lab-2

January 29, 2025

[1]:
```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load and preprocess the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values to range [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0

# Reshape the data to match the expected input shape of the model (28x28x1)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Convert labels to one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

print(f"Training data shape: {x_train.shape}, Test data shape: {x_test.shape}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434             192s
17us/step
Training data shape: (60000, 28, 28, 1), Test data shape: (10000, 28, 28, 1)
```

[2]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import layers, models

# Load and preprocess the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values to range [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
# Reshape the data to match the expected input shape of the model (28x28x1)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Convert labels to one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

print(f"Training data shape: {x_train.shape}, Test data shape: {x_test.shape}")

# Build a simple CNN model (baseline)
model_baseline = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model_baseline.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Train the baseline model
history_baseline = model_baseline.fit(x_train, y_train, epochs=10,␣
 ↪validation_data=(x_test, y_test))

# Evaluate the model
loss, accuracy = model_baseline.evaluate(x_test, y_test)
print(f'Baseline Model - Loss: {loss}, Accuracy: {accuracy}')

# Extract training and validation accuracy and loss from the history object
history_dict = history_baseline.history
train_acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']
train_loss = history_dict['loss']
val_loss = history_dict['val_loss']

# Plot the training and validation accuracy and loss
plt.figure(figsize=(12, 6))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(range(1, 11), train_acc, label='Training Accuracy')
plt.plot(range(1, 11), val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
```

```
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(range(1, 11), train_loss, label='Training Loss')
plt.plot(range(1, 11), val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Training data shape: (60000, 28, 28, 1), Test data shape: (10000, 28, 28, 1)

c:\Users\eakes\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Epoch 1/10
1875/1875             13s 6ms/step -
accuracy: 0.9052 - loss: 0.3061 - val_accuracy: 0.9867 - val_loss: 0.0422
Epoch 2/10
1875/1875             9s 5ms/step -
accuracy: 0.9870 - loss: 0.0423 - val_accuracy: 0.9887 - val_loss: 0.0337
Epoch 3/10
1875/1875             10s 5ms/step -
accuracy: 0.9911 - loss: 0.0273 - val_accuracy: 0.9910 - val_loss: 0.0285
Epoch 4/10
1875/1875             11s 6ms/step -
accuracy: 0.9936 - loss: 0.0200 - val_accuracy: 0.9911 - val_loss: 0.0268
Epoch 5/10
1875/1875             10s 5ms/step -
accuracy: 0.9957 - loss: 0.0133 - val_accuracy: 0.9899 - val_loss: 0.0331
Epoch 6/10
1875/1875             11s 6ms/step -
accuracy: 0.9966 - loss: 0.0124 - val_accuracy: 0.9899 - val_loss: 0.0314
Epoch 7/10
1875/1875             11s 6ms/step -
accuracy: 0.9969 - loss: 0.0089 - val_accuracy: 0.9889 - val_loss: 0.0405
Epoch 8/10
1875/1875             10s 5ms/step -
accuracy: 0.9973 - loss: 0.0084 - val_accuracy: 0.9904 - val_loss: 0.0310
```

```
Epoch 9/10
1875/1875             10s 5ms/step -
accuracy: 0.9977 - loss: 0.0072 - val_accuracy: 0.9913 - val_loss: 0.0299
Epoch 10/10
1875/1875             11s 6ms/step -
accuracy: 0.9986 - loss: 0.0041 - val_accuracy: 0.9921 - val_loss: 0.0344
313/313             1s 3ms/step -
accuracy: 0.9891 - loss: 0.0477
Baseline Model - Loss: 0.03439216688275337, Accuracy: 0.9921000003814697
```