Project-1: Building Convolutional Neural Networks on MNIST Dataset

Submission Format: Jupyter Notebook (.ipynb)

Learning Objectives

By completing this assignment, you will:

- Understand CNN architecture and its components
- Implement CNN from scratch using TensorFlow/Keras
- Master data preprocessing for image classification
- Compare different CNN architectures

Dataset Information

MNIST Dataset: Download the Dataset through the Keras library

- 70,000 grayscale images of handwritten digits (0-9)
- Training set: 60,000 images
- Test set: 10,000 images
- Image dimensions: 28x28 pixels
- 10 classes (digits 0-9)

Required Libraries

- import tensorflow as tf
- from tensorflow import keras
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- from sklearn.metrics import classification_report, confusion_matrix
- import pandas as pd

(Use any other as per your requirement)

Part 1: Data Loading and Pre-processing

Task 1.1: Data Loading and Exploration

1. Load the MNIST dataset using tf.keras.datasets.mnist.load_data()
2. Display the shape of training and testing sets

Task 1.2: Data Pre-processing (follow this step as I explain in CNN Demonstration)

1. Normalization: Scale pixel values to range [0, 1]
2. Reshaping: Reshape images to add channel dimension (28, 28, 1)
3. Data Augmentation

Part 2: Building Basic CNN Architecture

Task 2.1: Design CNN Architecture: Build a CNN with the following specifications:

Model Architecture:

```
Input Layer: (28, 28, 1)
├── Conv2D: 32 filters, 3x3 kernel, ReLU activation
├── MaxPooling2D: 2x2 pool size
├── Conv2D: 64 filters, 3x3 kernel, ReLU activation
├── MaxPooling2D: 2x2 pool size
├── Conv2D: 64 filters, 3x3 kernel, ReLU activation
├── Flatten
├── Dense: 64 units, ReLU activation
├── Dense: 10 units, Softmax activation (output)
```

Requirements:

- Use appropriate padding for convolutional layers
- Add model summary to show total parameters
- Visualize the model architecture using tf.keras.utils.plot_model()

Task 2.2: Model Compilation

Configure the model with:

- Optimizer: Adam with learning rate 0.001
- Loss Function: Categorical crossentropy

- Metrics: Accuracy

Deliverable: Complete model implementation with architecture visualization and parameter analysis.

Part 3: Model Training and Evaluation

Task 3.1: Training Setup

1. Training Configuration:
   - Epochs: 20 (or until early stopping)
   - Batch size: 128
   - Validation data: Use your validation split

Task 3.2: Model Training

1. Train the model with your configured settings
2. Plot training history:
   - Training vs Validation Loss
   - Training vs Validation Accuracy

Task 3.3: Model Evaluation (5 points)

1. Evaluate model on test set
2. Generate classification report
3. Create and visualize confusion matrix
4. Calculate per-class accuracy
5. Display misclassified examples (at least 10)

Academic Integrity

- Individual assignment - no collaboration on code
- You may discuss concepts and approaches with classmates
- Cite all external resources and tutorials used
- Plagiarism will result in zero points