Deep Learning Lab 6 Report

Shusrith S PES1UG22AM155
Siddhi Zanwar PES1UG22AM161

March 9, 2025

1 Adversarial Attacks

Adversarial attacks are methods that are specifically used to manipulate input data in order to mislead deep learning models. Adversarial attacks make use of small, well-crafted perturbations that may be imperceptible to humans but can mislead a model to make erroneous predictions with strong confidence. Because deep learning models, especially Convolutional Neural Networks (CNNs), depend on learned feature mappings and decision boundaries, slight alterations in pixel values can cause serious classification mistakes.

2 Real life implications

Adversarial attacks are extremely dangerous in safety-critical applications where deep learning models are utilized. Some of the primary concerns are:

2.1 Medical Imaging

Diagnostic systems based on AI use Convolutional Neural Networks (CNNs) to process medical scans like MRIs, X-rays, and CT scans. Adversarial perturbations can make small changes to images, causing misdiagnosis. For instance, an attack might alter an MRI scan to present a malignant tumor as benign, which could result in inappropriate treatment decisions.

2.2 Autonomous Driving

Deep learning models are used in self-driving cars for object recognition and scene understanding. Adversarial attacks may deceive such systems into misinterpreting road signs. For example, carefully placed stickers on a stop sign can lead the system to identify it as a speed limit sign, which can lead to dangerous circumstances.

2.3 Security Systems

Numerous authentication systems incorporate face recognition, fingerprint readers, or voice models for recognition. Attackers may use adversarial examples to impersonate another individual or evade security systems. This might allow unauthorized entry into restricted spaces or sensitive information, causing substantial security risks.

3 FGSM attack

The Fast Gradient Sign Method (FGSM) is a simple yet effective adversarial attack technique used to generate adversarial examples. It exploits a model's gradients to perturb an input in a way that maximizes the loss function, making the model misclassify the input.

3.1 Mathematical Formulation

FGSM modifies an input x by adding a small perturbation in the direction of the gradient of the loss function with respect to the input:

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

where:

- x is the original input (e.g., an image),
- x' is the adversarial example,
- ϵ is a small positive value that controls the strength of the perturbation,
- $J(\theta, x, y)$ is the loss function of the model with parameters θ,
- $\nabla_x J$ is the gradient of the loss function with respect to the input,
- $\text{sign}(\cdot)$ takes the sign of each element in the gradient, ensuring a minimal yet effective perturbation.

3.2 How FGSM Works

1. The attacker computes the gradient of the loss function with respect to the input image. This gradient indicates how the model's prediction would change if the input were modified.
2. A small perturbation is added in the direction that maximizes the model's loss.
3. The modified input, though nearly identical to the original image, causes the model to make a wrong prediction with high confidence.

3.3 Why FGSM Is Effective

- **Computationally Efficient:** Unlike iterative attacks, FGSM requires only a single gradient computation, making it fast and easy to execute.
- **Simple Yet Powerful:** Even with small values of ϵ, FGSM can significantly degrade model accuracy.
- **Foundation for More Advanced Attacks:** Many stronger adversarial attack methods, such as Projected Gradient Descent (PGD) and Basic Iterative Method (BIM), build upon the FGSM approach.

4 Training the model

4.1 Architecture Used

The model presented is a Convolutional Neural Network with Residual Blocks, inspired by the ResNet architecture. The model leverages residual learning, where shortcut connections help gradients flow through the network, allowing for deeper architectures without suffering from vanishing gradient issues. The network consists of an initial convolutional layer, followed by three stacked Residual Blocks with increasing channel sizes, concluding with global average pooling and a fully connected layer for classification.

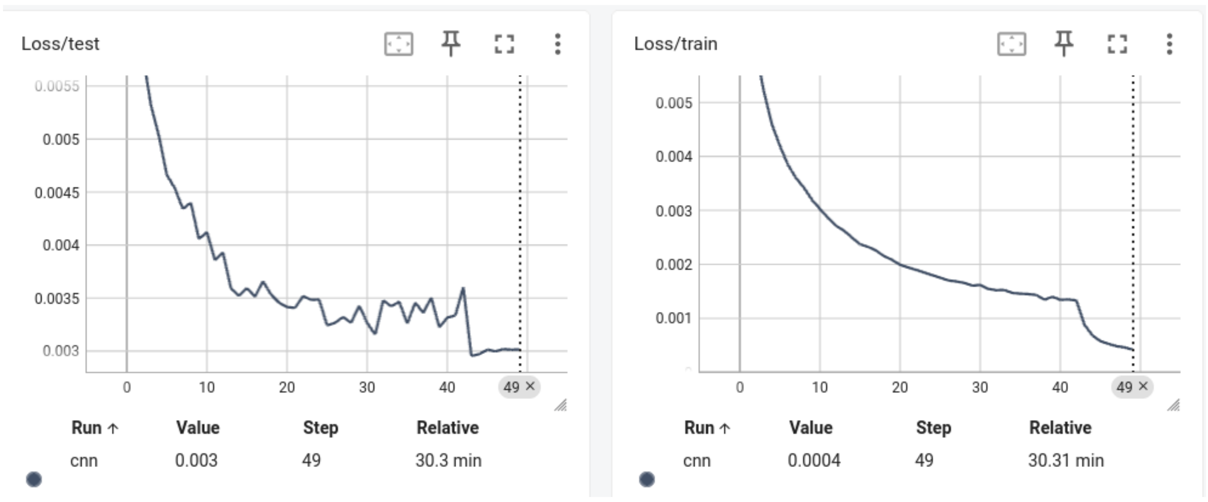
4.2 Description of Architecture

The first layer is a 3×3 convolution with 64 filters, batch normalization, and ReLU activation, followed by three Residual Blocks, where each block consists of two convolutional layers with batch normalization and ReLU activation. A crucial component of these blocks is the skip connection, which helps retain information from previous layers, preventing degradation in performance as the network deepens.

4.3 Design Choices

One key design choice is the use of Residual Blocks, which improve the network's ability to train deeper architectures effectively. The shortcut connection allows direct flow of gradients during backpropagation, reducing the risk of vanishing gradients. Another important choice is the use of Batch Normalization after every convolution, which helps in stabilizing training and accelerating convergence. Additionally, Adaptive Average Pooling is used before the final fully connected layer, allowing the model to handle varying input sizes efficiently while reducing overfitting compared to a traditional flattening approach.

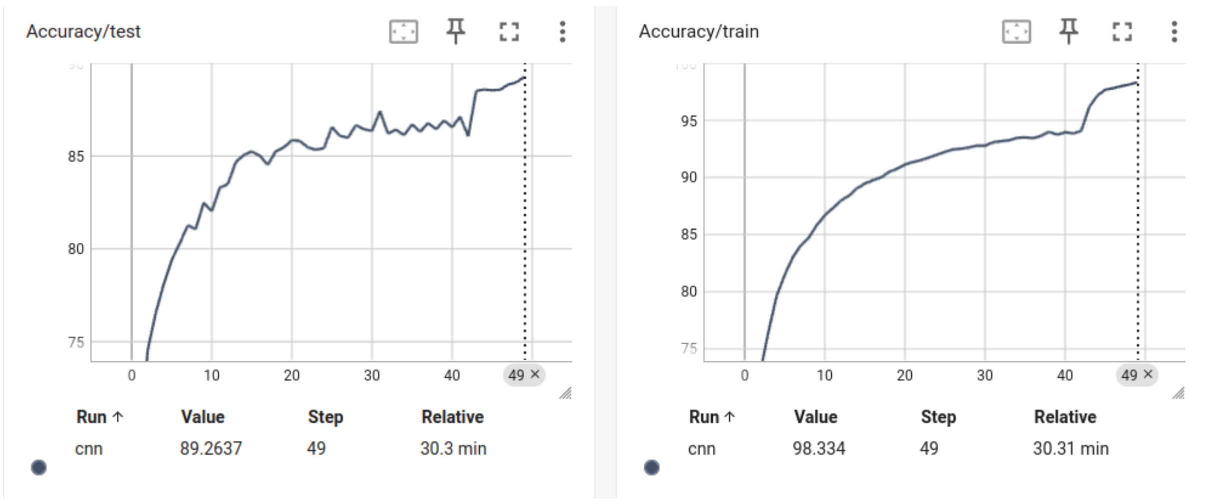
4.4 Loss and Accuracy plots



Run ↑	Value	Step	Relative
● cnn	0.003	49	30.3 min

Run ↑	Value	Step	Relative
● cnn	0.0004	49	30.31 min

Figure 1: Train and test loss plots



Run ↑	Value	Step	Relative
● cnn	89.2637	49	30.3 min

Run ↑	Value	Step	Relative
● cnn	98.334	49	30.31 min

Figure 2: Train and test accuracy plot

4.5 Accuracy

The model trains with a final train accuracy of 98.33% and a a test accuracy of 89.26%. The final losses for train and test are 4×10^{-4} and 3×10^{-3}. It is visible that the model has overfit despite best attempts to include various regularization techniques like Batch Normalization, learning rate scheduling and weight decay. It is the nature of deeper Convolution Neural Networks to tend to overfit and hence it can be concluded that better skip connections and fewer parameters might be needed to reduce the overfitting.

5 Implementing the attack

The FGSM attack is used to test how easily a neural network can be fooled by small changes in an image. It works by selecting a random image from the dataset and slightly modifying its pixel values in a way that confuses the model. This is done by calculating how much each pixel affects the model's prediction and then adjusting them in the direction that increases the chance of a wrong answer. The changes are small enough that they don't make the image look different to the human eye, but they are enough to trick the model. To keep the image valid, the pixel values are kept within a normal range. After modifying 100 images this way, the model's accuracy is checked to see how often it still predicts correctly. The average loss is also measured to understand how much the model struggles with these altered images. The results show how vulnerable the model is to tiny changes and how easily it can be misled. By tracking accuracy and loss, the experiment helps in understanding how strong or weak the model is against adversarial attacks like FGSM.

The obtained accuracy is 21% after implementing the attack, a huge drop from the initial 89%. The image shown below has an FGSM attack implemented on it, and is incorrectly classified by the model.

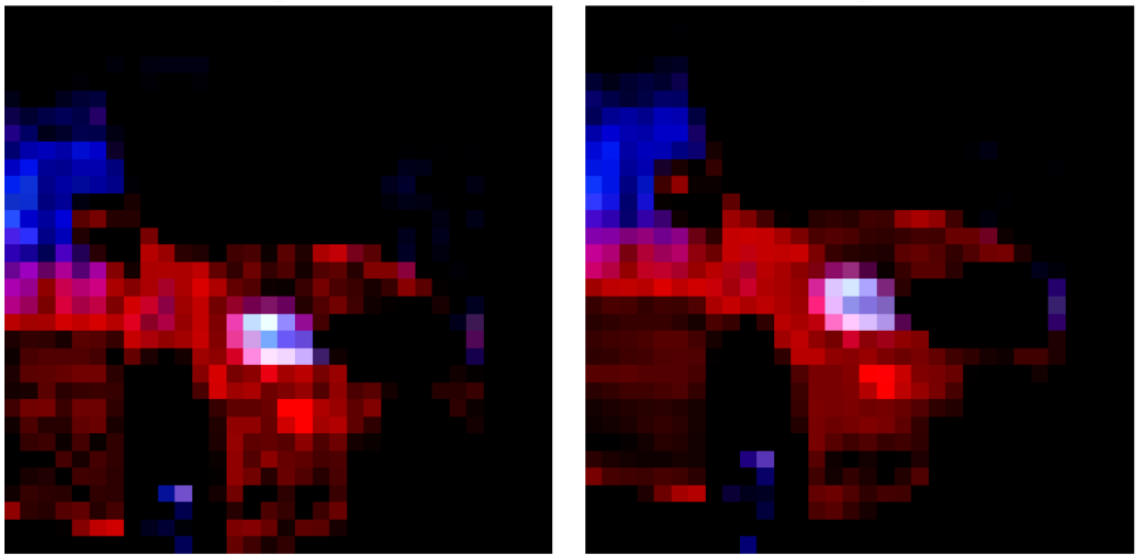


Figure 3: a) Image after implementing FGSM attack b) Original image

6 Defense techniques

6.1 Adversarial training

Adversarial training works by creating adversarial examples during training and adding them to the dataset to enhance model robustness. Through training to classify correctly perturbed inputs, the model gets more resistant to small, targeted perturbations. While strong, it increases computational cost heavily and does not always generalize well to out-of-distribution attack strategies. The image shown below has an FGSM attack implemented on it, but is still correctly classified by the model.

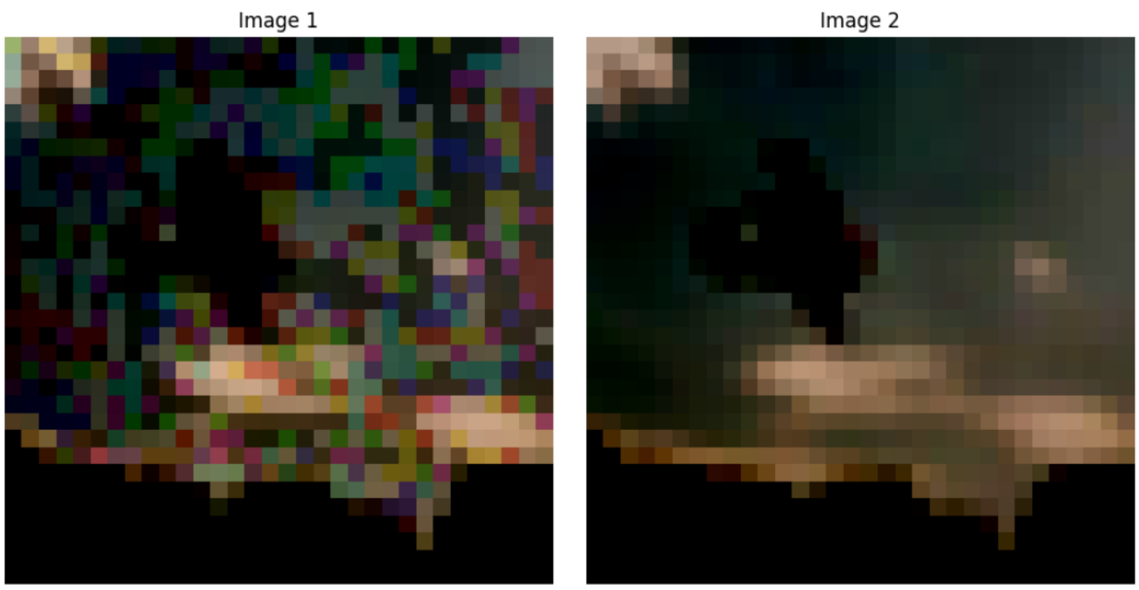


Figure 4: a) Image after implementing FGSM attack with adversarial training b) Original image

Accuracy 74%

6.2 Defensive Distillation

This approach trains a teacher model first and then leverages its smoothed output probabilities (obtained with an elevated temperature in softmax) to train a student model. By softening the sensitivity of the student model to subtle input variations, this method impairs the effectiveness of gradient-based adversarial attacks. Yet it has been found that adaptive attacks can still breach this defense and render it less effective against robust attacks. The image shown below has an FGSM attack implemented on it, but is still correctly classified by the model.

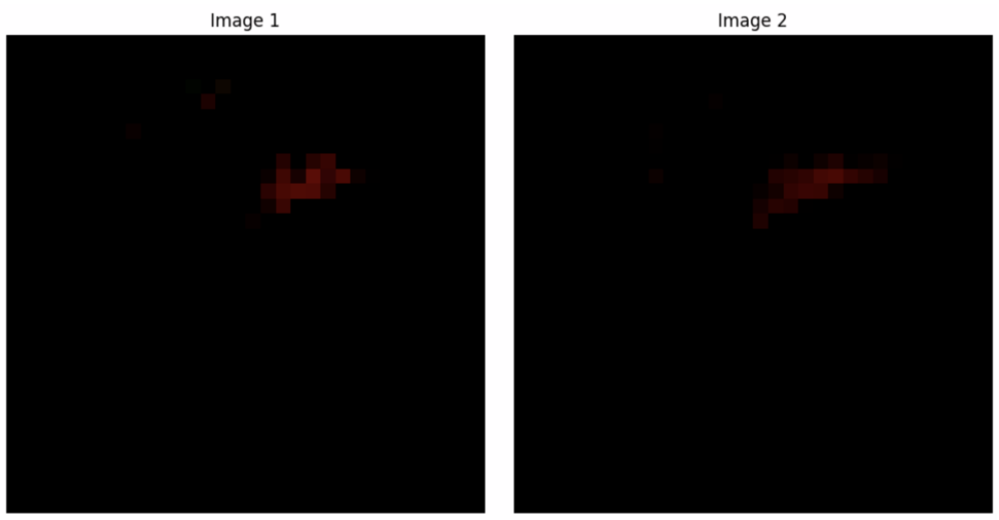


Figure 5: a) Image after implementing FGSM attack with defensive distillation b) Original image

Accuracy 60%

6.3 Gradient Masking

This defense tries to mask or modify gradients to thwart adversarial attacks from utilizing them to their fullest extent. Randomization of gradients, introduction of noise, or employing non-differentiable layers complicate attacks to create adversarial examples. Nevertheless, the method is not airtight since attacks are capable of exploiting black-box methods such as transfer attacks or gradient approximations in numbers to evade the defense. The image shown below has an FGSM attack implemented on it, but is still correctly classified by the model.

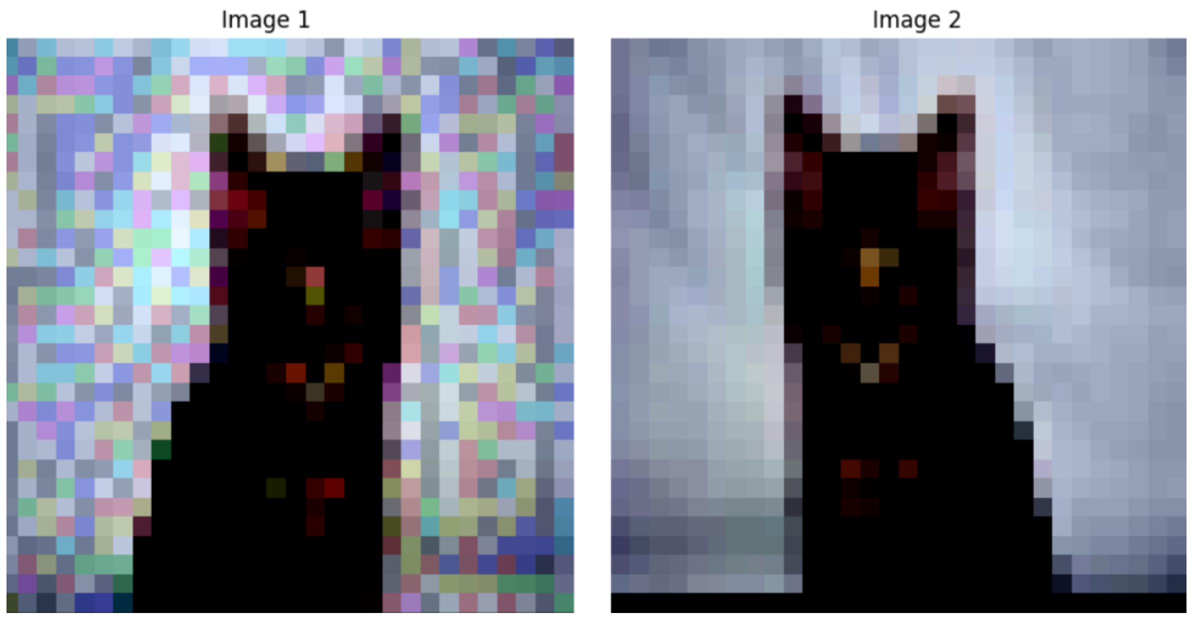


Figure 6: a) Image after implementing FGSM attack on a model with Gradient Masking b) Original Image

Accuracy 55%

6.4 Feature Squeezing

Feature squeezing decreases the sensitivity of models towards small perturbations by using simple transformations, including bit-depth reduction, blurring of images, or median filtering. By reducing the level of unwanted information in input features, it becomes more difficult for attackers to inject imperceptible adversarial noise. Though computationally cost-effective, sometimes it harms model accuracy on clean data as well. The image shown below has an FGSM attack implemented on it, but is still correctly classified by the model.

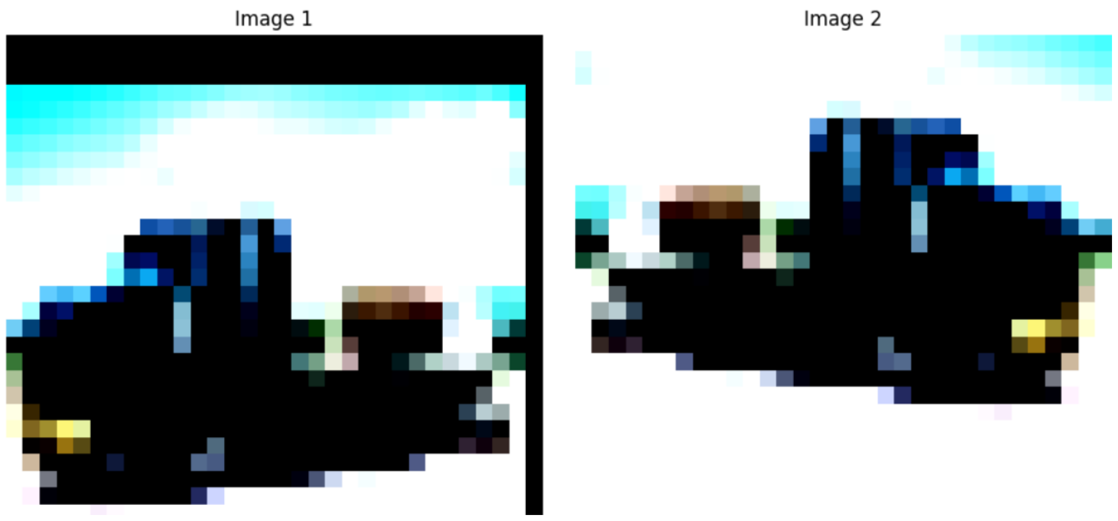


Figure 7: a) Image after implementing FGSM attack on a model with feature squeezing b) Original image

Accuracy 63%