Project: Simple Neural Network for Handwritten Digit Recognition

Description

The MNIST dataset consists of 70,000 images of handwritten digits, each of size 28x28 pixels. Our goal is to train a neural network to classify these images into their corresponding digit classes. The project involves the following steps:

1. **Data Preparation**: Load and preprocess the MNIST dataset.
2. **Model Creation**: Build a simple feedforward neural network.
3. **Model Training**: Train the model on the training set.
4. **Evaluation**: Evaluate the model's performance on the test set.
5. **Prediction**: Use the trained model to make predictions on new data.

Code Implementation

Below is the complete code for the project, including all the necessary steps:

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

# Step 1: Data Preparation
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# Convert labels to categorical format
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Step 2: Model Creation
# Initialize the model
model = Sequential()

# Add layers to the model
```

```
model.add(Flatten(input_shape=(28, 28)))  # Flatten the input
model.add(Dense(128, activation='relu'))  # Hidden layer with 128 neurons
model.add(Dense(10, activation='softmax')) # Output layer with 10 classes

# Step 3: Model Compilation
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Step 4: Model Training
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# Step 5: Model Evaluation
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy:.4f}')

# Step 6: Prediction
predictions = model.predict(x_test)

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(x_test[i], cmap='gray')
    plt.title(f'Predicted: {np.argmax(predictions[i])}, Actual: {np.argmax(y_test[i])}')
    plt.axis('off')
    plt.show()
```

Explanation of the Code

1. **Data Preparation**:
   - We load the MNIST dataset using `mnist.load_data()`, which returns training and test sets.
   - The pixel values are normalized to the range [0, 1] for better convergence during training.
   - Labels are converted to categorical format using `to_categorical()`.
2. **Model Creation**:
   - A sequential model is initialized, and we add layers:
     - **Flatten Layer**: Converts the 28x28 images into a 784-dimensional vector.
     - **Dense Layer**: A hidden layer with 128 neurons and ReLU activation function.
     - **Output Layer**: A dense layer with 10 neurons (for digits 0-9) and softmax activation function to output probabilities.
3. **Model Compilation**:
   - The model is compiled using the Adam optimizer and categorical cross-entropy loss function, with accuracy as a metric.
4. **Model Training**:
   - The model is trained on the training set for 10 epochs with a batch size of 32. We also use 20% of the training data for validation.
5. **Model Evaluation**:
   - The model is evaluated on the test set, and the accuracy is printed.

6. **Prediction**:
    - The model predicts the labels of the test set, and we visualize the first five test images along with their predicted and actual labels.

Conclusion

This project demonstrates how to build and train a simple neural network for digit recognition using the MNIST dataset. By following the steps and code provided, you can modify and expand upon this project to explore other datasets or enhance the neural network's architecture.