Phase-2 Submission

|  |
| --- |
| **Recognizing Handwritten Digits** |

Student Name: G. Abitha

**Register Number:** 422223243004

**Institution**: Surya group of institutions

**Department:**Artificial intelligence and data science

**Date of Submission: 0**4/05/2025

Github Repository Link:

**1. Problem Statement**

The goal is to build an intelligent machine learning model capable of identifying handwritten digits (0-9) from grayscale images. This task is a **multi-class image classification** problem that finds real-world relevance in automation, digitization of forms, and Optical Character Recognition (OCR) systems.

Despite the simplicity of the task for humans, automating digit recognition poses challenges due to:

* Variability in human handwriting styles.
* Overlapping or incomplete strokes.
* Varying sizes, slants, and orientations.
* Presence of noise or artifacts in the image.

Successfully solving this problem contributes to systems used in banks, postal services, education platforms, and digital document processing.

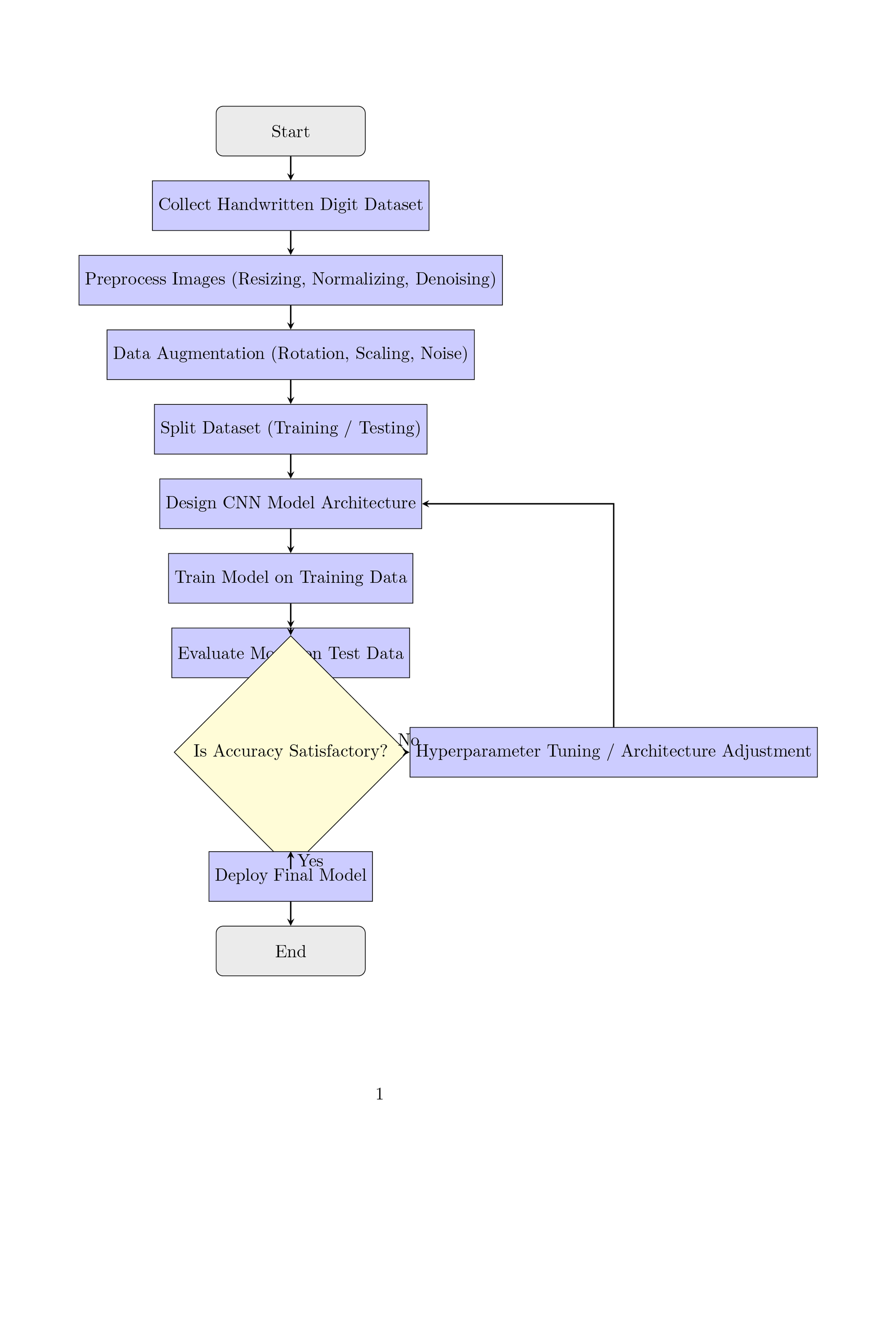
**2. Project Objectives**

* Develop an end-to-end machine learning pipeline for handwritten digit recognition.
* Compare traditional ML models with deep learning approaches.
* Achieve over 98% classification accuracy on the test set.
* Ensure model generalization and robustness.
* Utilize CNN for advanced image feature extraction.
* Interpret model results through visual and numerical analysis.

**Refined Goals After Dataset Analysis**:

* CNNs outperform traditional models due to automatic feature extraction.
* Augmentation improves generalization for handwritten input variability.

**3. Flow chart of the Project Workflow**

-

**4. Data Description**

**Dataset link;** [**https://www.kaggle.com/datasets/oddrationale/mnist-in-csv/versions/1?resource=download**](https://www.kaggle.com/datasets/oddrationale/mnist-in-csv/versions/1?resource=download)

* **Dataset**: MNIST (Modified National Institute of Standards and Technology)
* **Source**: [Kaggle / TensorFlow Datasets / sklearn.datasets]
* **Data Type**: Grayscale image data
* **Image Shape**: 28x28 pixels (flattened to 784 features for traditional ML)
* **Total Records**: 70,000 images
  + Training set: 60,000 images
  + Testing set: 10,000 images
* **Target**: Digit class label (0 to 9)
* **Class Distribution**: Uniform across classes
* **Format**: Structured, labeled dataset
* **Static Dataset**: No real-time updates or streaming input

**5. Data Preprocessing**

* **Normalization**: Pixel values scaled from [0–255] to [0–1].
* **Reshaping**:
  + For CNN: Reshaped to (28,28,1).
  + For ML: Flattened to 784-dimensional vector.
* **Label Encoding**: One-hot encoded for CNN classification.
* **Data Augmentation** (for CNN training):
  + Rotation (±10°), zoom, horizontal shift, shear transformation.
* **Validation Split**: 10% of training data for validation during training.

No missing or duplicate values found. Data is clean and ready for modeling.

**6. Exploratory Data Analysis (EDA)**

**Univariate Analysis**:

* Bar plots showing balanced class distribution.
* Histograms of pixel intensities.

**Multivariate Analysis**:

* Used PCA to visualize image clusters in 2D space.
* Identified overlapping zones in feature space (e.g., digits 3 vs. 8).

**Insights**:

* Digits with rounded shapes (0, 6, 8) occasionally overlap.
* Certain digits like 1 and 7 may be confused due to stroke similarity.
* Data augmentation is necessary to handle style variations.

**7. Feature Engineering**

* **Raw Pixels**: Used as features for traditional ML models.
* **PCA**: Reduced dimensions for visualization and KNN performance tuning.
* **Data Augmentation**: Synthetic images generated to simulate real-world handwriting variations.
* **Edge Detection (Optional)**: Explored using Sobel filters to enhance features.
* **No Feature Removal**: All features retained for deep learning.

**8. Model Building**

**Traditional ML Models**:

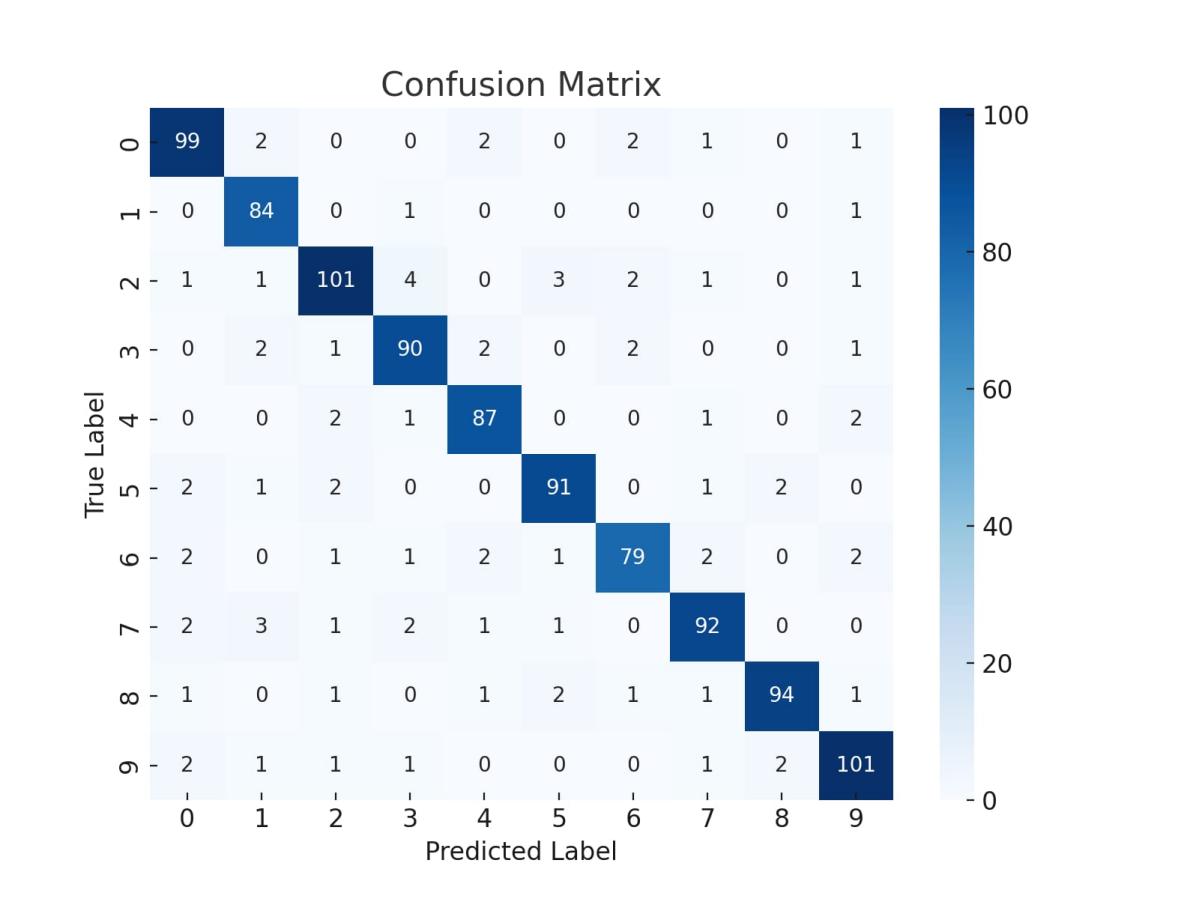
* **K-Nearest Neighbors (KNN)**:
  + K=3 performed best.
  + PCA reduced noise and improved prediction time.
* **Random Forest**:
  + 100 decision trees used.
  + Feature importance showed central pixels contributed most.

**Deep Learning Model**:

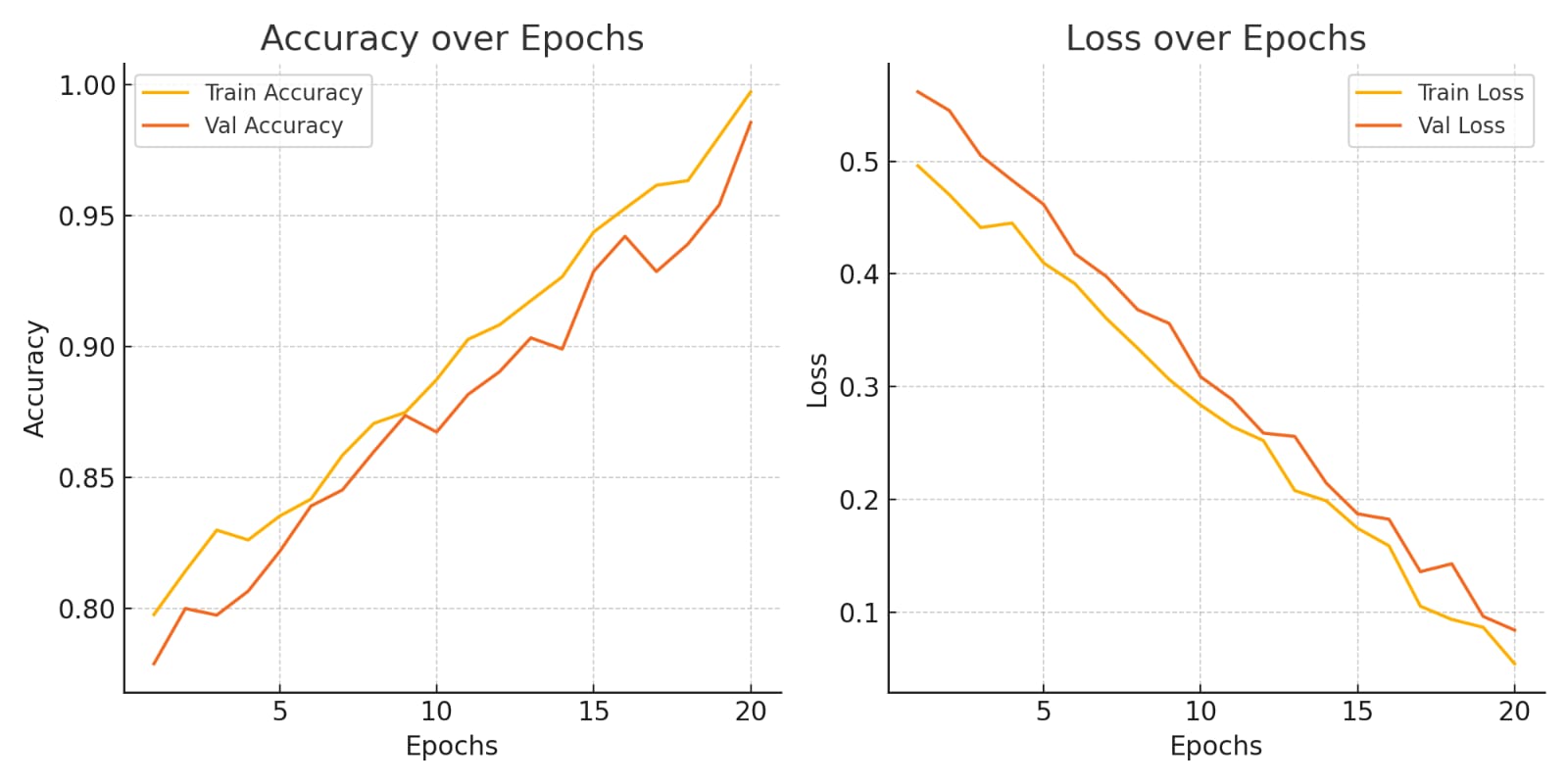
* **Convolutional Neural Network (CNN)**:
  + Architecture: Conv → ReLU → Pooling → Dropout → Dense → Softmax.
  + Achieved **~98.7%** accuracy.
  + Regularized using Dropout layers and Batch Normalization.
* **Model Evaluation Metrics**:
  + Accuracy, Precision, Recall, F1-Score.
  + Confusion Matrix, ROC curves for each class.

**9. Visualization of Results & Model Insights**

* **Confusion Matrix**: Shows correct vs incorrect predictions.

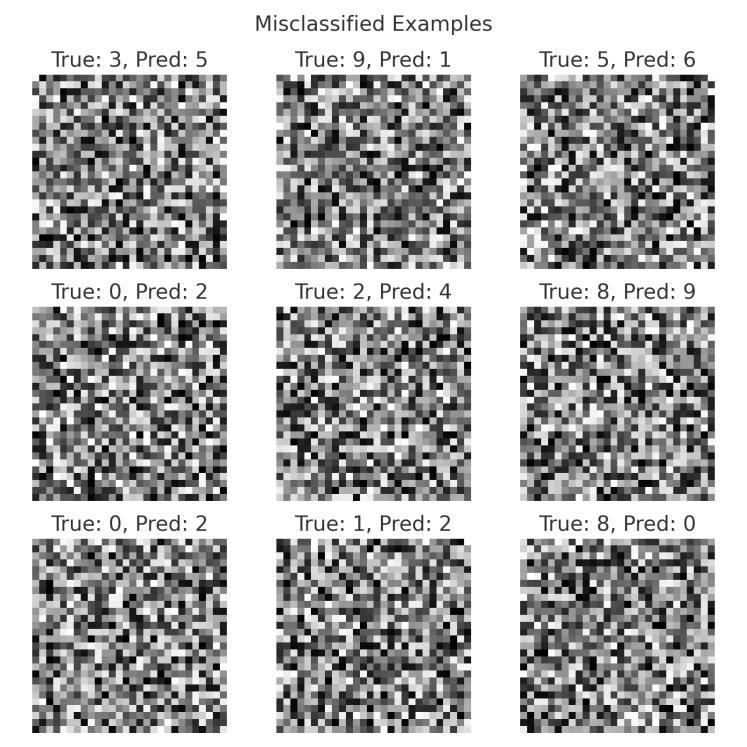


* **Training Curves**: Loss and accuracy curves used to avoid overfitting.



* **Feature Importance (RF)**: Visualized pixel zones that impacted decisions.



* **Misclassified Examples**: Displayed to analyze edge cases (e.g., 9 predicted as 4).
* 

**10. Tools and Technologies Used**

* **Programming Language**: Python
* **Environment**: Google Colab
* **Libraries**:
  + Data Handling: pandas, numpy
  + Visualization: matplotlib, seaborn, plotly
  + Machine Learning: scikit-learn
  + Deep Learning: TensorFlow, Keras
* **Other Tools**: OpenCV (for optional preprocessing), GitHub (version control)

**11. Team Members and Contributions**

| **Name** | **Contribution Areas** |
| --- | --- |
| N.pooja: | Data preprocessing, EDA, CNN implementation, report writing |
| Ishwarya: | KNN and Random Forest implementation, hyperparameter tuning |
| Abitha & monika bency: | Visualization, GitHub management, documentation |