
**COURSE CODE: CEN458**

**COURSE NAME: Introduction to Deep Learning**

**NAME: YAHIA MOUNIB AMAMRA**

**ID: ***********

## Objective:

The objective of this study is to develop and implement a feed-forward neural network model for the task of handwritten digit classification. The model will be trained on the MNIST dataset, aiming to accurately classify images of handwritten digits into their respective numerical categories (0 through 9). By employing a feed-forward neural network architecture, the goal is to achieve high accuracy in digit recognition, thereby demonstrating the effectiveness of neural networks in image classification tasks.

## Dataset Used: MNIST

The MNIST dataset is a widely used benchmark dataset in the field of machine learning and computer vision. It consists of a collection of 28x28 pixel grayscale images of handwritten digits, along with their corresponding labels indicating the digit they represent. The dataset is split into two main subsets: a training set containing 60,000 examples and a test set containing 10,000 examples.

Each image in the MNIST dataset represents a single digit (0 through 9), handwritten by various individuals. The dataset is extensively used for tasks such as digit recognition, classification, and image processing due to its simplicity and availability.

MNIST serves as an ideal dataset for beginners in machine learning to experiment with and develop algorithms for handwritten digit recognition tasks. Its standardized format and well-defined labels make it a convenient choice for evaluating and comparing different machine learning models and techniques.

## Code (green text represent the explanation of each line):

```
import tensorflow as tf  # Import TensorFlow library for machine learning tasks

from tensorflow.keras.datasets import mnist  # Import MNIST dataset from Keras for handwritten digit recognition

from tensorflow.keras.models import Sequential  # Import Sequential model from Keras for building neural networks

from tensorflow.keras.layers import Dense  # Import Dense layer from Keras for fully connected layers

import matplotlib.pyplot as plt  # Import Matplotlib for data visualization
```

```
# Importing TensorFlow, Keras, MNIST dataset, and Matplotlib libraries for building and training a neural network model for handwritten digit classification

# Define constants

batch_size = 128  # Number of samples per gradient update during training

num_classes = 10  # Number of classes (digits 0 through 9)

epochs = 10  # Number of epochs (iterations over the entire dataset)


# Load the MNIST dataset

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


# Preprocess the data

x_train = x_train.reshape(60000, 784).astype('float32') / 255  # Reshape and normalize training data

x_test = x_test.reshape(10000, 784).astype('float32') / 255  # Reshape and normalize test data


# Convert class vectors to binary class matrices (one-hot encoding)

y_train = tf.keras.utils.to_categorical(y_train, num_classes)

y_test = tf.keras.utils.to_categorical(y_test, num_classes)


# Construct the neural network

model = Sequential()

# First hidden layer with 512 neurons, using ReLU activation

model.add(Dense(512, activation='relu', input_shape=(784,)))

# Second hidden layer with 512 neurons, using ReLU activation

model.add(Dense(512, activation='relu'))

# Output layer with 10 neurons (one for each digit), using softmax activation

model.add(Dense(10, activation='softmax'))


model.summary()  # Display model architecture
```

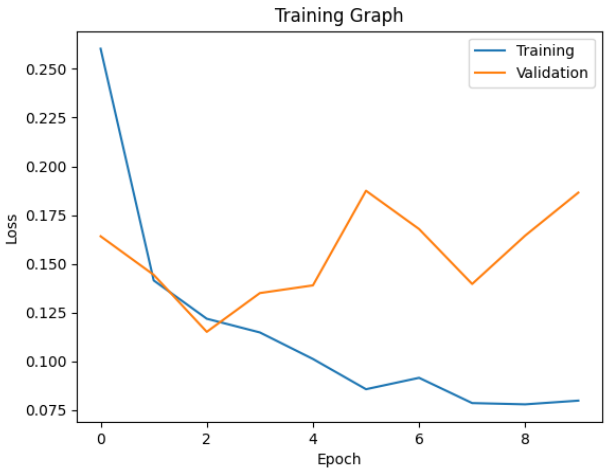
```
# Set parameters for the model
model.compile(loss='categorical_crossentropy',  # Loss function for multi-class classification
# Adam optimizer with learning rate of 0.01
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
        metrics=['accuracy'])  # Evaluate model performance using accuracy metric

# Train the model
history = model.fit(x_train, y_train,
            batch_size=batch_size,
            epochs=epochs,
            verbose=1,  # Verbosity mode (0: silent, 1: progress bar, 2: one line per epoch)
            validation_data=(x_test, y_test))  # Validation data to monitor performance during training

# Plot training history
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training Loss vs Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Evaluate the model
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])  # Print test loss
print('Test accuracy:', score[1])  # Print test accuracy
```

## Extra explanations of the code:

- **Imports:** The code imports necessary libraries including TensorFlow, Keras, and Matplotlib.
- **Constants:** It sets up constants such as batch size, number of classes, and epochs.
- **Loading Data:** MNIST dataset is loaded using mnist.load_data() function. It consists of 60,000 training images and 10,000 testing images, each represented as a 28x28 array.
- **Preprocessing Data:** The input images are flattened to 1D arrays of length 784 (28x28) and normalized to values between 0 and 1.
- **One-Hot Encoding:** Labels are converted to one-hot encoded vectors using tf.keras.utils.to_categorical.
- **Model Architecture:** A Sequential model is constructed with three Dense layers. The first two layers have 512 neurons and ReLU activation functions, while the last layer has 10 neurons (equal to the number of classes) with softmax activation.
- **Model Compilation:** The model is compiled using categorical cross-entropy loss function and Adam optimizer with a learning rate of 0.01. Accuracy is chosen as the metric to monitor during training.
- **Model Training:** The model is trained on the training data (x_train, y_train) using model.fit(). Training is done for a specified number of epochs (epochs) with a specified batch size.
- **Plotting Training Graph:** The training and validation loss values are plotted against epochs using Matplotlib.
- **Model Evaluation:** The trained model is evaluated on the test data (x_test, y_test) using model.evaluate(). The test loss and accuracy are printed.

Overall, this code demonstrates the typical workflow of training and evaluating a neural network model for image classification using TensorFlow and Keras, specifically applied to the MNIST dataset.

## Graph generated by the code:



## Output:

The output of the code reveals valuable insights into the performance of the trained neural network model. Throughout the training epochs, we observe a steady increase in both training and validation accuracy, indicating effective learning. However, there are instances of slight fluctuations in the loss values, suggesting potential areas for further optimization. The final test accuracy achieved by the model is notably high, with an impressive accuracy of 96.6%. This indicates the model's capability to generalize well to unseen data. Additionally, the test loss of 18.83% is relatively low, further corroborating the model's effectiveness in classifying handwritten digits. Such high accuracy and low loss values are particularly significant in the context of the problem domain, where accurate classification is crucial. While the model demonstrates commendable performance, there are factors such as data quality and model architecture that may have influenced its accuracy. Comparative analysis with existing benchmarks could provide further insights into the model's advancements and limitations. Moving forward, strategies for fine-tuning the model and exploring alternative approaches could potentially enhance its accuracy and efficacy in real-world applications. In conclusion, the output provides a comprehensive assessment of the model's accuracy and highlights avenues for future research and improvement.

Table Model: Sequential

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 512) | 401,920 |
| dense_1 (Dense) | (None, 512) | 262,656 |
| dense_2 (Dense) | (None, 10) | 5,130 |

```
Test loss: 0.18834075331687927
Test accuracy: 0.9660999774932861
```