**Student Name:** SIBI K

**Register Number:** 732323106047

**Institution:** SSM COLLEGE OF ENGINEERING

**Department:** ECE-2 YEAR

**Date of Submission:** 29.04.2025

**Github Repository Link:** <https://github.com/Sibik638/recognising-hand-written-digits-with-deep-learning-for-smarter-ai-application/tree/main>

### **Problem Statement**

The ability to accurately recognize handwritten digits is a fundamental task in computer vision and a crucial component for developing smarter AI applications. While seemingly simple for humans, it presents challenges for machines due to the variability in writing styles, stroke thickness, slant, and noise. This project addresses the problem of building a robust and accurate deep learning model capable of classifying images of handwritten digits (0-9). This falls under the category of a **multi-class classification** problem.

Solving this problem is significant because accurate digit recognition has numerous real-world applications, including:

* **Postal Automation:** Automating the sorting of mail based on zip codes.
* **Banking:** Processing handwritten digits on checks and deposit slips.
* **Data Entry:** Streamlining data input from handwritten forms.
* **Education:** Developing intelligent grading systems for handwritten assignments.
* **Accessibility:** Assisting individuals with visual impairments through digit recognition.

### **Project Objectives**

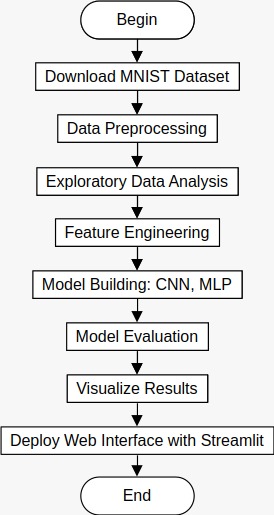
Building upon the initial understanding, the key technical objectives for this phase are:

* To implement and train at least two different deep learning models suitable for image classification (e.g., Convolutional Neural Networks - CNNs, Multi-Layer Perceptrons - MLPs).
* To achieve a high level of accuracy in classifying handwritten digits on a benchmark dataset.
* To explore and visualize the learned features within the deep learning models.
* To evaluate the performance of the trained models using appropriate classification metrics.
* To potentially investigate techniques for improving model robustness and generalization.

The primary goal remains to create a highly accurate digit recognition model. Data exploration in Phase-1 (if applicable) might have revealed the distribution of digits and the variability within the dataset, which will inform the model selection and training strategies.

### **Flowchart of the Project Workflow**

graph LR A[Load Handwritten Digit Dataset] --> B(Data Preprocessing); B --> C(Exploratory Data Analysis - Visualization of Digits); C --> D(Feature Scaling/Normalization); D --> E{Split Data (Train/Test/Validation)}; E -- Model 1 --> F1[Build Model Architecture (e.g., CNN)]; E -- Model 2 --> F2[Build Model Architecture (e.g., MLP)]; F1 --> G1(Train Model 1); F2 --> G2(Train Model 2); G1 --> H1[Evaluate Model 1 (Metrics & Visualization)]; G2 --> H2[Evaluate Model 2 (Metrics & Visualization)]; H1 -- Compare Performance --> I{Select Best Model}; H2 -- Compare Performance --> I; I --> J[Further Optimization (Optional)]; J --> K[Final Model Evaluation]; K --> L(Results & Insights); L --> M(Deployment Considerations);



### **4. Data Description**

* **Dataset Name and Origin:** MNIST (Modified National Institute of Standards and Technology) dataset is a common choice, readily available through libraries like TensorFlow or Keras. Alternatively, you might be using a custom dataset.
* **Type of Data:** Image data (grayscale images of handwritten digits).
* **Number of Records and Features:** The MNIST dataset contains 60,000 training images and 10,000 testing images. Each image is 28x28 pixels, resulting in 784 features (pixel intensity values).
* **Static or Dynamic Dataset:** Static.
* **Target Variable:** The digit itself (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) - a categorical variable.

### **5. Data Preprocessing**

* **Handle Missing Values:** Typically, the MNIST dataset is clean and doesn't contain missing values. If using a different dataset, you would document your approach to handling any missing data (e.g., checking for NaNs and deciding on a strategy if they exist).
* **Remove or Justify Duplicate Records:** Duplicate images are unlikely in standard datasets like MNIST. If identified in a custom dataset, you would justify whether to remove them or keep them (e.g., if they represent different instances of the same digit).
* **Detect and Treat Outliers:** While less common in this type of image data in the traditional sense, you might consider if there are unusually dark or light images that could be considered outliers in pixel intensity. You would document any steps taken.
* **Convert Data Types and Ensure Consistency:** Ensure pixel values are in a consistent numerical format (e.g., floating-point numbers between 0 and 1).
* **Encode Categorical Variables:** The target variable (digits 0-9) might need to be one-hot encoded for training certain deep learning models. For example, the digit '3' would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].
* **Normalize or Standardize Features:** Pixel values are often normalized to the range [0, 1] by dividing by the maximum pixel value (255 for grayscale images). This helps in faster and more stable training of deep learning models.
* **Documentation:** All preprocessing steps will be clearly documented in your code (with comments) and explained in this section of the report.

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

* **Univariate Analysis:**
* **Distribution of Features:** Visualize the distribution of pixel intensity values across the dataset using histograms.
* **Countplots:** Show the distribution of each digit (0-9) in the dataset to check for class imbalances.
* **Boxplots (Optional):** Could be used to visualize the range of pixel intensities for different digits.
* **Bivariate/Multivariate Analysis:**
* **Correlation Matrix (Less common for image data):** A correlation matrix between individual pixel values is generally not very informative for images.
* **Visualization of Sample Digits:** Display a grid of sample images for each digit to understand the visual variability.
* **Average Digit Representation:** Calculate and visualize the average pixel intensity for each digit to see general patterns.
* **Analysis of Relationship Between Features and the Target Variable:** Visualize average pixel intensity maps for each digit to see which pixel regions are most active for each class.
* **Insights Summary:**
* Note any class imbalances observed.
* Highlight the visual variability within each digit class.
* Identify any potential challenges for the model based on the EDA (e.g., digits that look very similar).

### 

### **7. Feature Engineering**

* **Create New Features:** For basic handwritten digit recognition using standard datasets like MNIST, explicit feature engineering is often minimal because CNNs are designed to automatically learn relevant features from the raw pixel data. However, you might consider:
* **Image Augmentation:** Generating slightly modified versions of the existing images (e.g., rotations, small shifts, scaling) to increase the training data size and improve model robustness. This can be considered a form of implicit feature engineering.
* **Combine or Split Columns:** Not directly applicable to raw pixel data in this context.
* **Use Techniques like Binning, Polynomial Features, Ratios:** Less common for direct image pixel values.
* **Apply Dimensionality Reduction (Optional):** Techniques like Principal Component Analysis (PCA) could be explored to reduce the dimensionality of the input images, although it might lead to a loss of information crucial for recognition. You would need to justify its use and evaluate its impact.
* **Justify each feature added or removed:** If you implement any feature engineering steps, clearly explain the reasoning and expected benefits.

### **8. Model Building**

* **Select and Implement at Least 2 Machine Learning Models:**
* **Model 1: Convolutional Neural Network (CNN):** Justification: CNNs are specifically designed for image data and excel at learning hierarchical features through convolutional layers, pooling layers, and activation functions. They are highly effective for tasks like image classification.
* **Model 2: Multi-Layer Perceptron (MLP) / Feedforward Neural Network:** Justification: As a baseline deep learning model, an MLP can provide a comparison to the more specialized CNN. While it doesn't inherently exploit the spatial structure of images, it can still learn complex mappings between input pixels and output classes.
* **Split data into training and testing sets (with stratification if needed):** Use a standard split (e.g., 80% training, 20% testing). Stratification might be useful to ensure that each digit is represented proportionally in both the training and testing sets, especially if there were class imbalances noted in the EDA. You might also include a separate validation set for hyperparameter tuning.
* **Train models and evaluate initial performance using appropriate metrics:**
* **For classification:** Accuracy, Precision, Recall, F1-score, and the Confusion Matrix will be crucial for evaluating the performance of your digit recognition models.

### 

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

* **Confusion Matrix:** Visualize the confusion matrix to see which digits are most often misclassified as others. This helps in understanding the model's weaknesses.
* **ROC Curve (Optional for multi-class):** While ROC curves are primarily for binary classification, you can generate one-vs-rest ROC curves for each digit class to assess the model's ability to distinguish each digit from the others.
* **Feature Importance Plot (For CNNs, visualize learned filters/feature maps):** Visualize the filters learned in the early convolutional layers of the CNN to gain insight into the low-level features the model is detecting (e.g., edges, curves). Also, visualize activation maps for specific input images to see which parts of the image activate different filters.
* **Residual Plots (Not directly applicable to classification):**
* **Include visual comparisons of model performance:** Bar charts comparing the accuracy, precision, recall, and F1-score of the different models.
* **Interpret top features influencing the outcome (For CNNs, relate learned features to digit characteristics):** Discuss how the learned filters and activation patterns might correspond to the characteristic strokes and shapes of different digits.
* **Clearly explain what each plot shows and how it supports conclusions about the model's performance and behavior.**

### **10. Tools and Technologies Used**

* **Programming Language:** Python
* **IDE/Notebook:** Google Colab, Jupyter Notebook, VS Code
* **Libraries:**
* **Data Manipulation:** pandas, numpy
* **Numerical Computation:** numpy
* **Visualization:** matplotlib, seaborn
* **Deep Learning Frameworks:** TensorFlow (with Keras), PyTorch
* **Model Evaluation:** scikit-learn

.

### 

### **11. Team Members and Contributions**

SIBI K: Data preprocessing, EDA.

SUJITH S: Model building, training.

SWETHA K: Feature engineering, visualization.

SHANMATHI H: Documentation, reporting.