In [20]:
```
# !pip install tensorflow
# !pip install plot_keras_history
```

In [21]:
```
import mlflow
import numpy as np

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
from plot_keras_history import show_history

from tensorflow.keras import regularizers

from tensorflow.keras.callbacks import ModelCheckpoint
import h5py
```

In [22]:
```
mlflow.set_experiment("BDL Assignment 5")
```

```
2024/04/21 14:55:24 INFO mlflow.tracking.fluent: Experiment with name 'BDL Assignment 5' does not exist. Creating a n
ew experiment.
```

Out[22]:
```
<Experiment: artifact_location='file:///d:/sem%208/CS5830/Assignment%205/mlruns/724536142507697262', creation_time=17
13691524826, experiment_id='724536142507697262', last_update_time=1713691524826, lifecycle_stage='active', name='BDL
Assignment 5', tags={}>
```

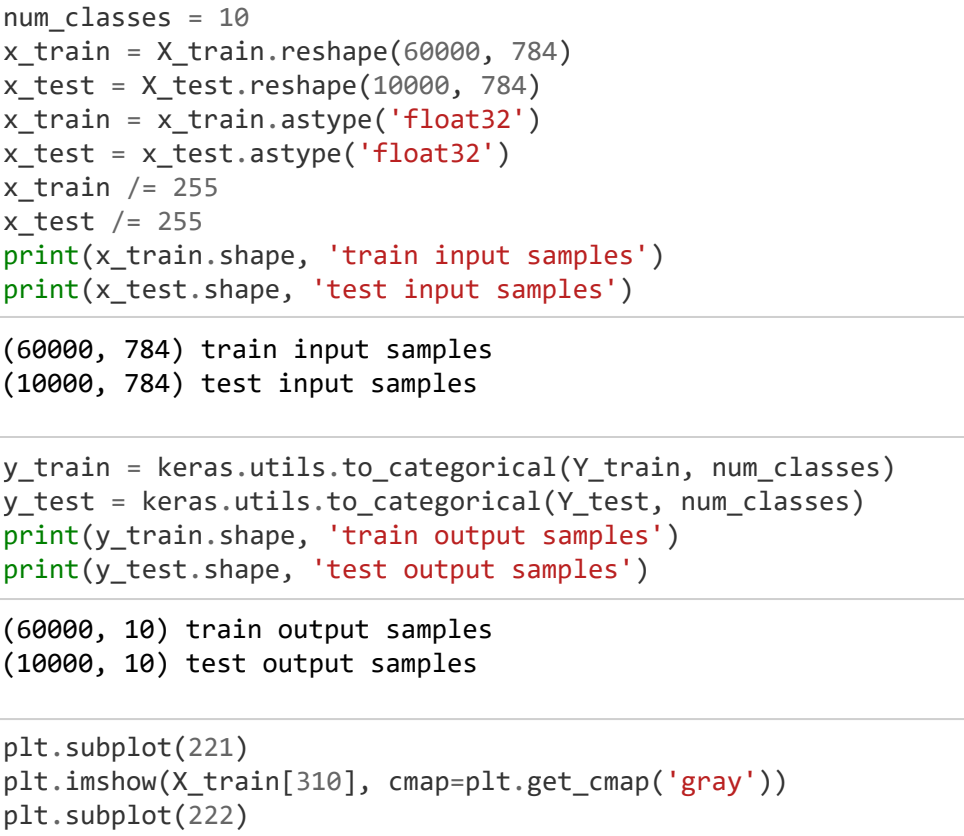
In [23]:
```
(X_train, Y_train), (X_test, Y_test) = keras.datasets.mnist.load_data()
num_classes = 10
x_train = X_train.reshape(60000, 784)
x_test = X_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape, 'train input samples')
print(x_test.shape, 'test input samples')
```

```
(60000, 784) train input samples
(10000, 784) test input samples
```

In [24]:
```
y_train = keras.utils.to_categorical(Y_train, num_classes)
y_test = keras.utils.to_categorical(Y_test, num_classes)
print(y_train.shape, 'train output samples')
print(y_test.shape, 'test output samples')
```

```
(60000, 10) train output samples
(10000, 10) test output samples
```

In [25]:
```
plt.subplot(221)
plt.imshow(X_train[310], cmap=plt.get_cmap('gray'))
plt.subplot(222)
plt.imshow(X_train[515], cmap=plt.get_cmap('gray'))
plt.subplot(223)
plt.imshow(X_train[1210], cmap=plt.get_cmap('gray'))
plt.subplot(224)
plt.imshow(X_train[2150], cmap=plt.get_cmap('gray'))
plt.show()
```

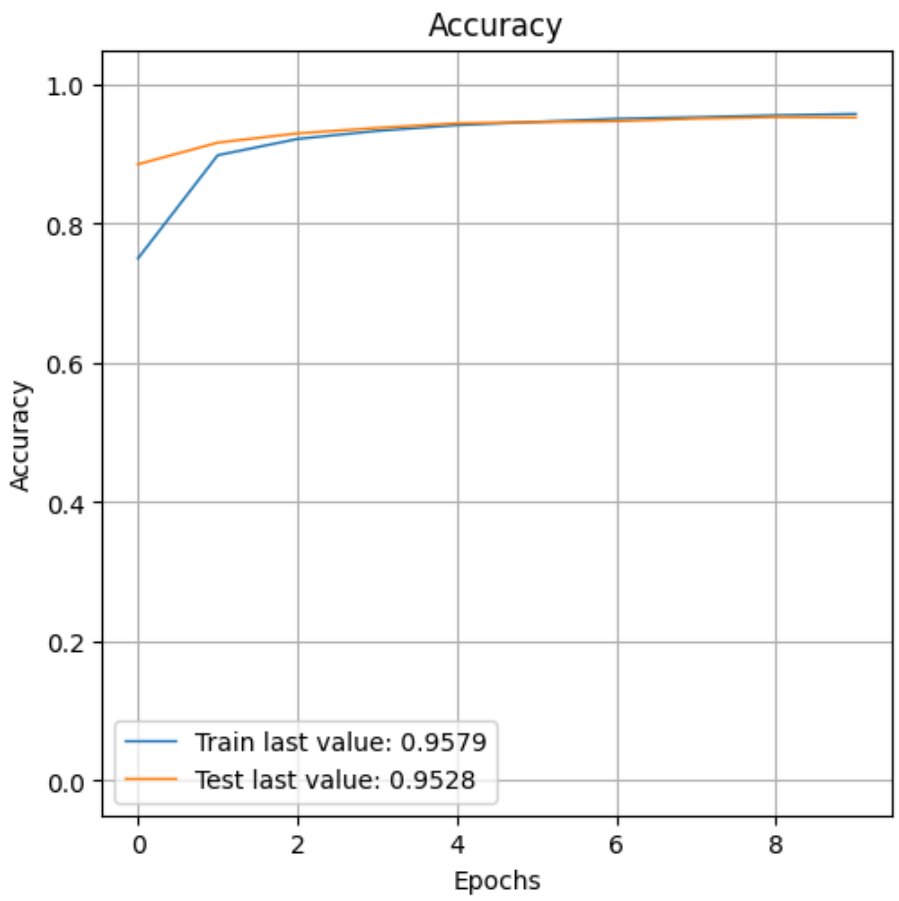


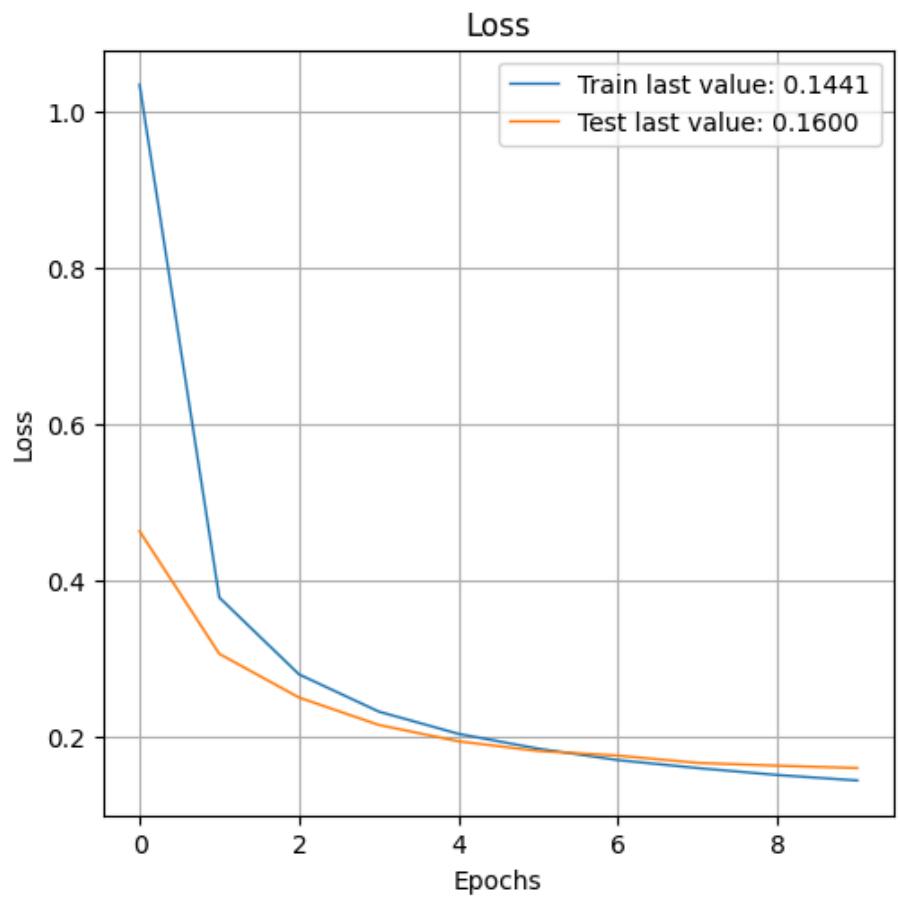
In [26]:
```
def build_basic_nn(params):
    mlp = Sequential([
        Dense(params["layer1_size"], activation=params['activation'], kernel_regularizer=params['regularizers'], input
_shape=(784,)),
        Dropout(params["dropout_rate_l1"]),
        Dense(params["layer2_size"], activation=params['activation'], kernel_regularizer=params['regularizers']),
        Dropout(params["dropout_rate_l2"]),
        Dense(params["output_size"], activation='softmax')
    ])
    return mlp

def train_mlp(mlp, train_params, x_train, y_train):

    if train_params["use_optimiser"] == True:
        opt_new = keras.optimizers.SGD(learning_rate=train_params["learning_rate"], momentum=train_params["momentum"])
        mlp.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=opt_new)
    else:
        mlp.compile(loss='categorical_crossentropy', metrics=['accuracy'])



    if train_params["early_stopping"] == True:
        checkpoint = ModelCheckpoint(r"mnist-epoch-{epoch:02d}.keras")
        history = mlp.fit(x_train, y_train, epochs=train_params["num_epochs"], validation_data=(x_test, y_test), callb
acks=[checkpoint])
        es = keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=2)
        mlp.fit(x_train, y_train, epochs=train_params["num_epochs"], validation_data=(x_test, y_test), callbacks=[es],
batch_size=train_params["batch_size"])
    else:
        history = mlp.fit(x_train, y_train, epochs=train_params["num_epochs"], validation_data=(x_test, y_test), batch
_size=train_params["batch_size"])

    show_history(history)

    return mlp


def mlp_mlflow_run(
    name,
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
):
    with mlflow.start_run(run_name=name):
        mlflow.log_params(mlp_params)
        mlflow.log_params(train_params)
        mlflow.set_tag("model_name", "MLP")

        model = build_basic_nn(mlp_params)
        model = train_mlp(model, train_params, x_train, y_train)

        loss, acc = model.evaluate(x_train, y_train, verbose=2)
        print("Train accuracy: {:5.2f}%".format(100*acc))

        mlflow.log_metric("Train Loss", loss)
        mlflow.log_metric("Train Accuracy", acc)

        loss, acc = model.evaluate(x_test, y_test, verbose=2)
        print("Test accuracy: {:5.2f}%".format(100*acc))

        mlflow.log_metric("Test Loss", loss)
        mlflow.log_metric("Test Accuracy", acc)

        mlflow.tensorflow.log_model(model, "Basic Neural Network")

        test_pt = 782
        plt.imshow(X_test[test_pt], cmap=plt.get_cmap('gray'))
        probs = model.predict(x_test[test_pt:test_pt+1], verbose=True)
        print("Predicted Digit:", np.argmax(probs))
        plt.title('Predicted Image')
        plt.savefig("Predicted_Image.png")
        mlflow.log_artifact("Predicted_Image.png")
        plt.show()
```

## Basic Neural Network

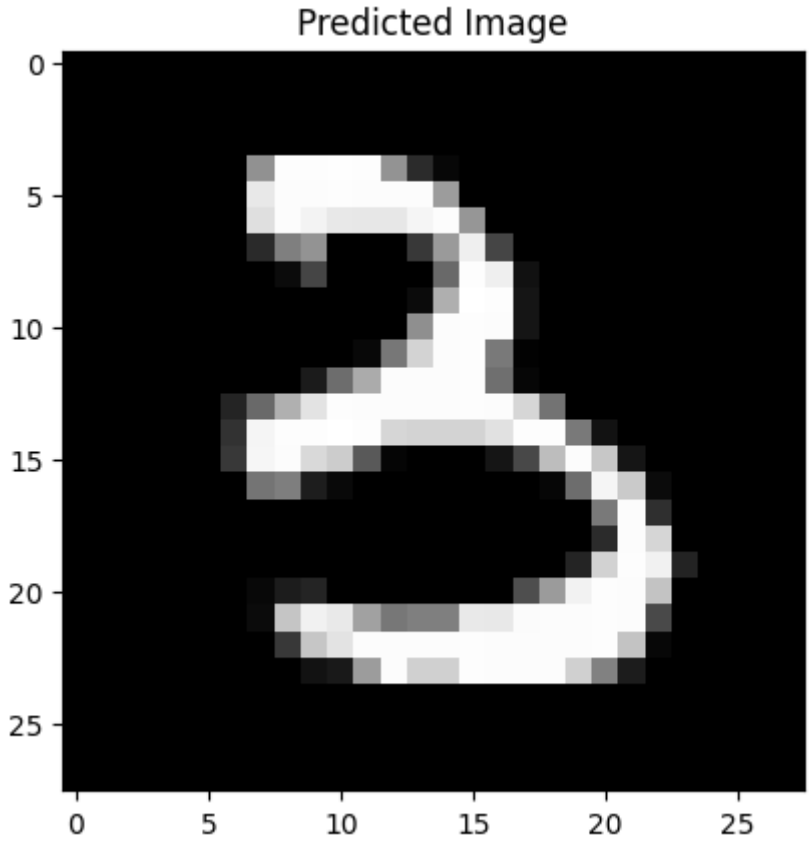
In [27]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 20,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=False,
    learning_rate=0.1,
    momentum=0.0,
    num_epochs=10,
    early_stopping= False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_base",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.5988 - loss: 1.5340 - val_accuracy: 0.8856 - val_loss: 0.463
0
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8880 - loss: 0.4224 - val_accuracy: 0.9167 - val_loss: 0.306
0
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9166 - loss: 0.3023 - val_accuracy: 0.9300 - val_loss: 0.250
1
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9328 - loss: 0.2400 - val_accuracy: 0.9379 - val_loss: 0.215
1
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9411 - loss: 0.2068 - val_accuracy: 0.9445 - val_loss: 0.194
2
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9470 - loss: 0.1862 - val_accuracy: 0.9462 - val_loss: 0.181
8
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9513 - loss: 0.1712 - val_accuracy: 0.9475 - val_loss: 0.175
9
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9525 - loss: 0.1618 - val_accuracy: 0.9512 - val_loss: 0.166
6
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9568 - loss: 0.1478 - val_accuracy: 0.9534 - val_loss: 0.163
0
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9590 - loss: 0.1405 - val_accuracy: 0.9528 - val_loss: 0.160
0
```



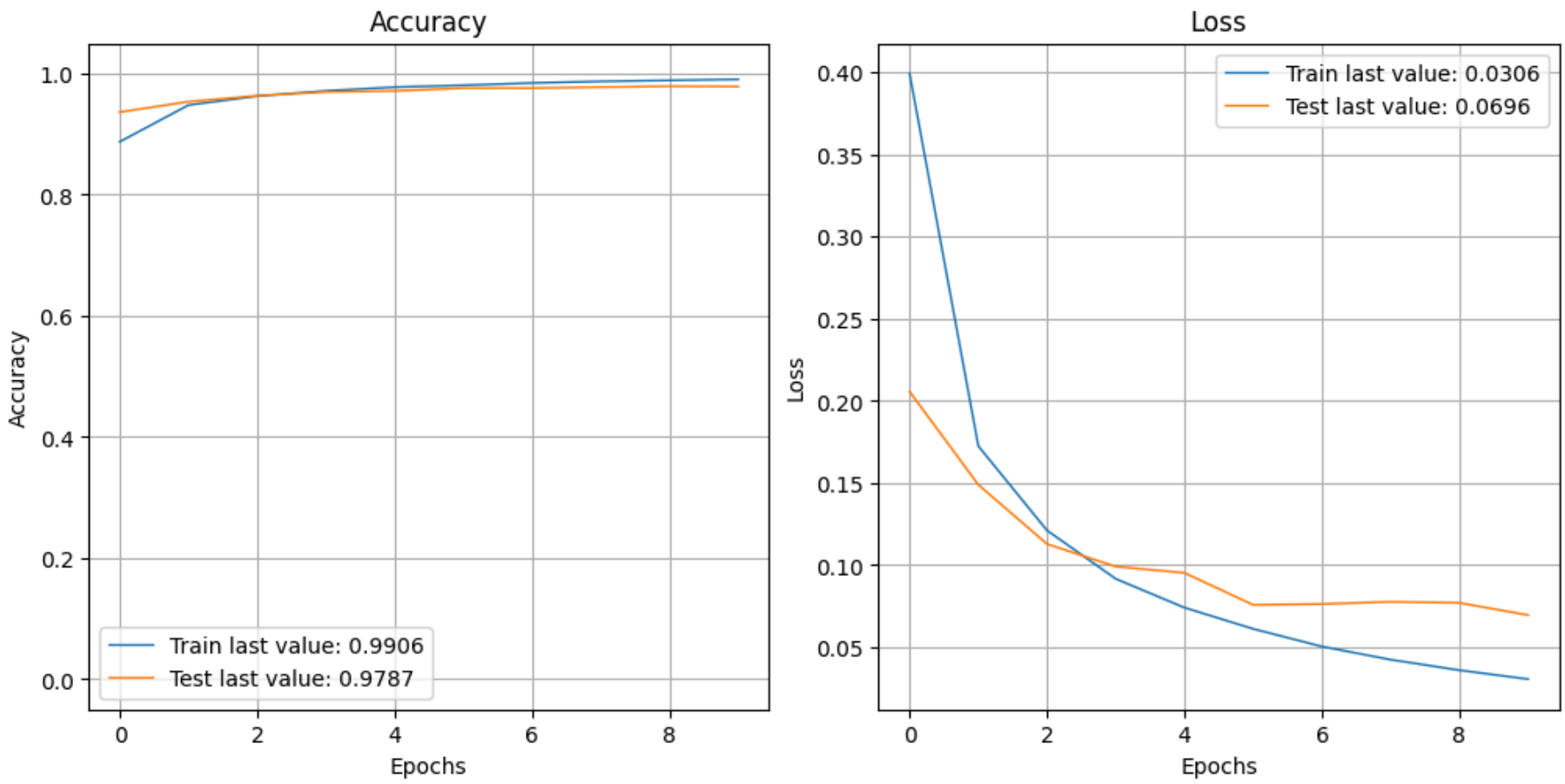

```
1875/1875 - 2s - 1ms/step - accuracy: 0.9612 - loss: 0.1338
Train accuracy: 96.12%
313/313 - 0s - 1ms/step - accuracy: 0.9528 - loss: 0.1600

2024/04/21 14:55:56 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 95.28%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Predicted Digit: 3
```


Predicted Image
0
5
10
15
20
25
0
5
10
15
20
25

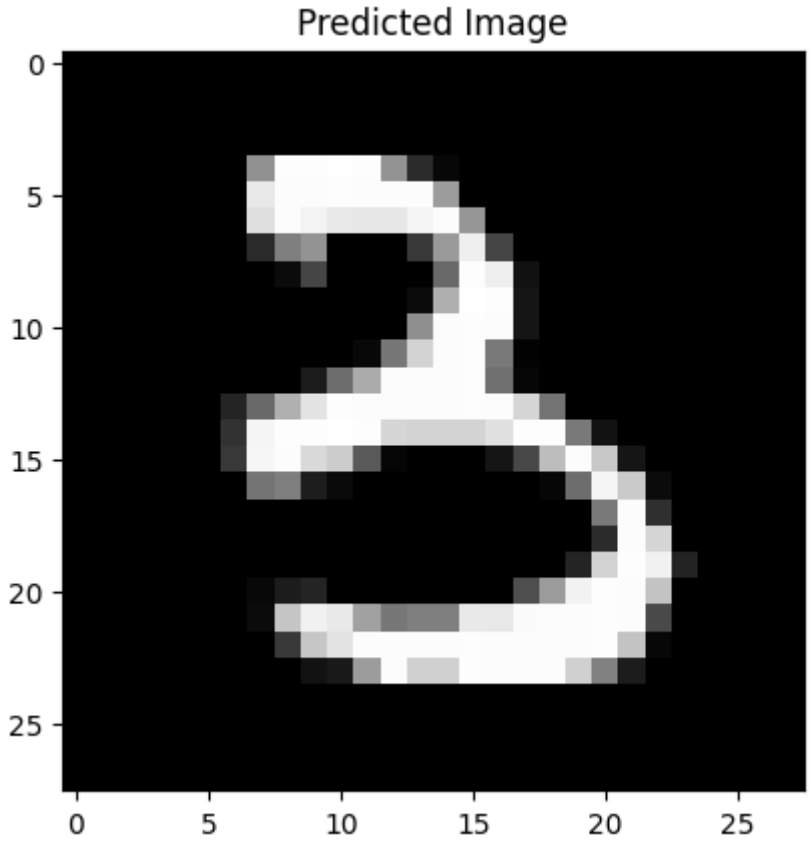
## Bigger Neural Network

In [28]:
```
mlp_params = {
    "layer1_size": 256,
    "dropout_rate_l1": 0,
    "layer2_size": 128,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=False,
    learning_rate=0.1,
    momentum=0.0,
    num_epochs=10,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_big_network",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8041 - loss: 0.7051 - val_accuracy: 0.9365 - val_loss: 0.205
6
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9425 - loss: 0.1878 - val_accuracy: 0.9538 - val_loss: 0.149
0
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9596 - loss: 0.1315 - val_accuracy: 0.9635 - val_loss: 0.112
9
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9716 - loss: 0.0912 - val_accuracy: 0.9699 - val_loss: 0.099
2
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9772 - loss: 0.0741 - val_accuracy: 0.9715 - val_loss: 0.095
3
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9809 - loss: 0.0619 - val_accuracy: 0.9763 - val_loss: 0.075
8
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9852 - loss: 0.0484 - val_accuracy: 0.9762 - val_loss: 0.076
4
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9874 - loss: 0.0433 - val_accuracy: 0.9777 - val_loss: 0.077
7
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9893 - loss: 0.0355 - val_accuracy: 0.9792 - val_loss: 0.077
1
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9907 - loss: 0.0302 - val_accuracy: 0.9787 - val_loss: 0.069
6
```




```
1875/1875 - 3s - 1ms/step - accuracy: 0.9936 - loss: 0.0229
Train accuracy: 99.36%
313/313 - 0s - 1ms/step - accuracy: 0.9787 - loss: 0.0696

2024/04/21 14:57:13 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 97.87%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
Predicted Digit: 3
```

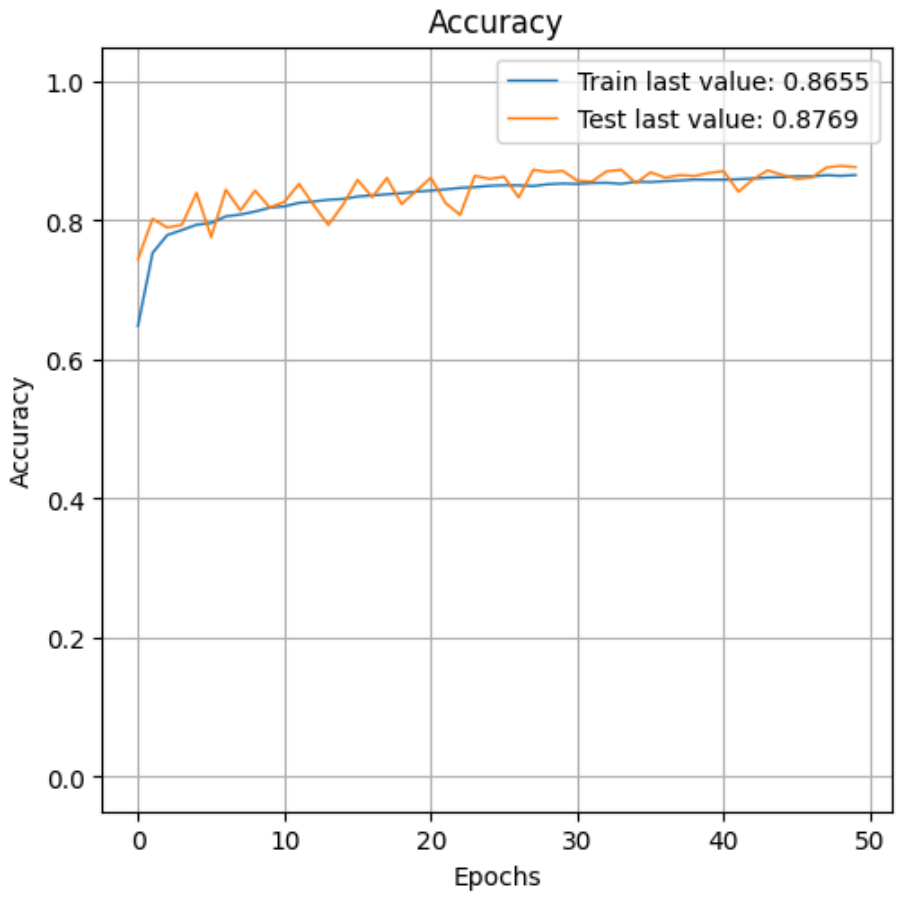

Predicted Image
0
5
10
15
20
25
0
5
10
15
20
25

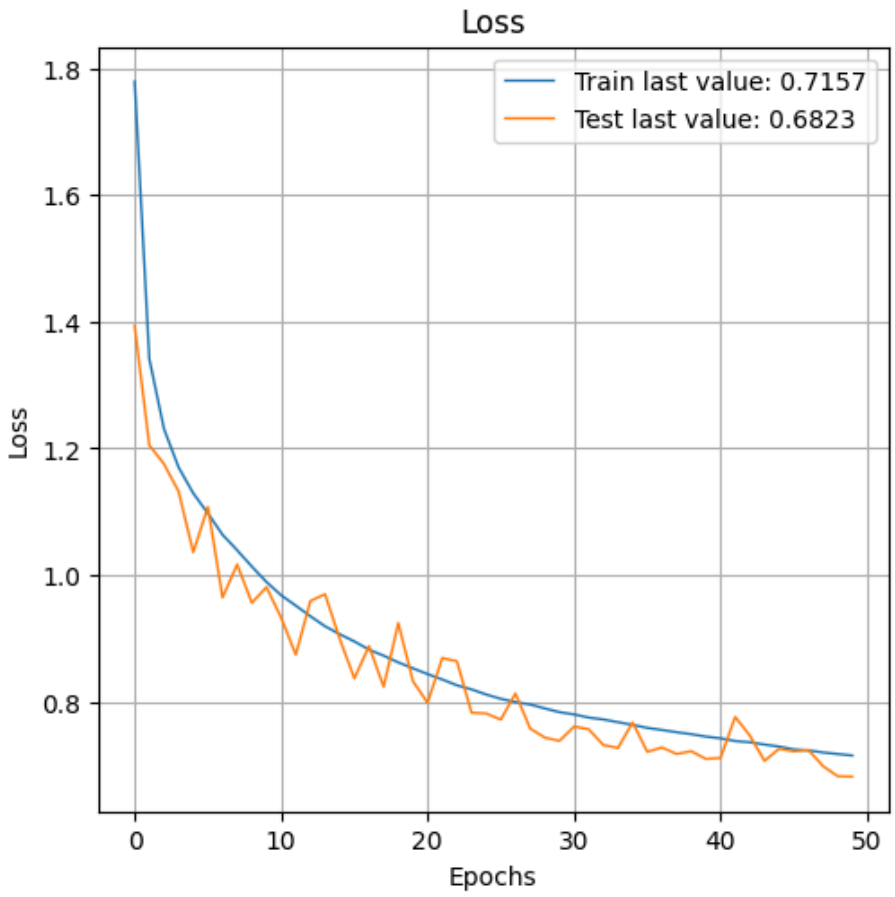
## Adding kernel regularization to the mix.

In [29]:
```
mlp_params = {
    "layer1_size": 256,
    "dropout_rate_l1": 0,
    "layer2_size": 128,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": regularizers.L2(0.01),
}

train_params = dict(
    use_optimiser=False,
    learning_rate=0.1,
    momentum=0.0,
    num_epochs=50,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_regularized",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

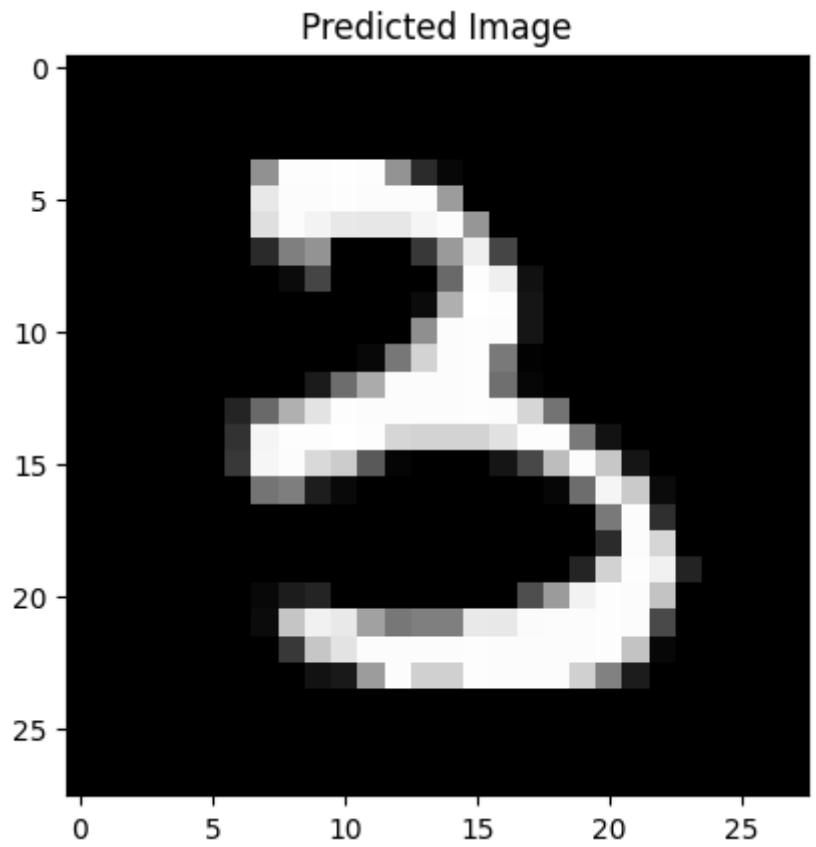
```
Epoch 1/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.5533 - loss: 2.3456 - val_accuracy: 0.7444 - val_loss: 1.393
3
Epoch 2/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7437 - loss: 1.3789 - val_accuracy: 0.8025 - val_loss: 1.205
1
Epoch 3/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7781 - loss: 1.2417 - val_accuracy: 0.7900 - val_loss: 1.175
6
Epoch 4/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7843 - loss: 1.1795 - val_accuracy: 0.7938 - val_loss: 1.132
1
Epoch 5/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.7929 - loss: 1.1383 - val_accuracy: 0.8399 - val_loss: 1.036
2
Epoch 6/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7948 - loss: 1.1060 - val_accuracy: 0.7761 - val_loss: 1.107
9
Epoch 7/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8055 - loss: 1.0704 - val_accuracy: 0.8445 - val_loss: 0.964
9
Epoch 8/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8073 - loss: 1.0464 - val_accuracy: 0.8144 - val_loss: 1.017
2
Epoch 9/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8122 - loss: 1.0182 - val_accuracy: 0.8432 - val_loss: 0.956
4
Epoch 10/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.8162 - loss: 0.9950 - val_accuracy: 0.8188 - val_loss: 0.981
2
Epoch 11/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8189 - loss: 0.9743 - val_accuracy: 0.8269 - val_loss: 0.933
0
Epoch 12/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8271 - loss: 0.9513 - val_accuracy: 0.8529 - val_loss: 0.874
7
Epoch 13/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8259 - loss: 0.9441 - val_accuracy: 0.8222 - val_loss: 0.959
5
Epoch 14/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8277 - loss: 0.9234 - val_accuracy: 0.7938 - val_loss: 0.970
1
Epoch 15/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8329 - loss: 0.9059 - val_accuracy: 0.8227 - val_loss: 0.899
6
Epoch 16/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8322 - loss: 0.9012 - val_accuracy: 0.8587 - val_loss: 0.837
4
Epoch 17/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8362 - loss: 0.8854 - val_accuracy: 0.8336 - val_loss: 0.888
3
Epoch 18/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8362 - loss: 0.8770 - val_accuracy: 0.8614 - val_loss: 0.824
2
Epoch 19/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8389 - loss: 0.8613 - val_accuracy: 0.8237 - val_loss: 0.924
5
Epoch 20/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8395 - loss: 0.8565 - val_accuracy: 0.8426 - val_loss: 0.832
9
Epoch 21/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8413 - loss: 0.8517 - val_accuracy: 0.8616 - val_loss: 0.798
7
Epoch 22/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8437 - loss: 0.8373 - val_accuracy: 0.8254 - val_loss: 0.869
5
Epoch 23/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8453 - loss: 0.8284 - val_accuracy: 0.8081 - val_loss: 0.864
4
Epoch 24/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8478 - loss: 0.8201 - val_accuracy: 0.8645 - val_loss: 0.783
3
Epoch 25/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8494 - loss: 0.8156 - val_accuracy: 0.8599 - val_loss: 0.782
0
Epoch 26/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8491 - loss: 0.8104 - val_accuracy: 0.8631 - val_loss: 0.772
2
Epoch 27/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8506 - loss: 0.8018 - val_accuracy: 0.8333 - val_loss: 0.813
7
Epoch 28/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8500 - loss: 0.7988 - val_accuracy: 0.8731 - val_loss: 0.758
3
Epoch 29/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8548 - loss: 0.7867 - val_accuracy: 0.8698 - val_loss: 0.743
9
Epoch 30/50
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8533 - loss: 0.7851 - val_accuracy: 0.8716 - val_loss: 0.738
7
Epoch 31/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8532 - loss: 0.7798 - val_accuracy: 0.8574 - val_loss: 0.761
3
Epoch 32/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8514 - loss: 0.7816 - val_accuracy: 0.8562 - val_loss: 0.757
2
Epoch 33/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8550 - loss: 0.7711 - val_accuracy: 0.8709 - val_loss: 0.732
1
Epoch 34/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8536 - loss: 0.7701 - val_accuracy: 0.8731 - val_loss: 0.727
7
Epoch 35/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8552 - loss: 0.7645 - val_accuracy: 0.8538 - val_loss: 0.767
9
Epoch 36/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8552 - loss: 0.7594 - val_accuracy: 0.8695 - val_loss: 0.721
4
Epoch 37/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8568 - loss: 0.7554 - val_accuracy: 0.8618 - val_loss: 0.728
4
Epoch 38/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8559 - loss: 0.7521 - val_accuracy: 0.8654 - val_loss: 0.718
0
Epoch 39/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8604 - loss: 0.7486 - val_accuracy: 0.8641 - val_loss: 0.722
4
Epoch 40/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8587 - loss: 0.7511 - val_accuracy: 0.8686 - val_loss: 0.710
2
Epoch 41/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8599 - loss: 0.7423 - val_accuracy: 0.8711 - val_loss: 0.711
8
Epoch 42/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8601 - loss: 0.7370 - val_accuracy: 0.8415 - val_loss: 0.776
6
Epoch 43/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8582 - loss: 0.7416 - val_accuracy: 0.8594 - val_loss: 0.747
2
Epoch 44/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8620 - loss: 0.7354 - val_accuracy: 0.8723 - val_loss: 0.707
1
Epoch 45/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8623 - loss: 0.7302 - val_accuracy: 0.8656 - val_loss: 0.726
3
Epoch 46/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8636 - loss: 0.7286 - val_accuracy: 0.8600 - val_loss: 0.722
2
Epoch 47/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8635 - loss: 0.7275 - val_accuracy: 0.8618 - val_loss: 0.724
5
Epoch 48/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8663 - loss: 0.7180 - val_accuracy: 0.8764 - val_loss: 0.698
9
Epoch 49/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8632 - loss: 0.7223 - val_accuracy: 0.8787 - val_loss: 0.682
9
Epoch 50/50
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8645 - loss: 0.7205 - val_accuracy: 0.8769 - val_loss: 0.682
3
```



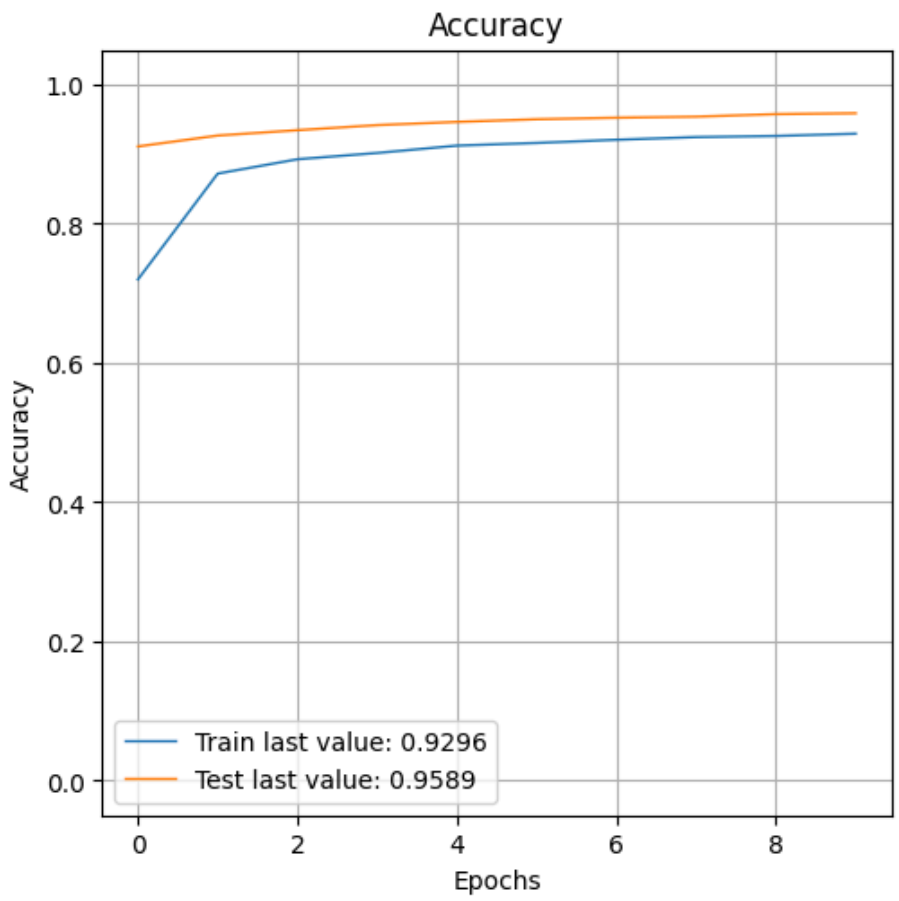

```
1875/1875 - 2s - 1ms/step - accuracy: 0.8727 - loss: 0.6946
Train accuracy: 87.27%
313/313 - 0s - 1ms/step - accuracy: 0.8769 - loss: 0.6823

2024/04/21 15:03:37 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 87.69%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
Predicted Digit: 3
```

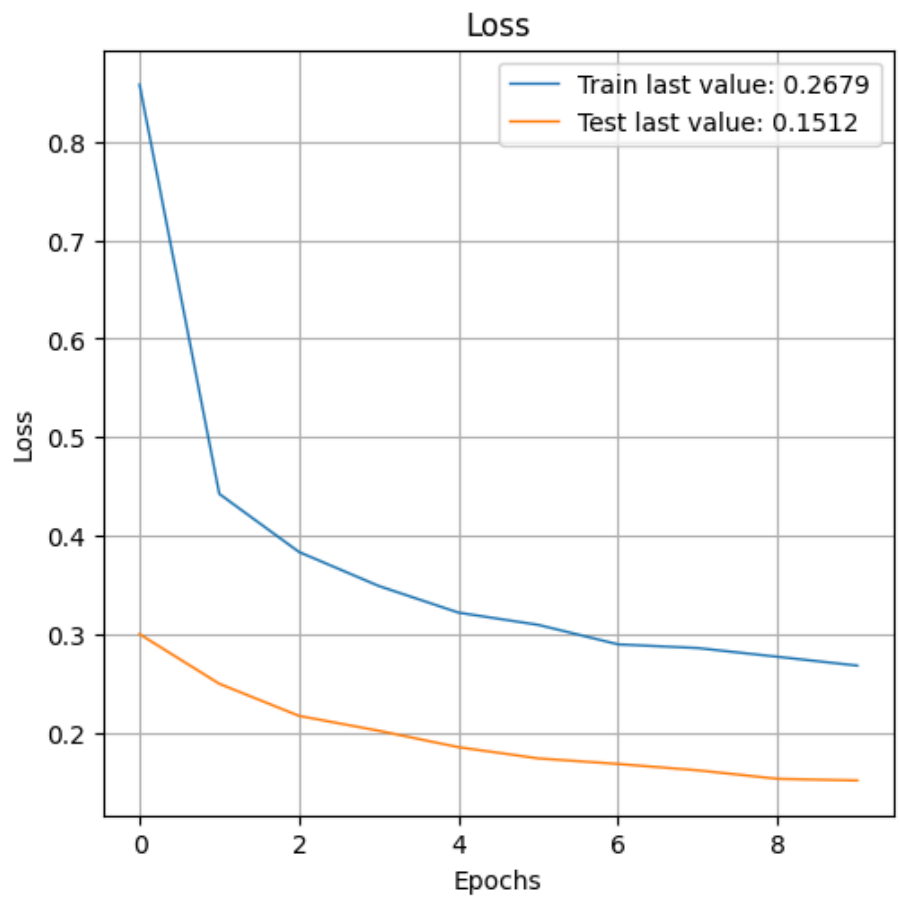


## Adding Dropout

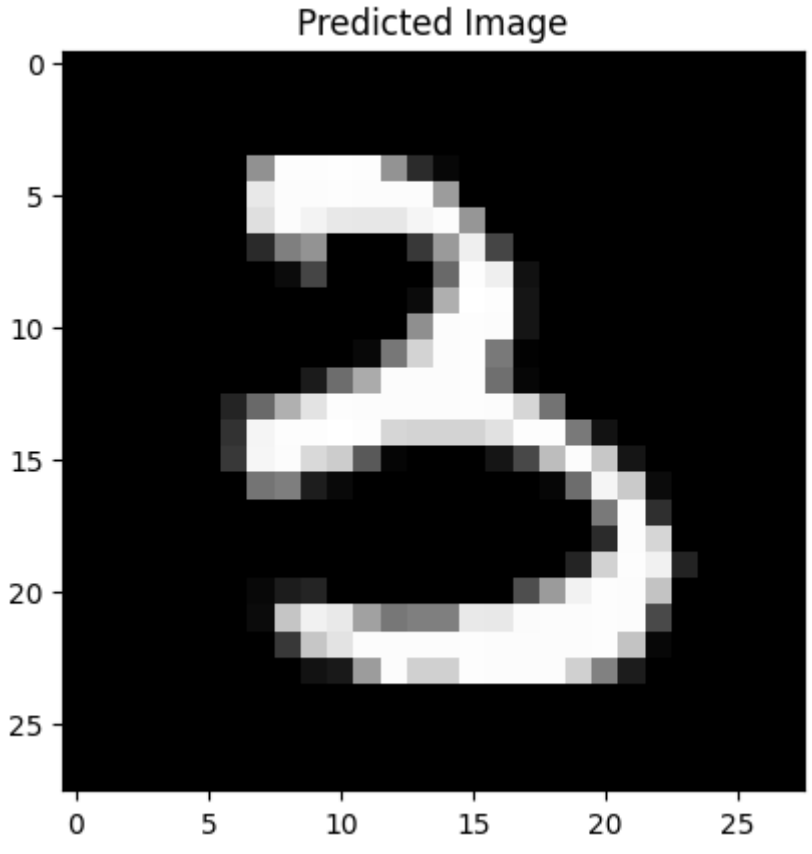
In [30]:
```
mlp_params = {
    "layer1_size": 256,
    "dropout_rate_l1": 0.7,
    "layer2_size": 128,
    "dropout_rate_l2": 0.6,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=False,
    learning_rate=0.1,
    momentum=0.0,
    num_epochs=10,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_with_dropout",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.5516 - loss: 1.3252 - val_accuracy: 0.9112 - val_loss: 0.299
9
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8660 - loss: 0.4626 - val_accuracy: 0.9269 - val_loss: 0.249
5
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8877 - loss: 0.3961 - val_accuracy: 0.9346 - val_loss: 0.216
8
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8996 - loss: 0.3554 - val_accuracy: 0.9419 - val_loss: 0.201
7
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9116 - loss: 0.3228 - val_accuracy: 0.9465 - val_loss: 0.185
0
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9142 - loss: 0.3157 - val_accuracy: 0.9503 - val_loss: 0.173
6
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9205 - loss: 0.2860 - val_accuracy: 0.9525 - val_loss: 0.168
0
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9262 - loss: 0.2839 - val_accuracy: 0.9540 - val_loss: 0.161
5
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9265 - loss: 0.2747 - val_accuracy: 0.9577 - val_loss: 0.152
9
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9275 - loss: 0.2709 - val_accuracy: 0.9589 - val_loss: 0.151
2
```



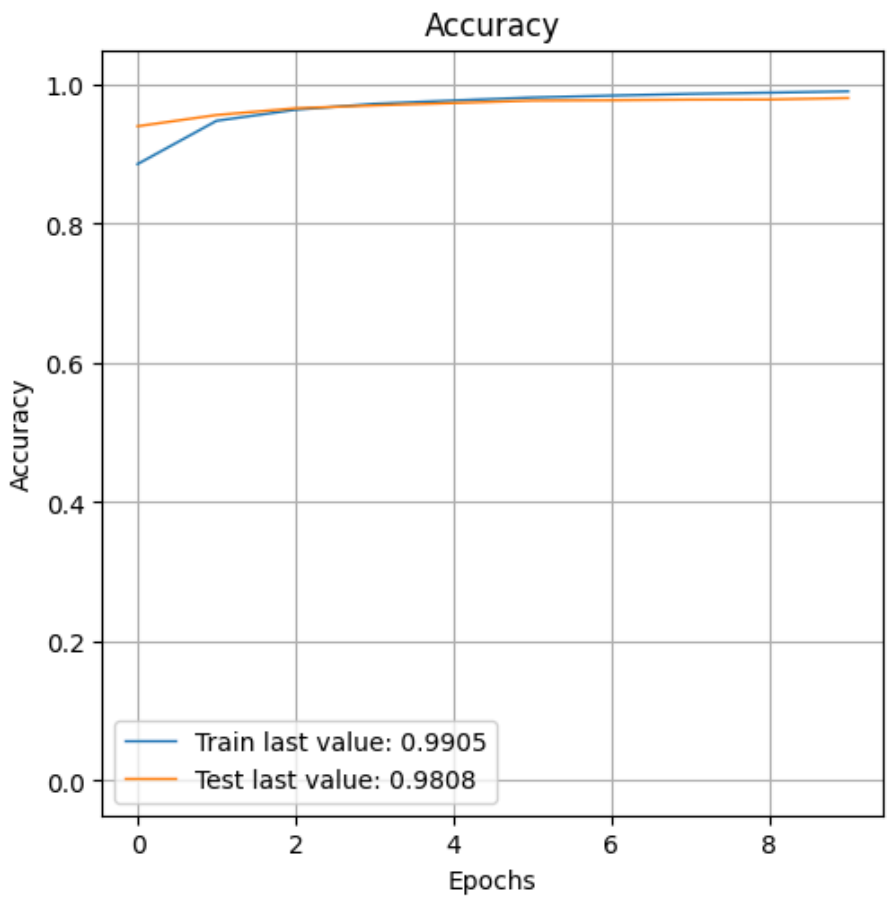

```
1875/1875 - 2s - 1ms/step - accuracy: 0.9610 - loss: 0.1421
Train accuracy: 96.10%
313/313 - 0s - 1ms/step - accuracy: 0.9589 - loss: 0.1512

2024/04/21 15:04:49 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 95.89%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
Predicted Digit: 3
```

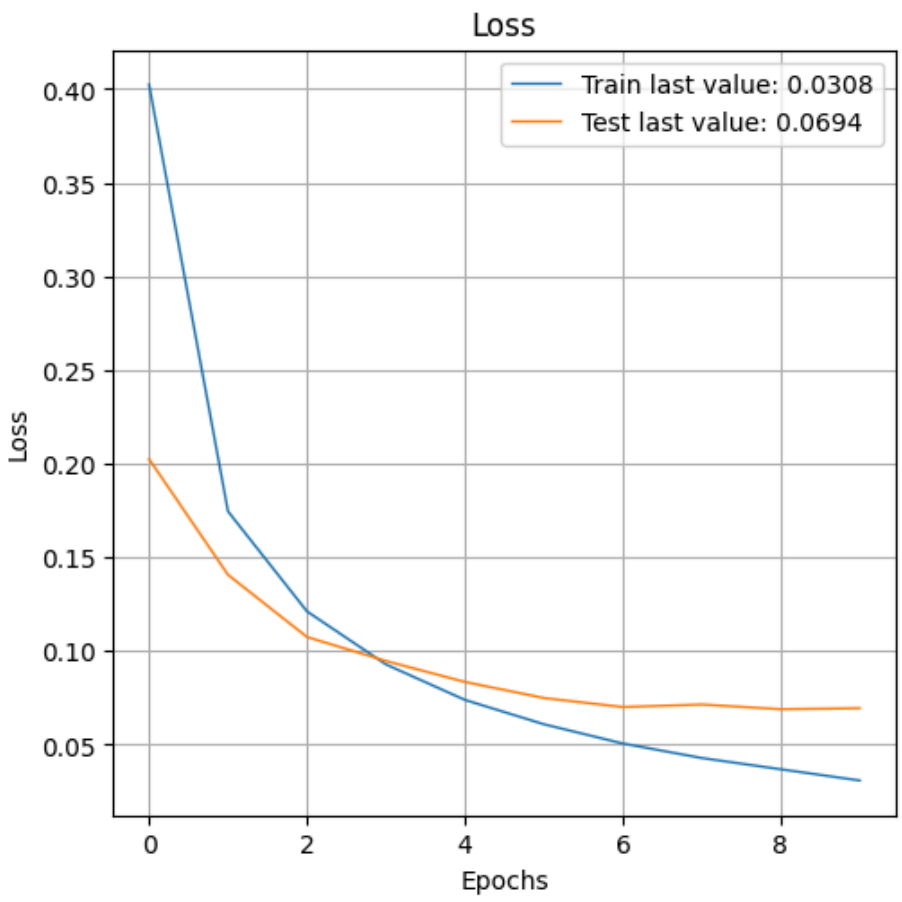

Predicted Image
0
5
10
15
20
25
0
5
10
15
20
25

## Adding Early Stopping Method

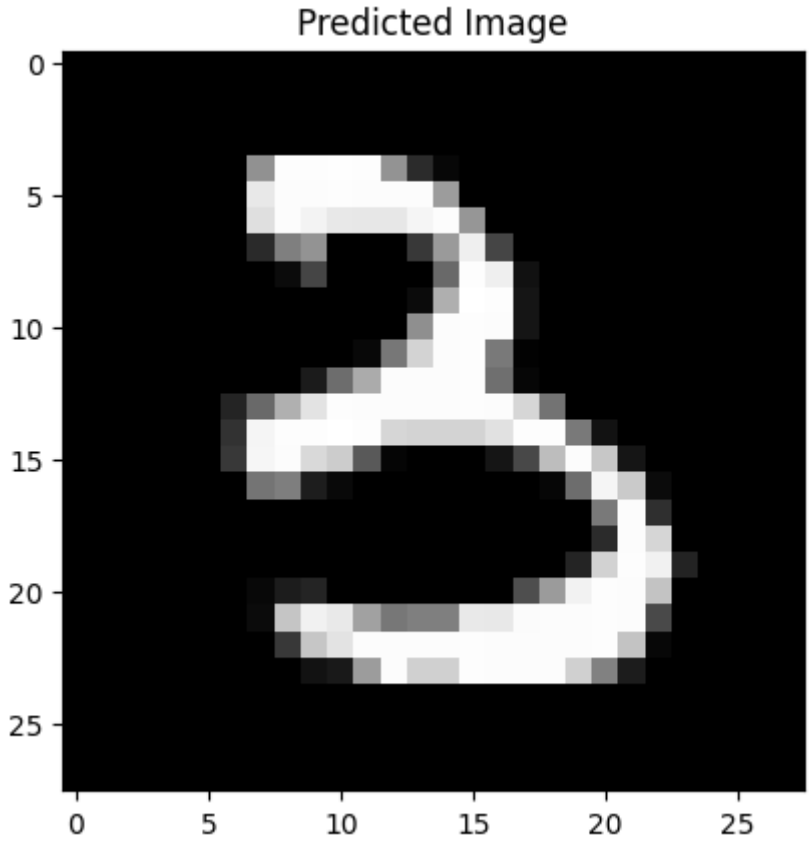
In [31]:
```
mlp_params = {
    "layer1_size": 256,
    "dropout_rate_l1": 0,
    "layer2_size": 128,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=False,
    learning_rate=0.1,
    momentum=0.0,
    num_epochs=10,
    steps_per_epoch=50,
    early_stopping=True,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_with_earlt_stopping",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - accuracy: 0.7980 - loss: 0.7143 - val_accuracy: 0.9403 - val_loss: 0.20
24
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9435 - loss: 0.1911 - val_accuracy: 0.9564 - val_loss: 0.140
6
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9631 - loss: 0.1269 - val_accuracy: 0.9661 - val_loss: 0.107
4
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9731 - loss: 0.0908 - val_accuracy: 0.9702 - val_loss: 0.094
6
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9779 - loss: 0.0719 - val_accuracy: 0.9736 - val_loss: 0.083
4
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9825 - loss: 0.0581 - val_accuracy: 0.9771 - val_loss: 0.074
9
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9841 - loss: 0.0498 - val_accuracy: 0.9776 - val_loss: 0.070
0
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9869 - loss: 0.0413 - val_accuracy: 0.9785 - val_loss: 0.071
4
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9893 - loss: 0.0339 - val_accuracy: 0.9787 - val_loss: 0.068
8
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9908 - loss: 0.0292 - val_accuracy: 0.9808 - val_loss: 0.069
4
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9927 - loss: 0.0233 - val_accuracy: 0.9782 - val_loss: 0.074
5
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9935 - loss: 0.0209 - val_accuracy: 0.9813 - val_loss: 0.072
2
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9948 - loss: 0.0185 - val_accuracy: 0.9786 - val_loss: 0.078
5
```




```
1875/1875 - 2s - 1ms/step - accuracy: 0.9954 - loss: 0.0156
Train accuracy: 99.54%
313/313 - 0s - 1ms/step - accuracy: 0.9786 - loss: 0.0785

2024/04/21 15:06:24 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 97.86%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step
Predicted Digit: 3
```

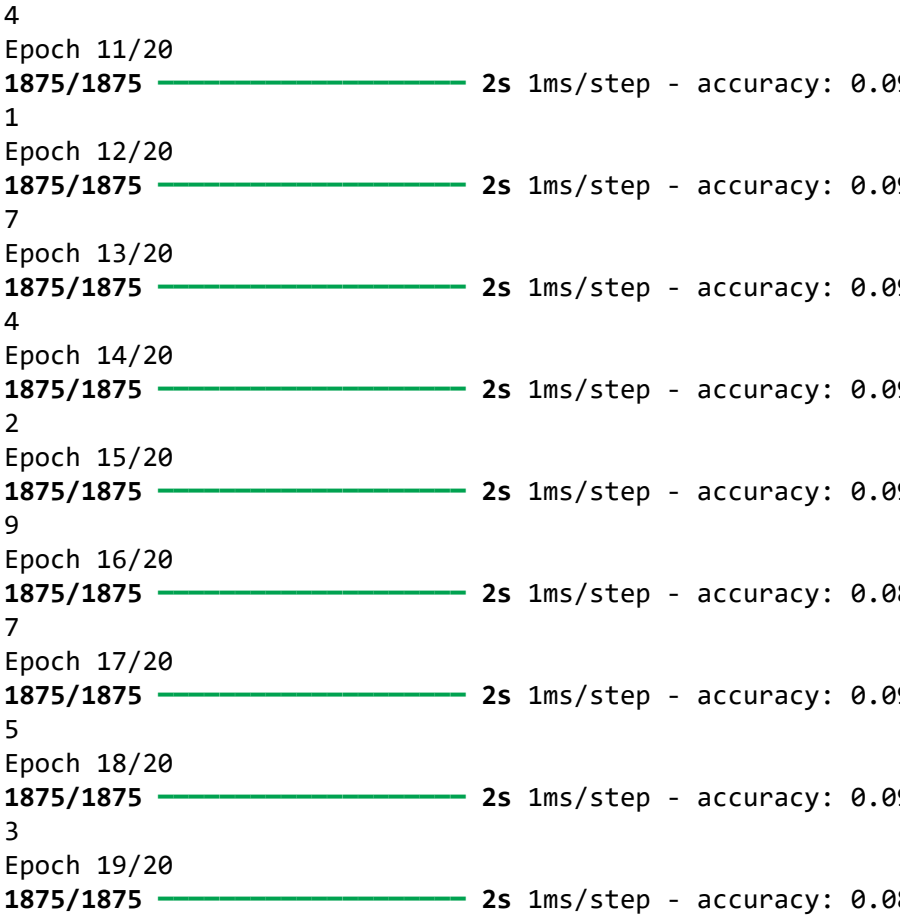


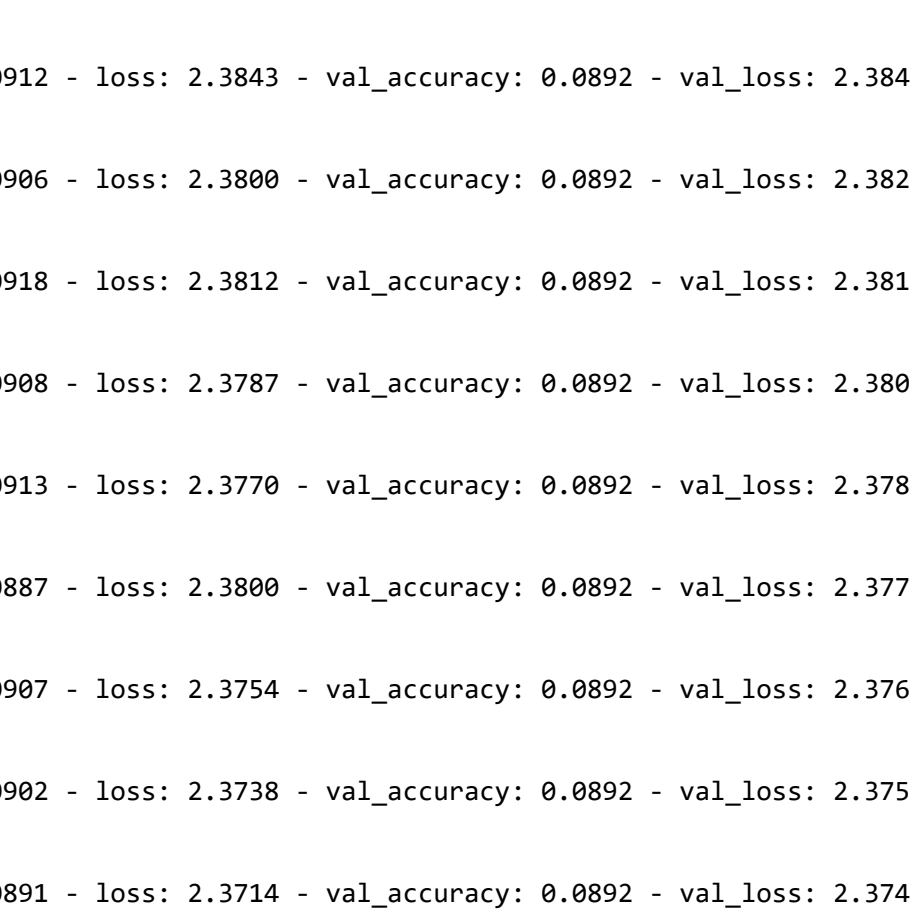

## Using Learning rates

### Low Learning Rate

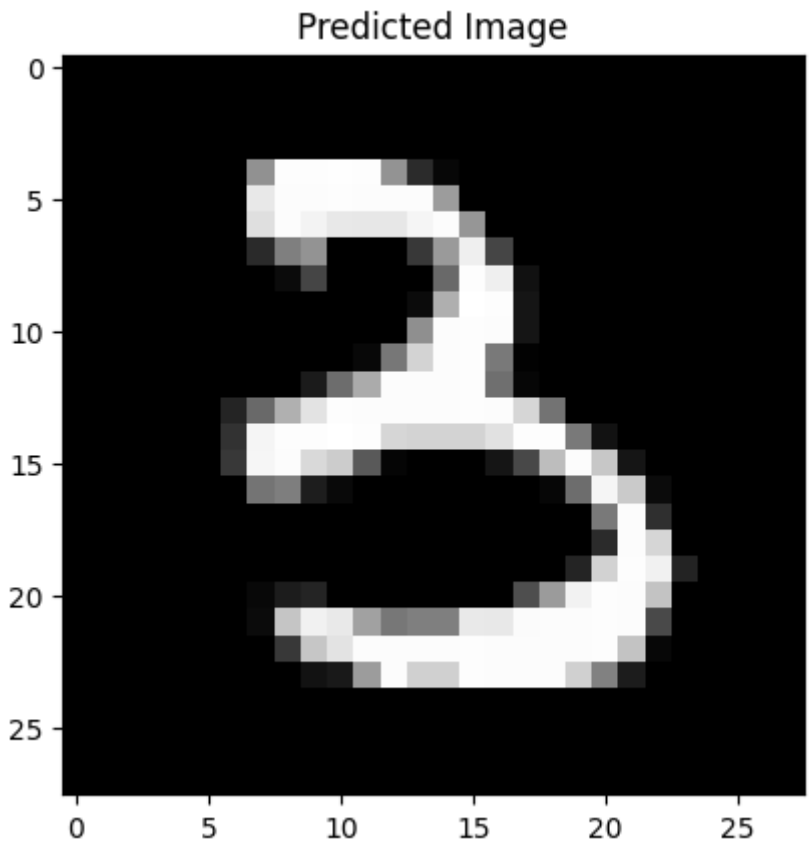
In [32]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 10,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=True,
    learning_rate=0.00001,
    momentum=0.0,
    num_epochs=20,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)


mlp_mlflow_run(
    "mlp_with_low_lwarning_rate",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.0889 - loss: 2.3968 - val_accuracy: 0.0892 - val_loss: 2.398
4
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0905 - loss: 2.3970 - val_accuracy: 0.0892 - val_loss: 2.396
9
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0902 - loss: 2.3933 - val_accuracy: 0.0892 - val_loss: 2.395
4
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0895 - loss: 2.3932 - val_accuracy: 0.0892 - val_loss: 2.393
9
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0899 - loss: 2.3925 - val_accuracy: 0.0892 - val_loss: 2.392
4
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0919 - loss: 2.3892 - val_accuracy: 0.0892 - val_loss: 2.391
0
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0923 - loss: 2.3875 - val_accuracy: 0.0892 - val_loss: 2.389
5
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0897 - loss: 2.3897 - val_accuracy: 0.0892 - val_loss: 2.388
1
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0890 - loss: 2.3861 - val_accuracy: 0.0892 - val_loss: 2.386
8
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0902 - loss: 2.3858 - val_accuracy: 0.0892 - val_loss: 2.385
4
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0912 - loss: 2.3843 - val_accuracy: 0.0892 - val_loss: 2.384
1
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0906 - loss: 2.3800 - val_accuracy: 0.0892 - val_loss: 2.382
7
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0918 - loss: 2.3812 - val_accuracy: 0.0892 - val_loss: 2.381
4
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0908 - loss: 2.3787 - val_accuracy: 0.0892 - val_loss: 2.380
2
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0913 - loss: 2.3770 - val_accuracy: 0.0892 - val_loss: 2.378
9
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0887 - loss: 2.3800 - val_accuracy: 0.0892 - val_loss: 2.377
7
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0907 - loss: 2.3754 - val_accuracy: 0.0892 - val_loss: 2.376
5
Epoch 18/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0902 - loss: 2.3738 - val_accuracy: 0.0892 - val_loss: 2.375
3
Epoch 19/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0891 - loss: 2.3714 - val_accuracy: 0.0892 - val_loss: 2.374
1
Epoch 20/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.0905 - loss: 2.3725 - val_accuracy: 0.0892 - val_loss: 2.372
9
```



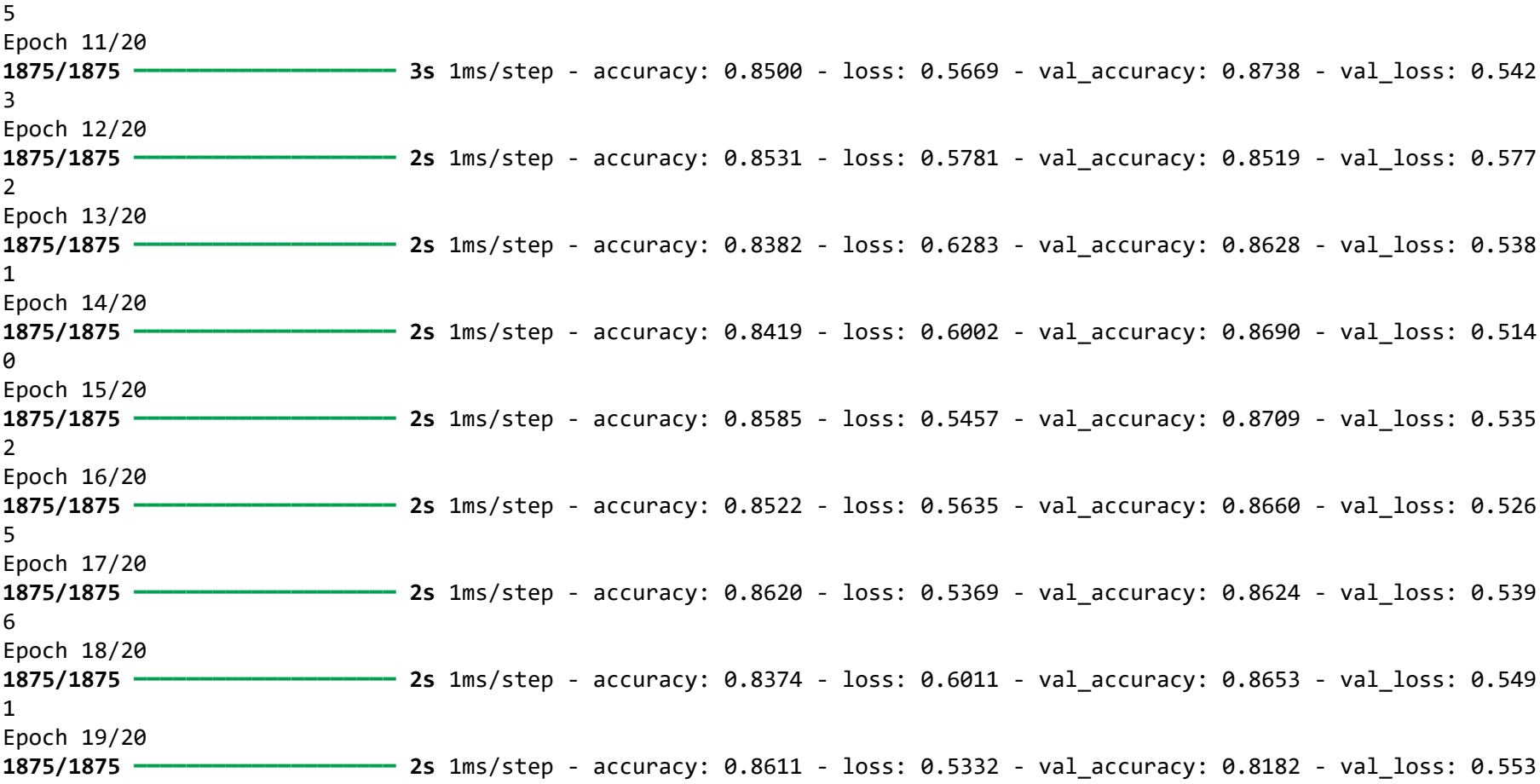
```
1875/1875 - 2s - 886us/step - accuracy: 0.0904 - loss: 2.3713
Train accuracy:  9.04%
313/313 - 0s - 1ms/step - accuracy: 0.0892 - loss: 2.3729

2024/04/21 15:07:19 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy:  8.92%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
Predicted Digit: 5
```



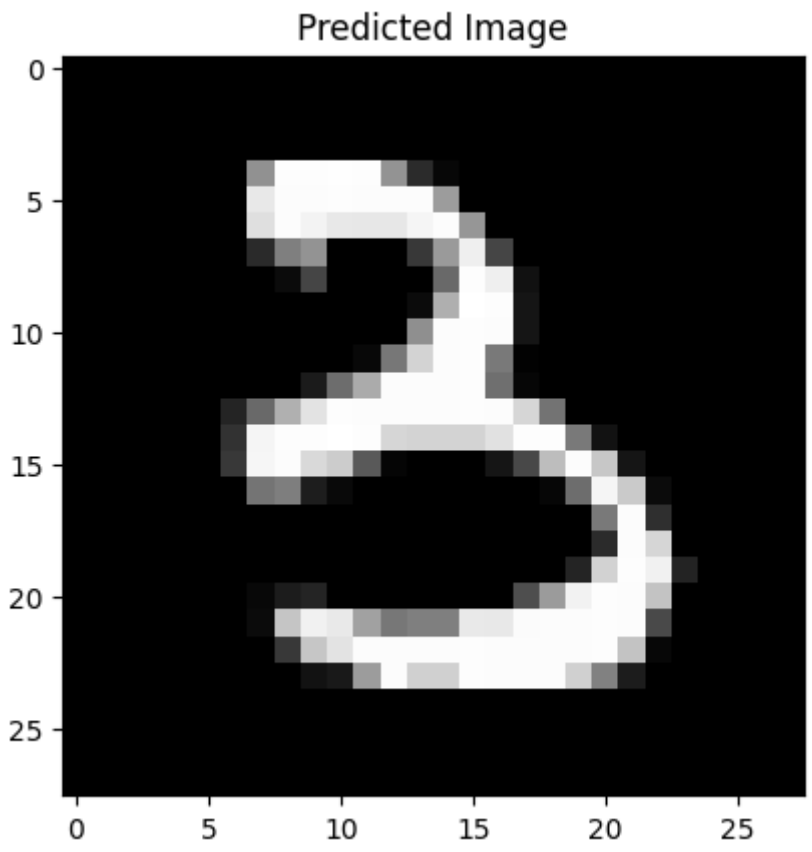
## High Learning Rate

In [33]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 10,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=True,
    learning_rate=10.0,
    momentum=0.0,
    num_epochs=20,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_with_high_learning_rate",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.1030 - loss: 2.4798 - val_accuracy: 0.2723 - val_loss: 2.287
5
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.2952 - loss: 1.9968 - val_accuracy: 0.5050 - val_loss: 1.696
4
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.5998 - loss: 1.1876 - val_accuracy: 0.7949 - val_loss: 0.817
9
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.7350 - loss: 0.9317 - val_accuracy: 0.7825 - val_loss: 0.848
1
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.7895 - loss: 0.7892 - val_accuracy: 0.8490 - val_loss: 0.649
0
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8081 - loss: 0.7310 - val_accuracy: 0.8371 - val_loss: 0.674
2
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8049 - loss: 0.7258 - val_accuracy: 0.7760 - val_loss: 0.835
8
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7965 - loss: 0.7627 - val_accuracy: 0.8257 - val_loss: 0.634
9
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8444 - loss: 0.6142 - val_accuracy: 0.8494 - val_loss: 0.670
6
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8467 - loss: 0.5785 - val_accuracy: 0.8554 - val_loss: 0.552
5
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8500 - loss: 0.5669 - val_accuracy: 0.8738 - val_loss: 0.542
3
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8531 - loss: 0.5781 - val_accuracy: 0.8519 - val_loss: 0.577
2
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8382 - loss: 0.6283 - val_accuracy: 0.8628 - val_loss: 0.538
1
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8419 - loss: 0.6002 - val_accuracy: 0.8690 - val_loss: 0.514
0
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8585 - loss: 0.5457 - val_accuracy: 0.8709 - val_loss: 0.535
2
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8522 - loss: 0.5635 - val_accuracy: 0.8660 - val_loss: 0.526
5
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8620 - loss: 0.5369 - val_accuracy: 0.8624 - val_loss: 0.539
6
Epoch 18/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8374 - loss: 0.6011 - val_accuracy: 0.8653 - val_loss: 0.549
1
Epoch 19/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8611 - loss: 0.5332 - val_accuracy: 0.8182 - val_loss: 0.553
7
Epoch 20/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8559 - loss: 0.5457 - val_accuracy: 0.8662 - val_loss: 0.543
9
```



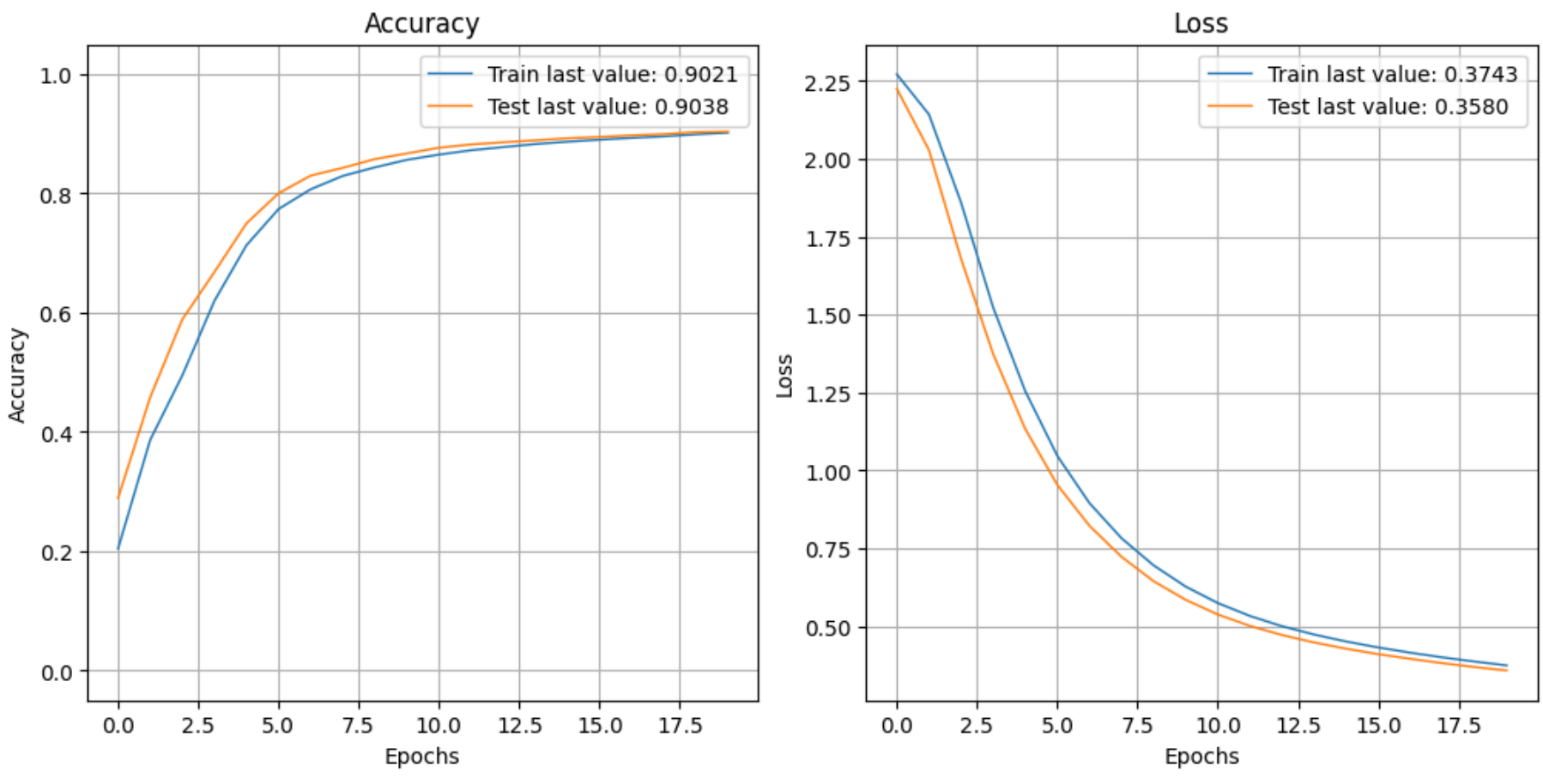
```
1875/1875 - 2s - 917us/step - accuracy: 0.8688 - loss: 0.5247
Train accuracy: 86.88%
313/313 - 0s - 990us/step - accuracy: 0.8662 - loss: 0.5439

2024/04/21 15:08:19 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 86.62%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
Predicted Digit: 3
```



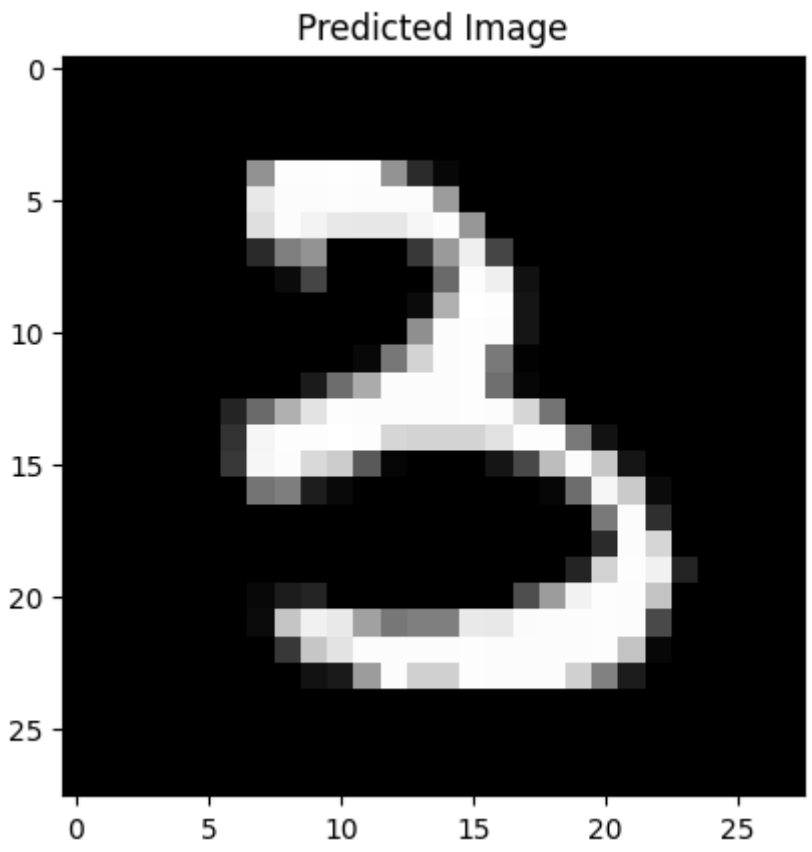
## Optimal Learning Rate

In [34]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 10,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=True,
    learning_rate=0.01,
    momentum=0.0,
    num_epochs=20,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_with_optimal_learning_rate",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.1600 - loss: 2.2960 - val_accuracy: 0.2891 - val_loss: 2.224
7
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.3469 - loss: 2.1886 - val_accuracy: 0.4581 - val_loss: 2.027
9
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.4718 - loss: 1.9457 - val_accuracy: 0.5887 - val_loss: 1.679
1
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.5945 - loss: 1.6035 - val_accuracy: 0.6682 - val_loss: 1.373
8
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.6915 - loss: 1.3165 - val_accuracy: 0.7499 - val_loss: 1.133
1
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7632 - loss: 1.0893 - val_accuracy: 0.8004 - val_loss: 0.953
4
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7981 - loss: 0.9329 - val_accuracy: 0.8300 - val_loss: 0.822
3
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8226 - loss: 0.8098 - val_accuracy: 0.8430 - val_loss: 0.722
6
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8381 - loss: 0.7164 - val_accuracy: 0.8577 - val_loss: 0.644
4
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8509 - loss: 0.6462 - val_accuracy: 0.8672 - val_loss: 0.584
6
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8650 - loss: 0.5829 - val_accuracy: 0.8768 - val_loss: 0.537
5
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8694 - loss: 0.5470 - val_accuracy: 0.8823 - val_loss: 0.500
9
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8763 - loss: 0.5089 - val_accuracy: 0.8859 - val_loss: 0.471
6
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8794 - loss: 0.4853 - val_accuracy: 0.8892 - val_loss: 0.447
8
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8871 - loss: 0.4530 - val_accuracy: 0.8926 - val_loss: 0.427
6
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8887 - loss: 0.4350 - val_accuracy: 0.8949 - val_loss: 0.410
1
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8927 - loss: 0.4199 - val_accuracy: 0.8976 - val_loss: 0.395
3
Epoch 18/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8943 - loss: 0.4040 - val_accuracy: 0.8996 - val_loss: 0.381
3
Epoch 19/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8987 - loss: 0.3895 - val_accuracy: 0.9029 - val_loss: 0.368
9
Epoch 20/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8993 - loss: 0.3838 - val_accuracy: 0.9038 - val_loss: 0.358
0
```



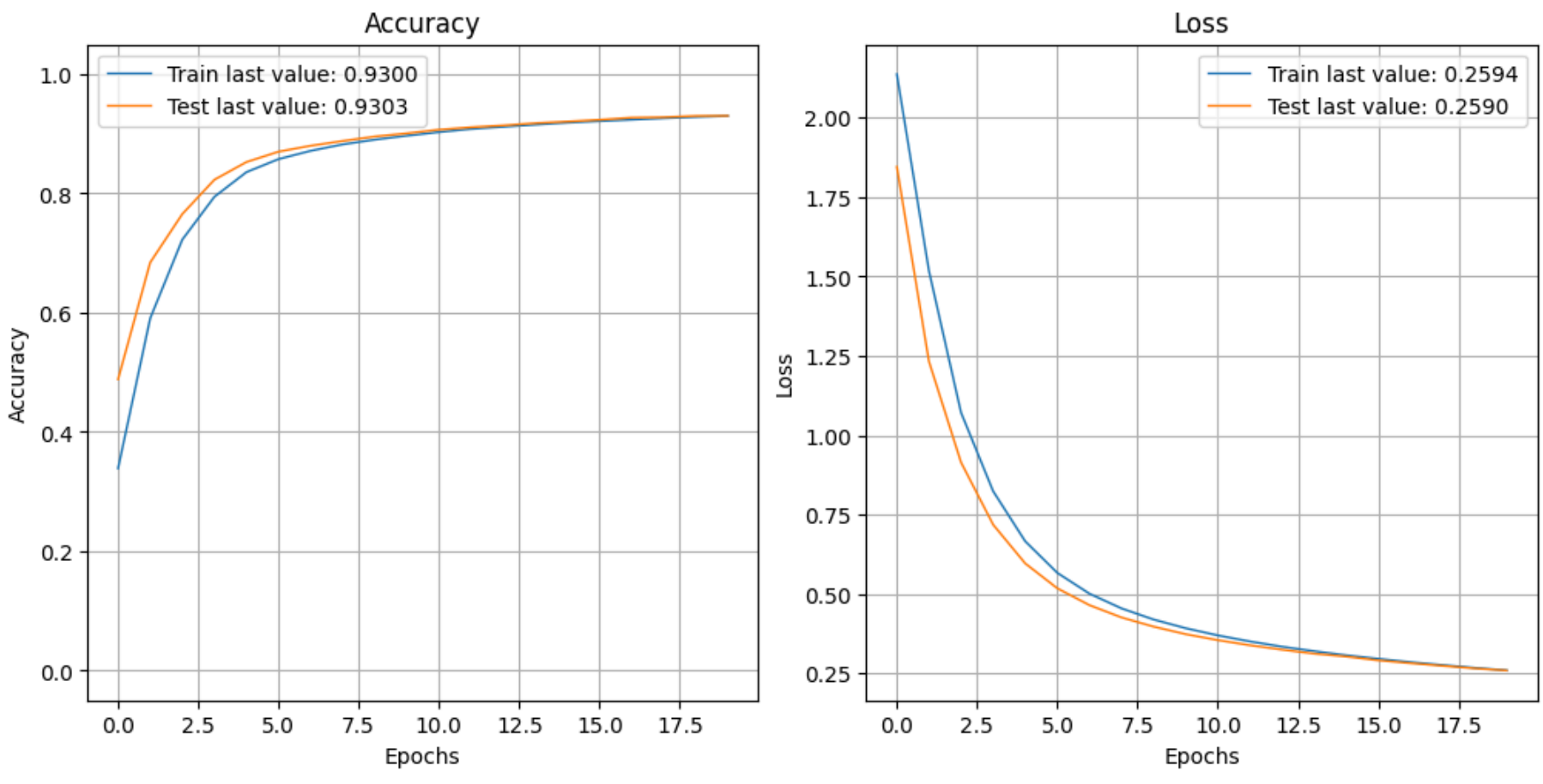
```
1875/1875 - 2s - 890us/step - accuracy: 0.9032 - loss: 0.3678
Train accuracy: 90.32%
313/313 - 0s - 965us/step - accuracy: 0.9038 - loss: 0.3580

2024/04/21 15:09:15 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 90.38%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
Predicted Digit: 3
```



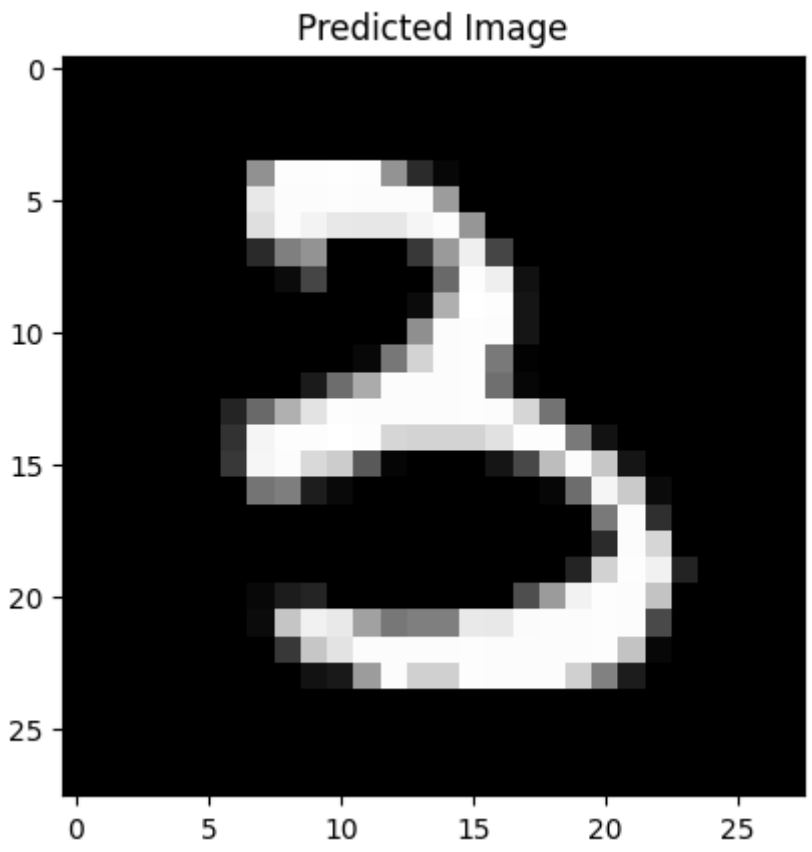
## optimal learning rate with momentum

In [35]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 10,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=True,
    learning_rate=0.01,
    momentum=0.5,
    num_epochs=20,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=32,
)

mlp_mlflow_run(
    "mlp_with_optimal_learning_rate_and_momentum",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.2214 - loss: 2.2460 - val_accuracy: 0.4884 - val_loss: 1.845
4
Epoch 2/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.5376 - loss: 1.6781 - val_accuracy: 0.6842 - val_loss: 1.232
5
Epoch 3/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7002 - loss: 1.1489 - val_accuracy: 0.7655 - val_loss: 0.914
4
Epoch 4/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.7785 - loss: 0.8689 - val_accuracy: 0.8231 - val_loss: 0.718
3
Epoch 5/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.8299 - loss: 0.6955 - val_accuracy: 0.8528 - val_loss: 0.596
1
Epoch 6/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8533 - loss: 0.5880 - val_accuracy: 0.8702 - val_loss: 0.517
7
Epoch 7/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8690 - loss: 0.5115 - val_accuracy: 0.8801 - val_loss: 0.464
7
Epoch 8/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8773 - loss: 0.4673 - val_accuracy: 0.8881 - val_loss: 0.425
9
Epoch 9/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8891 - loss: 0.4276 - val_accuracy: 0.8956 - val_loss: 0.397
0
Epoch 10/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8953 - loss: 0.3972 - val_accuracy: 0.9009 - val_loss: 0.373
2
Epoch 11/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8999 - loss: 0.3790 - val_accuracy: 0.9071 - val_loss: 0.354
4
Epoch 12/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9067 - loss: 0.3538 - val_accuracy: 0.9111 - val_loss: 0.338
3
Epoch 13/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9113 - loss: 0.3367 - val_accuracy: 0.9141 - val_loss: 0.324
4
Epoch 14/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9148 - loss: 0.3214 - val_accuracy: 0.9180 - val_loss: 0.311
9
Epoch 15/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - accuracy: 0.9183 - loss: 0.3091 - val_accuracy: 0.9207 - val_loss: 0.302
7
Epoch 16/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9201 - loss: 0.2998 - val_accuracy: 0.9236 - val_loss: 0.290
8
Epoch 17/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9229 - loss: 0.2899 - val_accuracy: 0.9272 - val_loss: 0.281
8
Epoch 18/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9253 - loss: 0.2785 - val_accuracy: 0.9280 - val_loss: 0.273
6
Epoch 19/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9272 - loss: 0.2706 - val_accuracy: 0.9301 - val_loss: 0.265
5
Epoch 20/20
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.9301 - loss: 0.2584 - val_accuracy: 0.9303 - val_loss: 0.259
0
```



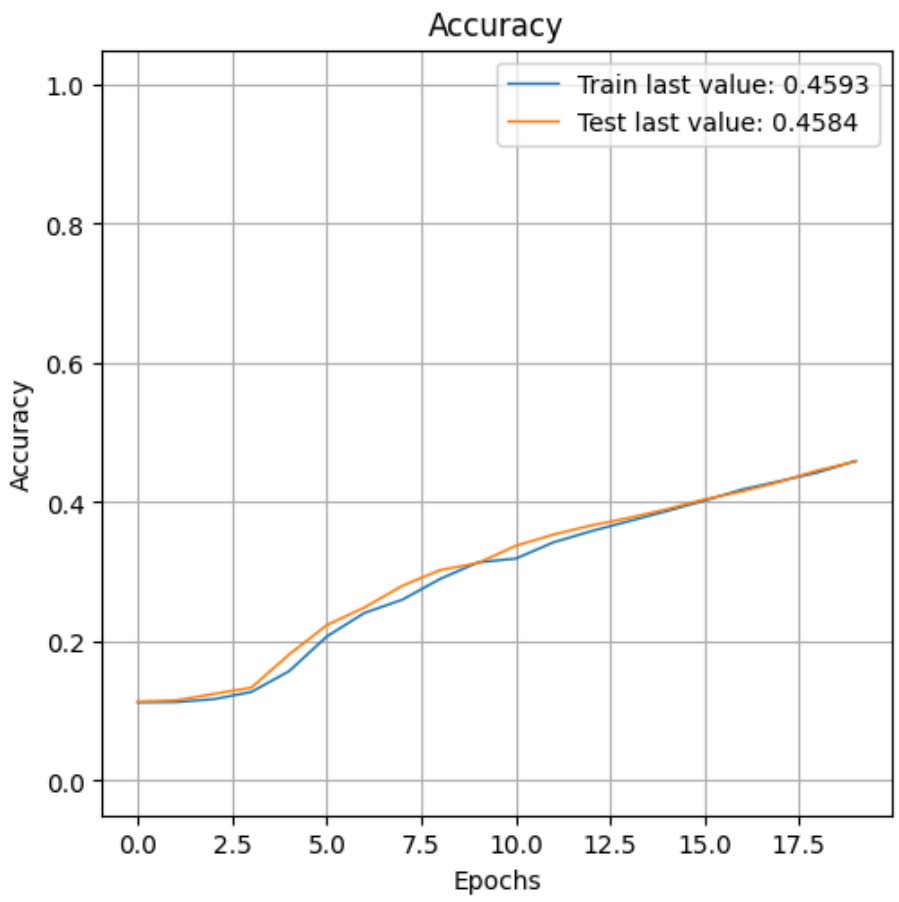
```
1875/1875 - 2s - 909us/step - accuracy: 0.9316 - loss: 0.2539
Train accuracy: 93.16%
313/313 - 0s - 853us/step - accuracy: 0.9303 - loss: 0.2590

2024/04/21 15:10:16 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 93.03%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
Predicted Digit: 3
```

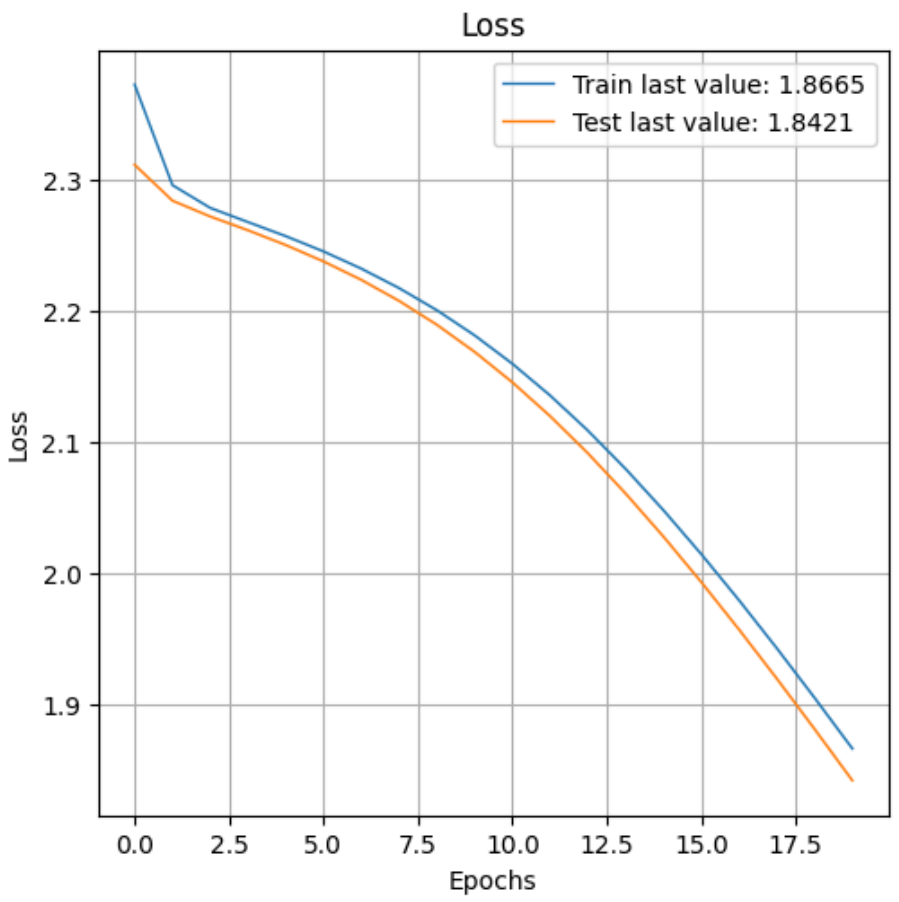


## Using Mini-batch SGD

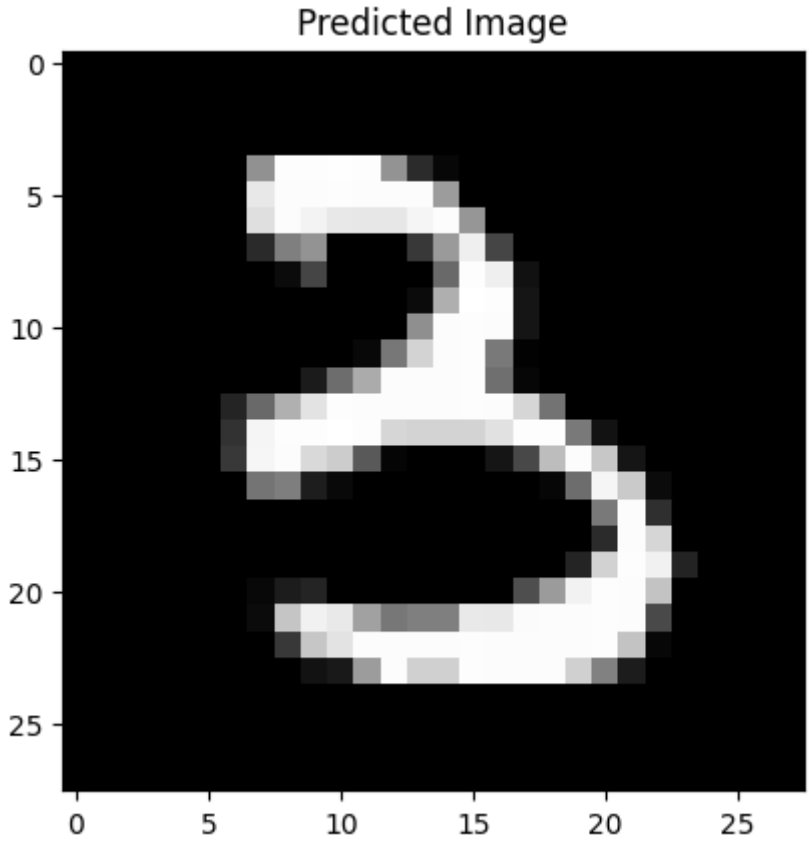
In [36]:
```
mlp_params = {
    "layer1_size": 20,
    "dropout_rate_l1": 0,
    "layer2_size": 10,
    "dropout_rate_l2": 0,
    "output_size": 10,
    "activation": 'sigmoid',
    "regularizers": None,
}

train_params = dict(
    use_optimiser=True,
    learning_rate=0.01,
    momentum=0.5,
    num_epochs=20,
    steps_per_epoch=50,
    early_stopping=False,
    batch_size=512,
)

mlp_mlflow_run(
    "mlp_with_mini_batch_and_momentum",
    mlp_params,
    train_params,
    x_train,
    y_train,
    x_test,
    y_test,
)
```

```
Epoch 1/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.1116 - loss: 2.4220 - val_accuracy: 0.1135 - val_loss: 2.3118
Epoch 2/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1140 - loss: 2.3029 - val_accuracy: 0.1151 - val_loss: 2.2844
Epoch 3/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1156 - loss: 2.2814 - val_accuracy: 0.1244 - val_loss: 2.2723
Epoch 4/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1232 - loss: 2.2707 - val_accuracy: 0.1334 - val_loss: 2.2618
Epoch 5/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1436 - loss: 2.2598 - val_accuracy: 0.1810 - val_loss: 2.2505
Epoch 6/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.1956 - loss: 2.2486 - val_accuracy: 0.2233 - val_loss: 2.2380
Epoch 7/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2321 - loss: 2.2359 - val_accuracy: 0.2488 - val_loss: 2.2239
Epoch 8/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2551 - loss: 2.2216 - val_accuracy: 0.2799 - val_loss: 2.2079
Epoch 9/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.2854 - loss: 2.2053 - val_accuracy: 0.3024 - val_loss: 2.1897
Epoch 10/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3079 - loss: 2.1869 - val_accuracy: 0.3126 - val_loss: 2.1690
Epoch 11/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3151 - loss: 2.1653 - val_accuracy: 0.3373 - val_loss: 2.1458
Epoch 12/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3363 - loss: 2.1415 - val_accuracy: 0.3535 - val_loss: 2.1200
Epoch 13/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3514 - loss: 2.1158 - val_accuracy: 0.3665 - val_loss: 2.0917
Epoch 14/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3696 - loss: 2.0854 - val_accuracy: 0.3775 - val_loss: 2.0609
Epoch 15/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3813 - loss: 2.0564 - val_accuracy: 0.3898 - val_loss: 2.0281
Epoch 16/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3965 - loss: 2.0233 - val_accuracy: 0.4040 - val_loss: 1.9935
Epoch 17/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4143 - loss: 1.9879 - val_accuracy: 0.4155 - val_loss: 1.9573
Epoch 18/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4239 - loss: 1.9539 - val_accuracy: 0.4295 - val_loss: 1.9198
Epoch 19/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4381 - loss: 1.9139 - val_accuracy: 0.4457 - val_loss: 1.8814
Epoch 20/20
118/118 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4560 - loss: 1.8762 - val_accuracy: 0.4584 - val_loss: 1.8421
```




```
1875/1875 - 2s - 849us/step - accuracy: 0.4642 - loss: 1.8465
Train accuracy: 46.42%
313/313 - 0s - 872us/step - accuracy: 0.4584 - loss: 1.8421

2024/04/21 15:10:33 WARNING mlflow.tensorflow: You are saving a TensorFlow Core model or Keras model without a signat
ure. Inference with mlflow.pyfunc.spark_udf() will not work unless the model's pyfunc representation accepts pandas D
ataFrames as inference inputs.

Test accuracy: 45.84%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
Predicted Digit: 3
```


Predicted Image
0
5
10
15
20
25
0
5
10
15
20
25