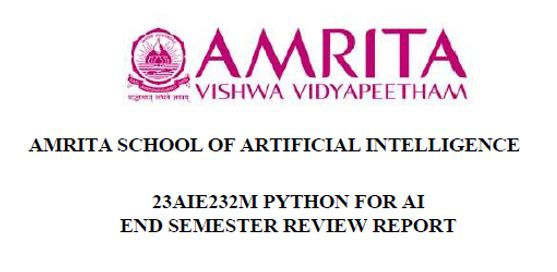
****

****

****

**Digit Classifier Using CNN**

End Semester Review Report

*Submitted by*

Kaushal Soundararajan – CB.EN.U4ECE23125

Nidhil Remila Shivakumaran – CB.EN.U4ECE23129

Sanjay Subramanyan S– CB.EN.U4ECE23147

Shriyaank R – CB.EN.U4ECE23151

Vivyn M – CB.EN.U4ECE23157

*in partial fulfilment for the award of the degree of*

BACHELORS OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

23AIE232M - PYTHON FOR AI



AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

APRIL – 2025

**Contents**

Acknowledgement 4

List of Figures 5

Abstract 6

1 Introduction 7

2 Literature Review 8

3 Technology and Libraries Used 11

4 Methodology 12

5 Proposed Work 13

6 Results 14

7 Challenges and Risks 22

8 Conclusion 22

9 References 22

10 Appendix 23

11 Contribution and Timeline 26

**Acknowledgement**

Please accept my sincere gratitude for all those who assisted and, in some way, supported us throughout this project. We especially appreciate our faculty mentor Ms. Chandini M for her enduring support, wise suggestions, and professional critique which shaped the outcome of our project in so many ways. Our thanks also go to our institution, Amrita Vishwa Vidyapeetham, for the infrastructure and education ecosystem that enabled us to undertake and complete this project successfully. My heartfelt appreciation goes to our wonderful teammates – Kaushal Soundararajan, Nidhil Remila Shivakumaran, Sanjay Subramanyan S, Shriyaank R and Vivyn M– for their committed effort, collaboration, and team spirit which turned this project into an enjoyable experience. Last but not least, we are grateful to every member of the faculty and classmates who at some point helped us directly or indirectly. This has been an enriching learning experience and we are grateful for all the support that we got throughout.

**List of Figures**

1. Scatterplot of average intensity of Digit 14
2. Histogram of Number of samples per digit class 14
3. Other Models 15
4. Inputs 18
5. Truth vs Prediction Outputs 18
6. Confusion Matrix 18
7. Loss and Accuracy Curves 19
8. Model Parameters 19
9. Convolution Layer 1 20
10. Convolution Layer 2 20
11. User Interface 21
12. Input and Output of User Interface 21

**Abstract**

A digit classifier based on Convolutional Neural Network (CNN) learns to recognize digits from images efficiently. The model is trained using MNIST dataset that contains digits (0 – 9). Furthermore, convolutional layers are utilized to extract features, while fully connected layers are used for classification. The method involves designing a CNN model that gives the best results during the classification of the images. The strategy outperformed machine learning techniques that manually extracted features from pixels intensities. This method is used in the processing of digitized documents, postal automation, and the banking system. The project aims to result in a high accuracy digit classification model to show the power of CNN in the application to image recognition.

**Introduction**

**Background and Motivation:**

The handwritten digit recognition is a very popular computer vision task. It has applications in document digitization, postal automation, and banking. Many traditional which are machine learning rely on manually engineered characteristics that do not generalize well to different handwritten types. Convolutional Neural Networks (CNNs) can automatically learn spatial hierarchies of features and classify images. This makes them widely used for digit recognition tasks.

**Problem Statement:**

Handwritten digit recognition is a challenging problem because of the difference in styles and noise and distortions of images. Traditional techniques call for major preprocessing and feature engineering, which makes them less flexible to data. This project is to develop a CNN-based classifier for handwritten digits 0 – 9.

**Objectives and Scope:**

This project’s main aim is to design a CNN model for handwritten digits classification with MNIST dataset. The model must be highly accurate yet functionally efficient. The above includes data use, preprocessing model training evaluation testing. Furthermore, the project investigates practical applications like automatic form filling and cheque clearing systems.

**Literature Review**

**Summary of Existing Work:**

Many researchers have studied handwritten digit classification using traditional techniques like k-nearest neighbours (KNN), support vector machines (SVM), and artificial neural networks (ANN). KNN is easy to use and quite effective, though it does become expensive computationally as the size grows. The digit classification has proven that SVM is a good technique for this task. Most importantly, the SVM has to do proper kernel function tuning for getting better results. ANN is capable of learning complex patterns but struggles to learn from high-dimensional image data without feature extraction. These methods depend on crafted features, making them less adaptable to multiple handwriting styles.

**How This Project Improves Upon Existing Solutions:**

This project unlike earlier technique adopts a convolutional neural network (CNN), which automatically learns spatial features present in the digit images eliminating the need for manual extraction of features. CNNs enhance the accuracy of classification tasks by capturing local patterns in images. The proposed method also applies data normalization and Dropout to improve generalization and robustness of the network. The project uses CNNs to provide an efficient and scalable solution for handwritten digit recognition as compared to simple machine learning techniques.

**Paper 1: A Robust Model for Handwritten Digit Recognition using Machine and Deep Learning Technique**

**Importance and Challenges:**

* HDR, or handwritten character recognition, is a key application of pattern recognition, which has significant real-world applications, including banking, postal services, and digitized document processing.
* Developing strong models is very crucial for the successful recognition of characters as they vary due to many reasons.

**Dataset:**

* The MNIST dataset, containing 70,000 grayscale images of handwritten digits (60,000 for training, 10,000 for testing), is a standard benchmark for HDR research.

**Other Machine Learning Approaches used:**

1. **Support Vector Machines (SVM):**
   * Widely used for classification tasks, SVM achieves recognition accuracies of up to 97.91% on MNIST, but struggles with scaling for large datasets and achieving higher accuracies needed in sensitive domains.
2. **K-Nearest Neighbour (KNN):**
   * KNN performs well for HDR with accuracies up to 96.8% but is computationally intensive, especially with large datasets.
3. **Comparison to CNN:**
   * SVM and KNN achieve moderate accuracy but lack the scalability and feature extraction efficiency provided by deep learning models.
4. **Convolutional Neural Networks (CNN):**
   * CNN has emerged as a superior model for HDR, leveraging convolutional and pooling layers to efficiently extract and classify features.
   * Researchers have achieved high accuracies using CNN:
     + 98.5% with extended minimal CNN.
     + 99.2% with CNN coupled with gradient descent.
     + 99.21% using DL4J-based CNN.

**Comparative Studies:**

* + CNN consistently outperforms traditional approaches like SVM and KNN in accuracy and computational efficiency, with models achieving over 99% accuracy on MNIST.

**Limitations:**

* Despite high accuracies, challenges remain in handling more complex handwriting datasets with cursive styles or multilingual characters.
* Higher accuracies may require deeper architectures, but this increases computational cost.

**Paper 2: A robust CNN model for handwritten digits recognition and classification**

1. **Historical Progress:**

* The early days of HDR's neural networks took place in 1980s and 1990s. Limited capacity of datasets and hardware became an impediment.
* The study of neural networks for high dynamic range arose again in the 2010s due to better computers and large datasets like MNIST.

1. **Significance of HDR:**

* Uses in online/offline identification, bank-check processing, postal-address interpretation.
* There are challenges like changes in writing and orientation.

1. **Traditional Techniques:**

* KNN and decision tree methods achieved moderate accuracies but could not scale or extract features for complex databases

1. **Deep Learning and CNNs:**

* CNNs are the go-to architecture for HDR, as they can take in large amounts of data without much pre-processing.
* Since the convolutional layers adeptly extract features, and the pooling layers diminish the dimensions and computing power,

.

1. **Limitations of Current Models:**

* Cursive and other irregular handwriting styles may affect accuracy.
* More hidden layers in CNNs may give us less accuracy because they may overfit or have computational limitation.

**Technologies and Libraries Used**

**Libraries:**

* NumPy
* Pandas
* Matplotlib
* Scikit-Learn
* TensorFlow
* Seaborn

**Methodology**

**Data Collection:**

* Dataset: Digit dataset (CSV format)
* Source: Provided training dataset (digitrain.csv)

**Data Preprocessing:**

* Normalize pixel values to [0,1].
* Reshape data for CNN input (28x28 images with 1 channel).
* One-hot encode labels for multi-class classification.
* Split dataset into 80% training and 20% validation.

**Exploratory Data Analysis (EDA):**

* Display sample images.
* Use histograms and scatterplots to analyse label distributions.
* Compute class frequencies to detect imbalances.

**Feature Engineering:**

* No manual feature extraction (CNNs automatically extract relevant features).
* Data augmentation to improve generalization.

**Model Building:**

* **Architecture:**
  + Two convolutional layers (32 and 64 filters, 3x3 kernel, ReLU activation).
  + Max-pooling layers (2x2 pool size) for down-sampling.
  + Dropout layers (0.25-0.5) to prevent overfitting.
  + Fully connected layers with so􀅌max activation for classification.
* **Compilation:**
  + Loss Function: Categorical Cross entropy
  + Optimizer: Adam
  + Metrics: Accuracy

**Training & Evaluation:**

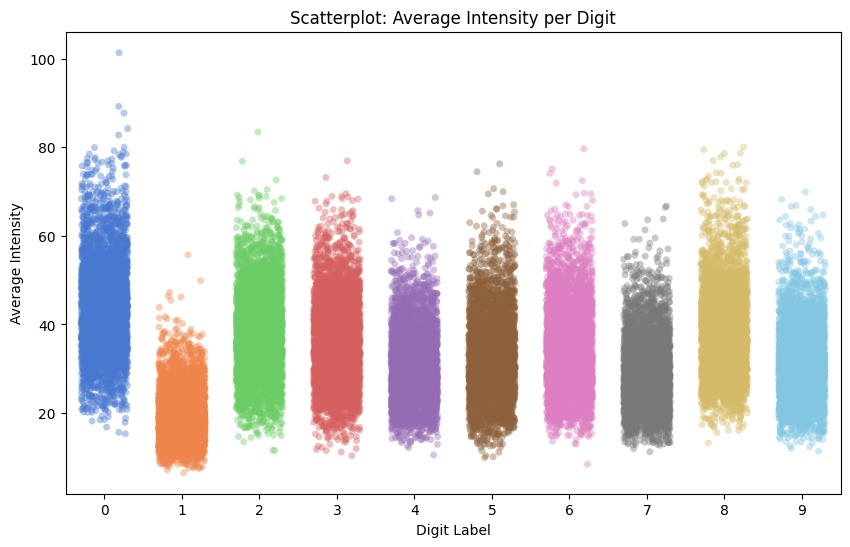
* **Hyperparameters:**
  + Epochs:10
  + Batch Size: 64
  + Learning Rate Scheduling: ReduceLROnPlateau
* **Evaluation Metrics:**
  + Accuracy, Loss
  + Confusion Matrix
  + Classification Report
* **Feature Map Visualization:**
  + Extract convolutional layer outputs to visualize feature maps for each digit.
  + Identify patterns learned by different layers.
* **Accuracy & Loss Curves:**
  + Training/validation accuracy and loss plotted over epochs.
  + CNN achieved 99% accuracy on validation data.

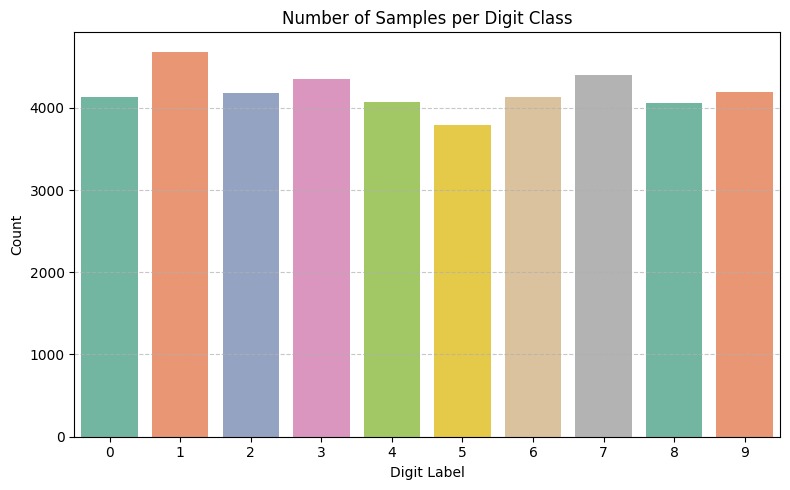
**Proposed Work**

* Successfully loaded and pre-processed dataset.
* Built and trained CNN model.
* Implemented EDA and feature visualization.
* Evaluated model performance.
* Generated confusion matrix and classification report.
* Further hyperparameter tuning.
* Testing with additional datasets for generalization.
* User interface to receive inputs has been implemented.

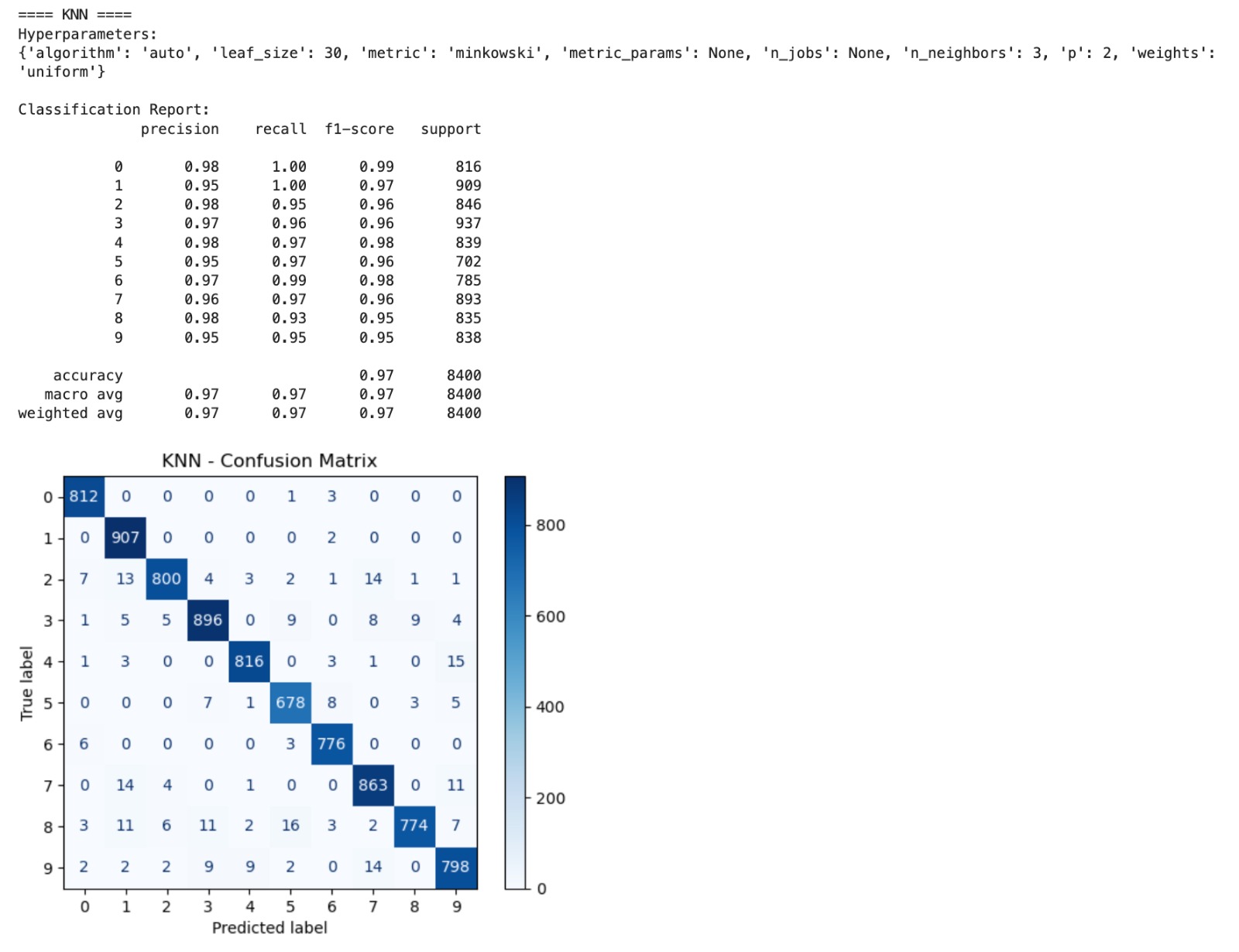
**Results**

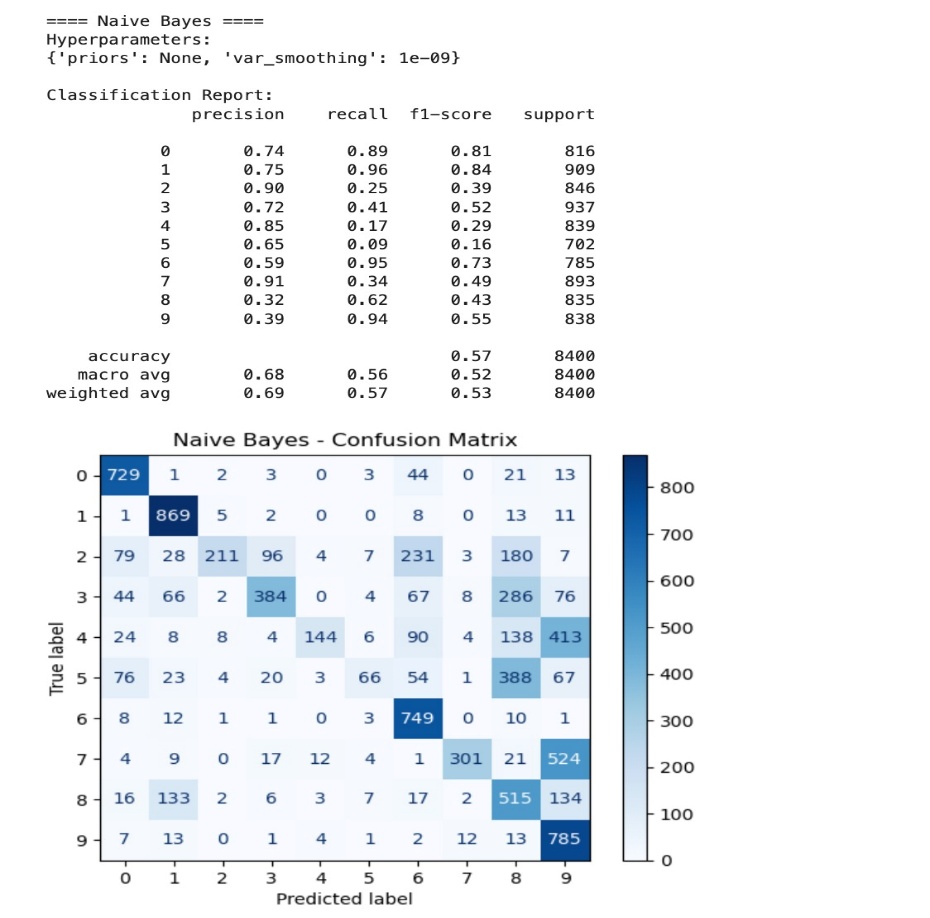
**EDA:**

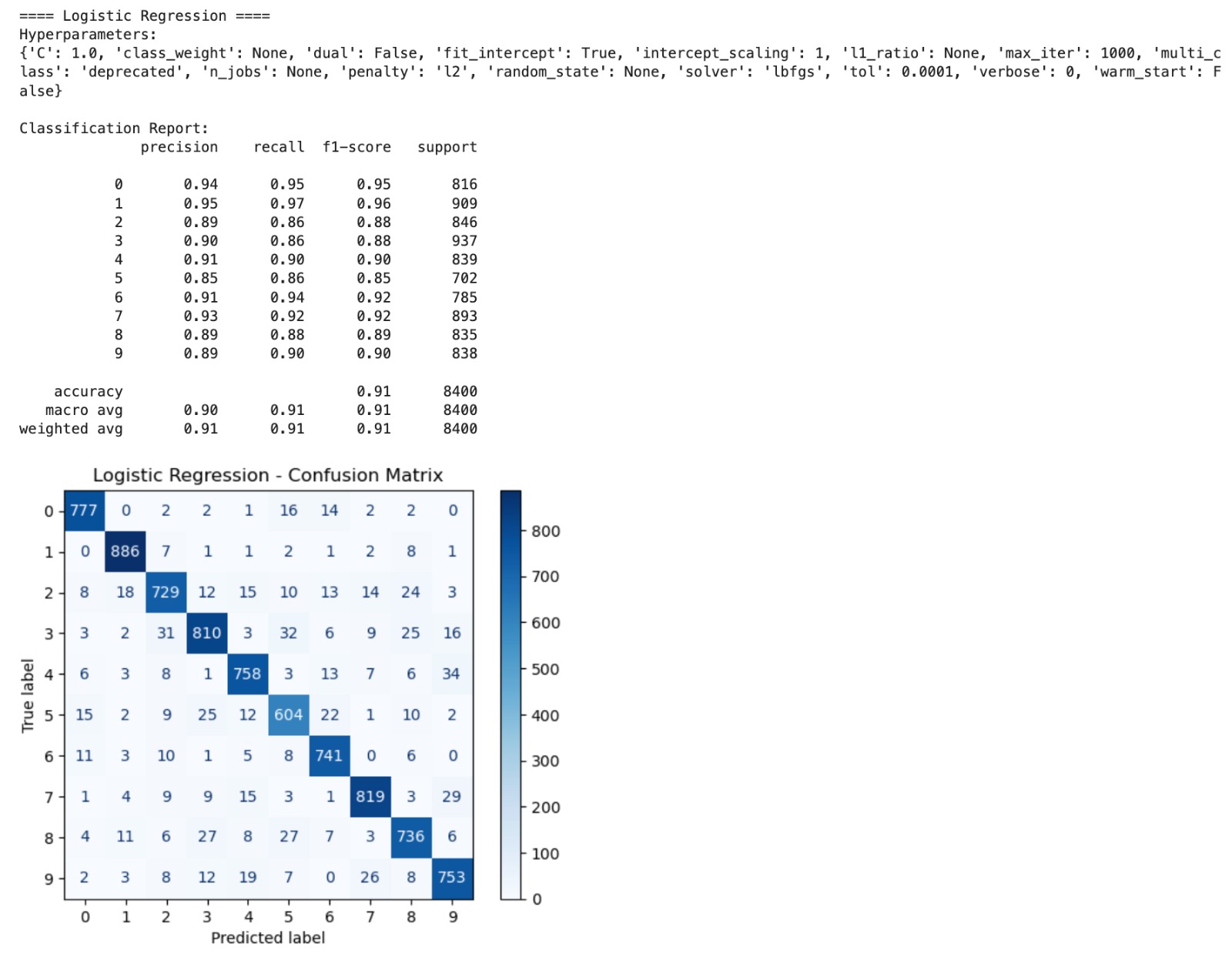


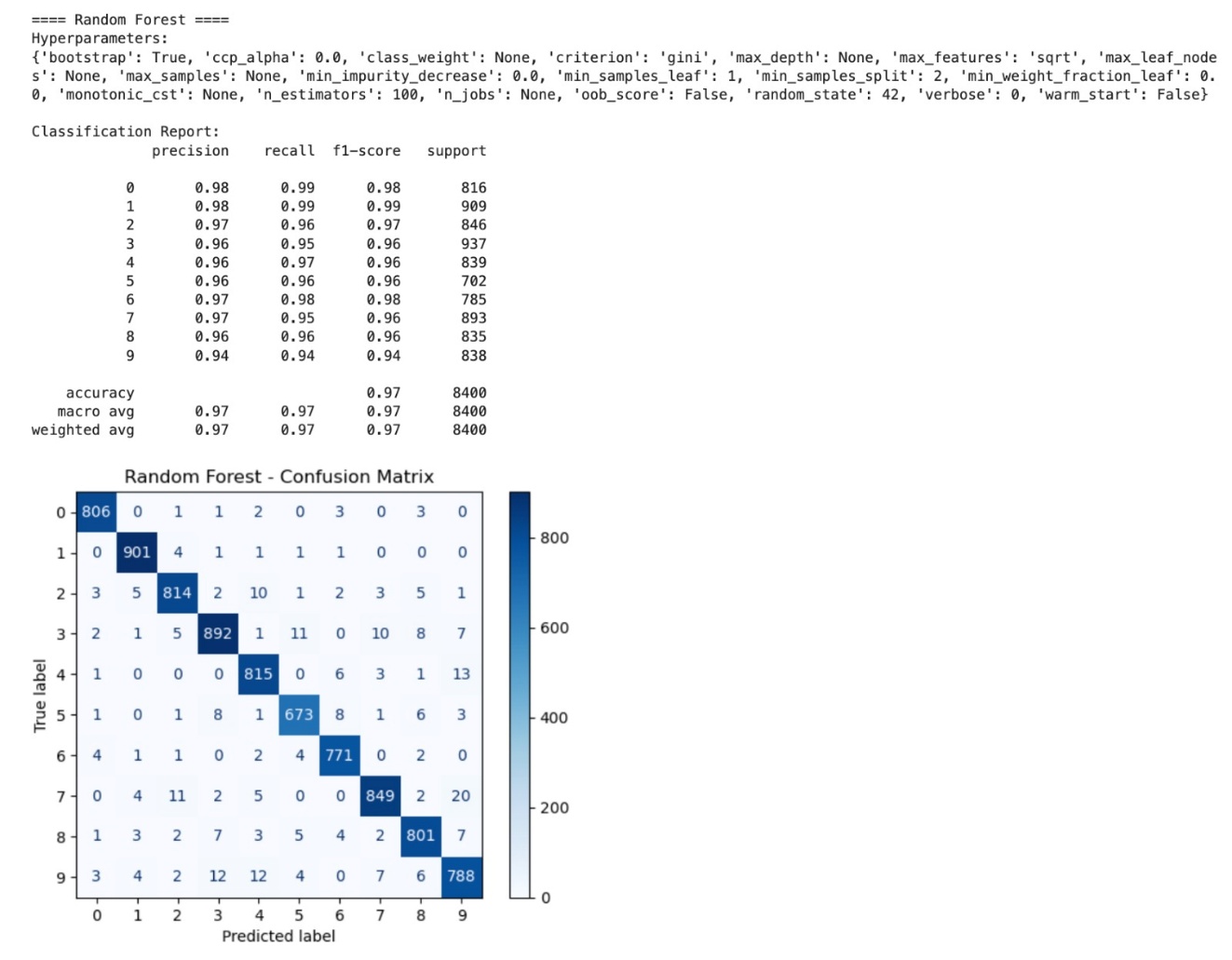


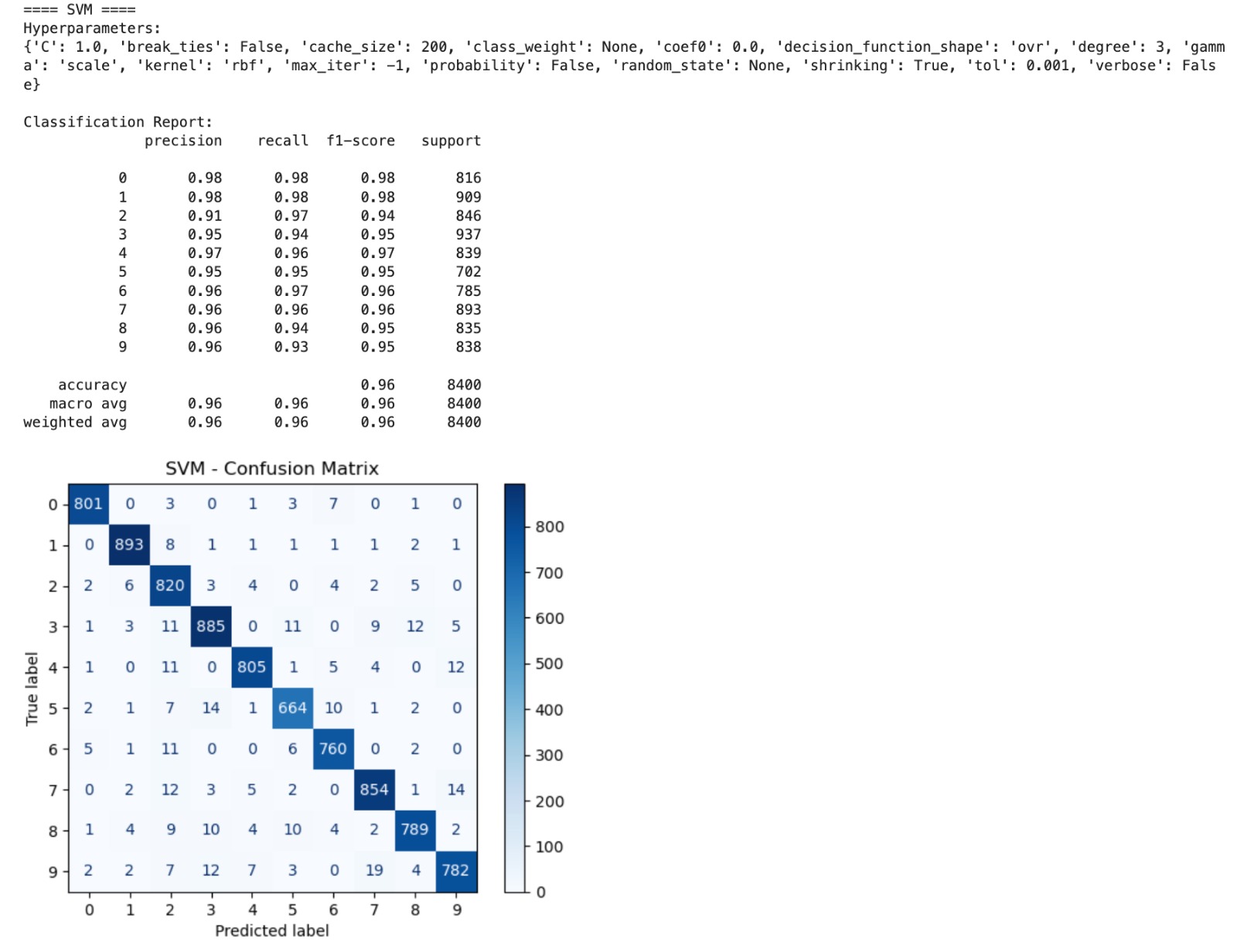
**Other Models:**

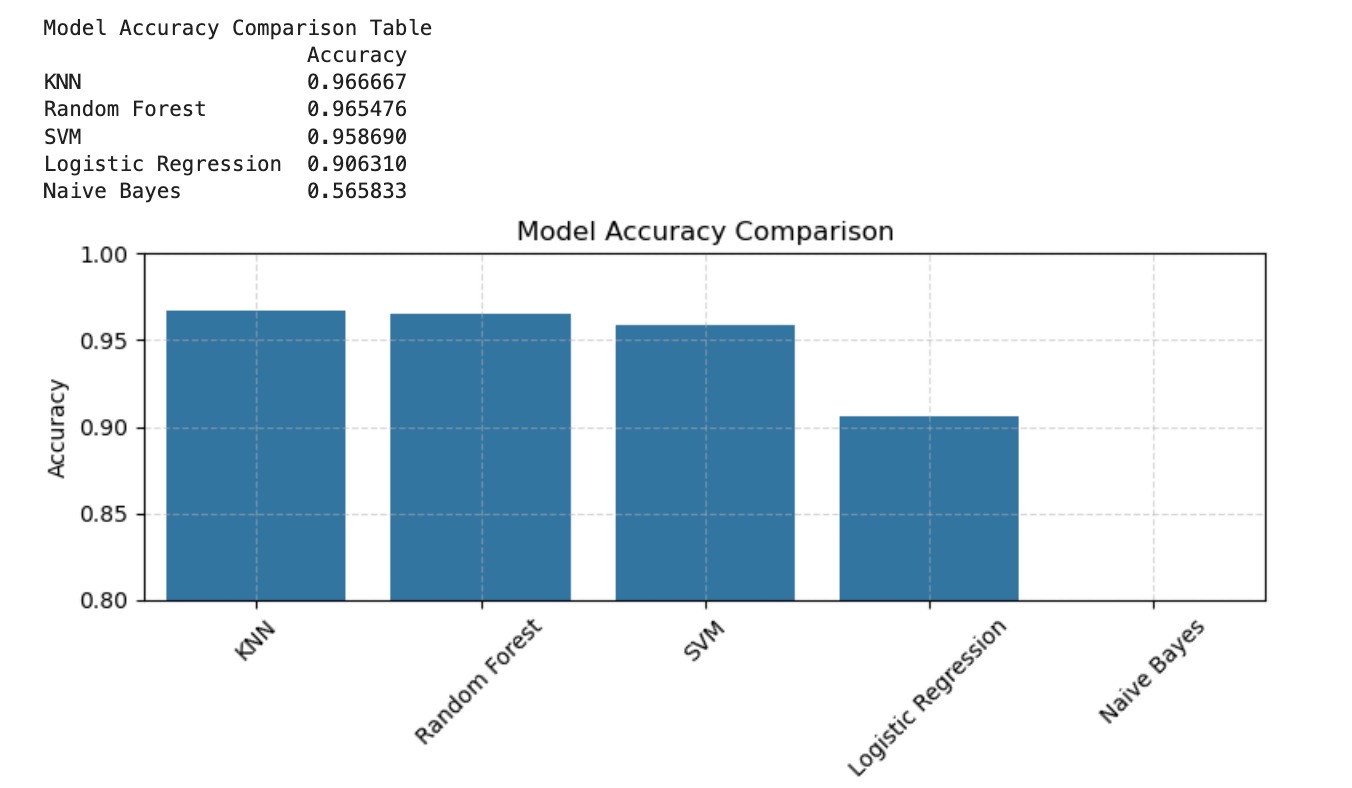




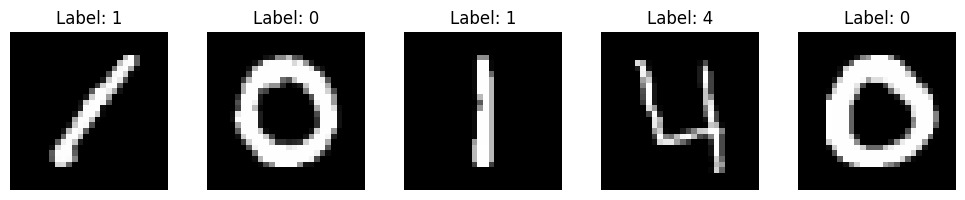




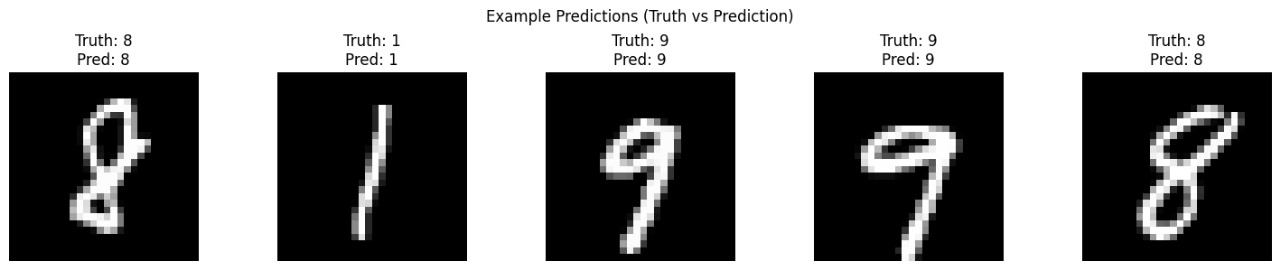




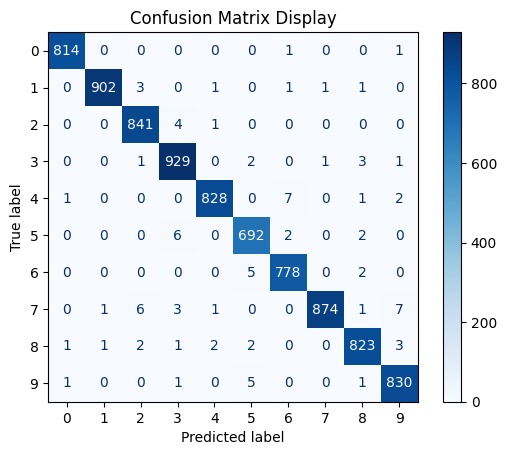
**Input:**



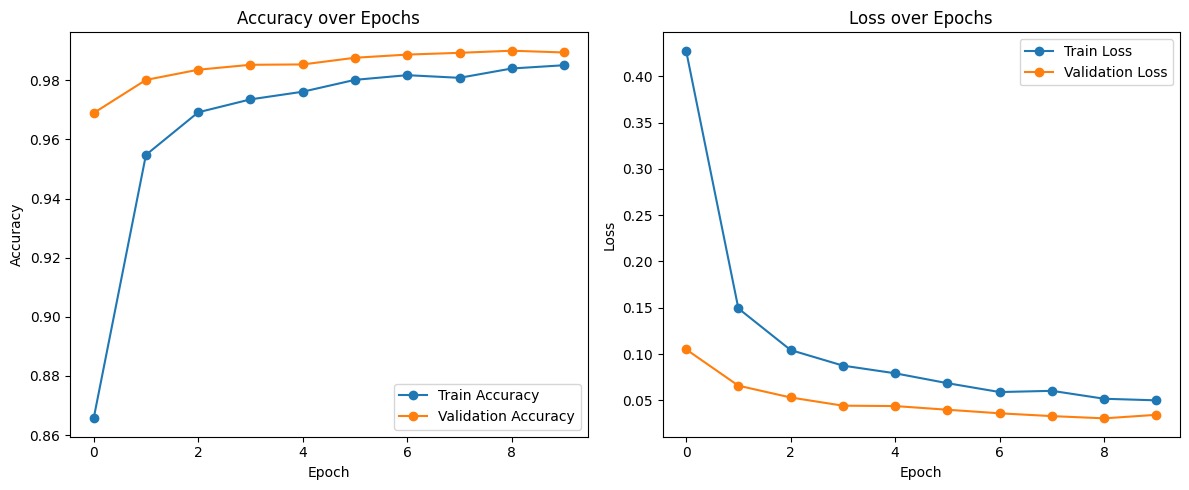
**Outputs:**



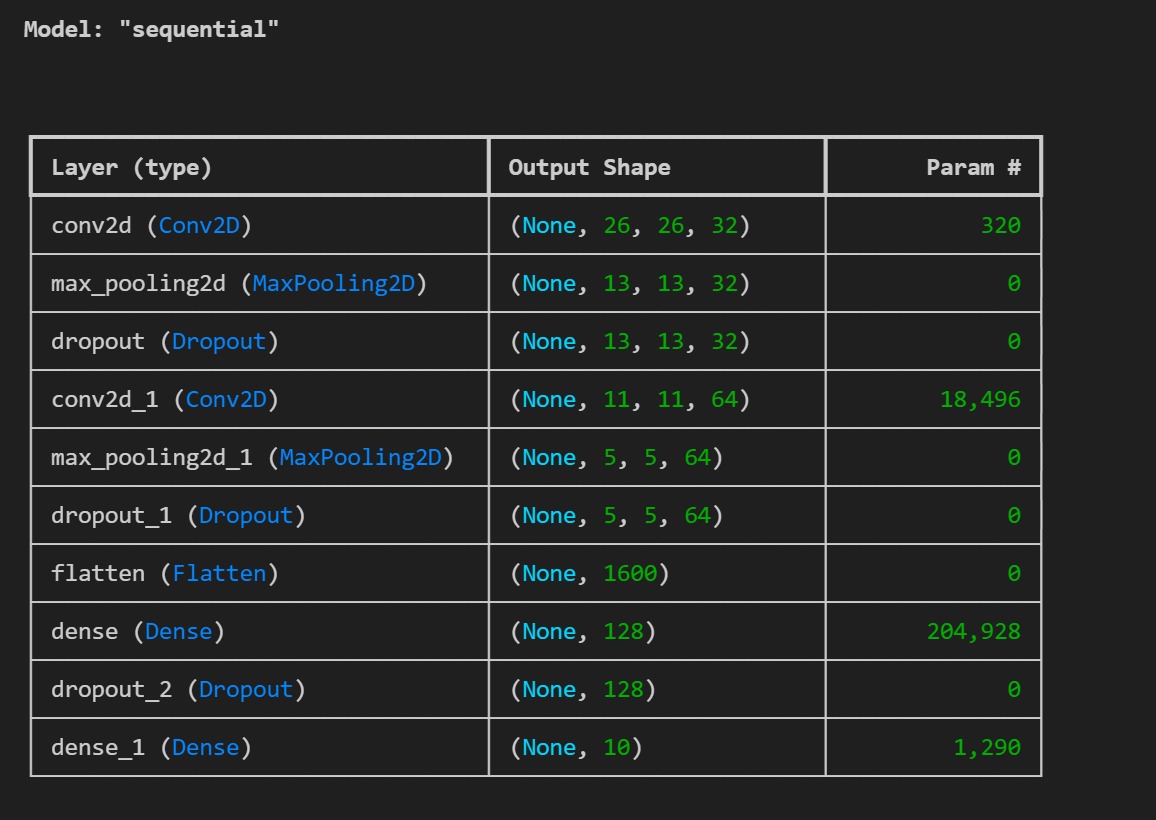
**Confusion Matrix:**



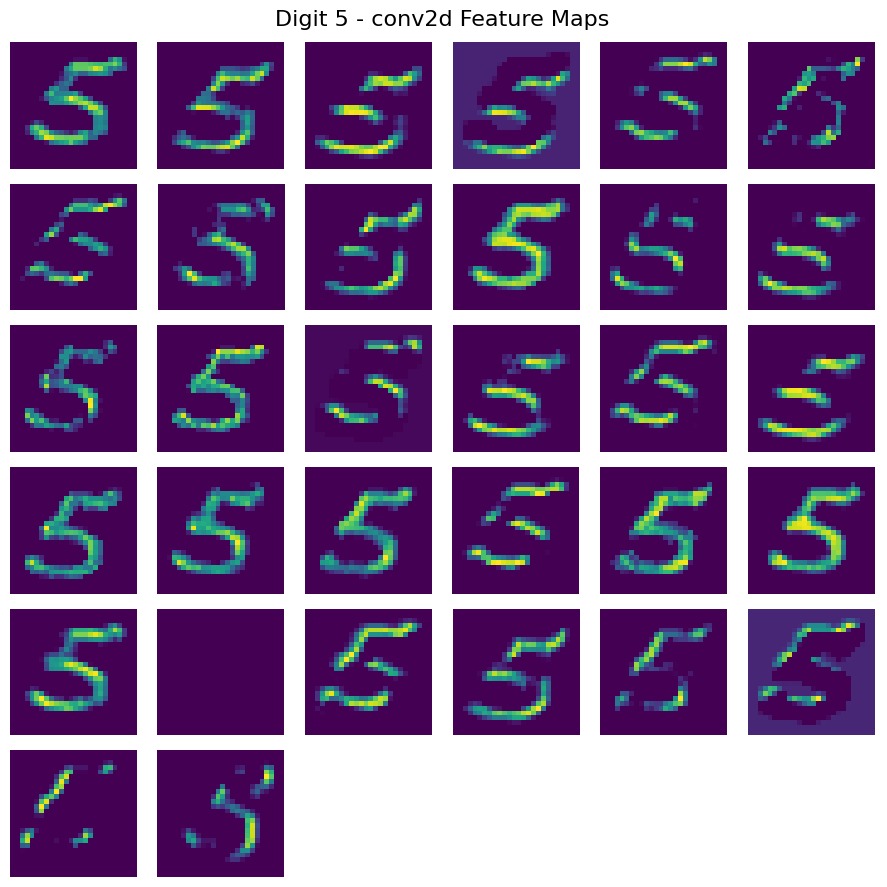
**Loss and Accuracy Curve:**



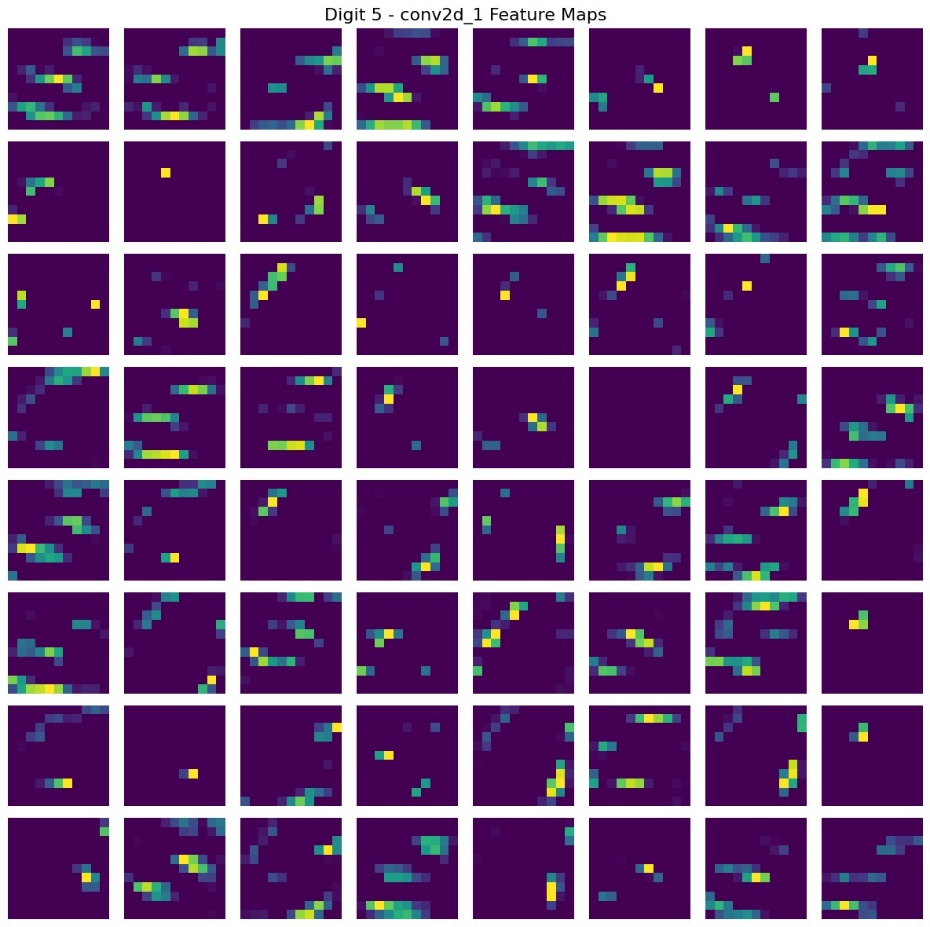
**Model Parameters:**



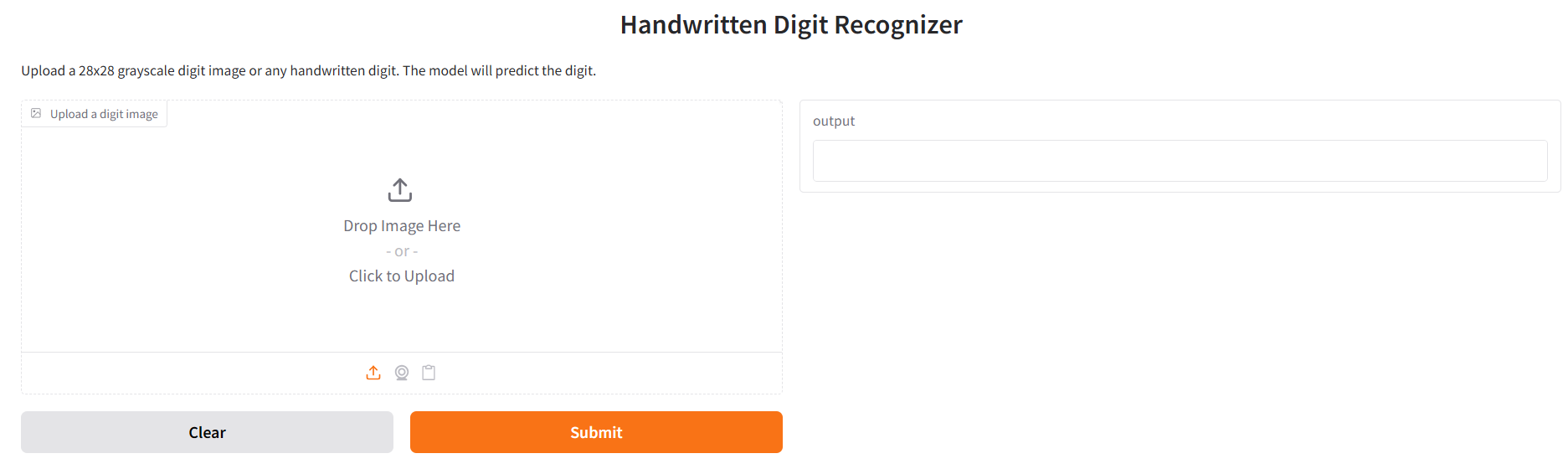
**Convolution Layer 1:**



**Convolution Layer 2:**

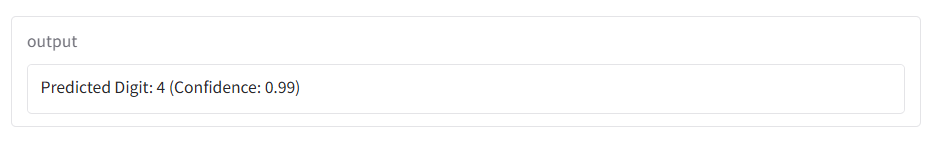


**User Interface:**



Input and Output:





**Challenges & Risks**

* Computational Limitations: Training deep networks requires high GPU resources.
* Overfitting: Managed using dropout and data augmentation.

**Conclusion**

The designed CNN model for classifying digit is more beneficial than machine learning models for image recognition. Because they take the raw image as input and automatically learn the hierarchical spatial feature, they don’t have to depend on manual features. This allows CNNs to obtain superior performance and scaling with complex datasets like MNIST.

This project was highly successful for the CNN model. By applying convolutional layers to extract features, the model can produce good accuracy. The convenience makes CNNs suited for spatial hierarchies and local dependencies which is difficult for conventional models.

**FUTURE WORKS:**

For improved accuracy of classification, a pretrained CNN model like ResNet can be used. The proposed models contain a deeper architecture and a higher number of filters and kernels: Thus, ResNet model are capable of capturing more complex variations like different writing styles, scale, shift and rotation invariance. Another option is to use a hybrid model that combines CNN with Support Vector Machine (SVM) instead of the usual softmax classifier. In this method, the CNN gets used to extract the image features while the SVM acts as the final classifier. SVMs perform very well for high-dimensional feature spaces. SVMs often outperform softmax as we go for fine-grained class boundaries.

**References**

[1] Agrawal, Ayush Kumar, et al. “A Robust Model for Handwritten Digit Recognition Using Machine and Deep Learning Technique.” *2021 2nd International Conference for Emerging Technology (INCET)*, 21 May 2021, pp. 1–4, [https://doi.org/10.1109/incet51464.2021.9456118. Accessed 19 Apr. 2025](https://doi.org/10.1109/incet51464.2021.9456118.%20Accessed%2019%20Apr.%202025).

[2] S. Ali, Z. Sakhawat, T. Mahmood, M. S. Aslam, Z. Shaukat and S. Sahiba, "A robust CNN model for handwritten digits recognition and classification," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications( AEECA), Dalian, China, 2020, pp. 261-265

[3] dataset : <https://www.kaggle.com/datasets/hojjatk/mnist-dataset?resource=download>

**Appendix**

**Code:**

**MODEL BUILDING:**

1. **Import Libraries:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

**2. Data Preprocessing**

# Normalize and reshape data

X = X / 255.0

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

# One-hot encode labels

from tensorflow.keras.utils import to\_categorical

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

**3. Build CNN Model**

model = Sequential([

Conv2D(32, (3,3), activation='relu', input\_shape=(28,28,1)),

MaxPooling2D(2,2),

Conv2D(64, (3,3), activation='relu'),

MaxPooling2D(2,2),

Dropout(0.25),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(10, activation='softmax')

])

**4. Compile and Train**

model.compile(optimizer='adam',loss='categorical\_crossentropy', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, batch\_size=64, validation\_split=0.2)

**5. Evaluate Model**

loss, accuracy = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {accuracy:.2f}")

**6. Plot Accuracy and Loss**

plt.plot(history.history['accuracy'], label='Train Acc')

plt.plot(history.history['val\_accuracy'], label='Val Acc')

plt.title('Model Accuracy')

plt.legend()

plt.show()

**UI BUILIDING:**

**1. Loading the Model:**

final\_model = tf.keras.models.load\_model(r"C:\Users\shriy\Downloads\CNN try 2\CNN try 2\model.h5")

**2. Formatting the Image:**

def format\_image(img):

img = img.resize((28, 28)) # Resize to 28x28

img\_array = np.array(img) # Convert to NumPy array

return img\_array

**3. Image Prediction:**

def predict\_digit(model, img\_array: np.ndarray) -> str:

data = img\_array.reshape(1, 28, 28, 1).astype('float32') / 255.0 # Reshape + normalize

prediction = model.predict(data) # Get prediction

digit = np.argmax(prediction) # Highest confidence class

return str(digit)

**4. FastAPI Endpoint for Prediction:**

@app.post("/predict")

async def predict(file: UploadFile = File(...)):

contents = await file.read() # Read uploaded image file

img = Image.open(io.BytesIO(contents)).convert('L') # Convert to grayscale

img\_array = format\_image(img) # Format image

digit = predict\_digit(final\_model, img\_array) # Predict digit

return {"Digit": digit} # Return response

**5. Root Endpoint:**

@app.get("/")

def read\_root():

return {"message": "MNIST digit recognition API is live!"}

**GitHub Repository:**

* <https://github.com/Cosmic-dorito/Digit-Classifier>

**Contributions**

* Kaushal (CB.EN.U4ECE23125) – CNN Theory, Project Report
* Nidhil (CB.EN.U4ECE23129) – Model Building and UI Code
* Sanjay (CB.EN.U4ECE23147) – Model Building Code
* Shriyaank (CB.EN.U4ECE23151) – User Interface Code
* Vivyn (CB.EN.U4ECE23157) – User Interface Code and PPT

**Timeline**

Week 1:

* Project Topic Selection
* Obtaining Dataset

Week 2:

* Collecting base papers and conducting a literature review
* Understanding CNN concept

Week 3:

* Data Preprocessing
* Model Building and Hyperparameter Tuning

Week 4:

* Performing EDA and obtaining results
* Finalizing CNN Model for Digit Classification

Week 5:

* Creating User Interface and integrating model with it
* Troubleshooting code

Week 6:

* Project Report
* PPT