversarial training is to enhance model robustness by minimizing the maximum potential classification loss due to small, norm-bounded perturbations in the input data. Formally, this approach can be expressed as a nested minimax optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}\left[\max_{\|\delta\|_p\le\epsilon,\ x+\delta\in[0,1]^n} L(f_\theta(x+\delta),y)\right] \quad (1)$$

where:

- θ represents the model parameters,
- $(x, y) \sim D$ denotes input-label pairs drawn from the data distribution D,
- δ is an adversarial perturbation constrained by $\|\delta\|_p \le \epsilon$,
- f_θ is the model or classifier,
- L indicates the loss function, typically cross-entropy,
- and the constraint $x + \delta \in [0, 1]^n$ ensures that the perturbed inputs remain valid and realistic.

The inner maximization aims to identify the perturbation δ that maximizes the classification loss for each input, effectively simulating the most harmful adversarial scenario. This step is commonly approximated using iterative attack methods such as Projected Gradient Descent (PGD). Subsequently, the outer minimization updates the model parameters θ to reduce the expected worst-case loss across the dataset, typically employing stochastic gradient descent (SGD).

Overall, this formulation ensures the trained model maintains high accuracy on both clean and adversarially perturbed inputs, significantly improving reliability and robustness in practical, real-world deployments.

IV. Methodology

We cast adversarial defense as the following minimax optimization problem:

$$\min_{\theta}\ \mathbb{E}_{(x,y)\sim D}\Big[\max_{\substack{\|\delta\|_\infty\le\epsilon\\ x+\delta\in[0,1]^n}} L\big(f_\theta(x+\delta),y\big)\Big]. \quad (2)$$

Here f_θ is a ResNet-50 classifier, L is cross-entropy, and $\epsilon = 8/255$.

A. Outer Minimization

We approximate (2) by alternating:

- **Inner** (adversarial example generation): solve $\max_{\|\delta\|_\infty\le\epsilon} L(f_\theta(x+\delta),y)$ via one of five solvers.
- **Outer** (parameter update): perform a gradient step

$$\theta \leftarrow \theta - \eta\,\nabla_\theta L\big(f_\theta(x+\delta^*),y\big)$$

with SGD, momentum 0.9, weight decay 5×10^{-4}, initial $\eta = 0.1$, decayed at epochs 100 and 150 over 200 epochs.

B. Inner Maximization Solvers

All solvers run for $T = 10$ iterations with per-solver step-sizes α tuned by grid search.

1) Projected Gradient Descent (PGD):

$$\delta_0 \sim \mathcal{U}(-\epsilon, \epsilon), \quad \delta_{t+1} = \Pi_{\|\delta\|_\infty \le \epsilon}\big(\delta_t + \alpha\,\mathrm{sign}(\nabla_\delta L)\big),$$

where Π denotes element-wise clipping.

2) Mirror Descent: Using the negative Shannon entropy as mirror map, we maintain dual variable u_t with

$$u_{t+1} = u_t + \alpha\,\nabla_\delta L(f_\theta(x+\delta_t), y), \quad \delta_{t+1} = \Pi_{\|\cdot\|_\infty \le \epsilon}\big(\nabla\Phi^*(u_{t+1})\big)$$

where Φ^* is the convex conjugate of the entropy regularizer.

3) Frank–Wolfe: At each step,

$$s_t = \arg\max_{\|s\|_\infty \le \epsilon} \langle s, \nabla_\delta L\rangle \;=\; \epsilon\,\mathrm{sign}(\nabla_\delta L),$$

$$\gamma_t = \tfrac{2}{t+2}, \quad \delta_{t+1} = (1-\gamma_t)\,\delta_t + \gamma_t\, s_t.$$

4) Augmented Lagrangian: Define

$$c(\delta) = \max\big(\|\delta\|_\infty - \epsilon,\, 0\big), \quad \mathcal{L}(\delta, \lambda) = L + \lambda\, c(\delta) + \tfrac{\mu}{2}\, c(\delta)^2.$$

We alternate:

$$\delta_{t+1} = \Pi_{\|\cdot\|_\infty \le \epsilon}\big(\delta_t + \alpha\,\nabla_\delta \mathcal{L}\big), \quad \lambda_{t+1} = \max\big(\lambda_t + \mu\, c(\delta_{t+1}),\, 0\big).$$

5) Composite: We form a one-step composite loss

$$L_{\mathrm{comp}} = w_1 L_{\mathrm{pgd}} + w_2 L_{\mathrm{md}} + w_3 L_{\mathrm{fw}} + w_4 L_{\mathrm{al}},$$

compute its gradient w.r.t. δ, and update via PGD:

$$\delta_{t+1} = \Pi_{\|\delta\|_\infty \le \epsilon}\big(\delta_t + \alpha\,\mathrm{sign}(\nabla_\delta L_{\mathrm{comp}})\big).$$

C. Implementation Details

- **Architecture:** ResNet-50 modified for CIFAR-10 (3×3 first conv, no initial pooling).
- **Batch size:** 128.
- **Preprocessing:** Random crop & flip for training; center crop for testing; standard CIFAR-10 normalization.
- **Hardware:** Single GPU training; inner-loop adds $\times T$ overhead per epoch.

D. Evaluation Metrics

- **Clean Accuracy**: classification accuracy on the unperturbed test set.
- **Robust Accuracy**: classification accuracy under each solver's attack (evaluated on test set).
- **Confidence Calibration**: distribution of maximum softmax probabilities on clean inputs, indicating model confidence.

V. Results

Table I presents the clean and adversarial accuracies for each of the six ResNet-50 defenses across six attack types. We also visualize the same data in Figure 1 (grouped bar chart) and the confidence distributions on clean data in Figure 2.

Across all six defenses, the Composite-trained model achieves the most consistent performance: it retains high clean accuracy (79.75%) while maintaining uniformly strong robustness (60%) under every attack. Figure 1 clearly shows that, unlike single-strategy defenses which may excel against one attack but falter on others, the Composite approach yields balanced, high bars in every column. Furthermore, the confidence score histogram in Figure 2 demonstrates that Composite-trained predictions are sharply peaked near 1.0 on clean inputs—indicative of well-calibrated, reliable confidence (comparable only to the Augmented Lagrangian defense).

Taken together, these results justify our selection of the Composite solver as the preferred robust training strategy: it maximizes worst-case performance across a diverse suite of adversarial attacks without sacrificing clean-data accuracy.

TABLE I
Accuracy (%) of Each Defense under Various Attacks

Defense \ Attack	Clean	PGD	Mirror	FW	AL	Composite
Clean-only	83.03	25.84	48.28	58.88	48.08	48.19
PGD-trained	78.50	41.01	45.98	49.09	45.99	46.01
Mirror-trained	79.10	32.19	40.46	42.53	40.49	40.43
FW-trained	76.85	5.31	33.92	70.11	33.56	33.80
AL-trained	80.35	29.48	59.99	60.93	59.74	60.05
Composite-trained	**79.75**	**28.51**	**60.15**	**60.91**	**60.07**	**60.04**

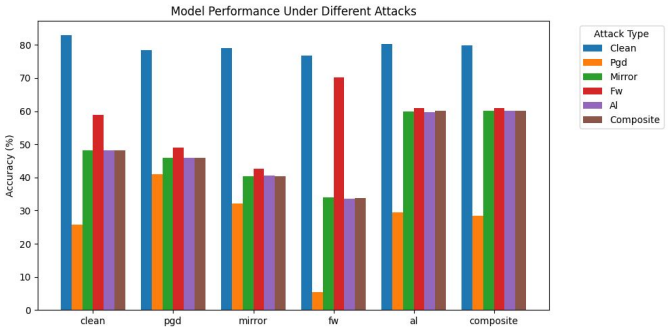

Fig. 1. Model Performance Under Different Attacks Grouped bar chart showing the clean-test accuracy (blue) and robustness under five adversarial attacks (PGD, Mirror Descent, Frank–Wolfe, Augmented Lagrangian, Composite) for each ResNet50 model trained with a different inner-max solver. Robustly trained models—especially those using Augmented Lagrangian and Composite losses—maintain substantially higher accuracy under attack compared to the clean-only baseline.

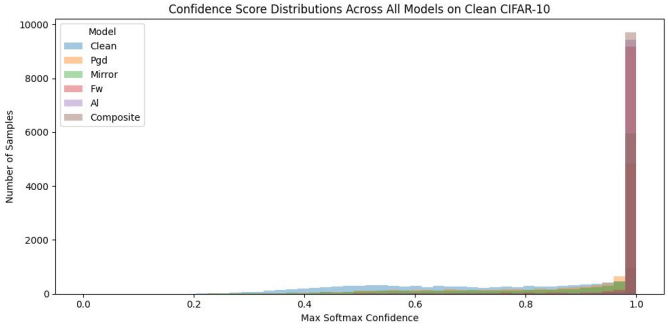

Fig. 2. Confidence Score Distributions for All Trained Models on Clean Data" Overlaid histograms of maximum softmax confidence on the CIFAR-10 test set for each ResNet50 model—trained on clean data and under five adversarial defense solvers (PGD, Mirror Descent, Frank–Wolfe, Augmented Lagrangian, Composite). Models with stronger robustness (AL, Composite) exhibit tighter confidence distributions, indicating more calibrated and reliable predictions even before attack.

VI. Challenges

Robust adversarial training presents several practical and theoretical challenges. Below we outline the key difficulties encountered in our study.

a) High Computational Overhead: Multi-step inner-max solvers such as PGD and Augmented Lagrangian multiply the per-epoch cost by the number of attack iterations ($T = 10$ in our experiments). On high-resolution datasets (e.g. ImageNet), this overhead becomes prohibitive even on large GPU clusters. Moreover, Composite training—which computes four separate losses per step—incurs roughly four times the cost of PGD alone. Reducing this overhead without compromising robustness remains an open problem.

b) Hyperparameter Sensitivity: Each solver requires carefully tuned step-sizes, restart strategies, and penalty coefficients (for AL). We found that small changes in α or μ can cause inner maximization to either underfit (weak attacks) or diverge (overly strong updates). This sensitivity complicates reproducibility and makes it difficult to recommend "default" settings that work across architectures and datasets.

c) Trade-Off Between Clean and Robust Accuracy: Adversarial training inherently balances two objectives: maintaining high clean-data performance and achieving strong robustness. In our results, the PGD-trained model maximizes PGD accuracy but suffers a 5% drop in clean accuracy compared to the clean-only baseline. Composite training narrows this gap but still yields a 3–4% trade-off. Understanding, quantifying, and minimizing this trade-off is a major theoretical and empirical challenge.

d) Limited Threat Model and Dataset Scope: Our work focuses exclusively on ℓ_∞-bounded perturbations on CIFAR-10. Real-world adversaries may exploit other norms (e.g. ℓ_2), spatial transformations, or even physically realizable attacks (e.g. adversarial stickers). Moreover, defenses that succeed on small benchmarks often fail on larger or more diverse datasets. Extending evaluations to multiple threat models and real-world scenarios is essential.

e) Certification vs. Empirical Robustness: While our solvers yield strong empirical robustness, they provide no formal guarantees. Certified defenses (e.g. MACER, randomized smoothing) can offer provable bounds on adversarial risk but often at the cost of lower empirical accuracy or greater complexity. Bridging the gap between certified and empirical methods—combining their strengths—remains an active research frontier.

VII. Future Work

Based on these challenges, we identify several promising directions for subsequent research.

a) Adaptive and Single-Step Methods: Develop adaptive iteration schedules—where the number of inner-max steps varies per sample or epoch—to reduce compute without sacrificing robustness. Investigate single-step or momentum-based approximations (e.g. FGSM with momentum) as efficient alternatives to multi-step attacks.

b) Broader Attack Taxonomy: Benchmark defenses against a wider range of adversaries, including ℓ_2 and Wasserstein-norm attacks, decision-based attacks (CW, DeepFool), spatial transformations, and unrestricted perturbations. Such studies will reveal which solvers generalize best across threat models.

c) Integration with Certified Defenses: Explore hybrid training schemes that combine solver-based adversarial training with certified robustness approaches (e.g. MACER's margin-aware loss or randomized smoothing). Alternating certified and empirical training phases may yield models that are both provably and practically robust.

d) Robust Feature Learning and Interpretability: Analyze the geometry of decision boundaries and the structure of features learned under different solvers. Understanding which features are most resistant to perturbation could inform new architecture designs and improve interpretability of robust models.

e) Transfer Learning and Domain Adaptation: Study how adversarially pre-trained backbones transfer to downstream tasks (e.g. object detection, segmentation) and different domains. Robust models often overfit to their original dataset; developing fine-tuning methods that preserve robustness is critical for real-world deployment.

f) Scalable, Hardware-Aware Defenses: Address deployment constraints such as inference latency, memory footprint, and energy consumption. Investigate quantization, pruning, and hardware-aware attack algorithms to enable robust models on edge devices and real-time systems.

Contributions

The project was a collaborative effort by all five team members. Each member contributed to the design, implementation, and analysis of robust training strategies under ℓ_∞ constraints. The final report was jointly written and reviewed by the entire team.

- **Abhinav**: Led the implementation of the Mirror Descent optimizer. Assisted in modularizing the trainer for solver dispatch. Contributed to data preparation and debugging support during attack integration.
- **Narendra**: Designed the composite loss strategy that combines multiple solvers. Contributed to configuration management, including per-solver hyperparameters. Participated in evaluation planning and analysis.
- **Nithin**: Implemented the PGD attack baseline and led the development of the unified training pipeline. Refactored `trainer.py`, implemented `main.py` orchestration, and managed clean/robust model saving. Led CIFAR-10 data loading, transforms, and visualization helpers.
- **Sneha**: Developed the Augmented Lagrangian attack module and led debugging efforts for loss aggregation and numerical stability. Generated confusion matrices and grouped bar plots for evaluation. Led the design and creation of the final presentation.
- **Will Loe**: Implemented the Frank-Wolfe optimizer and contributed to attack module design. Led evaluation au-

tomation, visualizations (e.g., adversarial samples), and contributed to CUDA integration and device management. Supported model refactoring from ResNet-18 to ResNet-50.

VIII. Code Repository

The complete implementation, including data preparation, model training scripts, and evaluation procedures used in this study, is publicly available at the GitHub repository linked below. This repository provides all necessary code and resources required to reproduce the results presented in this paper.

github.com/willloe/MSML604-Final_Project

References

[1] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1706.06083

[2] T. Tsiligkaridis and J. Roberts, "Understanding and Increasing Efficiency of Frank-Wolfe Adversarial Training," *arXiv preprint arXiv:2012.12368*, 2020. [Online]. Available: https://arxiv.org/abs/2012.12368

[3] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan, "Theoretically Principled Trade-off between Robustness and Accuracy," *arXiv preprint arXiv:1901.08573*, 2019. [Online]. Available: https://arxiv.org/abs/1901.08573

[4] R. Zhai, C. Dan, D. He, H. Zhang, B. Gong, P. Ravikumar, C.-J. Hsieh, and L. Wang, "MACER: Attack-Free and Scalable Robust Training via Maximizing Certified Radius," *arXiv preprint arXiv:2001.02378*, 2020. [Online]. Available: https://arxiv.org/abs/2001.02378