DEFENSE ALGORITHMS AGAINST A FAST GRADIENT SIGN METHOD ATTACK

Oğuz Kağan Koçak

Overview

Machine learning models, particularly deep neural networks, have achieved remarkable success in various applications. However, these models are vulnerable to adversarial attacks, where small, carefully crafted perturbations to the input data can cause the model to make incorrect predictions. This vulnerability poses significant security risks, especially in critical applications such as healthcare, finance, and autonomous systems.

Adversarial attacks are manipulations designed to fool machine learning models. By introducing subtle perturbations to input data, an attacker can cause the model to misclassify the input, often with high confidence. These attacks exploit the way models learn from data and reveal weaknesses in their decision boundaries.

The Fast Gradient Sign Method is one of the most well-known and straightforward techniques for generating adversarial examples. The simplicity and effectiveness of FGSM make it an excellent starting point for studying adversarial attacks.

Understanding FGSM

The Fast Gradient Sign Method (FGSM) is one of the most basic and widely studied adversarial attack methods in machine learning. It was introduced by Ian Goodfellow, Jonathon Shlens, and Christian Szegedy in their 2014 paper "Explaining and Harnessing Adversarial Examples." FGSM exploits the model's gradients to create adversarial examples, which are inputs to machine learning models that an attacker has intentionally perturbed to cause the model to make a mistake.

FGSM is a gradient-based attack, meaning it leverages the gradients (partial derivatives) of the model's loss function with respect to the input data. These gradients indicate how changes in the input affect the loss, providing a direction to modify the input to increase the loss.

The FGSM generates an adversarial example by perturbing the original input data in the direction of the gradient of the loss function with respect to the input. The mathematical formulation is as follows:

adv_example = input + $\epsilon \cdot \text{sign}(\nabla \text{inputloss})$

input: The original input data.

loss: The loss function used to train the model (e.g., cross-entropy loss).

∇inputloss: The gradient of the loss with respect to the input.

sign(): The sign function takes the sign of each element in the gradient.

ϵ: the bigger the ϵ, bigger the perturbation

CNN Model

First and foremost, to preform an attack to a model, we need the model itself. We chose to implement a simple Convolutional Neural Network (CNN) which is predominantly used to extract the feature from the grid-like matrix dataset (like images and videos). Since our project involves images, and a convolutional neural network is a relatively straightforward model, we decided it would be a good fit.

Our CNN model is designed to classify images. Here's a brief overview of its structure:

Convolutional Layers:

conv1: Takes the input image with 1 channel and applies 32 filters of size 3x3.

conv2: Takes the output from the first convolutional layer and applies 64 filters of size 3x3.

Activation Function:

Uses ReLU activation function after each convolution to introduce non-linearity.

Pooling Layers:

Uses max pooling with a 2x2 window after each convolution to shrink the size of the feature maps.

Fully Connected Layers:

fc1: Flattens the output from the second pooling layer and connects it to a dense layer.

A diagram of a diagram of a block diagram

Description automatically generated with medium confidencefc2: Connects the dense layer to the output layer with 10 units (corresponding to the 10 digits).

Training Method:

Uses cross-entropy loss and the Adam optimizer to train the model for a specified number of epochs.

Performing FGSM Attack

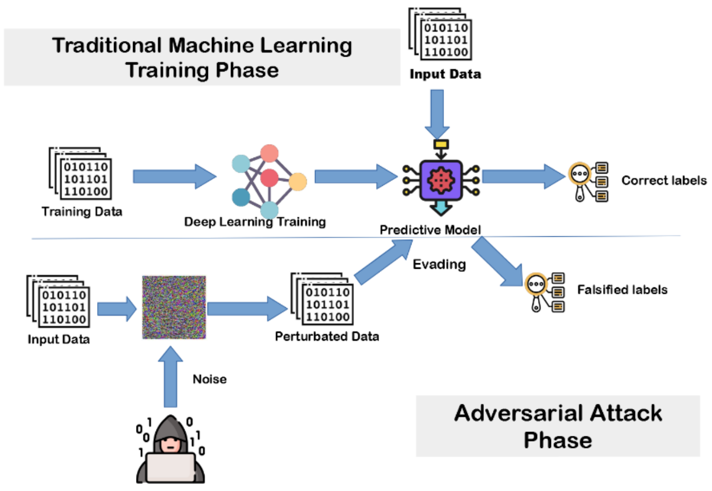
Our model is vulnerable to the FGSM attack due to its reliance on gradient-based optimization during training, which exposes it to perturbations crafted to maximize the loss. Neural networks' sensitivity to small changes in input data, combined with the assumption of local linearity in the model's behaviour exploited by FGSM, further exacerbates this vulnerability. Additionally, the use of the cross-entropy loss in training provides gradients that guide the attack towards generating effective adversarial examples, contributing to the model's susceptibility to misclassification.

Here’s a couple of examples of the attack being perform on images of the MNIST1 dataset:

DEFENSE MECHANISM AGAINST FGSM ATTACK

Introduction to Defense Mechanism

While deep learning models boast impressive capabilities, they can be tricked by adversarial attacks like FGSM. To ensure these models function reliably in real-world applications, implementing effective defenses is essential. This section explores various strategies to defend against FGSM attacks, including adversarial training, special input processing methods, defensive distillation, and gradient masking.



Adversarial Training Method

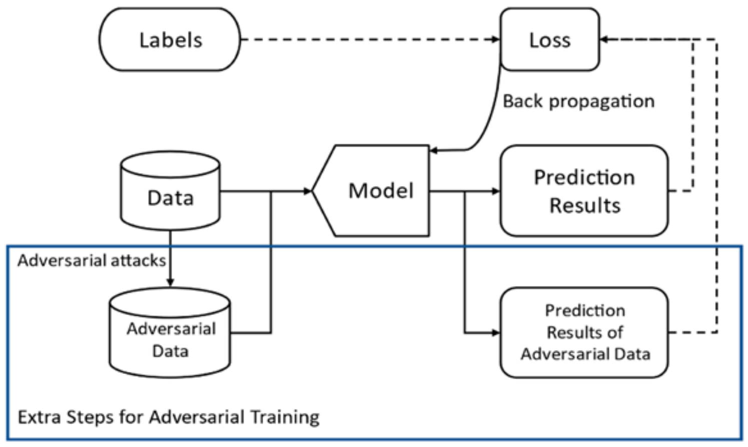
Adversarial training involves augmenting the training dataset with adversarial examples generated by FGSM. By training on these tricky cases, the model learns to better recognize and correctly classify perturbed inputs. The unique aspect of our approach is the use of varying epsilon values during adversarial training. This technique ensures the model is exposed to adversarial examples of different intensities, enhancing its robustness across a broader spectrum of attacks.

Implementation Overview:

FGSM Attack with Varying Epsilon Values: We generate adversarial examples using multiple epsilon values. This variation ensures the model is resilient to a range of perturbations.

Combining Adversarial Examples: We combine the original and adversarial examples into a single batch for training.

Training Loop: The model is trained on this augmented dataset, which includes both original and adversarial examples.



Stochastic Activation Pruning

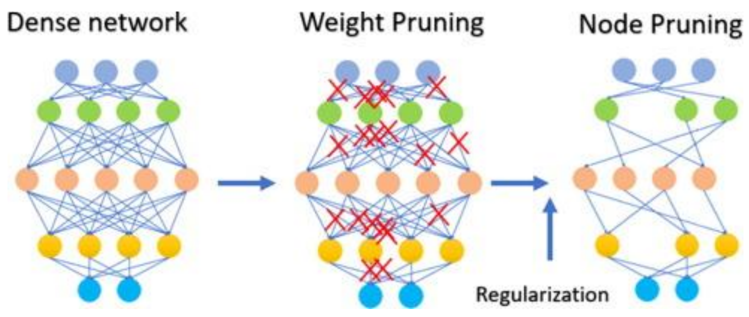
Stochastic activation pruning introduces randomness into the model by selectively deactivating a fraction of neurons during training. This technique prevents attackers from exploiting specific neurons, making the model more resilient to adversarial attacks.

Implementation Overview:

Model Architecture: The model architecture is modified to include stochastic activation pruning, where neurons are randomly deactivated based on a predefined probability.

Training Process: During training, the model selectively deactivates neurons, forcing it to rely on a diverse set of neurons, thus making it harder for adversarial attacks to succeed.

Training Process: The model is trained on these transformed inputs, which helps it become more robust against adversarial attacks.



In our project we applied the Pruning model in our CNN model. Here’s some examples of comparison of performance:

Dynamic Ensemble with Random Subnetwork Activation

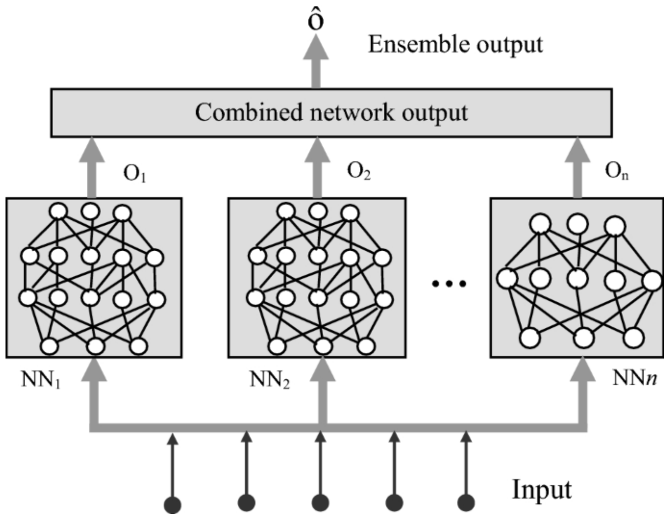
Dynamic ensemble with random subnetwork activation increases the model's uncertainty by processing each input with a randomly selected subnetwork. This technique creates additional complexity for attackers, as they cannot predict which subnetwork will be active for a given input.

Implementation Overview:

Dynamic Ensemble Model: The model consists of multiple subnetworks, each with its own set of parameters.

Random Subnetwork Activation: For each input, a random subnetwork is activated, providing diverse responses to adversarial examples.

Training Process: The model is trained using this dynamic ensemble approach, enhancing its robustness against adversarial attacks.



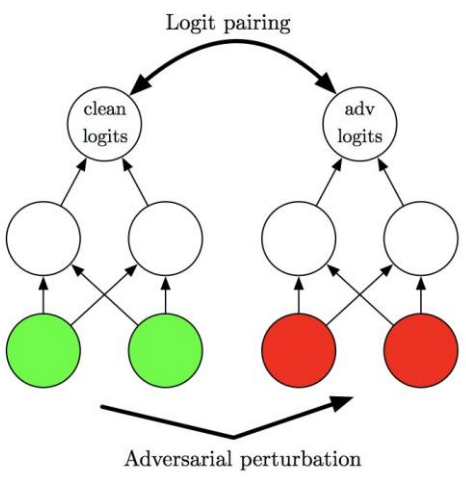
Adversarial Logit Pairing

Adversarial logit pairing minimizes the differences between the logits of clean and adversarial examples, making it harder for adversarial perturbations to change the model's predictions.

Implementation Overview:

Logit Pairing: The logits of clean and adversarial examples are paired to minimize their differences.

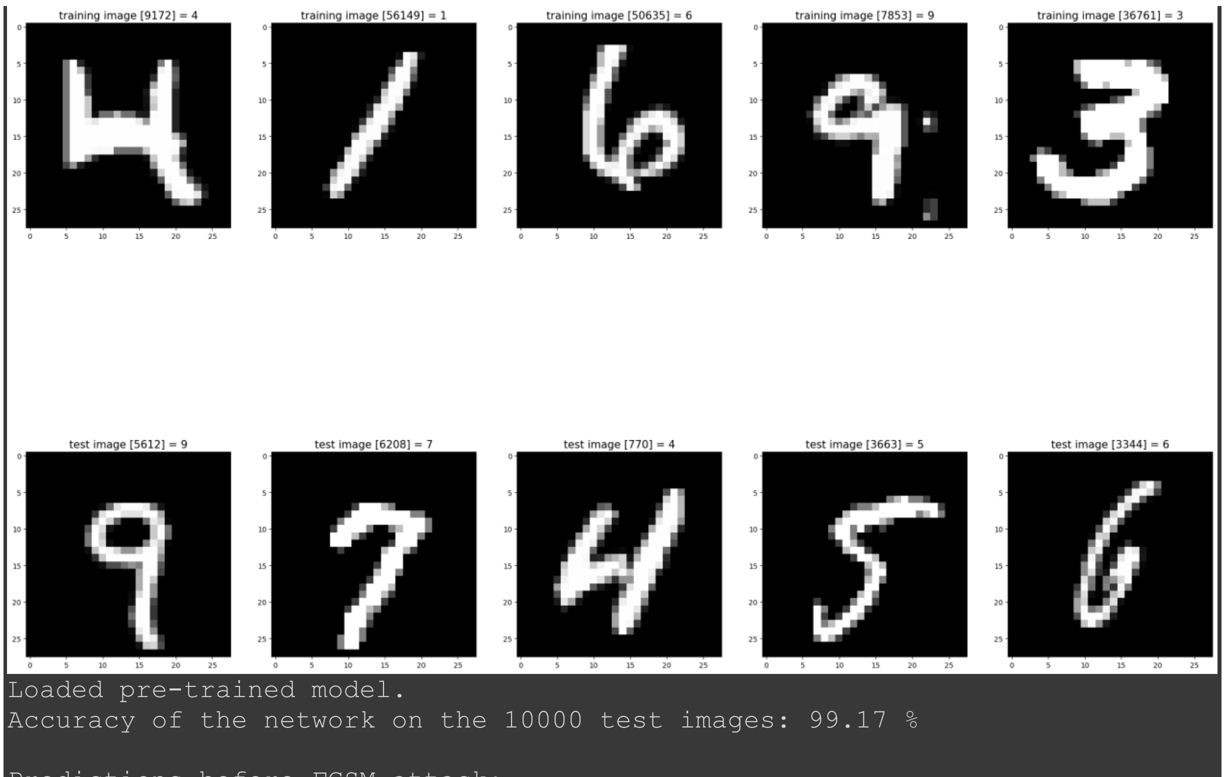
Training Process: The model is trained to produce similar logits for both clean and adversarial examples, enhancing its robustness against adversarial attacks.



Evaluation

The model's performance was evaluated on both clean and adversarial examples. We compared predictions before and after the FGSM attack and assessed the effectiveness of the defense mechanisms in recovering the correct labels.

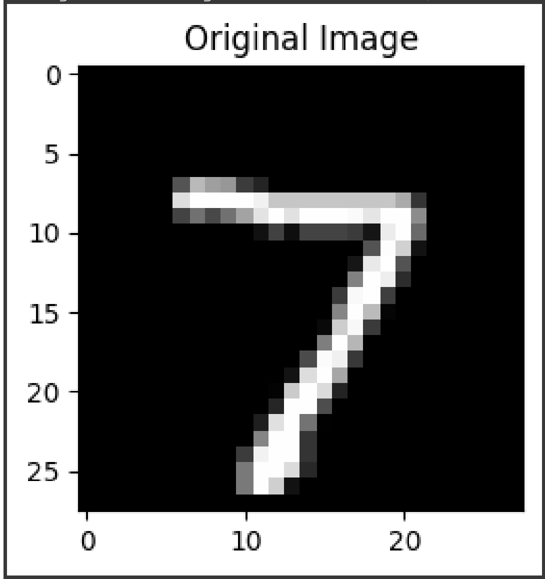
Results


training image [9172] = 4
training image [56149] = 1
training image [50635] = 6
training image [7853] = 9
training image [36761] = 3
test image [5612] = 9
test image [6208] = 7
test image [770] = 4
test image [3663] = 5
test image [3344] = 6

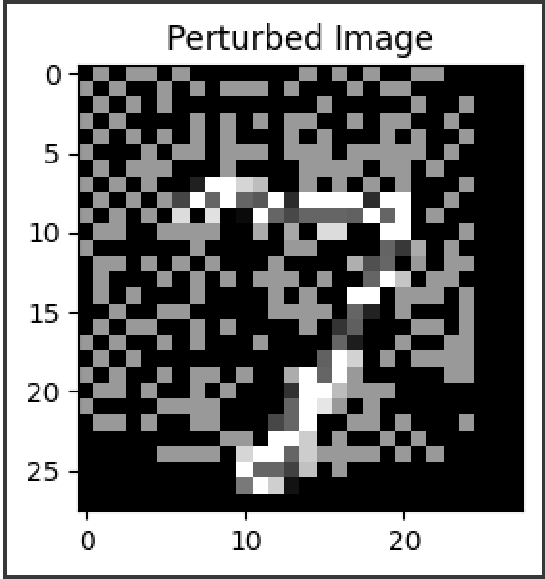
```
Loaded pre-trained model.
Accuracy of the network on the 10000 test images: 99.17 %

Predictions before FGSM attack:
Image 1: Original label: 7, Predicted label: 7
Image 2: Original label: 2, Predicted label: 2
Image 3: Original label: 1, Predicted label: 1
Image 4: Original label: 0, Predicted label: 0
Image 5: Original label: 4, Predicted label: 4
```

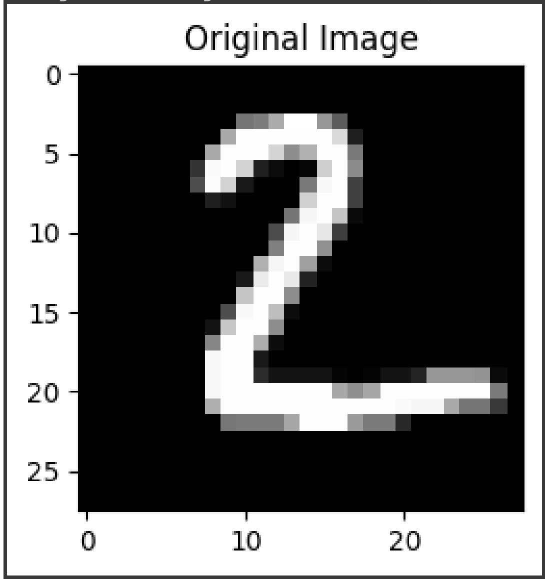
```
Predictions after FGSM attack:
Image 1: Original label: 7, Predicted label after FGSM attack: 8
```



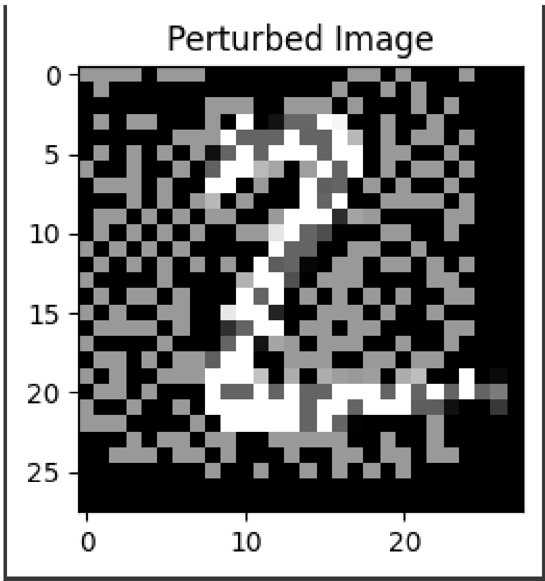
```
================================
```



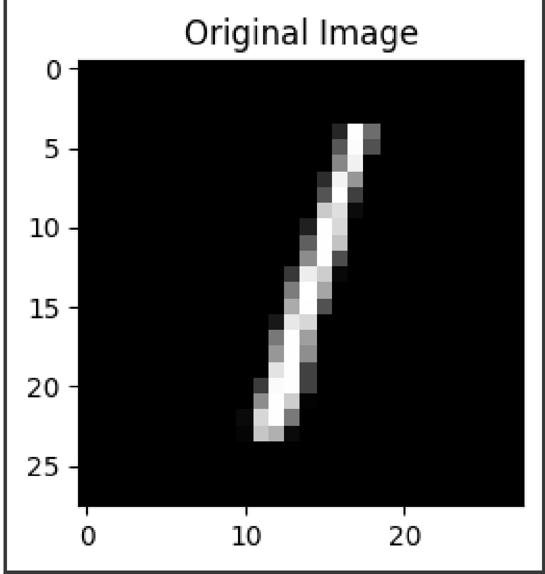
```
Image 2: Original label: 2, Predicted label after FGSM attack: 0
```



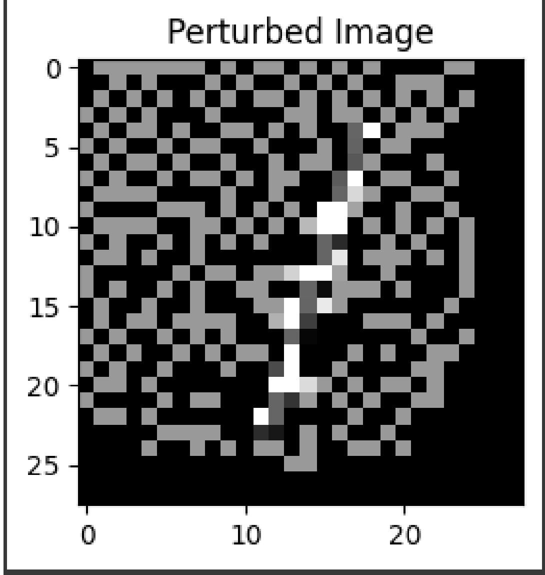
```
================================
```



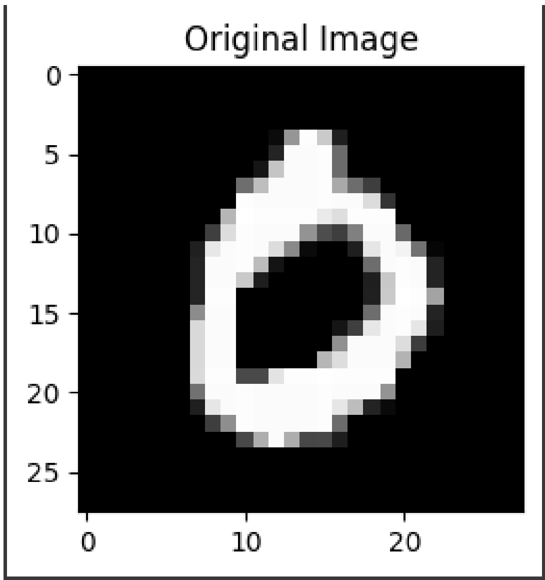
```
Image 3: Original label: 1, Predicted label after FGSM attack: 8
```



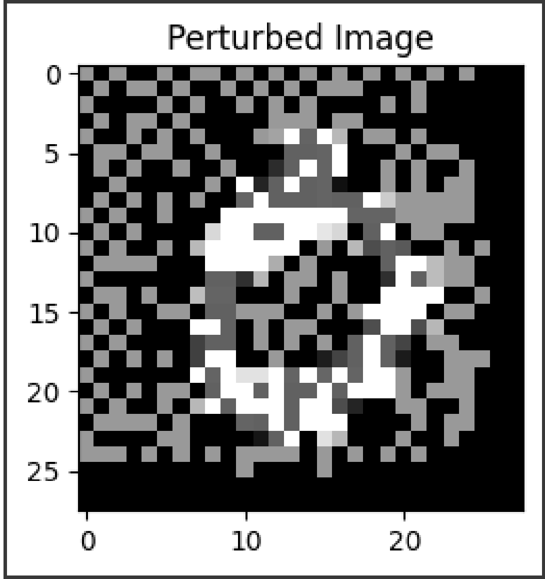
```
================================
```



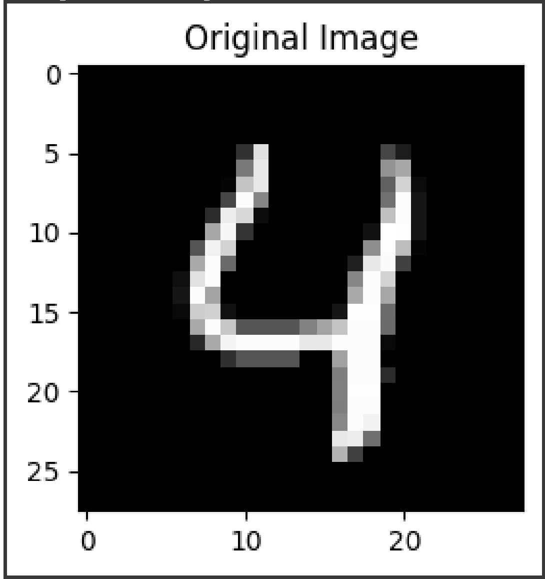
```
Image 4: Original label: 0, Predicted label after FGSM attack: 8
```



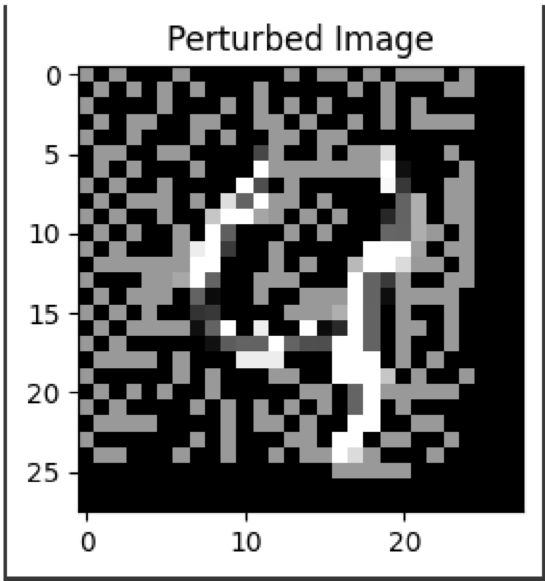
```
================================
```



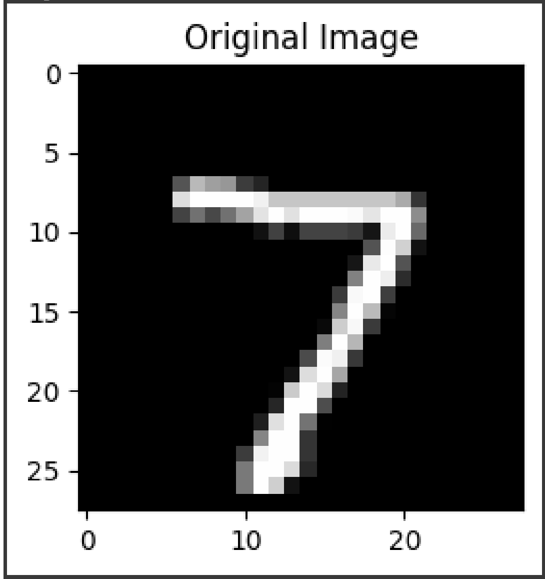
```
Image 5: Original label: 4, Predicted label after FGSM attack: 8
```



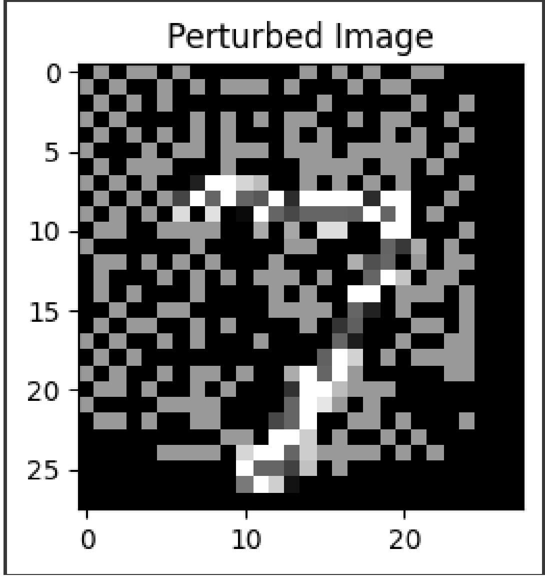
```
================================
```



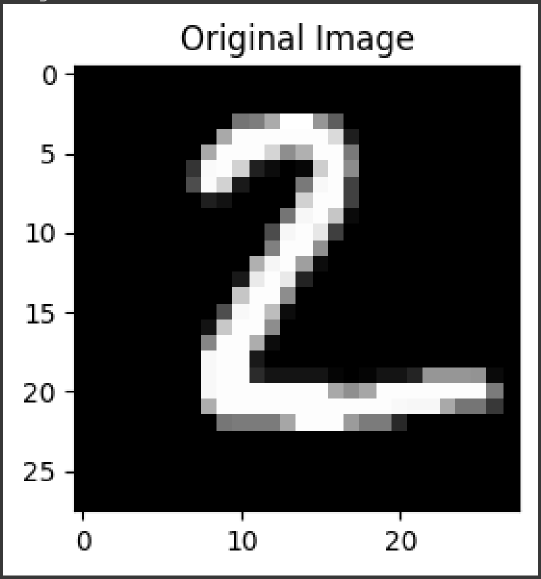
```
Predictions after FGSM attack and defense algorithms:
Image 1: Original label: 7, Corrected Predicted label after defense
algorithms: 7
```



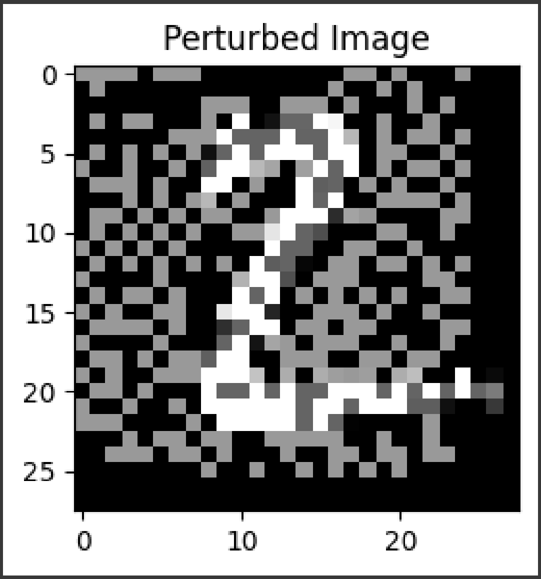
```
================================
```



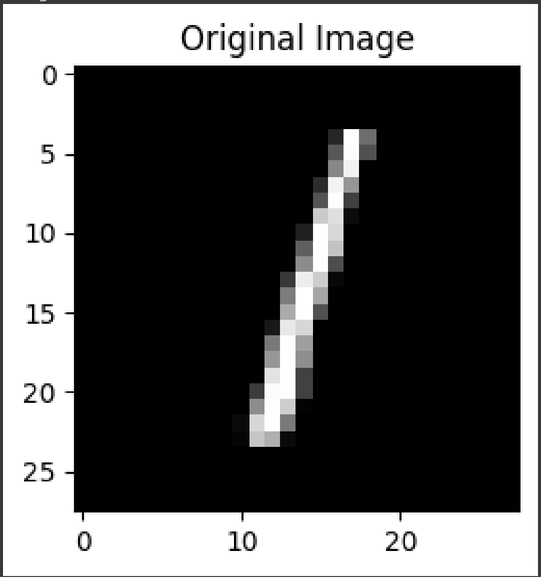
```
Image 2: Original label: 2, Corrected Predicted label after defense
algorithms: 2
```



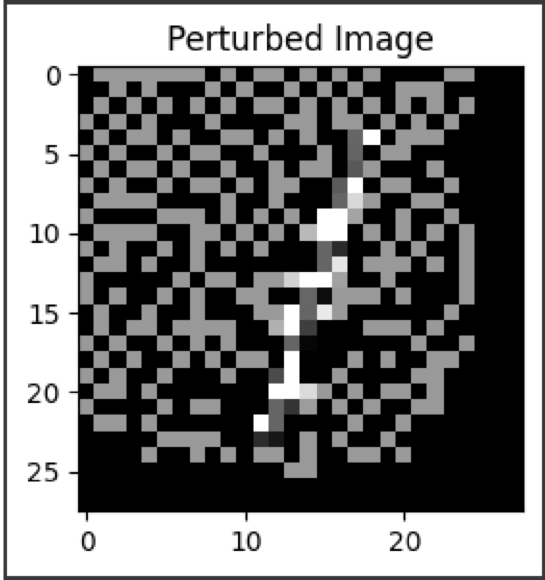
```
================================
```



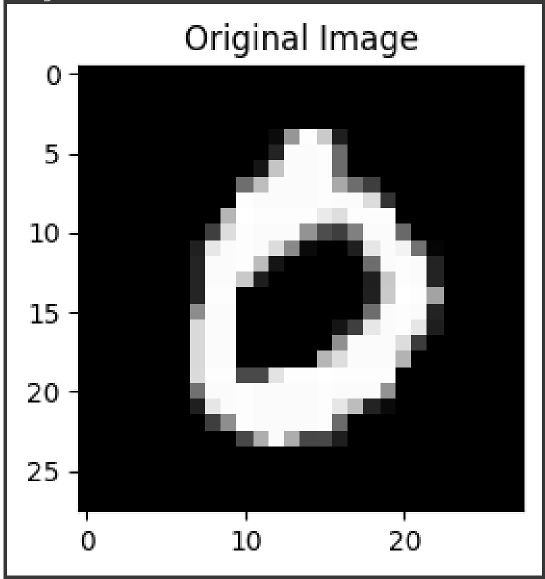
```
Image 3: Original label: 1, Corrected Predicted label after defense
algorithms: 1
```



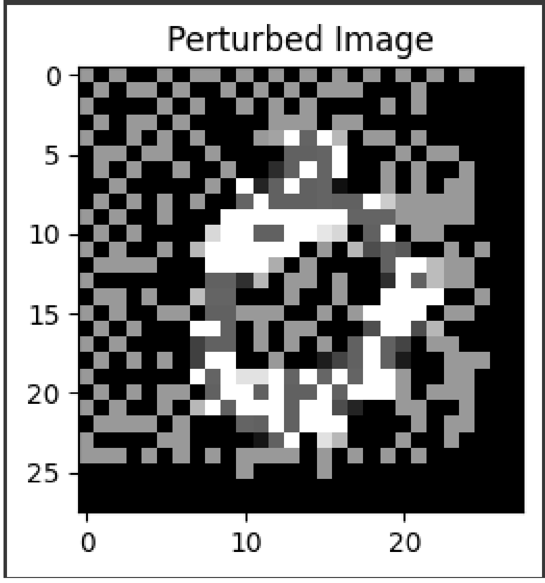
```
================================
```



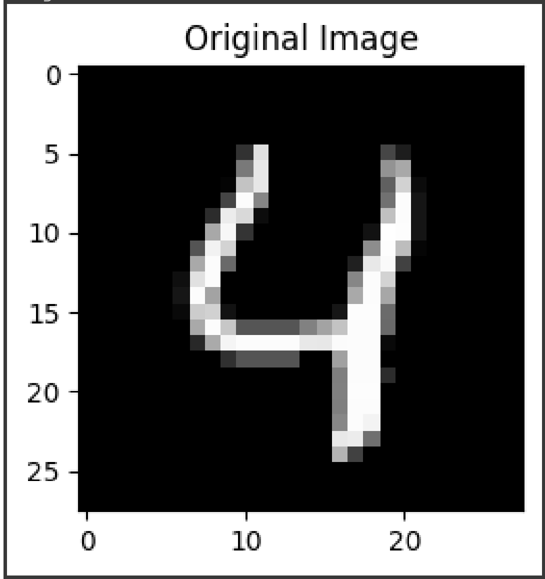
```
Image 4: Original label: 0, Corrected Predicted label after defense
algorithms: 0
```



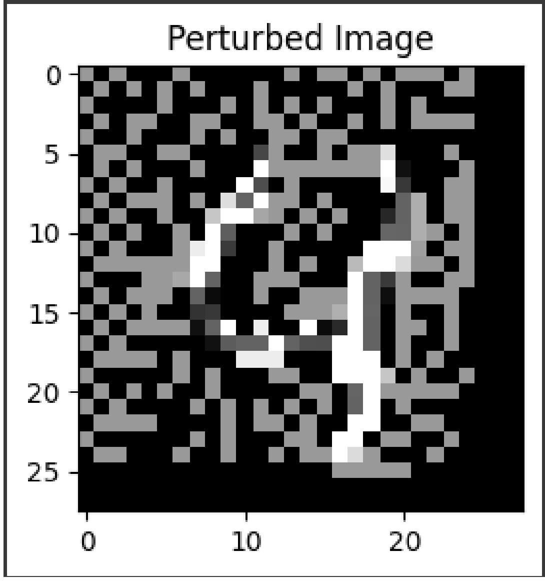
```
================================
```



```
Image 5 Original label: 4:, Corrected Predicted label after defense
algorithms: 4
```



```
================================
```



Accuracy on Clean Test Images: The base CNN model achieved an accuracy of approximately 99% on clean test images.

Effect of FGSM Attack: The FGSM attack significantly reduced the model's accuracy, demonstrating its vulnerability to adversarial perturbations.

Conclusion

The results demonstrate that the proposed defense mechanisms, especially the Dynamic Ensemble Model, significantly enhance the robustness of the CNN model against FGSM attacks. While the Stochastic Pruning Model also improves robustness, combining these strategies with Adversarial Logit Pairing during training further strengthens the model's defense capabilities. Future work could explore other types of adversarial attacks and additional defense strategies to build more resilient models.

REFERENCES

https://www.geeksforgeeks.org/introduction-convolution-neural-network/

https://meghakumar245.medium.com/cnn-model-architectures-with-their-strengths-and-weaknesses-4cf81fc49cc2

https://en.wikipedia.org/wiki/Adversarial_machine_learning

https://en.wikipedia.org/wiki/Logit

https://www.nature.com/articles/nrn2961

https://ieeexplore.ieee.org/abstract/document/8795523/

https://ieeexplore.ieee.org/abstract/document/9888103/

https://www.researchgate.net/publication/333609057_Dynamic_ensemble_models_to_predict_distributions_and_anthropogenic_risk_exposure_for_highly_mobile_species

https://www.researchgate.net/figure/Stochastic-pruning-eliminates-the-need-for-generating-failure-repair-histories_fig2_309005403