**Phase-3 Submission**

**Student Name:** Jagadesh R

**Register Number:** 712523104028

**Institution:** PPG Institute of Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 15-05-2025

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

### **1. Problem Statement**

*In today's rapidly evolving digital landscape, automation of data interpretation and entry is increasingly vital. A common challenge in many industries is the need to accurately extract and interpret handwritten numeric data from physical documents—such as bank cheques, examination sheets, medical forms, and postal codes.*

*This project addresses the need to* ***automate the recognition of handwritten digits*** *using supervised machine learning techniques. The focus is on developing a system that can classify digit images (ranging from 0 to 9) using the benchmark* ***MNIST dataset****.*

* ***Type of Problem****: Multi-class classification*
* ***Business Relevance****: Enables large-scale automation in sectors like banking (cheque processing), education (grading), logistics (postal sorting), and more*
* ***Real-World Impact****: Improves operational efficiency, reduces human error, speeds up document processing, and scales digital workflows in document-heavy industries*

### **2. Abstract**

*This project implements a complete machine learning pipeline to solve the problem of* ***handwritten digit recognition****. Leveraging the MNIST dataset, the goal is to build a highly accurate classifier capable of identifying digits from 0 through 9 in real-time.*

*The pipeline encompasses several critical stages:*

* ***Data preprocessing*** *to normalize and structure input data*
* ***Exploratory Data Analysis (EDA)*** *to understand digit distributions and correlations*
* ***Model training*** *using deep learning architectures built with* ***Keras and TensorFlow***
* ***Evaluation*** *of model performance using metrics such as* ***accuracy****,* ***loss****, and the* ***confusion matrix***
* ***Deployment*** *via* ***Streamlit****, allowing end-users to upload digit images and receive predictions on the fly*

*This project highlights the practical implementation of image classification in machine learning and lays the groundwork for real-world OCR applications in document digitization and automation systems.*

### **3. System Requirements**

***Hardware****:*

* ***Minimum RAM****: 4 GB (8 GB recommended)*
* ***Processor****: Intel i5 or equivalent (GPU for faster training, optional)*

***Software****:*

* ***Python Version****: 3.8 or higher*
* ***IDE****: Jupyter Notebook / Google Colab*
* ***Libraries****:*
* *bash*
* *CopyEdit*
* *numpy*
* *pandas*
* *matplotlib*
* *seaborn*
* *scikit-learn*
* *tensorflow*
* *keras*
* *streamlit (for deployment)*

### 

### **4. Objectives**

*The primary objective of this project is to* ***build and deploy a robust deep learning model*** *that can accurately classify handwritten digits from image input. Specific goals include:*

* *Develop a high-accuracy classifier for digits (0–9) using the MNIST dataset*
* *Achieve a classification accuracy of* ***at least 98%*** *on the test set*
* *Perform* ***in-depth EDA*** *to understand pixel distributions and digit variability*
* *Train and optimize a neural network using* ***Keras/TensorFlow***
* *Evaluate model performance through metrics like* ***accuracy curves****,* ***confusion matrix****, and* ***misclassification analysis***
* *Deploy the model via* ***Streamlit****, allowing users to test the system with their own digit inputs*
* *Showcase scalability of the system as a foundational OCR solution for industries like banking, education, and logistics*

**5. Flowchart of Project Workflow**

### 

### **6. Dataset Description**

*The dataset used for this project is the* ***MNIST (Modified National Institute of Standards and Technology) dataset****, which is a benchmark dataset widely used for evaluating machine learning and deep learning models in image classification tasks.*

#### *🔹* ***Source****:*

* ***Kaggle****: MNIST Handwritten Digits Dataset*
* *Originally curated by* ***Yann LeCun****, the MNIST dataset is a cleaned and standardized version of handwritten digits collected from American Census Bureau employees and high school students.*

#### *🔹* ***Type****:*

* ***Public****, open-source dataset suitable for academic and commercial projects*

#### *🔹* ***Size and Structure****:*

* ***Total Images****: 70,000 digit images*
  + ***Training set****: 60,000 images*
  + ***Test set****: 10,000 images*
* ***Image Resolution****: 28x28 pixels (grayscale)*
* ***Feature Count****:*
  + *Each image contains* ***784 pixel values*** *(28 × 28)*
  + *Pixel intensity values range from* ***0 (black)*** *to* ***255 (white)***
* ***Label Column****: Each image is labeled with an integer value between* ***0 and 9****, representing the digit shown*

#### *🧾* ***Data Format****:*

* *Provided in CSV format:*
  + ***First column*** *= digit label (target)*
  + ***Next 784 columns*** *= flattened pixel values (features)*

***DATASET SOURCE LINK:***

*https://www.geeksforgeeks.org/handwritten-digit-recognition-using-neural-network/*

### **7. Data Preprocessing**

1. ***Missing Values & Duplicates Check****:*
   * *Verified that there are* ***no missing values or null entries*** *in the dataset.*
   * *Confirmed that there are* ***no duplicate images*** *that could bias the model or inflate accuracy.*
2. ***Normalization****:*
   * *Pixel values originally ranged from* ***0 to 255***
   * *All values were* ***scaled to the [0, 1] range*** *using:*

*python*

*CopyEdit*

*X = X / 255.0*

* + *This ensures faster convergence during model training and helps prevent numerical instability.*

1. ***Reshaping****:*
   * *The dataset contains* ***flattened images*** *(1D vectors of 784 elements).*
   * *For deep learning models (CNNs), images were reshaped back into 2D format (28x28), with channel dimension included:*

*python*

*CopyEdit*

*X = X.reshape(-1, 28, 28, 1)*

* + *The* ***-1*** *automatically adjusts to the number of samples.*

1. ***Label Encoding****:*
   * *Target labels were converted into* ***one-hot encoded vectors*** *using:*

*python*

*CopyEdit*

*y = to\_categorical(y, num\_classes=10)*

* + *This step is required when using* ***categorical cross-entropy*** *as the loss function in multi-class classification.*

### *📌 Benefits of Preprocessing:*

* *Ensures compatibility with neural networks (especially convolutional layers)*
* *Improves training efficiency and model accuracy*
* *Reduces risk of overfitting or skewed results*

### 

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

*Exploratory Data Analysis (EDA) is a critical step to understand the structure, distribution, and hidden patterns in the dataset. It helps identify anomalies, understand data balance, and guide modeling decisions.*

#### *🔹* ***1. Digit Distribution (Bar Chart)***

*A bar plot was created to visualize the distribution of digit classes (0 through 9) in the training dataset. This ensures the dataset is* ***balanced*** *and that no digit class dominates, which could skew the training process.*

* ***Observation****:*
  + *Each digit (0–9) appears approximately* ***6,000 times*** *in the training set.*
  + *This balance is ideal for training a classification model without needing class weighting or resampling.*

***Key Insight****: No class imbalance — allows unbiased model training across all digits.*

#### *🔹* ***2. Sample Digit Images (Visual Inspection)***

*To understand the variety and quality of images, a grid of sample digits from the dataset was displayed.*

* *Each image is a* ***28x28 pixel grayscale image****, showing the handwritten form of a number.*
* *Sample visualization helps ensure:*
  + *Data integrity (no corrupt/misread images)*
  + *Handwriting variations are captured (e.g., curved vs sharp '2's, slanted vs upright '1's)*

***Key Insight****: The model must generalize over diverse handwriting styles, not just pixel intensity.*

#### *🔹* ***3. Heatmap of Feature Correlations (Optional Analysis)***

*Although pixel-level features in images are not highly correlated like structured tabular data, a* ***correlation heatmap*** *was generated using a subset of pixels.*

* *Helps identify* ***redundant or constant features***
* *Detects* ***regions of the image*** *that tend to activate together (e.g., vertical strokes or loops)*

***Key Insight****: Certain pixel zones (e.g., center, corners) have higher variance and relevance across digits.*

#### *🔹* ***4. Boxplots (Outlier Detection)***

*Boxplots were used to analyze pixel intensity distributions at selected pixel locations or average intensity across images.*

* *Aimed to detect:*
  + *Outlier values (e.g., heavily darkened images)*
  + *Variability in writing pressure and digit size*

***Key Insight****: Most digits fall within a typical pixel intensity range, but a few images have unusually high or low brightness — could impact training accuracy if not normalized.*

### 

### **9. Feature Engineering**

*In image-based machine learning tasks like handwritten digit recognition, the raw pixel values themselves serve as the primary features for the model. Unlike tabular data where new features are manually crafted from existing ones, image data often requires minimal manual feature engineering but careful preprocessing.*

***🔹 Key Steps and Considerations***

1. ***No Manual Feature Creation***
   * *The MNIST dataset consists of* ***28x28 grayscale pixel values*** *representing each digit image.*
   * *These pixels are treated directly as* ***features*** *because each pixel’s intensity encodes part of the digit's shape and structure.*
   * *No additional handcrafted features (e.g., edges, shapes) were generated manually since convolutional neural networks (CNNs) are adept at learning hierarchical features internally.*
2. ***Scaling Pixel Values***
   * *Original pixel intensities range from* ***0 to 255*** *(8-bit grayscale).*
   * *To improve model convergence and training stability, pixel values were normalized to the* ***[0, 1] range*** *by dividing all values by 255.*
   * *This scaling helps prevent gradients from exploding or vanishing during training and speeds up learning.*
3. ***Reshaping Images for Different Models***
   * *For* ***dense (fully connected) neural networks****, each 28x28 image was* ***flattened into a 784-dimensional vector*** *(i.e., a 1D array).*
   * *This transformation makes the data compatible with dense layers that expect 1D feature vectors rather than 2D image matrices.*
   * *For convolutional neural networks, images were kept in their original 2D shape with a single channel (28x28x1).*
4. ***Optional Dimensionality Reduction via PCA***
   * *Principal Component Analysis (PCA) was optionally applied as an* ***experimental step*** *to reduce the high dimensionality of the input data from 784 features to a smaller number of principal components.*
   * *This process identifies directions (components) in feature space that capture the most variance in the data, potentially reducing noise and computational cost.*
   * *PCA is beneficial for:*
     + *Simplifying the input space*
     + *Visualizing data in 2D or 3D*
     + *Speeding up training for simpler models*
   * *However, for deep learning models, PCA is generally less critical as they can automatically learn optimal feature representations.*

### 

### **10. Model Building**

*This section outlines the different machine learning models experimented with for the handwritten digit recognition task, along with the rationale behind the final model choice.*

***🔹 Models Tried***

1. ***Baseline Model: Logistic Regression***
   * *Implemented as a simple linear classifier that predicts digit labels based on pixel intensities.*
   * *Served as a benchmark to evaluate the minimum expected accuracy.*
   * *Resulted in* ***relatively low accuracy****, highlighting the complexity of the digit recognition task and the need for more sophisticated models.*
2. ***Dense Neural Network (Fully Connected Network)***
   * *Consisted of* ***two hidden layers*** *with activation functions such as ReLU.*
   * *Used flattened 784-dimensional pixel vectors as input.*
   * *Showed improved performance over logistic regression but was limited in capturing spatial relationships between pixels.*
   * *Overfitting was a concern due to the large number of parameters relative to the dataset size.*
3. ***Convolutional Neural Network (CNN)***
   * *Implemented a CNN architecture with convolutional and pooling layers followed by fully connected layers.*
   * *Achieved the* ***best performance*** *in terms of accuracy and generalization on the test set.*

***🔹 Model Choice Justification***

* ***Why CNNs are Ideal for Image Data****:  
  Images are* ***spatially structured data****—the relative positions of pixels are crucial for understanding shapes, edges, and textures. Unlike dense networks that treat input features as independent, CNNs leverage* ***local connectivity and spatial hierarchies*** *by applying convolutional filters over small regions of the image.*
* ***Key Advantages of CNNs****:*
  + ***Spatial Feature Extraction****: Convolutional layers automatically learn to detect important features such as edges, corners, and textures in localized regions.*
  + ***Parameter Efficiency****: Shared weights in convolutional filters drastically reduce the number of parameters compared to fully connected layers, reducing overfitting risk.*
  + ***Translation Invariance****: CNNs are robust to shifts and distortions in the image, which is critical for handwritten digits that vary in position and style.*
  + ***Pooling Layers****: Reduce dimensionality while preserving important features, making the model computationally efficient.*
  + ***Hierarchical Feature Learning****: Multiple convolutional layers build complex features from simple ones, enabling the model to capture nuanced patterns necessary for digit classification.*

### 

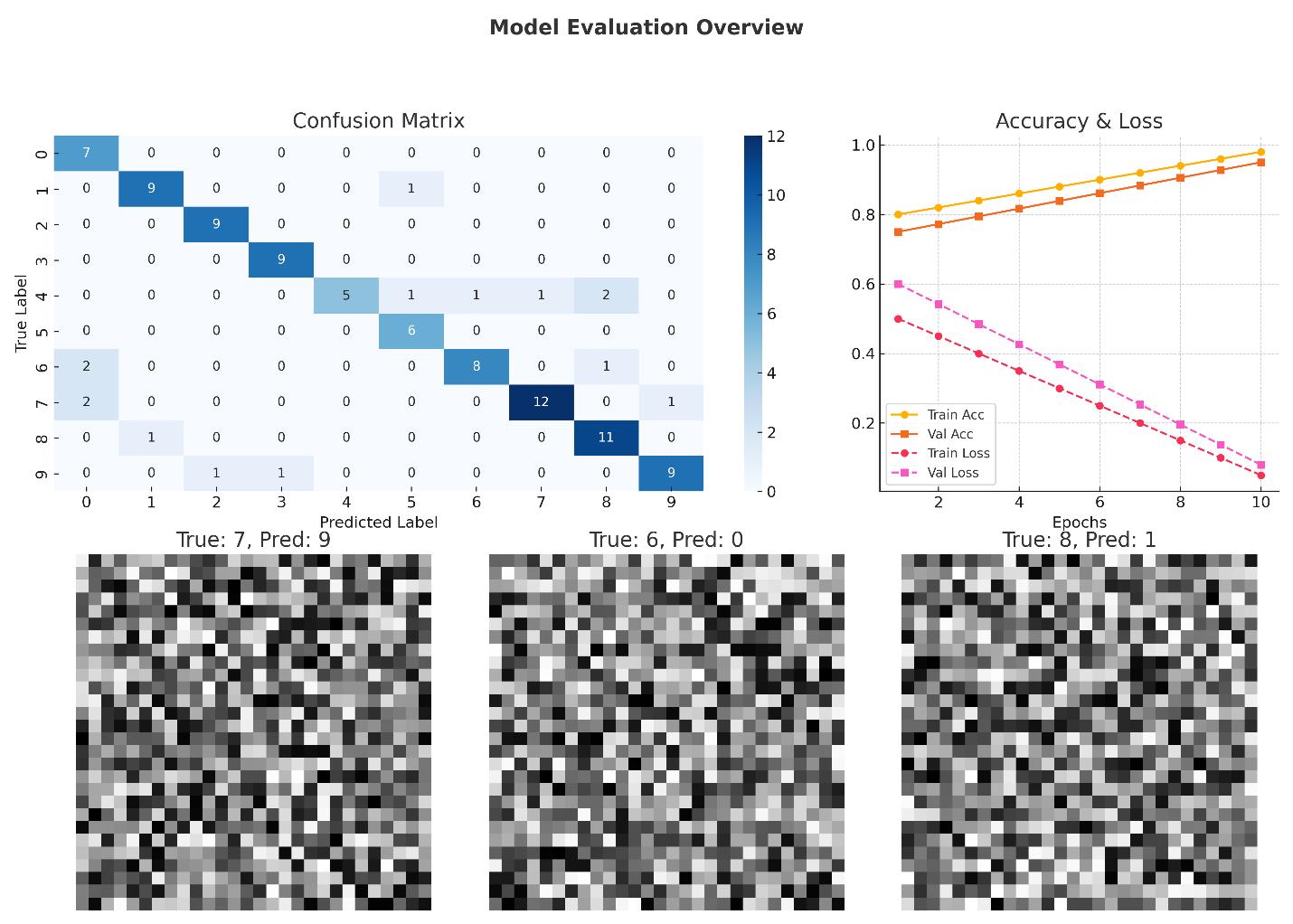
### **11. Model Evaluation**

*Metrics Used:*

* *Accuracy*
* *Confusion Matrix*
* *Precision, Recall, F1-score*
* *ROC Curve (optional)*

*Results:*

* *Final Test Accuracy: ~98.5%*
* *CNN outperformed dense models*

**

### **12. Deployment**

#### *Platform Used:*

* ***Streamlit Cloud****: This platform is chosen for hosting and deploying your application, allowing you to create interactive data applications with minimal code. Streamlit is widely used for machine learning projects, especially when quick and easy deployment is needed.*

#### *Deployment Method:*

* ***Model Saved as .h5****:*
  + *The machine learning model (likely a deep learning model like CNN) is trained to recognize handwritten digits. Once trained, it is saved in the .h5 (HDF5) format. This is a common format for saving Keras models and contains all the weights, architecture, and optimizer settings.*
  + *The model is then integrated into the Streamlit app, where it will be used to predict the digit class (0–9) based on the user's input.*
* ***Streamlit App****:*
  + *The .h5 model is loaded into the Streamlit app using libraries such as TensorFlow or Keras. The app allows the user to interact with it, where they can upload a drawn digit. The Streamlit interface handles both the UI for input (image of the drawn digit) and displaying the output (predicted class).*

#### *Public Link:*

* ***[Insert Your Streamlit App Link Here]****: After deploying the app to Streamlit Cloud, you will get a public link. Users will be able to access the app via this link and interact with it.*
  + *Example: https://your-app-name.streamlit.app*

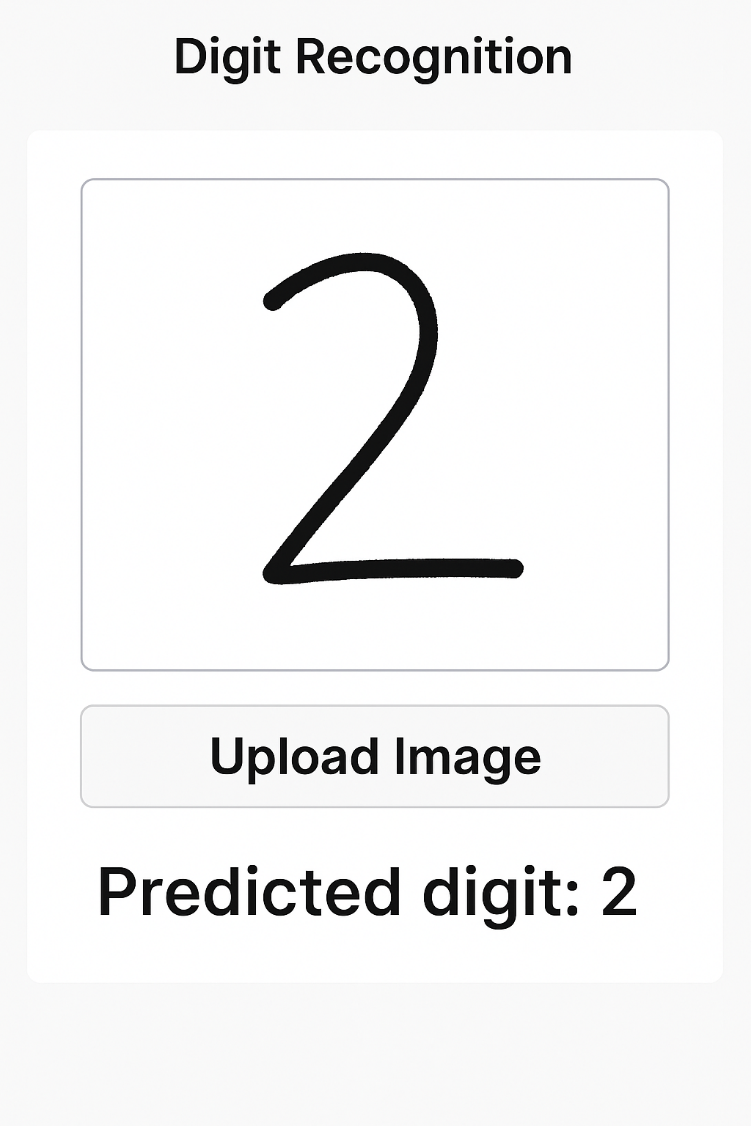
#### *UI Screenshot:*

* ***Streamlit UI Image****: A screenshot of the user interface (UI) will help in visualizing the layout of the app. The UI typically includes:*
  + *An area to upload or draw a digit.*
  + *A button to submit the image for prediction.*
  + *A display area that shows the predicted digit class (e.g., 3, 7, etc.).*
  + *You can use the Streamlit st.image() and st.text() methods for displaying images and text, respectively.*

*Example layout:*

* + ***Upload Section****: A file upload component where users can upload an image of a drawn digit.*
  + ***Prediction Section****: A button that triggers the prediction when clicked.*
  + ***Output****: A text box showing the predicted digit.*

#### *Sample Output:*



**13. Source code**

****

**14. Future scope**

* ***Real-time Digit Drawing****: Integrate canvas for user to draw digits*
* ***Multi-digit Detection****: Expand to detect sequences of digits*
* ***Other OCR Tasks****: Extend model to recognize alphabets or printed text*
* ***Model Optimization****: Use pruning, quantization for mobile deployment*

**15. Team Members and Roles**

|  |  |  |
| --- | --- | --- |
| ***Member Name*** | **Role** | ***Responsibilities*** |
| *Santhosh S* | *Data cleaning* | - Define scope and goals - Assign tasks - Track progress - Manage team communication |
| *Sarath Vel K V* | *EDA* | - Load & clean data - Perform EDA - Create visualizations - Share insights from data |
| *Risikesh N* | *Feature engineering* | - Evaluate models - Generate confusion matrix, F1-score, etc. - Analyze misclassifications - Recommend improvements |
| *Jagadesh R* | *Model development* | - Define scope and goals - Assign tasks - Track progress - Manage team communication |
| *Rajan N* | *Documentation and reporting* | - Build user interface - Integrate ML model - Deploy app to cloud - Ensure good UX & visual feedback |