Adversarial Robustness in Convolutional Neural Networks for Synthetic Aperture Radar (SAR) Imagery

Asher Ellis
asher.ellis@yale.edu

Advisor: Dr. Alex Wong
alex.wong@yale.edu

A Senior Thesis as a partial fulfillment of requirements for the Bachelor of Science in Applied Mathematics

Yale University
December 16, 2024

Acknowledgments

I would like to express my gratitude to my senior thesis advisor, Dr. Alex Wong, for his invaluable guidance and support throughout this project. I also am deeply grateful to my parents for always encouraging me to pursue my dreams. Finally, I want to thank my friends, who I'm lucky to learn from every day.

Abstract

This thesis investigates the adversarial robustness of convolutional neural networks (CNNs) for synthetic aperture radar (SAR) imagery, focusing on the classification of Soviet military vehicles from the MSTAR dataset. SAR technology is essential for defense and surveillance due to its ability to generate high-resolution imagery under all weather and lighting conditions. However, machine learning models for SAR are highly vulnerable to adversarial attacks—malicious perturbations that degrade performance. This study evaluates five CNN architectures, including custom and pre-trained models, under two adversarial attacks: the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD), with PGD serving as a powerful iterative benchmark. Results reveal significant trade-offs between clean accuracy and adversarial robustness. Adversarial training, which integrates adversarial examples into the training process, effectively enhances robustness, with models like Deeper CNN achieving a balance between clean accuracy and resilience to attacks. These findings highlight the challenges of designing robust models for high-stakes SAR applications.

Contents

1 Introduction

Recent advancements in machine learning have transformed applications in defense, surveillance, and environmental monitoring. Synthetic Aperture Radar (SAR), a critical technology in these fields, provides high-resolution imagery under all weather and lighting conditions. Unlike optical systems, SAR operates by simulating a large antenna aperture through the motion of a radar antenna, producing detailed imagery capable of detecting and classifying objects across vast areas. This capability enables SAR to penetrate atmospheric interference, such as clouds and rain, making it indispensable for tasks like military reconnaissance and disaster response.

However, the growing reliance on machine learning models introduces vulnerabilities: adversarial attacks exploit these models, leading to high-confidence misclassifications with potentially severe consequences. The Projected Gradient Descent (PGD) attack, in particular, is among the most powerful iterative benchmarks, revealing the susceptibility of models in high-dimensional feature spaces.

This thesis addresses the adversarial robustness of CNNs trained on SAR imagery, focusing on the classification of Soviet military vehicles using the MSTAR dataset. Five models are evaluated, including three custom and two pre-trained CNN architectures, against FGSM and PGD attacks, employing adversarial training to enhance model robustness. This study aims to identify effective strategies for designing reliable models in SAR applications.

2 Background

2.1 Synthetic Aperture Radar (SAR) Imagery

SAR is an advanced radar technology that constructs high-resolution images by simulating a large antenna aperture through the motion of a smaller physical antenna. This capability enables all-weather, day-and-night imaging, making SAR invaluable for defense, environmental monitoring, and disaster response. By transmitting microwave signals and analyzing the reflected signals, SAR can penetrate atmospheric interference, such as clouds and fog, to produce detailed imagery [6].

Despite its advantages, SAR imagery presents unique challenges. Speckle noise, caused by the coherent nature of radar signals, obscures fine details, complicating feature extraction. Additionally, geometric dependencies—interactions between radar signals and object properties such as orientation, material, and surface roughness—affect target appearance. Azimuthal variability further distorts object representations, requiring machine learning models to generalize effectively across diverse perspectives [6]. These challenges are particularly relevant to this study, as the MSTAR dataset includes SAR imagery under varying conditions.

2.2 Adversarial Attacks on Machine Learning Models

Adversarial attacks are malicious perturbations designed to deceive machine learning models. The Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al., is a single-step attack that perturbs inputs in the direction of the loss gradient, making it computationally efficient yet effective for revealing model vulnerabilities [2]. Another attack, Projected Gradient Descent (PGD), refines adversarial perturbations through iterative updates, producing stronger adversarial examples. PGD, introduced by Madry et al., is widely regarded as one of the most powerful benchmark attacks for evaluating adversarial robustness, especially in high-dimensional feature spaces [5]. These attacks exploit the vulnerabilities of neural networks, presenting critical challenges for applications where misclassifications could lead to improper decisions with severe consequences, such as in SAR-based military or surveillance operations.

2.3 Defensive Strategies

Adversarial training is a widely adopted defense mechanism, integrating adversarial examples into the training process to create robust decision boundaries [5]. This approach solves a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\eta\|_\infty \leq \epsilon} \mathcal{L}(f_\theta(x+\eta), y) \right].$$

The inner maximization generates adversarial examples $x + \eta$ by maximizing the loss function within a bounded perturbation (ϵ), while the outer minimization updates model parameters to minimize the loss over both clean and adversarial examples. Other possible strategies include defensive distillation, gradient masking, and adversarial detection, but these were not explored for the purposes of this project.

3 Methodology

3.1 Dataset

This thesis utilizes the Moving and Stationary Target Acquisition and Recognition (MSTAR) dataset, a gold standard in synthetic aperture radar (SAR) research [4]. MSTAR contains high-resolution grayscale radar imagery of military targets captured under diverse conditions, making it a robust resource for evaluating machine learning models in SAR-based applications. The dataset comprises eight target classes, including the 2S1 Self-Propelled Howitzer (1,164 samples), BRDM-2 Reconnaissance Vehicle (1,415 samples), BTR-60 Armored Personnel Carrier (1,353 samples), D7 Bulldozer (573 samples), SLICY Calibration Object (1,270 samples), T-62 Tank (1,144 samples), ZIL-131 Cargo Truck (1,146 samples), and the ZSU-23-4 Shilka Anti-Aircraft Gun (1,401 samples). These ample sample sizes ensure adequate data for training, validation, and testing to support effective model evaluation.

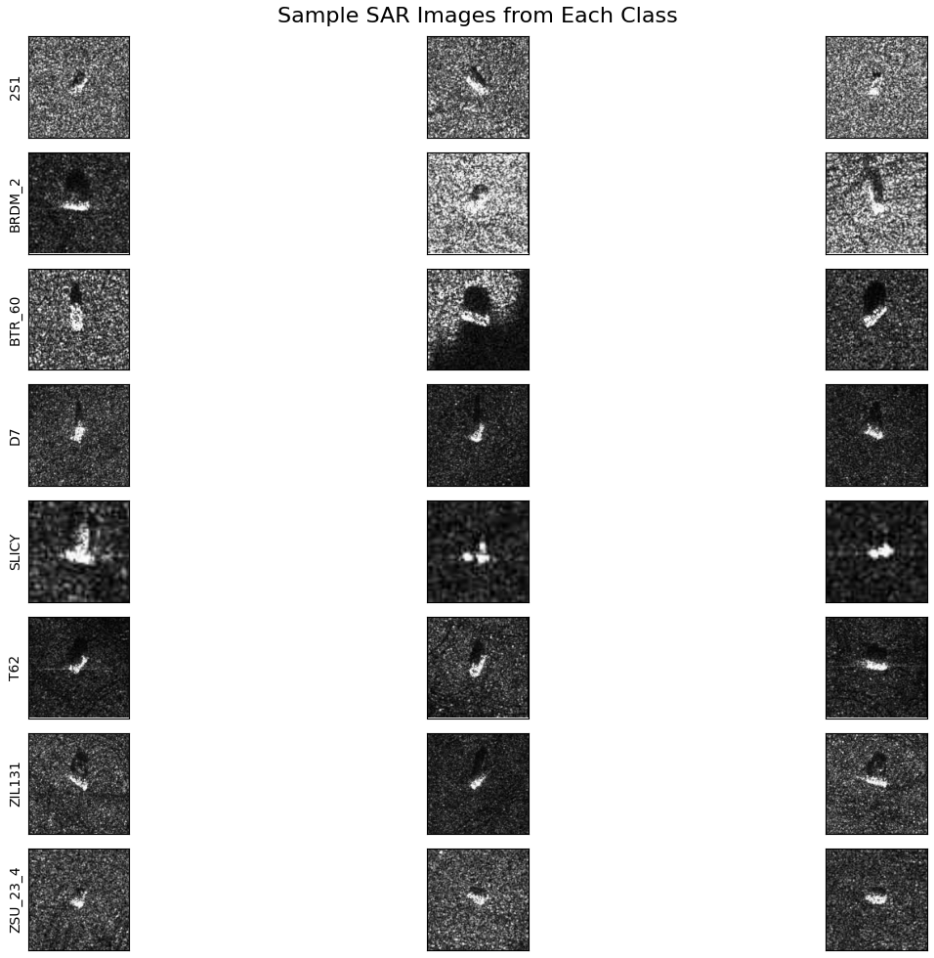

Figure 1: Sample SAR Images from Each Class in the MSTAR Dataset

The images in the dataset vary by azimuth angle, object orientation, and environmental conditions, simulating real-world complexities. However, SAR imagery introduces unique challenges. The coherent nature of radar signals generates granular speckle noise, which obscures fine details and complicates feature extraction [6]. Additionally, certain classes, such as the BTR-60 and BRDM-2, exhibit structural similarities that challenge model classification capabilities.

To prepare the data, images were resized to 128×128 pixels to standardize input dimensions and scaled to the $[0, 1]$ range by normalizing pixel intensities. This preprocessing step improves numerical stability and ensures consistency across models. The dataset was split into training (80%) and validation (20%), maintaining a stratified distribution of classes. These preprocessing measures ensured computational efficiency and reliable evaluation.

3.2 Experimental Setup

Experiments were conducted on a Jupyter Notebook hosted on Google Colab using NVIDIA A100 GPUs. The codebase for all experiments, including data preprocessing, model architectures, training, and evaluation, is available on GitHub: `https://github.com/asherellis/amth-491/blob/main/thesis_final_12_04_2024.ipynb`. Random seeds were used to ensure reproducibility, and tools such as Matplotlib were used to generate visualizations including accuracy curves and confusion matrices.

3.3 Model Architectures

The five machine learning models evaluated in this study includes three custom convolutional neural networks (CNNs) and two pre-trained architectures. These models were chosen to balance computational efficiency, simplicity, and feature extraction capabilities for SAR imagery. Table 1 summarizes the architectures and their key characteristics.

Table 1: Summary of Model Architectures

Model Name	**Type**	**Trainable Parameters**	**Key Features**
Baseline CNN	Custom CNN	591K	Three conv blocks, batch normalization, dropout
Deeper CNN	Custom CNN	1.2M	Deeper architecture, increased filters, dropout
Efficient CNN	Custom CNN	96K	Lightweight architecture, aggressive pooling
MobileNetV2	Pre-trained	165K (transfer)	Transfer learning with ImageNet weights
ResNet50	Pre-trained	526K (transfer)	Deep residual network with fine-tuning

3.3.1 Baseline CNN

The Baseline CNN consists of three convolutional blocks designed to capture essential features from SAR imagery:

- **First Block**: Two Conv2D layers with 64 filters each, using ReLU activation and batch normalization. This block is followed by a MaxPooling2D layer to reduce spatial dimensions.
- **Second Block**: Two Conv2D layers with 128 filters each, using ReLU activation and batch normalization. A MaxPooling2D layer follows, and a Dropout layer with a rate of 0.3 is applied to prevent overfitting.
- **Third Block**: A single Conv2D layer with 256 filters and ReLU activation, followed by batch normalization and a GlobalAveragePooling2D layer, which replaces the need for additional pooling layers.

The classification head comprises a Dense layer with 128 units and ReLU activation, followed by a Dropout layer with a rate of 0.4, and a final Dense layer with softmax activation for the 8 output classes. The model is trained with the Adam optimizer using a learning rate of 1×10^{-4}, reflecting its simpler architecture and stability during training.

3.3.2 Deeper CNN

Building on the Baseline CNN, the Deeper CNN introduces additional convolutional layers to enhance feature extraction:

- **First Block**: Two Conv2D layers with 64 filters each and ReLU activation, followed by a MaxPooling2D layer.
- **Second Block**: Two Conv2D layers with 128 filters each and ReLU activation, followed by a MaxPooling2D layer and a Dropout layer with a rate of 0.3 to prevent overfitting.
- **Third Block**: Two Conv2D layers with 256 filters each and ReLU activation, followed by a MaxPooling2D layer and a Dropout layer with a rate of 0.4.

The classification head includes a GlobalAveragePooling2D layer, a Dense layer with 256 units and ReLU activation, followed by a Dropout layer with a rate of 0.5, and a final Dense layer with softmax activation for the 8 output classes. The model is trained with the Adam optimizer and a learning rate of 1×10^{-3}, balancing complexity and overfitting.

3.3.3 Efficient CNN

The Efficient CNN emphasizes computational efficiency and includes:

- **First Layer**: A Conv2D layer with 32 filters and a kernel size of 5, using ReLU activation and padding set to 'same', followed by a MaxPooling2D layer with a pool size of 4.
- **Second Layer**: A Conv2D layer with 64 filters and a kernel size of 3, using ReLU activation, followed by a MaxPooling2D layer with a pool size of 4.
- **Third Layer**: A Conv2D layer with 128 filters and a kernel size of 3, using ReLU activation, followed by a MaxPooling2D layer with a pool size of 2.

The classification head consists of a GlobalAveragePooling2D layer, a Dropout layer with a rate of 0.4, and a Dense layer with softmax activation for the 8 output classes. This architecture achieves competitive performance with a reduced parameter count. The model is trained with the Adam optimizer using a learning rate of 1×10^{-3}.

3.3.4 Pre-trained Architectures

The MobileNetV2 and ResNet50 architectures leverage ImageNet weights for initialization and are fine-tuned to adapt pre-trained features to SAR imagery through a two-phase training process. MobileNetV2, introduced by Sandler et al. (2018) [7], is designed for efficient performance with low computational overhead, making it suitable for resource-constrained environments. ResNet50, developed by He et al. (2015) [3], employs a deep residual learning framework to facilitate the training of very deep networks while maintaining high accuracy on complex tasks. Both architectures leverage ImageNet weights for initialization and are fine-tuned to adapt pre-trained features to SAR imagery through a two-phase training process:

Phase 1: Transfer Learning

Only the custom top layers are trained, which include:

- A GlobalAveragePooling2D layer.
- A Dense layer with 128 units (for MobileNetV2) or 256 units (for ResNet50) and ReLU activation.
- A Dropout layer with a rate of 0.5.
- A Dense layer with softmax activation for the 8 output classes.

The base model layers remain frozen during this phase. The Adam optimizer with a learning rate of 1×10^{-3} is used.

Phase 2: Fine-Tuning

Approximately 20% of the base model's layers are unfrozen to adapt to SAR-specific features, while the remaining 80% remain frozen to retain the general features learned from ImageNet. Fine-tuning is performed with the Adam optimizer using a reduced learning rate of 1×10^{-5} for stability.

3.4 Training and Optimization

All models were trained using the Adam optimizer with a batch size of 32. The training protocol included measures to enhance model performance and prevent overfitting. One such measure was early stopping, whereby training was halted if validation accuracy did not improve for five consecutive epochs, and the best weights were restored. Additionally, learning rate reduction was used, such that if validation accuracy plateaued for three epochs, the learning rate was halved, with a minimum value of 1×10^{-6}. Custom CNNs were trained for up to 15 epochs, while pre-trained models underwent five epochs of transfer learning

followed by five epochs of fine-tuning. Employing these strategies achieved robust evaluation and model optimization for SAR imagery.

3.5 Adversarial Attacks

Adversarial examples are carefully crafted inputs designed to deceive machine learning models. These perturbations, often imperceptible to humans, exploit vulnerabilities in neural networks, causing them to make incorrect predictions with high confidence. This study evaluates model robustness using two widely adopted adversarial attack techniques: the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). These attacks leverage the high-dimensional feature spaces of neural networks to create small but intentional perturbations, revealing model fragility in high-stakes applications like SAR image classification.

3.5.1 Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method (FGSM), introduced by Goodfellow et al., generates adversarial examples by perturbing the input data in the direction of the gradient of the loss function [2]. The perturbed input x' is calculated as:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f_\theta(x), y)),$$

where:

- x is the original input,
- ϵ controls the magnitude of the perturbation,
- $\nabla_x \mathcal{L}$ is the gradient of the loss function with respect to the input x,
- $f_\theta(x)$ is the model's output,
- y is the true label.

The gradient's sign is used rather than the full gradient vector because it provides a computationally efficient way to maximize the loss function in the direction that increases prediction error. This ensures that the perturbation is minimal while still being highly effective at deceiving the model.

In this study, FGSM was implemented with ϵ values of 0.01, 0.05, and 0.1 to evaluate model performance under varying perturbation strengths. The implementation uses `tf.clip_by_value` to ensure pixel intensities remain within the valid range $[0, 1]$.

3.5.2 Projected Gradient Descent (PGD)

Projected Gradient Descent (PGD) extends FGSM by applying iterative updates to refine adversarial perturbations, creating stronger adversarial examples [5]. The iterative update rule is:

$$x^{(k+1)} = \text{Proj}_{\mathcal{B}_\epsilon}(x^{(k)} + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(f_\theta(x^{(k)}), y))),$$

where:

- $x^{(k)}$ is the perturbed input at iteration k,
- α is the step size,
- $\text{Proj}_{\mathcal{B}_\epsilon}$ projects the perturbed input back into the ϵ-ball around the original input, ensuring the perturbation remains bounded,
- $\mathcal{B}_\epsilon$ represents the allowable magnitude of perturbation.

The projection operator $\text{Proj}_{\mathcal{B}_\epsilon}(x')$ ensures that the perturbed input remains within the ϵ-ball around the original input x. It can be mathematically expressed as:

$$\text{Proj}_{\mathcal{B}_\epsilon}(x') = \min\left(\max(x, x' - \epsilon), x + \epsilon\right),$$

where x represents the original input, and ϵ controls the allowable magnitude of perturbation. This constraint ensures that adversarial examples remain realistic and imperceptible.

PGD incorporates random initialization within the ϵ-ball to enhance attack diversity and prevent reliance on a fixed starting point. Additionally, PGD refines perturbations over multiple iterations, with the step size α dictating the granularity of updates. The choice of α and ϵ is critical: a large α may overshoot optimal perturbations, while a small α might require a higher number of iterations to converge. This study used $\epsilon = 0.01, 0.05, 0.1$, step sizes of $\alpha = 0.002, 0.01, 0.02$, and 7 iterations to systematically evaluate model robustness under increasingly powerful attacks.

Both FGSM and PGD operate within an L_∞-bounded space to ensure that adversarial perturbations are imperceptible to humans. The L_∞-norm constraint ensures that the maximum perturbation magnitude for any pixel remains within ϵ. Mathematically:

$$\|x' - x\|_\infty \leq \epsilon,$$

where $\|\cdot\|_\infty$ represents the maximum absolute value across all pixel differences. This constraint is critical for maintaining the visual fidelity of adversarial examples.

3.5.3 Evaluation Metrics and Analysis

To systematically assess the impact of adversarial attacks, normalization techniques were applied to ensure numerical stability. In additional to accuracy, several performance degradation metrics were collected to provide a broader view of model vulnerabilities under adversarial perturbations. Specific metrics used included performance variability, involving comparative analysis of model performance between clean and perturbed inputs, and prediction confidence, calculated as the average maximum softmax probability of the predicted class for adversarial inputs, reflecting the model's certainty. Visualizations, including comparative bar charts and summary tables, are presented in the results section to highlight trade-offs in performance and robustness across different architectures and the strengths and vulnerabilities of the evaluated models under adversarial conditions.

3.6 Defensive Strategies

To enhance model robustness, adversarial training was employed by incorporating adversarial examples into the training process [5]. This approach solves a min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\|\eta\|_\infty \leq \epsilon} \mathcal{L}(f_\theta(x+\eta), y) \right].$$

The inner maximization generates adversarial examples $x + \eta$ by maximizing the loss function within an ϵ-bounded perturbation, while the outer minimization updates the model parameters θ to minimize the loss over both clean and adversarial inputs. In this study, adversarial training was conducted using two attack methods: Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). FGSM-based adversarial training introduced a single-step perturbation scaled by $\epsilon = 0.01$ or $\epsilon = 0.02$, creating adversarial examples for each batch during training. In contrast, PGD-based adversarial training used iterative updates to refine adversarial perturbations, with $\epsilon = 0.01$ or $\epsilon = 0.02$, step sizes of $\alpha = 0.002$ or $\alpha = 0.005$, and 10 iterations. Random initialization within the ϵ-ball around the input was incorporated into PGD to enhance attack diversity, and after each step, perturbed inputs were projected back into the valid range $[0, 1]$ to ensure valid pixel intensities.

During adversarial training, the combined loss function integrated both clean and adversarial inputs, formulated as:

$$\mathcal{L}_{\text{combined}} = 0.5 \cdot \mathcal{L}(f_\theta(x), y) + 0.5 \cdot \mathcal{L}(f_\theta(x+\eta), y),$$

where x represents clean inputs and $x + \eta$ represents adversarial examples. This loss function was optimized to preserve the model's accuracy on clean data while improving its robustness against adversarial attacks. To facilitate stable training, adversarial examples

were clipped at each step to satisfy the constraint $\|x' - x\|_\infty \leq \epsilon$, ensuring that perturbations remained within the allowable bounds.

Adversarial training was conducted for 15 epochs, and metrics were recorded at the end of each epoch for both clean and adversarial performance on the training and validation datasets. The recorded metrics included training loss, clean accuracy, adversarial accuracy, and the equivalent validation metrics.

4 Results

4.1 Clean Model Training Results

4.1.1 Overview and Training Dynamics

The clean models were trained to establish baseline performance on unperturbed data, providing insights into their generalization capabilities. Among the models, Efficient CNN demonstrated strong performance with a validation accuracy of 97.41% and smooth convergence, ending with a low final validation loss of 0.0709. Deeper CNN slightly outperformed Efficient CNN in terms of best validation accuracy (97.68%) and achieved a final validation accuracy of 96.83% with a validation loss of 0.0738. Early stopping at epoch 14 further prevented overfitting for this model. MobileNetV2 exhibited early overfitting, achieving a best validation accuracy of 96.25% at epoch 5, but its performance declined to a final validation accuracy of 90.07% with a high validation loss of 0.2896. ResNet50 displayed training instability and variability in loss trends, achieving a validation accuracy of 93.87% and a final validation loss of 0.2106. Baseline CNN, despite its simplicity, performed reliably with a validation accuracy of 96.30% and a validation loss of 0.0967. Table 2 summarizes the final performance metrics for all models, and Figure 2 illustrates the training and validation accuracy and loss trends.

Table 2: Final Performance Metrics for Clean Models

Model	**Train Acc.**	**Val Acc.**	**Best Val Acc.**	**Train Loss**	**Val Loss**	**Best Epoch**
Baseline CNN	96.30%	96.30%	96.30%	0.1081	0.0967	15
Deeper CNN	97.12%	96.83%	97.68%	0.0694	0.0738	14
Efficient CNN	95.92%	97.41%	97.41%	0.1103	0.0709	15
MobileNetV2	96.47%	90.07%	96.25%	0.1064	0.2896	5
ResNet50	91.44%	93.87%	93.87%	0.2703	0.2106	10

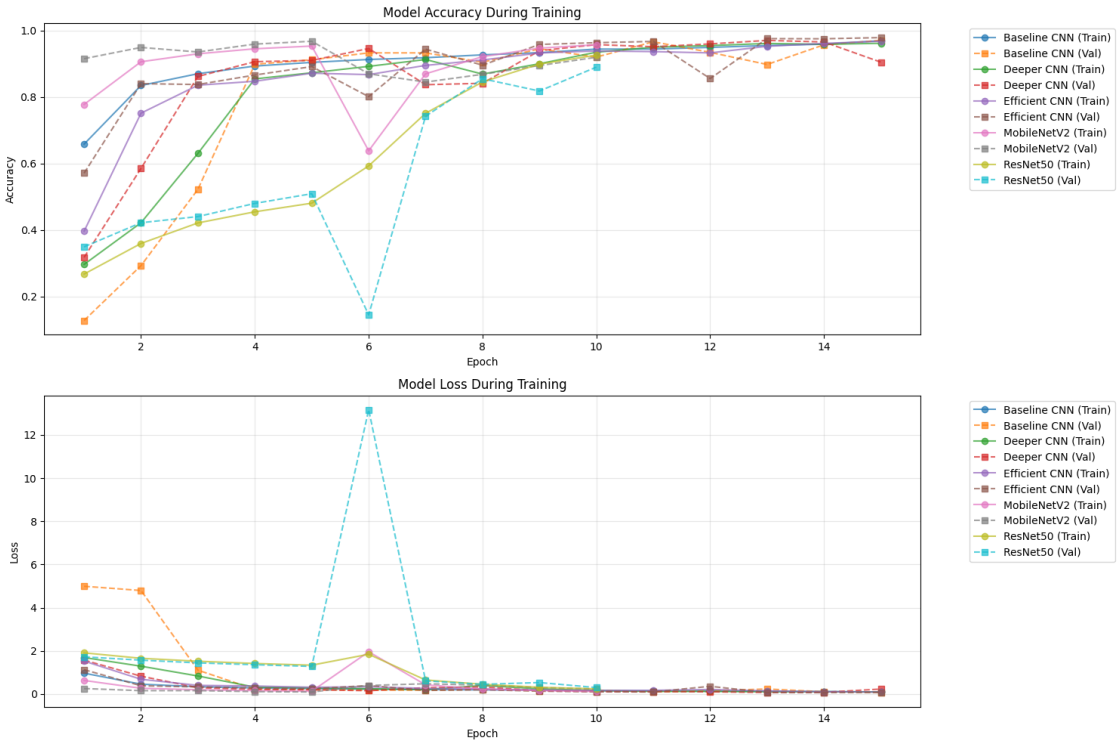

Figure 2: Training and Validation Accuracy and Loss for Clean Models.

4.1.2 Model Comparisons and Observations

Efficient CNN emerged as the top-performing model for validation accuracy, demonstrating robust generalization with a validation accuracy of 97.41% and smooth convergence, as reflected in its low validation loss of 0.0709. Deeper CNN closely followed, achieving a best validation accuracy of 97.68% and a final validation accuracy of 96.83%, making it highly competitive with Efficient CNN. MobileNetV2's early overfitting behavior, marked by a best validation accuracy of 96.25% at epoch 5, led to a decline in performance with a final validation accuracy of 90.07%. However, after adversarial training, MobileNetV2's clean accuracy improved significantly, indicating a regularizing effect of the adversarial examples. ResNet50, while achieving a validation accuracy of 93.87%, displayed higher variability in loss trends, likely due to its greater model complexity. Baseline CNN, though simpler than the other architectures, performed reliably, achieving a validation accuracy of 96.30%, but lacked the complexity required for higher performance. These results highlight the trade-offs between model architecture complexity and generalization performance, with Efficient CNN and Deeper CNN providing the best balance of accuracy and stability.

4.1.3 Macro-Average Metrics

To evaluate performance across classes, macro-average precision, recall, and F1-scores were computed (Table 3). Baseline CNN, Deeper CNN, and Efficient CNN achieved high F1-scores (0.96-0.98), highlighting their ability to balance precision and recall. MobileNetV2 had an F1-score of 0.86, reflecting challenges in consistently classifying all classes, potentially due to overfitting.

Table 3: Macro-Average Metrics for Clean Models

Model	**Precision**	**Recall**	**F1-Score**
Baseline CNN	0.96	0.96	0.96
Deeper CNN	0.97	0.98	0.98
Efficient CNN	0.98	0.97	0.97
MobileNetV2	0.90	0.86	0.86
ResNet50	0.94	0.93	0.93

4.2 Adversarial Robustness Evaluation

4.2.1 Overview

The robustness of the trained models was evaluated under adversarial attacks, specifically the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). Each attack was applied with varying perturbation magnitudes ($\epsilon = 0.01, 0.05, 0.1$) to assess the models' resilience. Metrics such as accuracy under attack and confidence of predictions were analyzed to understand the models' behavior in adversarial scenarios. Confidence was calculated as the average softmax probability of the predicted class.

4.2.2 Key Observations

The Baseline CNN demonstrated the highest adversarial accuracy across all perturbation levels for both FGSM and PGD attacks but showed overconfidence in incorrect predictions, particularly at higher ϵ values. Efficient CNN achieved strong clean accuracy but suffered significant degradation under adversarial attacks, with PGD accuracy dropping near zero for $\epsilon = 0.05$ and $\epsilon = 0.1$. Deeper CNN outperformed MobileNetV2 and ResNet50 in adversarial conditions but exhibited high variability, especially under PGD attacks at higher ϵ. MobileNetV2 and ResNet50 experienced substantial accuracy degradation for both FGSM and PGD, with MobileNetV2 failing almost entirely under PGD at $\epsilon = 0.05$ and $\epsilon = 0.1$.

4.2.3 Adversarial Attack Performance

The accuracy of models under FGSM attacks is depicted in Figure 3. Baseline CNN consistently outperformed other models across all perturbation levels (ϵ), demonstrating resilience against single-step attacks.

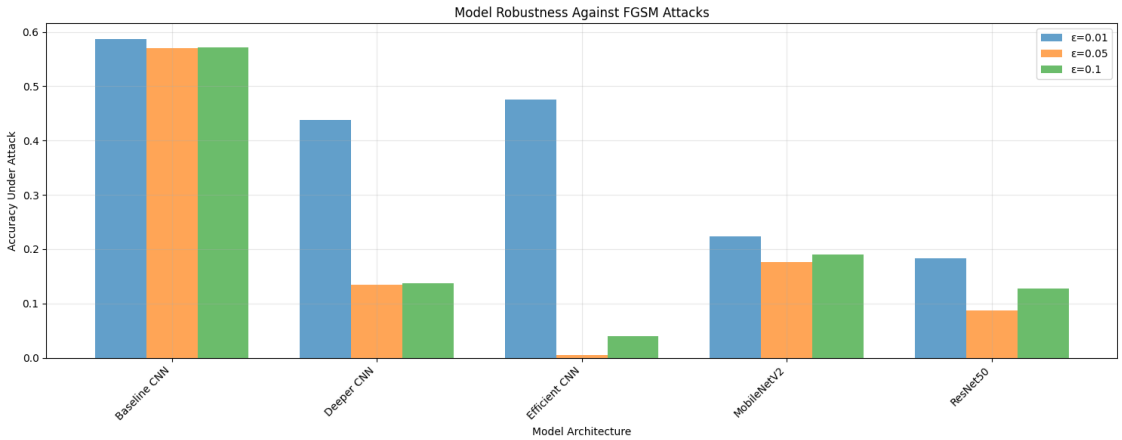

Figure 3: Accuracy of models under FGSM attacks with varying ϵ.

Figure 4 illustrates the confidence of models under FGSM attacks. Models maintained high confidence, even for incorrect predictions.

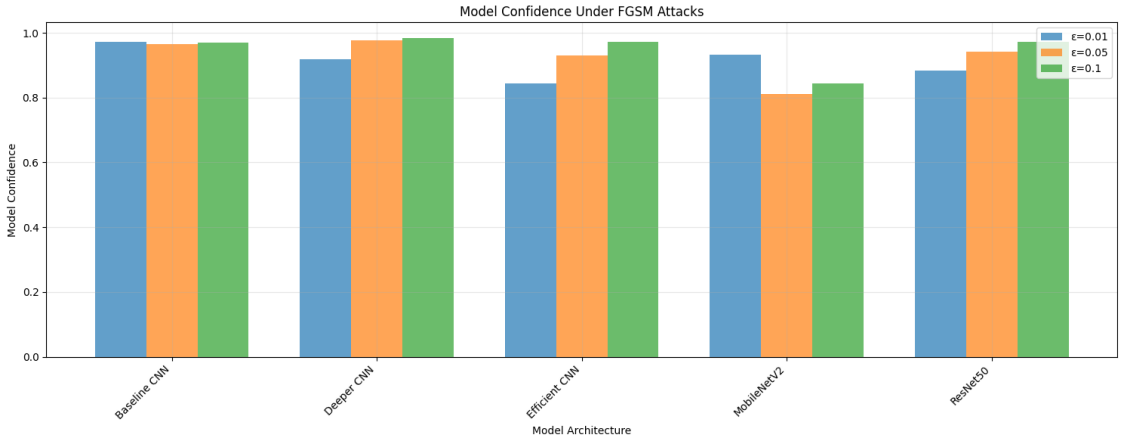

Figure 4: Model confidence under FGSM attacks with varying ϵ.

Table 4: Summary of FGSM Attack Results.

Model	**Acc** ($\epsilon = 0.01$)	**Conf** ($\epsilon = 0.01$)	**Acc** ($\epsilon = 0.05$)	**Conf** ($\epsilon = 0.05$)	**Acc** ($\epsilon = 0.1$)	**Conf** ($\epsilon = 0.1$)
Baseline CNN	62.7	95.7	59.3	94.7	58.7	98.4
Deeper CNN	45.8	92.5	10.1	95.2	14.4	96.3
Efficient CNN	46.5	83.3	1.6	90.3	5.7	98.2
MobileNetV2	18.6	92.9	16.8	81.5	19.9	83.0
ResNet50	18.9	85.7	16.1	92.3	16.2	96.3

Figure 5 shows model accuracy under PGD attacks. Baseline CNN exhibited the highest resilience, while Efficient CNN, MobileNetV2, and ResNet50 were highly susceptible to iterative attacks at $\epsilon = 0.05$ and $\epsilon = 0.1$.

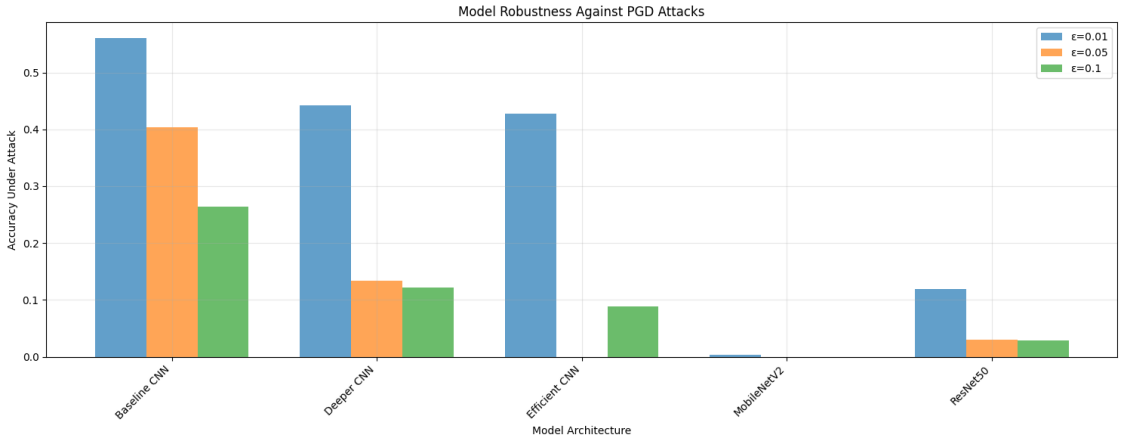

Figure 5: Accuracy of models under PGD attacks with varying ϵ. Iterative attacks significantly degrade performance, particularly for MobileNetV2 and Efficient CNN.

PGD Attack Results - Model Confidence As shown in Figure 6, all models maintained high confidence under PGD attacks, even for incorrect predictions. Baseline CNN and ResNet50 showed the highest confidence levels across all perturbation magnitudes.

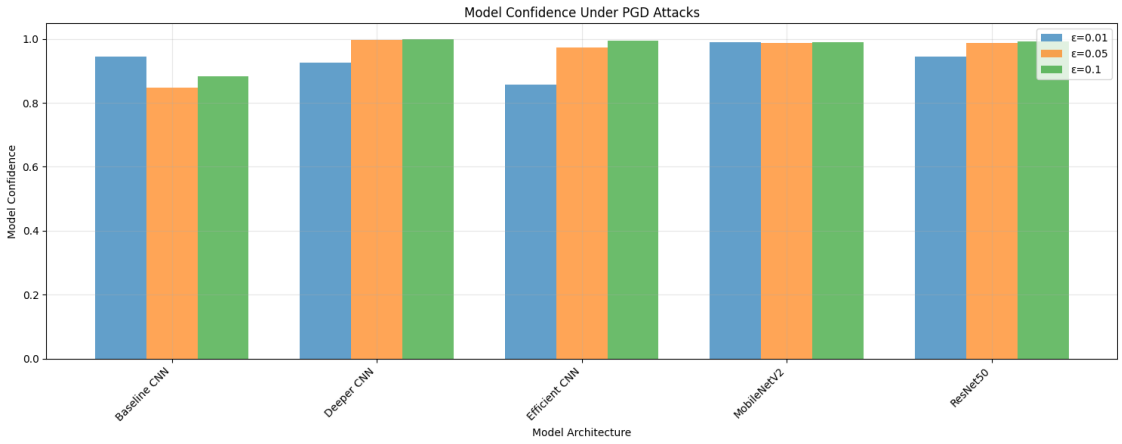

Figure 6: Model confidence under PGD attacks with varying ϵ.

Table 5: Summary of PGD Attack Results.

Model	**Acc** ($\epsilon = 0.01$)	**Conf** ($\epsilon = 0.01$)	**Acc** ($\epsilon = 0.05$)	**Conf** ($\epsilon = 0.05$)	**Acc** ($\epsilon = 0.1$)	**Conf** ($\epsilon = 0.1$)
Baseline CNN	59.0	95.6	37.0	84.3	24.9	92.6
Deeper CNN	41.9	94.1	2.7	99.6	0.6	99.8
Efficient CNN	44.1	85.2	0.0	96.9	11.6	99.8
MobileNetV2	0.1	98.6	0.0	98.4	0.0	99.2
ResNet50	14.9	92.6	1.8	98.5	1.2	99.1

The robustness evaluations revealed the vulnerability of clean-trained models in adversarial scenarios and highlighted the need for robust defenses. Baseline CNN consistently achieved the highest adversarial accuracy but exhibited overconfidence in incorrect predictions. Efficient CNN showed strong clean accuracy but was highly susceptible to iterative attacks at higher ϵ values. Deeper CNN demonstrated moderate robustness, outperforming MobileNetV2 and ResNet50 under adversarial conditions. MobileNetV2 and ResNet50 struggled significantly under PGD attacks, emphasizing their limitations in adversarial robustness.

4.3 Results for Adversarial Training

4.3.1 Training Parameters

The adversarial training experiments were conducted over 15 epochs using the Adam optimizer with a learning rate of 1×10^{-4}. The perturbation magnitude was set to $\epsilon = 0.01$. Two adversarial methods used for the initial attacks were employed during training: FGSM and PGD. Clean and adversarial accuracies were evaluated at each epoch to assess the trade-off between robustness and clean performance.

Note: Only Attempt 1 was conducted in this study. Attempts 2 and 3, which would have used higher ϵ values and more epochs, were planned but not executed due to time and compute resource constraints.

4.3.2 Training Metrics Across Models

Figure 7 illustrates the training and validation metrics across models during adversarial training. Deeper CNN and Efficient CNN showed improvement in both clean and adversarial accuracy over epochs. MobileNetV2 achieved high clean accuracy quickly but exhibited variability in adversarial accuracy, particularly under the PGD method.

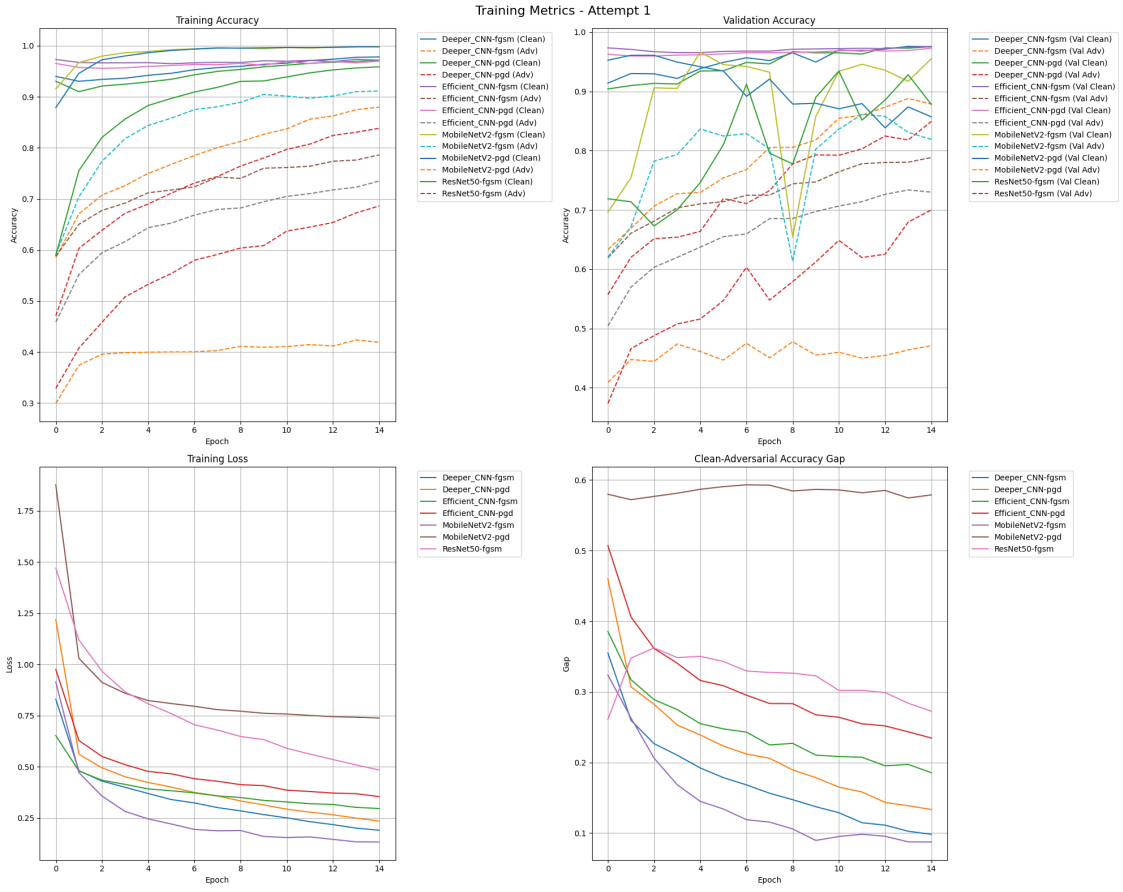

Figure 7: Training Metrics Across Models During Adversarial Training

The trends indicate that models trained with PGD-based adversarial examples generally required more epochs to stabilize compared to those trained with FGSM. This reflects the increased complexity and diversity of adversarial examples generated by iterative attacks, which can enhance robustness but also increase compute demands required for training.

Note: The Baseline CNN model was excluded from adversarial training due to persistent instabilities. NaN losses consistently appeared early in training, even with modifications to

learning rate and loss scaling, likely due to the model's simpler architecture which lacked the capacity to learn robust features under adversarial conditions.

4.3.3 Results Summary

Table 6 summarizes the results of adversarial training across different models and attack types.

Table 6: Performance Summary for Adversarial Training.

Model	Attack	Final Clean Acc	Final Adv Acc	Final Val Clean Acc	Final Val Adv Acc	Best Val Clean Acc	Final Loss
Deeper CNN	FGSM	97.82%	87.99%	97.55%	87.86%	97.60%	0.1904
Deeper CNN	PGD	97.17%	83.83%	97.55%	84.93%	97.55%	0.2346
Efficient CNN	FGSM	97.18%	78.62%	97.58%	78.84%	97.58%	0.2965
Efficient CNN	PGD	96.98%	73.51%	97.27%	73.01%	97.27%	0.3545
MobileNetV2	FGSM	99.89%	91.15%	95.52%	81.91%	96.61%	0.1330
MobileNetV2	PGD	99.79%	41.89%	85.71%	47.09%	96.09%	0.7381
ResNet50	FGSM	95.88%	68.61%	87.82%	70.01%	93.44%	0.4844

Figure 8 compares the clean accuracy, worst adversarial accuracy, and post-adversarial training accuracy for each model.

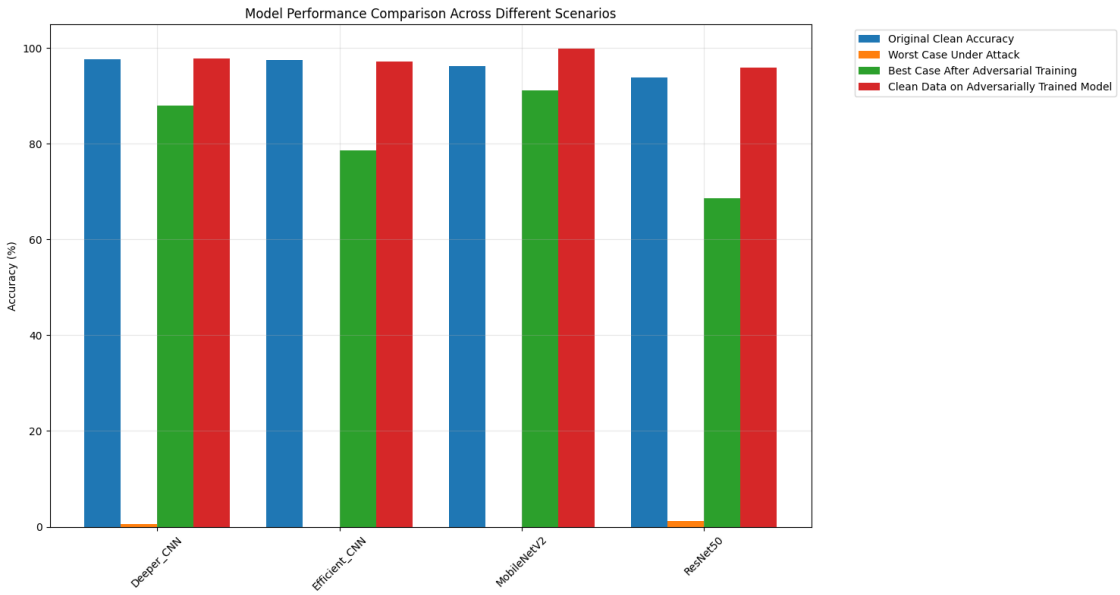

Figure 8: Performance Comparison Across Different Scenarios (Clean, Adversarial, and Post-Adversarial Training)

While MobileNetV2 achieved the highest clean accuracy after adversarial training, its adversarial robustness remained lower compared to Deeper CNN. Specifically, MobileNetV2 showed high adversarial accuracy under FGSM adversarial training (91.15%) but significantly lower adversarial accuracy under PGD adversarial training (41.89%).

4.3.4 Discussion on MobileNetV2's Performance

MobileNetV2's performance highlights an interesting phenomenon. Despite initial overfitting signs during clean training, adversarial training not only improved its robustness under FGSM attacks but also enhanced its clean accuracy from 90.07% to 99.89%. This suggests that adversarial training acted as a form of regularization, helping the model generalize better on clean data. However, the model's adversarial accuracy under PGD training remained low (41.89%), indicating that while it can adapt to single-step attacks, it struggles with iterative attacks. This underscores the importance of the type of adversarial training in developing robust models.

5 Discussion

5.1 Observations and Analysis

The results reveal distinct trade-offs between clean accuracy and adversarial robustness across models and attack types. MobileNetV2 achieved a high clean accuracy of 99.89% under FGSM adversarial training but exhibited significant vulnerability to iterative attacks. Before adversarial training, MobileNetV2's adversarial accuracy under PGD attacks ($\epsilon = 0.01$) was 0.1%. After adversarial training with PGD, it improved to 41.89%, indicating some resilience but still highlighting sensitivity to iterative attacks. This sensitivity underscores the model's reliance on the type of adversarial training employed.

Deeper CNN demonstrated a strong balance between clean and adversarial performance. Before adversarial training, it had a clean accuracy of 96.83% and an adversarial accuracy of 41.94% under PGD attacks ($\epsilon = 0.01$). After adversarial training with FGSM, its adversarial accuracy improved to 87.86%, with clean accuracy slightly increasing to 97.55%. This suggests that Deeper CNN is well-suited for applications requiring adversarial resilience without significant compromises in clean-data performance.

Efficient CNN showed strong clean accuracy (97.58%) and improved adversarial robustness after adversarial training. Before adversarial training, its adversarial accuracy under PGD attacks ($\epsilon = 0.01$) was 44.1%, which increased to 73.01% after PGD adversarial training. This indicates that Efficient CNN can be effectively enhanced for iterative attack resilience.

ResNet50 achieved moderate clean accuracy (93.87%) but struggled with adversarial robustness. Before adversarial training, its adversarial accuracy under PGD attacks ($\epsilon = 0.01$) was 14.9%, which improved to 70.01% after FGSM adversarial training. The resource-intensive nature of ResNet50 may limit its practical use in adversarial scenarios without further optimization.

5.2 Trade-offs Between Clean Accuracy and Adversarial Robustness

The trade-off between clean accuracy and adversarial robustness is a critical consideration in designing models for SAR image classification. Figure 9 illustrates this trade-off, showing that improvements in adversarial accuracy often come at the cost of clean accuracy.

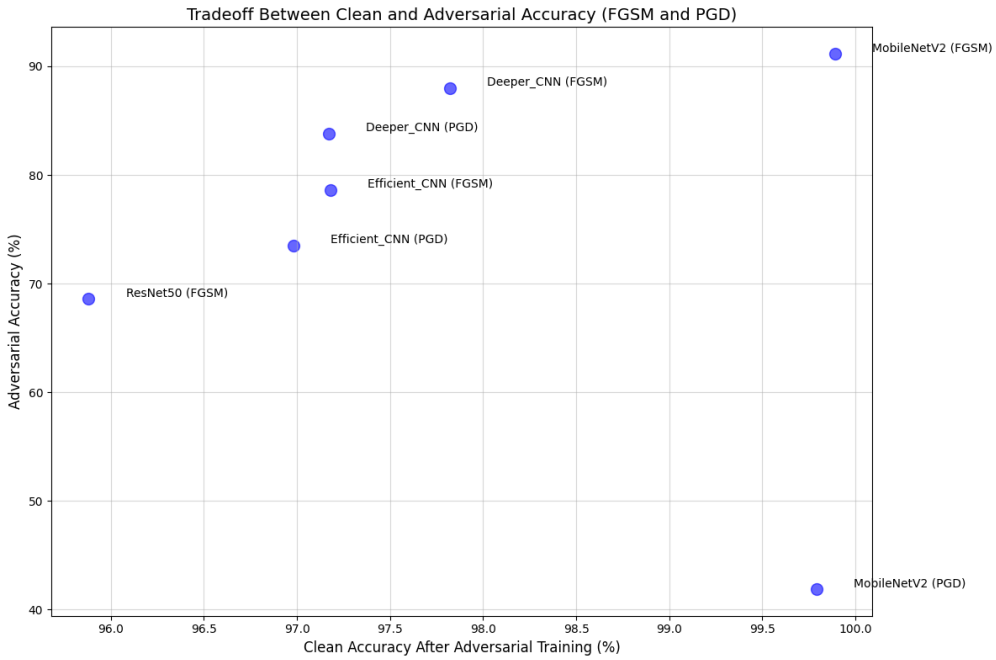

Figure 9: Trade-off between clean accuracy and adversarial accuracy for different models under FGSM and PGD adversarial training

It is evident that Deeper CNN maintains a strong balance between clean and adversarial accuracy under both FGSM and PGD training compared to MobileNetV2 and ResNet50. While MobileNetV2 achieves high clean accuracy after FGSM adversarial training, its adversarial accuracy under PGD attacks remains relatively low, highlighting the model's sensitivity to the type of attack and the limitations of single-step adversarial training.

6 Conclusions

This thesis evaluated the adversarial robustness of convolutional neural networks (CNNs) for synthetic aperture radar (SAR) image classification, using the MSTAR dataset as a benchmark. The results demonstrated that while models could achieve high accuracy on clean data, their performance degraded significantly under adversarial perturbations, especially iterative attacks like Projected Gradient Descent (PGD).

Among the evaluated models, Deeper CNN and Efficient CNN achieved a favorable balance between clean accuracy and adversarial robustness, particularly after undergoing adversarial training. Iterative adversarial training with methods like PGD substantially enhanced model resilience, often with minimal compromise to clean-data performance. By contrast, certain architectures and training routines showed sensitivity to the choice of defense strategies, highlighting the model- and parameter-dependent nature of training for robustness.

These findings reveal the inherent trade-offs between clean accuracy and adversarial robustness, emphasizing the importance of careful model architecture selection and the integration of adversarial training techniques to ensure robustness. For example, MobileNetV2, while benefiting significantly from adversarial training in terms of clean accuracy (increasing from 90.07% to 99.89%), struggled under iterative attacks, with adversarial accuracy under PGD training improving only marginally. This highlights the importance of matching training strategies to model-specific vulnerabilities and attack types.

However, the study also revealed the increased computational demands inherent in adversarial training, especially for iterative approaches like PGD. For example, PGD-based training required more epochs to stabilize and introduced higher computational overhead compared to FGSM-based training. There is also potential for overfitting to specific attack types and evolving adversarial tactics in response to adversarial training implementation, which could further complicate the development of robust systems.

Introducing regularization techniques (e.g., dropout, weight decay) and data augmentation can help prevent overfitting and improve generalization. For instance, adversarial training acted as a regularizer for MobileNetV2, enabling it to generalize better to clean data. Evaluating models under diverse adversarial scenarios is also critical for ensuring that defenses hold across a spectrum of attack strategies.

7 Future Work

Building on these findings, there are several avenues for further improving adversarial robustness in SAR image classification. Future work could explore more sophisticated defense mechanisms, such as randomized smoothing [1], feature denoising [9], or leveraging separate models to detect and exclude compromised samples, rather than relying solely on model robustness. These methods may help mitigate overfitting to specific attack types, as seen with MobileNetV2 under PGD adversarial training.

Developing adaptive training strategies that integrate a broad range of adversarial attacks, including adaptive and transferable attacks, could ensure that models generalize their defenses rather than overfitting to a single perturbation method [8]. This is especially critical given the results showing the vulnerability of models to PGD attacks even after adversarial training.

Incorporating SAR-specific information, such as noise characteristics and geometric scattering mechanisms, may also yield architectures and defenses more naturally resistant to common perturbations [8]. For instance, azimuthal variability and the impact of object orientation on radar reflections present opportunities for defense mechanisms tailored to SAR imagery. Leveraging domain-specific information could improve model robustness while reducing the need for computationally expensive iterative defenses. Finally, it would be valuable to evaluate adversarially trained models in operational environments to help verify their real-world applicability in dynamic settings.

References

[1] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, Feb 2019. [Online]. Available: `https://arxiv.org/abs/1902.02918`.

[2] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, March 2015. [Online]. Available: `https://arxiv.org/abs/1412.6572`.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, page 494 KB, Dec 2015. [Online]. Available: `https://arxiv.org/abs/1512.03385`.

[4] Air Force Research Laboratory. MSTAR Public Dataset. SDMS MSTAR Dataset Overview, Sandia National Laboratory, September 1995. [Online]. Available: `https://www.sdms.afrl.af.mil/index.php?collection=mstar&page=targets`.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, June 2017. [Online]. Available: `https://arxiv.org/abs/1706.06083`.

[6] Franz Meyer. Spaceborne synthetic aperture radar – principles, data access, and basic processing techniques. In A. Flores, K. Herndon, R. Thapa, and E. Cherrington, editors, *SAR Handbook: Comprehensive Methodologies for Forest Monitoring and Biomass Estimation*. NASA, 2019. [Online]. Available: `https://gis1.servirglobal.net/TrainingMaterials/SAR/Chp2Content.pdf`.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *arXiv preprint arXiv:1801.04381*, pages 4510–4520, Jan 2018. [Online]. Available: `https://arxiv.org/abs/1801.04381`.

[8] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, Feb 2020. [Online]. Available: `https://arxiv.org/abs/2002.08347`.

[9] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. *arXiv preprint arXiv:1812.03411*, Dec 2018. [Online]. Available: `https://arxiv.org/abs/1812.03411`.