LAB3

February 10, 2025

0.0.1 3.A: Task 1: Build a CNN Model for MNIST Classification

[1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt

# 1.1 Load and Preprocess the Dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

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

# Reshape dataset to include channel dimension
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Convert labels to categorical format
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

[2]:
```
# 1.2 Define a Basic CNN Model
model = keras.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')
])
```

```
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)
```

[3]:
```
# 1.3 Compile and Train the Model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=5, validation_data=(x_test,␣
 ↪y_test))
```

```
Epoch 1/5
1875/1875             14s 6ms/step -
accuracy: 0.9127 - loss: 0.2876 - val_accuracy: 0.9856 - val_loss: 0.0472
Epoch 2/5
1875/1875             10s 5ms/step -
accuracy: 0.9852 - loss: 0.0449 - val_accuracy: 0.9877 - val_loss: 0.0377
Epoch 3/5
1875/1875             10s 5ms/step -
accuracy: 0.9907 - loss: 0.0282 - val_accuracy: 0.9895 - val_loss: 0.0314
Epoch 4/5
1875/1875             10s 5ms/step -
accuracy: 0.9936 - loss: 0.0205 - val_accuracy: 0.9882 - val_loss: 0.0346
Epoch 5/5
1875/1875             15s 8ms/step -
accuracy: 0.9952 - loss: 0.0151 - val_accuracy: 0.9900 - val_loss: 0.0334
```

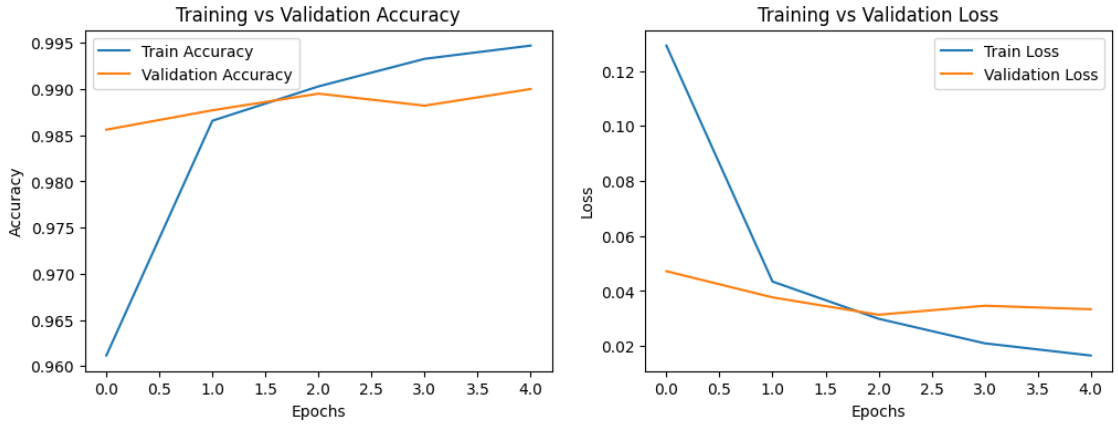
[4]:
```
# 1.4 Evaluate the Model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc:.4f}")

# Plot Training vs Validation Accuracy and Loss
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')
plt.show()
```

```
313/313 - 2s - 6ms/step - accuracy: 0.9900 - loss: 0.0334
```

```
Test accuracy: 0.9900
```



0.0.2 3.b) Task 2: Improve CNN Model Using Regularization and Batch Normalization

[6]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers
import matplotlib.pyplot as plt

# 1.1 Load and Preprocess the Dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

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

# Reshape dataset to include channel dimension
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Convert labels to categorical format
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

[7]:
```
# 1.1 Load and Preprocess the Dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

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

# Reshape dataset to include channel dimension
x_train = x_train.reshape(-1, 28, 28, 1)
```

```
x_test = x_test.reshape(-1, 28, 28, 1)

# Convert labels to categorical format
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

[8]:
```
# 2.1 Define an Improved CNN Model with Regularization and Batch Normalization
model = keras.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', kernel_regularizer=regularizers.
 ↪l2(0.001), input_shape=(28,28,1)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2,2)),
    layers.Dropout(0.3),
    layers.Conv2D(64, (3,3), activation='relu', kernel_regularizer=regularizers.
 ↪l2(0.001)),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2,2)),
    layers.Dropout(0.3),
    layers.Flatten(),
    layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.
 ↪001)),
    layers.Dropout(0.5),
    layers.Dense(10, activation='softmax')
])
```

[9]:
```
# 2.2 Compile and Train the Improved Model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test,␣
 ↪y_test))
```

```
Epoch 1/10
1875/1875             20s 9ms/step -
accuracy: 0.8126 - loss: 0.8975 - val_accuracy: 0.9819 - val_loss: 0.2577
Epoch 2/10
1875/1875             17s 9ms/step -
accuracy: 0.9585 - loss: 0.3322 - val_accuracy: 0.9882 - val_loss: 0.2130
Epoch 3/10
1875/1875             17s 9ms/step -
accuracy: 0.9651 - loss: 0.2894 - val_accuracy: 0.9859 - val_loss: 0.2113
Epoch 4/10
1875/1875             18s 10ms/step -
accuracy: 0.9688 - loss: 0.2720 - val_accuracy: 0.9873 - val_loss: 0.2094
Epoch 5/10
1875/1875             18s 10ms/step -
accuracy: 0.9717 - loss: 0.2605 - val_accuracy: 0.9881 - val_loss: 0.1944
```

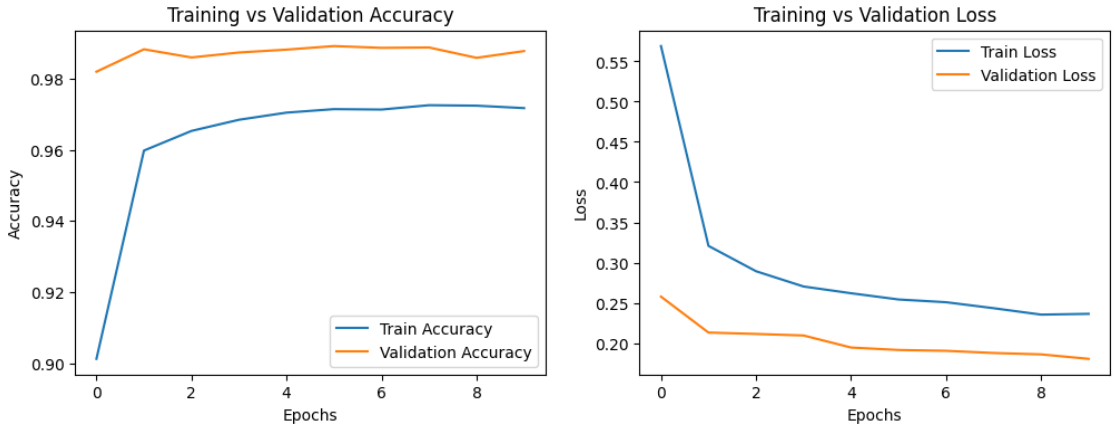
```
Epoch 6/10
1875/1875             18s 10ms/step -
accuracy: 0.9715 - loss: 0.2532 - val_accuracy: 0.9891 - val_loss: 0.1914
Epoch 7/10
1875/1875             20s 11ms/step -
accuracy: 0.9721 - loss: 0.2469 - val_accuracy: 0.9886 - val_loss: 0.1903
Epoch 8/10
1875/1875             20s 11ms/step -
accuracy: 0.9730 - loss: 0.2431 - val_accuracy: 0.9887 - val_loss: 0.1876
Epoch 9/10
1875/1875             20s 11ms/step -
accuracy: 0.9731 - loss: 0.2357 - val_accuracy: 0.9858 - val_loss: 0.1859
Epoch 10/10
1875/1875             21s 11ms/step -
accuracy: 0.9737 - loss: 0.2300 - val_accuracy: 0.9877 - val_loss: 0.1803
```

[10]:
```python
# 2.3 Evaluate the Model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc:.4f}")

# Plot Training vs Validation Accuracy and Loss
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')
plt.show()
```

```
313/313 - 1s - 3ms/step - accuracy: 0.9877 - loss: 0.1803
Test accuracy: 0.9877
```


Training vs Validation Accuracy
Accuracy
Epochs
Train Accuracy
Validation Accuracy
0.90
0.92
0.94
0.96
0.98
0
2
4
6
8
Training vs Validation Loss
Loss
Epochs
Train Loss
Validation Loss
0.20
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0
2
4
6
8