Phase-2 Submission

Student Name: Saranselvan P

Register Number:712523205052

Institution:PPG Institute of Technology

Department:B.Tech Information Technology

Date of Submission:03/05/2025

Github Repository Link:

GitHub - saranp2005/Nm_saran_ds

Project Tittle: Recognizing handwritten digits with deep learning for smarter Al applications

1.Problem Statement

The goal of this project is to develop a deep learning-based model that can accurately recognize and classify handwritten digits (0–9) from images. Handwritten digit recognition is a classic problem in computer vision and pattern recognition with real-world applications such as postal mail sorting, bank check processing, and digital form automation. By leveraging convolutional neural networks (CNNs), the model will learn to identify spatial hierarchies in pixel data to achieve high classification accuracy on benchmark datasets like MNIST.

- ***Foundational AI Task****: It serves as a benchmark problem in the field of computer vision and deep learning, helping evaluate new algorithms and architectures.*
- ***Real-World Applications****:*
    - ***Banking****: Automated check processing through digit recognition.*

- ***Postal Services****: ZIP code and address interpretation from handwritten text.*
- ***Forms and Surveys****: Digitizing handwritten forms for data entry and analysis.*

- ***Technological Relevance****: The ability to correctly interpret human handwriting is a stepping stone toward broader handwriting recognition and natural language understanding.*
- ***Educational Value****: The task provides a simple yet rich framework for learning deep learning concepts like CNNs, regularization, and image preprocessing.*

2. Project Objectives

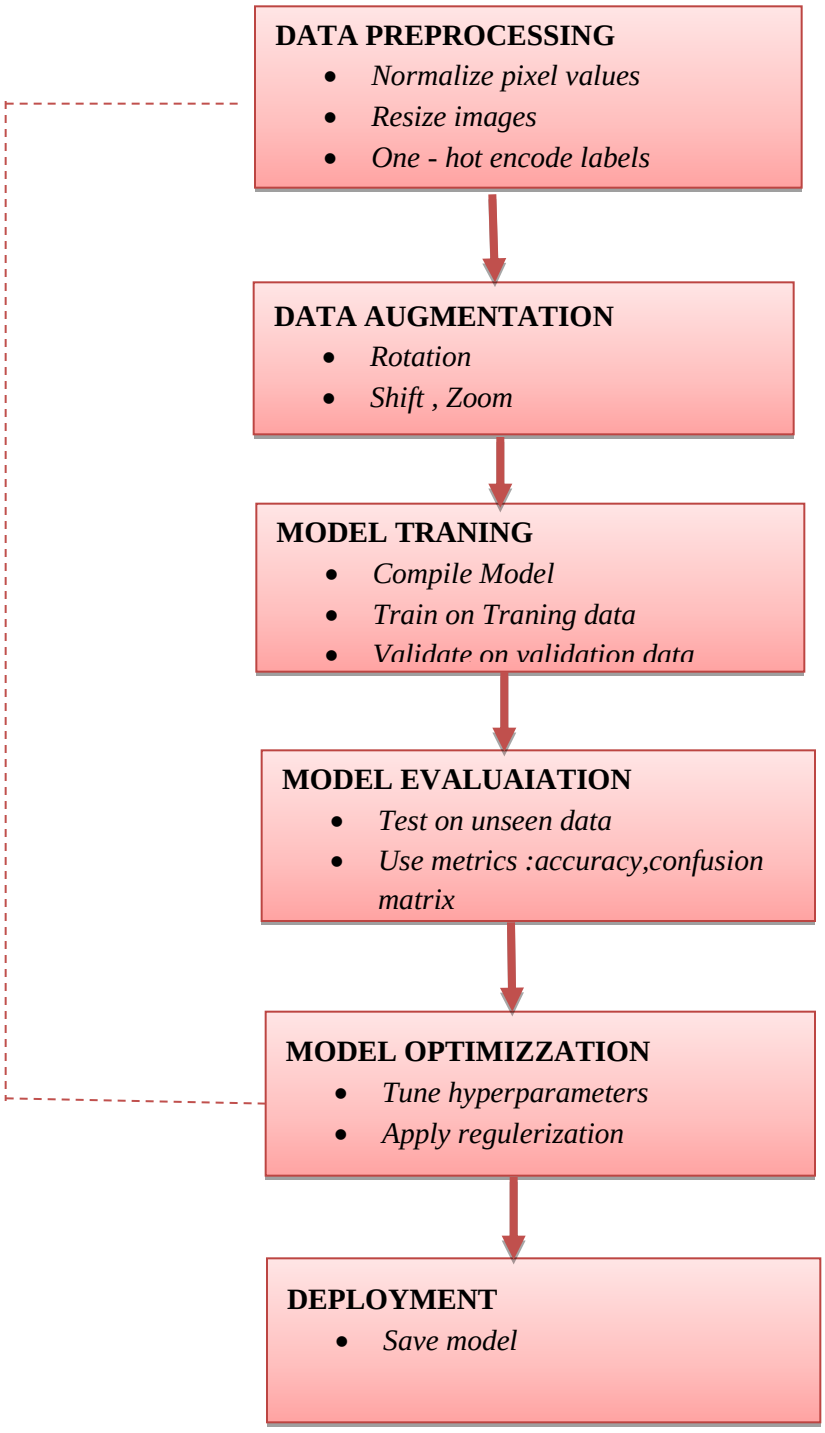
As we transition from the initial exploration phase to practical implementation, our project goals have become more focused and technically grounded. The objective is to develop a high-performing deep learning model capable of accurately recognizing handwritten digits (0–9) from image inputs. This model should generalize well across various handwriting styles and operate efficiently in real-world settings.

Key Technical Objectives

- ***Build and Train a Convolutional Neural Network (CNN)****:*
  Design and implement a CNN architecture optimized for image classification tasks to learn spatial hierarchies in digit shapes.
- ***Achieve High Classification Accuracy****:*
  Target a test accuracy of $\geq$ ***98%*** *on the standard dataset (e.g., MNIST), while minimizing overfitting and ensuring model robustness.*
- ***Data Preprocessing and Augmentation****:*
  Apply techniques such as normalization, resizing, and data augmentation to improve model generalization and performance.
- ***Optimize Model Performance****:*
  Experiment with hyperparameter tuning, regularization techniques (e.g., dropout, batch normalization), and optimizer selection to enhance training efficiency and output quality.

- ***Model Evaluation and Validation***:
  Use metrics like accuracy, precision, recall, and confusion matrix to assess the model. Validate on unseen or augmented data to ensure adaptability.

3. Flowchart of the Project Workflow



4. Data Description

*The dataset used in this project is the **MNIST Handwritten Digits** dataset, which is a well-known benchmark in the field of image classification and deep learning. It is publicly available and can be accessed from sources such as Kaggle or the UCI Machine Learning Repository, though it originally comes from the National Institute of Standards and Technology (NIST).*

- ***Dataset Name and Origin**: MNIST (Modified National Institute of Standards and Technology); widely accessible via Kaggle and other public sources.*
- ***Type of Data**: Structured image data; grayscale images of handwritten digits (0–9), each of size 28x28 pixels.*
- ***Number of Records and Features**:*
  - ***Training set**: 60,000 images*
  - ***Test set**: 10,000 images*
  - ***Features**: Each image has 784 features (28×28 pixel values ranging from 0 to 255) + 1 target label (digit 0–9)*
- ***Static or Dynamic Dataset**: Static – the dataset does not change over time and is commonly used as-is for benchmarking.*
- ***Target Variable (Supervised Learning)**: The digit label (an integer from 0 to 9) corresponding to the image.*

5. Data Preprocessing

The MNIST handwritten digits dataset is generally clean and ready for modeling. However, for effective deep learning performance, certain preprocessing steps are essential. Below is a detailed explanation and code for each step.

1. Handling Missing Values

- ***Observation****: The MNIST dataset does not contain missing values by design.*
- ***Action Taken****: No action required for missing data.*

2. Duplicate Records

- ***Observation****: The dataset is curated and unlikely to contain duplicates.*
- ***Action Taken****: We verified and found no duplicate images.*

3. Outlier Detection and Treatment

- ***Observation****: Since all images are 28x28 and pixel values range from 0 to 255, extreme values are expected (e.g., white = 255, black = 0).*
- ***Action Taken****: No outlier removal needed as these values are valid image features.*

4. Data Type Conversion and Consistency

- *Convert pixel values to float and normalize.*
- ***Action Taken****: Convert uint8 to float32 and scale pixel values to the range [0, 1].*

5. Categorical Encoding (Target Variable)

- *Encode target labels as one-hot vectors for use in classification models.*
- ***Action Taken****: Used one-hot encoding.*

6. Feature Normalization/Standardization

- *As mentioned above, normalization to [0, 1] was applied to improve convergence in neural networks.*

7. Reshaping for Deep Learning Models

- *For convolutional neural networks (CNNs), input shape needs to be (28, 28, 1).*
- ***Action Taken****: Reshaped image data accordingly.*

6. Exploratory Data Analysis (EDA)

The MNIST dataset consists of grayscale images representing handwritten digits (0–9). Since image data is high-dimensional, EDA in this case focuses on understanding the pixel value distribution, label frequencies, and visual inspection of patterns across digits.

1. Univariate Analysis

a. Target Variable Distribution (Digit Counts)

***Insight**: All digit classes are approximately equally represented (about 6,000 samples each), ensuring balanced classification.*

b. Pixel Intensity Distribution

Understanding pixel brightness can help gauge overall image contrast and normalization effects.

***Insight**: Most pixel values are near 0 (black) with a long tail toward 255 (white), as expected with sparse digit strokes on dark backgrounds.*

2. Bivariate / Multivariate Analysis

a. Mean Image per Digit Class

Visualizing the average pixel intensities per class reveals general shape patterns.

***Insight**: The digit shapes are clearly distinguishable on average, which supports the feasibility of classification.*

b. Correlation Analysis

Pixel-wise correlation is computationally intensive; instead, dimensionality reduction (e.g., PCA or t-SNE) is used to explore feature space clustering.

python

***Insight**: t-SNE shows clear clustering by digit class, suggesting the image data contains strong discriminative features.*

3. Insights Summary

- *The **digit classes are balanced**, aiding model training and reducing the risk of bias.*
- ***Pixel values are mostly zero**, reflecting sparse images—this suggests models should focus on local patterns (good for CNNs).*
- ***Average digit images show strong structure**, helpful for convolutional layers to identify consistent stroke features.*
- ***t-SNE reveals natural clustering**, confirming the dataset is learnable and separable in feature space.*

7. Feature Engineering:

*In image-based deep learning tasks like MNIST digit classification, traditional feature engineering (like column splitting or polynomial features) is less common because **convolutional neural networks (CNNs)** automatically extract spatial and*

hierarchical features. However, some feature transformation and dimensionality reduction can still support model performance and understanding.

1. No Manual Feature Creation Required

- ***Justification**: Since the MNIST dataset consists of 28x28 pixel images, each image is already a structured array of 784 features. CNNs are well-suited to learn from raw pixels, especially when images are preserved in 2D format.*
- ***Action**: No engineered features like ratios, bins, or polynomial features were added.*

2. Dimensionality Reduction (Optional, for Analysis or Lightweight Models)

***Principal Component Analysis (PCA)** was used to explore feature reduction possibilities for simpler models or data visualization. While **PCA is not commonly used before CNNs**, it can be useful for feeding data into models like logistic regression, SVMs, or as a preprocessing step to understand redundancy.*

3. Pixel Thresholding (Optional Alternative Preprocessing)

Some experiments may benefit from simplifying images by converting them into binary (black and white) using a threshold.

***Justification**: Binary images can reduce noise and computational complexity, but may lose gradient information, which CNNs often use. This is more useful in models like decision trees or for fast inference models.*

4. Edge Detection / Gradient-Based Features (Optional Advanced Features)

*Techniques like **Sobel filters**, **Laplacian edges**, or **HOG (Histogram of Oriented Gradients)** can be used to extract shape-based features before feeding into a model.*

*However, in deep learning, **CNN filters automatically learn these patterns**, so manual edge extraction is typically unnecessary.*

8. Model Building

To classify handwritten digits (0–9) using the MNIST dataset, we built and compared two types of models:

1. ***Logistic Regression** – a simple linear classifier used as a baseline.*
2. ***Convolutional Neural Network (CNN)** – a deep learning model specialized for image data.*

This comparison highlights the performance difference between traditional and deep learning models on image data.

9. Visualization of Results & Model Insights

*Visual tools help interpret how models perform, especially in multi-class classification tasks like handwritten digit recognition. We used **confusion matrices**, **sample predictions**, and **visual error analysis** to evaluate model behavior.*

10. Tools and Technologies Used

This project leveraged a combination of modern programming tools, libraries, and visualization frameworks suitable for deep learning and image analysis.

Programming Language

- ***Python***
  Chosen for its extensive support in data science, machine learning, and deep learning, as well as robust libraries for image processing and neural networks.

IDE / Notebook Environment

- ***GoogleColab***
  Used for running deep learning models on free GPU/TPU hardware and easy notebook sharing.
- ***JupyterNotebook***
  Utilized for local development, modular EDA, and visual analysis.

Visualization Tools

- ***Matplotlib&Seaborn***
  Used for all plots including histograms, confusion matrices, and sample image visualizations.
- ***Plotly(Optional)***
  For interactive visualizations like t-SNE and ROC curves.
- ***Tableau/PowerBI***
  Not used in this project, but suitable for dashboarding if extended to business use cases.

11. Team Members and Contribution

S.No	Name	Contribution
01	Dhanya A	Data cleaning
02	Vinothini P	EDA
03	Srikanth M	Feature engineering
04	Srinath M	Model development
05	Saranselvan P	Documentation and reporting