**Phase-2 Submission**

**Student Name:**Jagadesh R

**Register Number:** 712523104028

**Institution:** PPG Institute of Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 09-05-2025

**Github Repository Link:** https://github.com/Jagadesh-RV/NM\_Jagadesh\_Recognizing\_Handwritten\_Digits.git

### **1. Problem Statement**

*The goal is to develop a system capable of* ***recognizing handwritten digits (0–9)*** *from grayscale images using machine learning and deep learning techniques. This is a* ***multi-class classification*** *problem involving image data.*

***Updated Understanding****: The MNIST dataset, while clean and well-labeled, still presents challenges in distinguishing similar digits (e.g., 3 vs. 5 or 4 vs. 9). The model should generalize well and be robust to varied handwriting styles.*

***Why It Matters****: Accurate digit recognition is vital in numerous real-world applications, including* ***postal mail sorting, form digitization, bank check processing****, and* ***digit-based authentication systems****. Automating this task reduces human error and speeds up processes in sectors like finance, logistics, and public administration*

### **2. Project Objectives**

***Technical Objectives****:*

* *Build and evaluate multiple models including traditional ML (e.g., k-NN, SVM) and deep learning (CNNs).*
* *Preprocess image data for optimal model training.*
* *Perform feature extraction or reduction (e.g., PCA) to improve training efficiency where appropriate.*
* *Tune hyperparameters to improve model accuracy and generalization.*
* *Compare performance using metrics:* ***accuracy, precision, recall, F1-score, confusion matrix****.*

***Updated Goals****:*

* *After exploring the data, the focus has shifted towards optimizing deep learning models, especially CNNs, given their superior performance on image data.*
* *Real-time prediction with user image input has become a key deployment goal.*

### **3. Flowchart of the Project Workflow**

### 

### 

### **4. Data Description**

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

### *Source: MNIST via GeeksforGeeks*

### *Type: Structured image data Format: 28×28 grayscale images*

### *Records: 60,000 training samples 10,000 test samples*

### *Target Variable: Digit label (0–9)*

### *Static/Dynamic: Static dataset*

### **5. Data Preprocessing**

* ***Missing Values****: None (clean dataset)*
* ***Duplicates****: Verified and removed if any*
* ***Normalization****: Pixel values scaled to range [0, 1]*

*python*

*X = X / 255.0*

* ***Reshaping for CNN****:*

*python*

*CopyEdit*

*X = X.reshape(-1, 28, 28, 1) # Adding channel dimension*

* ***Label Encoding****: Already in numeric format (0-9)*
* ***Train-Test Split****:*

*python*

*from sklearn.model\_selection import train\_test\_split*

*X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, stratify=y)*

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

#### ***Univariate Analysis***

* *Countplot of digit frequency: Shows even class distribution*
* *Image grid: Random samples for each digit (to visually inspect variation)*

#### ***Bivariate Analysis***

* *Heatmap of pixel correlation*
* *PCA scatterplots (optional) to explore separability*

#### ***Insights Summary***

* *Some digits have significant visual overlap (e.g., 5 and 3)*
* *CNNs are expected to learn spatial features better than classical ML models*

### **Primary Features**: Pixel values of images (784 features)

* **Transformations**:
  + **Flattening** for classical ML models
  + **Reshaping to 28×28×1** for CNN input
* **PCA** (optional for dimensionality reduction):

python

CopyEdit

from sklearn.decomposition import PCA

pca = PCA(n\_components=50)

X\_pca = pca.fit\_transform(X\_flat)

### **7. Feature Engineering**

### ***Primary Features****: Pixel values of images (784 features)*

* ***Transformations****:*
  + ***Flattening*** *for classical ML models*
  + ***Reshaping to 28×28×1*** *for CNN input*
* ***PCA*** *(optional for dimensionality reduction):*

*python*

*from sklearn.decomposition import PCA*

*pca = PCA(n\_components=50)*

*X\_pca = pca.fit\_transform(X\_flat)*

### **8. Model Building**

#### ***Models Used***

1. ***k-NN***
2. ***SVM***
3. ***CNN (Convolutional Neural Network)***

#### ***Why These Models****:*

* ***k-NN****: Simple baseline, effective for small-scale classification*
* ***SVM****: Performs well in high-dimensional spaces*
* ***CNN****: Best suited for spatial image data*

#### ***Metrics****:*

* *Accuracy*
* *Precision, Recall, F1-Score (macro and per-class)*
* *Confusion Matrix*

*python*

*from sklearn.metrics import classification\_report, confusion\_matrix*

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

* ***Confusion Matrix****: Highlights digit-wise errors*
* ***Accuracy Comparison****:*
  + *Bar chart comparing different models*
* ***Feature Importance****: For traditional ML (SVM weights)*
* ***CNN Filter Visualization*** *(optional): Visualize first-layer filters*

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

* ***Programming Language****: Python*
* ***IDE/Notebook****: Google Colab*
* ***Libraries****:*

1. *numpy, pandas*
2. *matplotlib, seaborn, plotly*
3. *scikit-learn, tensorflow.keras, OpenCV*

* ***Deployment Tools****: Flask or Streamlit*

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

|  |  |
| --- | --- |
| ***Member Name*** | ***Responsibilities*** |
| *Santhosh S* | *Data cleaning* |
| *Sarath Vel K V* | *EDA* |
| *Risikesh N* | *Feature engineering* |
| *Jagadesh R* | *Model development* |
| *Rajan N* | *Documentation and reporting* |