NATIONAL INSTITUTIONAL RANKING FRAMEWORK
NIRF-2024
Engineering Rank Band (151-200)
Pharmacy Rank - 77
Innovation Rank Band (11-50)

www.kiet.edu
Delhi-NCR, Ghaziabad

KIET GROUP OF INSTITUTIONS
Connecting Life with Learning

CELEBRATING 25 YEARS OF EXCELLENCE

ACCREDITED WITH GRADE A+ NAAC
Accredited by
NAAC with Grade "A+"

Internal Quality Assurance Cell
Work Towards Excellence
IQAC
KIET Group of Institutions

Introduction to AI(AI101B)

HandWritten Digit Recognition

session 2024-25

Even Semester

Team Leader: **Aaryan Gautam** and University Roll no. : **202410116100003**

Member Name: **Aditya Chaudhary** and University Roll no. : **202410116100010**

Member Name: **Akash Chaudhary** and University Roll no. : **202410116100015**

Project Supervisor : Mr. Apoorv Jain

Content

- **Introduction**
- **Methodology**
- **Code**
- **Screenshots Output**

Introduction

In recent years, artificial intelligence (AI) and machine learning (ML) have seen significant advancements, especially in the field of image processing and pattern recognition. One of the foundational problems in this domain is **Handwritten Digit Recognition**, which involves classifying images of handwritten digits into their corresponding numeric labels (0–9).

- This task is not only fundamental for learning and understanding machine learning concepts but also has practical applications in areas such as **postal mail sorting**, **bank cheque processing**, and **form digitization**. The ability of a machine to read and interpret handwritten text mimics human capabilities and opens doors to developing more complex document recognition systems.

In this project, a **Convolutional Neural Network (CNN)** — a type of deep learning model — is trained on the MNIST dataset to recognize and classify handwritten digits with high accuracy. CNNs are particularly well-suited for image-based tasks as they are capable of automatically detecting spatial hierarchies and local patterns in images.

This report outlines the step-by-step development of the digit recognition model, including data preprocessing, model design, training, evaluation, and visualization of results. The objective is to demonstrate how AI can be effectively applied to solve real-world pattern recognition tasks with minimal human intervention.

Methodology

1. Data Collection

The MNIST dataset was used, which is a well-known benchmark in image classification tasks .It contains **70,000 grayscale images** of handwritten digits (0–9) of size 28x28 pixels.

2. Data Preprocessing

Images were **normalized** by scaling pixel values between 0 and 1 to help the model train efficiently. Each image was reshaped to include a single channel (for grayscale) with a shape of (28, 28, 1).

Methodology

c) Data Visualisation

- A few sample images were plotted to understand the nature of the data..
- This helped ensure the dataset was loaded correctly and gave visual insight into digit variety and handwriting styles..

3. Model Training

The model was compiled using the **Adam optimizer** and **Categorical Crossentropy** as the loss function.

Code Typed

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
```

Code Typed

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 2. Visualize some samples
plt.figure(figsize=(10, 5))
for i in range(12):
    plt.subplot(3, 4, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.suptitle('Sample Digits from MNIST Dataset',
 fontsize=16)
plt.tight_layout()
plt.show()
```

Code Typed

```
 x_train = x_train.reshape(-1, 28, 28, 1) / 255.0
x_test = x_test.reshape(-1, 28, 28, 1) / 255.0
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

# 4. Create the model (Simple CNN)
model = Sequential([
    Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(pool_size=(2,2)),
    Conv2D(64, kernel_size=(3,3), activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

Code Typed

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train_cat, epochs=5, validation_split=0.2)

# 6. Plot training vs validation accuracy and loss
plt.figure(figsize=(12, 5))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

Code Typed

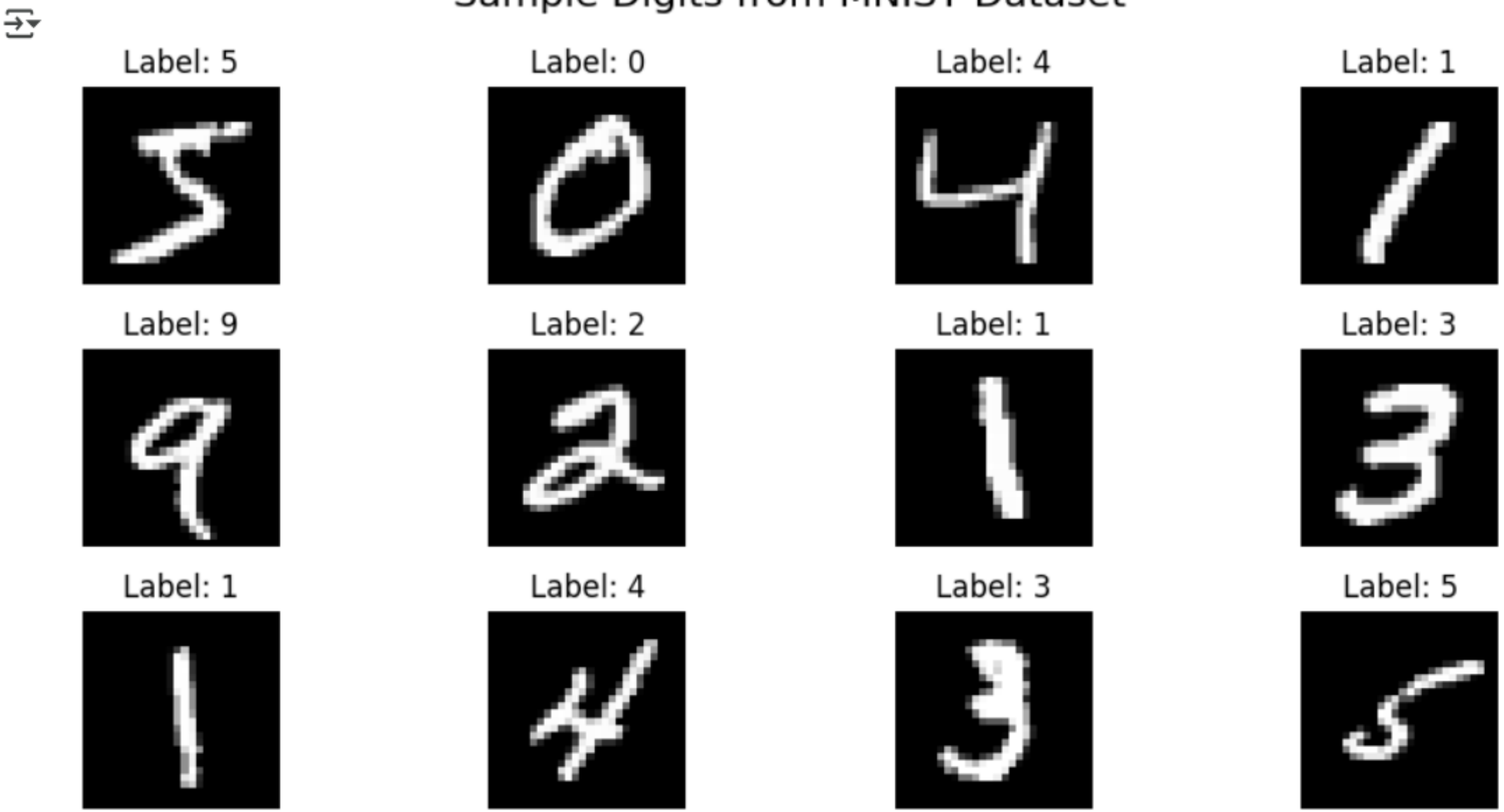
```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

# 7. Evaluate on test data
test_loss, test_acc = model.evaluate(x_test, y_test_cat)
print(f"\nTest Accuracy: {test_acc:.4f}")
```

Code Typed

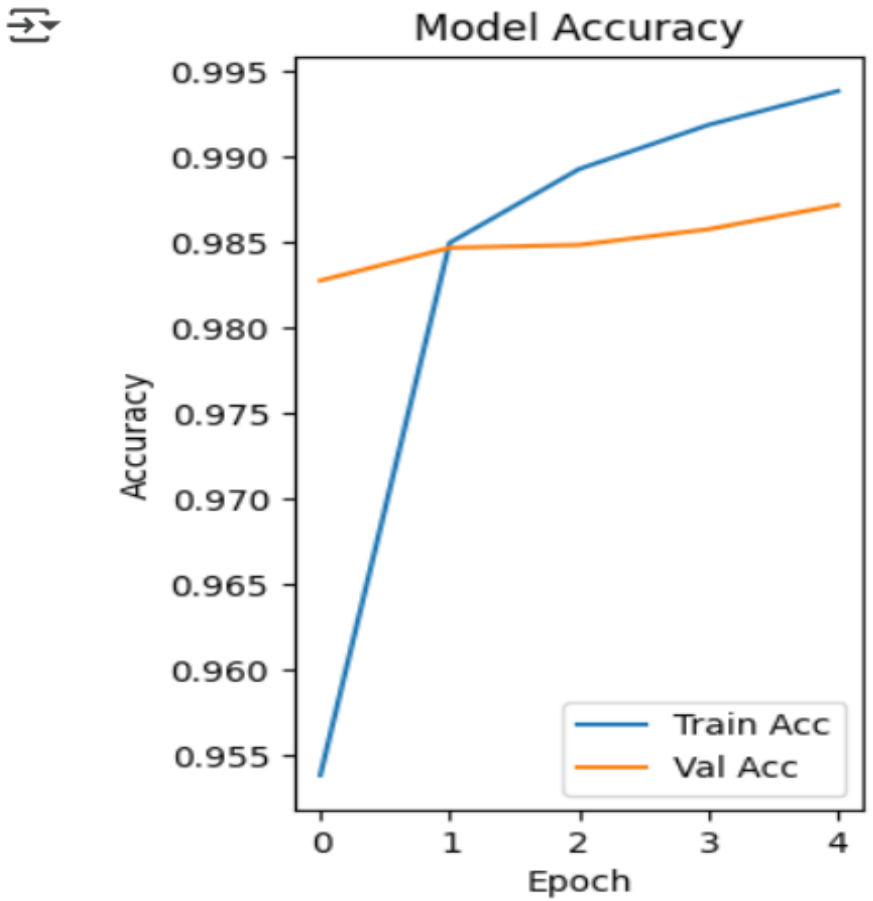
```
y_pred = model.predict(x_test).argmax(axis=1)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=range(10), yticklabels=range(10))
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# 9. Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Screenshots Output

- plt.figure(figsize=(10, 5))
- for i in range(10):
- plt.subplot(2, 5, i + 1)
- plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
- plt.title(f"True: {y_test[i]}\nPred: {y_pred[i]}")
- plt.axis('off')
- plt.tight_layout()
- plt.suptitle("Predictions on Test Images", fontsize=16)
- plt.show()


Sample Digits from MNIST Dataset
Label: 5
Label: 0
Label: 4
Label: 1
Label: 9
Label: 2
Label: 1
Label: 3
Label: 1
Label: 4
Label: 3
Label: 5

Screenshots Output



Screenshots Output