In-Lab

Task 1

Code:

```
#Import necessary libraries
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense
from keras.datasets import mnist
from keras.utils import to_categorical
```

In-Lab

Task 2

Code:

```

# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
# Reshape and normalize the images
train_images = train_images.reshape((60000, 28, 28,
1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28,
1)).astype('float32') / 255
```

Code:

```

# One-hot encode the labels
train_labels = to_categorical (train_labels)
test_labels = to_categorical(test_labels)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

Task 3

Code:

```
# Build the CNN model
model = Sequential()
```

```
# Step 1: Convolutional Layer with ReLU activation
model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)))
# Step 2: Max Pooling Layer
model.add(MaxPooling2D((2, 2)))
# Step 3: Convolutional Layer with ReLU activation
model.add(Conv2D(64, (3, 3), activation='relu'))
# Step 4: Max Pooling Layer
model.add(MaxPooling2D((2, 2)))
# Step 5: Flatten Layer
model.add(Flatten())
# Step 6: Dense (Fully Connected) Layer with ReLU
activation
model.add(Dense (64, activation='relu'))
# Step 7: Output Layer with Softmax activation (for multi-
class classification)
model.add(Dense (10, activation='softmax'))
```

Task 4

Code:

```
# Compile the model
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

Task 5

Code:

```
#Train the model
model.fit(train_images, train_labels, epochs=5, batch_size=64,
validation_data=(test_images,
test_labels))
```

```
Epoch 1/5
938/938 [==============================] - 54s 57ms/step - loss: 0.1680 - accuracy: 0.9503 - val_loss: 0.0533 - val_accuracy: 0.9841
Epoch 2/5
938/938 [==============================] - 52s 55ms/step - loss: 0.0532 - accuracy: 0.9836 - val_loss: 0.0408 - val_accuracy: 0.9873
Epoch 3/5
938/938 [==============================] - 54s 58ms/step - loss: 0.0376 - accuracy: 0.9887 - val_loss: 0.0393 - val_accuracy: 0.9874
Epoch 4/5
938/938 [==============================] - 54s 57ms/step - loss: 0.0283 - accuracy: 0.9910 - val_loss: 0.0314 - val_accuracy: 0.9894
Epoch 5/5
938/938 [==============================] - 53s 57ms/step - loss: 0.0223 - accuracy: 0.9930 - val_loss: 0.0318 - val_accuracy: 0.9904
<keras.src.callbacks.History at 0x7fa91757cdc0>
```

Task 6

Code:

```
# Evaluate the model on the test set
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print (f'Test Accuracy: {test_acc}')
```

Result :

```
Epoch 1/5
938/938 [==============================] - 49s 52ms/step -
loss: 0.1702 - accuracy: 0.9485 - val_loss: 0.0595 -
val_accuracy: 0.9808
Epoch 2/5
938/938 [==============================] - 50s 53ms/step -
loss: 0.0536 - accuracy: 0.9834 - val_loss: 0.0316 -
val_accuracy: 0.9894
Epoch 3/5
938/938 [==============================] - 47s 50ms/step -
loss: 0.0374 - accuracy: 0.9884 - val_loss: 0.0340 -
val_accuracy: 0.9888
Epoch 4/5
938/938 [==============================] - 49s 53ms/step -
loss: 0.0277 - accuracy: 0.9911 - val_loss: 0.0374 -
val_accuracy: 0.9884
Epoch 5/5
938/938 [==============================] - 48s 52ms/step -
loss: 0.0216 - accuracy: 0.9932 - val_loss: 0.0303 -
val_accuracy: 0.9896
313/313 [==============================] - 3s 9ms/step -
loss: 0.0303 - accuracy: 0.9896
Test Accuracy: 0.9896000027656555
```