exercise_1

November 5, 2025

1 Exercise-1: Training Deep Neural Network on MNIST

Train a controlled deep neural network on the MNIST dataset. Set random seeds to 42. Load and preprocess MNIST. Build the network using the following configuration: - Flatten input images to $28 \times 28 = 784$ features - 3 hidden layers, 64 neurons each - ELU activation function - He normal initialization - Output layer: 10 neurons with softmax - Optimizer: Nadam - learning_rate = 0.001, loss=sparse_categorical_crossentropy - EarlyStopping callback: monitor validation loss, patience = 5, restore best weights - epochs = 50, batch size = 32 - Use only the first 1000 training samples and first 200 test samples

1.1 Q1.1 Report the obtained test accuracy.

[ ]:
```python
# Load MNIST dataset
from tensorflow.keras.datasets import mnist

RANDOM_SEED = 42
hidden_layers = 3
hidden_neurons = 64
output_neurons = 10
learning_rate = 0.001

# Load data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess data, normalize and reshape
x_train = x_train.reshape(-1, 28 * 28).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28 * 28).astype('float32') / 255.0

# Use only the first 1000 training samples and first 200 test samples
x_train = x_train[:1000]
y_train = y_train[:1000]
x_test = x_test[:200]
y_test = y_test[:200]

import tensorflow as tf
tf.random.set_seed(RANDOM_SEED)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Nadam
from tensorflow.keras.callbacks import EarlyStopping

# Build the model
model = Sequential()
# Input layer
model.add(Dense(hidden_neurons, activation='elu',␣
 ↪kernel_initializer='he_normal', input_shape=(28 * 28,)))
# Hidden layers, 3 layers with ELU activation and He normal initialization
for _ in range(hidden_layers - 1):
    model.add(Dense(hidden_neurons, activation='elu',␣
 ↪kernel_initializer='he_normal'))
# Output layer with softmax activation
model.add(Dense(output_neurons, activation='softmax'))
model.compile(optimizer=Nadam(learning_rate=learning_rate),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


# Early stopping callback
earl = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Train the model
history = model.fit(x_train,
                    y_train,
                    epochs=50,
                    batch_size=32,
                    validation_split=0.2,
                    callbacks=[earl],
                    verbose=1)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
Epoch 1/50
25/25                1s 4ms/step -
accuracy: 0.5200 - loss: 1.5309 - val_accuracy: 0.7900 - val_loss: 0.8540
Epoch 2/50
25/25                0s 1ms/step -
accuracy: 0.8425 - loss: 0.5828 - val_accuracy: 0.8600 - val_loss: 0.5678
Epoch 3/50
25/25                0s 1ms/step -
accuracy: 0.8950 - loss: 0.3513 - val_accuracy: 0.8650 - val_loss: 0.5182
Epoch 4/50
25/25                0s 1ms/step -
accuracy: 0.9287 - loss: 0.2478 - val_accuracy: 0.8650 - val_loss: 0.5160
Epoch 5/50
```

```
25/25            0s 1ms/step -
accuracy: 0.9575 - loss: 0.1807 - val_accuracy: 0.8650 - val_loss: 0.5234
Epoch 6/50
25/25            0s 1ms/step -
accuracy: 0.9737 - loss: 0.1324 - val_accuracy: 0.8600 - val_loss: 0.5344
Epoch 7/50
25/25            0s 1ms/step -
accuracy: 0.9837 - loss: 0.0962 - val_accuracy: 0.8500 - val_loss: 0.5520
Epoch 8/50
25/25            0s 1ms/step -
accuracy: 0.9887 - loss: 0.0681 - val_accuracy: 0.8450 - val_loss: 0.5784
Epoch 9/50
25/25            0s 1ms/step -
accuracy: 0.9975 - loss: 0.0476 - val_accuracy: 0.8500 - val_loss: 0.6056
Test Accuracy: 84.50%
```