STUDENT NAME: [KARTHIKA]

REGISTER NUMBER: [5119232O5022]

INSTITUTION: [PRIYADHARSHINI ENGINEERING COLLEGE]

DEPARTMENT: [B.TECH/IT]

DATE OF SUBMISSION: [05.05.2025]

GITHUB REPOSITORY LINK: [UPDATE THE PROJECT SOURCE CODE TO YOUR GITHUB REPOSITORY]

1. Problem Statement

In the digital era, converting handwritten inputs into machine-readable text is essential for automation and accessibility. This project addresses the challenge of recognizing handwritten digits (0–9) using deep learning techniques. It is a multi-class image classification problem.

Accurate digit recognition has broad applications such as automated data entry, postal code recognition, and form digitization. Leveraging deep learning, particularly Convolutional Neural Networks (CNNs), can significantly enhance recognition accuracy and processing efficiency, making AI systems smarter and more applicable in real-world scenarios.

2. Project Objectives

Implement a robust CNN model to accurately classify handwritten digits.

Achieve high classification performance on a standard dataset (MNIST).

Explore and apply best practices in data preprocessing and model evaluation.

Visualize model predictions and understand the importance of learned features.

Evaluate whether model generalization improves through techniques such as regularization and data augmentation.

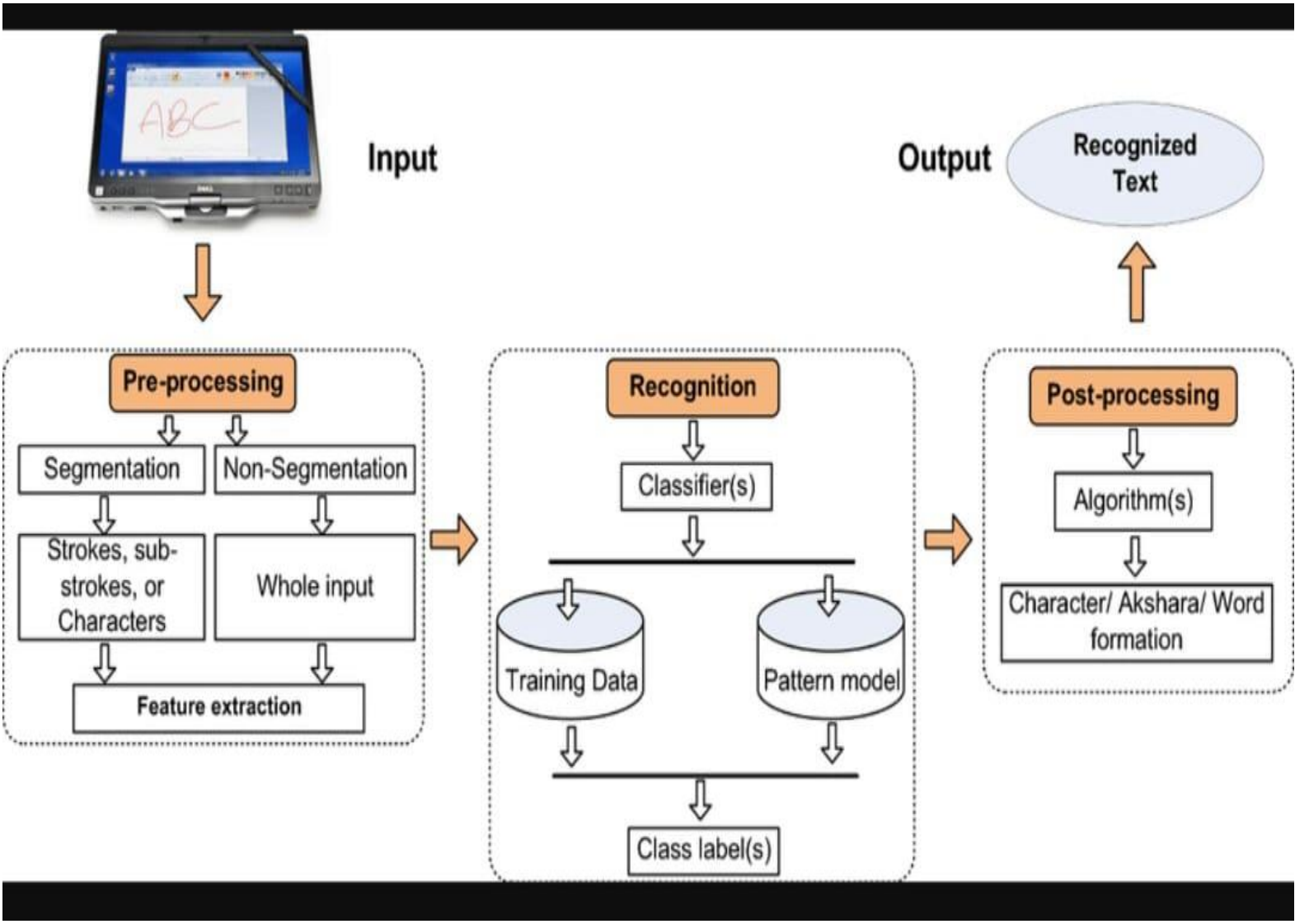
Traditional handwritten digit recognition system includes two stages: feature extraction and classification.

Most of them use shallow structures to deal with computing units and limited kinds of samples, such as Support Vector Machines (SVM) [14].

Faced with complex classification problems, the generalization ability and performance of SVM are insufficient when the samples have rich meaning

Recently, Convolutional Neural Network (CNN) [15] has been widely used in the field of computer vision because of its excellent performance and has made outstanding achievements in the accuracy of various machine learning tasks.

3. Flowchart of the Project Workflow



1. Input

User writes on a touchscreen or digital device (e.g., letters like 'ABC').

2. Pre-processing

- Segmentation: Breaks input into strokes or characters.

- Non-Segmentation: Uses whole input directly.

- Feature Extraction: Identifies key shapes and patterns from the input.

3. Recognition

- Classifiers use:

  - Training Data and

  - Pattern Models

- Output: Predicted Character Labels (e.g., A, B, C).

4. Post-processing

- Applies algorithms to:

  - Correct errors

  - Form meaningful words or syllables (like Aksharas).

5. Output

Final Recognized Text is displayed.

4. Data Description

The Convolutional Neural Network algorithm is a supervised deep learning algorithm used for both classification and regression.

Dataset: MNIST (Modified National Institute of Standards and Technology)

Source: Available from Kaggle or tensorflow.keras.datasets

Type: Image data (unstructured)

Size: 70,000 images (60,000 training, 10,000 testing); each 28x28 grayscale pixel

Target Variable: Digit label (0–9)

Static or Dynamic: Static

Format: Each image is a 28x28 matrix with pixel intensity values (0–255)

5. Data Preprocessing

The 'label' column (numbers) is encoded into a hot vector for each class and there are 10 classes (0-9).

The training data is segmented into two exclusives which are a train and validation sets.

Normalized pixel values to the [0, 1] range by dividing by 255.

Reshaped data to match CNN input requirements (28x28x1).

Converted target labels to categorical using one-hot encoding.

Verified absence of missing or duplicate entries.

Split dataset into training, validation, and testing sets (80/10/10).

Machine Learning provides several methods through which human efforts can be reduced in recognizing the manually written digits

6. Exploratory Data Analysis (EDA)

- Univariate Analysis:

Visualized digit distributions to ensure class balance.

Plotted example digit images to confirm clarity.

- Bivariate/Multivariate Analysis:

Correlation between pixel intensity patterns and label distribution.

- Insights:

The dataset is well-balanced across all classes.

Some digits (e.g., 3 and 5) have overlapping features, which may pose classification challenges.

7. Feature Engineering

Pixel intensities used directly as input features.

Applied padding and rotation augmentation to increase training data diversity. No manual feature extraction; instead relied on CNN to learn features hierarchically.

8. Model Building

- Implemented and compared:

Baseline CNN with 2 convolutional layers.

Improved CNN with dropout, batch normalization, and increased depth.
Evaluation Metrics: Accuracy, Precision, Recall, F1-score.
Achieved accuracy > 98% on validation data.

Dataset split: Training (80%), Validation (10%), Testing (10%).

9. Visualization of Results & Model Insights

- Confusion Matrix: Identified which digits were most frequently confused. Accuracy Curve: Showed training vs validation accuracy over epochs.

- Loss Curve: Helped monitor overfitting.

- Sample Predictions: Displayed correctly and incorrectly classified images.

- Feature Maps: Visualized CNN filters to understand learned patterns.

10. Tools and Technologies Used

- Programming Language: Python
- IDE/Notebook: Google Colab
- Libraries: TensorFlow, Keras, NumPy, Pandas, Matplotlib, Seaborn
- Visualization Tools: Matplotlib, Seaborn

- The proposed Handwritten digit recognition using Deep Learning has different modules for working its efficient functionality.

- These modules help the system to work efficient and also well in performance as well as in accuracy.

- In this model we will have to import the dependencies that are required for the creation of this system and such will be numpy, Pandas, Matplotlib, Train_test_split, Conv2D, Tensorflow, Statistics, keras.

- Only by importing these domains we will be able to train the data.

11. Team Members and Contributions

Name Contribution

[Member 1] Data Preprocessing

[Member 2] EDA

[Member 3] Model Building and Evaluation

[Member 4] Visualization and Documentation