People's Democratic Republic of Algeria
Ministery of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Computer Science
Department of Computer Science

DISSERTATION

Presented in fulfillment of the requirements of obtaining the degree

Master in Computer Science

Specialty:Business Intelligence Engineering

THEME

Adversarial Attacks And Defense Mechanisms In Deep Learning

Presented by:

SAIDANI ALA

KHOUDOUR MERIEM ANFEL

Publicly defended on: 10/06/2025

In front of the jury composed of:

President: Dr.Saifi Abdelhamid

Examiner: Dr.Belhadj Foudil

Supervisor: Dr. Beghoura Mohamed Amine

2024/2025

Dedication

To my loving parents and dear siblings Thank you for being my unwavering support system. Your sacrifices, love, and constant encouragement have shaped the person I am today. This achievement is as much yours as it is mine.

To my respected supervisor, Mr. Mohamed Amine Beghora Thank you for your valuable guidance, insightful feedback, and continued support throughout this journey. Your mentorship has been a key part of my academic growth.

To my precious best friends, Fatima and Meriem Anfel Thank you for always being by my side through the highs and the lows.

Thank you for your laughter, loyalty, and for reminding me I was never alone.

Your friendship, loyalty, and endless encouragement made every step easier.

And most importantly..

Last but not least, I wanna thank me.

I wanna thank me for believing in me. I wanna thank me for doing all this hard work. I wanna thank me for having no days off. I wanna thank me for never quitting. I wanna thank me for tryna do more right than wrong. I wanna thank me for just being me at all times.

Ala

So, here we are done and dusted (finally!). Honestly, I’m still amazed we made it through without losing our minds.
To my friends Ala, Fatima, Lydia and all my girlies thanks for being the perfect mix of support and distraction.
And a special shoutout to my dad, who somehow pretended to understand my endless ramblings and still managed to believe in me.
If this thesis taught me anything, it’s that procrastination is an art form. . . but hey, we finished! Now, onto the next adventure preferably one with less stress and more sleep.

Meriem Anfel

Abstract

This work explores the adversarial vulnerabilities of deep learning models in image classification, with a focus on evaluating and defending against evasion-based attacks. Using the MNIST dataset and a ResNet18 architecture, we implemented several notable adversarial attacks, including FGSM, PGD, Clean Label, Backdoor (BadNet), and Square Attack.

To mitigate these threats, we applied a variety of defense mechanisms across three categories: preprocessing (Gaussian noise, bit-depth reduction, JPEG compression), training-based (adversarial training, label smoothing), and postprocessing (confidence thresholding, randomized smoothing). Evaluation was conducted using standard performance metrics and qualitative visualizations.

The results confirm the effectiveness of adversarial training and hybrid approaches in enhancing model robustness. This work provides a reproducible framework and contributes to ongoing efforts toward secure and resilient deep learning systems.

Keywords: Deep learning, adversarial attacks, model robustness, image classification, adversarial training, defense mechanisms.

Résumé

Ce travail explore les vulnérabilités adversariales des modèles d'apprentissage profond en classification d'images, avec un accent particulier sur l'évaluation et la défense contre les attaques de type évasion. En utilisant le jeu de données MNIST et l'architecture ResNet18, nous avons mis en œuvre plusieurs attaques adversariales notables, notamment FGSM, PGD, Clean Label, Backdoor (BadNet) et Square Attack.

Pour atténuer ces menaces, nous avons appliqué divers mécanismes de défense répartis en trois catégories : prétraitement (bruit gaussien, réduction de profondeur de bits, compression JPEG), entraînement (entraînement adversarial, lissage des étiquettes), et post traitement (seuil de confiance, lissage aléatoire). L'évaluation a été réalisée à l'aide de métriques standard de performance et de visualisations qualitatives.

Les résultats confirment l'efficacité de l'entraînement adversarial et des approches hybrides pour renforcer la robustesse des modèles. Ce travail propose un cadre reproductible et contribue aux efforts en cours pour le développement de systèmes d'apprentissage profond sûrs et résilients.

Mots-clés : Apprentissage profond, attaques adversariales, robustesse des modèles, classification d'images, entraînement adversarial, mécanismes de défense.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

1.1 Context

In recent years, deep learning has revolutionized the field of artificial intelligence (AI), particularly in areas such as image classification, natural language processing, and speech recognition. Despite these successes, deep neural networks remain fragile when exposed to subtle and carefully crafted perturbations known as adversarial examples. These examples can cause misclassifications with high confidence, even though the modifications are imperceptible to the human eye. This vulnerability presents a critical challenge, especially in security-sensitive applications like autonomous vehicles, biometric authentication, and medical diagnostics.

1.2 Problem Statement

The vulnerabilities discussed in this study are not merely theoretical they can actually be exploited through adversarial attacks, which capitalize on the structural and behavioral weaknesses of deep neural networks. Such vulnerabilities have significant implications for the safety, trustworthiness, and security of deep learning applications, particularly in high risk situations. Therefore, it is critical to understand how these adversarial strategies are perpetrated in order to develop strong and defensible systems.

The lack of robustness in deep learning models highlights a pressing need to systematically investigate the nature of adversarial threats and implement effective countermeasures. Existing

models, although accurate under normal conditions, often fail under adversarial pressure. This discrepancy raises fundamental questions about the reliability of these models and the adequacy of current defense mechanisms.

1.3 Objectives of the Study

The objective of this study is to implement a structured experimental approach that tests, analyzes, and strengthens the security of image classification models, particularly using the MNIST dataset. We aim to simulate a range of adversarial attacks and apply multiple defense techniques in order to assess their effectiveness under different threat scenarios. Through this, we seek to provide insights into the robustness of commonly used architectures and explore practical defense strategies that enhance model reliability in adversarial environments.

1.4 Structure of the Thesis

This thesis is structured into main chapters as follows:

- **Chapter 2: Deep Learning Fundamentals and Vulnerabilities** — This chapter provides a foundational overview of deep learning, covering essential concepts such as neural networks, activation functions, and training mechanisms. It also delves into the inherent weaknesses of deep learning models, such as overfitting, high-dimensional input spaces, and sensitivity to input perturbations, which contribute to their susceptibility to adversarial manipulation.
- **Chapter 3: Adversarial Attacks and Defense Mechanisms** — This chapter classifies and explains different types of adversarial attacks including evasion, poisoning, backdoor, and black-box attacks. It also discusses threat models (white-box vs. black-box), and presents a wide range of defense techniques, categorized into preprocessing, training-time, postprocessing, and certified defenses, along with references to relevant work in the literature.

- **Chapter 4: Methodology** — This chapter describes the experimental framework, starting with the MNIST dataset and the architecture of the adapted ResNet18 model. It then details the implementation of five adversarial attacks (FGSM, PGD, Clean Label, BadNet, Square) and various defense strategies applied at different stages. A global schematic of the experimental workflow is also presented.

- **Chapter 5: Experimental Results and Evaluation** — This chapter presents the quantitative and qualitative results of the experiments. It includes evaluations of attack severity, the effectiveness of each defense mechanism, and their performance under different threat settings. Results are supported by accuracy, precision, recall, F1-score, confusion matrices, success rates, and visualizations such as bar charts and prediction comparisons.

- **Chapter 6: Conclusion and Future Work** — The final chapter synthesizes the main findings of the research, discusses the strengths and limitations of the proposed approach, and outlines potential directions for future work. This includes suggestions for improving robustness, testing on more complex datasets or architectures, and applying certified defenses in real-world applications.

Chapter 2

Deep Learning Foundations and Model Vulnerabilities

2.1 Introduction

Deep learning has transformed artificial intelligence by allowing models to learn patterns directly from hundreds of thousands of examples of raw data. These models have demonstrated Extraordinary success for problems including: image classification, speech recognition, and natural language processing. However, despite their impressive performance, deep learning models exhibit a critical weakness in the presence of small perturbations, known as adversarial examples. In this chapter, we covers the theoretical background needed to understand how deep learning models work, and thus why they can be influenced by adversarial attacks, and will act as an introduction and grounding for the adversarial attacks that we will discuss.

2.2 Overview of Deep Learning

2.2.1 Definition and Scope

Deep learning is a subfield of machine learning that utilizes deep neural networks to automatically learn and identify patterns in data. Traditional systems required hand engineered features and use simple neural networks with one or two computational layers while deep learning models use multilayered neural networks to train the models that enable the modeling of highly

abstract and intricate data relationships and automatically learn representations of features from raw input uniquely. This is one of the factors that makes deep learning especially compelling for domains such as image analysis, natural language processing, and speech recognition. Deep learning has played an important part of the growth of artificial intelligence and has allowed a system to achieve human-level performance on tasks that humans have yet to solve. [1]

2.2.2 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are machine learning algorithms inspired by the central nervous system. They were first conceived in 1943 when McCulloch and Pitts published a study presenting the mathematical model inspired by the structure and functioning of the human brain. They consist of interconnected nodes analogous to neurons organized into layers, and operate using inputs, weights, and biases to process data and make predictions. Each node applies an activation function to the weighted sum of its inputs and passes the result to the next layer. [2]

Structure of a Neuron Artificial neurons receive input signals, calculate each with appropriate weight, add a bias term, and put through an activation function. The general output is sent to the neurons in the next layer.

Types of Layers : There are three primary types of layers in a neural network:

- **Input Layer**: Receives raw data (e.g., image pixels, word tokens) and sends it to the network.
- **Hidden Layers**: Perform intermediate computations and feature extraction. Deep neural networks (DNNs) include multiple hidden layers, making them capable of modeling high-level abstractions.
- **Output Layer**: Produces the final result (e.g., classification, regression output), depending on the specific task.

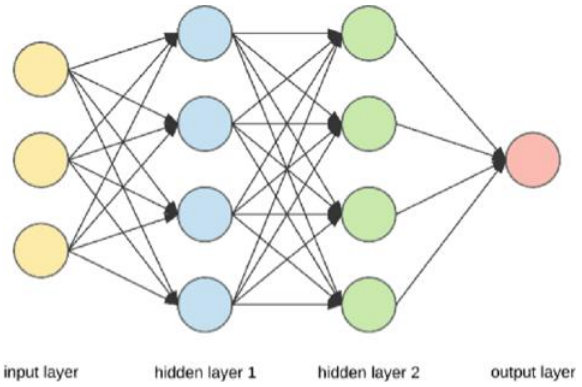


Figure 2.1: Illustration of input, hidden, and output layers in a deep learning model [1].

2.3 Network Architectures

Throughout the process of developing deep learning models for classifying images, many well established architectures were developed. The two most prominent examples are LeNet and ResNet as they had the largest impact on the progression of convolutional neural network (CNN) designs.

Convolutional Neural Networks Convolutional Neural Networks (CNNs) [3] are deep learning models designed to process data with a grid like topology such as images. They are the foundation for most modern computer vision applications, built upon several key components that work together to detect features within visual data :

- **Convolutional Layers:** These layers apply convolutional operations to input images using filters or kernels to detect features such as edges, textures and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.
- **Pooling Operations:**They down sample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. Max pool- ing is a common pooling operation where we select a maximum value from a group of neighboring pixels.
- **Fully Connected Layers:**These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.

.

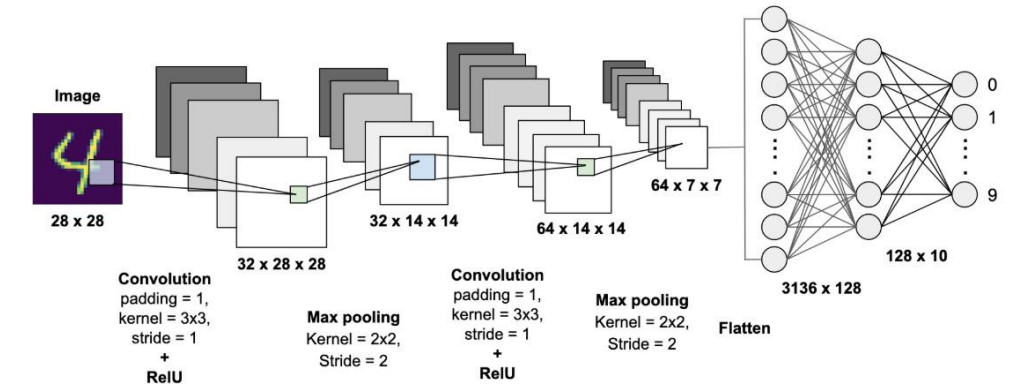


Figure 2.2: Convolutional Neural Network (CNN) architecture from input image to final prediction[2].

ResNet Residual Networks (ResNets), introduced by He et al. in 2015[4], were designed to address the degradation problem that arises when training very deep networks. The key innovation in ResNet is the use of residual blocks with shortcut (skip) connections, which allow the model to learn residual mappings instead of direct transformations. This approach enables the effective training of much deeper networks by preserving gradient flow across layers.

The general form of a residual block is:

$$\mathbf{y} = F(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

where F is the learned residual function, and $\mathbf{x}$ is the input passed through an identity shortcut.

Activation Function They introduce non-linearity to the model by allowing it to learn more complex relationships in the data. Common activation functions include:

ReLU (Rectified Linear Unit): Efficient and widely used; it outputs the input if positive, and zero otherwise.

Sigmoid: Maps input to a range between 0 and 1; useful for binary classification tasks.

ResNet18 Model Residual Networks is a lightweight variant of the ResNet architecture consisting of 18 layers, specifically designed to provide a balance between computational efficiency and learning capability. It starts with a 7x7 convolutional layer followed by a max-

pooling operation, and then passes through four residual stages—each containing two residual blocks. These blocks apply convolutional operations while maintaining identity shortcut connections to preserve gradient flow. The network concludes with a global average pooling layer and a fully connected output layer. Despite its relative simplicity, ResNet18 achieves high performance on image classification tasks and is especially suitable for datasets like MNIST where deeper models may be unnecessary or prone to overfitting.

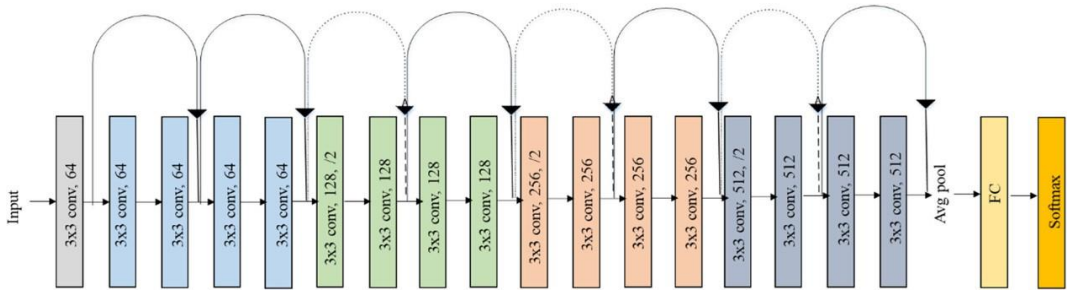


Figure 2.3: ResNet18 architecture: the network consists of a series of convolutional layers, residual (skip) connections, average pooling, a fully connected (FC) layer, and a softmax output[3].

2.4 Model Training in Deep Learning

Deep learning models analyze training data in order to learn how to produce correct outputs. The model makes adjustments to neural network weights and biases to maximize accuracy while minimizing error, until reliable outputs will exist within a range of accuracy or until they submit to predefined performance measures. The process of maximizing accuracy while minimizing error is continuous through forward propagation and backpropagation. [5]

Forward Propagation : During forward propagation, input data passes through the layers of the network one at a time. Each layer performs operations on the input to provide a higher-level representation of the input and predict output..

Backpropagation and Gradient Computation : Backpropagation is the core algorithm used to update network parameters. It calculates the gradient of the loss function with respect to each weight using the chain rule of calculus and adjusts the weights in the opposite direction of the gradient to minimize the error. This process is often combined with optimization algorithms such as:Stochastic Gradient Descent(SGD),Adam (Adaptive Moment Estimation).

Loss Function: A loss function is used as a metric to quantify the performance of the neu- ral network model on the training data, and it measures the difference between the predicted outputs and the target outputs. During training, it is the focus of our optimization procedure to minimize this loss using techniques such as gradient descent. Common loss functions include mean squared error for regression tasks and cross-entropy loss for classification tasks.

Training a deep learning model from scratch can require massive amounts of data and use large amounts of computational power. To save time and money, a foundational model can be trained for new tasks with transfer learning algorithms

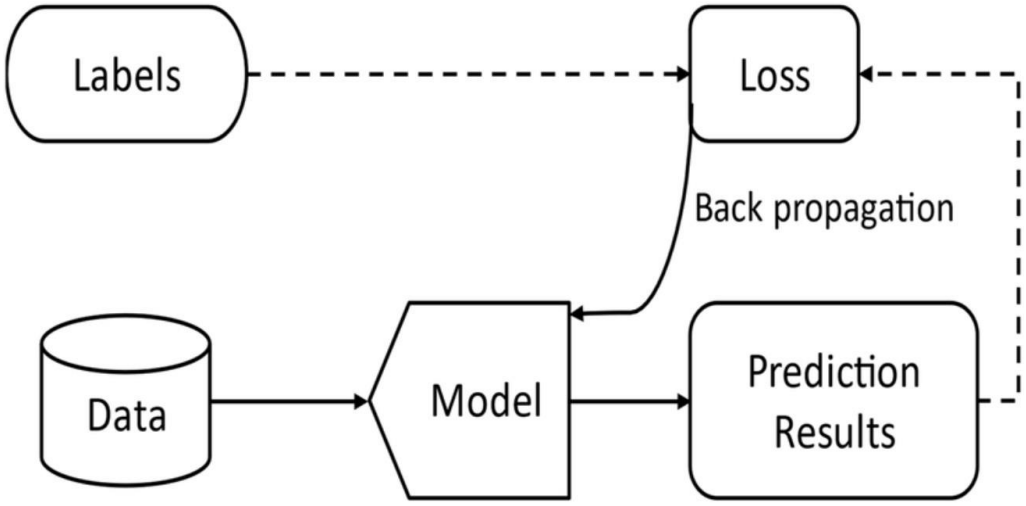


Figure 2.4: Conventional training process of a neural network model: forward pass from data to predictions, followed by backpropagation using the loss between predictions and labels.

Optimization Algorithms (Optimizers) Optimization algorithms are fundamental compo- nents in the training of deep learning models,which are essential to the process of iteratively updating model parameters to minimize the loss function. Ultimately, optimization algorithms are responsible for how the model learns from the data, peturbating weights and biases in a way that makes performance and convergence better.

- **Stochastic Gradient Descent :** is one of the most simple and common optimization methods in Deep Learning. It often updates model parameters θ by taking a step in the opposite direction of the gradient of the loss function $L(\theta)$ scaled by a learning rate η:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta L(\theta)$$

For SGD to be a stochastic method, it operates on mini-batches of data. Hence, it is stochastic in that processes randomly select a mini-batch for training and also makes it faster. But yielding stochasticity can also yield very slow convergence for large data sets and especially deep networks, and makes the training subject to local minima or saddle points.

- **Adam :** is a high-level optimizer that introduces the advantages of momentum and RMSProp, while also maintaining moving averages of the gradients and squared gradients to adapt the learning rate for every parameter. Adam is quite useful in practice for deep architectures and large datasets because it can handle a sparse gradient and noisy updates, and has been widely used as default, to train deep neural networks.

2.5 Deep Learning in Security-Critical Fields

Deep learning is a powerful form of artificial intelligence that has transformed our capabilities with state of the art accuracy in areas such as image classification, natural language processing, and anomaly detection. Deep learning's rapid expansion into critical-security-related spaces, which require reliability and resilience, allows us to make quicker and more accurate decisions in areas like cybersecurity, biometrics, health care, surveillance, and fraud detection.

By leveraging massive datasets and sophisticated architectures, deep learning models can automatically learn to detect subtle patterns, often exceeding human capabilities. However, the use of deep learning in sensitive environments also raises concerns related to adversarial robustness, interpretability, and data privacy.

This section explores key applications of deep learning in various security-critical fields, highlighting benefits, limitations, and future directions.

Cybersecurity : Deep learning models have shown strong potential in cybersecurity tasks such as intrusion detection systems (IDS), malware classification, and network traffic monitoring. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can detect anomalous or suspicious behaviors in real time without requiring constant retraining, making them superior to traditional systems in adaptability. Long Short-Term Memory (LSTM) models

further enhance this capability by modeling temporal relationships in network logs, aiding in the detection of stealthy, persistent threats.[6]

Biometric Authentication : In biometric systems, deep learning—particularly CNNs has enhanced facial recognition, fingerprint matching, and iris detection. These models extract high level features overlooked by conventional techniques, improving accuracy and robustness. Notable applications include FaceNet and DeepFace, which achieve near-human performance and are now widely deployed in mobile authentication, secure facility access, and border control systems.[7] [8]

Medical Security Systems : Deep learning is increasingly used in healthcare both for diagnostic accuracy and data security. Models such as CNNs can analyze medical images (e.g., X-rays, MRIs) to detect conditions like cancer or pneumonia. Concurrently, these models can monitor access patterns in Electronic Health Records (EHRs) to flag suspicious behavior. A prominent example is CheXNet, a deep CNN that matches radiologist performance in diagnosing pneumonia from chest X-rays.[9]

Surveillance and National Security : In surveillance, deep learning enables real time tracking and anomaly detection. Systems built on models like YOLO (You Only Look Once) can identify people and vehicles in crowded environments with high precision. These technologies are widely used in airports, public events, and border monitoring. Additionally, autonomous drones equipped with deep learning facilitate threat assessment in disaster and conflict zones.[10]

Financial Fraud Detection : Banks and financial institutions use deep learning to detect fraudulent transactions by learning patterns in historical data. Autoencoders and RNNs can identify deviations from typical behavior, flagging potential fraud. Generative Adversarial Networks (GANs) are also employed to simulate fraudulent behavior during training, improving model robustness and adaptability.[11]

The application of deep learning in sensitive security environments increases capabilities while potentially introducing new attack surfaces, and will be discussed in the next section.

2.6 Vulnerabilities of Deep Learning Models

Although deep learning has the potential to transform modern systems, its integration into real-world—especially security-critical—applications exposes a number of vulnerabilities. These vulnerabilities can be exploited to undermine the reliability, integrity, and privacy of both the model and its output. This section highlights major categories of risks that compromise the robustness of deep learning systems.

Adversarial Examples : One of the most prominent vulnerabilities is susceptibility to adversarial examples inputs intentionally crafted with subtle perturbations that mislead the model into making incorrect predictions, despite appearing normal to human observers. This is particularly dangerous in high stakes applications like facial recognition, autonomous driving, or medical diagnostics, where small errors can have severe consequences. [12]

Overfitting and Lack of Generalization : Deep learning models typically require large and diverse datasets to generalize effectively. When trained on limited, biased, or unrepresentative data, models may overfit performing well on training data but poorly on unseen examples. In high-risk domains, such fragility severely undermines trust and renders the system ineffective in novel or evolving environments.[13]

Model Interpretability and Explainability : Deep neural networks often operate as opaque "black boxes." Their predictions are not easily interpretable, which is problematic in regulated or ethical domains such as healthcare, finance, or law enforcement. Without explainability, it becomes difficult to ensure compliance, detect errors, or trace decisions after failures.[14]

Privacy Leakage and Model Inversion : Deep models can unintentionally memorize and expose private information from their training data. Techniques such as model inversion or membership inference allow adversaries to reconstruct sensitive inputs, which is particularly concerning in sectors like healthcare or legal systems, where data confidentiality is critical.[15]

2.7 Reasons for Deep Learning Fragility

Despite their success, deep learning models are known to be fragile. This fragility manifests in poor generalization, vulnerability to adversarial inputs, and difficulty diagnosing errors. Below

are key structural and theoretical factors contributing to this instability.

High-Dimensional Input Space : Deep models often process high dimensional data (e.g., images or sequences), which creates sparse and unintuitive geometries. Small perturbations in such spaces can easily traverse decision boundaries, leading to unexpected and incorrect outputs.[16]

Over Parameterization and Redundancy : Modern neural networks are often over parameterized containing more parameters than necessary. This leads to redundancy and unstable decision boundaries, increasing the chance of overfitting and adversarial susceptibility.[17]

Sensitivity to Input Perturbations : Deep networks are highly sensitive to minor input changes. Even imperceptible noise can alter predictions, revealing a lack of continuity in the model's decision surface and making adversarial attacks effective.[12]

Lack of Built In Robustness : Unlike classical statistical models that enforce stability through regularization, deep networks are generally optimized for accuracy alone. Without explicit robustness constraints, they perform well in ideal conditions but fail under stress such as noise or distribution shifts.[18]

Poor Interpretability and Debugging : The hierarchical and non-linear nature of deep models makes them difficult to debug. Errors are hard to trace, and internal reasoning remains obscure—hindering trust and adoption in mission-critical applications. [14]

Reliance on Training Distribution : Deep models assume the testing data comes from the same distribution as training data. When faced with domain shifts or adversarially modified inputs, performance can degrade drastically—highlighting the need for generalization beyond seen data.[19]

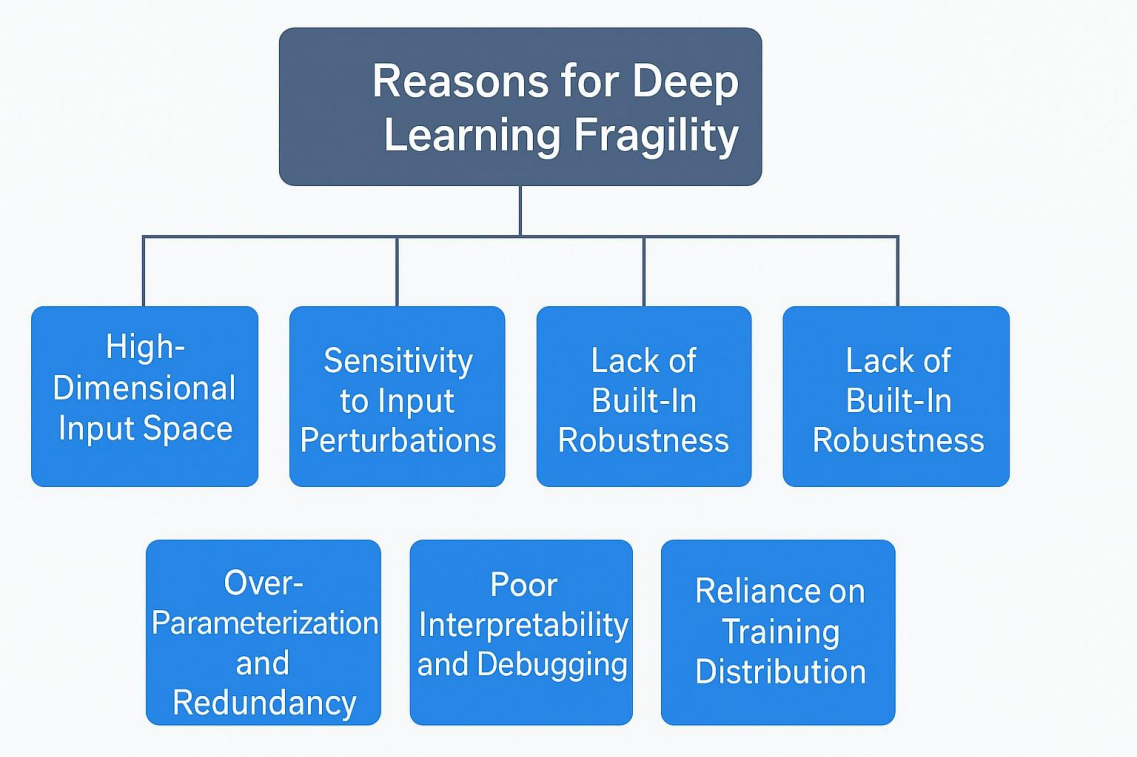


Figure 2.5: Illustration of the main reasons for deep learning fragility, including high-dimensional input space, over-parameterization, input sensitivity, lack of robustness, poor interpretability, and distributional dependence.

2.8 Conclusion

In this chapter, we introduced the foundations of deep learning, including neural network architectures, training mechanisms, and their applications in security critical fields. While deep learning models offer high performance across various domains, we highlighted their key vulnerabilities such as susceptibility to adversarial examples, overfitting, lack of interpretability, and sensitivity to distributional shifts. These weaknesses, both structural and behavioral, compromise their reliability and pose serious risks in real world deployment. Understanding these limitations is essential, as they form the foundation of the adversarial threats discussed in the next chapter.

The next chapter focuses on adversarial attacks, exploring their underlying principles, threat models, and classifications. This forms the basis for analyzing and addressing the critical challenges posed by adversarial machine learning.

Chapter 3

Adversarial Attacks and Defense

3.1 Introduction

Adversarial examples have emerged as a critical vulnerability in deep learning systems. These are inputs intentionally crafted with subtle perturbations often imperceptible to the human eye that can lead models to produce incorrect or even dangerous predictions. Despite their impressive performance on standard datasets, deep neural networks are surprisingly fragile when exposed to such manipulations, raising significant concerns in safety critical applications such as autonomous vehicles, medical diagnostics, and financial systems.

Understanding the nature and structure of adversarial attacks is essential for both assessing the robustness of existing models and guiding the development of more resilient architectures. By analyzing how these attacks exploit model weaknesses, researchers can better design mitigation strategies and incorporate robustness into training paradigms.

3.2 Definition Adversarial Attacks

An adversarial attack in deep learning refers to the intentional manipulation of input data to deceive a neural network into making incorrect predictions. The most prevalent form of such attacks involves the creation of adversarial examplesinputs that are subtly and strategically perturbed to mislead the model. Although these modified inputs appear virtually identical to the original data from a human perspective, they can significantly alter the output of the model.

This phenomenon exposes a critical vulnerability in deep learning systems, especially in high-stakes applications such as image recognition, autonomous driving, and medical diagnosis. [20]

3.3 Types of Adversarial Attacks

Adversarial attacks can be categorized into different types based on their execution and objectives:

3.3.1 Based on Attack Goals (Common Attack Types) :

- **1. Evasion Attacks:**These techniques are most often employed in a setting where an attacker introduces adversarial examples to fool a trained model into making incorrect predictions. This is the most commonly studied type of attacks on image recognition, natural language processing, and speech recognition systems.[21]

- **2. Poisoning Attacks:** Poisoning attacks are done during the training phase as opposed to evasion attacks. Here, the attacker grows a mischief in the training data with a view to creating vulnerabilities in the model, allowing for exploitation at a later stage. Adversarial perturbations, when applied to real-world datasets, can trigger misclassification of diseases in models based on AI diagnostic systems.

- **3. Backdoor Attacks:** In this attack, a hidden pattern (or trigger) is embedded into the training data. When this trigger appears in an input, the model produces a specific, often malicious, output. Backdoor attacks are particularly dangerous in cybersecurity and biometric authentication systems.

3.3.2 Based on Access to the Model (Threat Models):

- **1. White-box attacks** assume the adversary has full knowledge of the model's architecture, weights, and gradients, allowing them to craft precise adversarial examples [17] .

Example: Evasion of Malware Detection: An attacker gains full access to an AI-enabled malware detection system, together with its architecture and parameters. Using this knowledge, they produce malware which, in the eyes of the system, would appear harmless, by encoding it well enough to evade detection. Having insight into how the model makes decisions provides the attacker with the ability to design malware specifically evading the classification of a threat

impact: -Antivirus software fails to detect threats. -Sensitive systems become vulnerable to cyber-attacks. -Large-scale data breaches and financial losses.

- **2. Black-box attacks** assume limited or no knowledge of the model. Attackers generate adversarial examples by querying the model and observing its outputs, making them more practical for real-world exploitation .[12]

  Example: Fooling Online Fraud Detection Systems: The attacker tries to bypass detection by the bank's fraud detection AI without really knowing its internal structure. They make numerous unimportant low-value fraudulent transactions, trying out several of the different variations until they find patterns that elude the bank's fraud detection system. By observing what transactions cause alarms and which ones don't, they reverse-engineer the model's behavior and engage in one large-scale fraud without raising any alarms.

  impact: -Banks suffer financial losses. -Customer data and funds are at risk. -The integrity of financial AI systems is compromised.

3.4 Adversarial Attack Techniques

3.4.1 Fast Gradient Sign Method

The foundational work of Goodfellow et al. [21] introduced the Fast Gradient Sign Method (FGSM), demonstrating that model gradients can be exploited to craft perturbations that lead to misclassification, FGSM is one of the earliest and most widely used adversarial attack techniques. It is a white-box attack that perturbs the input data in the direction that maximally increases the model's loss, using the gradient of the loss function with respect to the input.

Given a small perturbation parameter ε, FGSM adds a calculated noise to the original input

x to create an adversarial example x':

$$x' = x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y)), \tag{3.1}$$

where $L(x, y)$ is the loss function, and $\nabla_x L$ is its gradient. This method allows attackers to generate adversarial examples in a single step, making it computationally efficient and effective, particularly for testing model robustness in white-box settings.

In targeted attacks, the sign is reversed to decrease the loss with respect to a chosen target class.

Definition :FGSM Adversarial Attack by Goodfellow et al. 2014 Let $x \in \mathrm{R}^n$ be a legitimate input data that are correctly classified as class y by an ML classifier f. The FGSM with L_∞-norm bounded perturbation magnitude generates an adversarial example $x' \in \mathrm{R}^n$ by maximizing the loss function $L(x', y)$ subject to the constraint $\|x' - x\|_\infty \leq \varepsilon$. That is,

$$x' = \begin{cases} x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y)^T), & \text{untargeted} \\ x - \varepsilon \cdot \text{sign}(\nabla_x L(x, t)^T), & \text{targeted on } t. \end{cases} \tag{3.2}$$

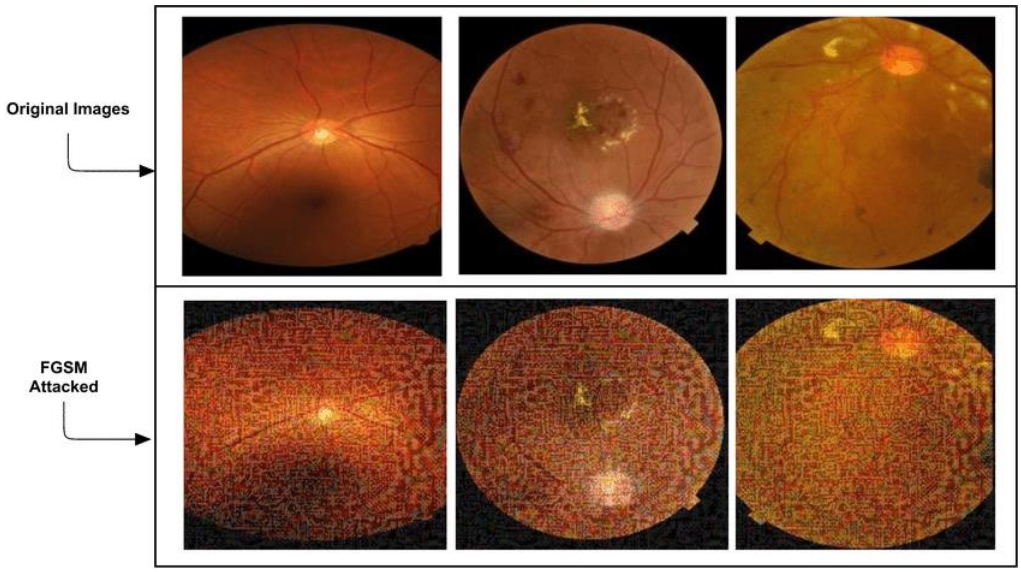


Figure 3.1: Adversarial perturbations generated using the FGSM attack on retinal images[4].

This image shows original retinal images (top) and their FGSM-attacked versions (bottom). A small perturbation,

$$x^{'} = x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y))$$

causes the model to misclassify, revealing its sensitivity to adversarial noise.

3.4.2 Projected Gradient Descent (PGD) Attack

Building upon FGSM, Madry et al. [17] proposed Projected Gradient Descent (PGD), a more powerful iterative method that remains a benchmark for evaluating adversarial robustness.

Definition 2 (PGD Attack): Let $x \in \mathrm{R}^n$ be a legitimate input correctly classified as class y by a classifier C. Given a loss function L, the PGD attack aims to find an adversarial example $x^{'} \in \mathrm{R}^n$ by solving the following optimization problem:

$$\text{maximize}_{\delta} \quad L(x + \delta, y)$$

subject to:

$$\|\delta\|_p \leq \varepsilon, \qquad x + \delta \in [0, 1]^n$$

Here, δ represents the perturbation added to the original input, constrained by the L_p-norm ball of radius ε. The adversarial example is then $x^{'} = x + \delta$.

The PGD attack is an iterative method that applies the following update rule:

$$x^{(t+1)} = \Pi_{B_\varepsilon(x)} \quad x^{(t)} + \alpha \cdot \text{sign}(\nabla_x L(x^{(t)}, y))$$

where:

- $x^{(t)}$ is the adversarial example at iteration t,
- α is the step size,
- $\nabla_x L(x^{(t)}, y)$ is the gradient of the loss with respect to the input,
- $\Pi_{B_\varepsilon(x)}$ denotes the projection operator onto the L_p-norm ball $B_\varepsilon(x)$.

This method ensures that the perturbation remains within the specified norm constraint, making the adversarial example imperceptible while effectively causing misclassification.

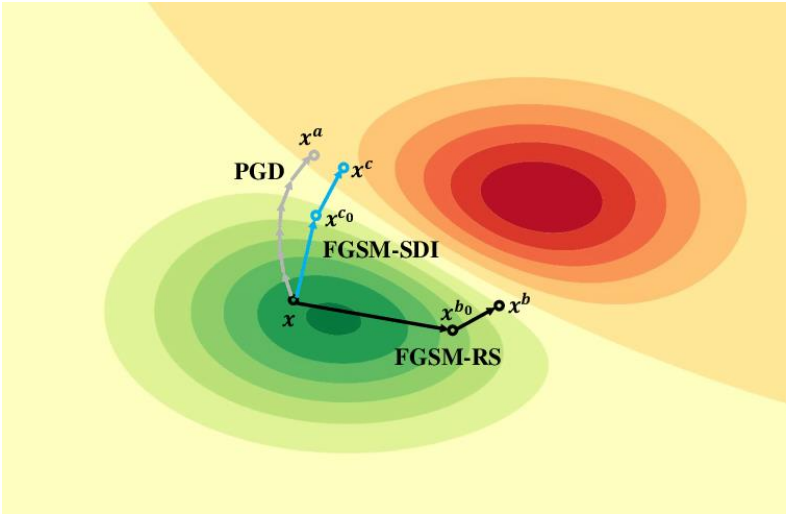


Figure 3.2: Illustration of adversarial examples generated using the PGD attack. The diagram shows how iterative perturbations move the input x toward regions of misclassification under different variants such as FGSM-RS, FGSM-SDI, and PGD[5].

The PGD attack is widely regarded as one of the most potent first-order adversarial attacks. It has been extensively used to evaluate and enhance the robustness of deep learning models, particularly through adversarial training techniques [17].

3.4.3 BadNet Attack

The BadNet attack is a classic example of a backdoor attack on deep neural networks (DNNs). Unlike typical evasion attacks, which modify the input at inference time, BadNet operates during training planting a hidden trigger in the model that activates malicious behavior only when a specific pattern is present. The BadNet framework by Gu et al. [22] exemplifies how triggers can be embedded in training data to elicit targeted misclassifications while preserving clean performance.

Definition (BadNet Attack): Let $x \in \mathrm{R}^n$ be a legitimate input correctly classified as class y by a classifier C. The BadNet attack embeds a trigger pattern δ such that the modified input $x' = x + \delta$ is always classified as a target class $t \neq y$, even though the model performs normally on clean inputs.

The attacker injects poisoned data into the training set by adding a small trigger (e.g., a pixel patch or watermark) to images and labeling them with a target class *t*. During training, the model learns to associate the trigger with *t*. At test time, the model misclassifies any input containing the trigger—even if the base input belongs to a different class.

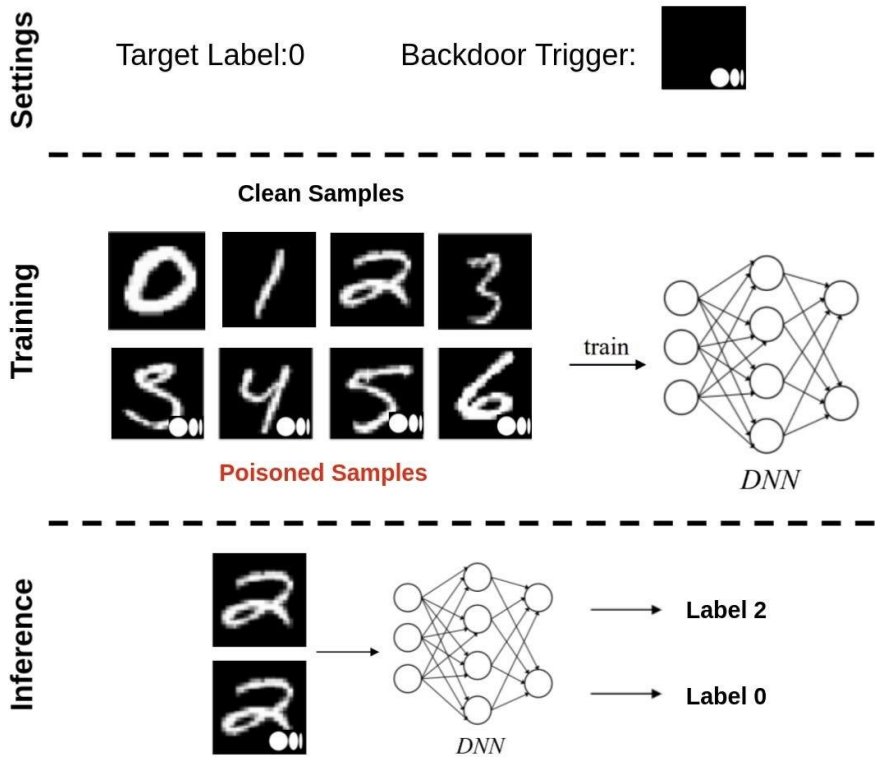


Figure 3.3: Illustration of the BadNet backdoor attack: a trigger (e.g., white pixel in the corner) is added during training to poison a subset of inputs. During inference, this trigger causes targeted misclassification (e.g., label 2 misclassified as label 0), while clean inputs are classified correctly[6]..

3.4.4 Transfer Attack

Szegedy et al. [12] introduced the concept of *transferability*, where adversarial examples crafted for one model can also deceive other models with different architectures or training data. This property is particularly useful in black-box attack scenarios, where the adversary does not have access to the internal parameters of the target model.

To exploit transferability, an attacker first trains a substitute (or surrogate) model using a labeled dataset. If labels are not accessible, they may be inferred through querying the target model or approximated using publicly available data. Once the substitute model is trained, the attacker applies white-box attack techniques—such as FGSM, PGD, or C& W to gener-

ate adversarial examples. These adversarial inputs are then transferred to the target model, often successfully causing misclassifications, even though the target model was never directly accessed.

Transfer-based attacks are especially dangerous because they enable real-world exploitation with minimal knowledge of the victim system.

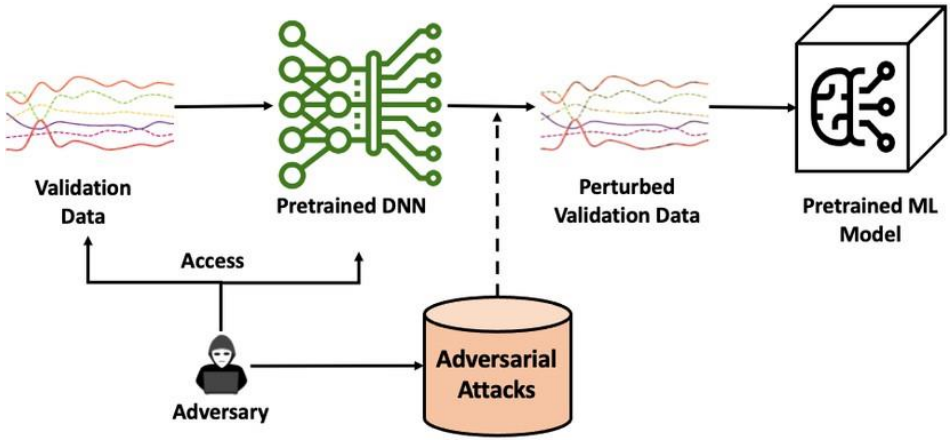


Figure 3.4: Illustration of adversarial attack transferability. An adversary generates adversarial examples using a pretrained DNN and applies them to another ML model[7].

3.4.5 Decision-based Attack

Decision-based attacks generate adversarial examples using only the model's predicted class label, without requiring access to gradients or confidence scores. This makes them practical for real-world black-box scenarios. In this subsection, we review three notable decision-based attacks: Boundary Attack, HopSkipJump Attack, and Square Attack.

The Triangle Attack [23] and BO-DBA [24], achieve adversarial success using only output labels, making them especially useful in black-box scenarios where internal model details are inaccessible.

3.4.5.1 Boundary Attack

Relying neither on training data nor on the assumption of transferability, the boundary attack uses a simple rejection sampling algorithm with a constrained independent and identically distributed Gaussian distribution as a proposed distribution and a dynamic step-size adjustment inspired by Trust Region methods to generate minimal perturbation adversarial samples. The

boundary attack algorithm is given as follows. First, a data point is sampled randomly from either a maximum entropy distribution (for an untargeted attack) or a set of data points belonging to the target class (for a targeted attack). The selected data point serves as a starting point. At each step of the algorithm, a random perturbation is drawn from a proposed distribution such that the perturbed data still lies within the input domain and the difference between the perturbed image and the original input is within the specified maximum allowable perturbation ε. Newly perturbed data is used as a new starting point if it is misclassified for an untargeted attack (or misclassified as the target class for a targeted attack). The process continues until the maximum number of steps is reached.

The boundary attack is conceptually simple, requires little hyperparameter tuning, and performs as well as the state-of-the-art gradient attacks in both targeted and untargeted computer vision scenarios without algorithm knowledge [25]. Furthermore, it is robust against common deceptions such as gradient obfuscation or masking, intrinsic stochasticity, or adversarial training. However, the boundary attack has two main drawbacks. First, the number of queries for generating an adversarial sample is large, making it impractical for real-world applications [26]. Instead of a rejection sampling algorithm, Metropolis-Hastings sampling may be a better option since it does not simply discard the rejected sample. Secondly, it only considers L_2- norm.

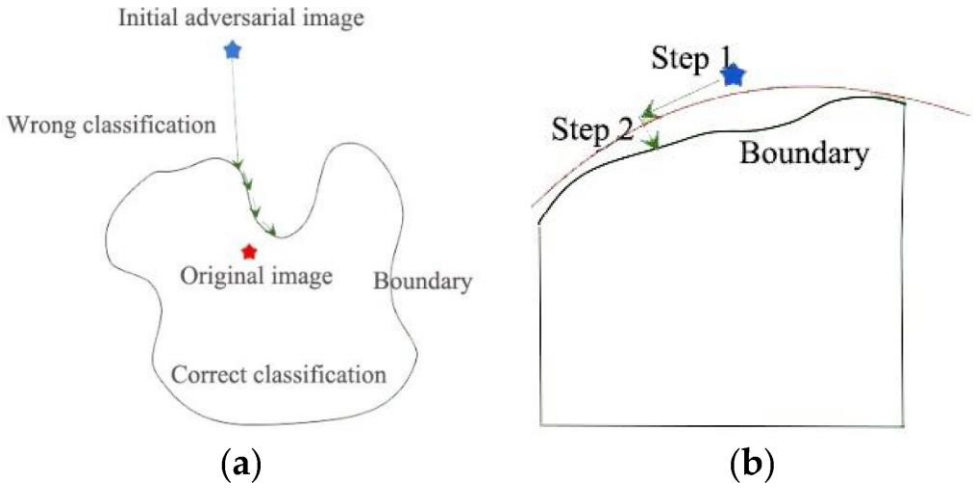


Figure 3.5: Illustration of the Boundary Attack. (a) The attack starts from a random adversarial image and follows a path toward the decision boundary while maintaining adversarial properties. (b) The refinement process ensures minimal perturbation[8].

3.4.5.2 HopSkipJump Attack

Conversely, the HopSkipJump attack is a family of query-efficient algorithms that generates both targeted and untargeted adversarial examples for both L_2 and L_∞-norm distances. Furthermore, the HopSkipJump attack is more query efficient than the Boundary attack [27], and it is a hyperparameter-free iterative algorithm. The HopSkipJump attack is defined as follows.

Definition 5 (HopSkipJump Attack by Chen et al. 2020): A HopSkipJump attack is a decision-based attack that generates the adversarial example x' by solving the following optimization problem:

$$\min_{x'} \| x' - x \|_p, \tag{3.3}$$

subject to the constraint $\varphi_x(x') = 1$, where $p \in \{2, \infty\}$,

$$\varphi_x(x') = \text{sign}(S_x^*(x')) = \begin{cases} 1, & \text{if } S_x^*(x') > 1 \\ -1, & \text{otherwise} \end{cases} \tag{3.4}$$

and

$$S_x(x') = \begin{cases} [f(x')]_t - \max_{j \neq t} [f(x')]_j, & \text{if target on } t \\ \max_{j \neq y} [f(x')]_j - [f(x')]_y, & \text{if untargeted} \end{cases} \tag{3.5}$$

Note that S_x^* is similar to (3.18) with $\kappa = 0$. Let us discuss HopSkipJump L_2 based targeted attack in detail. Interested readers can consult [27] for untargeted attack or L_2 based attacks.

The HopSkipJump L_2 based targeted attack is an iterative algorithm consisting mainly an initialization step (step 1) and three iterative steps (steps 2-4 are repeated until the maximum number of iterations specified by an attacker is reached.):

1. Select a random data point $\tilde{x}_0$ from the target class (initialization).

2. Approach the decision boundary via binary search:

$$x_t = \alpha_t x + (1 - \alpha_t)\tilde{x}_t,$$

where α_t is determined by binary search so that x_t lies on the decision boundary.

3. Estimate the gradient direction (similar to FGSM):

$$v_t = \frac{\nabla S(x_t, \delta_t, B_t)}{\| \nabla S(x_t, \delta_t) \|_2},$$

with $\delta_t = \frac{\|x_t - x\|_2}{n}$.

4. Update the sample point using step size ξ_t:

$$\xi_t = \frac{\|x_t - x\|_2}{\sqrt{t}},$$

reducing ξ_t by half until the new sample remains adversarial,

$$\tilde{x}_t = x_t + \xi_t v_t = x_t + \xi_t \frac{\nabla \hat{S}(x_t, \delta_t, B_t)}{\| \nabla \hat{S}(x_t, \delta_t) \|_2}.$$

3.4.5.3 Square Attack

The Square Attack by Andriushchenko et al. [28] utilizes square-shaped perturbations in a randomized iterative process, providing strong performance in decision based adversarial settings. It's a query-efficient black-box attack capable of generating both targeted and untargeted adversarial examples. It operates under the L_2 or L_∞ norm constraints and requires only the model's final decision (label), making it applicable in decision-based attack settings. Unlike gradient-based methods, Square Attack performs randomized perturbations in the input space, specifically using square-shaped patterns.

Definition 6 (Square Attack by Andriushchenko et al. 2020): Let f be a classifier and x an input sample correctly classified as class y. The Square Attack generates an adversarial example x' by solving:

$$\min_{x'} \|x' - x\|_p \quad \text{subject to} \quad f(x') \neq y, \quad x' \in [0, 1]^n, \tag{3.6}$$

where $p \in \{2, \infty\}$ and perturbations are applied in square-shaped blocks at randomly selected positions in x.

The attack follows an iterative random search procedure:

1. **Initialization:** Start with a perturbed version of the input x (either random or based on prior knowledge).
2. **Perturbation Step:** At each iteration, select a square region in the input and apply a fixed or randomly chosen perturbation.
3. **Evaluation:** Query the model to check if misclassification is achieved. If so, update x'; otherwise, try a new region or perturbation.
4. **Repeat:** Continue the process until a successful adversarial example is found or the maximum query budget is reached.

The Square Attack does not require access to model gradients or confidence scores, making it suitable for practical black-box scenarios. Despite its simplicity, it achieves competitive performance against many black-box defenses and has become a standard benchmark in adversarial robustness evaluations.

3.4.6 Clean Label Data Poisoning Attack

Clean label data poisoning attacks compromise deep learning models by corrupting the model's learning process with poisoned training samples that are wrongly labelled-in a way to cause specific misclassifications in target classes. Clean-label data poisoning attacks leverage the models learning process without modifying labels or accessing the labelling pipeline.

Typically used in targeted scenarios, clean-label attacks embed adversarial features into seemingly benign inputs, preserving model performance on clean data while manipulating predictions on specific instances. A common example is in facial recognition, where an innocuous image causes misclassification of a target individual during inference.

This subsection outlines a practical clean-label poisoning approach applicable to both transfer learning and end-to-end training setups.

3.4.6.1 Blending-Based Clean-Label Attack

Shafahi et al. [29] demonstrated that blending-based clean label attacks where source images are imperceptibly mixed with target class samples can effectively poison neural networks without explicit triggers or label control. In the blending-based clean-label attack, a portion of training samples from the source class (e.g., class 7) were blended with images from the target class (e.g., class 1) using a fixed blending ratio. The resulting images remained visually similar to the source class but included subtle features of the target class, without introducing visible triggers. These blended images were then relabeled to the target class and inserted into the training data.

During training, the model incorporated these visually consistent yet label-manipulated samples, which encouraged it to associate ambiguous features with the target class. As a result, the trained model exhibited a higher misclassification rate of class-7 images as class 1 at inference, even though the poisoned samples were hard to distinguish from genuine data.

This method demonstrates how data-level blending, combined with label manipulation, can achieve effective targeted attacks while preserving the visual plausibility of training images.

3.4.6.2 Feature Collision Attack

The Feature Collision Attack, proposed by Shafahi et al. [29], assumes a threat model in which the attacker has no access to the training data, no control over the labeling process, and no ability to alter the target instance during inference". However, the attacker does possess knowledge of the architecture and parameters of the feature extractor used in the model.

The core idea is to craft a poison instance that is visually similar to a base instance from class c (where $c \neq t$), but lies close to a chosen target instance from class t in the feature space. As a result, the poison sample is labeled as class c by human annotators, but the model after training learns to associate it with the target, causing a targeted misclassification at test time.

Definition 17 (Feature Collision Attack by *Shafahi et al.*, 2018)] Given a feature extractor f, a target instance x_t from class t, and a base instance x_b that belongs to a targeted class b such that $b \neq t$, it is called a feature collision attack if an attacker finds a poison instance x_p as follows:

$$x_p = \underset{x}{\operatorname{argmin}} \|f(x) - f(x_t)\|_2^2 + \beta \|x - x_b\|_2^2.$$

In the Feature Collision Attack, the parameter β balances two objectives: bringing the poison instance closer to the target in feature space, while keeping it visually similar to a base instance from another class. The optimization minimizes a weighted sum of these two distances, enabling targeted misclassification without altering labels.

Base instances are selected from classes different from the target, and some are more effective than others. Shafahi et al. [29] used a forward–backward splitting algorithm to solve this problem. Although highly successful in transfer learning, the attack is less effective in end-to-end or black-box settings. Enhancements like watermarks and multiple poison samples can improve results but may reduce stealth by introducing visual artifacts.

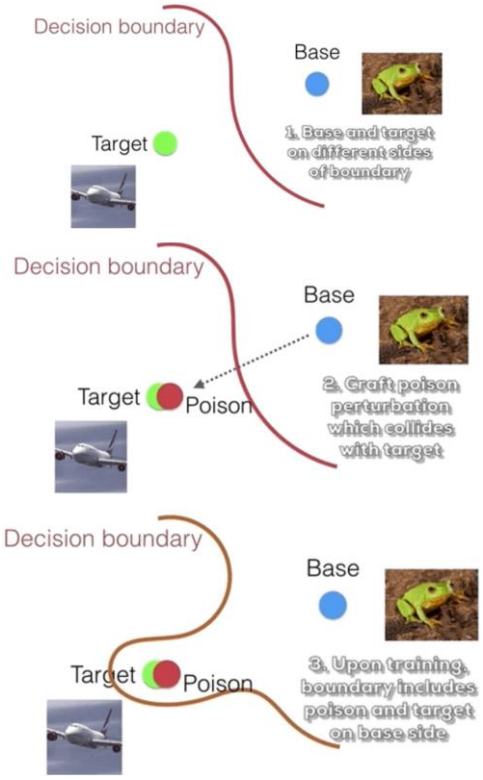


Figure 3.6: Illustration of Feature Collision Attack: A poison instance is crafted to collide with the target in feature space, affecting model decisions[9].

The image demonstrates a Feature Collision Attack, considered a data poisoning strategy.

1. Before the Attack:Both the target (green) and base (blue) are classified correctly since they are on separate sides of the decision boundary.

2. Poisoning:The attacker adds a carefully crafted poison sample (red), which closely resembles the target in feature space.

3. After Model Training:The decision boundary moves, resulting in the model misclassifying the target as belonging to the base class (for example, an airplane is misclassified as a frog). The model can be fooled without a change in the obvious label, making it difficult to detect.

3.5 Definition Adversarial Defense

Adversarial Defense Mechanisms are methods for shielding deep learning models from the weaknesses shown via adversarial attacks. These attacks have the capability to exploit subtle perturbations within input data. They do this for misleading model predictions, posing risks to

critical systems such as autonomous vehicles, healthcare diagnostics, and security applications. Defense mechanisms frequently aim to further improve model robustness. These mechanisms ensure greatly reliable performance even when exposed to quite malicious or manipulated inputs. Threats from adversarial attacks can matter more for machine learning (ML) system reliability. With subtle input data from changes, attackers may skew model forecasts of sorts, posing security threats within autonomous vehicles and biometric IDs.

3.6 Preprocessing Defenses Mechanisms

Preprocessing defenses have become a meaningful protective measure to fix these weaknesses. These techniques do indeed sanitize inputs before they are then duly processed by the target model, neutralizing adversarial noise while still indeed preserving the semantic content of the data. Preprocessing methods, unlike model-specific defenses, are architecture-agnostic, this making them highly flexible to diverse domains, as well as fields such as computer vision, speech recognition, and natural language processing.[30]

"Preprocessing-based defenses include methods like feature squeezing [31], JPEG compression [32], and additive Gaussian noise [33], each aiming to neutralize adversarial perturbations before model inference.

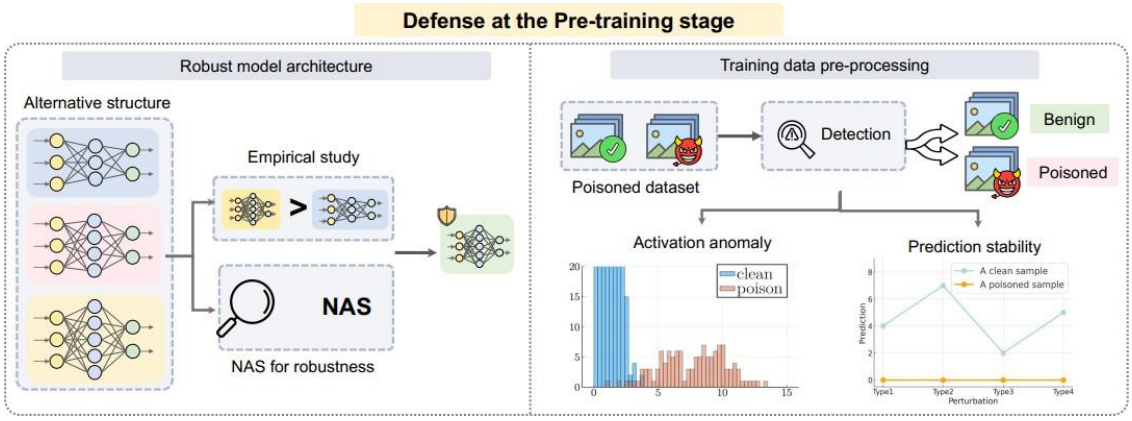


Figure 3.7: Defense mechanisms applied at the pre-training stage[10].

3.6.1 Gaussian Noise Injection

Gaussian Noise Defense is a relatively simple defense that can improve model robustness against adversarial attacks by adding random Gaussian noise to the input during either training

or inference.

In this process, the model's decision boundaries become smoother as it reduces sensitivity to small perturbations, thus, adversarial examples become more challenging to craft in order to fool the model.

Given a clean input x, the defended input x' is generated by adding Gaussian noise:

$$x' = x + N(0, \sigma^2)$$

where:

$N(0, \sigma^2)$ is Gaussian noise with mean 0 and variance σ^2.

σ controls the magnitude of the noise balancing robusteness vs clean accuracy.

This defense can be performed in two ways:

1. accumulate perturbation at training-time, injecting noise at each input during training to develop better generalization.
2. accumulate perturbation at test-time, injecting noise during inference to mask adversarial perturbation.

Definition: Gaussian Noise Defense

Let $x \in \mathrm{R}^n$ be a legitimate input correctly classified by a model f. The Gaussian Noise Defense generates a noise-augmented input x' by:

$$x' = x + N(0, \sigma^2 \cdot I)$$

subject to $\|x' - x\|_2 \leq \delta$, where δ bounds the expected perturbation. [34]

Efficacy

- **Top Row:** Original clean inputs, correctly classified by the model.
- **Bottom Row:** Same inputs augmented with Gaussian noise. Although the noise is visible, it does not alter human perception of the image. The model still classifies the inputs

correctly, demonstrating increased robustness. These noise-injected inputs remain viable for prediction.

3.6.2 Bit Depth Reduction Defense

Bit Depth Reduction is a preprocessing technique to act as a defense against adversarial attacks that typically rely on quantized pixel values, such as a mapper that quantizes the bit depth (or bit precision) of an input image. By reducing the reference precision of the pixel values, this method removes fine-grained adversarial perturbations while preserving essential visual features.[35]

Given an input image x with n bit precision, the defended image x' is generated by reducing its bit depth to k bits ($k < n$):

$$x' = \text{round}\left(x \cdot \frac{2^k - 1}{2^n - 1}\right) \cdot \frac{2^n - 1}{2^k - 1}$$

where:

- n: Original bit depth (e.g., 8-bit for standard images).
- k: Reduced bit depth (e.g., 3-bit or 4-bit).
- **Effect**: Coarse quantization "merges" subtle adversarial noise into larger pixel steps, making perturbations ineffective.

Definition: Bit Depth Reduction Defense

Let $x \in \mathrm{R}^n$ be a legitimate input (e.g., an image) correctly classified by a model f. The Bit Depth Reduction Defense generates a quantized input x' by:

$$x' = Q_k(x)$$

where: $Q_k(\cdot)$ is a k-bit quantization function, and $\|x' - x\|_\infty \leq \varepsilon$ (bounded perturbation).

Efficacy

- **Top Row (Original Images):** High-bit depth inputs (e.g., 8-bit) are correctly classified by the model.

- **Bottom Row (Reduced Bit Depth):** Inputs are quantized to k-bit precision. Adversarial perturbations become imprecise or disappear due to the coarse quantization.
- For moderate k values (e.g., 3–4 bits), the visual quality remains largely unaffected for humans, while model robustness increases significantly (e.g., under FGSM attack).

3.6.3 JPEG Compression

Compounding JPEG compression has become a common means of defending against adversarial attacks by preprocessing images. It is a lossy compression process in which the high-frequency components of an image are removed, thus removing adversarial perturbations while preserving a correct classification.[35]

Definition: JPEG Compression Defense

Let $x \in \mathrm{R}^{hwc}$ be an input image (height h, width w, channels c) correctly classified by a model f.

The JPEG-compressed defended input x' is generated by:

$$x' = \mathrm{JPEG}_{\mathrm{quality}}(x)$$

where:

- $\mathrm{JPEG}_{\mathrm{quality}}$: Applies JPEG compression with a specified quality factor $q \in [1, 100]$.
- **Effect**: High-frequency perturbations are discarded during compression, while semantic content remains intact.

Adversarial Robustness Guarantee: For an adversarial example $x^* = x + \delta$, JPEG compression enforces:

$$\| x' - x \|_2 \leq \eta \text{ (bounded distortion)}$$

where η depends on q. Lower q increases robustness but may degrade clean accuracy.

Mechanism

- **Color Space Conversion (RGB → YCbCr):** Separates the luminance (Y) component from chrominance (Cb, Cr), focusing compression on perceptual sensitivity.

- **Discrete Cosine Transform (DCT):** Decomposes the image into low- and high-frequency components.
- **Quantization:** Applies coarse quantization to high-frequency components, which typically contain adversarial perturbations.
- **Entropy Encoding:** Compresses the image by removing imperceptible redundancies.

Effectiveness

- **Top Row (Original Images):** Clean images correctly classified by the model.
- **Middle Row (Adversarial Examples):** Images perturbed by FGSM or PGD, causing misclassification.
- **Bottom Row (JPEG-Defended):** JPEG-compressed images (quality factor $q = 50$–75) where adversarial noise is diminished by quantization. The model recovers correct predictions with no noticeable degradation in image quality to the human eye.

3.6.4 Feature Squeezing Defense

Feature Squeezing is an input preprocessing technique that restricts the model's effective feature space by compressing the input data into lower-bit or smoothed representations. This makes it harder for adversarial perturbations—often embedded in high-frequency or fine-grained features—to remain effective. [36]

Definition of Feature Squeezing

Given an input x, feature squeezing applies one or more squeezing functions φ_i to produce transformed inputs. The model's output on x is then compared to the output on each squeezed input:

$$\text{If } f(x) \boxminus f(\varphi_i(x)) \text{ for any } i, \text{ then } x \text{ is flagged as adversarial.}$$

Common Squeezing Methods:

- **Spatial Smoothing:** Applies median blur or Gaussian filtering to remove high-frequency noise.

Example: $\varphi_{\text{median}}(x) = \text{median\_filter}(x, k)$, where k is the kernel size.

- **Non-Local Means Denoising:** Advanced smoothing that preserves edges while eliminating noise.

How It Works:

1. **Preprocess Input:** The input x is transformed using one or more squeezing functions $\varphi_1, \varphi_2, \ldots, \varphi_k$.

2. **Consistency Checking:** Compare predictions: if any $f(x) \neq f(\varphi_i(x))$, then x is suspected to be adversarial.

3. **Ensemble Squeezing:** Multiple squeezing methods can be combined to enhance detection effectiveness.[36]

Impacts of Feature Squeezing

Advantages:

- **Low Computational Cost:** Bit reductions and filters are computationally efficient ($O(1)$ per pixel).
- **Model-Agnostic:** Can be used with any model without retraining.
- **Effective Against LSB Attacks:** Prevents perturbations based on least significant bits.

Limitations:

- **No Formal Guarantees:** Effectiveness is empirical; not provably robust.
- **Potential Accuracy Loss:** Excessive squeezing may degrade performance on clean inputs.
- **Vulnerable to Adaptive Attacks:** Strong adversaries (e.g., PGD) may bypass the defense.[37]

3.7 Training-Time Defenses Mechanisms

Model-based defenses strive to enhance the resilience of machine learning (ML) systems to adversarial attacks by incorporating robustness in the training process or architecture of the

model. In contrast to preprocessing approaches that sanitize inputs prior to machine learning model input, these methods change how the model learns or perceives data, resulting in robustness by default in the model's predictions. This methodology is indispensable in high-stakes applications (e.g., medical diagnostics, autonomous systems) where adversarial threats demand a level of resilience (robustness). [30]

3.7.1 Adversarial Training

Adversarial training refers to training a model on adversarial examples that are generated during training where a model is explicitly trained on adversarial examples to improve robustness (e.g., using PGD Attacks(Inner Maximization): The inner maximization is routinely solved using Projected Gradient Descent (PGD):

$$\delta_{t+1} = \text{Proj}_{\Delta}(\delta_t + \alpha \cdot \text{sign}(\nabla_x L(f_\theta(x + \delta_t), y)))$$

where Proj_{Δ}: perturbations back into the valid set Δ. This generates the worst case perturbations for model resilience to learn from.[38]

Madry et al. [17] demonstrated that adversarial training using PGD adversaries significantly improves model robustness against a wide range of perturbations.

Mechanism Adversarial training forces the model to learn robust representations by minimizing the worst-case loss over perturbed inputs. This process strengthens the model's ability to resist adversarial attacks by explicitly exposing it to worst-case examples during training. While highly effective, adversarial training significantly increases computational cost and may not generalize well to unseen or novel attack strategies.

Strengths: proven robustness against gradient-based attacks.

Limitations: expensive computationally; accuracy on clean data may decrease (robustness-accuracy trade-off).

Definition of Adversarial Training

Let f_θ be a model trained on dataset D. The adversarial training objective is:

$$\theta^* = \arg\min_\theta \mathrm{E}_{(x,y)} \max_{\|\delta\|_p \leq \varepsilon} L(f_\theta(x+\delta), y)$$

subject to $\|x' - x\|_p \leq \varepsilon$, where ε controls perturbation magnitude. where:

- θ = model parameters.
- D = data distribution.
- δ = adversarial perturbation bounded by ε under ℓ_p-norm (e.g., ℓ_∞).
- L = loss function (e.g., cross-entropy).[39]

Efficacy

- **Top Row:** Original clean inputs, correctly classified by the model.
- **Bottom Row:** Inputs perturbed using adversarial attacks (e.g., PGD). Despite perturbations, the model maintains correct classifications, demonstrating improved robustness due to adversarial training.

Trade-offs

- **Robustness vs. Clean Accuracy:** Adversarial training often leads to reduced performance on clean data due to alterations in decision boundaries.
- **Computational Cost:** Multi-step training methods such as PGD-based adversarial training are computationally expensive, although they offer better robustness than simpler single-step approaches.

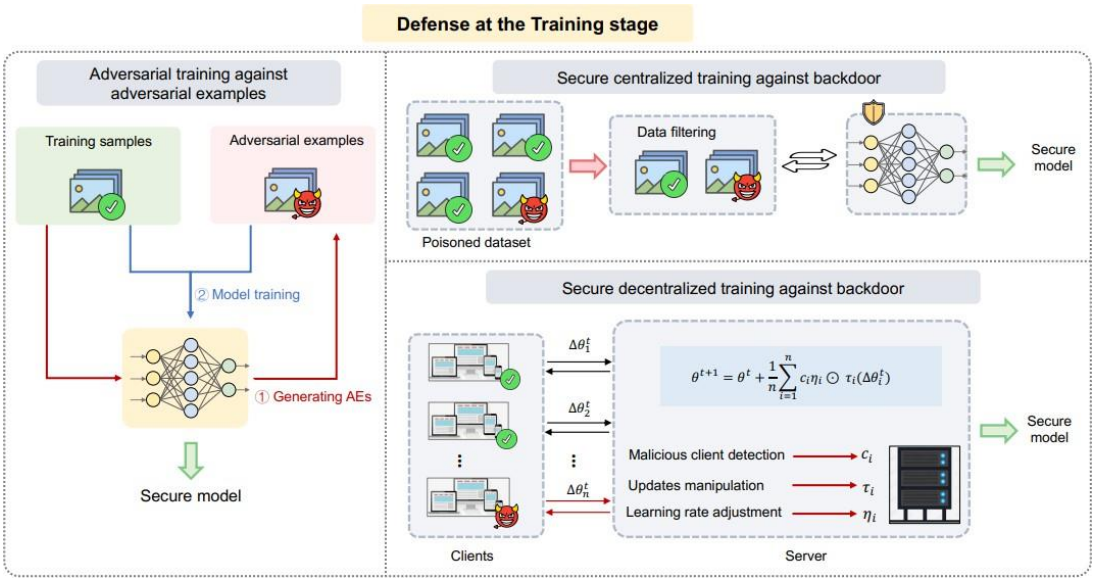


Figure 3.8: Overview of defense mechanisms applied during the training stage. The figure illustrates three categories: adversarial training against adversarial examples, secure centralized training against backdoor attacks using data filtering, and secure decentralized training leveraging client-server coordination to detect malicious clients and adjust learning updates[11]..

3.7.2 Label Smoothing Training

Label Smoothing is a regularization method used to improve generalization of models by reducing over-confidence on hard-labeled classification tasks. Instead of one-hot encoded labels (i.e. [1, 0, 0]), Label Smoothing assigns some smoothed probabilities over the non-target classes, so that the model will learn more robust and less overfitted decision boundaries. For a classification task with K classes, the one-hot label representing a true class Y is now replaced with a smoothed distribution. [40]

Definition of Label Smoothing Training

For a model f_θ with logits z and softmax $p_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, the label-smoothed cross-entropy loss becomes:

$$L_{\text{LS}} = -\sum_{i=1}^{K} y_i^{\text{LS}} \log(p_i)$$

This penalizes overconfident predictions (logits with extreme values) and encourages a more uniform distribution over incorrect classes.[40][41]

Impact

- **Affects Generalization & Calibration:** Label smoothing mitigates overfitting by reducing model confidence on the training set, which results in better generalization on unseen data.
- **Reduces Overfitting:** By softening the target labels, the model avoids memorizing exact outputs, leading to improved performance on test data.
- **Better Calibration:** The smoothed label distribution helps align predicted probabilities with the true uncertainty, particularly beneficial in avoiding overconfident but incorrect predictions.
- **Increases Adversarial Robustness:**
  - **Soften Decision Boundaries:** Prevents overly confident predictions, making the model more resistant to small perturbations in the input.
  - **Empirical Robustness:** Enhances resistance to attacks like FGSM without needing explicit adversarial training.[40]

Trade-offs & Limitations

- **Underconfidence:** Excessive smoothing (e.g., smoothing factor $\alpha \geq 0.2$) can result in underconfident predictions, reducing performance on clean data.
- **Limited Standalone Protection:** Label smoothing provides auxiliary robustness but is not as effective as adversarial training or noise injection against strong attacks like PGD.

Efficacy

- **Top Row:** One-hot encoded labels produce sharp and overconfident predictions (e.g., [0.99, 0.01, 0.00]).
- **Bottom Row:** Smoothed labels (e.g., [0.90, 0.05, 0.05]) yield softer, more balanced predictions, improving generalization and robustness.

3.7.3 SafetyNet Defense

SafetyNet is a model-based defense that incorporates a detection subnetwork into the architecture to identify and reject adversarial inputs during inference. Unlike adversarial training, it does not alter the decision boundaries but instead learns to recognize patterns indicative of adversarial perturbations. [37]

Definition: SafetyNet Training

Given a model f_θ composed of two branches:

- **Classifier branch:** Predicts class labels $\hat{y}$.
- **Detector branch:** Outputs adversarial probability $d \in [0, 1]$ (with $d = 1$ indicating high adversarial likelihood).

The training minimizes a joint loss function:

$$L_{\text{total}} = L_{\text{classifier}}(y, \hat{y}) + \lambda \cdot L_{\text{detector}}(d, d_{\text{true}})$$

where:

- $L_{\text{classifier}}$: Cross-entropy loss for classification.
- L_{detector}: Binary cross-entropy loss for adversarial detection.
- λ: Trade-off parameter (e.g., $\lambda = 0.5$).

Training Data:

- Clean samples (x_{clean}, y) with $d_{\text{true}} = 0$.
- Adversarial samples (x_{adv}, y) with $d_{\text{true}} = 1$.

Impact

- **Enhanced Discriminative Detection:**
  - *Flags perturbations:* Adversarial examples are detected by identifying anomalous patterns in feature space, such as high-frequency noise.
  - *Feature squeezing synergy:* Often used in conjunction with input transformations like median filtering to suppress adversarial artifacts.

- **Preserves Clean Accuracy:**
  - *Modular design:* The classifier remains unaffected for clean inputs, preserving baseline model performance.
  - *Selective rejection:* Only inputs with detector output $d > \tau$ (e.g., $\tau = 0.5$) are rejected.
- **Trade-offs and Limitations:**
  - *Adaptive attacks:* Advanced attacks like BPDA may bypass detection using gradient obfuscation.
  - *False positives:* The system may wrongly reject legitimate inputs that contain natural noise.
  - *Training cost:* Requires adversarial sample generation for supervised training of the detector.

Efficacy

Scenario & **Detector Output**	**Action**
Clean Input & $d \approx 0$	Classify as normal
Adversarial Input & $d \approx 1$	Reject the input
Ambiguous Input & $0.3 < d < 0.7$	Flag for review

Table 3.1: SafetyNet Defense Efficacy Based on Detector Score

Visualization:

- Top Row: Clean input → Low detector score ($d = 0.1$).
- Bottom Row: Adversarial input → High detector score ($d = 0.9$).

3.8 Post-processing Defenses Mechanisms

Post-processing defenses are methods undertaken to the outputs or predictions from a machine learning model, typically after the machine learning model has been invoked, with the goal of reducing adversarial attacks, or increasing robustness. Post-processing techniques provide a foil to other defenses that modify either the training, or the model algorithm itself (for example

adversarial training), in that they only handle the final prediction, without altering the model itself.

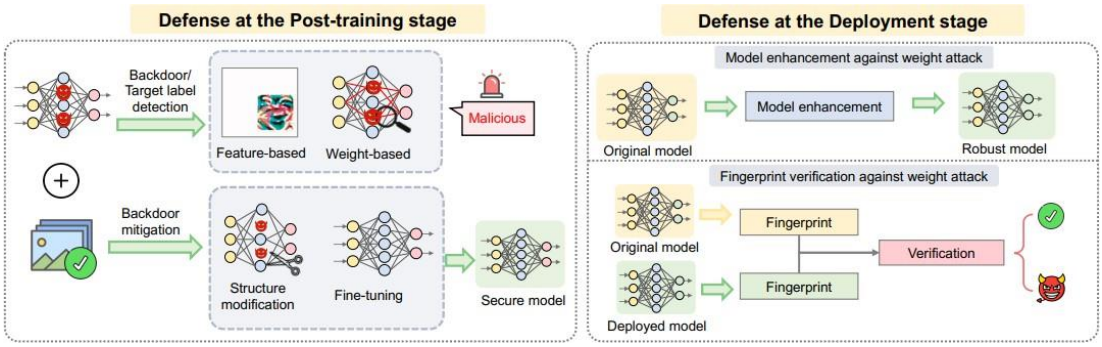


Figure 3.9: Defense strategies at the post-training and deployment stages. Left: Post-training defenses include backdoor and target label detection via feature or weight-based analysis, and backdoor mitigation through structural modifications and fine-tuning. Right: Deployment-stage defenses involve model enhancement to counteract weight attacks and fingerprint verification to validate model integrity after deployment[12].

3.8.1 Randomized Smoothing

Randomized smoothing is a probabilistic defense that converts a classifier into a certifiably robust one by adding noise to inputs. A smoothed classifier is defined as:

$$f'(x) = \arg\max_y \mathrm{P}_{\eta \sim N(0,\sigma^2 I)}[f(x+\eta) = y]$$

where:

- $N(0, \sigma^2 I)$ is Gaussian noise added to the input.
- $f'(x)$ represents the smoothed classifier.

This technique provides a robustness guarantee under ℓ_2 norm perturbations and is scalable to large models.[39] [42]

Definition of Randomized Smoothing

Given a base classifier f, the smoothed classifier g is defined as:

$$g(x) = \underset{c \in Y}{\operatorname{argmax}} \mathrm{P}_{\delta \sim N(0, \sigma^2 I)}[f(x + \delta) = c]$$

where:

- δ is Gaussian noise $N(0, \sigma^2 I)$.
- The smoothed classifier g returns the most probable class under random noise.[43]

Impact Randomized smoothing enhances the robustness of AI models by injectivity adding Gaussian noise to the inputs, thereby enabling a certifiable guarantee of robustness against adversarial perturbations. This technique is model-agnostic, easy to implement, and suitable for deployment in safety-critical applications. However, it requires careful parameter tuning (e.g., noise variance σ^2) and introduces additional computational overhead during inference. Despite its strengths, Randomized Smoothing can still be bypassed by sophisticated adaptive attacks.

3.8.2 Confidence Thresholding Defense

Confidence Thresholding is a straight forward but effective inference-time defense that fil- ters possible adversarial examples by rejecting ones with low-confidence predictions. Specifically, if the model's predicted likelihood of the top class is below a certain threshold, the input is marked suspicious or rejected completely. This limits attacks that rely on making low-confidence predictions from minimal perturbations for misclassification.[44]

Definition of Confidence Threshold

Given a model f_θ with softmax output p, the predicted class y' and its confidence p' are:

$$y' = \underset{i}{\operatorname{argmax}}\, p_i, \qquad p' = \max_i p_i$$

The defense rejects inputs where $p' < \tau$, where $\tau \in [0, 1]$ is a tunable threshold.

Impact

- **advantages:**
  - *Attack Mitigation:* Effective against low-confidence adversarial examples (e.g., FGSM, JSMA) that produce uncertain model outputs.
  - *Computational Overhead:* Adds minimal inference-time cost, making it efficient for deployment.
  - *Interpretability:* The confidence threshold τ provides an intuitive way to balance robustness and coverage.[45]
- **Limitations:**
  - *Bypass by High-Confidence Attacks:* Stronger adversaries (e.g., PGD) can generate confident but incorrect predictions ($p' \geq \tau$).
  - *Reduced Coverage:* Legitimate but difficult inputs (e.g., ambiguous or noisy) may be incorrectly rejected.
  - *Threshold Tuning:* Requires careful calibration using both clean and adversarial data to manage false positives and negatives.

Trade-offs

- **High τ:** Enhances robustness by rejecting uncertain predictions, but at the cost of rejecting more legitimate inputs.
- **Low τ:** Improves acceptance of clean inputs but increases the risk of admitting adversarial examples.

Efficacy

- **Top Row:** Clean input ($\hat{p} = 0.95$) $\Rightarrow$ Accepted.
- **Middle Row:** Low-confidence adversarial input ($\hat{p} = 0.30$) $\Rightarrow$ Rejected.
- **Bottom Row:** High-confidence adversarial input ($\hat{p} = 0.99$, wrong class) $\Rightarrow$ Bypasses threshold (requires complementary defense).

3.8.3 Majority Voting (Ensemble Methods) Defense

Majority Voting is a post-processing defense strategy that relies on an ensemble of models to detect adversarial examples. By leveraging the diversity of multiple classifiers, the method checks for consistency among their predictions. Adversarial examples are flagged when models disagree, under the premise that such discrepancies signal abnormal behavior.

Definition of Majority Voting Defense

Let $\{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_M}\}$ be an ensemble of M independently trained models. For a given input x:

- Each model outputs a prediction $f_{\theta_i}(x)$.
- If all predictions agree (i.e., unanimous vote), x is accepted as legitimate.
- If there is any disagreement, x is flagged as adversarial.

Formally:

$$\text{Consensus}(x) = \begin{cases} \text{Legitimate}, & \text{if } f_{\theta_1}(x) = f_{\theta_2}(x) = \cdots = f_{\theta_M}(x) \\ \text{Adversarial}, & \text{otherwise} \end{cases}$$

How It Works:

1. **Ensemble Construction:**
   - Use models with diverse architectures (e.g., ResNet, ViT, CNN).
   - Vary training data (e.g., clean, augmented, adversarial).
   - Randomize model initializations for heterogeneity.
2. **Inference Process:**
   - Pass the input x to each model.
   - Collect predictions: `[model1(x), model2(x), model3(x)]`.
   - Apply voting rule:
     - **Unanimous** ⇒ Accept prediction.
     - **Disagreement** ⇒ Reject as adversarial.

3. **Augmented Variants:**

    - **Weighted Voting:** Assign higher weight to more robust models.
    - **Confidence Voting:** Accept only when high-confidence models agree.

Impact of Majority Voting:

Advantages:

- **Robustness:** An attacker must deceive all models, making attacks exponentially harder.
- **No Training Cost:** Uses pretrained models; no additional training needed.
- **Interpretability:** Model disagreement is an intuitive indicator of suspicious input.

Limitations:

- **Computational Cost:** Requires multiple forward passes; slower inference.
- **Bypass by Adaptive Attacks:** Ensemble-aware attacks (e.g., ensemble adversarial training) can circumvent defense.
- **False Positives:** May incorrectly reject legitimate but ambiguous inputs near decision boundaries.

3.9 Hybrid & Emerging Defense Mechanisms: Bridging Robustness and Innovation

As adversarial attacks grow in complexity, they often surpass the protection offered by a single defense method. Hybrid defenses combine multiple strategies (e.g., pre-processing, post-processing methods, adversarial training) to establish layered security. Meanwhile, emerging defenses employ advanced technologies, such as quantum computing and neuromorphic architectures, to adapt to the evolving threat landscape in machine learning.[38]

Common Hybrid Defense Strategies

- **Pre-processing Combined with Adversarial Training:**
  Mechanism: Combines input filtering (e.g. randomized smoothing) with training on

adversarial examples.

Strengths: Effective against a wide range of attacks, including PGD and CW.

Limitations: Computationally expensive due to the need to generate adversarial examples during training.

- **Dynamic Defense Pipeline:**

  Mechanism: Applies sequential defenses (for example, DefenseGAN → adversarial detector → ensemble prediction).

  Strengths: Adapts to attack patterns and provides layered resistance, especially in white-box scenarios.

  Limitations: Introduces high latency and complexity.

- **Cross-Layer Robustness:**

  Mechanism: Implements protections across model layers, e.g., gradient masking early, feature squeezing late.

  Strengths: Obstructs gradient-based attacks across the model.

  Limitations: Requires architectural redesign, reducing ease of deployment.[42]

- **Human-in-the-Loop Verification:**

  Mechanism: Flags low-confidence predictions for human review (e.g., in healthcare or finance).

  Strengths: Effective in high-risk domains.

  Limitations: Not scalable for real-time or high-throughput systems.

3.10 Certified Defenses Mechanisms: Mathematically Guaranteed Robustness

Certified defenses provide formal mathematical guarantees that a model will remain robust under specified perturbation constraints, such as ℓ_p-bounded adversarial attacks. These defenses are crucial in mission-critical domains such as autonomous driving, cybersecurity, and medical diagnosis.[46]

Certified defenses, such as randomized smoothing [47], convex relaxation techniques [48], and interval bound propagation [49], offer provable robustness under norm-bounded perturbations,

though they face scalability limitations.

3.10.0.1 Convex Relaxation Methods (e.g., CROWN, Deep Poly)

Mechanism: Re-parameterized neural network activations into convex functions to compute an upper bound on the model's output under perturbation.

Certification: Provable robustness under ℓ_p norm constraints.

Strengths: Generates tight bounds; compatible with ReLU networks.

Limitations: Computationally intensive and restricted to specific activation types.[34]

3.10.1 Lipschitz-Constrained Networks

Mechanism: Enforces Lipschitz continuity to limit how adversarial perturbations propagate.

Certification: Guarantees ℓ_2 robustness based on the Lipschitz constant.

[50] *Strengths:* Simple to implement (e.g., via spectral normalization).

Limitations: Can reduce model expressiveness and performance on clean data.

3.10.1.1 Strengths and Limitations

Strength	**Limitation**
Provable security guarantees	Conservative bounds may reduce practical robustness
No reliance on attack-specific assumptions	High computational overhead for certification
Compatible with formal verification	Limited scalability to large models (e.g., ViTs)

Table 3.2: Strengths and Limitations of Certified Defenses

3.10.2 Evaluation and Trade-offs

Method	**Robustness Guarantee**	**Clean Accuracy**	**Certification Cost**	**Scalability**
Randomized Smoothing	Probabilistic (ℓ_2)	High	Very High	High
IBP	Deterministic (ℓ_∞)	Moderate	Low	Low
CROWN	Deterministic (ℓ_p)	Moderate	High	Moderate
Lipschitz Networks	Deterministic (ℓ_2)	Low	Low	Moderate

[49]

Table 3.3: Comparison of Certified Defense Methods Based on Key Evaluation Criteria.

3.11 Conclusion

This chapter examined the fragility of deep learning models to adversarial examples, while detailed prominent attack strategies (evasion, poisoning, and backdoor). The principal attacks included explained (FGSM, PGD, and decision-based attacks (e.g., Boundary, Square), poisoning, and clean-label poisoning).

In response to these threats, a structured taxonomy of defense mechanisms was introduced: preprocessing methods (e.g., Gaussian noise, JPEG), model-based defenses (e.g., adversarial training and label smoothing), and post-processing methods (e.g., confidence thresholding, randomized smoothing). Finally, hybrid defenses and certified defenses are discussed in addition to their potential to provide stronger robustness guarantees.

Chapter 4

Methodology

4.1 Introduction

The methodology was designed to provide a structured, and comprehensive benchmark of adversarial vulnerabilities and defense effectiveness in image classification. By exploring multiple attack surfaces and defense categories, the framework enables quantitative and qualitative insights into model robustness under adversarial conditions.

4.2 Project Description

We begin by describing the characteristics of the MNIST dataset, followed by the architectural choice of the ResNet18 model adapted for grayscale images. Next, we introduce the overall framework of our approach, which is divided into several key experimental phases:

- **Loading and training the ResNet18 model on MNIST data**, followed by evaluation on clean test samples.
- **Applying various adversarial attacks**, including FGSM, PGD, Clean Label, BadNet, and Square Attack, in order to assess model vulnerability to evasion, poisoning, and backdoor scenarios under both white box and black box settings.
- **Implementing adversarial defense mechanisms** categorized into three main types:
  - *Preprocessing defenses*: Gaussian noise injection, bit-depth reduction, JPEG com-

pression.

– *Postprocessing defenses*: Confidence thresholding, randomized smoothing.

– *Training time defenses*: Adversarial training against each individual attack.

- **Evaluating performance using standard metrics**: Accuracy, Precision, Recall, F1 score, Confusion Matrix, and Attack Success Rate.

These steps aim to build a comprehensive and comparative evaluation of model robustness and the effectiveness of applied defense strategies, within a controlled experimental framework based on MNIST and a convolutional architecture. As shown in Figure 4.1, the overall pipeline integrates adversarial attacks and defense mechanisms.

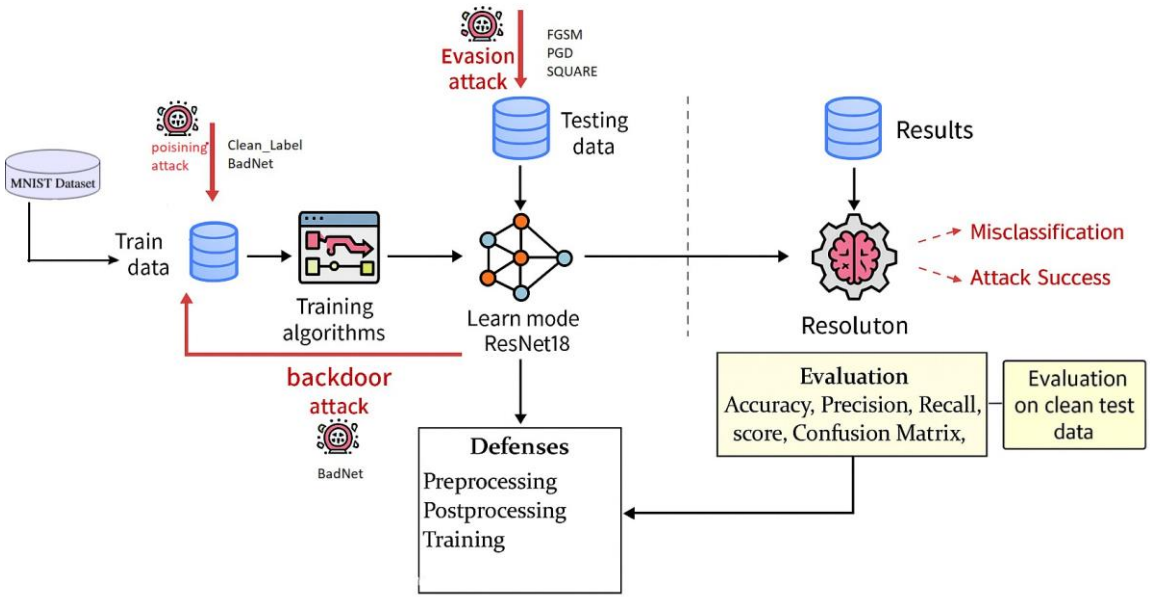


Figure 4.1: Adversarial attacks and defense evaluation pipeline using MNIST and ResNet18. The diagram illustrates data flow, attack injection, model training, defense integration, and evaluation outcomes.

4.3 Dataset Description (MNIST)

The MNIST dataset [51], a widely used benchmark in adversarial machine learning, comprises 70,000 grayscale images of handwritten digits (60,000 for training and 10,000 for testing),

each sized 2828 pixels. Representing digits 0–9, the dataset is well-balanced across classes. All images are single-channel with pixel intensities normalized to [0, 1]. MNIST's simplicity, diversity, and established use make it ideal for evaluating adversarial attacks and defense strategies under controlled conditions.

Attribute	**Value**
Image Size	2828
Color Channels	1 (Grayscale)
Classes	10 (Digits 0–9)
Training Samples	60,000
Testing Samples	10,000
Total Features per Image	784

Table 4.1: Summary of MNIST Dataset Characteristics

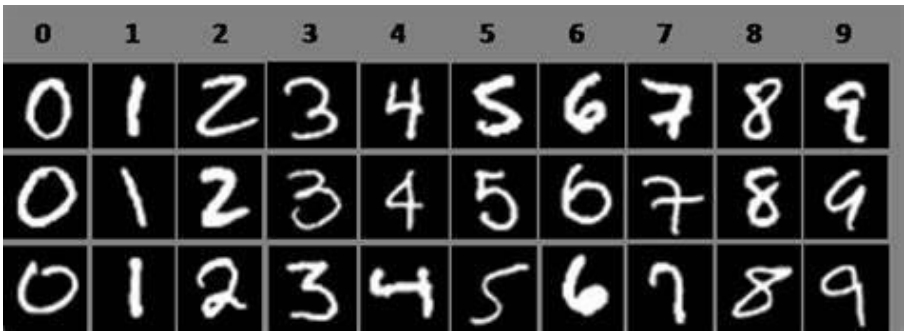


Figure 4.2: Example samples from the MNIST dataset showing digits from 0 to 9.

4.4 Model Used (ResNet18)

The model used in this study is a customized ResNet18 architecture adapted for the MNIST dataset. ResNet18 is a convolutional neural network known for its residual learning framework, which allows training of deep models by introducing shortcut connections that mitigate vanishing gradient issues .Originally designed for RGB (Red,Green,Blue) images, the architecture was modified to accept single channel grayscale inputs by adjusting the first convolutional layer to accommodate 1 input channel instead of 3 .The standard 77 convolutional layer was replaced by a 33 kernel to better suit the small 2828 grayscale images. The final classification head was replaced with a three layer fully connected network (256 128 10) using ReLU activations and dropout for regularization.

Training was conducted using the Adam optimizer (learning rate: 0.001) and cross-entropy loss for 100 epochs. The 60,000 training images were split into 80% training and 20% validation. At each epoch, validation accuracy was monitored, and the model achieving the best performance was saved as a checkpoint.

4.5 Adversarial Attacks

This section presents the five adversarial attacks implemented in our study to evaluate model vulnerability under different threat scenarios, including both white-box and black-box settings.

Attack	Type	Threat Model	Perturbation Details
FGSM	Evasion	White-box	$\varepsilon \in \{0.00, \ldots, 0.40\}$
PGD	Evasion	White-box	$\varepsilon = 0.3, \alpha = 0.01$, 40 steps
Clean Label	Poisoning	White-box	60% class-7 blended to class-1
Backdoor	Poisoning	White-box	90% class-7 with random 4×4 trigger
Square	Evasion	Black-box	$\varepsilon = 0.2$, 200 queries

Table 4.2: Summary of adversarial attacks used in this study.

We implemented five attacks in our practical evaluation: FGSM was run as a white-box evasion attack on the MNIST test set with varying ε (0.00 to 0.40); PGD was applied using $\varepsilon = 0.3$, $\alpha = 0.01$, and 40 iterations; for the clean label attack, 60% of class-7 training images were blended with class-1 and perturbed with Gaussian noise; a backdoor attack was conducted by injecting a 44 white square into 90% of class-7 training images and retraining the model on the poisoned data; and the Square Attack was performed using `torchattacks` with $\varepsilon = 0.2$ and 200 queries. Each attack was evaluated on the MNIST test set using Accuracy, Precision, Recall, F1-score, confusion matrices, and bar charts, with results saved in corresponding files.

4.6 Defense Mechanisms

This section outlines the defense strategies applied to counter adversarial attacks. These mechanisms are categorized into preprocessing, postprocessing, and training-time defenses.

4.6.1 Preprocessing Defenses

Defense Method	Key Parameters
Gaussian Noise	$\sigma = 0.3$
Bit Depth Reduction	3-bit quantization
JPEG Compression	Quality = 50

Table 4.3: Preprocessing defense mechanisms and their configurations.

Three preprocessing defenses were evaluated to enhance robustness against adversarial attacks: Gaussian noise (with $\sigma = 0.3$) was injected into input images, bit depth reduction quantized images to 3-bit precision, and JPEG compression with a quality factor of 50 was applied to adversarial samples. Each defense was systematically tested against all five attack types (FGSM, PGD, Clean Label, Backdoor, and Square), with model performance assessed using accuracy, precision, recall, F1-score, and confusion matrices. All results were saved and visualized for direct comparison across attack and defense scenarios.

4.6.2 Postprocessing Defenses

Defense Method	Key Parameters
Confidence Thresholding	Softmax threshold = 0.9
Randomized Smoothing	$\sigma = 0.25$, 50 samples

Table 4.4: Postprocessing defense mechanisms and their configurations.

For post-processing defenses, confidence thresholding was implemented by rejecting predictions with softmax confidence below 0.9, thereby filtering out low-confidence classifications under attack. Randomized smoothing was applied by generating 50 noisy versions of each input using Gaussian noise ($\sigma = 0.25$) and assigning the final class via majority vote. Both methods were evaluated against all attack types using accuracy, precision, recall, F1-score, and confusion matrix visualizations to assess their impact on adversarial robustness.

4.6.3 Training-Time Defenses

Defense Method	Key Parameters
Adv. Training (FGSM)	$\varepsilon = 0.25$, 50% mixed data
Adv. Training (PGD)	$\varepsilon = 0.3$, $\alpha = 0.01$, 40 steps
Adv. Training (Clean Label)	60% poisoned 7 1 samples
Adv. Training (Backdoor)	90% poisoned with 4×4 trigger
Adv. Training (Square)	$\varepsilon = 0.2$, 200 queries
Label Smoothing	Smoothing factor = 0.1

Table 4.5: Training-time defense mechanisms and their configurations.

For adversarial training, the model was retrained by mixing clean data with adversarial examples specific to each attack: FGSM ($\varepsilon = 0.25$), precomputed PGD samples ($\varepsilon = 0.3, \alpha = 0.01$, 40 steps), blended clean-label poisons (class 7/1), and backdoor samples with random white square triggers (90% of class-7 images). Square Attack ($\varepsilon = 0.2$) adversarial samples were similarly incorporated. Each training regime ran for 20 epochs using the Adam optimizer, starting from clean weights. Label smoothing (factor 0.1) was also applied to improve generalization and provide moderate resistance. All approaches were systematically evaluated under both clean and attack scenarios using accuracy, precision, recall, F1-score, and confusion matrix visualizations..

4.7 Evaluation Metrics and Visualization Tools

To assess the performance of the ResNet18 model under adversarial attacks and defense mechanisms, a set of standard classification evaluation metrics is used. These metrics provide a quantitative assessment of model robustness under both clean and adversarial conditions.

4.7.1 Classification Metrics

- **Accuracy:** The proportion of correctly predicted instances out of the total number of samples. It gives a general sense of model performance but may be misleading in imbalanced datasets.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** Measures the proportion of correctly predicted positive observations out of all predicted positives. It is particularly useful when the cost of false positives is high.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity):** Indicates the proportion of correctly predicted positives out of all actual positives. It is crucial in contexts where missing a true class is costly.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** The harmonic mean of precision and recall. It provides a balance between the two, especially in imbalanced datasets.

$$\text{F1-Score} = 2\frac{\text{PrecisionRecall}}{\text{Precision} + \text{Recall}}$$

4.7.2 Confusion Matrix

The confusion matrix is a visualization tool that displays the number of correct and incorrect predictions made by the model, broken down by each class. It is especially useful for analyzing misclassifications in adversarial settings.

4.7.3 Bar Chart Visualization

Bar charts are employed to compare performance metrics (Accuracy, Precision, Recall, F1-Score) across different conditions: clean data, adversarial attack, and defense-applied scenarios. This provides a clear visual summary of how adversarial attacks degrade model performance and how each defense mitigates it.

4.8 Conclusion

In this chapter, we presented the methodology adopted to assess the robustness of deep learning models against adversarial attacks. We described the dataset used, the ResNet18 model adapted for MNIST, and the step-by-step experimental process involving the generation of ad-

versarial examples, the application of various defense mechanisms, and the use of standardized evaluation metrics. This structured approach ensures a comprehensive assessment of both vulnerabilities and mitigation strategies.

In the next chapter, we will present the implementation results and discuss the performance outcomes in detail.

Chapter 5

Experimental Results and Evaluation

5.1 Introduction

In the previous chapter, we detailed the methodological steps taken to develop and structure our experimental framework. This chapter presents the practical implementation and results obtained from applying adversarial attacks and defense mechanisms on a ResNet18 model trained on the MNIST dataset. The objective is to assess the impact of each attack and the effectiveness of defense strategies using various evaluation metrics, with a focus on robustness and classification performance.

5.2 Tools and Libraries

The implementation and evaluation were carried out using the following hardware and software components:

Hardware Environment

Component	Specification
Processor	Intel Core i7-12500 (2.5 GHz, 12th Gen)
Motherboard	Gigabyte H610M H V2 DDR4
RAM	16 GB DDR4
Storage	SSD (Windows 11 installed)
GPU	NVIDIA GEFORCE RTX 3050

Table 5.1: Hardware Configuration Used for Experiments

Software Environment

The experiments were conducted using Microsoft Windows 11 Professionnel (Build 26100) as the operating system. Python version 3.12.10 served as the main programming language for implementing and running all modules.

For model training and inference, the PyTorch deep learning framework was used, alongside Torchvision which facilitated loading the MNIST dataset and applying image transformations. Adversarial examples were generated using the Torchattacks and Foolbox libraries, both of which offer a range of white box and black box attack implementations.

For evaluation and visualization, Matplotlib and Seaborn were employed to plot metrics, bar charts, and confusion matrices. To compute core classification metrics such as accuracy, precision, recall, and F1-score, Scikit-learn was utilized. Additionally, NumPy and Pandas were used for numerical computations and organizing result logs.

The TQDM library provided progress bars during model training and testing loops, enhancing monitoring. OpenCV (cv2) was integrated specifically for applying JPEG compression during preprocessing based defense experiments.

Development Tools

- **Visual Studio Code:** Primary development environment with Jupyter (.ipynb) support.
- **Jupyter Notebook:** For iterative experimentation and visualization.

5.3 Evaluation on Clean Data

To establish a baseline performance, we trained a ResNet18 model on the MNIST dataset using clean, unperturbed data. The model was trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The best validation accuracy achieved during training was **99.35%**, and the final test accuracy reached **99.19%**.

Performance Metrics

The model was evaluated on the entire MNIST test set using standard classification metrics. The obtained results are summarized below:

- **Accuracy:** 99.19%
- **Precision:** 0.9919
- **Recall:** 0.9919
- **F1 Score:** 0.9919

Visual Analysis

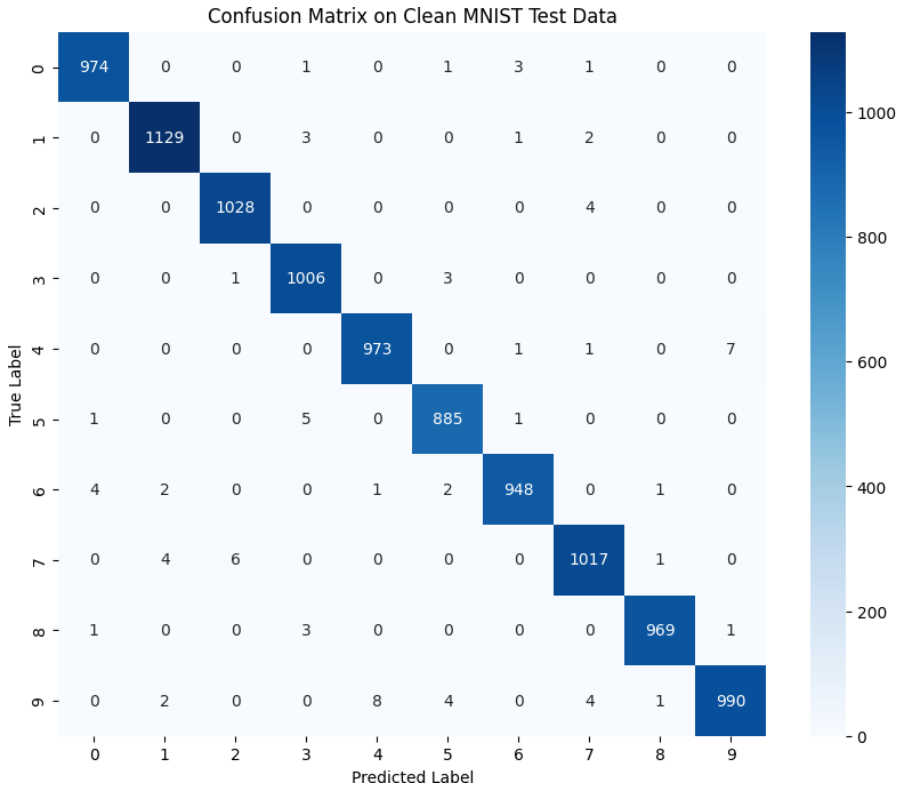


Figure 5.1: Confusion Matrix on Clean MNIST Test Data

Figure 5.2 illustrates the per-class accuracy of the model. As shown, the model consistently performs well across all digit classes, with class ’9’ exhibiting the lowest accuracy among them (still above 98%).

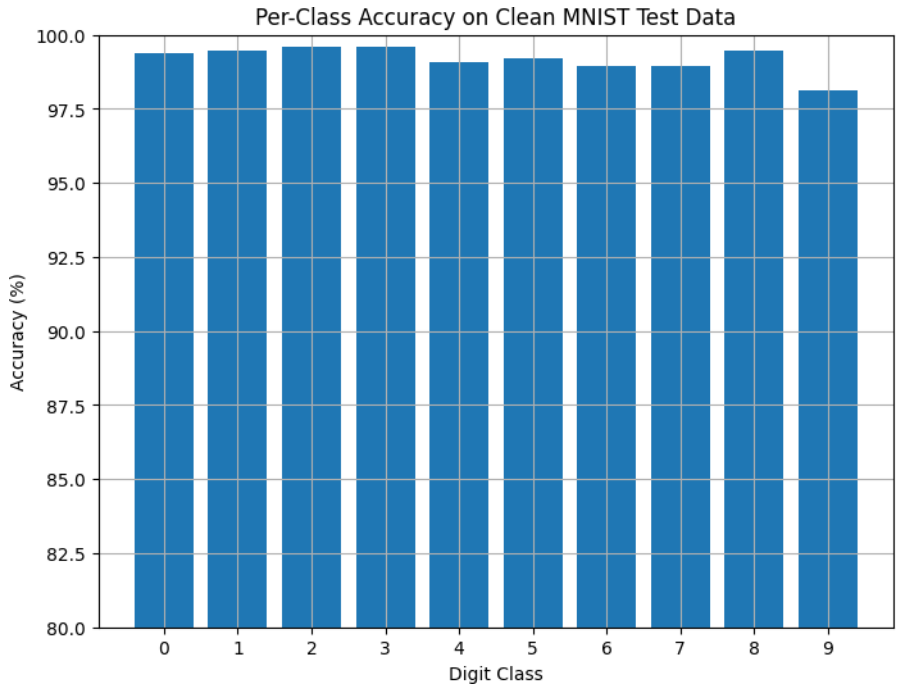


Figure 5.2: Per-Class Accuracy on Clean MNIST Test Data

These results confirm the reliability and generalization ability of the clean-trained ResNet18 model on standard MNIST data. This performance serves as the reference baseline to assess degradation due to adversarial attacks in subsequent sections.

5.4 Impact of Adversarial and Poisoning Attacks on the Clean Model

5.4.1 Evaluation Under FGSM Attack

To assess the vulnerability of the trained ResNet18 model to adversarial attacks, we applied the FGSM attack on the MNIST test set using varying perturbation levels ($\varepsilon = 0.05$ to 0.40). We assessed the model's performance under these adversarial conditions and provided visual insights for clarity. The observed results are presented in table 5.2 . The comparison results are illustrated in figure 5.6

Epsilon	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
0.00 (Clean)	99.19	99.19	99.19	99.19
0.05	95.24	95.21	95.21	95.19
0.10	83.24	83.40	83.13	83.02
0.20	55.39	54.14	55.33	53.29
0.30	49.59	47.66	49.57	47.17
0.40	48.71	48.05	48.69	46.86

Table 5.2: Classification metrics of ResNet18 under FGSM attack with varying ε.

Effect of Perturbation Magnitude

The relationship between attack strength and model accuracy is depicted in Figure 5.3. This figure clearly demonstrates that even small perturbations can significantly degrade model performance. For $\varepsilon = 0.05$, accuracy remains relatively high, but beyond $\varepsilon = 0.10$, the drop becomes drastic. At $\varepsilon = 0.30$, the model accuracy falls below 50%, indicating widespread misclassification.

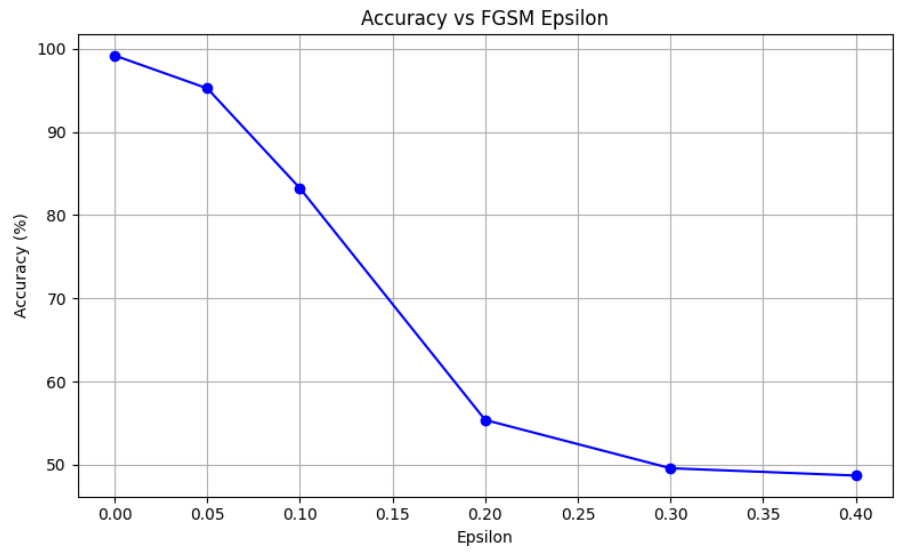


Figure 5.3: Test accuracy of ResNet18 on MNIST under FGSM attack with increasing ε.

To gain deeper insight into the impact of the attack, Figure 5.4 presents the confusion matrix for $\varepsilon = 0.30$.

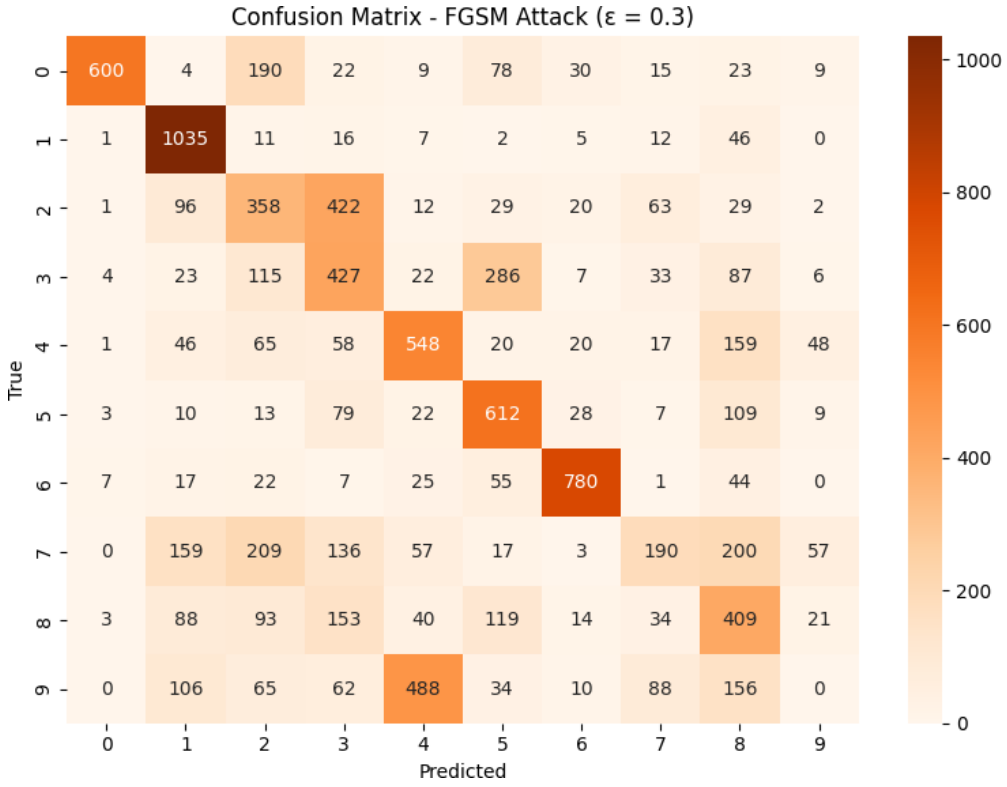


Figure 5.4: Confusion matrix of ResNet18 predictions on MNIST under FGSM attack ($\varepsilon = 0.30$).

Compared to the clean scenario, the FGSM confusion matrix shows a significant drop in diagonal dominance, indicating widespread misclassifications. Prominent off-diagonal entries reflect increased confusion among digit classes under attack.

Qualitative Example: Prediction Before and After Attack

To visually demonstrate the adversarial effect, Figure 5.5 shows an example where a clean image of the digit ’7’ is correctly classified, but the adversarially perturbed version ($\varepsilon = 0.30$) is misclassified as ’8’.

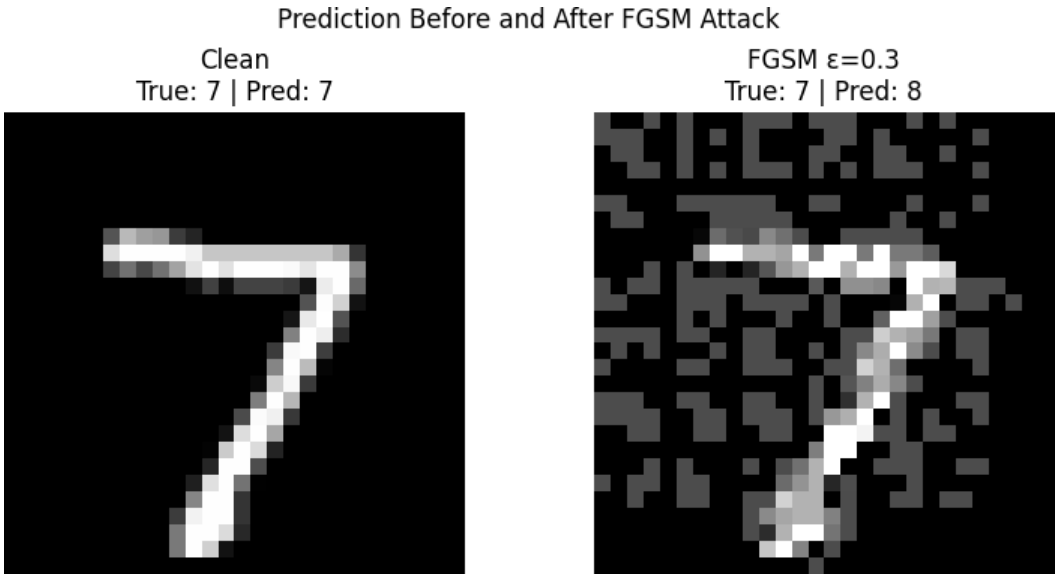


Figure 5.5: Sample prediction before (left) and after (right) FGSM attack. The adversarial image leads to incorrect classification.

The adversarial perturbation is barely perceptible to the human eye but is sufficient to completely alter the model's prediction, highlighting the effectiveness and subtlety of the FGSM attack.

Comparison of Clean vs. Attacked Metrics

Figure 5.6 contrasts the key classification metrics under clean and adversarial ($\varepsilon = 0.30$) conditions.

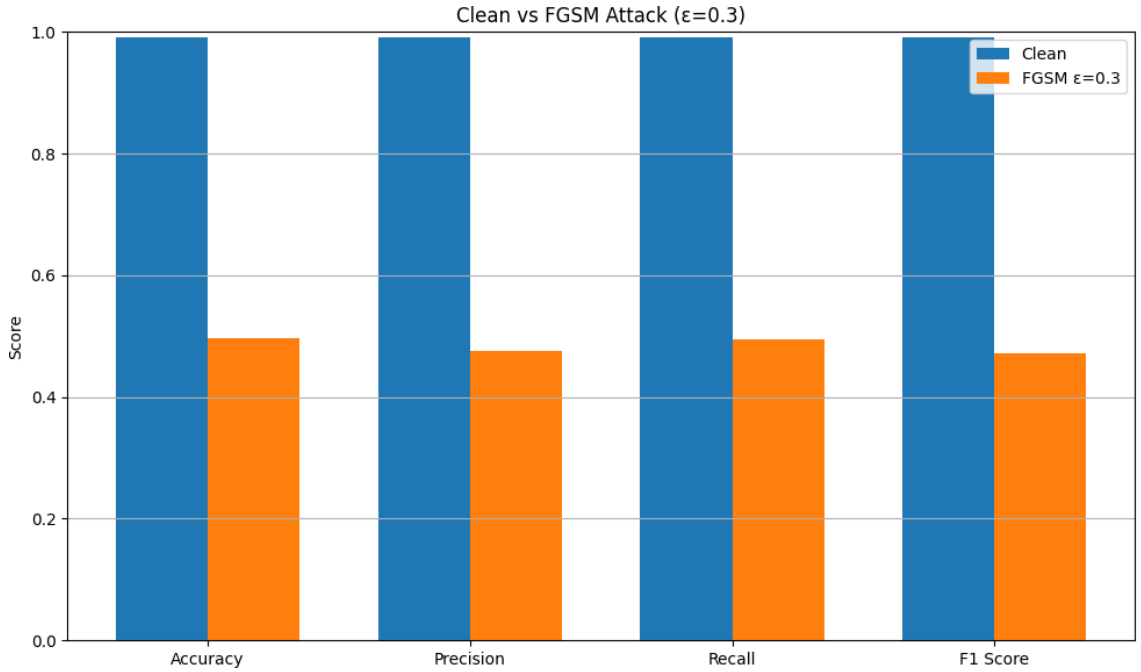


Figure 5.6: Comparison of accuracy, precision, recall, and F1-score for clean and FGSM-attacked data ($\varepsilon = 0.30$).

The significant drop in all metrics when moving from clean to adversarial data further quantifies the model's vulnerability to FGSM attacks.

Discussion

The FGSM evaluation reveals that the ResNet18 model is highly susceptible to adversarial examples. Small perturbations introduced by FGSM can severely compromise model performance, emphasizing the importance of adversarial robustness in security-critical applications.

5.4.2 Evaluation Under PGD Attack

To further assess the robustness of the ResNet18 model, we evaluated its performance against the Projected Gradient Descent (PGD) attack. This section details the classification results, confusion analysis, and qualitative examples under a representative attack strength ($\varepsilon = 0.3$). The observed results are presented in table 5.3. The comparison results are illustrated in figure 5.7

Condition	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Clean	99.19	99.19	99.19	99.19
PGD $\varepsilon = 0.3$	47.18	45.05	47.17	45.11

Table 5.3: Classification metrics under PGD attack ($\varepsilon = 0.3$) compared to clean baseline.

Figure 5.7: Comparison of accuracy, precision, recall, and F1-score for clean and PGD-attacked data ($\varepsilon = 0.3$).

The results reveal a marked reduction degradation in performance under PGD attack. All classification metrics drop from approximately 99% to below 48%, confirming the effectiveness of PGD in generating highly misleading adversarial examples.

The confusion matrix in Figure 5.8 illustrates how predictions are distributed across classes for the PGD attack ($\varepsilon = 0.3$).

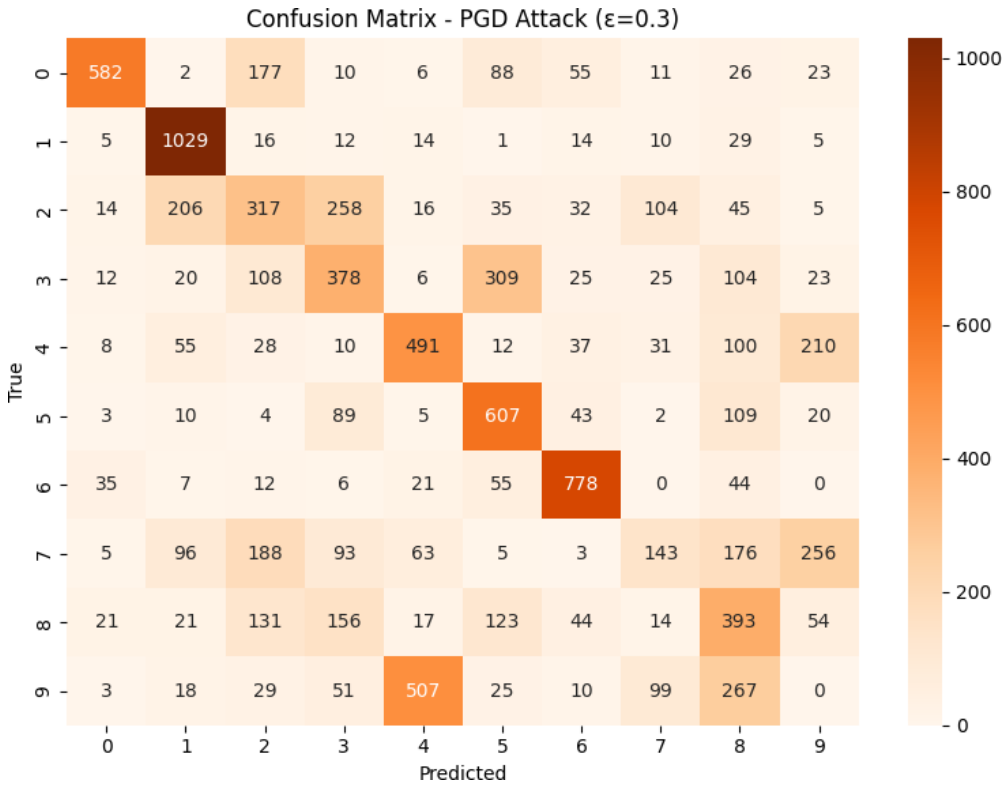


Figure 5.8: Confusion matrix for ResNet18 predictions under PGD attack ($\varepsilon = 0.3$).

PGD causes widespread confusion between classes. Misclassifications are frequent and scattered, particularly among visually similar digits. The diagonal of correct classifications is greatly diminished compared to the clean baseline.

Qualitative Example: Prediction Before and After Attack

Figure 5.9 displays an example of a clean image (correctly classified as ’7’) and its adversarial counterpart (misclassified as ’8’).

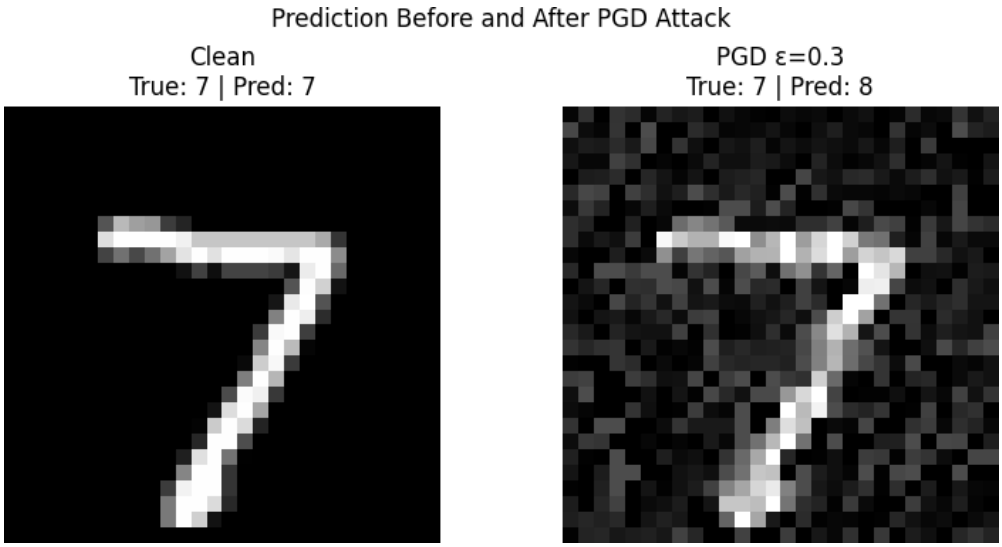


Figure 5.9: Prediction example: clean image (left, correctly classified) and adversarial image (right, misclassified) under PGD attack ($\varepsilon = 0.3$).

Discussion

The PGD evaluation highlights that the ResNet18 model, although highly accurate on clean MNIST images, is extremely vulnerable to adversarial attacks. PGD in particular is highly effective, reducing accuracy to under 48% for $\varepsilon = 0.3$ and causing widespread confusion among classes. These results underscore the need for robust defense mechanisms in real-world applications.

5.4.3 Evaluation Under Clean Label Attack (7→1)

To evaluate the resilience of the ResNet18 model against data poisoning attacks,we implemented an enhanced blended clean label attack, only during the training phase, targeting class 7 to be mislabeled as class 1. In this scenario, 60% of class 7 samples in the training set were blended with class 1 images to create visually plausible but maliciously labeled examples.Table 5.4 summarizes the key results,The attack success rate is also reported.Figure 5.10 compares the main metrics between the clean test set and the blended attack setting.

Condition	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Clean Test Set	99.19	99.19	99.19	99.19
Blended Clean Label Attack	94.39	89.41	87.05	88.08

Table 5.4: Classification metrics for ResNet18 under enhanced blended clean label attack (7→1, 60% poisoning).

Attack Success Rate (7→1): 52.24%

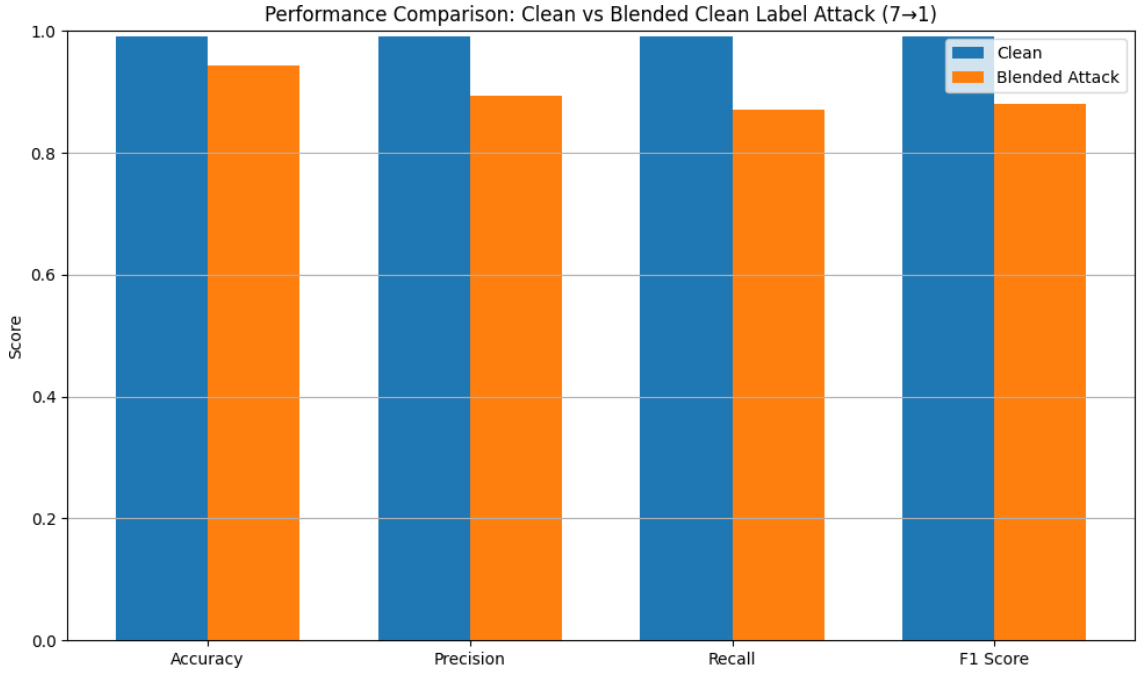


Figure 5.10: Performance comparison for clean vs. enhanced blended clean label attack (7→1).

All evaluation metrics show a decline under the blended attack, most notably in recall and F1-score. This indicates the model's reduced ability to correctly identify digits when presented with poisoned training data.

Figure 5.11 displays the confusion matrix for the blended attack.

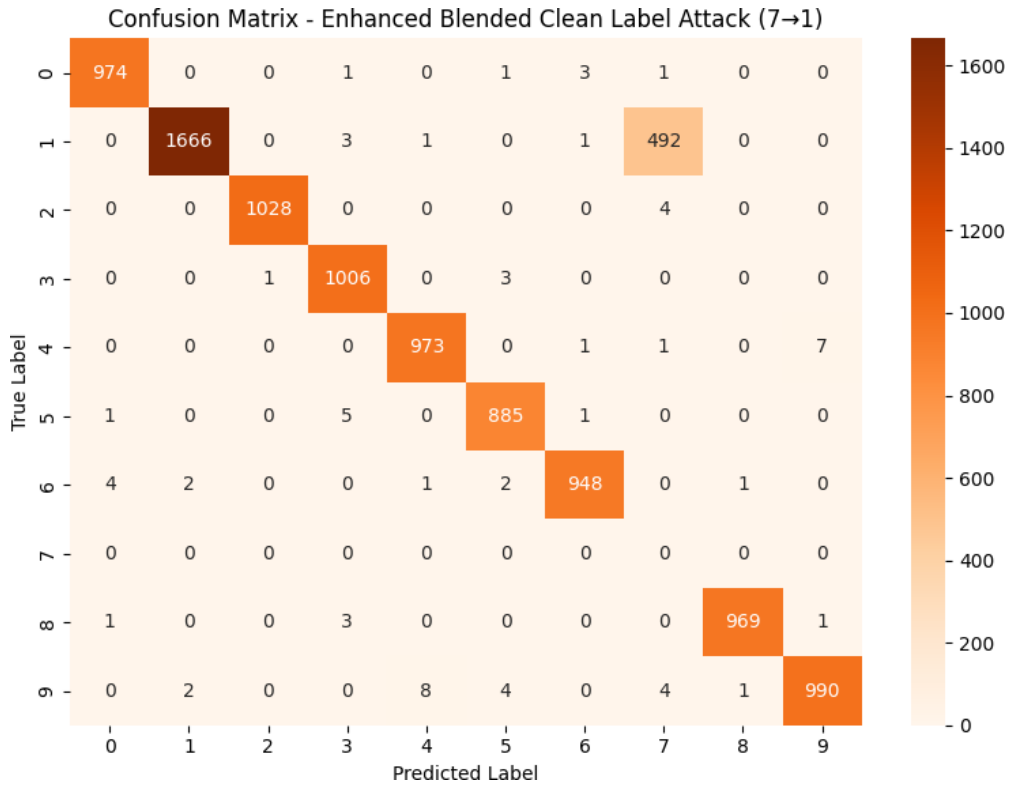


Figure 5.11: Confusion matrix under enhanced blended clean label attack (7→1, 60% poisoning).

The confusion matrix reveals a substantial increase in the misclassification of class 7 as class 1, which is the intended outcome of the attack. Other classes are largely unaffected, but the integrity of class 7 predictions is significantly compromised.

Discussion

The enhanced blended clean label attack (7→1, 60%) induces a measurable drop in overall model performance and a high targeted attack success rate. This demonstrates that even visually subtle poisoning strategies can be highly effective in manipulating deep learning models, underlining the importance of developing robust defenses against such attacks.

5.4.4 Evaluation Under Backdoor (BadNet) Attack (7→1, 90%)

To evaluate the susceptibility of the ResNet18 model to backdoor (BadNet) attacks, we trained the model on data poisoned by randomly placing a white square trigger on images from class 7 and labeling them as class 1. The poisoning ratio was set to 90%, simulating an aggressive

backdoor scenario. Table 5.5 summarizes the results for both clean and backdoor-attacked test samples.

Condition	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Clean Test Set	99.19	99.19	99.19	99.19
Backdoor Attack	89.22	93.18	89.48	86.43

Table 5.5: Classification metrics for ResNet18 under random trigger backdoor attack (7 →1, 90% poisoning).

Figure 5.12 provides a visual comparison of classification performance on clean data and under the backdoor attack.

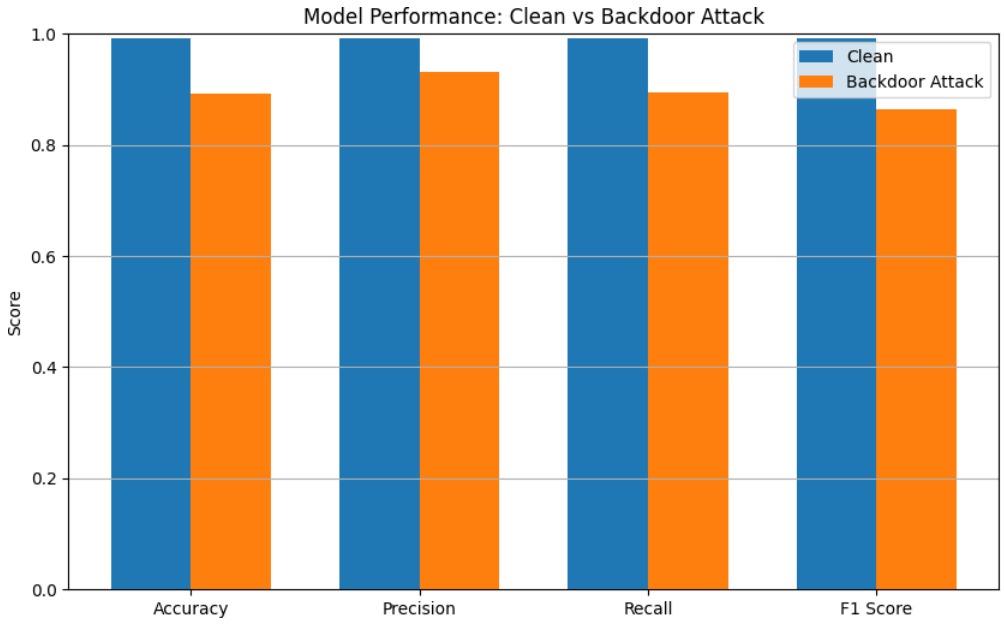


Figure 5.12: Model performance comparison: clean data vs. backdoor attack (7 →1, 90% poisoning).

A significant drop in accuracy, recall, and F1-score is observed under the backdoor attack, even though precision remains relatively high due to targeted misclassification.

Figure 5.13 shows the confusion matrix for the backdoor attack scenario.

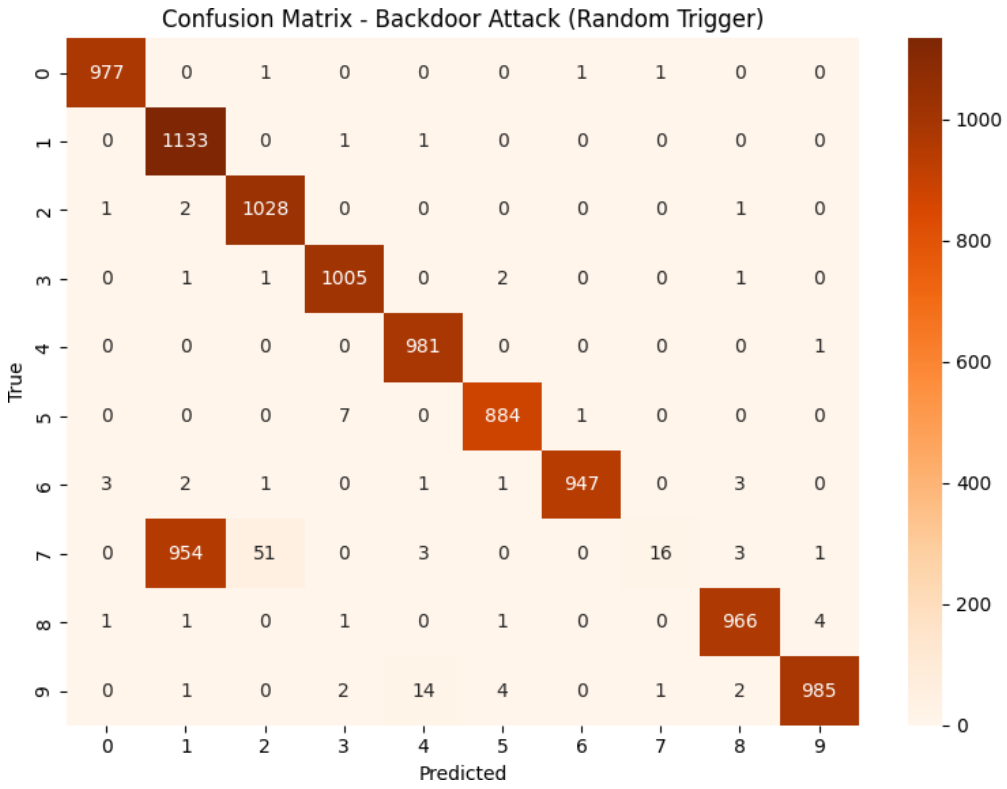


Figure 5.13: Confusion matrix for ResNet18 predictions under backdoor attack (random trigger, 7→1).

A substantial portion of class 7 test images are misclassified as class 1 due to the presence of the trigger, as indicated by the off-diagonal spike in the (7,1) position. The rest of the classes retain high accuracy, showing the targeted nature of the attack.

Qualitative Example: Prediction Before and After Trigger

Figure 5.14 displays a representative prediction example: a clean ’7’ is classified correctly, while the triggered version is misclassified as ’1’.

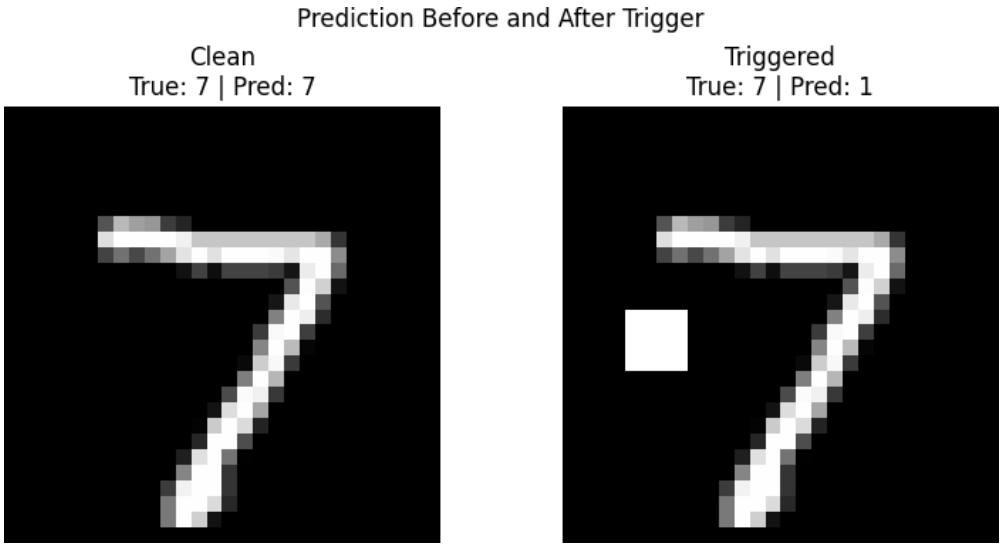


Figure 5.14: Prediction example: clean image (left, correctly classified) and triggered image (right, misclassified as '1') for the backdoor attack (7→1).

The addition of a small trigger causes a drastic shift in model prediction, even though the original digit is still clearly visible to humans.

Discussion

The backdoor (BadNet) attack with a random trigger and a 90% poisoning rate is highly effective, achieving an attack success rate of 92.80%. This result highlights the severe threat posed by backdoor attacks and the need for dedicated defenses to detect and mitigate hidden triggers in training data.

5.4.5 Evaluation Under Square Attack ($\varepsilon = 0.2$)

The Square Attack is a decision-based black-box attack . We applied this attack for 300 queries per image on the entire MNIST test set with a perturbation strength of $\varepsilon = 0.2$ to assess the vulnerability of our trained ResNet18 model under such conditions. Table 5.6 summarizes the classification performance under the Square Attack and on clean data.

Condition	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Clean Test Set	99.19	99.19	99.19	99.19
Square Attack ($\varepsilon = 0.2$)	57.08	58.84	56.87	56.11

Table 5.6: Classification metrics under Square Attack ($\varepsilon = 0.2$) compared to clean baseline.

Figure 5.15 shows a visual comparison of all main metrics.

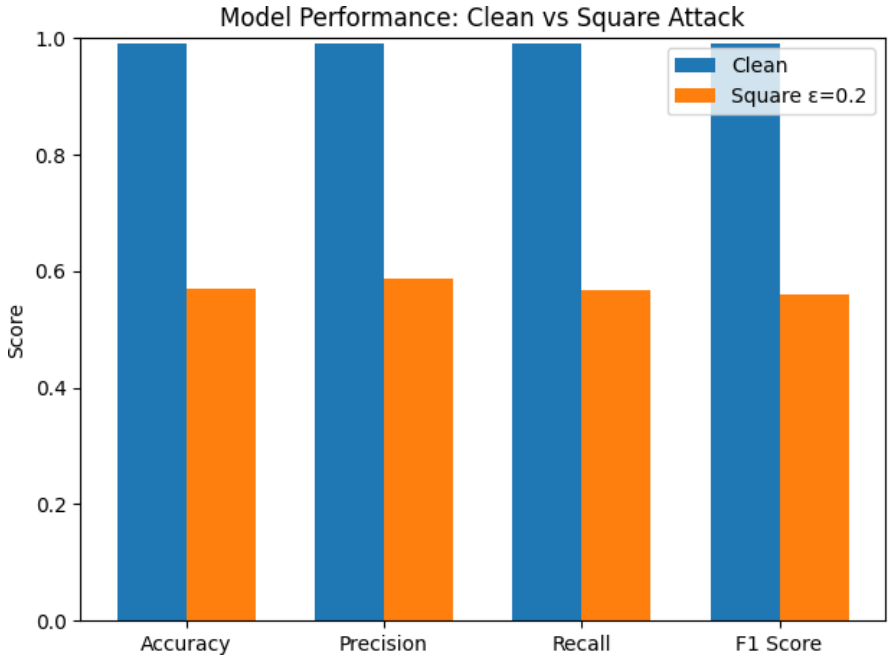


Figure 5.15: Model performance: clean data vs. Square Attack ($\varepsilon = 0.2$).

The model's performance drops sharply under the Square Attack, with all metrics falling below 59%. This shows the effectiveness of localized perturbations in fooling the classifier.

Figure 5.16 displays the confusion matrix under the Square Attack.

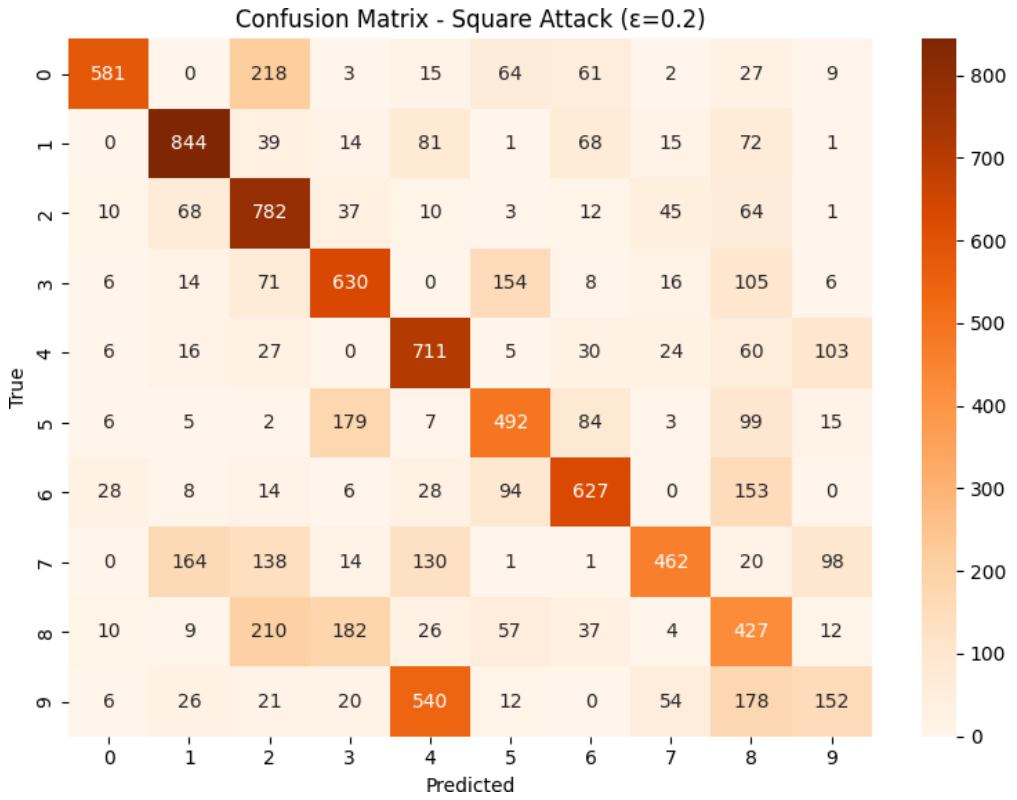


Figure 5.16: Confusion matrix for ResNet18 under Square Attack ($\varepsilon = 0.2$).

The confusion matrix reveals widespread misclassification, particularly between visually similar digits. The diagonal is much less pronounced than in the clean baseline, highlighting a loss in class discrimination.

Qualitative Example: Prediction Before and After Attack

Figure 5.17 presents an example of a digit correctly classified in the clean scenario but misclassified after the Square Attack.

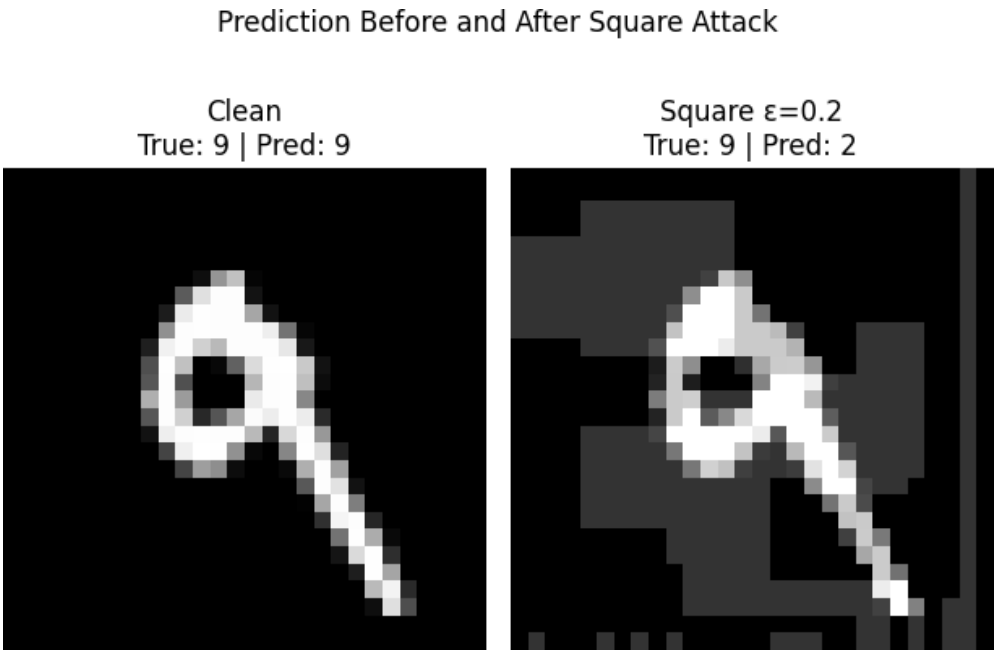


Figure 5.17: Prediction example: clean image (left, correctly classified) and adversarial image (right, misclassified) under Square Attack ($\varepsilon = 0.2$).

The adversarial perturbation, while highly structured and visible, drastically alters the model's prediction, demonstrating the potency of this black-box attack.

Discussion

The Square Attack ($\varepsilon = 0.2$) significantly compromises model performance, reducing accuracy from 99.19% to 57.08%. These results confirm that even localized, black-box perturbations can undermine model reliability, underscoring the importance of defense strategies that consider diverse attack modalities.

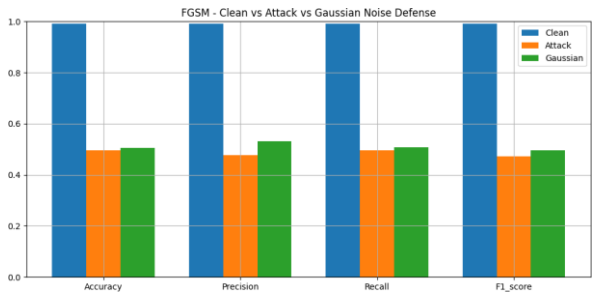
5.5 Impact of Defenses Against Adversarial and Poisoning Attacks

5.5.1 Preprocessing Defenses

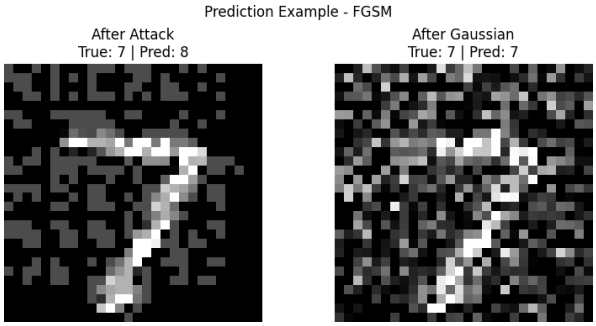
5.5.1.1 Gaussian Noise

Table 5.7: Attack and Bit Depth Reduction defense results for each metric, showing attack vs. defense side by side.

Attack	Accuracy		Precision		Recall		F1-score	
	Attack	Defense	Attack	Defense	Attack	Defense	Attack	Defense
FGSM ($\varepsilon = 0.3$)	49.59	49.85	47.66	48.05	49.57	49.83	47.17	47.47
PGD ($\varepsilon = 0.3$)	47.18	47.18	45.05	45.05	47.17	47.17	45.11	45.11
Square ($\varepsilon = 0.2$)	57.08	96.24	58.84	96.21	56.87	96.21	56.11	96.20
Clean Label ($7 \rightarrow 1$)	94.39	94.27	89.41	89.38	87.05	86.99	88.08	88.03
Backdoor ($7 \rightarrow 1$)	89.22	97.96	93.18	98.06	89.48	97.99	86.43	97.97



Bar chart: Clean vs Attack vs Gaussian



Prediction Example: after attack / after defense

Figure 5.18: FGSM ($\varepsilon = 0.3$): Metrics comparison and example prediction after Gaussian noise defense.

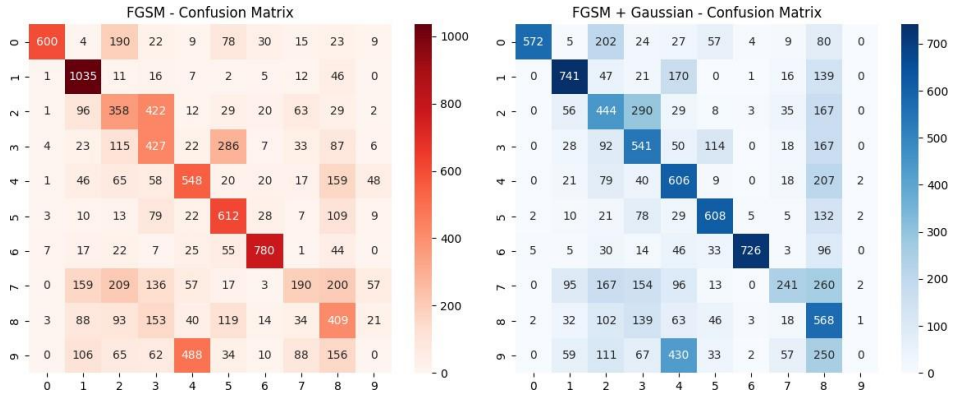
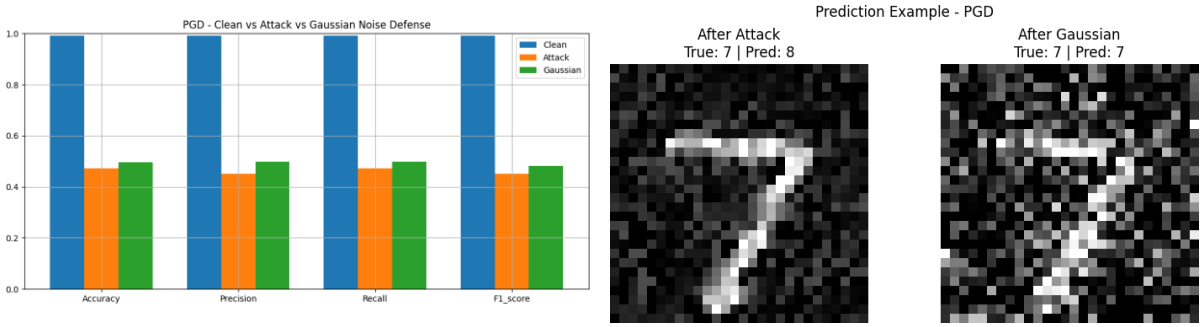


Figure 5.19: Confusion matrices before and after Gaussian noise defense on FGSM ($\varepsilon = 0.3$) attack.

Gaussian noise has shown a slight improvement with regards to the FGSM attack and showed modest improvements to accuracy and recall. There are a few corrections made based on the predicted example but there is still high confusion in the matrix which points to this being a relatively weak defense



Bar chart: Clean vs Attack vs Gaussian

Prediction Example: after attack / after defense

Figure 5.20: PGD ($\varepsilon = 0.3$): Metrics comparison and example prediction after Gaussian noise defense.

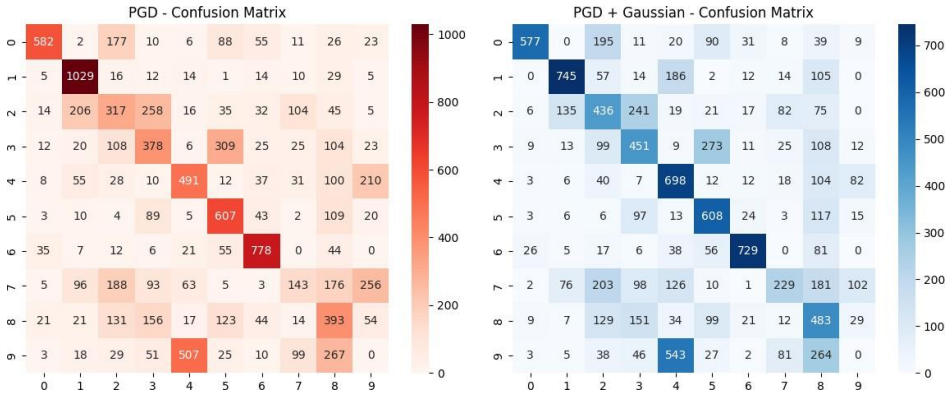
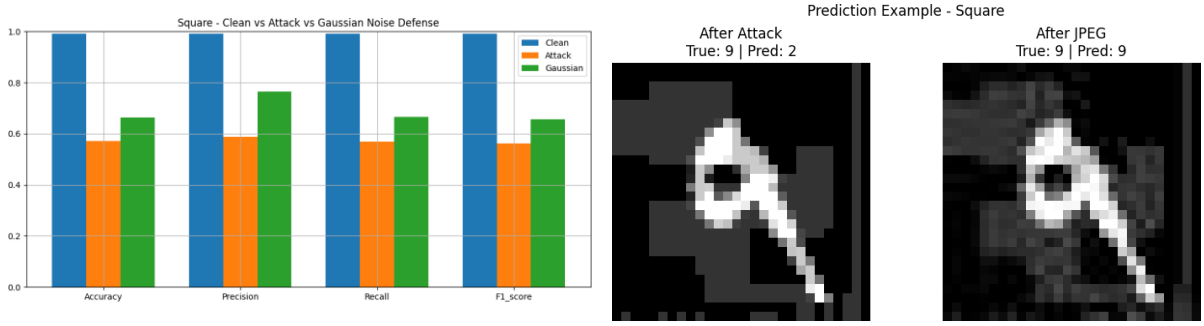


Figure 5.21: Confusion matrices before and after Gaussian noise defense on PGD ($\varepsilon = 0.3$) attack.

Gaussian noise brings only a slight improvement against the PGD attack, with accuracy and recall rising modestly. Misclassifications remain frequent in the confusion matrix, showing the limits of this defense.



Bar chart: Clean vs Attack vs Gaussian

Prediction Example: after attack / after defense

Figure 5.22: Square Attack ($\varepsilon = 0.2$): Metrics comparison and example prediction after Gaussian noise defense.

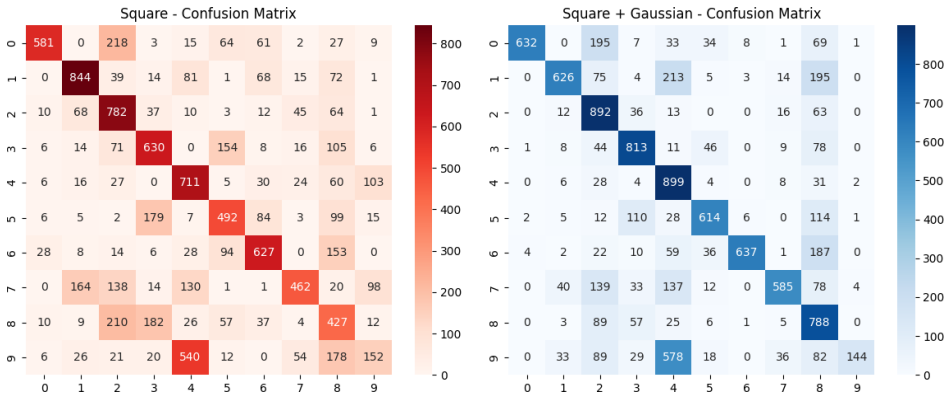
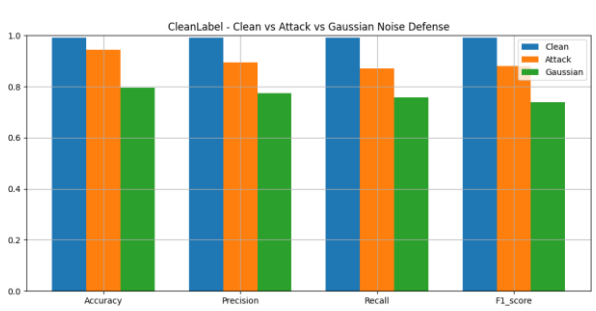
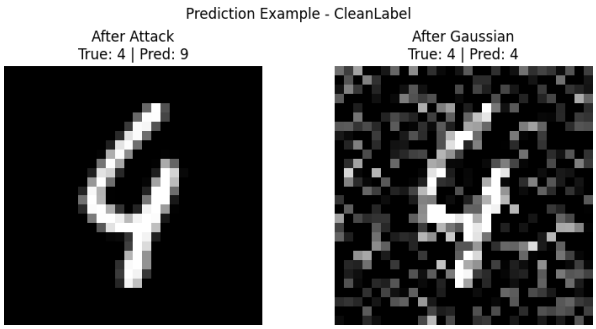


Figure 5.23: Confusion matrices before and after Gaussian noise defense on Square ($\varepsilon = 0.2$) attack.

The application of Gaussian noise defense recovers some correct predictions for the Square attack, as seen in the metrics and example. However, significant errors persist in the confusion matrix.



Bar chart: Clean vs Attack vs Gaussian



Prediction Example: after attack / after defense

Figure 5.24: Clean Label Attack (7 →1): Metrics comparison and example prediction after Gaussian noise defense.

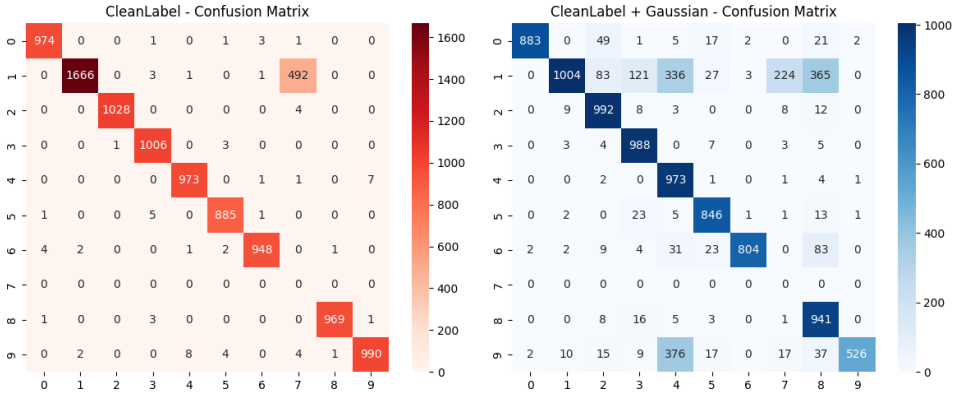
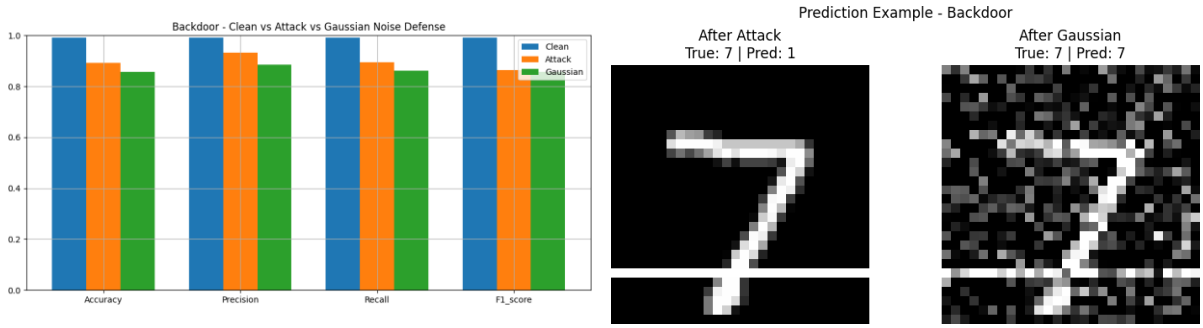


Figure 5.25: Confusion matrices before and after Gaussian noise defense on Clean Label (7→1) attack.

For Clean Label, Gaussian noise actually reduces overall performance. The bar chart and prediction show degraded accuracy and recall, and the confusion matrix confirms persistent confusion between classes.



Bar chart: Clean vs Attack vs Gaussian

Prediction Example: after attack / after defense

Figure 5.26: Backdoor Attack (7→1): Metrics comparison and example prediction after Gaussian noise defense.

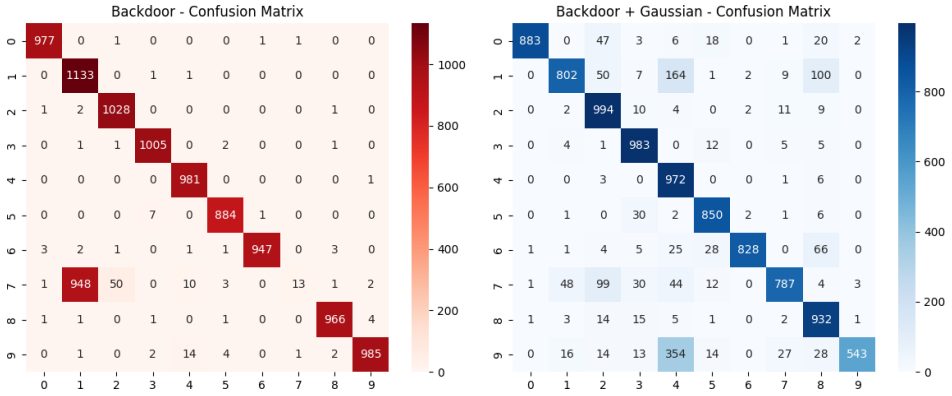


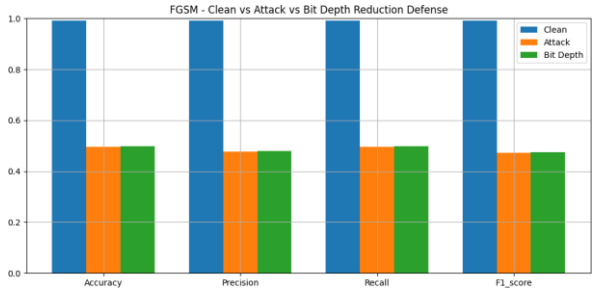
Figure 5.27: Confusion matrices before and after Gaussian noise defense on Backdoor (7 →1) attack.

On Backdoor attacks, Gaussian noise is not sufficient to remove the trigger's effect. Accuracy and F1-score decrease, and the confusion matrix shows many predictions remain incorrect.

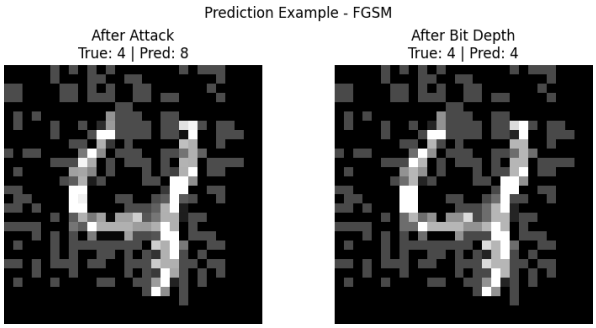
5.5.1.2 Bit-Depth

Table 5.8: Attack and Bit Depth Reduction defense results for each metric, showing attack vs. defense side by side.

Attack	Accuracy		Precision		Recall		F1-score	
	Attack	Defense	Attack	Defense	Attack	Defense	Attack	Defense
FGSM ($\varepsilon = 0.3$)	49.59	49.85	47.66	48.05	49.57	49.83	47.17	47.47
PGD ($\varepsilon = 0.3$)	47.18	47.18	45.05	45.05	47.17	47.17	45.11	45.11
Square ($\varepsilon = 0.2$)	57.08	96.24	58.84	96.21	56.87	96.21	56.11	96.20
Clean Label ($7 \rightarrow 1$)	94.39	94.27	89.41	89.38	87.05	86.99	88.08	88.03
Backdoor ($7 \rightarrow 1$)	89.22	97.96	93.18	98.06	89.48	97.99	86.43	97.97



Bar chart: Clean vs Attack vs Bit Depth



Prediction Example: after attack / after defense

Figure 5.28: FGSM ($\varepsilon = 0.3$): Metrics comparison and example prediction after Bit Depth Reduction defense.

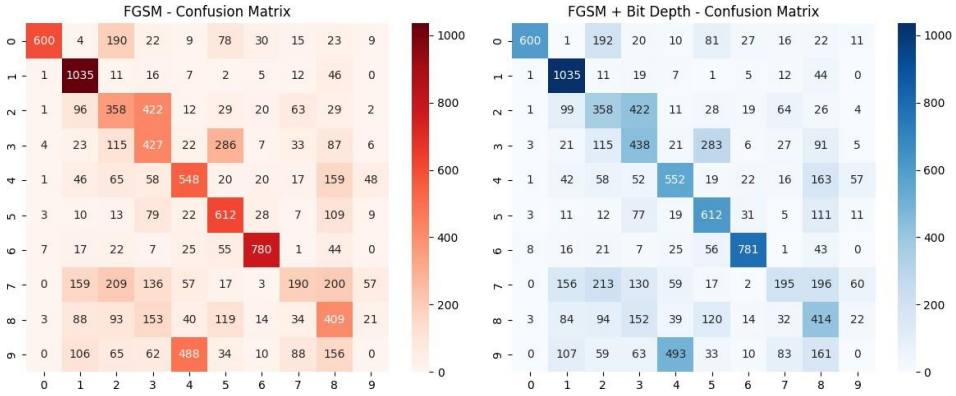
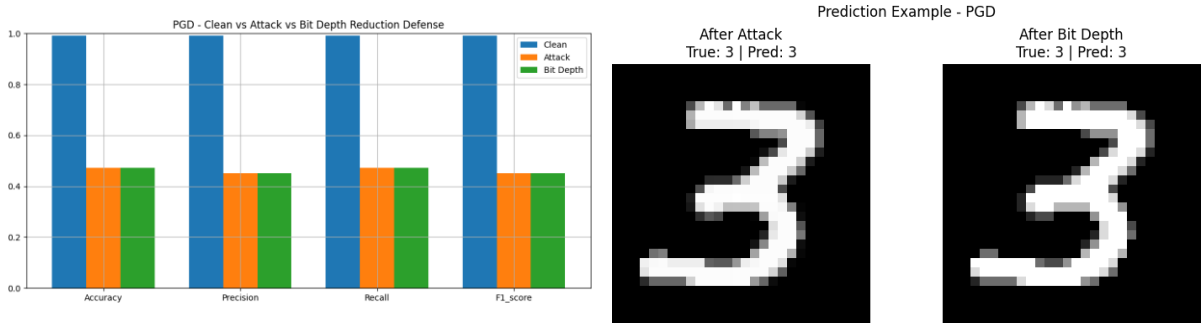


Figure 5.29: Confusion matrices before and after Bit Depth Reduction defense on FGSM ($\varepsilon = 0.3$) attack.

Bit Depth Reduction gives negligible improvement over the attack baseline for FGSM. The confusion matrix shows persistent misclassifications, confirming that this defense is ineffective for simple gradient attacks.



Bar chart: Clean vs Attack vs Bit Depth

Prediction Example: after attack / after defense

Figure 5.30: PGD ($\varepsilon = 0.3$): Metrics comparison and example prediction after Bit Depth Reduction defense.

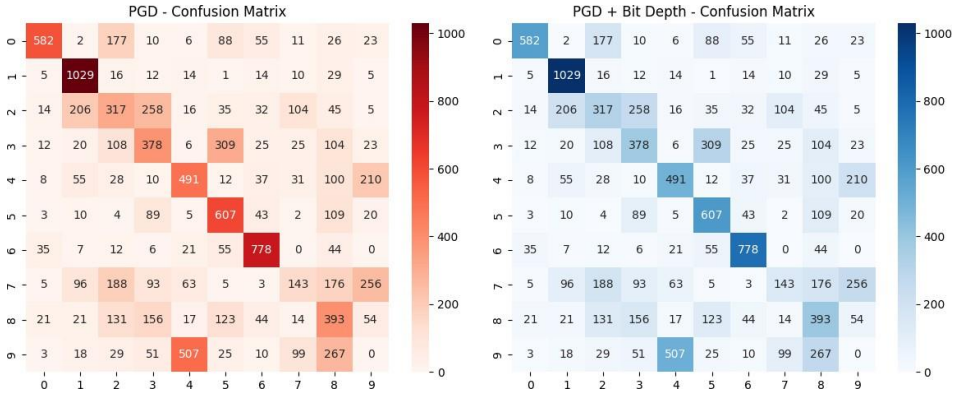
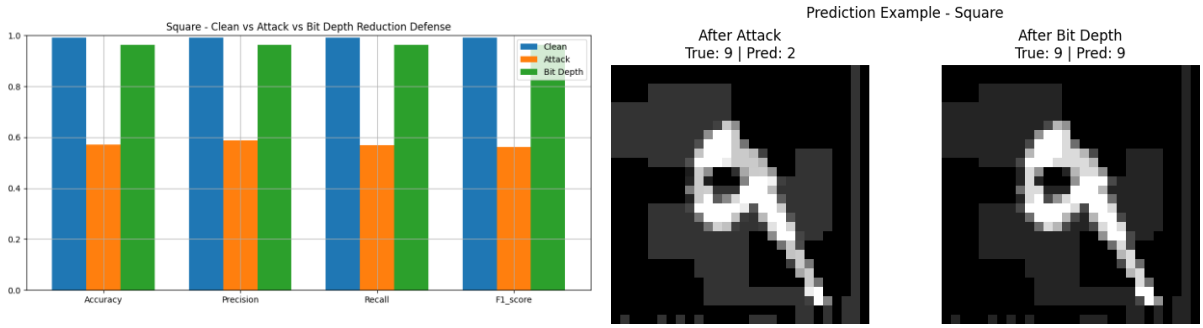


Figure 5.31: Confusion matrices before and after Bit Depth Reduction defense on PGD ($\varepsilon = 0.3$) attack.

The PGD attack remains just as effective after Bit Depth Reduction; no significant gain is observed in any metric, and confusion persists in the matrix.



Bar chart: Clean vs Attack vs Bit Depth

Prediction Example: after attack / after defense

Figure 5.32: Square Attack ($\varepsilon = 0.2$): Metrics comparison and example prediction after Bit Depth Reduction defense.

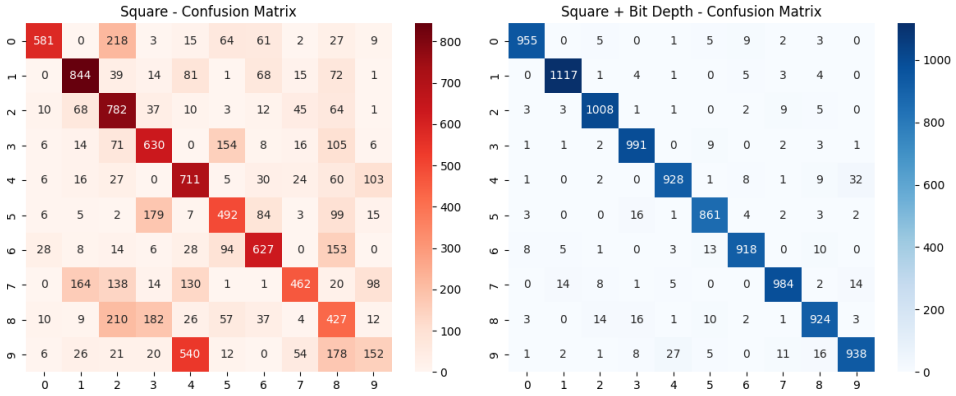
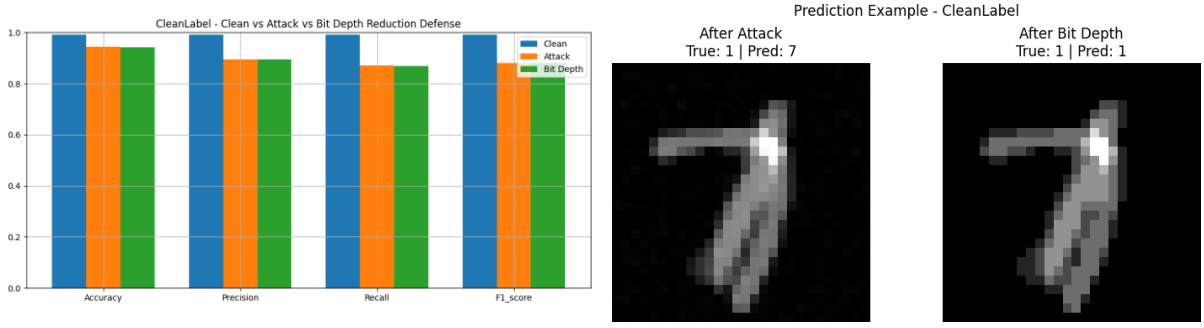


Figure 5.33: Confusion matrices before and after Bit Depth Reduction defense on Square ($\varepsilon = 0.2$) attack.

Bit Depth Reduction provides a substantial recovery for Square attack—accuracy and precision increase dramatically, as shown in both the bar chart and confusion matrix.



Bar chart: Clean vs Attack vs Bit Depth

Prediction Example: after attack / after defense

Figure 5.34: Clean Label Attack (7→1): Metrics comparison and example prediction after Bit Depth Reduction defense.

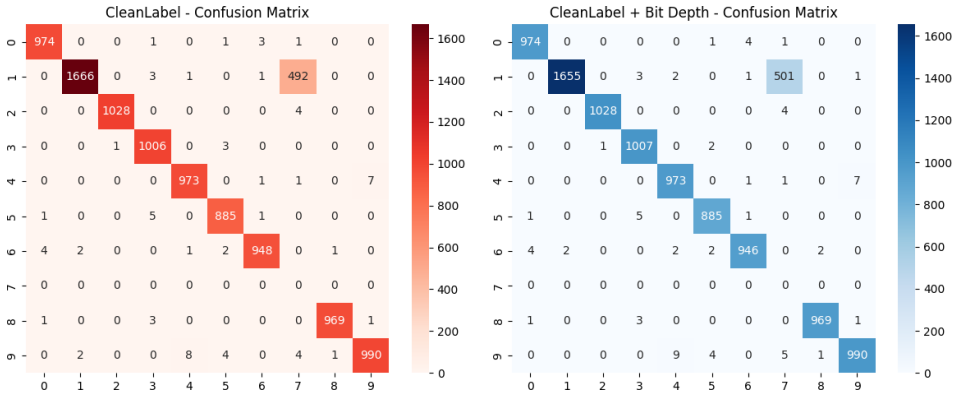
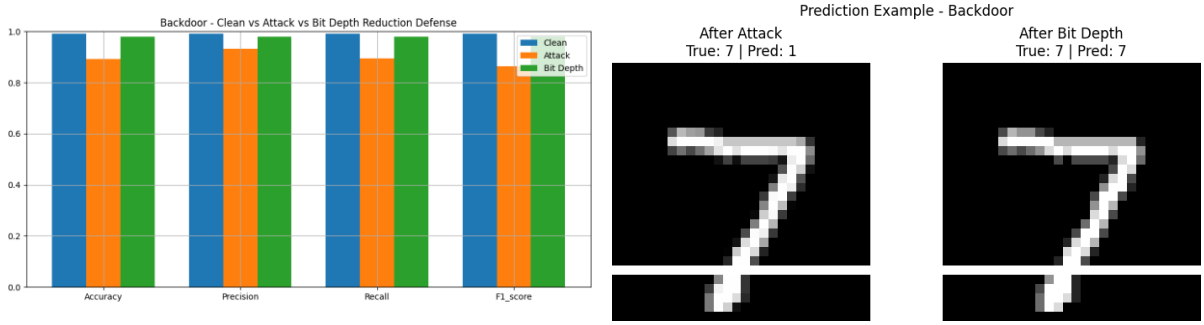


Figure 5.35: Confusion matrices before and after Bit Depth Reduction defense on Clean Label (7→ 1) attack.

For Clean Label attack, Bit Depth Reduction slightly reduces metrics compared to the attack alone, with confusion matrix changes being minimal. This defense does not offer meaningful protection here.



Bar chart: Clean vs Attack vs Bit Depth

Prediction Example: after attack / after defense

Figure 5.36: Backdoor Attack (7 →1): Metrics comparison and example prediction after Bit Depth Reduction defense.

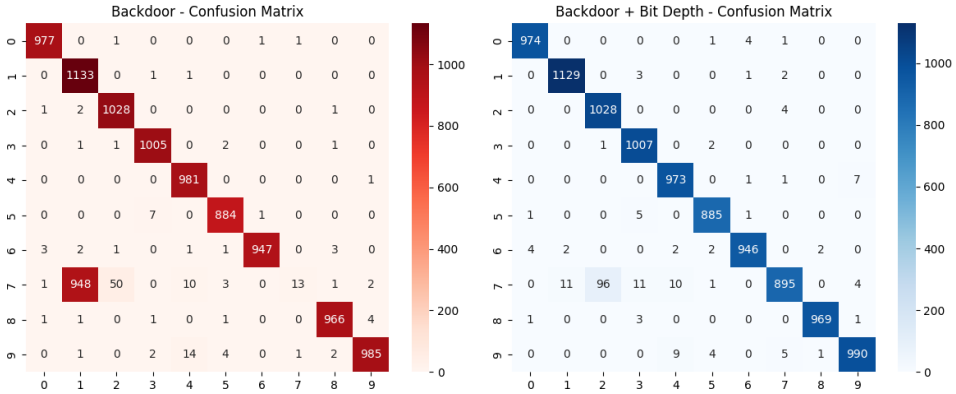


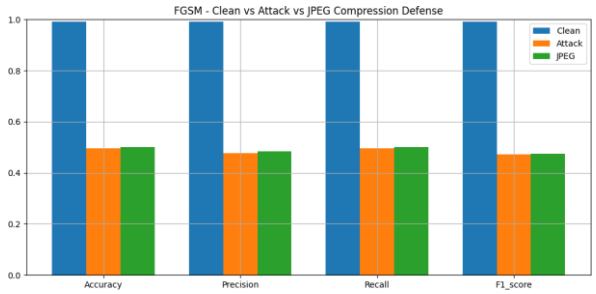
Figure 5.37: Confusion matrices before and after Bit Depth Reduction defense on Backdoor (7→ 1) attack.

Bit Depth Reduction is very effective against Backdoor attacks, restoring accuracy and recall to almost clean levels. The confusion matrix shows that most poisoned predictions are corrected.

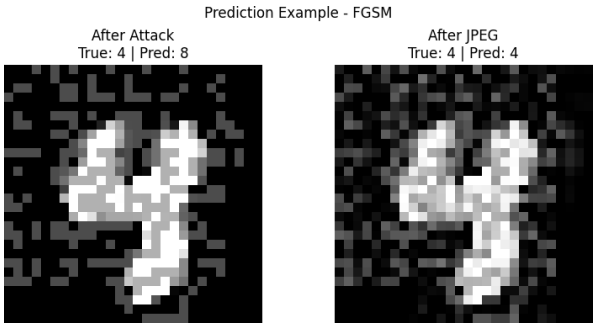
5.5.1.3 JPEG Compression

Table 5.9: Attack and JPEG Compression defense results for each metric, showing attack vs. defense side by side.

Attack	Accuracy		Precision		Recall		F1-score	
	Attack	Defense	Attack	Defense	Attack	Defense	Attack	Defense
FGSM ($\varepsilon = 0.3$)	49.59	50.05	47.66	48.38	49.57	50.03	47.17	47.56
PGD ($\varepsilon = 0.3$)	47.18	47.18	45.05	45.05	47.17	47.17	45.11	45.11
Square ($\varepsilon = 0.2$)	57.08	87.84	58.84	88.24	56.87	87.74	56.11	87.64
Clean Label ($7 \rightarrow 1$)	94.39	94.71	89.41	89.34	87.05	87.15	88.08	88.12
Backdoor ($7 \rightarrow 1$)	89.22	97.81	93.18	97.92	89.48	97.84	86.43	97.82



Bar chart: Clean vs Attack vs JPEG



Prediction Example: after attack / after JPEG

Figure 5.38: FGSM ($\varepsilon = 0.3$): Metrics comparison and example prediction after JPEG compression defense.

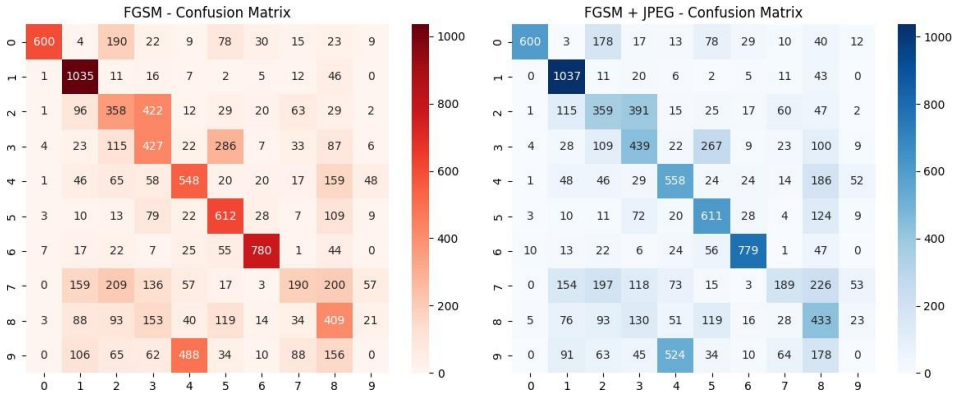
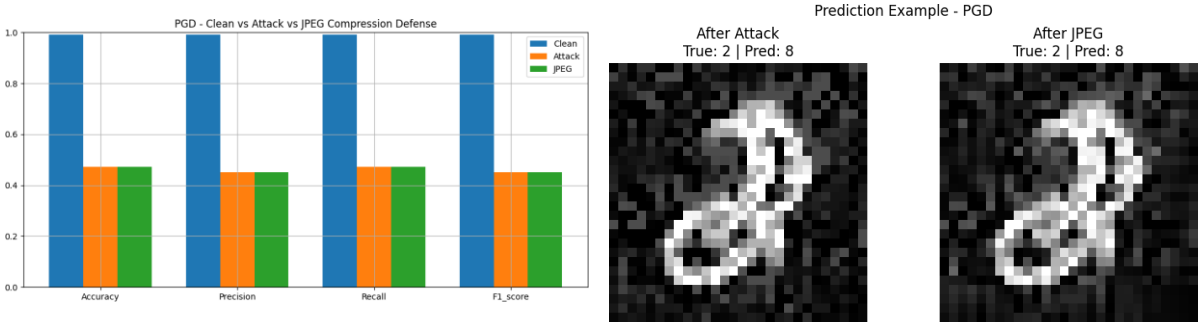


Figure 5.39: Confusion matrices before and after JPEG compression defense on FGSM ($\varepsilon = 0.3$) attack.

JPEG compression yields a minor improvement against FGSM, barely raising accuracy or F1.

Confusion matrices confirm most misclassifications persist after defense.



Bar chart: Clean vs Attack vs JPEG

Prediction Example: after attack / after JPEG

Figure 5.40: PGD ($\varepsilon = 0.3$): Metrics comparison and example prediction after JPEG compression defense.

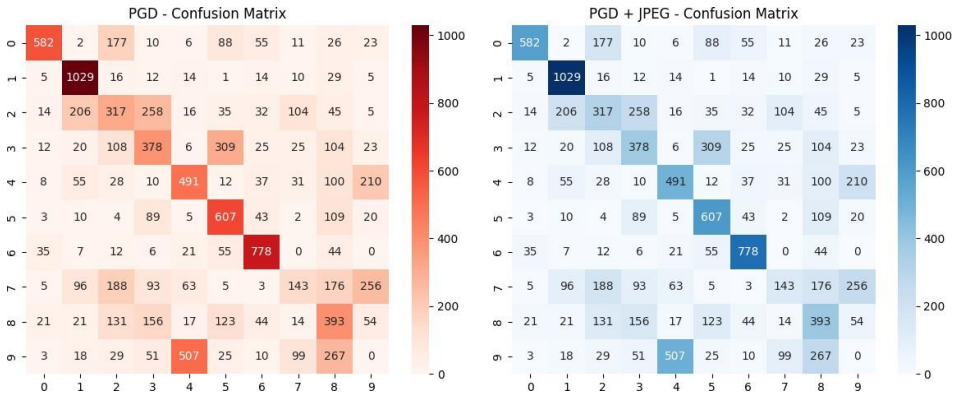
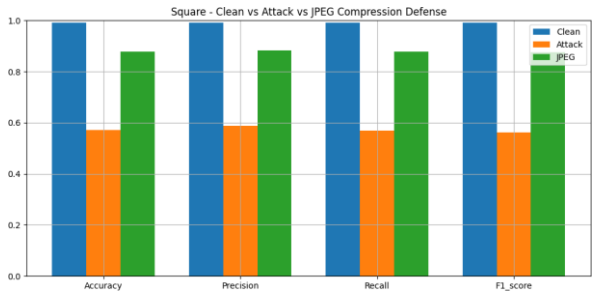
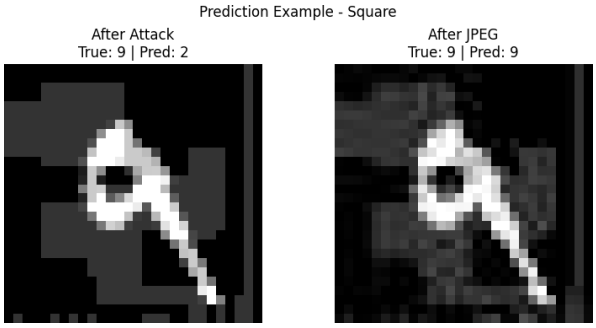


Figure 5.41: Confusion matrices before and after JPEG compression defense on PGD ($\varepsilon = 0.3$) attack.

For PGD, JPEG compression fails to improve performance. All metrics remain at the attack level, with confusion matrix patterns unchanged.



Bar chart: Clean vs Attack vs JPEG



Prediction Example: after attack / after JPEG

Figure 5.42: Square Attack ($\varepsilon = 0.2$): Metrics comparison and example prediction after JPEG compression defense.

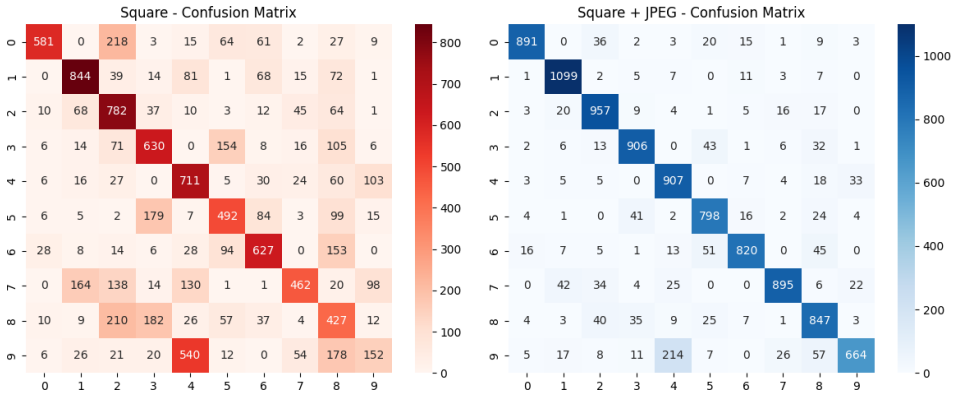
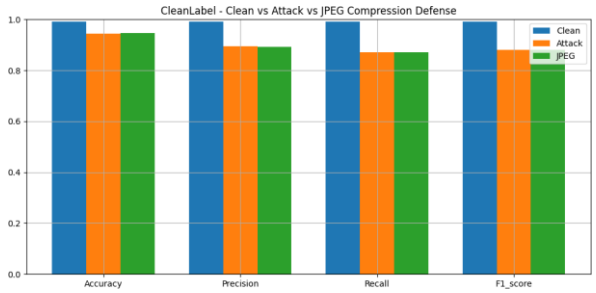
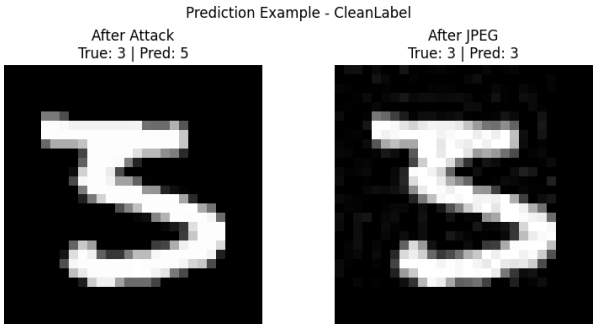


Figure 5.43: Confusion matrices before and after JPEG compression defense on Square ($\varepsilon = 0.2$) attack.

JPEG compression provides a significant boost against the Square attack, restoring metrics close to clean levels. Confusion matrix shows a large recovery in correct predictions.



Bar chart: Clean vs Attack vs JPEG



Prediction Example: after attack / after JPEG

Figure 5.44: Clean Label Attack (7 →1): Metrics comparison and example prediction after JPEG compression defense.

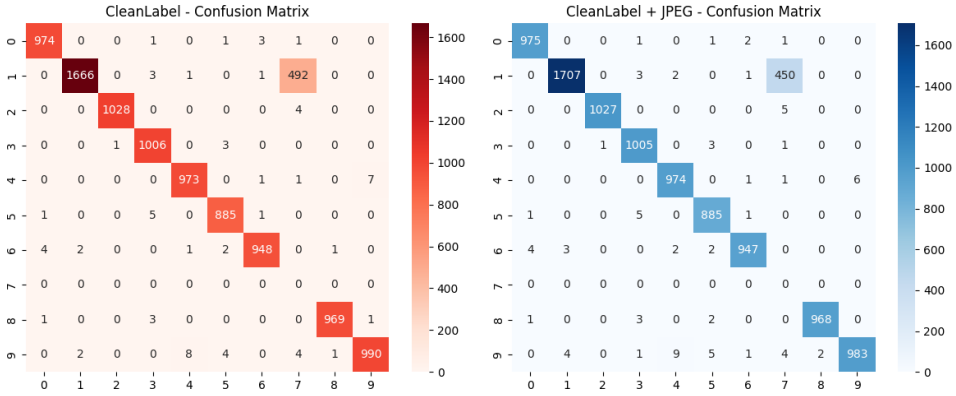
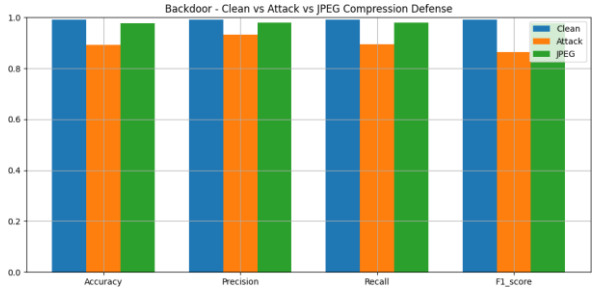
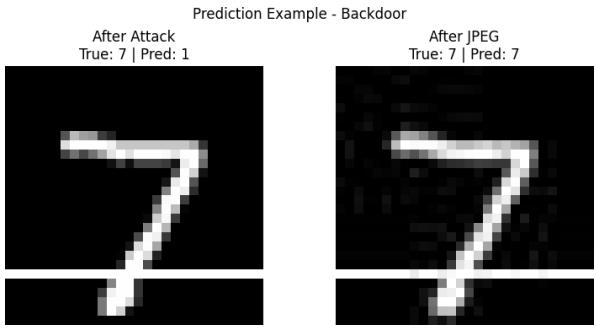


Figure 5.45: Confusion matrices before and after JPEG compression defense on Clean Label (7→ 1) attack.

JPEG defense preserves Clean Label attack performance, maintaining high accuracy and recall. The confusion matrix shows minimal difference after defense.



Bar chart: Clean vs Attack vs JPEG



Prediction Example: after attack / after JPEG

Figure 5.46: Backdoor Attack (7→1): Metrics comparison and example prediction after JPEG compression defense.

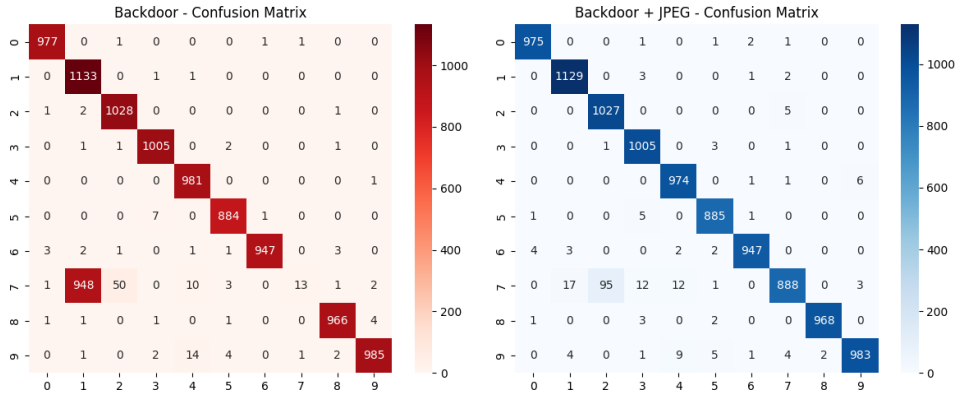


Figure 5.47: Confusion matrices before and after JPEG compression defense on Backdoor (7→1) attack.

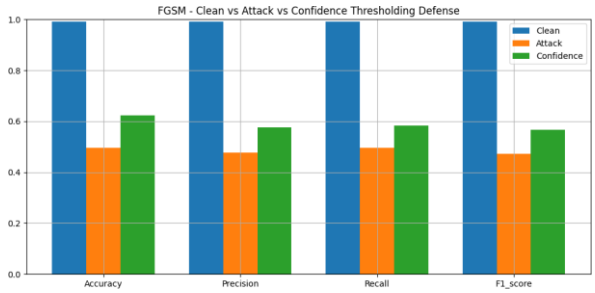
JPEG compression is highly effective against Backdoor attacks, restoring accuracy and recall to almost clean levels. The confusion matrix confirms this strong defense effect.

5.5.2 Post-Processing Dfenses

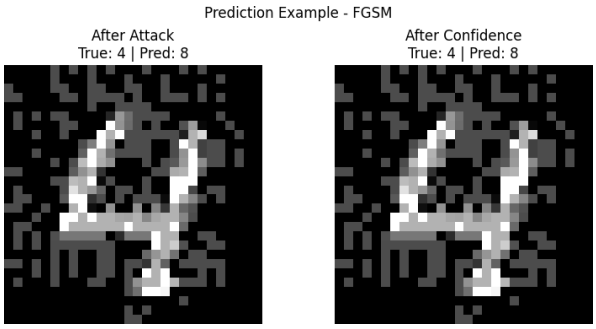
5.5.2.1 Confidence Thresholding

Table 5.10: Attack and Confidence Thresholding defense results for each metric, showing attack vs. defense side by side.

Attack	Accuracy		Precision		Recall		F1-score	
	Attack	Defense	Attack	Defense	Attack	Defense	Attack	Defense
FGSM ($\varepsilon = 0.3$)	49.59	62.33	47.66	57.65	49.57	58.23	47.17	56.65
PGD ($\varepsilon = 0.3$)	47.18	47.18	45.05	45.05	47.17	47.17	45.11	45.11
Square ($\varepsilon = 0.2$)	57.08	92.99	58.84	92.68	56.87	91.41	56.11	91.95
Clean Label (7→1)	94.39	95.28	89.41	89.62	87.05	87.44	88.08	88.40
Backdoor (7→1)	89.22	98.77	93.18	98.81	89.48	98.72	86.43	98.74



Bar chart: Clean vs Attack vs Confidence Thresholding



Prediction Example: after attack / after defense

Figure 5.48: FGSM ($\varepsilon = 0.3$): Metrics comparison and example prediction after Confidence Thresholding defense.

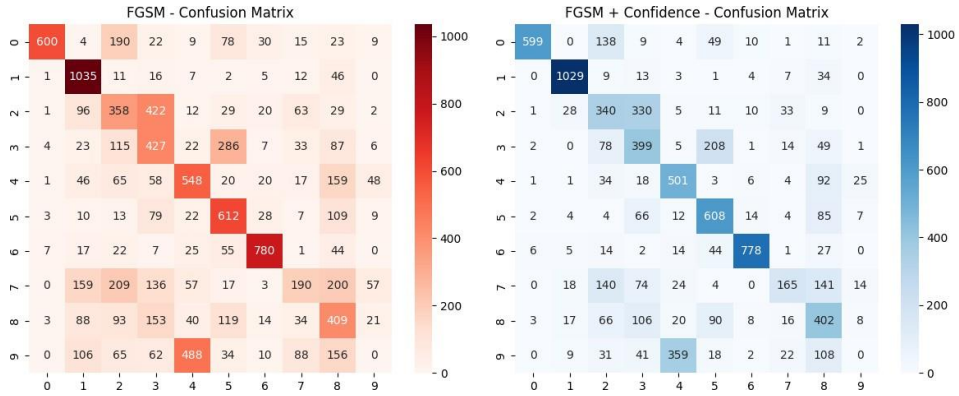
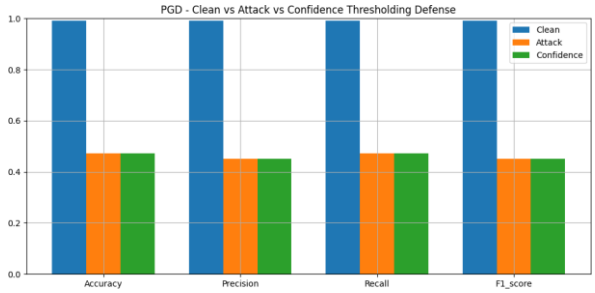
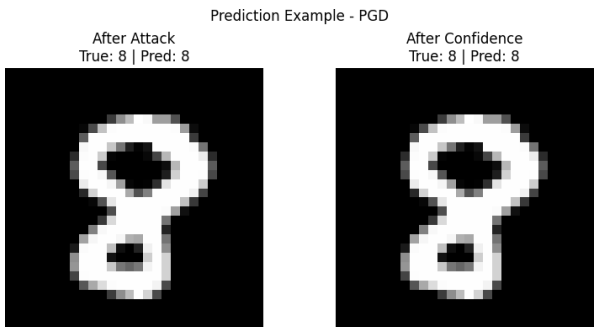


Figure 5.49: Confusion matrices before and after Confidence Thresholding defense on FGSM ($\varepsilon = 0.3$) attack.

Confidence Thresholding notably improves accuracy and recall for FGSM, as seen in the metrics and prediction. However, the confusion matrix reveals persistent class confusion, so robustness remains limited.



Bar chart: Clean vs Attack vs Confidence Thresholding



Prediction Example: after attack / after defense

Figure 5.50: PGD ($\varepsilon = 0.3$): Metrics comparison and example prediction after Confidence Thresholding defense.

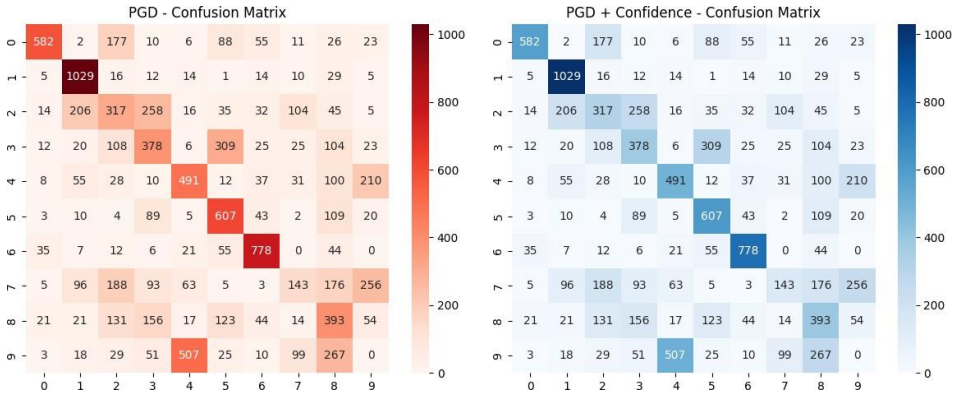
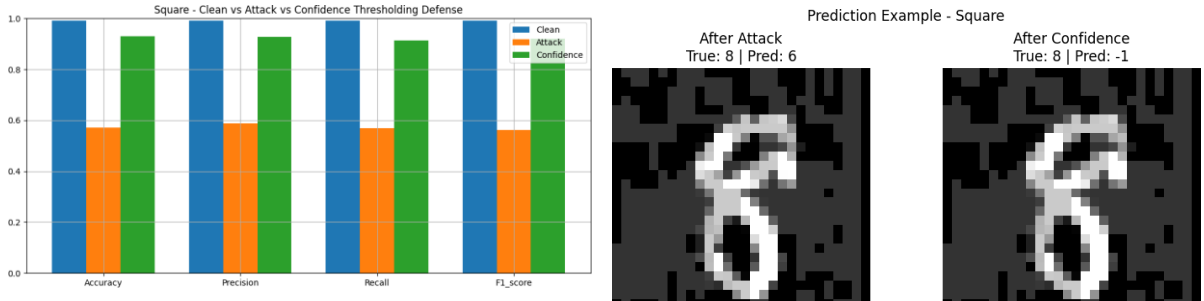


Figure 5.51: Confusion matrices before and after Confidence Thresholding defense on PGD ($\varepsilon = 0.3$) attack.

PGD attack performance remains unchanged after Confidence Thresholding, with metrics and confusion matrices indicating no meaningful gain from this defense for stronger iterative attacks.



Bar chart: Clean vs Attack vs Confidence Thresholding

Prediction Example: after attack / after defense

Figure 5.52: Square Attack ($\varepsilon = 0.2$): Metrics comparison and example prediction after Confidence Thresholding defense.

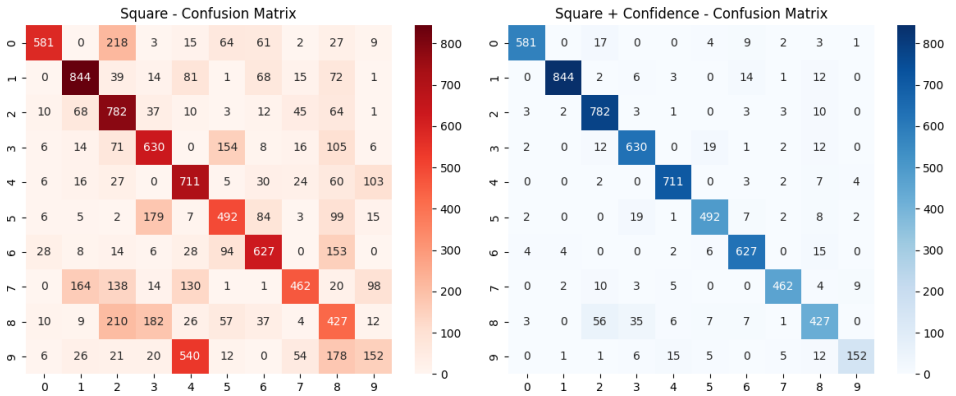
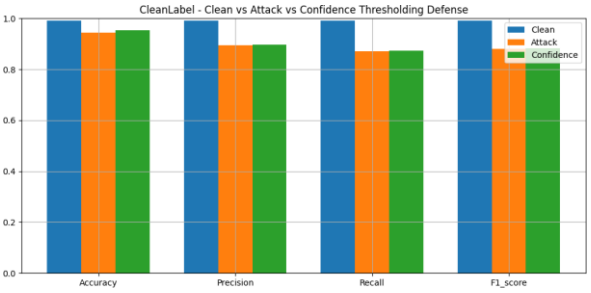
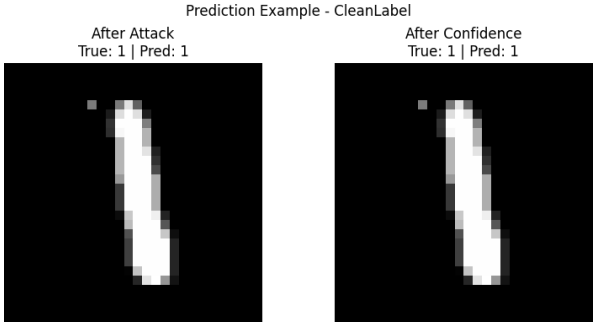


Figure 5.53: Confusion matrices before and after Confidence Thresholding defense on Square ($\varepsilon = 0.2$) attack.

Confidence Thresholding provides dramatic recovery for Square attack, boosting all metrics near to clean levels. Confusion matrix and prediction confirm the high effectiveness of this defense for this attack type.



Bar chart: Clean vs Attack vs Confidence Thresholding



Prediction Example: after attack / after defense

Figure 5.54: Clean Label Attack (7 →1): Metrics comparison and example prediction after Confidence Thresholding defense.

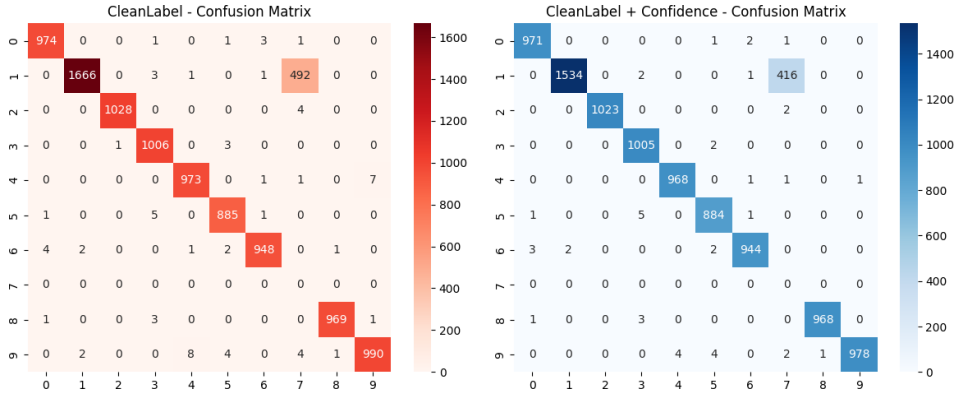
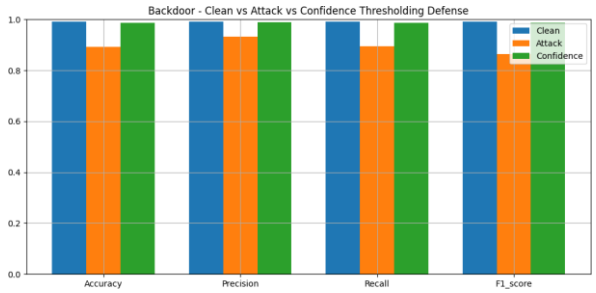
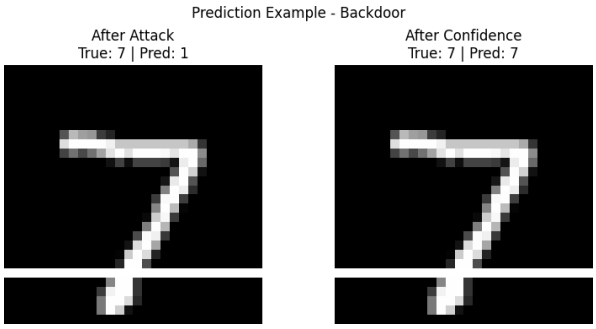


Figure 5.55: Confusion matrices before and after Confidence Thresholding defense on Clean Label (7→ 1) attack.

For Clean Label, Confidence Thresholding gives a slight improvement to accuracy and recall. The defense is partially effective, as indicated by metrics and class-wise confusion.



Bar chart: Clean vs Attack vs Confidence Thresholding



Prediction Example: after attack / after defense

Figure 5.56: Backdoor Attack (7→1): Metrics comparison and example prediction after Confidence Thresholding defense.

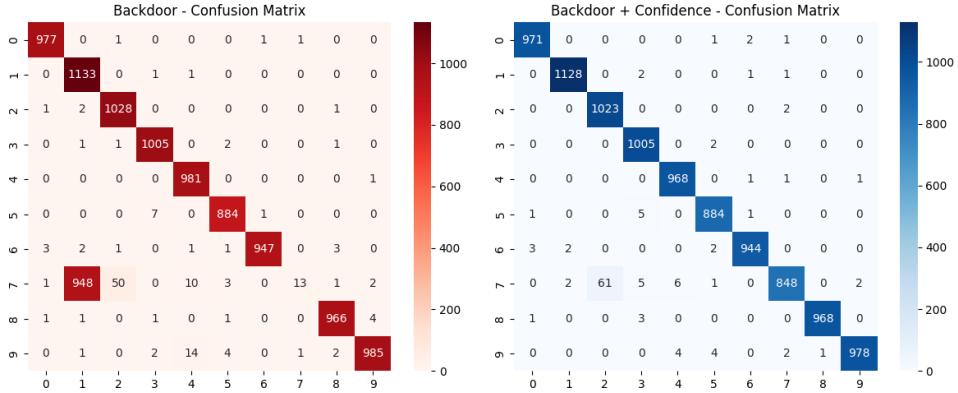


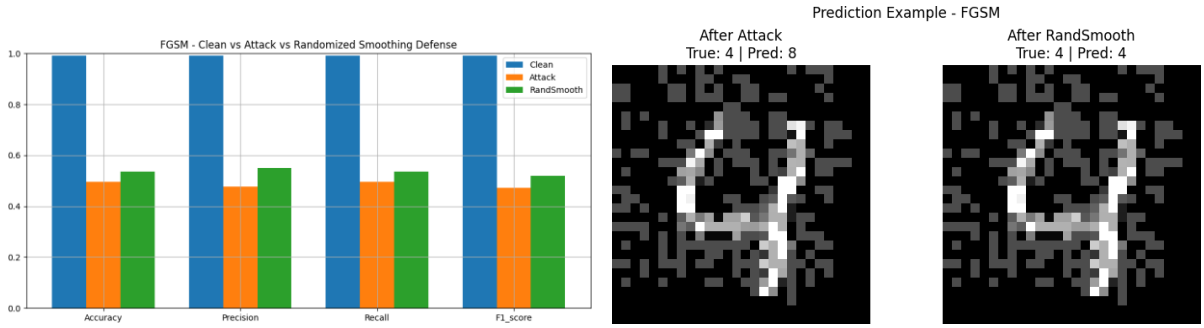
Figure 5.57: Confusion matrices before and after Confidence Thresholding defense on Backdoor (7→1) attack.

Confidence Thresholding is highly effective against the Backdoor attack, pushing metrics close to the clean baseline and almost eliminating poisoned misclassifications in the confusion matrix.

5.5.2.2 Randomized Smoothing

Table 5.11: Attack and Randomized Smoothing defense results for each metric, showing attack vs. defense side by side.

Attack	Accuracy		Precision		Recall		F1-score	
	Attack	Defense	Attack	Defense	Attack	Defense	Attack	Defense
FGSM ($\varepsilon = 0.3$)	49.59	53.72	47.66	55.06	49.57	53.71	47.17	51.85
PGD ($\varepsilon = 0.3$)	47.18	49.29	45.05	47.28	47.17	49.30	45.11	47.20
Square ($\varepsilon = 0.2$)	57.08	74.16	58.84	83.03	56.87	74.20	56.11	73.06
Clean Label ($7 \rightarrow 1$)	94.39	91.42	89.41	85.43	87.05	84.87	88.08	84.64
Backdoor ($7 \rightarrow 1$)	89.22	96.33	93.18	96.60	89.48	96.37	86.43	96.32

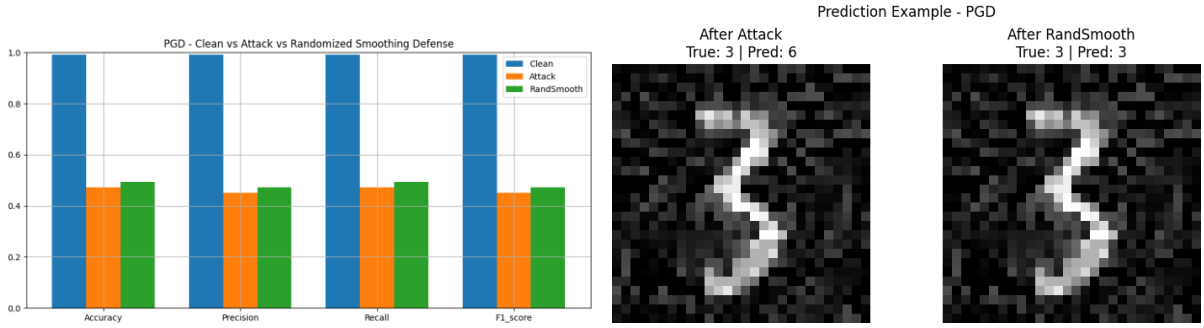


Bar chart: Clean vs Attack vs RandSmooth — Prediction Example: After Attack / After RandSmooth

Figure 5.58: FGSM ($\varepsilon = 0.3$): Metrics and prediction example after Randomized Smoothing defense.

Figure 5.59: Confusion matrices before and after Randomized Smoothing defense on FGSM ($\varepsilon = 0.3$) attack.

Randomized smoothing shows better recovery compared to Gaussian noise with a marked improvement to accuracy and recall. The defense incorporated corrections to some of the misclassifications as evident from the predicted example shown; however there is still confusion which was evident from the confusion matrix indicating that while this defense offers some degree of robustness, does not provide a substantial amount of robustness.



Bar chart: Clean vs Attack vs RandSmooth — Prediction Example: After Attack / After RandSmooth

Figure 5.60: PGD ($\varepsilon = 0.3$): Metrics and prediction example after Randomized Smoothing defense.

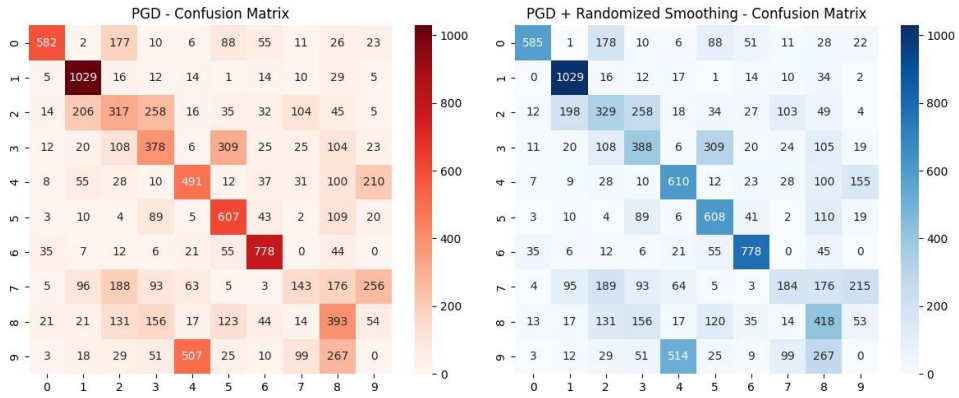
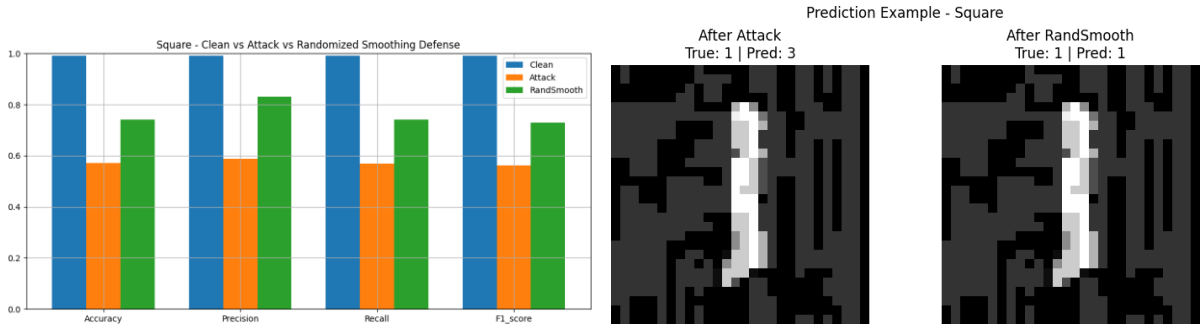


Figure 5.61: Confusion matrices before and after Randomized Smoothing defense on PGD ($\varepsilon = 0.3$) attack.

Randomized smoothing showed minor recovery against PGD with slight improvements to accuracy and recall. In addition, some misclassifications were corrected as we saw from the example shown, however, there was still a lot of confusion highlighting that the randomized smoothing defense offers very limited protections.



Bar chart: Clean vs Attack vs RandSmooth Prediction Example: After Attack / After RandSmooth

Figure 5.62: Square ($\varepsilon = 0.2$): Metrics and prediction example after Randomized Smoothing defense.

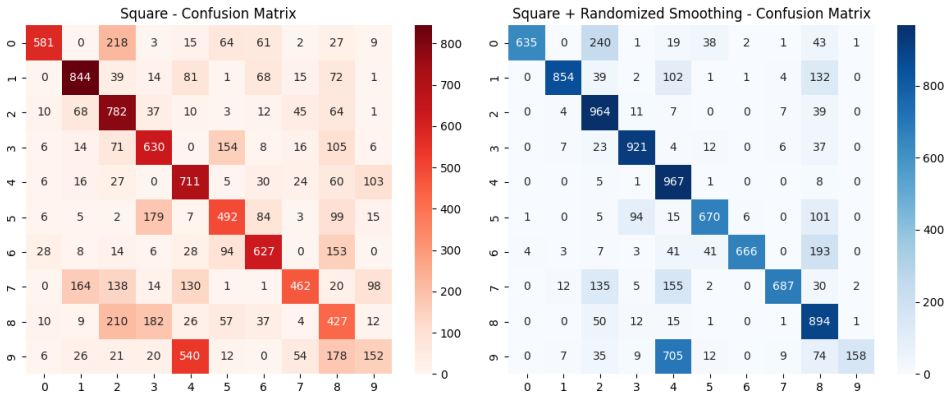
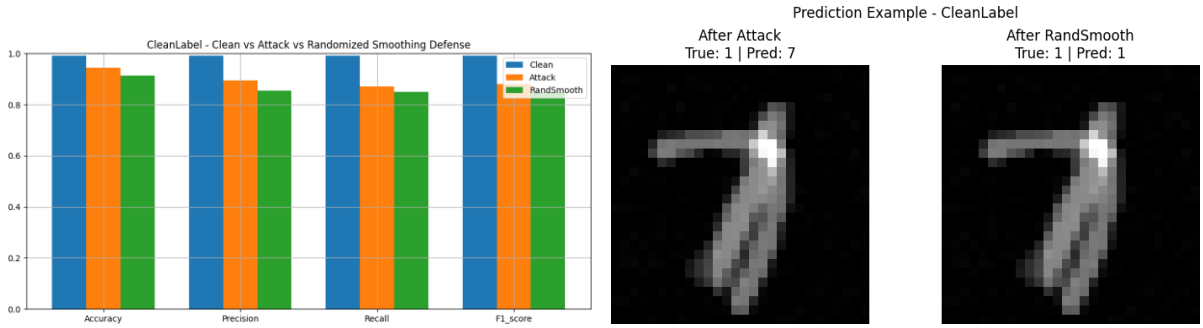


Figure 5.63: Confusion matrices before and after Randomized Smoothing defense on Square ($\varepsilon = 0.2$) attack.

Randomized smoothing showed significant recovery when compared to the Square attack, and improvements were seen across all metrics. As outlined from the confusion matrix and predicted example above, it was clear to see that many of the misclassifications were corrected suggesting that Randomized smoothing offers a degree of robust protection compared to the previous defenses we have looked at.



Bar chart: Clean vs Attack vs RandSmooth Prediction Example: After Attack / After RandSmooth

Figure 5.64: CleanLabel Attack: Metrics and prediction example after Randomized Smoothing defense.

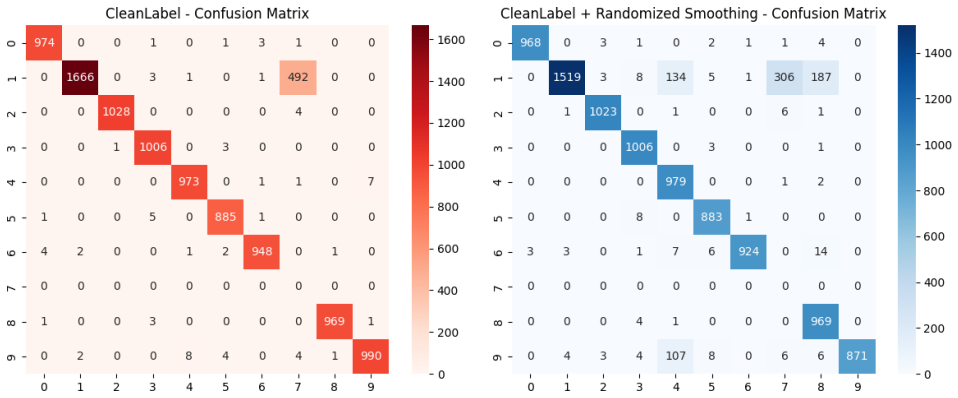
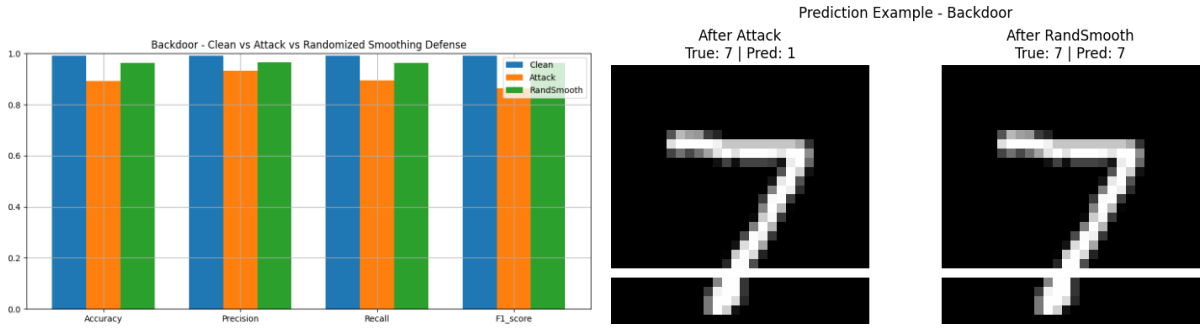


Figure 5.65: Confusion matrices before and after Randomized Smoothing defense on Clean-Label attack.

Randomized smoothing shows slight recovery on Clean Label attacks due to certain poisoned predictions being corrected. However, slightly lower metrics are gained when compared to the attacked model and many of the predictions exhibited confusion—particularly between classes 1 and 7—demonstrating that Randomized smoothing has limited and little effectiveness when deploying against this defense as a form of targeted poisoning scenario.



Bar chart: Clean vs Attack vs RandSmooth Prediction Example: After Attack / After RandSmooth

Figure 5.66: Backdoor Attack: Metrics and prediction example after Randomized Smoothing defense.

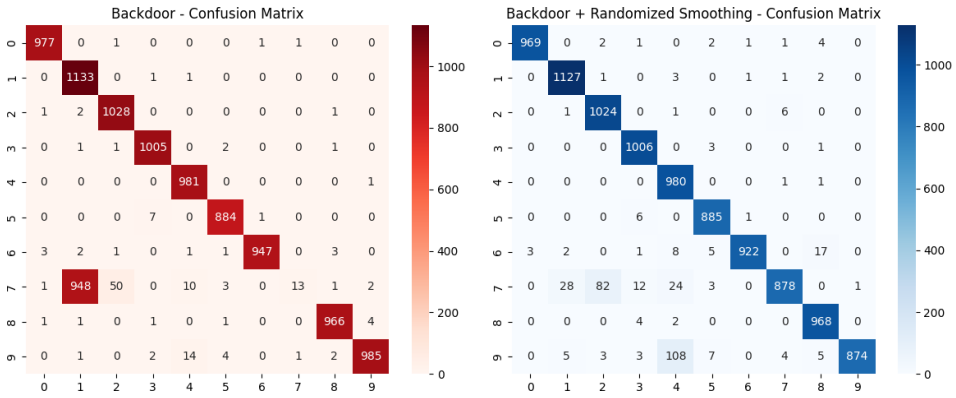


Figure 5.67: Confusion matrices before and after Randomized Smoothing defense on Backdoor attack.

Adversarial training demonstrates strong recovery against the manual backdoor attack by successfully correcting targeted misclassifications. As illustrated, several poisoned inputs originally misclassified (e.g., 7 1) are reverted back to their true class after defense application. The confusion matrix shows a significant reduction in mispredictions in class 7, although minor confusion remains. Compared to the attacked model, all evaluation metrics—including accuracy, precision, recall, and F1-score—recover to near-clean levels. This suggests that adversarial training is highly effective in mitigating the impact of backdoor poisoning, even when the trigger is subtle or manually placed.

5.5.3 Training

5.5.3.1 Adversarial Training

The ResNet18 model was adversarially trained on MNIST using mixed clean and FGSM-generated adversarial examples for 20 epochs. The training loss initially fluctuated but ultimately stabilized at low values by the final epochs, indicating robust convergence and improved resistance to adversarial perturbations (see Table **??**).

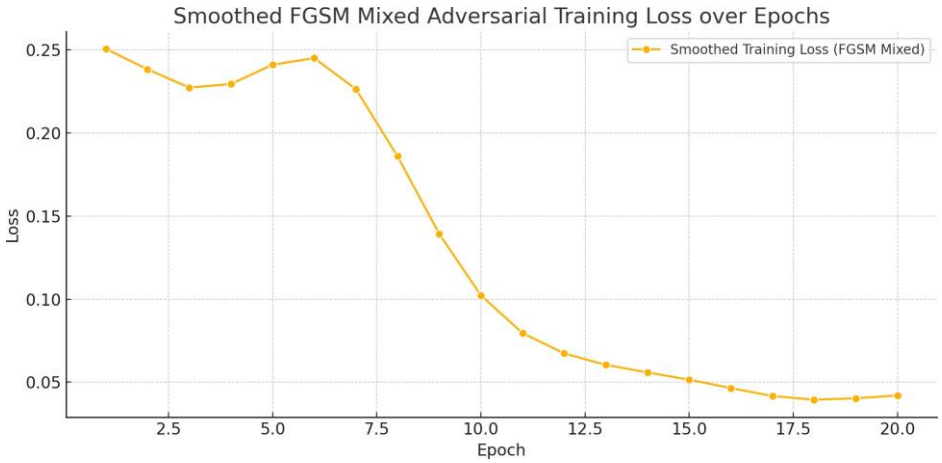


Figure 5.68: Smoothed training loss over epochs for FGSM Mixed Adversarial Training.

Metric	Clean	FGSM Attack	Adv. Training (Defense)
Accuracy	99.19%	49.59%	79.25%
Precision	99.19%	47.66%	80.45%
Recall	99.19%	49.57%	79.14%
F1-score	99.19%	47.17%	78.43%

Table 5.12: Comparison of clean accuracy, FGSM attack results, and FGSM adversarial training defense on MNIST.

Adversarial training with FGSM examples significantly improved model robustness, raising the accuracy under attack from 49.59% to 79.25%, and the F1-score from 47.17% to 78.43% (see Table 5.12). This demonstrates the effectiveness of adversarial training in mitigating the impact of FGSM attacks.

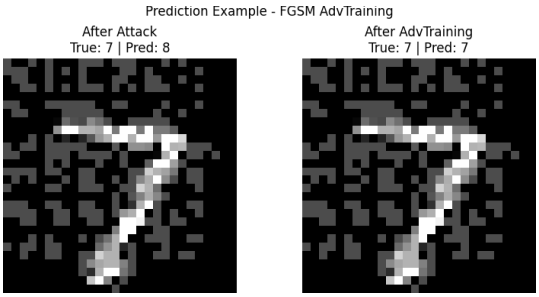


Figure 5.69: Prediction examples for FGSM attack: after attack (left) and after adversarial training (right).

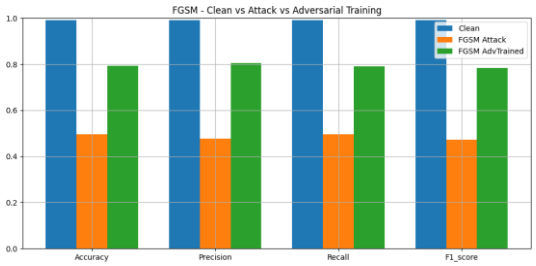


Figure 5.70: FGSM: Clean vs Attack vs Adversarial Training (bar chart of metrics).

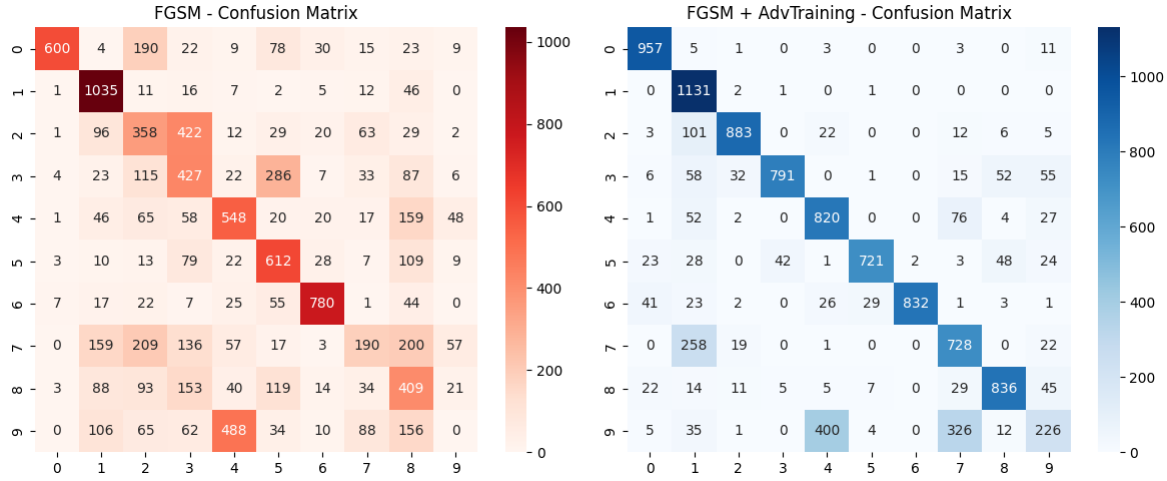


Figure 5.71: Confusion matrices for FGSM attack (left: no defense, right: after adversarial training).

Adversarial training provides strong recovery against FGSM attacks, significantly boosting accuracy and recall compared to the attacked model. The corrected prediction example confirms improved robustness, while the confusion matrix shows reduced misclassifications. However, some confusion remains, indicating that the defense, while effective, is not entirely foolproof.

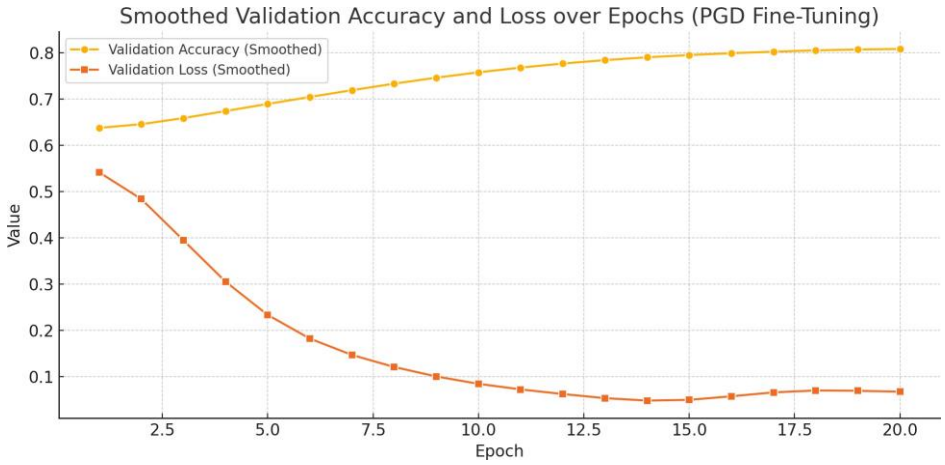


Figure 5.72: Smoothed validation accuracy and loss over epochs during PGD Adversarial Fine-Tuning.

Table 5.13: Evaluation metrics for PGD attack and PGD adversarial training on MNIST.

Metric	Clean	PGD Attack	PGD AdvTrain
Accuracy	99.19%	47.18%	95.66%
Precision	99.19%	45.05%	95.67%
Recall	99.19%	47.17%	95.64%
F1-score	99.19%	45.11%	95.64%

After adversarial training with PGD examples, the ResNet18 model's robustness improved substantially. While the attack reduced clean model accuracy to 47.18% , adversarial train- ing recovered accuracy to 95.66%. Precision, recall, and F1 score also increased sharply, indicating that the defense was highly effective against PGD perturbations (see Table 5.13). This highlights the significant impact of adversarial training in mitigating the effects of strong gradient-based attacks like PGD.

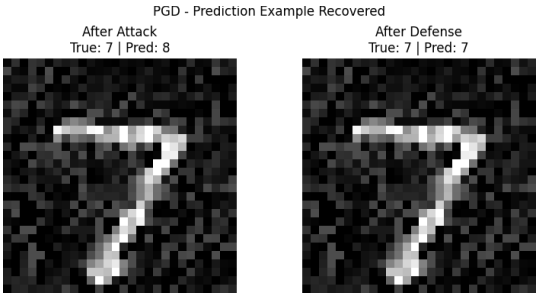


Figure 5.73: Prediction examples for PGD attack: after attack (left) and after adversarial training (right).

Figure 5.74: PGD: Clean vs Attack vs Adversarial Training (bar chart of metrics).

After adversarial training with PGD examples, the ResNet18 model's robustness improved substantially. While the attack reduced clean model accuracy to 47.18%, adversarial train- ing recovered accuracy to 95.66%. Precision, recall, and F1 score also increased sharply, indicating that the defense was highly effective against PGD perturbations (see Table 2.13). This highlights the significant impact of adversarial training in mitigating the effects of strong gradient-based attacks like PGD.

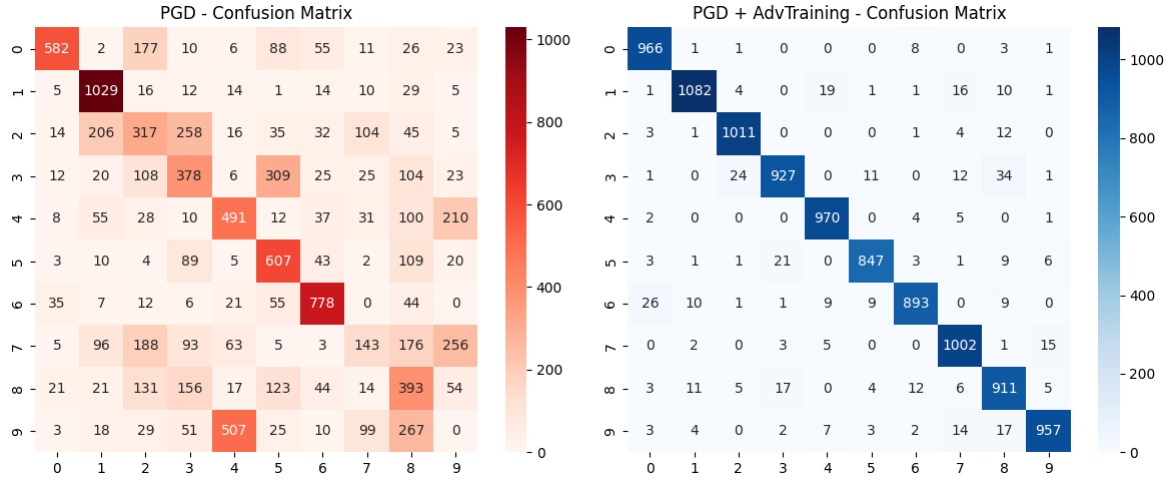


Figure 5.75: Confusion matrices for PGD attack (left: no defense, right: after adversarial training).

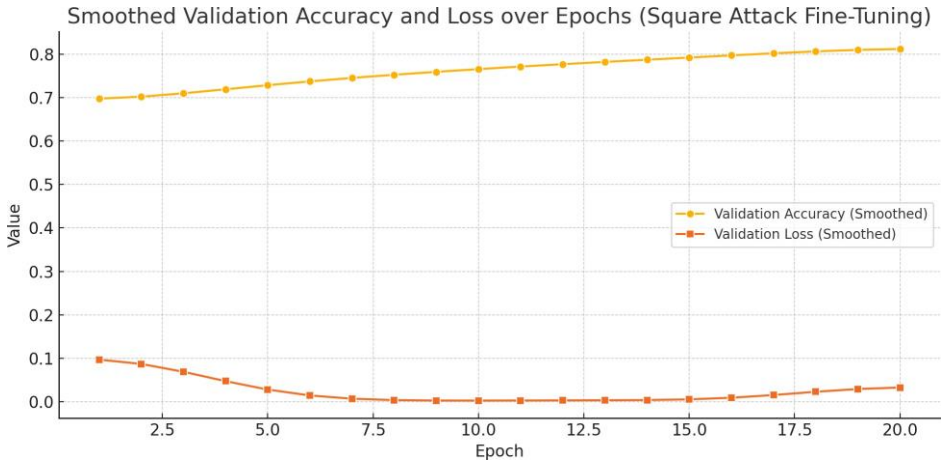


Figure 5.76: Smoothed validation accuracy and loss over epochs during fine-tuning with Clean + Square Attack data.

Metric	Clean	Square Attack	Adv. Training (Defense)
Accuracy	99.19%	57.08%	99.95%
Precision	99.19%	58.84%	99.95%
Recall	99.19%	56.87%	99.95%
F1-score	99.19%	56.11%	99.95%

Table 5.14: Comparative statistics for Clean data, Square Attack, and Square Attack Adversarial Training defense.

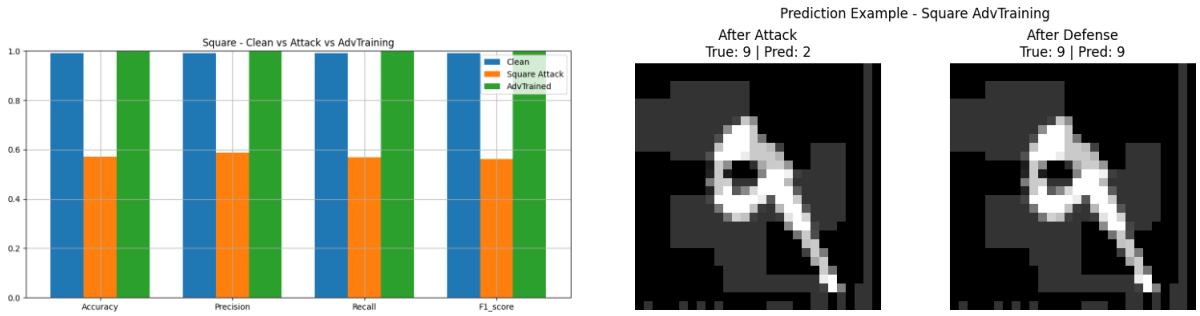


Figure 5.77: (Left) Metrics Comparison: Clean vs Square Attack vs Adversarial Training. (Right) Prediction Example: Square Attack (Left) vs. After Defense (Right).

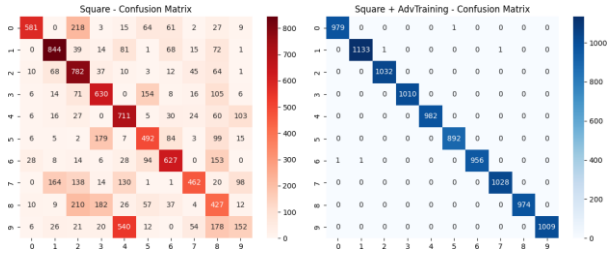


Figure 5.78: Confusion Matrices: (Left) Square Attack on original model, (Right) After Adversarial Training

Adversarial training delivers near-complete recovery against Square attacks, restoring all metrics close to clean performance. The corrected prediction confirms effective robustness, and the confusion matrix shows minimal misclassifications. This defense proves highly reliable against decision-based perturbations like the Square attack.

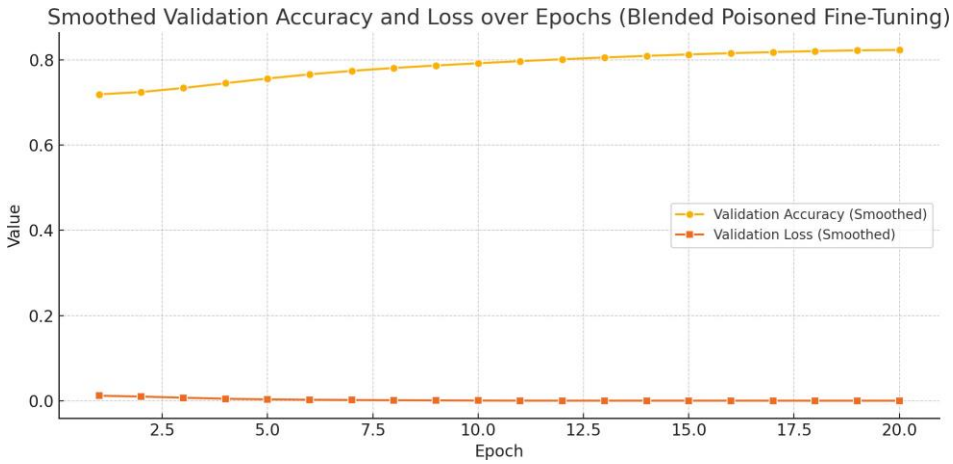


Figure 5.79: Smoothed validation accuracy and loss over epochs during fine-tuning with Clean + Enhanced Blended Poisoned data.

Table 5.15: Blended Clean Label Attack Evaluation (No Retraining): Metrics Comparison

Metric	Clean	Attack	Defense
Accuracy	99.19%	94.39%	94.17%
Precision	99.19%	89.41%	89.89%
Recall	99.19%	87.05%	87.27%
F1 Score	99.19%	88.08%	88.38%

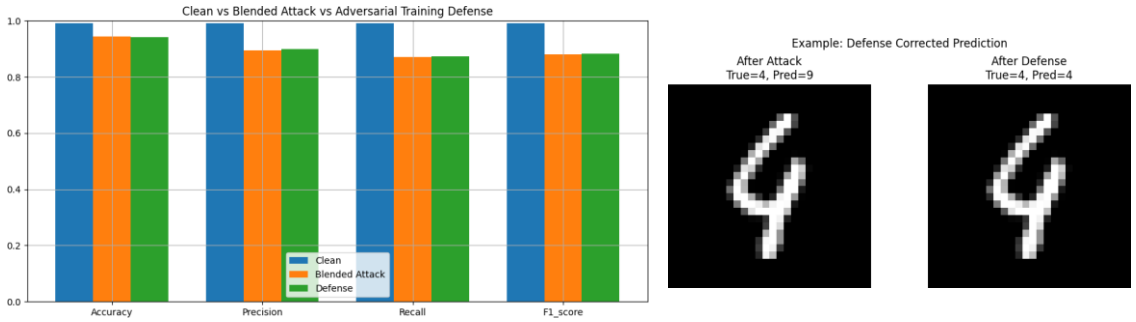


Figure 5.80: (Left) Comparative metrics for Clean, Blended Clean Label Attack, and Adversarial Training Defense; (Right) Prediction example: The model correctly recovers the true label after defense.

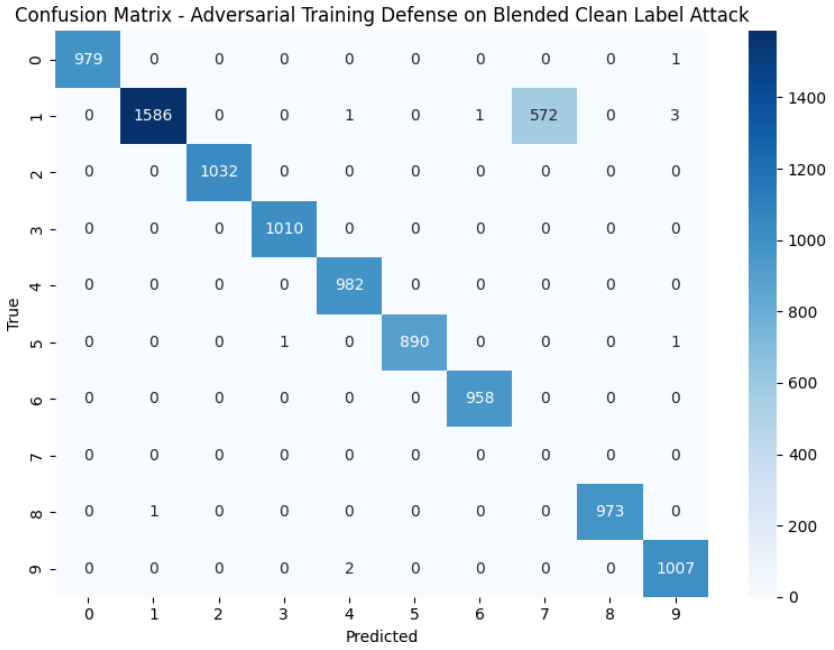


Figure 5.81: Confusion Matrix for Blended Clean Label Attack with Adversarial Training Defense.

Adversarial training shows solid defense against the blended clean label attack, restoring misclassified examples and maintaining strong performance across metrics. The corrected prediction and confusion matrix confirm improved robustness, though minor confusion remains in the targeted class, suggesting partial vulnerability persists.

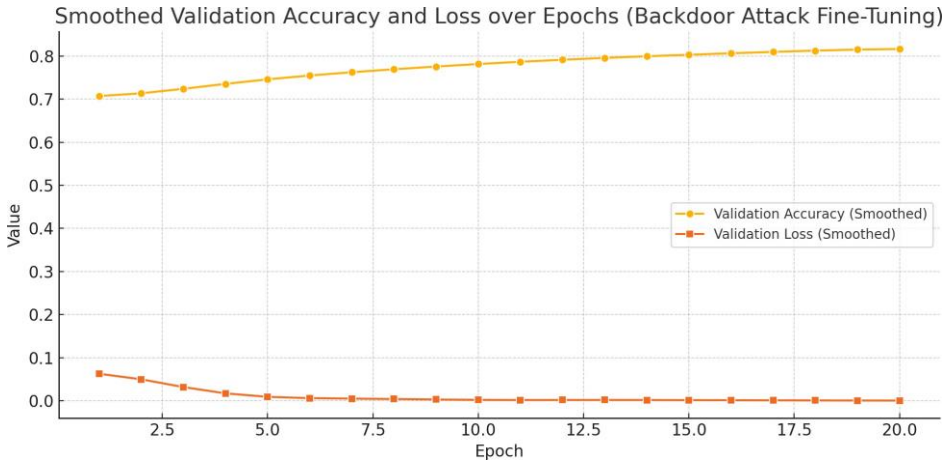


Figure 5.82: Smoothed validation accuracy and loss over epochs during fine-tuning with Clean + Backdoor Attack data.

Table 5.16: Backdoor Attack Evaluation (Random Trigger): Metrics Comparison

Metric	**Clean**	**Attack**	**Defense**
Accuracy	99.19%	89.22%	98.36%
Precision	99.19%	93.18%	98.53%
Recall	99.19%	89.48%	98.40%
F1 Score	99.19%	86.43%	98.37%

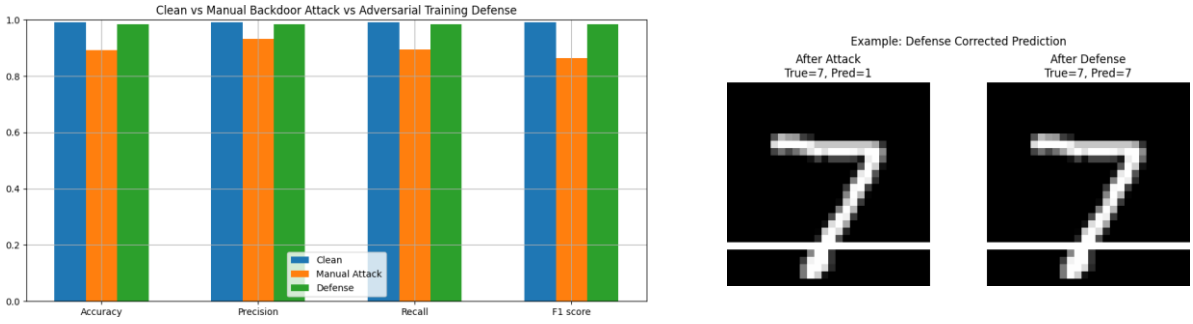


Figure 5.83: (Left) Comparative metrics for Clean, Backdoor Attack (Random Trigger), and Adversarial Training Defense. (Right) Prediction example: Defense successfully restores correct classification.

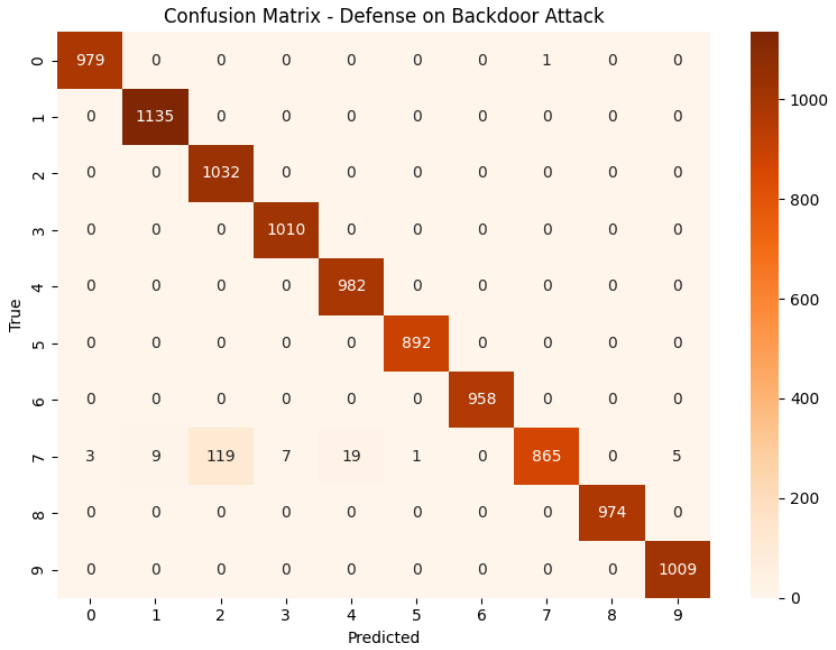


Figure 5.84: Confusion Matrix for Backdoor Attack with Adversarial Training Defense.

Adversarial training effectively mitigates the manual backdoor attack, restoring both accuracy and reliability. The prediction example confirms successful correction, and the confusion matrix reveals that most targeted misclassifications are reduced. Some residual confusion in the poisoned class persists, but overall robustness is notably improved

5.5.3.2 Label Smoothing

The ResNet18 model was trained on MNIST using label smoothing for 20 epochs. The training loss stabilized around 0.56 after the final epoch, indicating a smooth and stable convergence trend (see Figure **5.85** for the validation accuracy curve).

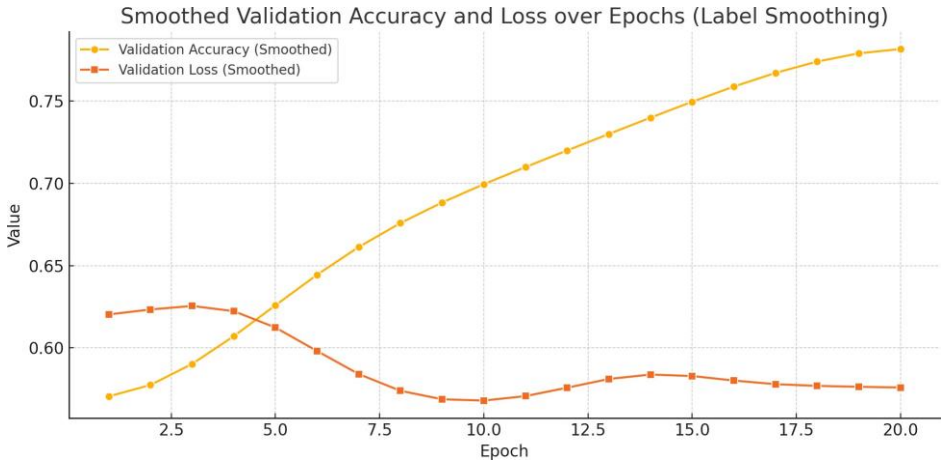
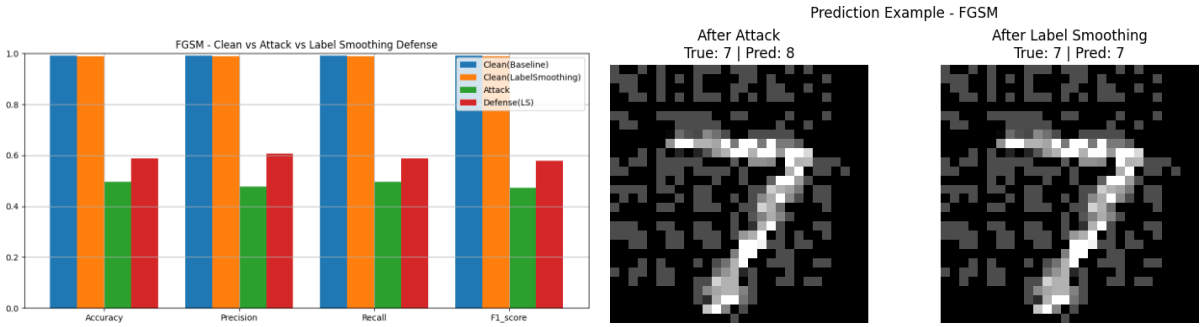


Figure 5.85: Validation Accuracy and Loss over Epochs during Label Smoothing training.



Bar chart: Clean vs Attack vs Label Smoothing

Prediction Example: after attack / after label smoothing

Figure 5.86: FGSM Attack: Metrics comparison and example prediction after Label Smoothing defense.

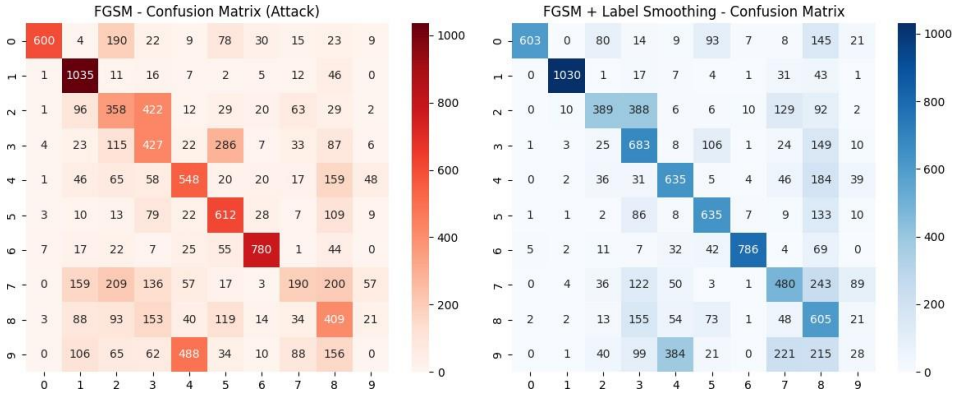
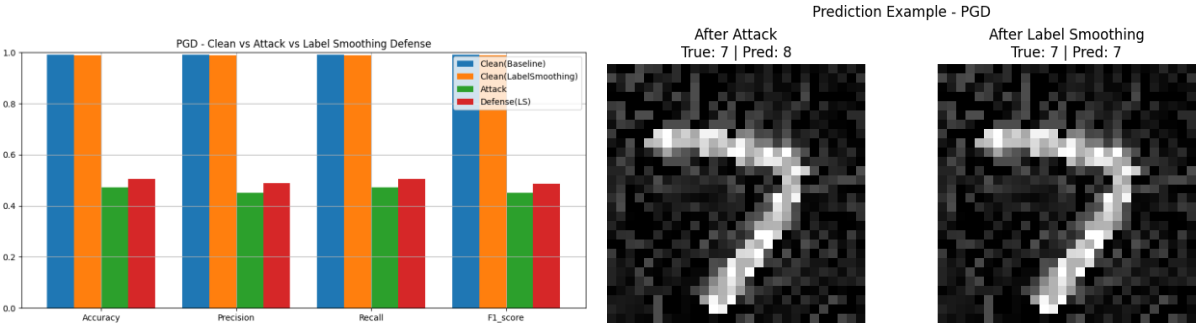


Figure 5.87: Confusion matrices before and after Label Smoothing defense on FGSM attack.

Label smoothing offers partial protection against FGSM attacks by reducing model over-confidence and correcting some adversarial misclassifications, as seen in the prediction exam-

ple. However, the performance recovery is modest while better than no defense, the confusion matrix and metrics show it falls short of fully restoring clean accuracy.



Bar chart: Clean vs Attack vs Label Smoothing

Prediction Example: after attack / after label smoothing

Figure 5.88: PGD Attack: Metrics comparison and example prediction after Label Smoothing defense.

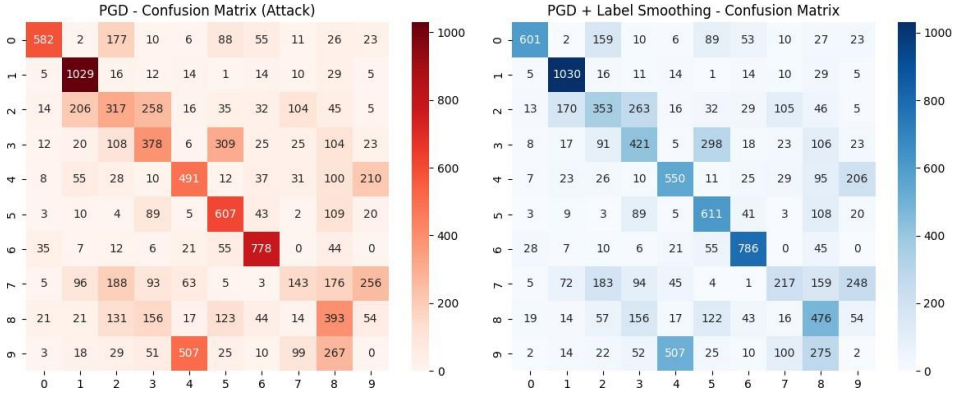
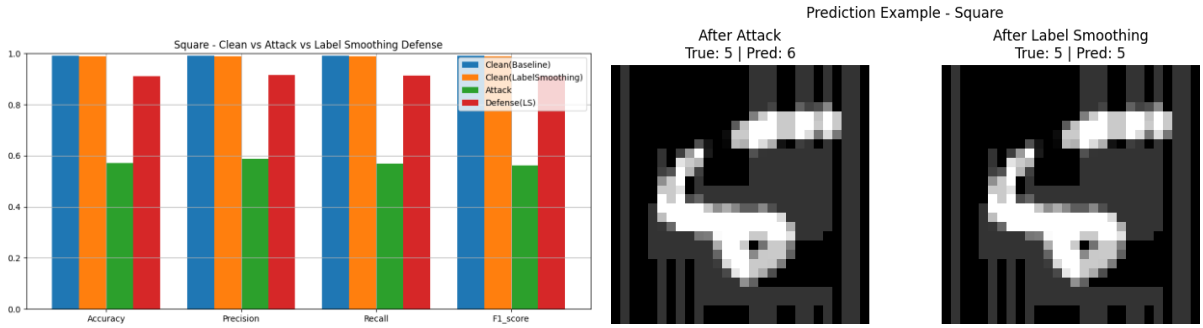


Figure 5.89: Confusion matrices before and after Label Smoothing defense on PGD attack.

Label smoothing shows limited improvement against PGD attacks. While it helps cor- rect some predictions—as seen in the example—and slightly boosts metrics compared to the attacked model, the confusion matrix reveals persistent misclassifications. This suggests that label smoothing alone is not sufficient for defending against stronger iterative attacks like PGD.



Bar chart: Clean vs Attack vs Label Smoothing

Prediction Example: after attack / after label smoothing

Figure 5.90: Square Attack: Metrics comparison and example prediction after Label Smoothing defense.

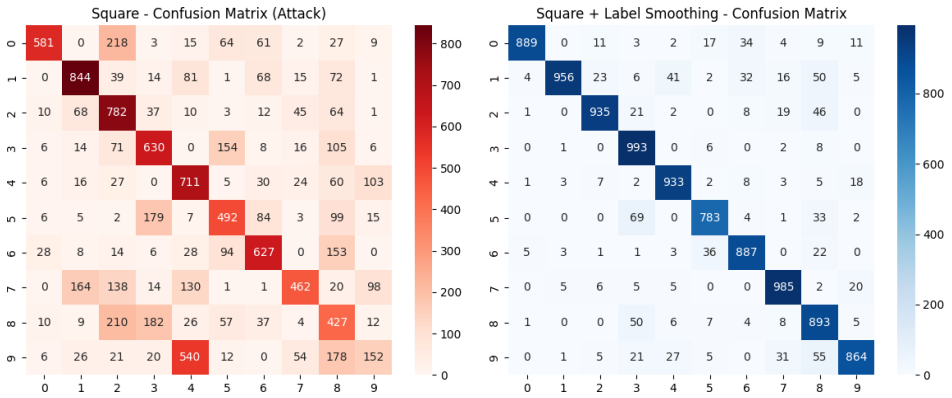
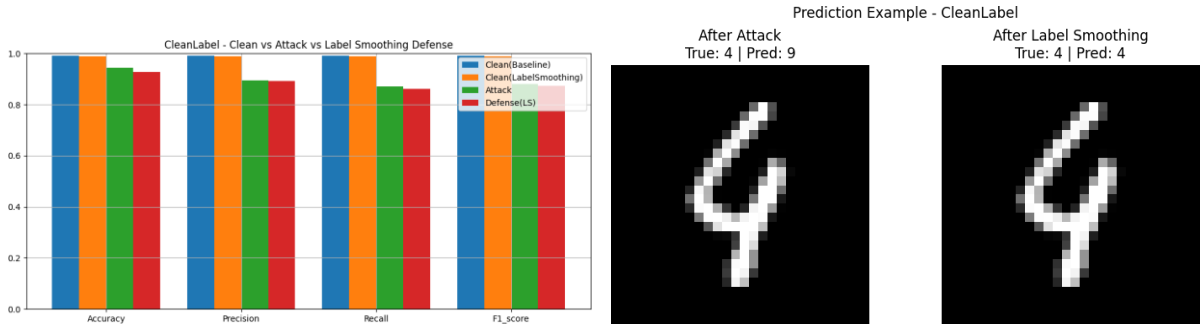


Figure 5.91: Confusion matrices before and after Label Smoothing defense on Square attack.

Label smoothing provides moderate robustness against the Square attack. It corrects some adversarial misclassifications, as seen in the prediction example, and improves overall metrics. However, confusion matrix analysis reveals persistent errors across several classes, indicating that while helpful, this defense is not fully reliable against decision-based attacks like Square.



Bar chart: Clean vs Attack vs Label Smoothing

Prediction Example: after attack / after label smoothing

Figure 5.92: CleanLabel Attack: Metrics comparison and example prediction after Label Smoothing defense.

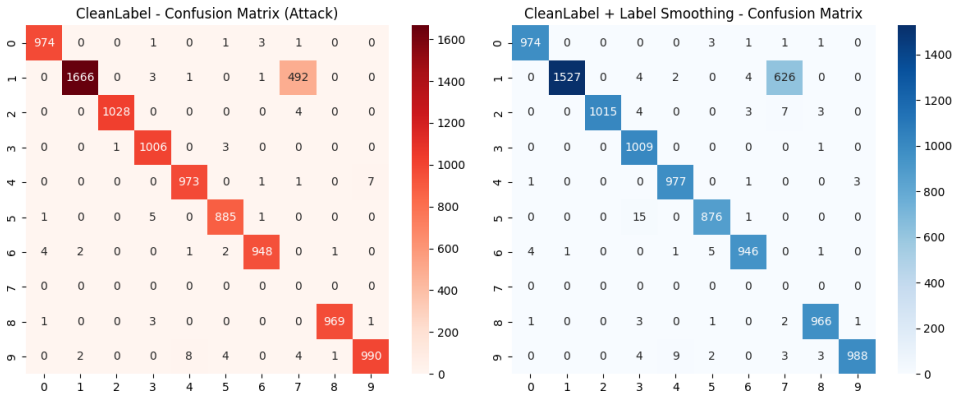
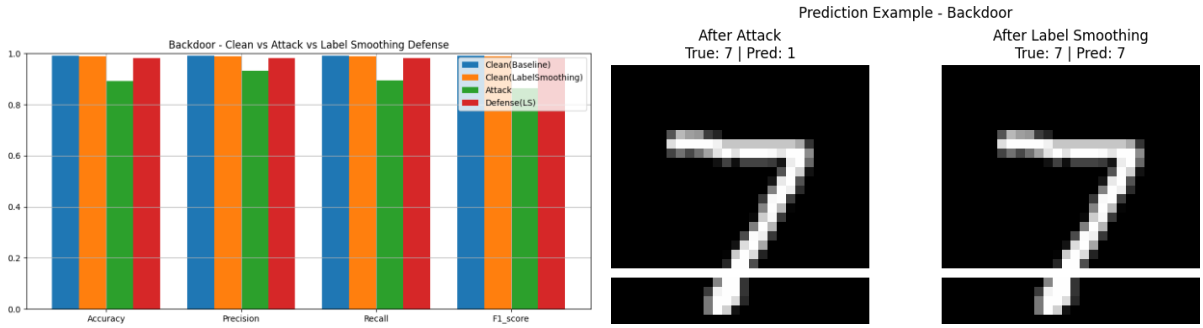


Figure 5.93: Confusion matrices before and after Label Smoothing defense on CleanLabel attack.

Label smoothing shows limited effectiveness against the Clean Label attack. While it helps correct some targeted misclassifications—as shown in the prediction example—and slightly improves metrics, the confusion matrix reveals that class confusion, especially between targeted classes, remains noticeable. This highlights the need for stronger or complementary defenses in targeted poisoning scenarios.



Bar chart: Clean vs Attack vs Label Smoothing

Prediction Example: after attack / after label smoothing

Figure 5.94: Backdoor Attack: Metrics comparison and example prediction after Label Smoothing defense.

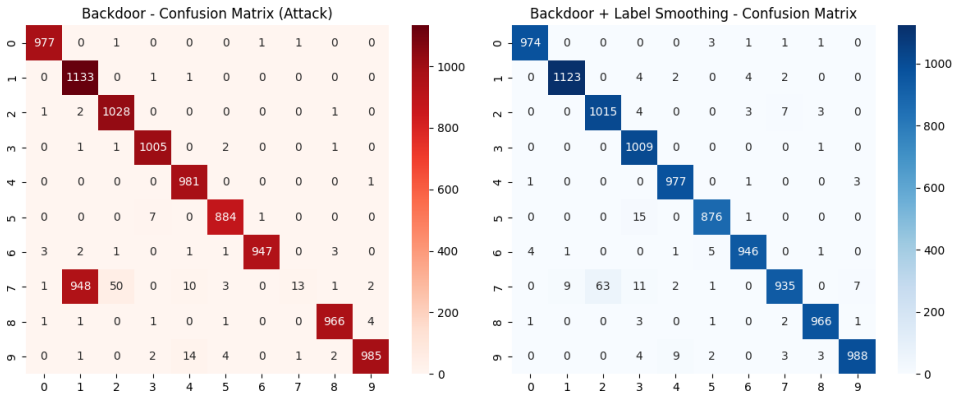


Figure 5.95: Confusion matrices before and after Label Smoothing defense on Backdoor attack.

Label smoothing shows partial recovery against backdoor attacks, correcting some poisoned predictions as shown in the example. While the overall metrics improve, the confusion matrix still reveals clear traces of the trigger's effect—especially on class 7—indicating that this defense alone is insufficient to fully neutralize backdoor manipulations.

Chapter 6

General Conclusion

In this work, we have studied the robustness of deep learning models, specifically the rest-net architecture, against adversarial attacks in the context of image classification. This work addresses a growing concern in the field of artificial intelligence, where maliciously crafted inputs can significantly degrade the performance and reliability of deep learning based systems deployed in real-world applications.

To achieve this, we proposed a comprehensive methodology combining adversarial at- tack implementation and defense evaluation. Our experimental framework was built upon the MNIST dataset and the ResNet18 convolutional neural network architecture. We generated adversarial examples using various evasion-based attacks, including FGSM, PGD, Clean Label, Backdoor (BadNet), and Square Attack. Each attack was carefully designed and analyzed to simulate realistic threat models and assess their impact on model predictions.

To counteract these attacks, we evaluated a diverse range of defense mechanisms. These included preprocessing techniques (Gaussian noise, bit-depth reduction, JPEG compression), training-time defenses (adversarial training, label smoothing), and postprocessing strategies (confidence thresholding, randomized smoothing). Each method was evaluated using the state of the art metrics: accuracy, precision, recall, and F1-score, along with visualizations including bar charts, confusion matrices, and prediction examples before and after defense.

Our experiments revealed that while certain attacks (e.g., backdoor or clean label) are highly deceptive, well-designed defense strategies such as adversarial training and smoothing tech-

niques can significantly mitigate their effects and restore model reliability.

For future work, we suggest exploring more advanced threat models such as black-box or transfer-based attacks, integrating certified defenses for provable robustness, and applying our evaluation framework to more complex datasets and architectures. Furthermore, expanding the training data with more diverse adversarial patterns could enhance the practical applicability of the defense mechanisms.

Ultimately, this work contributes to a deeper understanding of adversarial vulnerabilities in deep learning and highlights promising directions for building more secure and trustworthy AI systems.

Bibliography

[1] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Deep learning applications and challenges in big data analytics," Journal of Big Data,vol. 2, no. 1, pp. 1–21, 2015. [Online]. Available: https://www.researchgate.net/figure/An-illustration-of-deep-learning-with-two-hidden-layers_fig2_331988523

[2] Unknown, "Pytorch cnn mnist tutorial," https://imagetou.com/my-img-pytorch-cnn-mnist-tutorial.html, n.d., accessed: June 28, 2025.

[3] S. Paul, B. Mondal, A. Das, and P. K. Choudhury, "Intel-ligent recognition of the indian medicinal plant leaf using deepconvolutional neural network," Ecological Informatics, vol.69,p.101633,2022.[Online].Available: https://www.researchgate.net/figure/ResNet-18-CNN-transfer-learning-model-in-pictorial-form_fig8_363695411J.

[4] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha, "A survey on adversarial machine learning attacks," IEEE Access, vol. 8, pp.144 601–144 618, 2020. [Online]. Available: https://www.researchgate.net/figure/Addition-of-FGSM-attacks-The-first-row-shows-the-original-images-while-the-second-row_fig3_352182522

[5] Y. Zhang, Z. Zhang, X. Xie, B. Li, and Y. Wang, "Towards fast and lightweight adversarial training: An effective pgd-based approach," in Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI). IJCAI, 2021, pp. 3350–3356. [Online]. Available: https://www.researchgate.net/figure/Adversarial-example-generation-process-of-PGD-AT-12-FGSM-RS-30-and-our-FGSM-SDI-in_fig1_355224088

[6] H. Hasan. (2022) Data poisoning and backdoor attacks: An overview (part 1).Accessed: June 28, 2025. [Online]. Available: https://hammoudhasan.medium.com/data-poisoning-and-backdoor-attacks-an-overview-part-1-4507329e8f1a

[7] M. Alqahtani, A. Almomani, M. Tolba, H. Alwageed, A. Alshdadi, and A. Alshdadi, "An ensemble transfer-based adversarial attack framework," Sensors,vol. 23, no. 1, pp. 1–22,

2023. [Online]. Available: https://www.researchgate.net/figure/Adversarial-Transfer-Attack-Framework_fig3_375594560

[8] C. Yang, W. Wang, Z. Chen, Y. Li, Y. Xie, and Z. Yan, "Adversarial attack and defense through adversarial training and feature fusion for malware detection," Applied Sciences,vol.11no.21,p.10479, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/21/10479

[9] A. Shafahi et al., "Poison frogs! targeted clean-label poisoning attacks on neural networks," Advances in Neural Information Processing Systems (NeurIPS), 2018.

[10] B. Wu, S. Wei, M. Zhu, M. Zheng, Z. Zhu, M. Zhang, H. Chen, D. Yuan, L. Liu,and Q. Liu, "Defenses in adversarial machine learning: A survey," arXiv preprintarXiv:2312.08890, 2023.

[11] L. Deng, "The mnist database of handwritten digit images for machine learning research," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[12] Schmidhuber,"Deep learning in neural networksAnoverview,"*NeuralNetworks*,vol.61, pp. 85–117, 2015.M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[13] Techopedia, "Deep learning - definition, meaning and use cases," https://www.techopedia.com/definition/deep-learning, 2024, accessed: 21 May 2025.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[16] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[17] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[18] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-

level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[19] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta,T.Duan, D. Ding, A. Bagul, C. Lan

glotz, K. Shpanskaya, M. Lungren, and A. Ng, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[20] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, arXiv preprint arXiv:1804.02767.

[21] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, pp. 448–455, 2019.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[23] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, Y. Bengio, A. Fischer, A. Courville, and E. Bengio, "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 233–242.

[24] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[25] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[26] J. Gilmer, S. S. Schoenholz, G. Riley, O. Vinyals, and G. E. Dahl, "Adversarial spheres," in *International Conference on Learning Representations*, 2018.

[27] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations (ICLR)*, 2018.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[29] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, "Dataset shift in machine learning," in *The MIT Press*, 2008.

[30] B. Wang and N. Z. Gong, "Adversarial attacks and defenses in deep learning: From a

perspective of cybersecurity," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[32] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 33–44.

[33] F. Tramèr, M. Jagielski, N. Carlini, and D. Boneh, "Triangle attack: Query-efficient decision-based adversarial attack," in *European Conference on Computer Vision*. Springer, 2022, pp. 682–699.

[34] S. Li, Y. Liu, and H. Su, "Bo-dba: Query-efficient decision-based adversarial attacks via bayesian optimization," in *International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: https://openreview.net/forum?id=beiz51zcm-H

[35] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *International Conference on Learning Representations (ICLR)*, 2018.

[36] J. Rauber *et al.*, "Towards practical decision-based attacks against machine learning models," *arXiv preprint arXiv:2007.12062*, 2020.

[37] J. Chen, M. Jordan *et al.*, "Hopskipjumpattack: A query-efficient decision-based attack," *IEEE European Symposium on Security and Privacy*, 2020.

[38] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," *European Conference on Computer Vision (ECCV)*, 2020.

[39] A. Shafahi *et al.*, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[40] A. Paya *et al.*, "Apollon: A robust framework for adversarial defense in deep learning," *Journal of Adversarial Machine Learning*, 2024.

[41] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” in *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2017.

[42] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, “A study of the effect of jpeg compression on adversarial images,” in *arXiv preprint arXiv:1608.00853*, 2016.

[43] X. Liu, S. Li, X. Cao, and H. Jin, “Feature distillation: Dnn feature map compression in adversarial defense,” *arXiv preprint arXiv:1803.06372*, 2018.

[44] M. Lecuyer *et al.*, “Certified robustness to adversarial examples with differential privacy,” *IEEE Symposium on Security and Privacy (SP)*, 2019.

[45] C. Guo *et al.*, “Countering adversarial images using input transformations,” in *International Conference on Learning Representations (ICLR)*, 2018.

[46] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” *Network and Distributed System Security Symposium (NDSS)*, 2018.

[47] A. Khamaiseh *et al.*, “Adversarial example detection using a multilayer classifier in deep neural networks,” *Security and Privacy*, 2022.

[48] K. Ren *et al.*, “Adversarial attacks and defenses in deep learning: A survey,” *IEEE Access*, 2020.

[49] S. Jha *et al.*, “Adversarial training for certifiable robustness,” *Transactions on Machine Learning Research*, 2025.

[50] R. Muller, S. Kornblith, and G. Hinton, “When does label smoothing help?” *NeurIPS*, 2019.

[51] L. Goibert *et al.*, “Adversarial training through robust feature matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[52] C. Xie *et al.*, “Mitigating adversarial effects through randomization,” *International Conference on Learning Representations (ICLR)*, 2018.

[53] J. Zhou *et al.*, “Improving randomized smoothing with generative models,” *Journal of Machine Learning Research*, 2024.

[54] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *International Conference on Learning Representations (ICLR)*, 2017.

[55] D. Stutz, M. Hein, and B. Schiele, “Confidence-calibrated adversarial training,” *NeurIPS*, 2020.

[56] H. Zhang *et al.*, “Efficient neural network robustness certification with general activation functions,” *NeurIPS*, 2018.

[57] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, “Certified adversarial robustness via randomized smoothing,” in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 1310–1320.

[58] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, “Efficient neural network robustness certification with general activation functions,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[59] S. Gowal *et al.*, “Scalable verified training for provably robust image classification,” *International Conference on Computer Vision (ICCV)*, 2019.

[60] Y. Tsuzuku, I. Sato, and M. Sugiyama, “Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks,” *NeurIPS*, 2018.

[61] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.