**CS550/DSL501: Machine Learning (2024–25–M)**

**Project Phase-II**

**Team Name: Model Mavericks**


| Team Members | |
|---|---|
| Arey Pragna Sri | - 12240230 |
| Matcha Jhansi Lakshmi | - 12241000 |
| Nannepaga Vanaja | - 12241110 |


**GitHub Link:** Link

# 1 Individual Contributions

## 1.1 Setup and Data Preparation

**Contributed by:** Pragna
**Purpose:** To import necessary libraries and load the MNIST dataset.
**Reshaping:** The images are reshaped to include a channel dimension (needed for CNNs), from (28, 28) to (28,28,1).
**Data Normalization:** This normalizes the pixel values of the MNIST images from 0-255 to 0-1.

## 1.2 Model Creation

**Contributed by:** Vanaja
**Purpose:** Defines the structure of the Convolutional Neural Network (CNN) used to classify MNIST digits.
**CNN Architecture:** The model consists of 2 convolutional layers with 32 and 64 filters, respectively, followed by max-pooling layers to down-sample the feature maps. A fully connected (Dense) layer with 128 units is added before the final output layer, which uses softmax activation for classification across the 10 possible digit classes.

## 1.3 Initial Training on Clean Data

**Contributed by:** Vanaja
**Purpose:** To train the CNN model on the original, unmodified MNIST dataset.
**Training Details:** The model is trained for 5 epochs with a batch size of 128. The data is split into training and validation sets, and the model's accuracy and loss are calculated on both the clean training and validation datasets.

## 1.4 PGD (Projected Gradient Descent) Adversarial Attack

**Contributed by:** Pragna
**Purpose:** To generate adversarial examples using PGD.

**PGD Process:** For each image, the gradient of the loss with respect to the image is computed. A small step ($\alpha$) is taken in the direction of the gradient to modify the image. The perturbed image is clipped to stay within a predefined range ($\epsilon$) from the original image and to ensure pixel values remain between 0 and 1. The process is repeated for a set number of iterations ($num\_iterations$) to create adversarial examples that fool the model.

### 1.5 Visualization

**Contributed by:** Jhansi

**Purpose:**To demonstrate the extent of perturbations applied to input data to mislead the model's predictions.
It provides a clear comparison between clean and adversarial examples, highlighting how minimal but targeted alterations in the input can cause the model to make incorrect predictions.

### 1.6 Adversarial Training

**Contributed by:** Jhansi
**Purpose:**Retraining the model on both clean and adversarial images to make it robust against adversarial attacks.
**Training Process:**Clean images and adversarial counterparts are combined to form a new training set. The model is retrained on this mixed dataset, which helps it learn to recognize both clean and adversarial inputs to improve its robustness.

### 1.7 Evaluation

**Contributed by:** Jhansi
**Purpose:**Evaluating the model's performance after adversarial training on both clean and adversarial test data.
**Evaluation on Clean Data:** The model is tested on unmodified images to check how well it performs on the original dataset after adversarial training.
**Evaluation on Adversarial Data:** The model is also evaluated on adversarial examples to measure how resistant it is to attacks after adversarial training.

## 2 Tasks and Milestones

### 2.1 List of Tasks and Milestones Achieved in This Phase

- Successfully built and trained a CNN model tailored for handwritten digit recognition with two convolutional layers, max-pooling, and fully connected layers.
- Generated adversarial examples using the PGD method with epsilon constraints to simulate real-world adversarial attacks.
- Combined adversarial and clean examples to perform adversarial training and improved the model's robustness against adversarial inputs.

## 2.2 Pending Tasks to be Addressed in the Final Phase

- **Hyperparameter Tuning:** Need to test different values for epsilon, alpha, and num_iterations to improve the effectiveness of adversarial training.
- **Advanced Adversarial Attack Techniques:** Involves implementing and testing other adversarial attack strategies (e.g., FGSM, CW attacks) to evaluate the model's robustness across multiple attack methods.
- **Model Evaluation Metrics:** To Use additional evaluation metrics (e.g., precision, recall, F1-score) to get a more comprehensive understanding of the model's performance.
- **Robustness Visualization:** Visualizing decision boundaries and perturbation effects for clean vs. adversarial examples.
- **Model Fine-Tuning:** Fine-tune the CNN architecture for better performance under adversarial attacks, potentially adding regularization techniques.
- **Website Development:** To create a website to showcase the digit recognition model, demonstrating its performance and robustness against adversarial attacks.