**Image Classification Using CNN (Convolutional Neural Network)**

**1. Introduction**

This project involves building a **Convolutional Neural Network (CNN)** for **image classification** using the **MNIST dataset**. The dataset consists of **28x28 grayscale images** of handwritten digits (0-9). The goal is to classify images into their respective digit categories.

**2. Dataset Overview**

* **Dataset:** MNIST (handwritten digits)
* **Classes:** 10 (Digits 0-9)
* **Total Images:** 70,000 (60,000 for training, 10,000 for testing)

**3. Implementation Steps**

**Step 1: Import Required Libraries**

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

**Step 2: Load and Explore the Dataset**

# Load the MNIST dataset

mnist = keras.datasets.mnist

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Normalize pixel values (0 to 1 range)

X\_train, X\_test = X\_train / 255.0, X\_test / 255.0

# Display dataset shape

print(f"Training Data Shape: {X\_train.shape}, Labels: {y\_train.shape}")

print(f"Testing Data Shape: {X\_test.shape}, Labels: {y\_test.shape}")

**Step 3: Visualize Sample Images**

plt.figure(figsize=(10,5))

for i in range(6):

plt.subplot(2,3,i+1)

plt.imshow(X\_train[i], cmap='gray')

plt.title(f"Label: {y\_train[i]}")

plt.axis('off')

plt.show()

**Step 4: Build a CNN Model**

# Reshape images for CNN input

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

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

# Build the CNN model

model = keras.Sequential([

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

keras.layers.MaxPooling2D(2,2),

keras.layers.Conv2D(64, (3,3), activation='relu'),

keras.layers.MaxPooling2D(2,2),

keras.layers.Flatten(),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(10, activation='softmax')

])

# Compile the model

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

# Display model summary

model.summary()

**Step 5: Train the CNN Model**

history = model.fit(X\_train, y\_train, epochs=5, validation\_data=(X\_test, y\_test))

**Step 6: Evaluate Model Performance**

# Evaluate on test set

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {test\_acc:.2f}")

# Plot training history

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

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.legend()

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.title('Training Progress')

plt.show()

**4. Results and Observations**

* **Model Accuracy:** ~99% (on training set), ~98% (on test set)
* **CNN Layers:**
  + 2 convolutional layers with max pooling
  + 1 fully connected dense layer
  + Softmax output layer (10 classes)
* **Training Performance:**
  + The model achieves **high accuracy** on handwritten digit recognition.
  + More epochs and tuning can further improve performance.

**5. Conclusion**

* **CNNs** are powerful for image classification.
* The model performs well with minimal preprocessing.
* Can be improved using **data augmentation** or **deeper architectures**.