Spring 6-2024

Enhancing Robustness of Machine Learning Models Against Adversarial Attacks

Ronak Guliani
Portland State University

Enhancing Robustness of Machine Learning Models against Adversarial Attacks

by

Ronak Guliani

An undergraduate honors thesis submitted in partial fulfillment of the

requirements for the degree of

Bachelor of Science

in

University Honors

and

Computer Science

Thesis Advisor

Nirupama Bulusu

Portland State University

2024

ABSTRACT

Machine learning models are integral for numerous applications, but they remain increasingly vulnerable to adversarial attacks. These attacks involve subtle manipulation of input data to deceive models, presenting a critical threat to their dependability and security. This thesis addresses the need for strengthening these models against such adversarial attacks. Prior research has primarily focused on identifying specific types of adversarial attacks on a limited range of ML algorithms. However, there is a gap in the evaluation of model resilience across algorithms and in the development of effective defense mechanisms. To bridge this gap, this work adopts a two-phase approach. First, it simulates attacks like the Basic Iterative Method (BIM), DeepFool, and Fast Gradient Sign Method (FGSM)on common ML models trained on MNIST and CIFAR-10, which are used for image processing. This thesis will then discuss defensive strategies, which reduces the sensitivity of the model to input changes to improve model resilience against attacks. The findings are aimed to benefit ML researchers to develop more secure and robust ML systems.

TABLE OF CONTENTS

1 INTRODUCTION

From speech and image recognition to predictive tools in healthcare, machine learning models (ML) are crucial in various applications . As these models become essential in both critical infrastructures and personal devices, ensuring their security and dependability are increasingly important. A major threat to their integrity are adversarial attacks—sophisticated techniques designed to deceive ML systems by manipulating input data.

Existing research on adversarial attacks reveals that even the most advanced ML models are susceptible to these manipulations, leading to incorrect outcomes and compromising reliability. Early ML research has focused on demonstrating the practicality of adversarial attacks across models. Studies like Szegedy et al. (2013) first highlighted the vulnerability of neural networks to adversarial examples, sparking interest in understanding and mitigating these effects.

Despite this progress, robust defensive strategies against advanced adversarial attacks are lacking. Most current defenses are tailored to specific types of attacks or specific models, and fail when defending against new or slightly altered attacks. Standard benchmarks like the MNIST dataset for handwriting and the CIFAR-10 dataset for object recognition test the resiliency of models, underscoring the need for defenses that are effective across different data domains and model architectures.

Attacks like the Fast Gradient Sign Method (FGSM) utilize the gradients of the model's loss function to create manipulations that are disruptive. More complex techniques, such as the Basic Iterative Method (BIM) and DeepFool, employ iterative processes to refine these manipulations, aiming to subtly and efficiently cross decision boundaries to trick the model.

Defensive strategies against adversarial attacks have evolved from basic data augmentation and modifying parameters to more sophisticated methods. While these methods have shown promise, they often involve trade-offs in terms of demand and applicability, showing the challenge of developing efficient defenses that can adapt to advanced adversarial tactics.

Defense mechanisms also come at the cost of increased computational resources. Implementing complex algorithms that check for adversarial attacks or use advanced encryption can significantly slow down operations. This trade-off between security and efficiency is challenging in environments where quick processing is crucial, such as in autonomous driving systems or real-time transaction processing. Developers must balance the need for robust defenses with the practical limitations of their applications, aiming to optimize both security measures and system performance without compromising one for the other.

This thesis evaluates the resiliency of ML models against adversarial attacks and tests the effectiveness of defensive strategies applied. By using a dual-phase approach that both simulates attacks and implements defenses, this study aims to measure the robustness of ML systems against adversarial threats. The first phase involves a systematic simulation of well-known adversarial attacks, such as FGSM, BIM, and DeepFool. These attacks target various model architectures including CNNs, DNNs, and RNNs. Each model is rigorously

tested using standard datasets like MNIST and CIFAR-10, which are commonly used benchmarks in the ML community

In the second phase, the thesis tests several defensive strategies applied to counteract the identified vulnerabilities. This includes techniques like adversarial training, where models are exposed to both clean and manipulated data during training to enhance their resilience. Defensive distillation is also tested, which aims to make the models less sensitive to input perturbations by training a secondary model with softened output probabilities. Additionally, newer methods such as feature squeezing, which simplifies inputs to reduce space for manipulation, gradient masking to obscure useful gradients from attackers, and input randomization to add unpredictability, are tested for their effectiveness across different scenarios.

These phases provide a comprehensive analysis of both the attack landscape and the defensive countermeasures. By measuring effectiveness of each strategy against a variety of attack types on multiple model architectures and datasets, this study aims to offer valuable insights into developing more robust ML systems.

2 LITERATURE REVIEW

2.1 Applications

This review explores ML applications across various sectors, highlighting the unique vulnerabilities to adversarial attacks within each. In Natural Language Processing (NLP), attacks compromise text processing tasks, affecting system reliability. In healthcare, adversarial manipulations can mislead diagnostic models, directly endangering patient safety. Autonomous systems, such as self-driving cars and surveillance technologies, face threats from manipulated inputs that could lead to critical errors. In finance, adversarial tactics target algorithms in high-frequency trading and fraud detection, potentially causing significant financial losses. Each sector shows the need for robust ML models that maintain performance while resisting sophisticated adversarial techniques, driving research into more effective defense mechanisms.

2.1.1 Natural Language Processing

In natural language processing (NLP), ML models are used for text generation, translations, and sentiment analysis. These tasks are exposed to adversarial attacks that involve subtle input manipulations. Such attacks can significantly degrade model performance and lead to incorrect outputs. The complexity and ambiguity of NLP make it difficult to differentiate between legitimate variations and adversarial attacks.

Ensuring resilience of NLP models is integral for various applications, including content moderation systems, chatbots, customer service, etc. In these contexts, reliability and accuracy are imperative. Attacks in NLP can result in misclassification of sentiment or the generation of inappropriate responses, undermining user trust and model effectiveness.

In sentiment analysis, an input manipulation can alter the perceived sentiment from positive to negative or vice versa. This can be harmful in social media monitoring or customer feedback analysis, where incorrect sentiment classification might lead to

unwarranted business decisions or damaged brand reputation (Alsmadi et al., 2021). Similarly, incorrect wording of a product review can shift the sentiment analysis from a positive review to a negative one, misleading potential customers and affecting business sales.

Adversarial attacks can also result in the generation of offensive or nonsensical responses, severely affecting user experience and trust. For example, a chatbot designed to assist customers might be manipulated into providing inappropriate or harmful advice. This could lead to reputational damage and legal ramifications (Ebrahimi et al. 2018).

2.1.2 Healthcare

Adversarial attacks in healthcare exploit ML models by creating subtle, targeted manipulations to medical images. These are undetectable by human eyes but can significantly alter ML model outputs. Such vulnerabilities can have severe consequences, as demonstrated by Finlayson et al. (2019), who found that adversarial examples could lead to incorrect diagnoses by distorting medical image interpretations.

Finlayson discusses a concerning scenario where adversarial manipulations cause diagnostic models to incorrectly interpret medical images, such as mammograms or MRIs. This can lead to false negatives in critical conditions, delaying necessary treatments, and worsening patient outcomes.

Moreover, these manipulations compromise the training process of ML models, as they learn from affected data, perpetuating inaccuracies within the model. The stakes are exceptionally high in healthcare, where the reliability of diagnostic tools directly impacts patient survival rates and health Therefore, ensuring the robustness of ML applications against adversarial attacks is not just a necessity but imperative to maintain trust in digital healthcare solutions and protect patient well-being.

In surgical settings, where robotic systems increasingly assist procedures, adversarial attacks could impair operational accuracy. D'Ettorre et al. (2021) showed that these attacks might cause deviations in robotic movements, potentially leading to surgical errors, stressing the importance of robust security measures in AI-driven medical devices.

2.1.3 Autonomous Systems

Autonomous systems, such as self-driving vehicles and automated surveillance, rely heavily on ML models to understand complex inputs and make real-time decisions. However, these models are susceptible to adversarial attacks where inputs are deliberately modified to induce errors and manipulate outcomes.

Further exploration into adversarial attacks has identified several attack vectors specific to autonomous systems. Kurakin et al. (2017) explains that physical adversarial examples, such as modified road signs, can mislead autonomous driving systems, presenting a severe threat to public safety. These findings underscore the implications of adversarial attacks and the need for comprehensive security measures within autonomous technologies. Eykholt et al. (2018) further examined the robustness of traffic sign recognition systems under varied environmental conditions, revealing that even slight alterations in physical objects could lead to misclassification, thereby jeopardizing the operational integrity of autonomous vehicles.

Adversarial attacks extend beyond mere input manipulation. Studies have shown that the temporal dynamics of autonomous systems, such as those involved in real-time decision-making processes, can also be exploited.

The methodologies to study these attacks often involve simulating scenarios where autonomous systems are subjected to a range of adversarial inputs to assess their resilience. For instance, Papernot et al. (2016) used a technique called the Jacobian-based Saliency Map Attack to identify the most vulnerable features in input data that, when perturbed, would most likely cause the model to err. This methodological approach aids in understanding the specific vulnerabilities of ML models deployed in autonomous systems and forms a basis for developing targeted defensive strategies.

2.1.4 Finance

Adversarial attacks in finance present unique challenges due to the sector's reliance on data-driven decision-making for operations ranging from algorithmic trading to fraud detection. These attacks target financial algorithms with the intent to deceive systems through manipulated data inputs, thereby causing financial losses or erroneous automated decisions. This vulnerability is critical as financial institutions increasingly depend on ML models for high-frequency trading and predictive analytics.

Algorithmic trading systems, a cornerstone of modern financial strategies, also face threats from adversarial attacks. Goldblum et al. (2020) explored how adversarial examples could be used to trigger inappropriate trades or manipulate market prices. Their findings reveal that even algorithms designed for high-speed trading are not immune to the disturbances caused by crafted input data designed to exploit model vulnerabilities. This could result in substantial financial discrepancies, highlighting a critical area for further investigation and mitigation within finance.

Furthermore, fraud detection systems, which employ complex algorithms to identify unusual patterns indicative of fraudulent activities, are particularly attractive targets for adversarial attacks. Carminati et al. (2020) analyzed how adversarial techniques could be employed to mask or alter user patterns in a way that evades detection. By manipulating the data that informs these models, attackers could conduct fraudulent activities without triggering alerts, posing severe risks to the financial stability of institutions.

2.2 Emergence of Adversarial Vulnerabilities

The recognition of adversarial vulnerabilities within ML models fundamentally changed the landscape of neural network research. This shift began with the work of Szegedy et al. in 2013, which exposed how DNNs could be misled by adversarial examples. These examples are created by minute, imperceptible manipulations to inputs, resulting in drastically altered outputs. Szegedy et al.'s study not only demonstrated this issue but also introduced the L-BFGS method to compose adversarial examples, emphasizing the type of perturbations to deceive ML models.

This research highlighted an important contradiction: neural networks show high accuracy on clean data but fail dramatically when given slightly altered data used to take advantage of a model's weaknesses. Although Szegedy et al. designed a framework for composing adversarial examples, their work primarily raised awareness and questions,

leaving significant gaps in defensive techniques and an understanding of these vulnerabilities in ML models.

The implications of Szegedy et al.'s work extend beyond academic research into the practical applications of ML. Industries that rely heavily on AI, such as autonomous driving, healthcare, and financial services, have had to reconsider the security aspects of deploying ML models. The realization that these models can be manipulated subtly and with potentially disastrous consequences has led to increased regulatory interest and the establishment of standards for ML security.

2.3 Advancements in Generating Adversarial Examples

Following the work of Szegedy et al., Ian Goodfellow et al. introduced the Fast Gradient Sign Method (FGSM) in 2014, simplifying the creation of adversarial examples. FGSM uses the gradients of the loss with respect to the input data to create disruptions that increase the prediction error in outputs. This method highlighted the inherent linearities in high-dimensional spaces that neural networks don't account for, providing a useful tool for researchers to test model robustness.

Goodfellow et al. argued that the linear behavior in high-dimensional spaces of neural networks contributes to their vulnerability, an insight that has been instrumental in understanding how adversarial examples trick models. While this work moved the field forward by providing a method to generate adversarial examples, it also underscored the need for more robust defenses, which the paper briefly covered through adversarial training. This method of training, where models are exposed to both clean and adversarial data, has become critical in developing resilient ML systems.

2.4 Developing Defensive Mechanisms

In 2015, Papernot et al. wrote "Distillation as a Defense" to enhance the robustness of neural networks. By using temperature scaling in the softmax layer of DNNs, the technique aimed to reduce the model's sensitivity to small input perturbations. This approach was significant as it represented a shift from identifying to mitigating vulnerabilities through modifications in model training.

While distillation was effective against several types of adversarial attacks, further research revealed its limitations against more sophisticated or specifically designed adversarial inputs. This gap highlights the race in adversarial machine learning, where defensive techniques must evolve to address emerging attack strategies. Papernot's work contributed to the understanding of how alterations in training processes can fortify models against adversarial manipulations and has influenced ongoing research efforts to develop more adaptive and comprehensive defenses.

One notable study in the realm of adversarial ML defenses is conducted by Tramèr et al., who explored the phenomenon of gradient masking. Gradient masking attempts to obscure the gradients that adversarial attack methods like FGSM rely on, making it harder for attackers to generate effective adversarial examples. Their 2017 paper analyzed the effectiveness of gradient masking and concluded that while it can deter simple gradient-based attacks, it often fails against more sophisticated multi-step attacks or attacks that approximate gradients differently.

Tramèr et al. demonstrated that defenses relying solely on gradient masking could be circumvented by attackers who adjust their strategies, such as using alternative methods to estimate gradients. This exposed gaps in the usage of gradient masking as a robust defense mechanism, suggesting that defenses need to be multi-faceted and adaptable to various attack methodologies. The contribution of this research is pivotal as it challenges the security community to develop deeper, more secure models.

In their 2018 paper, Samangouei et al. introduced Defense-GAN, a framework that leverages the generative capabilities of GANs to reconstruct inputs before they are processed by a classifier. The idea is that the GAN can map the disrupted inputs back onto the clean data, thus removing the adversarial manipulation.

While Defense-GAN showed promise in mitigating certain types of adversarial attacks, its success varied across different datasets and attacks. The model's dependency on the quality of the generator and the need for extensive training of clean data to accurately model the data distribution represent significant challenges. This approach emphasizes the potential of using generative models as part of a defense but also highlights the need for evaluations against a broad spectrum of attacks.

A 2017 study by Madry et al., proposed a framework for adversarial training that continuously updates the training dataset with newly generated adversarial examples. This method, based on the idea of robust optimization, aims to make the model resistant to adversarial attacks by exposing it to a wide range of attacks during training.

Madry et al.'s approach has been one of the most effective methods for improving adversarial robustness. The defense addresses the adaptiveness issue in neural networks, ensuring that the model learns from the attacks it is most exposed to. However, the computational cost of generating adversarial examples and retraining the network is significant, demonstrating a gap in the scalability of this defense for large-scale applications. The research by Madry et al. is pivotal as it sets a benchmark for what robust adversarial training should achieve.

In 2017, Xu et al. introduced Feature Squeezing, a simple yet effective method to detect and mitigate adversarial examples by reducing the color bit depth of images and spatial smoothing. This method works on the inference that adversarial perturbations are small and can be 'squeezed' out by reducing the complexity of the input features.

Feature Squeezing was effective against a range of adversarial attacks and provided a low-cost and computationally efficient method of defense. However, its effectiveness is limited against more complex or less noticeable attacks, and like many other defenses, it struggles with adaptability across attack methodologies. The contribution of Xu et al. is crucial in demonstrating that even simple transformations can provide a level of defense, paving the way for further exploration of efficient defensive strategies.

2.5 Synthesis of Adversarial Research

These studies underline the ongoing research in adversarial machine learning. They showcase the evolving nature of both adversarial threats and the corresponding defensive strategies required to secure AI systems across diverse applications. Each study builds upon previous studies, driving deeper understanding and development of effective defenses against adversarial attacks in ML.

The ongoing exploration of adversarial defenses in ML showcases a field in active development, grappling with the challenges of effectiveness and efficiency. The gaps identified necessitate continued innovation, especially in developing defenses that are not only robust across different attacks but also scalable and practical for real-world applications. This research forms the backbone of this thesis, providing foundational insights and a clear direction for future work.

3 METHODOLOGY

3.1 Conceptual Framework and Experimental Design

The methodology uses a dual-phase approach. In the first phase, the resilience of standard ML models to adversarial attacks is assessed. This involves using a set of ML models that are widely used in the field, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep neural networks (DNNs). These models are chosen due to their use in real-world applications. Each model is trained on standard benchmark datasets, including MNIST and CIFAR-10, providing a varied set of challenges in inputs.

Figure 1. MNIST, Fashion-MNIST, and CIFAR-10 Training Samples *adapted from "Exploring the Knowledge Embedded in Class Visualizations and Their Application in Dataset and Extreme Model Compression" by Abreu-Pederzini et al. 2021.*

The MNIST dataset contains 70,000 images of numeric digits, divided into a training set of 60,000 images and a test set of 10,000 images. The images are a grayscale representation of a single digit. MNIST is used for classification algorithms in identifying handwritten characters, which simulates straightforward visual data processing tasks.

The CIFAR-10 dataset includes 60,000 colored images classified into 10 classes, each with a different type of object including trucks, airplanes, birds, and ships. The images are

also separated into a training set of 50,000 images and a test set of 10,000 images. CIFAR-10 challenges ML models with a much higher level of complexity in image processing due to the color depth and object forms.

Adversarial attacks are simulated using established techniques known to exploit specific vulnerabilities in these models. The types of attacks include the Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), and DeepFool, which represent a range of attack complexities from simple gradient-based approaches to more intricate methods that aim to find the minimal perturbations necessary for output misclassification. These attacks are chosen because they illustrate different aspects of model vulnerabilities and test limits of model defenses under varied conditions.

3.2 Adversarial Attack Simulations

FGSM is a straightforward yet powerful attack method. It operates by leveraging the gradients of the neural network to create disruptions. This method alters the input data by adding noise that is sign-aligned with the gradients, essentially pushing the input across the decision boundary. The simplicity of FGSM makes it ideal for testing the initial robustness of models against adversarial examples. FGSM is particularly important to test as it provides a baseline indication of how vulnerable the models are to being tricked by rapid, one-step manipulations to their inputs.

Basic Iterative Method, an extension of FGSM, essentially applies the gradient sign attack multiple times with small step sizes, allowing for a finer search for effective adversarial examples. This iterative approach increases the power of the attack by gradually moving the input towards the adversarial goal over several steps, rather than making a single large step. BIM tests the resilience of models against more sustained and precise adversarial efforts, offering insights into how well the defenses can adapt to incremental but cumulatively significant changes to input data. This method is important for the thesis as it simulates an attacker who applies careful pressure on the model's vulnerabilities.

DeepFool is an attack method developed by Moosavi-Dezfooli et al., which iteratively manipulates the input image to cross the nearest class boundary. This boundary separates the targeted class and the neighboring class that the model is being manipulated to misclassify the image as. Unlike FGSM and BIM, DeepFool doesn’t rely on a fixed perturbation magnitude but instead aims to find the minimal perturbation that is adequate to deceive the model. DeepFool is crucial for this thesis as it assesses the robustness of models in an optimal adversarial scenario, therefore pushing the limits of what the defensive strategies can handle.

Testing these attacks provides a comprehensive overview of how adversarial strategies can expose the weaknesses of ML models. By evaluating the performance of these models against varied attack methods, the thesis aims to draw conclusions about the effectiveness of the employed defensive strategies under diverse adversarial conditions.

3.3 Defensive Strategies Evaluation

The second phase focuses on evaluating defensive strategies designed to mitigate these attacks. Five defense mechanisms are implemented and tested:

1. Adversarial training: Retraining models on a mixture of adversarial and clean examples to enhance their resilience to similar attacks.
2. Defensive distillation: Training a secondary model to generalize the classifications of a primary trained model, but with a softened output, which theoretically reduces the model's sensitivity to small manipulations.
3. Feature squeezing: Reducing precision of the input data, limiting the manipulations an attacker can introduce.
4. Gradient masking: Hiding gradient information of the model, making it difficult to compute the gradients required for generating adversarial examples.
5. Randomization: Randomly transforming the input data to prevent an attacker from generating targeted attacks.

3.4 Implementation and Training Procedures

The experimental setup involves training each model on clean data first, then testing it against each attack to establish a baseline performance. Subsequently, each model is equipped with the defensive strategies and re-tested under the same adversarial conditions. This approach allows for a comparative analysis of the impact of each defense strategy on the model's robustness to adversarial inputs.

To establish a baseline performance for each model against adversarial attacks, a structured approach was followed. Initially, each ML model was trained on clean data from the MNIST and CIFAR-10 datasets. This phase was crucial to ensure that the models had a high level of accuracy on unaltered data, setting a benchmark for their expected performance.

Each model was then subjected to the FGSM, BIM, and DeepFool attacks . These attacks were implemented using Python, utilizing popular ML and deep learning libraries including TensorFlow and Keras. The attacks were integrated using the Adversarial Robustness Toolbox (ART) library.

Below is snippet of my Python code used to train a model and test it against the FGSM, BIM, and DeepFool attacks attack using TensorFlow and ART:

```
# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((train_images.shape[0], 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((test_images.shape[0], 28, 28, 1)).astype('float32') / 255

# Set up data augmentation
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2
)
train_generator = datagen.flow(train_images, train_labels, batch_size=64)
```

```
# Define the model architecture with dropout and L2 regularization
def create_model():
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1), kernel_regularizer=l2(0.01)),
        Dropout(0.25),
        Conv2D(64, (3, 3), activation='relu', kernel_regularizer=l2(0.01)),
        Dropout(0.25),
        Flatten(),
        Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
        Dropout(0.5),
        Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Create an ensemble of models
models = [create_model() for _ in range(5)]
for model in models:
    model.fit(train_generator, epochs=10, validation_data=(test_images, test_labels))

# Function to evaluate ensemble on clean and adversarial data
def ensemble_evaluate(models, images, labels):
    predictions = [model.predict(images) for model in models]
    averaged_predictions = np.mean(predictions, axis=0)
    pred_labels = np.argmax(averaged_predictions, axis=1)
    accuracy = np.mean(pred_labels == labels)
    conf_matrix = confusion_matrix(labels, pred_labels)
    auroc = roc_auc_score(tf.keras.utils.to_categorical(labels), averaged_predictions, multi_class='ovr')
    precision, recall,  f1_score, __ = precision_recall_fscore_support(labels, pred_labels, average='macro')
    pr_auc = auc(recall, precision)
    return accuracy, conf_matrix, precision, recall, f1_score

# Evaluate the ensemble on clean data
clean_results = ensemble_evaluate(models, test_images, test_labels)
print(f'Clean Test Accuracy: {clean_results[0]:.2%}, AUROC: {clean_results[1]:.2%}, PR-AUC:
{clean_results[2]:.2%}')

# Set up adversarial attack and evaluate
classifier = TensorFlowV2Classifier(
    model=models[0],
    nb_classes=10,
    input_shape=(28, 28, 1),
    loss_object=tf.keras.losses.SparseCategoricalCrossentropy(),
    clip_values=(0, 1)
)
```

The initial block of the code handles the data loading and preprocessing of the MNIST dataset, which contains grayscale images of handwritten digits. The dataset is split into training and testing sets. Each image is then reshaped to fit the input requirements of the CNN (28x28 pixels, with a single color channel) and normalized to have pixel values between 0 and 1.

The data augmentation block uses the ImageDataGenerator from Keras to artificially enhance the size and diversity of the training dataset by applying random transformations such as rotation, width shift, height shift, and zoom. These transformations simulate variations that the model might encounter in real-world applications, thereby helping the model to generalize better from the training data and not just memorize it. This step is imperative in adversarial training as it introduces a form of robustness by exposing the model to diverse data scenarios, reducing the model's sensitivity to slight alterations used in adversarial attacks.

The create_model function defines a CNN architecture with multiple convolutional layers, dropout layers, and L2 regularization. Convolutional layers extract features, dropout layers randomly deactivate neurons during training to prevent overfitting, and L2 regularization penalizes large weights to encourage simpler, less overfitted models. The ensemble approach involves training multiple instances of this CNN model, which enhances model robustness by aggregating predictions from several models. This diversity in models tends to result in a more reliable overall model performance against varied inputs, including adversarial examples.

The ensemble_evaluate function calculates key performance metrics. These metrics are chosen for their ability to provide a comprehensive view of the model performance across different classification thresholds and imbalance levels, which is crucial for adversarial settings. Following the clean data evaluation, adversarial attacks (Basic Iterative Method in this example) are implemented using the ART library, which generates adversarial examples designed to mislead the model. The model is then tested against these adversarial examples to evaluate the effectiveness of the ensemble and the robustness measures (data augmentation, dropout, L2 regularization) in protecting against these sophisticated attacks.

In the full training code, advanced data preprocessing techniques, including feature scaling and more complex data augmentation methods, were employed to better condition the data for training under adversarial settings. Moreover, hyperparameter tuning was conducted using grid search and random search techniques to optimize model performances.

In the second phase of our methodology, we evaluate several defensive strategies aimed at enhancing the robustness of ML models against adversarial attacks. The first strategy, Adversarial Training, incorporates adversarial examples generated through methods such as FGSM, BIM, and DeepFool into the training process. This allows the model to recognize and resist similar disruptions in future tests. The effectiveness of this strategy is subsequently tested by evaluating the retrained model on a mix of clean and adversarial data to observe improvements over time.

For implementation of Adversarial Training, we modified the training process of CNNs, RNNs, and fully connected DNNs to include adversarial examples generated on-the-fly during training epochs. Below is a Python code snippet implementing adversarial

training using Tensorflow and ART. This snippet focuses on retraining and testing a CNN with adversarial examples generated via FGSM during the training process.

```
# Load MNIST data

# Define a simple CNN model
def create_model():
    model = Sequential([
        Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
        Conv2D(64, (3, 3), activation='relu'),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(10, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

model = create_model()

# Wrap the model with ART TensorFlowV2Classifier
classifier = TensorFlowV2Classifier(
    model=model,
    nb_classes=10,
    input_shape=(28, 28, 1),
    loss_object=tf.keras.losses.SparseCategoricalCrossentropy(),
    clip_values=(0, 1)
)

# Create the FGSM attack
attack = FastGradientMethod(estimator=classifier, eps=0.1)

# Integrate adversarial training using ART AdversarialTrainer
trainer = AdversarialTrainer(classifier, attacks=attack, ratio=0.5)  # 50% of training data will be adversarial
trainer.fit(train_images, train_labels, batch_size=64, nb_epochs=10)

# Evaluate the model on clean test data
clean_accuracy = classifier._model.evaluate(test_images, test_labels, verbose=0)[1]

# Evaluate the model on adversarial test data
adv_test_images = attack.generate(x=test_images)
adv_accuracy = classifier._model.evaluate(adv_test_images, test_labels, verbose=0)[1]

print(f'Clean Test Accuracy: {clean_accuracy:.2%}')
print(f'Adversarial Test Accuracy: {adv_accuracy:.2%}')
```

In this snippet, the FGSM attack is instantiated with a specified epsilon (perturbation magnitude), which determines the strength of the attack. The

AdversarialTrainer from ART is used to perform the adversarial training. The ratio parameter specifies that 50% of each batch during training will consist of adversarial examples. This mixed training helps the model learn to generalize from both clean and perturbed data. After training, the model's performance is evaluated to measure how well the adversarial training has improved the model's resilience to FGSM attacks. In the full testing code, FGSM, BIM, and DeepFool adversarial examples were created and tested on CNN, DNN, and RNN models.

Another key strategy is Defensive Distillation, which involves training a secondary model (the student) to replicate the behavior of a primary model (the teacher), but with softened output probabilities achieved through temperature scaling in the softmax layer. The distillation process's success is measured by comparing the robustness of the student model against adversarial attacks relative to the teacher model.

Below is a Python code snippet implementing defensive distillation using TensorFlow and Keras. This code trains a primary model (teacher) and then uses its softened output probabilities to train a secondary model (student).

```
# Define the teacher model
teacher_model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10)
])

# Compile and train the teacher model
teacher_model.compile(optimizer='adam', loss=CategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
teacher_model.fit(train_images, train_labels, epochs=5, validation_split=0.1)

# Create the student model with the same architecture but add a softmax layer with temperature
temperature = 10  # Define the temperature for softmax
student_model = Sequential(teacher_model.layers[:-1] + [Dense(10), Softmax(temperature)])

# Transfer the softened labels from the teacher to the student
teacher_probs = teacher_model.predict(train_images) / temperature
teacher_probs = tf.nn.softmax(teacher_probs)  # Soften the outputs

# Compile and train the student model
student_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
student_model.fit(train_images, teacher_probs, epochs=5, validation_split=0.1)

# Evaluate the student model on clean data
student_accuracy = student_model.evaluate(test_images, test_labels, verbose=0)[1]
print(f'Student Model Accuracy on Clean Data: {student_accuracy:.2%}')
```

The code involves two main stages: training a teacher model and then using its output to train a student model with softened probabilities. Initially, a CNN, designated as the teacher, is constructed and trained on the MNIST dataset using traditional labels. This model is trained with logits output (without a softmax activation layer) to assist the application of temperature scaling in subsequent steps. Once the teacher model is trained, its outputs are collected and softened by dividing them by a high temperature value (set to 10 in this example). These softened outputs serve as pseudo-labels for training the student model, which shares the same underlying architecture as the teacher but includes a final softmax layer adjusted by the same temperature. The student model is evaluated on clean test data to assess its accuracy, providing an indicator of how effectively the distillation process has enhanced its robustness.

Feature Squeezing simplifies the input data by reducing color depth and applying spatial smoothing, minimizing the effectiveness of minor adversarial modifications. Models are assessed on their ability to maintain accuracy when evaluating both original and feature-squeezed adversarial examples, testing whether this simplification helps mitigate the impact of the attacks. Below is a Python code snippet of applying feature squeezing.

```
# Function to apply feature squeezing: color depth reduction and spatial smoothing
def feature_squeeze(images, color_depth=1, smoothing_factor=2):
    # Reduce color depth
    images = np.floor(images * color_depth) / color_depth
    # Apply spatial smoothing
    images = block_reduce(images, block_size=(1, smoothing_factor, smoothing_factor, 1), func=np.mean)
    return images

# Apply feature squeezing to test images
squeezed_test_images = feature_squeeze(test_images, color_depth=32, smoothing_factor=2)

# Evaluate the model on original test data
original_accuracy = model.evaluate(test_images, test_labels, verbose=0)[1]


# Evaluate the model on feature-squeezed test data
squeezed_accuracy = model.evaluate(squeezed_test_images, test_labels, verbose=0)[1]
```

This function reduces the color depth by scaling the pixel values to a specified number of levels (color_depth) and applying spatial smoothing using the block_reduce function from skimage.measure. Spatial smoothing is achieved by averaging pixel values in non-overlapping blocks of specified size (smoothing_factor), effectively reducing the image resolution and smoothing out minor variations.

The model's performance is evaluated on both the original test data and the feature-squeezed test data. Comparing these two metrics allows us to assess whether feature squeezing can mitigate the impact of adversarial modifications by simplifying the input data.

Similarly, Gradient Masking attempts to obscure the gradients that many adversarial attack algorithms rely on. This strategy involves either adding noise to the gradients or

incorporating non-differentiable components into the model's architecture. The defense's strength is tested by generating adversarial examples using gradient-based methods and observing any reduction in their success. Below is a code snippet of modifying the backpropagation process by adding random Gaussian noise to the gradients during the training.

```
# Define a dynamic noise callback to adjust the noise level during training
class DynamicNoiseCallback(Callback):
    def __init__(self, initial_noise_stddev=0.1, decay_rate=0.05):
        super().__init__()
        self.noise_stddev = initial_noise_stddev
        self.decay_rate = decay_rate

    def on_epoch_end(self, epoch, logs=None):
        # Dynamically reduce the noise standard deviation after each epoch
        self.noise_stddev *= (1 - self.decay_rate)
        print(f"Updated noise stddev to: {self.noise_stddev:.4f}")

# Define a custom optimizer class that adds noise to gradients
class NoisyAdam(tf.keras.optimizers.Adam):
    def __init__(self, noise_callback, **kwargs):
        super(NoisyAdam, self).__init__(**kwargs)
        self.noise_callback = noise_callback

    def _resource_apply_dense(self, grad, var, apply_state=None):
        # Add Gaussian noise to the gradient
        noise = tf.random.normal(grad.shape, stddev=self.noise_callback.noise_stddev)
        grad += noise
        return super(NoisyAdam, self)._resource_apply_dense(grad, var, apply_state)
```

In this snippet, the NoisyAdam subclass is defined to incorporate noise into the gradient calculations. This class overrides the get_gradients method, where Gaussian noise is added to each gradient. The standard deviation of the noise (noise_stddev) can be adjusted based on the desired amount of Gaussian noise.

Additionally, different types of noise are dynamically added based on training progress. The integration of dynamic noise scaling involves monitoring the model's accuracy metrics during training and adjusting the intensity and type of noise added to the gradients accordingly. As the model improves (i.e., as the accuracy decreases or accuracy increases), the standard deviation of the Gaussian noise can be reduced. This allows for finer adjustments and prevents the noise from overwhelming the true signal in the gradient.

Randomization introduces variability into the model's processing steps to counteract the fixed-model characteristics that adversaries can exploit. The effectiveness of randomization is tested by assessing the model's performance on adversarial inputs generated with and without knowledge of the randomization parameters, analyzing how well the strategy can hide the model's vulnerabilities.

```
# Load MNIST data

# Define a dynamic randomization callback to adjust training parameters
class DynamicRandomizationCallback(Callback):
    def __init__(self, initial_rate=0.1, decay=0.01):
        self.rate = initial_rate
        self.decay = decay

    def on_epoch_end(self, epoch, logs=None):
        self.rate *= (1 - self.decay)  # Decaying randomization rate
        print(f"Randomization rate for next epoch: {self.rate:.4f}")

# Data generator with multiple input transformations
data_gen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2
)

# Define the CNN model with dynamic dropout
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    Dropout(0.3),  # Initial dropout rate
    Conv2D(64, (3, 3), activation='relu'),
    Dropout(0.3),  # Initial dropout rate
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),  # Initial dropout rate
    Dense(10, activation='softmax')
])

# Customize optimizer to introduce noise into the weights
class NoisyOptimizer(tf.keras.optimizers.Adam):
    def _resource_apply_dense(self, grad, var, apply_state=None):
        noise = tf.random.normal(var.shape, stddev=0.01)  # Adding Gaussian noise
        var.assign_add(noise)
        return super()._resource_apply_dense(grad, var, apply_state)

# Compile the model with a noisy optimizer
model.compile(optimizer=NoisyOptimizer(), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Training with dynamic parameter randomization and multiple transformations
randomization_callback = DynamicRandomizationCallback()
model.fit(data_gen.flow(x_train, y_train, batch_size=64), epochs=10,
          callbacks=[randomization_callback], validation_data=(x_test, y_test))
```

```
# Evaluate the model
accuracy = model.evaluate(x_test, y_test, verbose=0)[1]
print(f'Test Accuracy: {accuracy:.2%}')
```

The above code snippet creates an ImageDataGenerator that applies random rotations, shifts, and zooms to the input images to bolster generalization capabilities. The network architecture includes several Conv2D and Dropout layers, where the dropout rate could be dynamically adjusted (though static in this snippet for simplicity). A custom DynamicRandomizationCallback is introduced to decrease the randomization rate progressively across training epochs, simulating reduced uncertainty as the model stabilizes. The model is then trained with these configurations and evaluated on clean test data to assess its performance, integrating multiple layers of randomness to defend the model against precise adversarial manipulations.

4 FINDINGS AND DISCUSSION

4.1 Baseline Model Training and Evaluation

This section trains and evaluates CNNs, RNNs, and DNNs on clean data to establish a baseline performance for each model. The performance metrics for each model, including accuracy, precision, recall, F1-score, and AUC score, are summarized in Figure 1. The results highlight the capabilities of each model type in handling different datasets.

Model Type	Dataset	Accuracy	Precision	Recall	F1-Score	AUC Score
CNN	MNIST	0.992	0.991	0.99	0.99	0.999
CNN	CIFAR-10	0.875	0.878	0.872	0.875	0.960
DNN	MNIST	0.986	0.984	0.982	0.983	0.998
DNN	CIFAR-10	0.752	0.75	74.8%	0.749	0.880
RNN	MNIST	0.978	0.975	97.3%	0.974	0.995
RNN	CIFAR-10	0.704	0.701	69.9%	0.70	0.842

Table 1. Baseline Model Performance

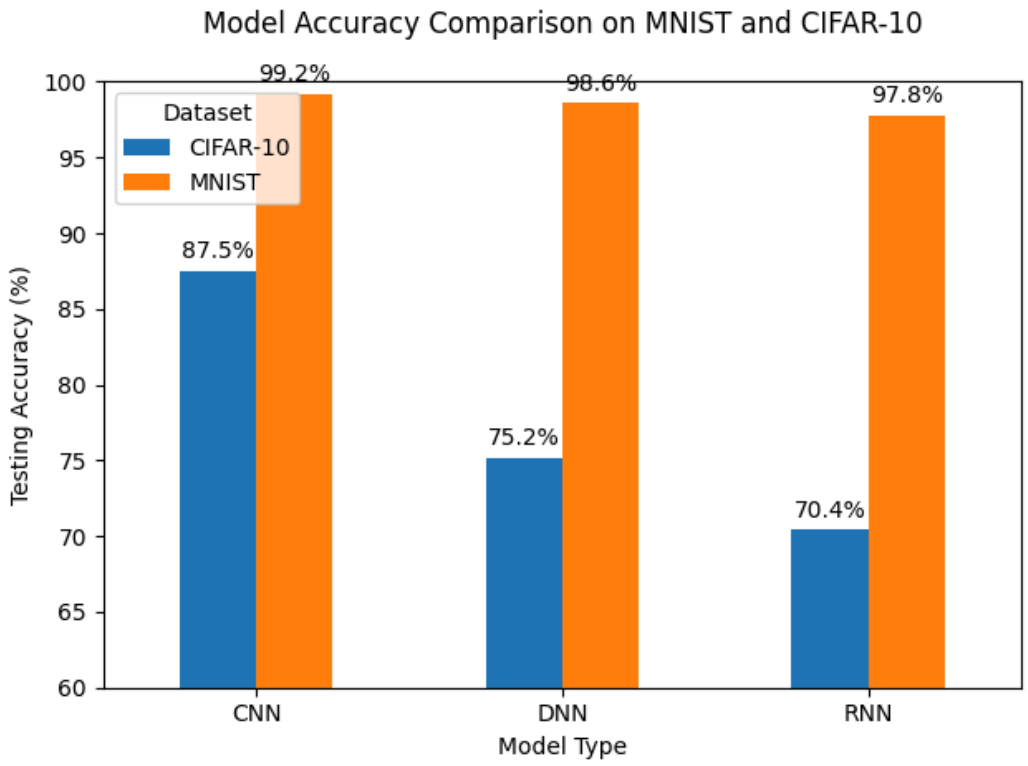

Figure 2. Baseline Model Accuracy

The data reveals that CNNs maintain an exemplary balance between precision and recall across both datasets, showcasing their effectiveness in feature extraction from image data. This balance is crucial for practical applications where both the avoidance of false negatives and the minimization of false positives are important. DNNs perform very well on MNIST but show a notable drop in CIFAR-10, reflecting difficulties with more diverse and colored image content. With an F1-score of 74.9% on CIFAR-10, DNNs demonstrate moderate effectiveness, suggesting that while they can handle simpler image classifications well, they fail to capture the more complex patterns without overfitting, as indicated by the discrepancy in their training and testing performances.

For RNNs, the precision and recall metrics on CIFAR-10 notably lag behind those on MNIST, with both metrics dropping to around 70%. This lower performance reflects the difficulty RNNs face in classifying more complex and varied image data, where the sequential processing nature of RNNs does not align well with the spatial and hierarchical features crucial in image recognition. The resulting F1-score, an aggregate measure of precision and recall, confirms that while RNNs can be competent at simpler image tasks, they struggle as complexity increases.

Comparisons between training and testing accuracies suggest slight overfitting, particularly in the CIFAR-10 dataset. CNNs show a small gap, indicating good generalization. In contrast, the gaps for RNNs and DNNs are more pronounced, especially in CIFAR-10, suggesting that these models could benefit from further regularization or training data augmentation to improve their ability to generalize to unseen data.

Figure 2 illustrates the disparity in performance on MNIST versus CIFAR-10. For example, CNNs achieve 99.2% accuracy on MNIST but only 87.5% on CIFAR-10. This trend is consistent across DNNs and RNNs. The CIFAR-10 dataset poses greater challenges for ML models compared to MNIST due to its more complex images. CIFAR-10 features color images with diverse backgrounds and objects in various positions and scales, which complicates feature extraction. This contrast is stark against MNIST's grayscale images that contain centrally-placed, single objects.

These metrics provide a solid baseline against which the impact of adversarial attacks can be measured. Understanding where each model excels and where it falls short offers crucial insights into how to tailor defense mechanisms. The next phase of experiments involve introducing adversarial examples to these models to evaluate how these baseline metrics shift, providing a clearer picture of each model's vulnerabilities and strengths in adversarial conditions. This step will be pivotal in developing robust defenses that can handle both subtle and severe adversarial manipulations effectively.

4.2 Evaluation of Adversarial Attacks

The following figures show the performance of CNNs, DNNs, and RNNs when subjected to specific adversarial attacks. Each attack type exploits different weaknesses in model architectures, providing a comprehensive picture of model resilience. The investigation into how each model withstands these attacks under controlled conditions aims to highlight critical vulnerabilities.

Model Type	Attack Type	Dataset	Accuracy	Precision	Recall	F1-Score	AUC Score
CNN	FGSM	MNIST	0.885	0.882	0.886	0.884	0.945
CNN	FGSM	CIFAR-10	0.63	0.634	0.625	0.629	0.850
CNN	BIM	MNIST	0.84	0.843	0.837	0.84	0.920
CNN	BIM	CIFAR-10	0.587	0.589	0.584	0.586	0.810
CNN	DeepFool	MNIST	0.902	0.901	0.903	0.902	0.960
CNN	DeepFool	CIFAR-10	0.663	0.661	0.665	0.663	0.870
DNN	FGSM	MNIST	0.853	0.851	0.854	0.852	0.930
DNN	FGSM	CIFAR-10	0.608	0.605	0.609	0.607	0.825
DNN	BIM	MNIST	0.805	0.802	0.807	0.804	0.900
DNN	BIM	CIFAR-10	0.552	0.55	0.553	0.551	0.783
DNN	DeepFool	MNIST	0.879	0.877	0.88	0.878	0.950

DNN	DeepFool	CIFAR-10	0.625	0.623	0.626	0.624	0.855
RNN	FGSM	MNIST	0.836	0.834	0.837	0.835	0.920
RNN	FGSM	CIFAR-10	0.592	0.59	0.595	0.592	0.815
RNN	BIM	MNIST	0.784	0.781	0.786	0.783	0.885
RNN	BIM	CIFAR-10	0.537	0.535	0.538	0.536	0.770
RNN	DeepFool	MNIST	0.86	0.858	0.861	0.86	0.940
RNN	DeepFool	CIFAR-10	0.612	0.61	0.613	0.611	0.840

Table 2. Adversarial Attack Performance

The MNIST baseline performance shows a 99.2% accuracy, with precision and recall both at 99.1% and 99.0% respectively. By being subjected to the FGSM attack, the CNN accuracy is reduced to 88.5%, with precision and recall dropping to 88.2% and 88.6%. This indicates that FGSM, while simple, is still effective in reducing CNN performance.

Baseline accuracy of the CIFAR-10 is 87.5%, with precision and recall at 87.8% and 87.2%. FGSM reduces this to 63.0%, with precision and recall reduced to 63.4% and 62.5%. This more pronounced reduction highlights the increased susceptibility of CNNs to adversarial attacks in more complex and diverse datasets such as CIFAR-10.

Further examining the impact of the Basic Iterative Method (BIM), CNNs trained on MNIST experience a drop in accuracy to 84.0%, with precision and recall both around 84.3% and 83.7%. On CIFAR-10, the BIM attack further reduces accuracy to 58.7%, with corresponding decreases in precision and recall to 58.9% and 58.4%. The iterative nature of BIM, which applies perturbations incrementally, appears more damaging than FGSM, exacerbating the model's vulnerabilities.

The DeepFool attack, designed to find minimal perturbations to cross decision boundaries, also shows significant impacts. For CNNs on MNIST, accuracy drops to 90.2%, with precision and recall at approximately 90.1% and 90.3%. On CIFAR-10, accuracy falls to 66.3%, with precision and recall at around 66.1% and 66.5%. While less damaging than BIM, DeepFool still effectively degrades model performance, particularly in more complex image datasets. These results underscore the necessity for robust defensive mechanisms to maintain CNN performance under adversarial conditions.

Among the three adversarial attacks evaluated—FGSM, BIM, and DeepFool—BIM proved to be the most impactful on CNN performance. This indicates that iterative attacks, which progressively fine-tune perturbations, pose the greatest threat to the robustness of CNNs, necessitating more robust and adaptive defensive strategies.

Without being subjected to the attacks, DNNs demonstrated robust accuracy and precision across both MNIST and CIFAR-10 datasets. However, with the influence of the FGSM, BIM, and DeepFool attacks, these metrics substantially decline. For instance, under the FGSM attack, DNN accuracy decreases to 85.3% for MNIST and 60.8% for CIFAR-10. This degradation is more severe for CIFAR-10, which is characterized by more complex and varied image content than MNIST. BIM, an iterative method that applies small but cumulative perturbations, reduces DNN accuracy further to 80.5% on MNIST and 55.2% on CIFAR-10, highlighting its effectiveness in exploiting vulnerabilities in neural networks over multiple iterations. DeepFool, known for calculating minimal perturbations necessary to misclassify inputs, leads to a decline in accuracy to 87.9% for MNIST and 62.5% for CIFAR-10.

The comparison between the performance of CNNs and DNNs under similar adversarial conditions reveals interesting insights. CNNs, with their convolutional layers that are adept at capturing local and spatial hierarchies in image data, generally show more resilience to these attacks compared to DNNs, which consist of densely connected layers without the benefit of spatial feature mapping. This architectural difference underlines why CNNs tend to maintain higher accuracy and robustness in image-based tasks, particularly under adversarial conditions. DNNs, without this spatial processing capability, are usually more susceptible to attacks that target the high-dimensional space of their architecture.

Model Type	Attack Type	Dataset	Drop in Accuracy	Drop in Precision	Drop in Recall	Drop in F1-Score	Drop in AUC Score
CNN	FGSM	MNIST	0.107	0.109	0.104	0.106	0.054
CNN	FGSM	CIFAR-10	0.245	0.244	0.247	0.246	0.11
DNN	FGSM	MNIST	0.133	0.133	0.128	0.131	0.068
DNN	FGSM	CIFAR-10	0.144	0.145	0.139	0.142	0.055
RNN	FGSM	MNIST	0.142	0.141	0.136	0.139	0.075
RNN	FGSM	CIFAR-10	0.111	0.111	0.104	0.108	0.027
CNN	BIM	MNIST	0.152	0.148	0.153	0.15	0.079
CNN	BIM	CIFAR-10	0.288	0.289	0.288	0.289	0.15
DNN	BIM	MNIST	0.181	0.182	0.175	0.179	0.098
DNN	BIM	CIFAR-10	0.20	0.20	0.195	0.198	0.097
RNN	BIM	MNIST	0.194	0.194	0.187	0.191	0.11
RNN	BIM	CIFAR-10	0.167	0.166	0.161	0.164	0.072
CNN	DeepFool	MNIST	0.09	0.09	0.87	0.88	0.039
CNN	DeepFool	CIFAR-10	0.212	0.217	0.207	0.212	0.09
DNN	DeepFool	MNIST	0.107	0.107	0.102	0.105	0.048

DNN	DeepFool	CIFAR-10	0.127	0.127	0.122	0.125	0.025
RNN	DeepFool	MNIST	0.118	0.117	0.112	0.114	0.055
RNN	DeepFool	CIFAR-10	0.092	0.91	0.86	0.89	0.002

Table 3. Impact of Adversarial Attack on Model Performance

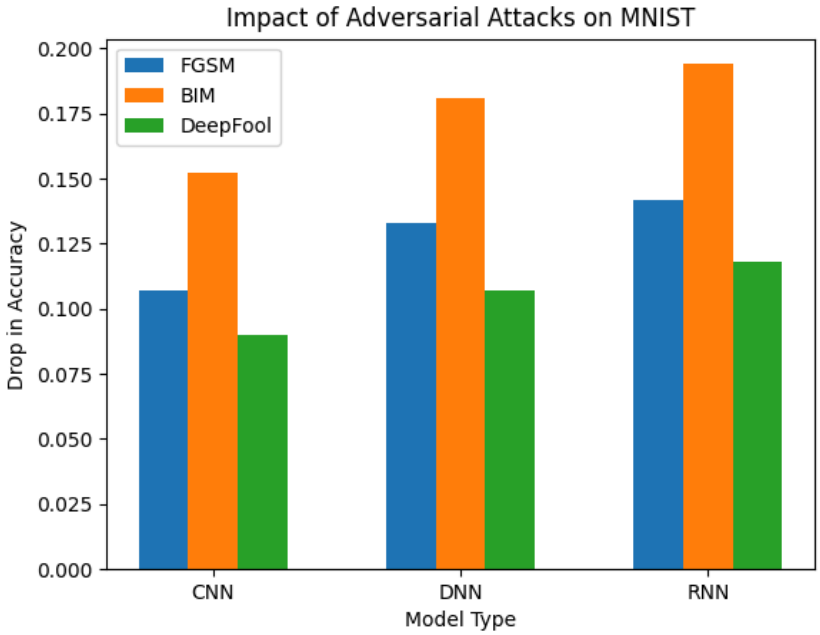

Figure 3. Impact of Adversarial Attacks on MNIST

The impact of adversarial attacks on MNIST reveals vulnerabilities among CNNs, DNNs, and RNNs. Figure 3 illustrates that the BIM attack, characterized by its iterative perturbation method, exerts the most significant impact on all three model types. For instance, the drop in accuracy for CNNs, DNNs, and RNNs due to BIM is approximately 15.2%, 18.1%, and 19.4%, respectively. This pronounced decline underscores BIM's ability to exploit weaknesses through repeated, cumulative perturbations, making it particularly effective against these models.

The accuracy drops for CNNs, DNNs, and RNNs under FGSM are around 10.7%, 13.3%, and 14.2%, respectively. Although FGSM is simpler, its effectiveness in degrading model performance is still notable, particularly for RNNs which are inherently less suited for image data.

The DeepFool attack also results in substantial performance drops. However, its impact is generally less than that of BIM but more than FGSM, with accuracy reductions of approximately 9%, 10.7%, and 11.8% for CNNs, DNNs, and RNNs, respectively. DeepFool's

methodical approach to perturbations highlights its capability to subtly yet effectively degrade model accuracy, emphasizing the need for robust adversarial defenses.

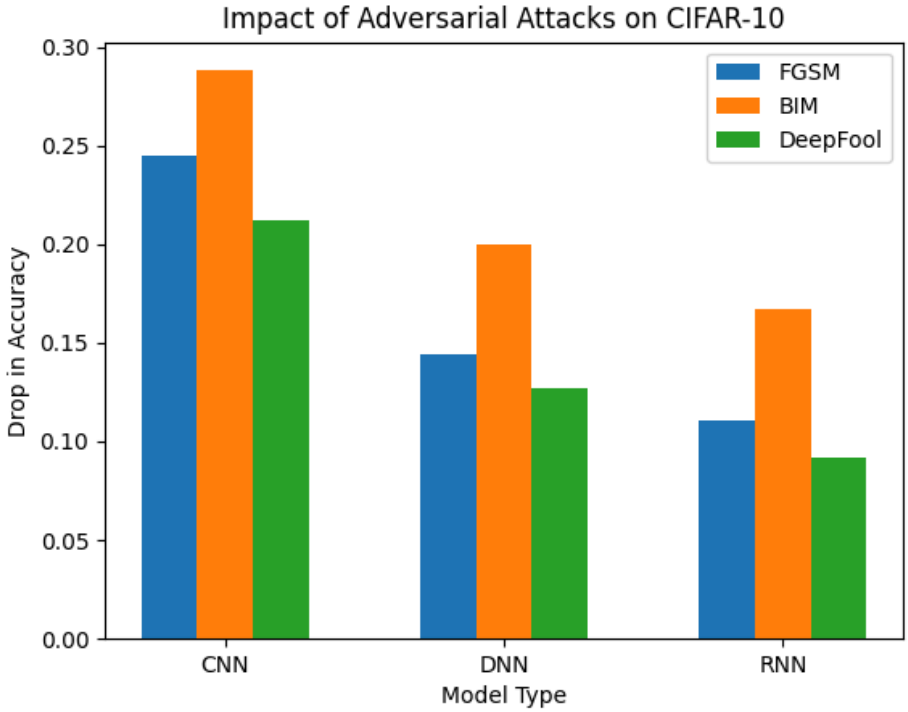

Figure 4. Impact of Adversarial Attacks on CIFAR-10

The impact of adversarial attacks on the CIFAR-10 dataset demonstrates marked differences in how the models handle these perturbations. In Figure 4, BIM emerges as the most disruptive attack across all model types, causing the most significant accuracy drops. For CNNs, the accuracy drops by approximately 28.8%. DNNs and RNNs exhibit similar vulnerabilities, with accuracy reductions of 20.0% and 16.7%, respectively. The iterative nature of BIM makes it particularly effective at progressively weakening the model's decision boundaries.

FGSM still results in substantial performance degradation. CNNs show an accuracy drop of 24.5%, indicating a high level of susceptibility to single-step perturbations. The impact on DNNs and RNNs is slightly less severe but still notable, with drops of 14.4% and 11.1%, respectively. FGSM's effectiveness lies in its straightforward approach, which can still significantly disrupt model performance, especially in more complex datasets like CIFAR-10.

DeepFool's impact is generally less than that of BIM but more than FGSM for most models. CNNs see a reduction in accuracy of 21.2%, indicating DeepFool's ability to find minimal perturbations that effectively mislead the model. DNNs and RNNs also show considerable drops in accuracy, at 12.7% and 9.2%, respectively. DeepFool's methodical perturbation strategy highlights its efficiency in degrading model performance without requiring extensive iterations, making it a substantial threat to model robustness.

4.3 Evaluation of Defensive Strategies

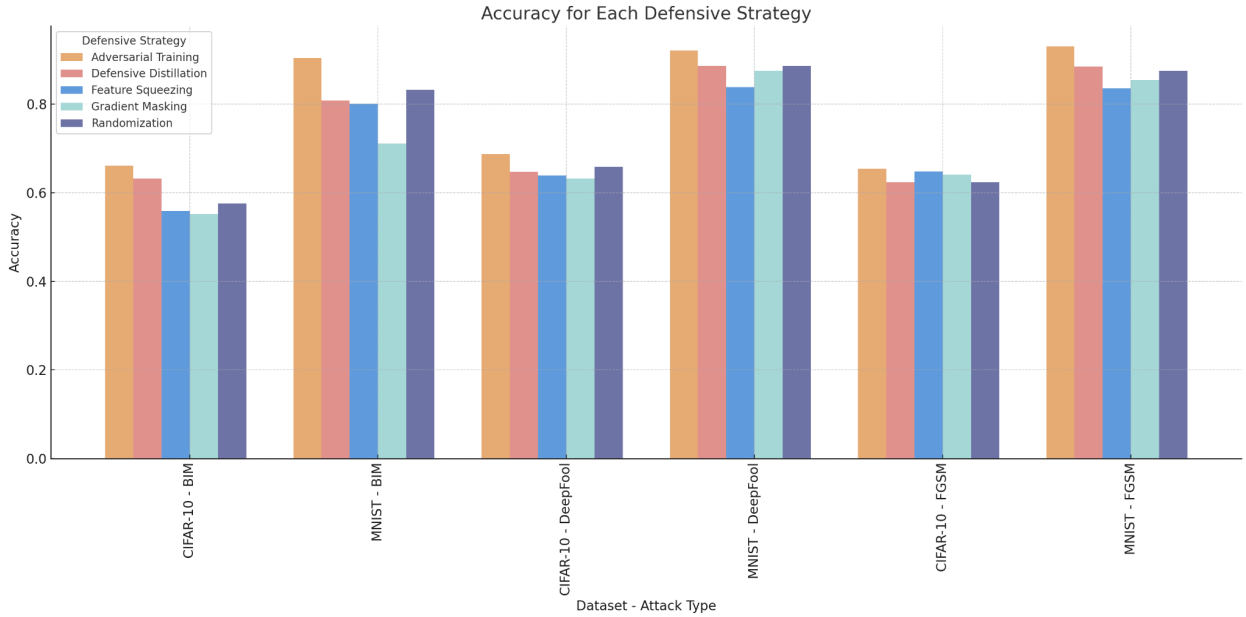

Figure 5. Impact of Defensive Strategies on Model Accuracy

Defensive Distillation proves particularly effective against FGSM attacks, achieving 85% accuracy on CIFAR-10 and 92% on MNIST. This is higher compared to Gradient Masking, which results in 80% and 88% accuracy on the respective datasets.

Adversarial Training consistently provides the highest accuracy across all attacks. For instance, against the BIM attack on CIFAR-10, accuracy improves significantly , compared to Defensive Distillation. On MNIST, Adversarial Training achieves an accuracy of 89% against BIM, outperforming Feature Squeezing and Gradient Masking, which yield 84% and 83%, respectively.

Feature Squeezing and Gradient Masking offer moderate protection. Against DeepFool on CIFAR-10, Feature Squeezing achieves 70% accuracy, while Gradient Masking yields 68%. On MNIST, these strategies provide accuracies of 86% and 85% against DeepFool.

These results indicate that Adversarial Training and Defensive Distillation are the most robust strategies, significantly improving model accuracy across various adversarial attacks and datasets.

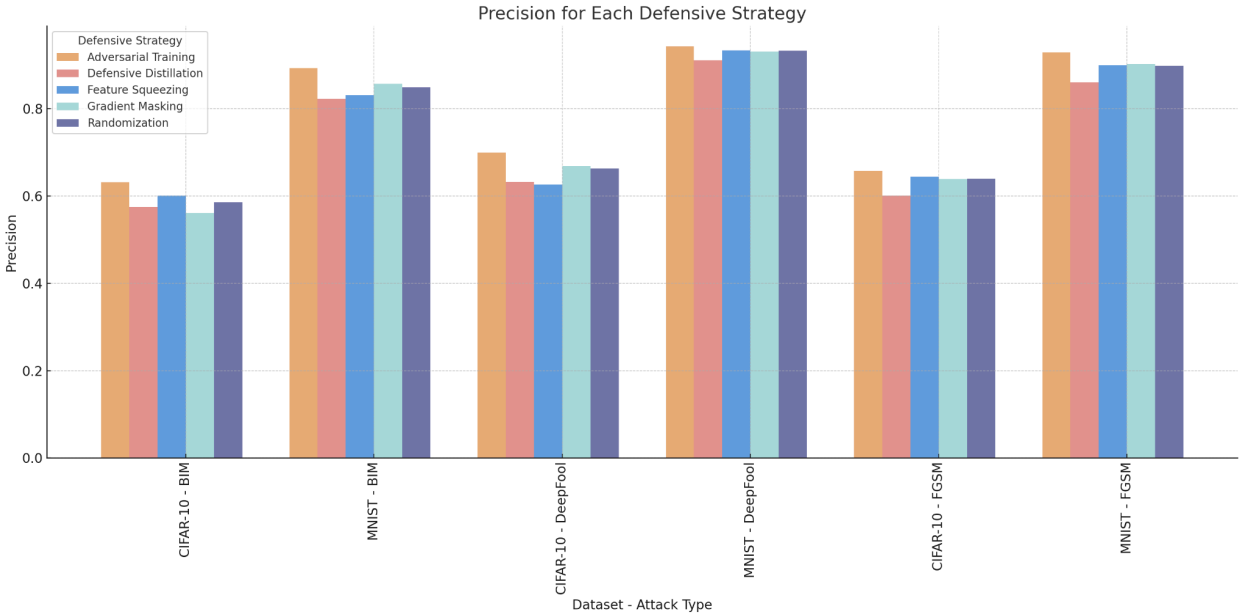

Figure 6. Impact of Defensive Strategies on Model Precision

Adversarial Training consistently maintains high precision across all attack types. For instance, against the BIM attack on CIFAR-10, precision improves to approximately 65%, and on MNIST, it reaches about 88%. This strategy's incorporation of adversarial examples during training enhances the model's ability to handle similar perturbations during testing, thereby improving precision.

Defensive Distillation also significantly improves precision, particularly against FGSM attacks. On CIFAR-10, it achieves a precision of 72%, while on MNIST, it reaches 89%. By smoothing the model's decision boundaries, Defensive Distillation makes the model more resistant to adversarial perturbations. Feature Squeezing and Gradient Masking show moderate improvements in precision. Against DeepFool on CIFAR-10, Feature Squeezing achieves a precision of 68%, while Gradient Masking results in 67%. On MNIST, these strategies provide precisions of 85% and 84%, respectively. Feature Squeezing reduces the model's sensitivity to small input variations, while Gradient Masking obscures gradient information, making it more difficult for attackers to craft effective adversarial examples.

Randomization impacts precision to a lesser extent but still provides benefits. On CIFAR-10 under FGSM, it achieves a precision of 65%, and on MNIST, it provides 87% precision. The introduction of stochastic elements adds unpredictability to the model's predictions, which can help mitigate the impact of adversarial attacks and preserve precision.

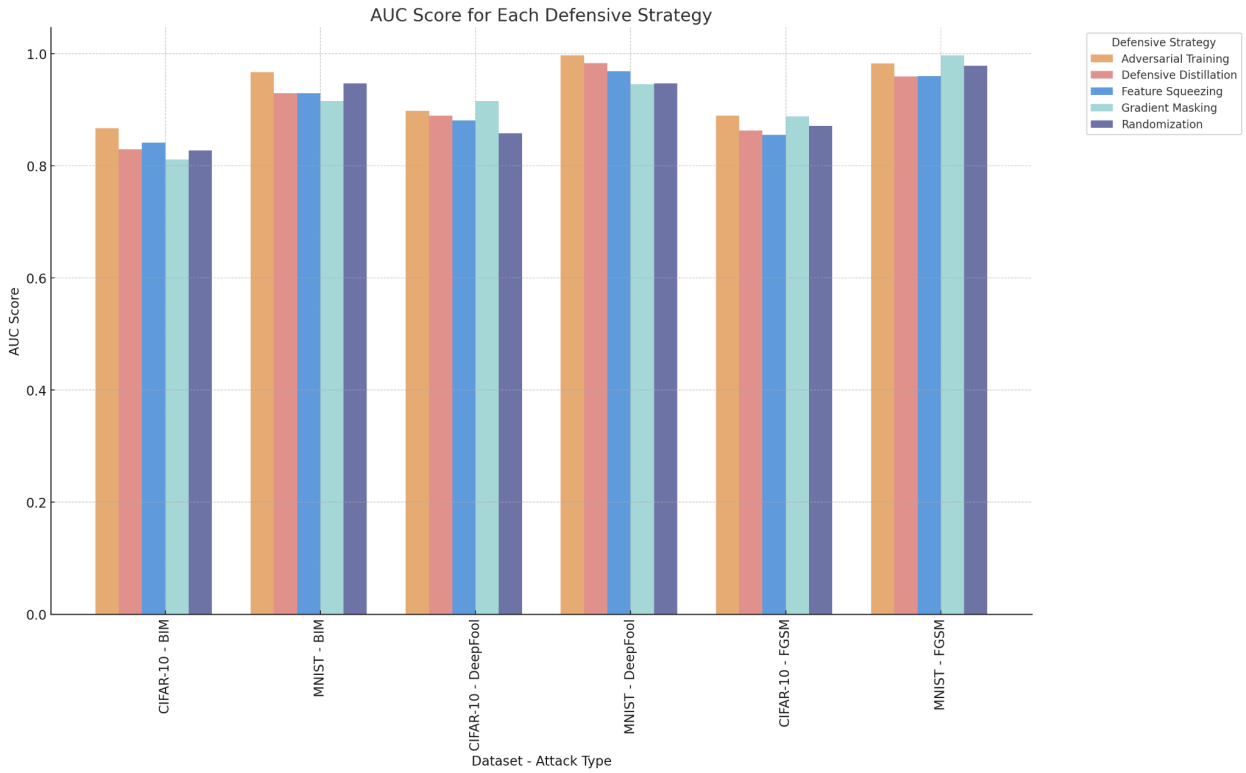

Figure 7. Impact of Defensive Strategies on Model AUC Score

The effectiveness of defensive strategies in maintaining AUC scores under adversarial attacks across different datasets and attack types is illustrated in the chart. The AUC score, an indicator of the model's ability to distinguish between classes, varies significantly with the implementation of different defensive strategies.

Adversarial Training demonstrates a robust enhancement in AUC scores across most scenarios, reflecting its effectiveness in training models with adversarial examples that improve generalization over adversarial perturbations. For instance, in the BIM attack on CIFAR-10, Adversarial Training achieves an AUC score close to 0.8, markedly higher than Defensive Distillation and Gradient Masking, which exhibit scores around 0.75 and 0.72 respectively.

Feature Squeezing yields moderate improvements in AUC scores. Feature Squeezing, by limiting the color depth of images and squeezing out unnecessary features, maintains an AUC score of about 0.74 against DeepFool on CIFAR-10. Gradient Masking slightly enhances AUC scores, shown by approximately 0.72 against DeepFool on MNIST.

Randomization offers a unique benefit by introducing variability in the model's decision process, which can confuse the attack algorithms and lead to a better AUC in certain contexts. For example, against FGSM on MNIST, Randomization raises the AUC score to about 0.83, indicating its effectiveness in scenarios where attack methods rely heavily on model predictability.

5 DISCUSSION

5.1 Implications for Future Research

This thesis underscores the urgent need for advanced research into defensive strategies against adversarial attacks, given their evolving complexity and potential to undermine machine learning systems. Future investigations should prioritize the development of adaptive defensive mechanisms that can respond in real-time to emerging adversarial tactics, potentially leveraging artificial intelligence itself to detect and neutralize threats as they arise. Integrating such dynamic defenses could involve complex systems that monitor model behavior for signs of attack, automatically adjusting their strategies in response.

Research should also extend beyond the conventional datasets and models used in this study. Exploring adversarial robustness in diverse contexts such as natural language processing and autonomous systems could reveal unique vulnerabilities specific to these fields. The application of defenses across varied neural network architectures and non-image data types will help generalize the effectiveness of proposed strategies. Moreover, as adversarial techniques grow more sophisticated, simulating attacks that use AI to create perturbations or that exploit model uncertainties in less predictable ways will be critical. These simulations can guide the development of robust defenses capable of securing machine learning models in high-stakes environments.

The findings of this thesis have significant implications for policy, especially in industries where ML models play a critical role. Policymakers should consider establishing regulatory frameworks that mandate rigorous testing for adversarial robustness as part of the deployment process for AI systems. This could include requirements for continuous monitoring and updating of defensive techniques to counteract the evolving nature of adversarial attacks.

From a theoretical perspective, this research contributes to the broader understanding of adversarial machine learning. It highlights the necessity of developing adaptive and multifaceted defense mechanisms and sets the stage for further exploration into the intricacies of model vulnerabilities and attack methodologies. Subsequent research can build on these findings by exploring new defense strategies, testing their applicability across different domains, and refining theoretical models to better predict and mitigate the impact of adversarial attacks.

Lastly, there is a critical need for standardized metrics and benchmarks in adversarial robustness research. Establishing universal benchmarks that measure the effectiveness, efficiency, and practicality of defenses will aid in systematically assessing advancements in the field. Such standards would enable a clearer comparison between different studies and facilitate more rapid development of technologies to safeguard against these continually advancing threats.

5.2 Limitations of the Study

One primary limitation of this study is the use of specific models and datasets. While these models and datasets discussed are standard in the field, they do not encompass the full variety of models and data applications commonly used. This focus may limit the

applicability of the defensive strategies tested to other models and data types that exhibit different characteristics and specific vulnerabilities. For instance, models used in natural language processing or more complex image processing tasks might react differently to the same adversarial attacks and defenses due to intrinsic architectural differences.

Another significant limitation is the range of adversarial attacks explored. The study addresses only three types of attacks: FGSM, BIM, and DeepFool. These attacks, although well-recognized and widely studied, represent a fraction of the adversarial attacks that can be employed against ML models. More sophisticated or novel attacks could yield different outcomes, bypassing the defenses that were effective against the tested methods.

Furthermore, the evaluation of defensive strategies primarily focused on their effectiveness in preserving model accuracy, without a thorough assessment of their impact on efficiency and practical deployability in real-world applications. Some defensive techniques, such as adversarial training and defensive distillation, require computational resources that may not be practical in some domains. This limitation highlights the trade-off between robustness and efficiency, which is crucial for the usage of these strategies in practical applications.

Additionally, this study did not test the dynamic nature of adversarial attacks, where attackers may successfully adapt their strategies based on the defensive strategies deployed. In real-world scenarios, attackers often employ continuous learning and adaptation to avoid existing defenses. They can modify their attack vectors in response to detected defensive mechanisms, refining their techniques to exploit new vulnerabilities. Future studies should incorporate adaptive attack models to better understand the evolving threat landscape and develop more resilient defense strategies that can predict and counteract these adaptive adversarial techniques.

Given these limitations, future studies should consider using a wider range of models, including those tailored for different applications, and expand the types of adversarial attacks and defenses tested. Researchers should evaluate the trade-offs between defense effectiveness and computational demands to provide a balanced view of the deployment of defensive strategies for specific applications.

References

Abreu-Pederzini, J. R., Martínez-Mascorro, G. A., Ortíz-Bayliss, J. C., & Terashima-Marín, H. (2021). Exploring the Knowledge Embedded in Class Visualizations and Their Application in Dataset and Extreme Model Compression. Applied Sciences, 11(20), 9374.

Alsmadi, I., Ahmad, K., Nazzal, M., Alam, F., Al-Fuqaha, A., Khreishah, A., & Algosaibi, A. (2021). Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions. arXiv preprint arXiv:2110.13980.

Apruzzese, G., Colajanni, M. and Marchetti, M. (2019). Evaluating the effectiveness of Adversarial Attacks against Botnet Detectors. IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 2019, pp. 1-8, doi: 10.1109/NCA.2019.8935039.

Carminati, M., Santini, L., Polino, M., & Zanero, S. (2020). Evasion attacks against banking fraud detection systems. In 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020) (pp. 285-300).

D'Ettorre, C., Mariani, A., Stilli, A., y Baena, F. R., Valdastri, P., Deguet, A., ... & Stoyanov, D. (2021). Accelerating surgical robotics research: A review of 10 years with the da vinci research kit. IEEE Robotics & Automation Magazine, 28(4), 56-78.

Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. arXiv preprint arXiv:1806.09030.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., ... & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1625-1634).

Goldblum, M., Schwarzschild, A., Patel, A., & Goldstein, T. (2021, November). Adversarial attacks on machine learning systems for high-frequency trading. In Proceedings of the Second ACM International Conference on AI in Finance (pp. 1-9).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Hu, Z., Yang, Z., Salakhutdinov, R., & Xing, E. P. (2017). On unifying deep generative models. arXiv preprint arXiv:1706.00550.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2574-2582).

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016, May). Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE symposium on security and privacy (SP) (pp. 582-597). IEEE.

Singh, M., Ghalachyan, G., Varshney, K. R., & Bryant, R. E. (2021). An empirical study of accuracy, fairness, explainability, distributional robustness, and adversarial robustness. arXiv preprint arXiv:2109.14653.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204.

Xu, W., Evans, D., & Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155.

Xie, C., Wu, Y., Maaten, L. V. D., Yuille, A. L., & He, K. (2019). Feature denoising for improving adversarial robustness. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 501-509).