# Evaluating Adversarial Robustness of Pretrained Image Classifiers

# 1. Introduction

In recent Years, deep learning algorithms have been outstanding in performing classification of images. Despite their remarkable performance, deep learning models have significant limitations caused by the model's unique sensitivity to adversarial examples, or inputs specifically designed to mislead a model. This project studies the adversarial robustness of a pretrained image classifier based on ResNet50 architecture. I have analyzed the bean leaf images with respect to how susceptible the model is to adversarial perturbations through three different black box attack methods: Square Attack, EAD Attack, and ZOO Attack. In addition, analyzed the results of a defense method known as Feature Squeezing, which lessens the impact of an attack based on image quantization and smoothing.

2. Project MotivationWith the increasing deployment of AI in agriculture, medical imaging, and autonomous systems, the reliability of machine learning models becomes critical. These models are often treated as black boxes, making them susceptible to adversarial inputs crafted without access to internal weights. This project aims to:

* Assess the robustness of a pretrained ResNet50 classifier under black-box attack conditions.
* Compare the effectiveness of three adversarial algorithms in fooling the model.
* Evaluate Feature Squeezing as a defense mechanism for input sanitization.
* Provide insights into the practical limitations of pretrained models in adversarial settings.

# 3. Dataset and Preprocessing

The dataset used in this study consists of annotated images of bean leaves categorized into three classes: healthy, angular leaf spot, and bean rust. The images were provided in ZIP archives corresponding to training, validation, and test sets. Using Python’s zipfile library, these were extracted into respective directories. The data was then loaded using TensorFlow’s `image\_dataset\_from\_directory` API. Each image was resized to 224x224 pixels to match the input requirement of the ResNet50 model. The data was batched with a size of 32 for training and validation, and 50 for the test set. Labels were automatically inferred based on the folder structure. Data was also prefetched to optimize performance.

# 4. Model Training

Employed a transfer learning approach using the ResNet50 architecture pretrained on the ImageNet dataset. The top classification layers of the model were removed (`include\_top=False`) and replaced with a custom classifier comprising a global average pooling layer and a dense output layer with softmax activation for three-class classification. The base ResNet50 model was frozen during training to retain learned features and reduce computational cost. The network was compiled using the Adam optimizer and sparse categorical cross-entropy loss. The model was trained for 30 epochs on the training set, with validation monitored at each epoch.

# 5. Model Performance

After training, the model achieved excellent performance on the validation set, with a final validation accuracy of 95.49% and validation loss of 0.1429. The training accuracy reached 93.03%, indicating minimal overfitting. The loss and accuracy curves plotted over the 30 epochs revealed smooth convergence. This strong baseline performance highlights the classifier’s competence on clean data.

Performance Summary:

• Final Training Accuracy: 93.03%  
• Final Training Loss: 0.2028  
• Validation Accuracy: 95.49%  
• Validation Loss: 0.1429

Additionally, the training and validation performance over epochs is visualized below:

**A screenshot of a graph

AI-generated content may be incorrect.**

# 6. Adversarial Attacks

Implemented three types of black-box adversarial attacks: Square Attack, Elastic-Net Attack (EAD), and Zeroth Order Optimization (ZOO) Attack. These attacks do not require access to the model's gradients or internal architecture, making them more applicable to real-world black-box scenarios.

• Square Attack: This is a query-efficient black-box attack that perturbs square patches of the input image using randomly sampled values. The attack is designed to find minimal modifications that result in misclassification. It is particularly effective on models with limited defenses.

• EAD Attack: The Elastic-Net Attack is an optimization-based method that extends the Carlini-Wagner attack by applying both L1 and L2 regularizations to generate perturbations. It is designed to produce sparse yet effective adversarial examples, balancing imperceptibility and fooling power.

• ZOO Attack: The Zeroth Order Optimization attack estimates gradients through finite differences, allowing adversarial samples to be crafted without knowledge of the model’s parameters. It is especially effective in strict black-box conditions, though it can be computationally intensive due to high query counts.

# 7. Feature Squeezing Defense

Feature Squeezing is a preprocessing-based defense technique that aims to reduce the input space's complexity by compressing the image. This technique includes bit-depth reduction (e.g., from 8-bit to 4-bit) and optional smoothing (e.g., median filtering). The underlying idea is that adversarial perturbations often manifest as fine-grained noise in high-resolution image space. By squeezing the features into a lower-dimensional representation, these perturbations can be nullified or made less effective. This defense does not require retraining the model and is easy to implement, making it an attractive first line of defense.

# 8. Evaluation

Evaluated the model using both defended and undefended versions to measure the success rates of the attacks. For each epsilon (perturbation strength), Recorded whether the model was fooled (i.e., produced incorrect output) and the number of queries required. Visualizations such as confusion matrices and adversarial vs. squeezed image plots were used to support the analysis. These comparisons help us understand the effectiveness of Feature Squeezing under each attack scenario.

# 9. Feature Squeezing Effect Visualization

To understand the mechanics of feature squeezing, Visualized an adversarial image before and after squeezing, as well as the pixel-wise difference. The adversarial image shows minor changes imperceptible to the human eye. After squeezing, the image retains its general structure but eliminates high-frequency artifacts. The difference map highlights where modifications were made, reinforcing the notion that squeezing targets subtle perturbations.

# 10. Attack Success Table (With vs Without Defense)

Compiled a table to quantitatively summarize attack outcomes with and without the feature squeezing defense applied. Each attack was tested across a range of epsilon values, and Recorded whether the model misclassified the sample, both with and without defense.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attack** | **Epsilon** | **Success Without Defense** | **Success With Defense** | **Queries Used** |
| Square Attack | 5.000 | True | True | 0 |
| Square Attack | 10.000 | True | True | 0 |
| Square Attack | 20.000 | True | True | 0 |
| Square Attack | 30.000 | True | True | 0 |
| Square Attack | 50.000 | True | True | 0 |
| EAD Attack | 0.005 | True | True | 1 |
| EAD Attack | 0.010 | True | True | 1 |
| EAD Attack | 0.020 | True | True | 1 |
| EAD Attack | 0.050 | True | True | 1 |
| EAD Attack | 0.100 | True | True | 1 |
| ZOO Attack | 0.010 | True | True | 0 |
| ZOO Attack | 0.030 | True | True | 0 |
| ZOO Attack | 0.050 | True | True | 0 |
| ZOO Attack | 0.070 | True | True | 0 |
| ZOO Attack | 0.100 | True | True | 0 |

Observation: Across all attacks and perturbation strengths, the model remained vulnerable, even after applying feature squeezing. This demonstrates that while feature squeezing has some effect on modifying inputs, it is not sufficient alone to fully prevent adversarial misclassification.

# 11. Limitations

This work has a few limitations that should be acknowledged:

* **Limited Evaluation Samples:** Some evaluations were done using a small number of test images, which may not reflect full dataset behavior.
* **No Adversarial Detection Metrics:** The current evaluation focuses on misclassification; a more complete analysis would include detection success or confidence shift tracking.

# 12. Technical Challenges and Solutions

During this project, several technical difficulties were encountered and addressed:

* **Model Training Time:** Training a ResNet50 model with full-size bean leaf images on Google Colab took considerable time. To mitigate this, the model's base was frozen, and only the final classification layers were trained.
* **Attack Implementation Complexity:** Implementing the ZOO and EAD attacks required careful adaptation, especially since they needed to operate in a black-box setting. Custom code was written to simulate finite-difference approximations and add regularization terms efficiently.
* **Visualization Overhead:** Rendering visual comparisons (e.g., adversarial vs. squeezed vs. original) for multiple attack types was compute-intensive. A selective sampling approach was used to visualize only representative images.
* **Query Management:** Some attacks required hundreds or thousands of queries per image. To stay within reasonable limits, epsilon values were adjusted and capped to maintain test feasibility.

# 13. Ethical Considerations While adversarial attack research is crucial for improving AI safety, it also comes with ethical responsibilities. This project focuses on understanding model vulnerabilities, not exploiting them. All techniques used are meant to raise awareness and encourage the development of more resilient and secure AI systems. If deployed irresponsibly, similar techniques could harm critical systems in healthcare, security, or infrastructure. 14. Real World Use Case:

* **Plant Disease Diagnosis**: In agriculture apps, adversarial noise could hide symptoms, causing delayed treatment.
* **Security Systems**: Visual recognition models (e.g., for faces, license plates) can be tricked by minor physical alterations.

# 15. Conclusion

This project illustrates the importance of evaluating model robustness against adversarial attacks. Despite achieving high performance on clean validation data, the ResNet50 classifier was easily misled by small perturbations generated by black-box attacks. Feature Squeezing, though conceptually simple and computationally efficient, did not offer robust protection against these attacks. Future work could explore combining multiple defense techniques, adversarial training, or incorporating detection mechanisms to improve resilience.

## **References**

1. **TensorFlow Documentation**  
   Used for model training and image preprocessing.  
   <https://www.tensorflow.org/>
2. **Keras Applications – ResNet50**  
   Source for the pretrained ResNet50 model used in transfer learning.  
   <https://keras.io/api/applications/resnet/>
3. **Square Attack (Official GitHub Repository)**  
   A query-efficient black-box attack using randomly placed square perturbations.  
   <https://github.com/max-andr/square-attack>
4. **EAD Attack – IBM Adversarial Robustness Toolbox**  
   Elastic-Net Attack combining L1 and L2 distortions.  
   <https://github.com/IBM/EAD-Attack>
5. **ZOO Attack (Zeroth Order Optimization)**  
   Gradient-free black-box attack based on finite-difference approximations.  
   <https://github.com/huanzhang12/ZOO-Attack>
6. **Feature Squeezing – NDSS Symposium 2018**  
   Original academic paper introducing feature squeezing for adversarial defense.  
   Xu, W., Evans, D., & Qi, Y. (2018).  
   “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks.”  
   <https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_03A-4_Xu_paper.pdf>
7. **Matplotlib, NumPy, and Seaborn**  
   Used for visualizing training curves, adversarial images, and confusion matrices.  
   <https://matplotlib.org/>  
   <https://numpy.org/>  
   <https://seaborn.pydata.org/>
8. **Bean Leaf Dataset**  
   Public image dataset containing labeled samples of healthy and diseased bean leaves. Often used in image classification and plant pathology research.  
   <https://github.com/trizkynoviandy/bean-leaf-classification>
9. **ChatGPT   
   Used AI on how to load the dataset for train, Val and test.**<https://chat.openai.com/>