Exploring Adversarial Robustness in Deep Learning through Shift Invariance

August 14, 2023

1 Introduction

1.1 Shift Invariance Can Reduce Adversarial Robustness

Shift invariance, a pivotal property in Convolutional Neural Networks (CNNs), plays a crucial role in enhancing classification performance. However, an intriguing insight revealed by the paper *Shift Invariance Can Reduce Adversarial Robustness* [1] lies in the observation that while invariance to circular shifts can elevate classification efficacy, it may simultaneously render models more susceptible to adversarial attacks. This report serves as an extension report for the practical course *Analysis of new phenomena in machine/deep learning*, building upon the groundwork laid during the reproduction phase. This report examines the intricate interplay between shift consistency scores, and adversarial robustness. For a comprehensive view of the code, data, and additional plots, refer to the associated Jupyter notebooks and Python scripts in the Github/Gitlab repository.

1.2 Potential Extensions

During the reproduction phase, we reproduced the main plots, made some small adjustments, and experimented with additional variations for verification and a deeper understanding. Building upon these changes, our goal in the extension phase is to delve further into the exploration of adversarial robustness.

We had three possible extensions to explore. **Transferability of Attacks** where we saw that the plots indicated that models with similar architectures tend to exhibit similar levels of robustness when subjected to adversarial data produced by similar model structures. This idea of transferability can be used as a middle ground between between white-box and black-box attacks.

Another avenue we discussed was the investigation of other model architectures such as **RNNs and Transformers**. However, given the existing literature on RNNs and Transformers in the context of adversarial robustness, we opted to focus on the most promising extension: **Exploring Robustness Parameters**. This choice is driven by our belief that understanding the impacts of various hyperparameters on robustness and shift invariance will provide valuable insights to get better models. By examining the deviations from the expected order of robustness, we aim to gain a deeper understanding of the intricate interplay between architecture, shift invariance and robustness.

In the subsequent sections of this report, we present our methodology, experimental results, and analysis, shedding light on the extension phase of our study.

2 Exploring Robustness Parameters

From the reproduction phase, we gained valuable insights into the robustness of various deep learning models on both the MNIST and Fashion MNIST datasets.

MNIST dataset: The paper's hypothesis of higher consistency scores indicating lower invariance held true in most cases, specifically the models used in the paper. The order from most to least robust consisted of models like *simple_FC_*1024, *simple_FC_*256, *simple_Conv_*10_512, *simple_Conv_*2, *simple_Conv_*12_2048, *simple_Conv_NL*, and *simple_Conv_max*. However, the actual order of robustness, especially for $norm = \infty$ and $\epsilon = 2$, deviates slightly from the hypothetical order. The observed order, from the most to least robust, is: *simple_Conv_2*, *simple_FC_1024*, *simple_FC_256*, *simple_Conv_max*, *simple_Conv_10_512*, *simple_Conv_12_2048*, and *simple_Conv_NL*. Nonetheless, the models used in the paper follow the hypothetical order, Figure: 1.

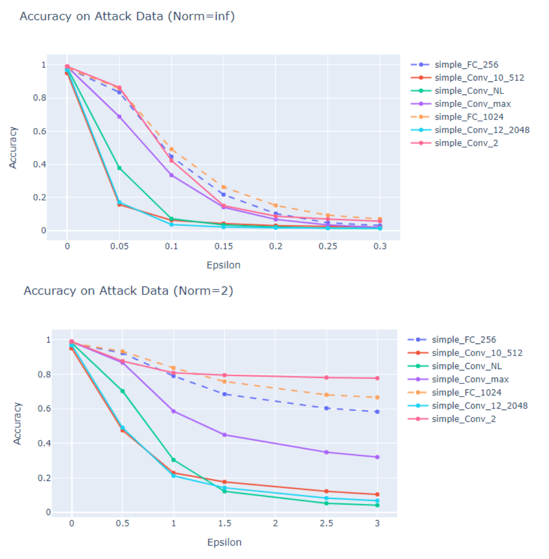


Figure 1: Accuracy on attack data for MNIST dataset

- Interestingly, the models *simple_Conv_max* and *simple_Conv_2* exhibit some unexpected results, which deviate from the expected robustness order.

Similar trends emerged in the **Fashion MNIST dataset**, reinforcing the relationship between consistency scores and invariance. The hypothetical order of robustness aligned closely with models like *simple_FC_2*, *simple_Conv_2*, *simple_FC_2_256*, *simple_FC_3_512*, *simple__RNN*, *simple_Conv_max*, *simple_Conv_10_512*, *simple_Conv_12_2048*, and *simple_Conv_NL*. The actual order of robustness, particularly for $norm = \infty$ and $\epsilon = 3$, does not perfectly align with the hypothetical order (while the models used in the paper follow the hypothetical order). The observed order, from the most robust to the least robust, is: *simple_Conv_2*, *simple_FC_2*, *simple_FC_2_256*, *simple_FC_3_512*, *simple_Conv_12_2048*, *simple__RNN*, *simple_Conv_10_512*, *simple_Conv_NL*, and *simple_Conv_max*. Figure: 2.

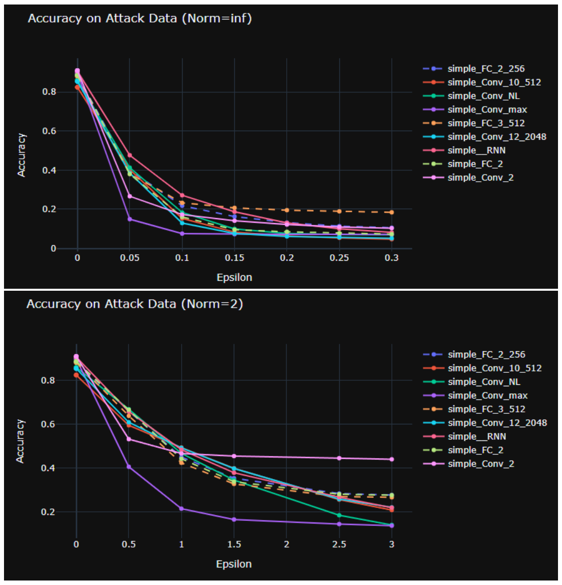


Figure 2: Accuracy on attack data for Fashion MNIST dataset

- Overall, the observed results demonstrate some deviations from the expected order of robustness, and specific models, such as *simple_Conv_max* and*simple_Conv_2*, display unexpected behaviors in both datasets. Nonetheless, the majority of the models follow the anticipated trend, indicating that consistency scores are generally inversely related to adversarial robustness, as initially hypothesized.

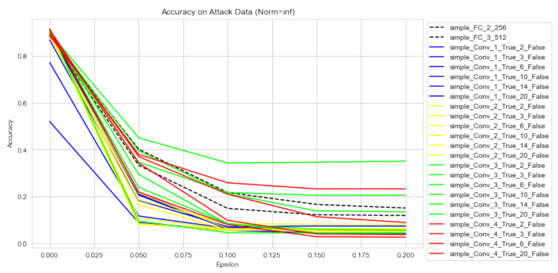
2.1 Experimental setting

Our extension focuses on exploring robustness parameters, aiming to discern the influence of hyperparameters on model robustness. To this end, we employed a similar experimental setting as the paper. Employing the MNIST dataset and later Fashion MNIST, we varied pooling types and the number of layers, each of which had emerged as key hyperparameters during our reproduction phase. Moreover, we compared two kernel sizes 10 (the one used in the paper) and 3 (the commonly used). We also compared the impact of adding a dense layer before the output layer. The models are named *simple_Conv_X_A_Y_B* , X:n_layers, Y:kernel_size, A: (True for max_pooling False for average_pooling) and add_dense: (True for dense layer added, False for not added).

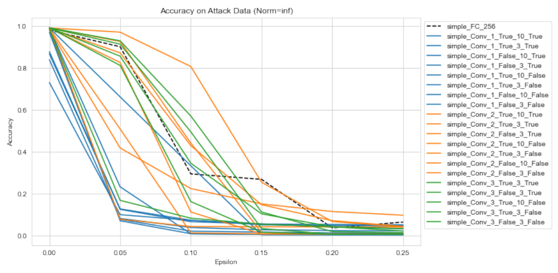
Experimental settings: All the models were trained for 20 epochs using ADAM optimizer, a batch size of 200, and a learning rate of 0.01 decreased by a factor of 10 at the 10^{th} and 15^{th} epochs. Robustness evaluation involved the PGD l_2 and l_∞ attacks with varied epsilon values, and instead of limiting the iterations to 10 we use a larger iteration limit of 100. By applying ART attacks, we created adversarial data and subsequently evaluated each model's shift consistency. The resulting robust accuracies and shift consistencies provide a dataset for the analysis of the impact of individual parameters on model robustness and shift invariance.

2.2 Results

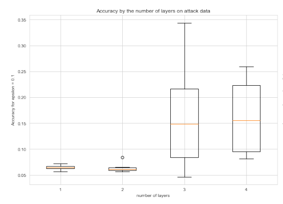
The goal is to identify models that exhibit higher adversarial robustness at least compared to the baseline fully connected (FC) model, while also have a good shift consistency score. The following general trends and findings emerged from the extension experiments:



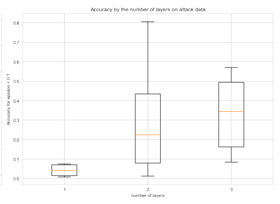
(a) Accuracy on attack data for Fashion MNIST



(b) Accuracy on attack data for MNIST



(c) Accuracy on attack data for Fashion MNIST dataset (norm ∞,$\epsilon = 0.1$) by the number of layers



(d) Accuracy on attack data for MNIST dataset (norm 2,$\epsilon = 0.1$) by the number of layers

Figure 3: Robust accuracy by the number of layers, for MNIST and Fashion MNIST

- **Impact of Model Depth**: Increasing the number of layers generally led to greater adversarial robustness for both the MNIST and Fashion MNIST datasets. Models with more layers exhibited improved performance against adversarial attacks, suggesting that deeper architectures contribute to enhanced robustness. However, this trend was not necessarily synonymous with reduced shift invariance. While single-layer models demonstrated higher shift invariance, models with multiple layers retained acceptable consistency levels without significantly compromising robustness (adding more layers doesn't mean necessarily being less shift invariant). 3

- **Pooling layers**: Max pooling enhanced robustness for the single hidden layer model, whereas average pooling showed better performance for models with multiple layers. This phenomenon was explored further in [2], we will include it in a later report. And also, pooling strategy did not significantly impact shift consistency. Intuitively, adversarial attacks involve adding small perturbations to input images to mislead the model's predictions. Average pooling gives importance to both significant and less significant features. While max pooling by retaining only the most significant features, can discard some of the less significant features, reducing the impact of noise. This might explain why max pooling is better at least for the one layer case. 4

- **Kernel Size**: Models with kernel size 3

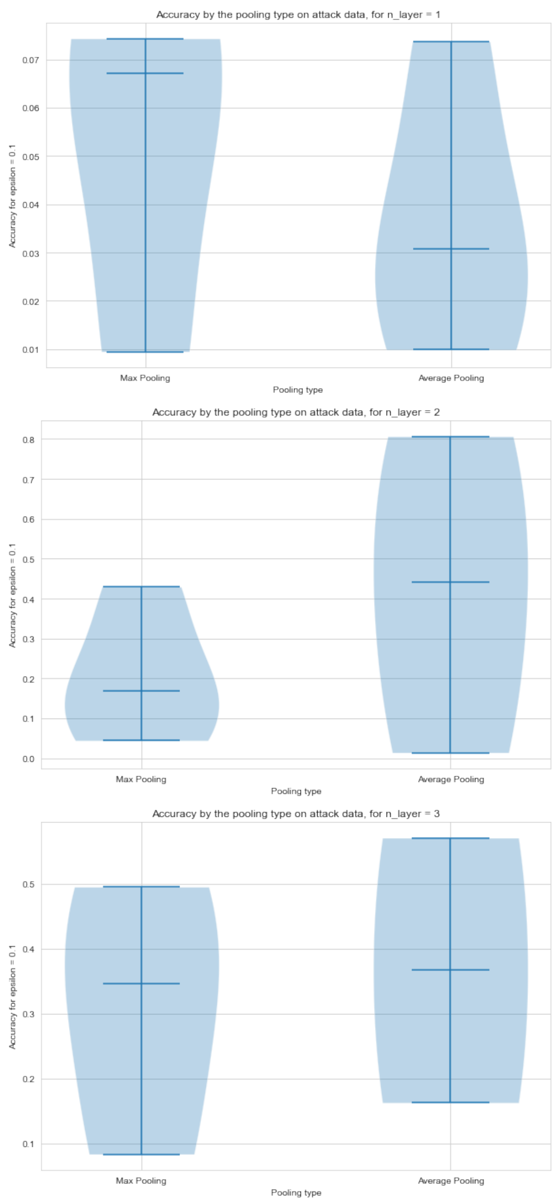
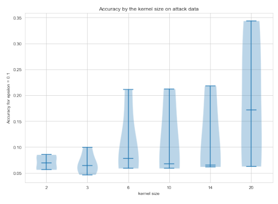
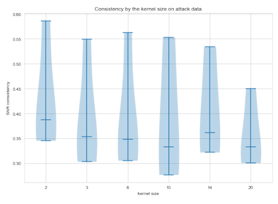


Figure 4: Accuracy on attack data for MNIST dataset by Pooling type, by the number of layers

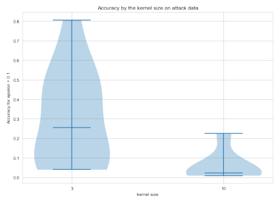
generally exhibited greater robustness and lower shift invariance than those with kernel size 10 (this might explain why the paper used kernel size 10). This finding suggests that smaller kernel sizes contribute to better robustness. However, this trend was not consistently observed across all experiments. Further exploration with various kernel sizes on the Fashion MNIST dataset (testing on 6 kernel sizes) revealed that while smaller kernel sizes generally correlated with higher shift invariance, the relationship between kernel size and robustness is more nuanced. For example on average having a smaller kernel size leads to more shift invariance, but the average of shift consistency for 14 is higher than 10. Also On average having a bigger kernel size leads to better robustness but the models of the kernel size 6 are slightly more robust on average than 10. For very big kernel sizes (relative to the input size) , for example 20 in our case, the model structure becomes closer to a fully connected model so it makes sense that it behaves more like a Fully connected. We can say that there is a general trend but it is not always true. 5 6



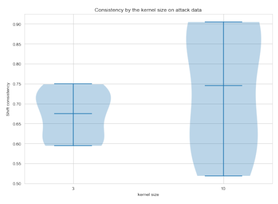
(a) Accuracy on attack data for Fashion MNIST by kernel size



(b) Shift consistency for Fashion MNIST by kernel size



(c) Accuracy on attack data for MNIST by kernel size



(d) Shift consistency for MNIST by kernel size

Figure 5: Adversarial robustness, Shift consistency and kernel size

- **Additional Dense Layer**: Adding an extra dense layer often resulted in improved

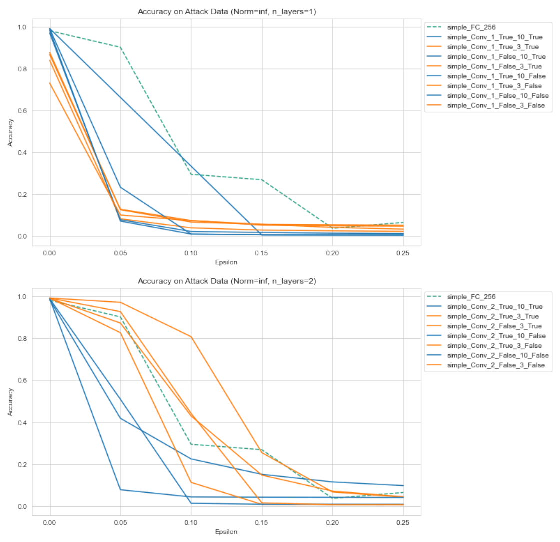


Figure 6: Accuracy on attack data for MNIST dataset by kernel size, by the number of layers

robustness without significantly impacting shift invariance. But we should note that the size of the added dense layer is significant; larger dense layers tend to make the model behave more like a fully connected model. 7

- **Noteworthy Models**: Several models exhibited robustness levels comparable to or even surpassing FC models. Notable examples for the MNIST data set include *simple_Conv_2_True_3_True*, *simple_Conv_2_False_3_True*, *simple_Conv_3_False_3_True*, and *simple_Conv_3_True_3_False*. 8

While individual hyperparameters did contribute to adversarial robustness, the overall robustness of a model appears to be influenced by a combination of parameters. To systematically determine the best set of parameters for achieving both high adversarial robustness and acceptable shift invariance, a more comprehensive approach is needed. This process is detailed in the following section.

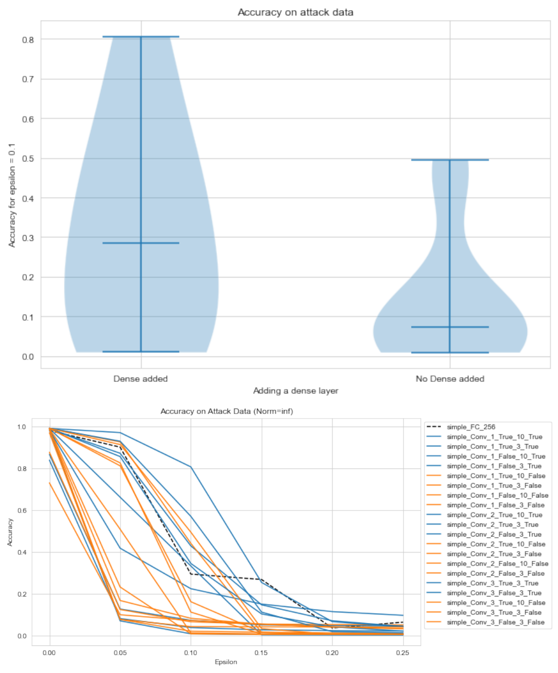


Figure 7: Impact of adding a dense layer on robust accuracy for MNIST dataset

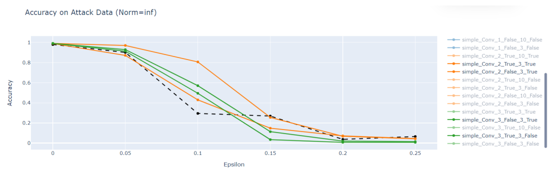
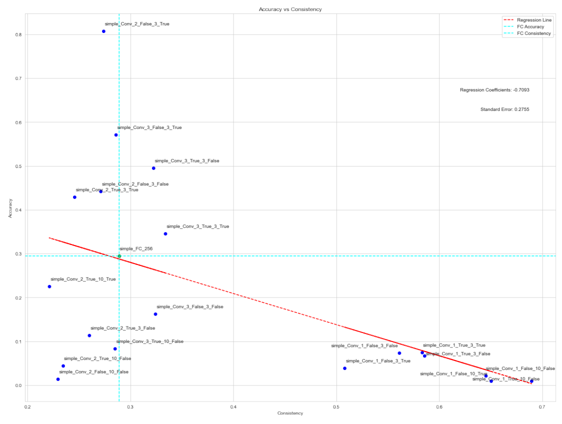


Figure 8: Accuracy on attack data for MNIST dataset, for some models, norm: ∞

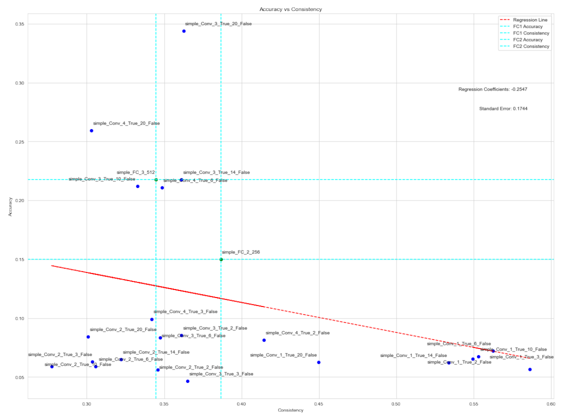
2.2.1 Graph Shift-Invariance-Adversarial-Robustness Analysis

To identify models that strike a balance between robustness and shift invariance, a Graph Shift-Invariance-Adversarial-Robustness analysis was employed. This analysis involved plotting models on a graph with robustness (adversarial accuracy) on one axis and shift invariance (consistency) on the other. The graph was divided into four zones using the fully connected models as a reference point: 10 9 12

- **Top Left**: This zone typically encompasses the robust models, often including robust



(a) The graph for MNIST data set, for $norm_{\infty}, epsilon = 0.1$



(b) The graph for Fashion MNIST data set, for $norm_{\infty}, epsilon = 0.1$

Figure 9: Shift-Invariance-Adversarial-Robustness Plot

FC models. Models here exhibit high adversarial accuracy but low shift invariance.

- **Top Right**: This is the desired zone,or Goal zone, representing models that are both robust and shift invariant.
- **Bottom Left**: Models in this zone are neither robust nor shift invariant, making it an undesirable region, the dead zone.
- **Bottom Right**: This zone is usually occupied by classic shift invariant CNN models (followiong the paper), demonstrating high shift consistency but low adversarial accuracy.

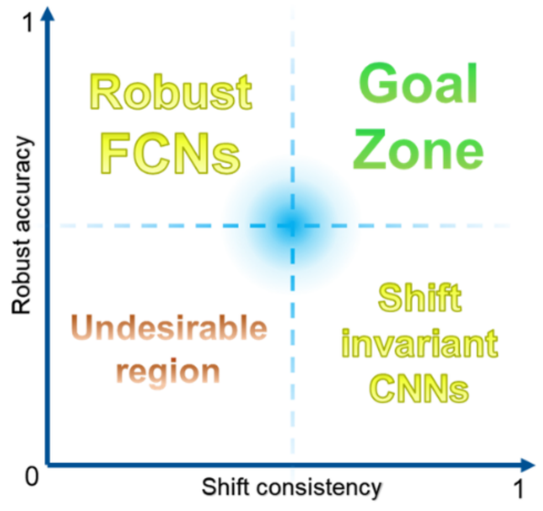
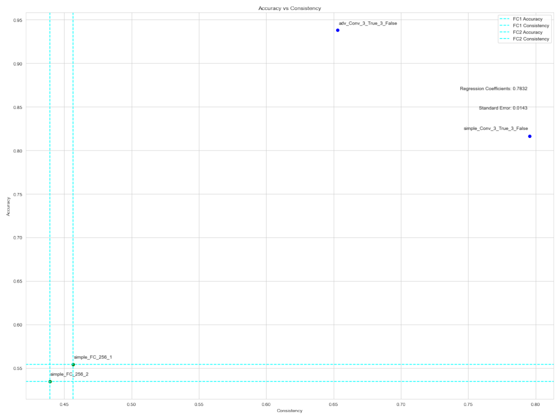


Figure 10: Graph Shift-Invariance-Adversarial-Robustness

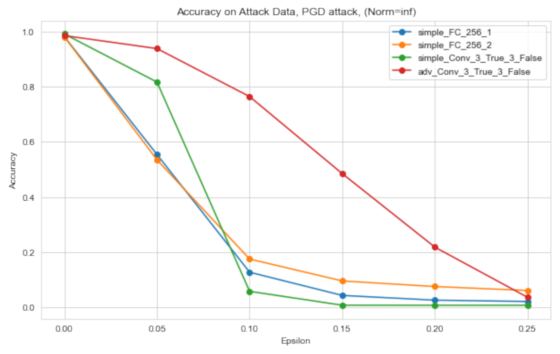
The proposed process to identify robust models involves the following steps:

1. **Training on a Smaller Dataset**: Train various model configurations on a smaller dataset and plot their positions on the Graph Shift-Invariance-Adversarial-Robustness. 13
2. **Identify Goal Zone Models**: Select models that fall within or near the goal zone (top right) of the graph.
3. **Full Training and Improvement**: Fully train the selected models. Improve and fine-tune these models using additional techniques. 11
4. **Prioritize Based on Requirements**: Depending on priorities (e.g., robustness, shift invariance), select models that align with the preferred characteristics. For example if our priority is to have the most robust model while keeping shift invariance. First we get a model from the Goal zone that has the best shift invariance, then we retrain the model using adversarial training. The other technique include: Neural_structured_learning [3] for a better robustness, or using anti-aliasing with the

classical pooling layers [4]...



(a) Shift-Invariance-Adversarial-Robustness graph after adversarial training



(b) Robust Accuracy for the chosen model, and the adversarial trained model compared with FCNs

Figure 11: Using Shift-Invariance-Adversarial-Robustness Plot and adversarial training to get better models

This process was successfully demonstrated with the MNIST dataset, 11 where models from the goal zone were retrained using adversarial training, leading to enhanced robustness.

3 Conclusion

In conclusion, while individual hyperparameters contribute to robustness, achieving an optimal trade-off between adversarial robustness and shift invariance requires a comprehensive exploration and fine-tuning of multiple parameters. The proposed Graph Shift-Invariance-Adversarial-Robustness analysis provides a systematic approach to identify models that meet both criteria effectively.

References

[1] Songwei Ge et al. *Shift Invariance Can Reduce Adversarial Robustness*. 2021. arXiv: 2103.02695 [cs.LG].

[2] Cong Xu and Min Yang. *Understanding Adversarial Robustness from Feature Maps of Convolutional Layers*. 2022. arXiv: 2202.12435 [cs.CV].

[3] *Neural Structured Learning — TensorFlow — tensorflow.org*. https://www.tensorflow.org/neural_structured_learning.

[4] Richard Zhang. *Making Convolutional Networks Shift-Invariant Again*. 2019. arXiv: 1904.11486 [cs.CV].

[5] Pin-Yu Chen and Sijia Liu. *Holistic Adversarial Robustness of Deep Learning Models*. 2023. arXiv: 2202.07201 [cs.LG].

A Additional Figures

B Theorems and Proofs

[TO DO] [1][2]

C The Adversarial Attacks

This section is based on the Holistic Adversarial Robustness of Deep Learning Models paper [5]. It provides an overview of common adversarial attacks that exploit vulnerabilities in AI models during both the training and deployment phases. The objectives of these attacks are summarized in Table 1.

C.1 Training-Phase Attacks

Training-phase attacks involve manipulating the training data to create vulnerabilities in the resulting AI model.

Poisoning Attack aims to craft a poisoned dataset D_{poison} that causes models trained on it to perform poorly on the test data D_{test}. This involves altering D_{train} with label flipping,

Attack	Objective
Poisoning	Design a poisoned dataset D_{poison} such that models trained on D_{poison} fail to generalize on D_{test} (i.e. $\hat{y}_\theta(x_{\text{test}}) \neq y_{\text{test}}$)
Backdoor	Embed a trigger Δ with a target label t to D_{train} such that $\hat{y}_\theta(x_{\text{test}}) = y_{\text{test}}$ but $\hat{y}_\theta(x_{\text{test}} + \Delta) = t$
Evasion (untargeted)	Given f_θ, find x_{adv} such that x_{adv} is similar to x but $\hat{y}_\theta(x_{\text{adv}}) \neq y$
Evasion (targeted)	Given f_θ, find x_{adv} such that x_{adv} is similar to x but $\hat{y}_\theta(x_{\text{adv}}) = t$

Table 1: Objectives of adversarial attacks.

data addition or deletion, or feature modification. The model converges to a "bad" local minimum due to the poison data.

Backdoor Attack involves embedding a trigger Δ with a modified target label t into a subset of D_{train}. The model behaves normally in the absence of the trigger but predicts any input with the trigger as label t. This attack can be stealthy and challenging to detect.

C.2 Deployment-Phase Attacks

Deployment-phase attacks focus on generating adversarial examples that evade the model's prediction.

Untargeted Evasion Attack aims to find an example $\mathcal{T}(x)$ similar to x, such that the model predicts $\mathcal{T}(x)$ differently from its original label y.

Targeted Evasion Attack similarly finds $\mathcal{T}(x)$, but with the goal of making the model predict a specific target label t instead of the true label y.

These attacks often involve perturbing the input within certain similarity norms (ℓ_p norms), where each norm type measures the distance between the original and perturbed input in a specific way.

White-Box Attack assumes full knowledge of the target model's architecture and parameters, allowing attackers to craft adversarial examples using backpropagation. This is the type of models that we were studying in this project.

Black-Box Attack works with limited knowledge of the target model, usually only observing its predictions. Soft-label black-box attacks use confidence scores, while hard-label black-box attacks only have access to top-1 predictions.

Transfer Attack is a black-box strategy that uses adversarial examples generated from a white-box surrogate model to attack the target model. This takes advantage of the transferability of adversarial examples across models.

C.3 Examples of Attacks

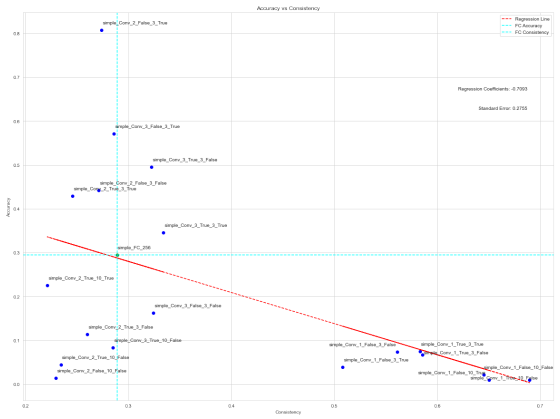
Common attack methods include:

- *FGM (Fast Gradient Method)*: A white-box attack that perturbs input using the gradient of the model's loss with respect to the input.

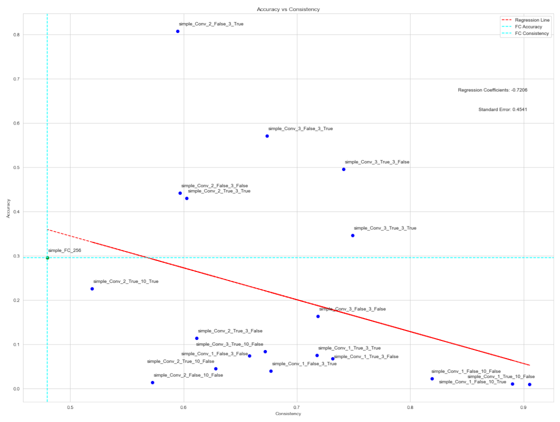
- *PGD (Projected Gradient Descent)*: An iterative version of FGM that applies small perturbations while staying within a bounded norm.

- *DeepFool*: An untargeted attack that iteratively pushes the input across the decision boundary by perturbing it in the direction of the decision boundary's normal vector.

These examples highlight the diversity of adversarial attacks and their potential to undermine AI model reliability.

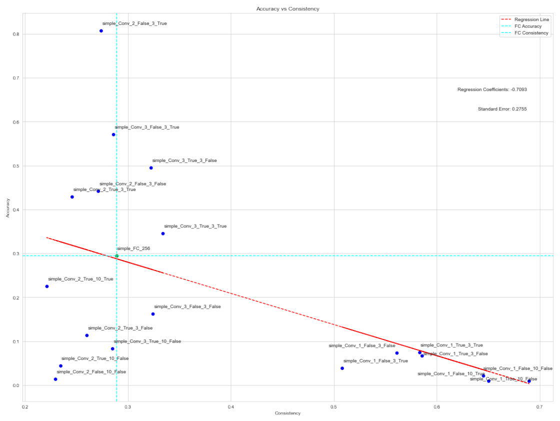


(a) The graph for MNIST data set, for $norm_\infty, epsilon = 0.1$, with the $maxshift = 10$ for consistency score measurement

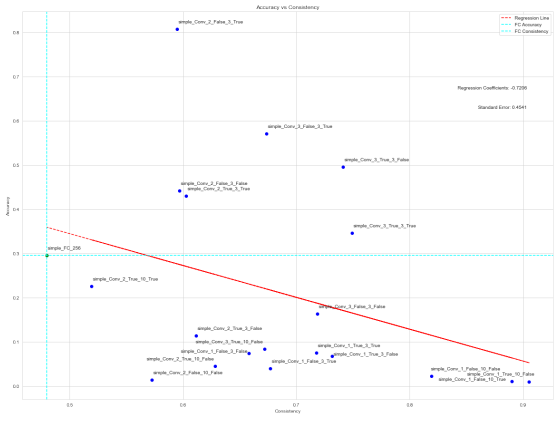


(b) The graph for MNIST data set, for $norm_\infty, epsilon = 0.1$, with the $maxshift = 5$ for consistency score measurement

Figure 12: Shift-Invariance-Adversarial-Robustness Plot for different consistency measurements



(a) The graph for MNIST data set, for $norm_\infty, epsilon = 0.1$, with the $maxshift = 10$ for consistency score measurement



(b) The graph for MNIST data set, for $norm_\infty, epsilon = 0.1$, with the $maxshift = 5$ for consistency score measurement

Figure 13: Shift-Invariance-Adversarial-Robustness Plot for different consistency measurements

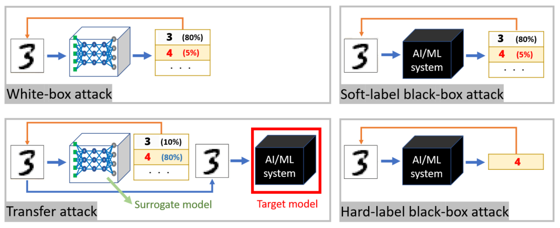


Figure 14: Evasion Attack Taxonomy