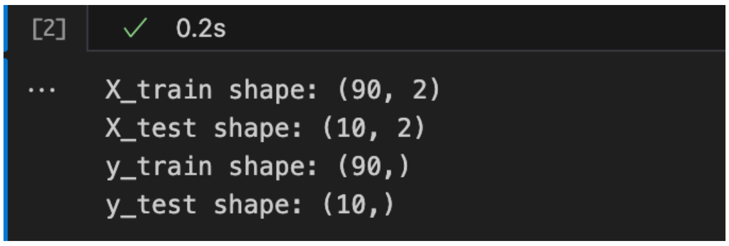
Keras DNN/CNN Report

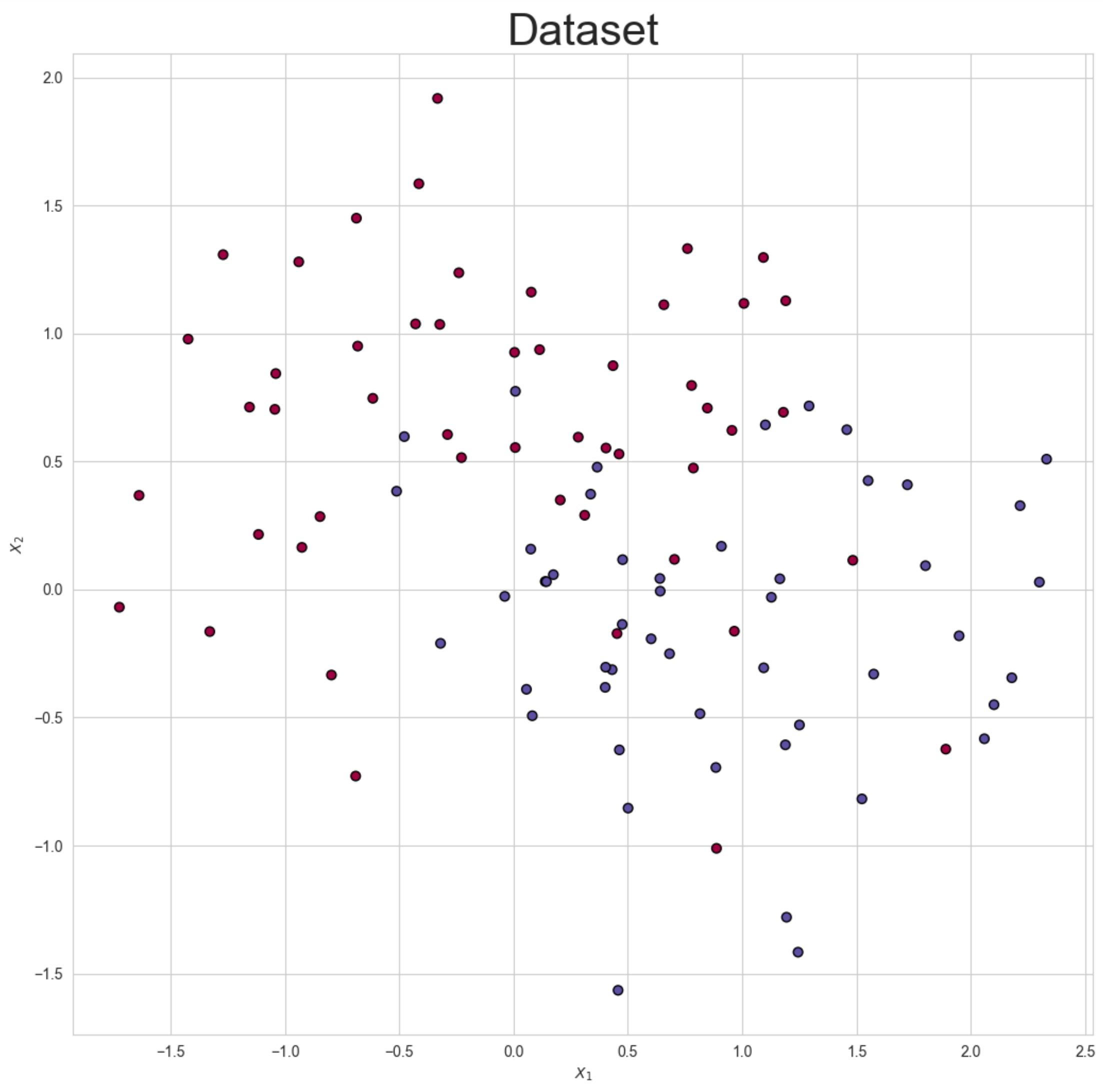
Yuhuan Huang

Part1:

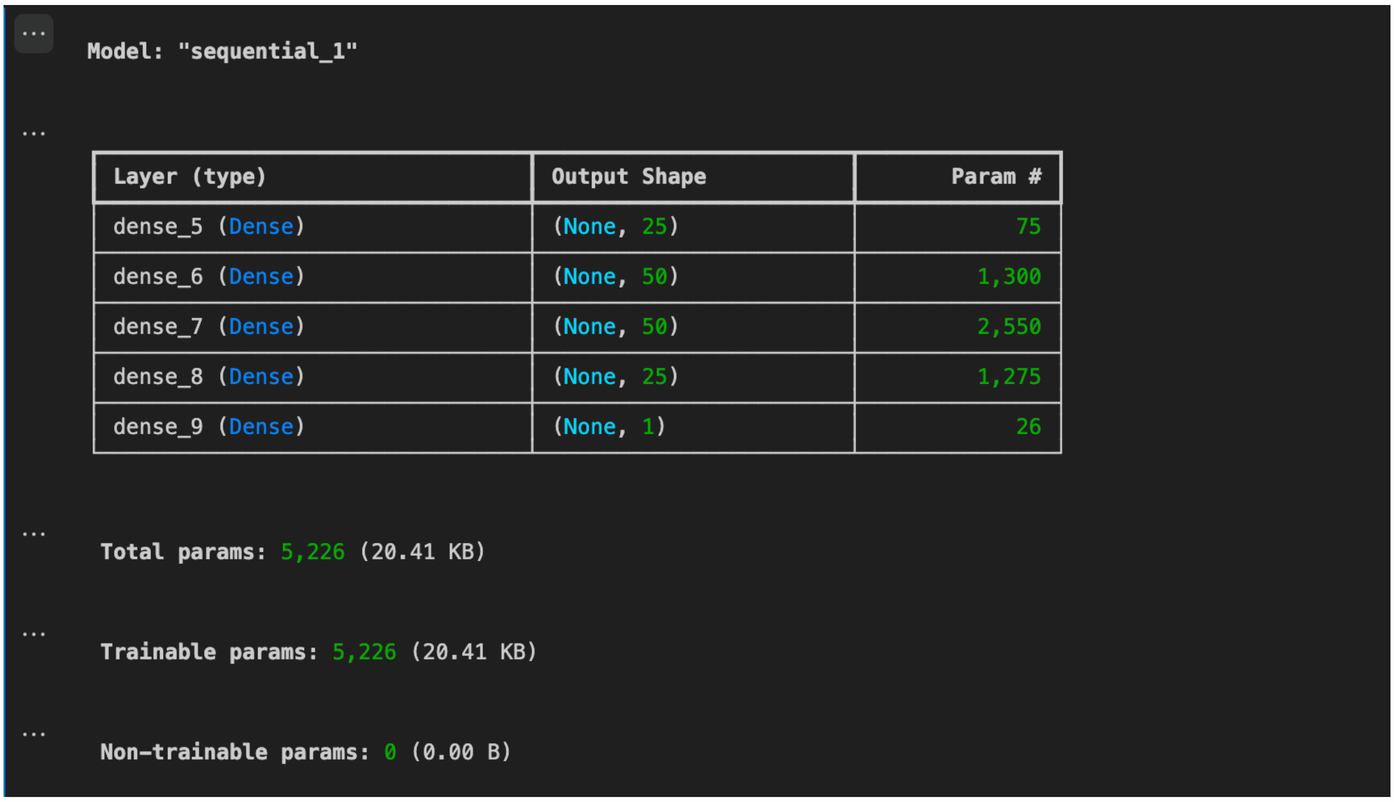
Console output:



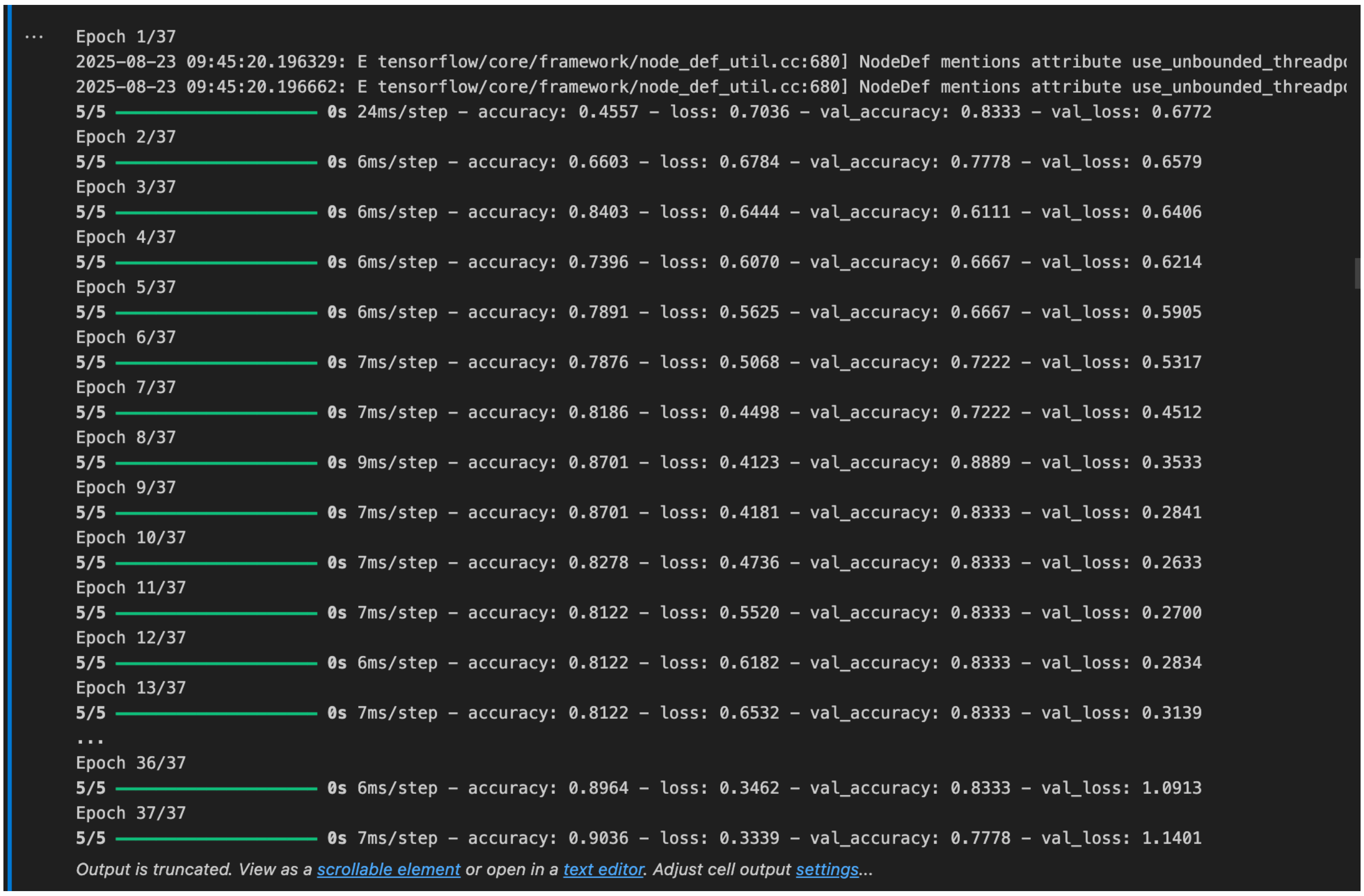
```
[2]  ✓ 0.2s

...  X_train shape: (90, 2)
     X_test shape: (10, 2)
     y_train shape: (90,)
     y_test shape: (10,)
```



RMSprop

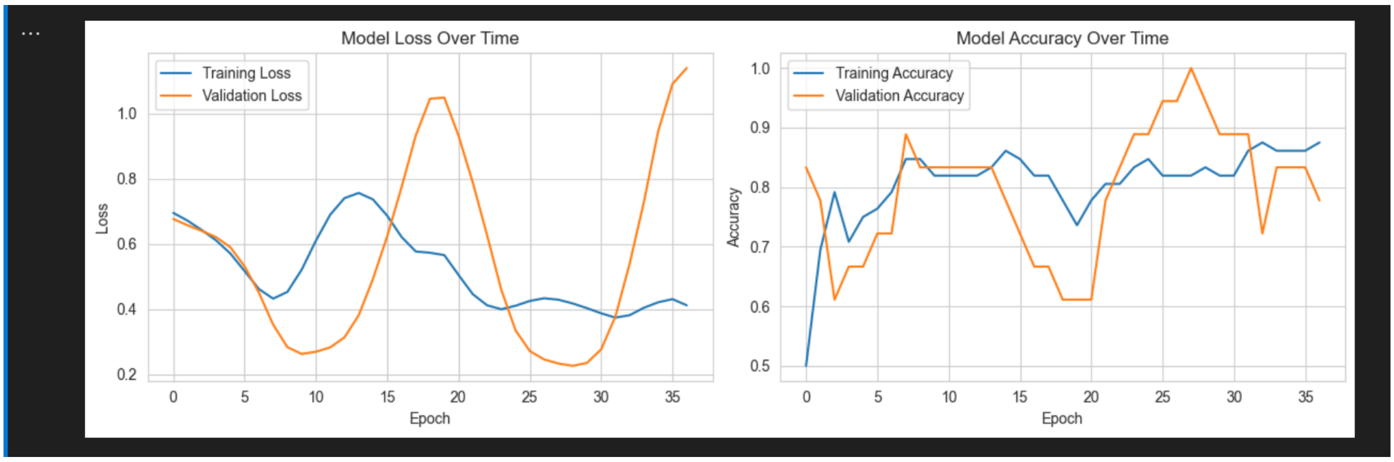


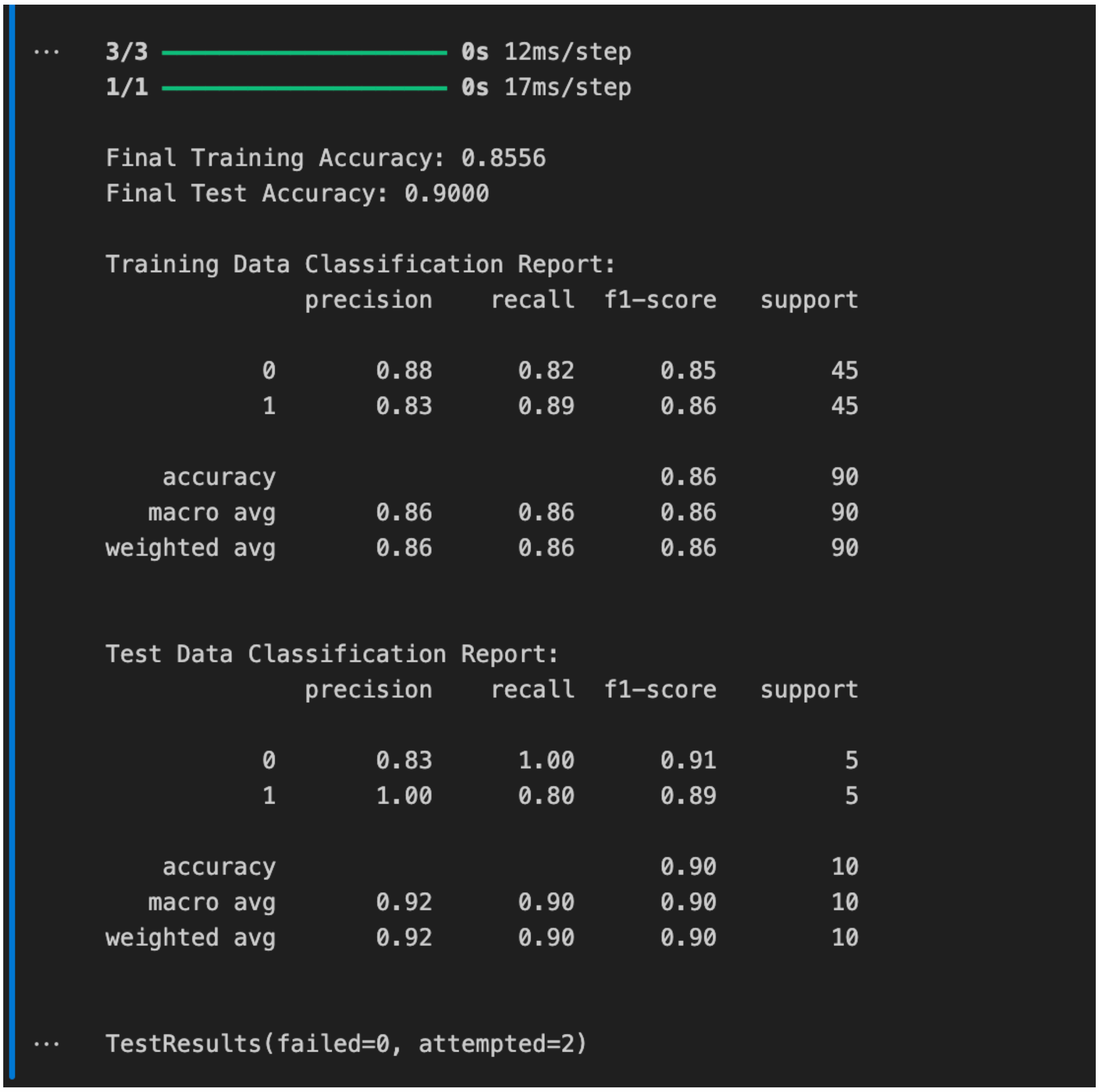
```
Model: "sequential_1"

┌──────────────────────────────┬────────────────────┬──────────────┐
│ Layer (type)                 │ Output Shape       │      Param # │
├──────────────────────────────┼────────────────────┼──────────────┤
│ dense_5 (Dense)              │ (None, 25)         │           75 │
├──────────────────────────────┼────────────────────┼──────────────┤
│ dense_6 (Dense)              │ (None, 50)         │        1,300 │
├──────────────────────────────┼────────────────────┼──────────────┤
│ dense_7 (Dense)              │ (None, 50)         │        2,550 │
├──────────────────────────────┼────────────────────┼──────────────┤
│ dense_8 (Dense)              │ (None, 25)         │        1,275 │
├──────────────────────────────┼────────────────────┼──────────────┤
│ dense_9 (Dense)              │ (None, 1)          │           26 │
└──────────────────────────────┴────────────────────┴──────────────┘

 Total params: 5,226 (20.41 KB)

 Trainable params: 5,226 (20.41 KB)

 Non-trainable params: 0 (0.00 B)
```



```
Epoch 1/37
2025-08-23 09:45:20.196329: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
2025-08-23 09:45:20.196662: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.4557 - loss: 0.7036 - val_accuracy: 0.8333 - val_loss: 0.6772
Epoch 2/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6603 - loss: 0.6784 - val_accuracy: 0.7778 - val_loss: 0.6579
Epoch 3/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8403 - loss: 0.6444 - val_accuracy: 0.6111 - val_loss: 0.6406
Epoch 4/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7396 - loss: 0.6070 - val_accuracy: 0.6667 - val_loss: 0.6214
Epoch 5/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7891 - loss: 0.5625 - val_accuracy: 0.6667 - val_loss: 0.5905
Epoch 6/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7876 - loss: 0.5068 - val_accuracy: 0.7222 - val_loss: 0.5317
Epoch 7/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8186 - loss: 0.4498 - val_accuracy: 0.7222 - val_loss: 0.4512
Epoch 8/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8701 - loss: 0.4123 - val_accuracy: 0.8889 - val_loss: 0.3533
Epoch 9/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8701 - loss: 0.4181 - val_accuracy: 0.8333 - val_loss: 0.2841
Epoch 10/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8278 - loss: 0.4736 - val_accuracy: 0.8333 - val_loss: 0.2633
Epoch 11/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8122 - loss: 0.5520 - val_accuracy: 0.8333 - val_loss: 0.2700
Epoch 12/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8122 - loss: 0.6182 - val_accuracy: 0.8333 - val_loss: 0.2834
Epoch 13/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8122 - loss: 0.6532 - val_accuracy: 0.8333 - val_loss: 0.3139
...
Epoch 36/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8964 - loss: 0.3462 - val_accuracy: 0.8333 - val_loss: 1.0913
Epoch 37/37
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9036 - loss: 0.3339 - val_accuracy: 0.7778 - val_loss: 1.1401
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...





```
...   3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
      1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step

      Final Training Accuracy: 0.8556
      Final Test Accuracy: 0.9000

      Training Data Classification Report:
                    precision    recall  f1-score   support

                 0       0.88      0.82      0.85        45
                 1       0.83      0.89      0.86        45

          accuracy                           0.86        90
         macro avg       0.86      0.86      0.86        90
      weighted avg       0.86      0.86      0.86        90


      Test Data Classification Report:
                    precision    recall  f1-score   support

                 0       0.83      1.00      0.91         5
                 1       1.00      0.80      0.89         5

          accuracy                           0.90        10
         macro avg       0.92      0.90      0.90        10
      weighted avg       0.92      0.90      0.90        10


...   TestResults(failed=0, attempted=2)
```

Reflection:

1. How did your model perform as compared with the DNN framework model that you implemented previously?

   I think in this problem, no previous DNN was implemented(?)

2. How many epochs did you run the training for? Why did you choose this?

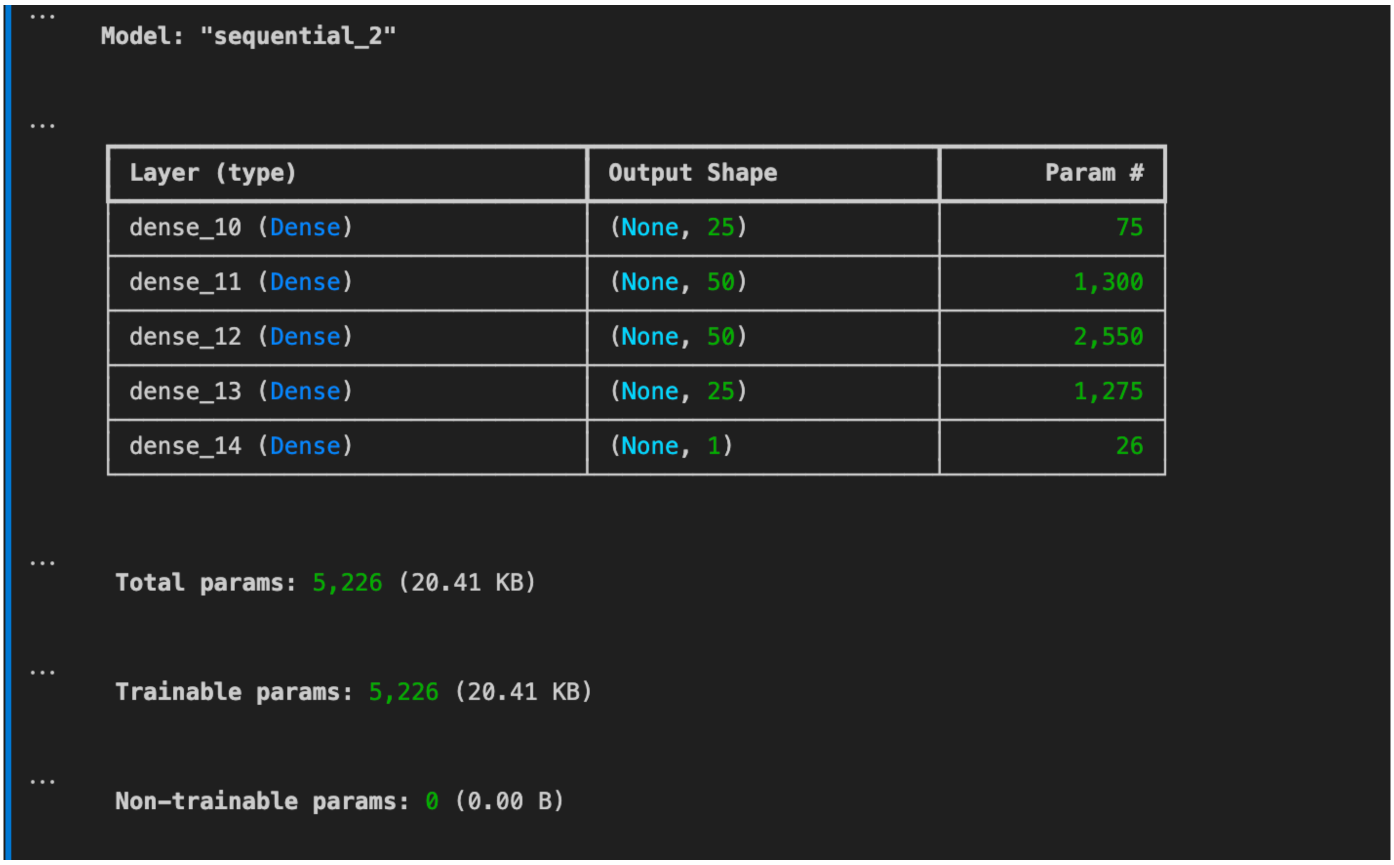
   I ran 37 epochs. Because I found that it is a good choice in terms of the training accuracy and testing accuracy.

3. How did your model speed compare with the DNN framework model that you implemented previously?

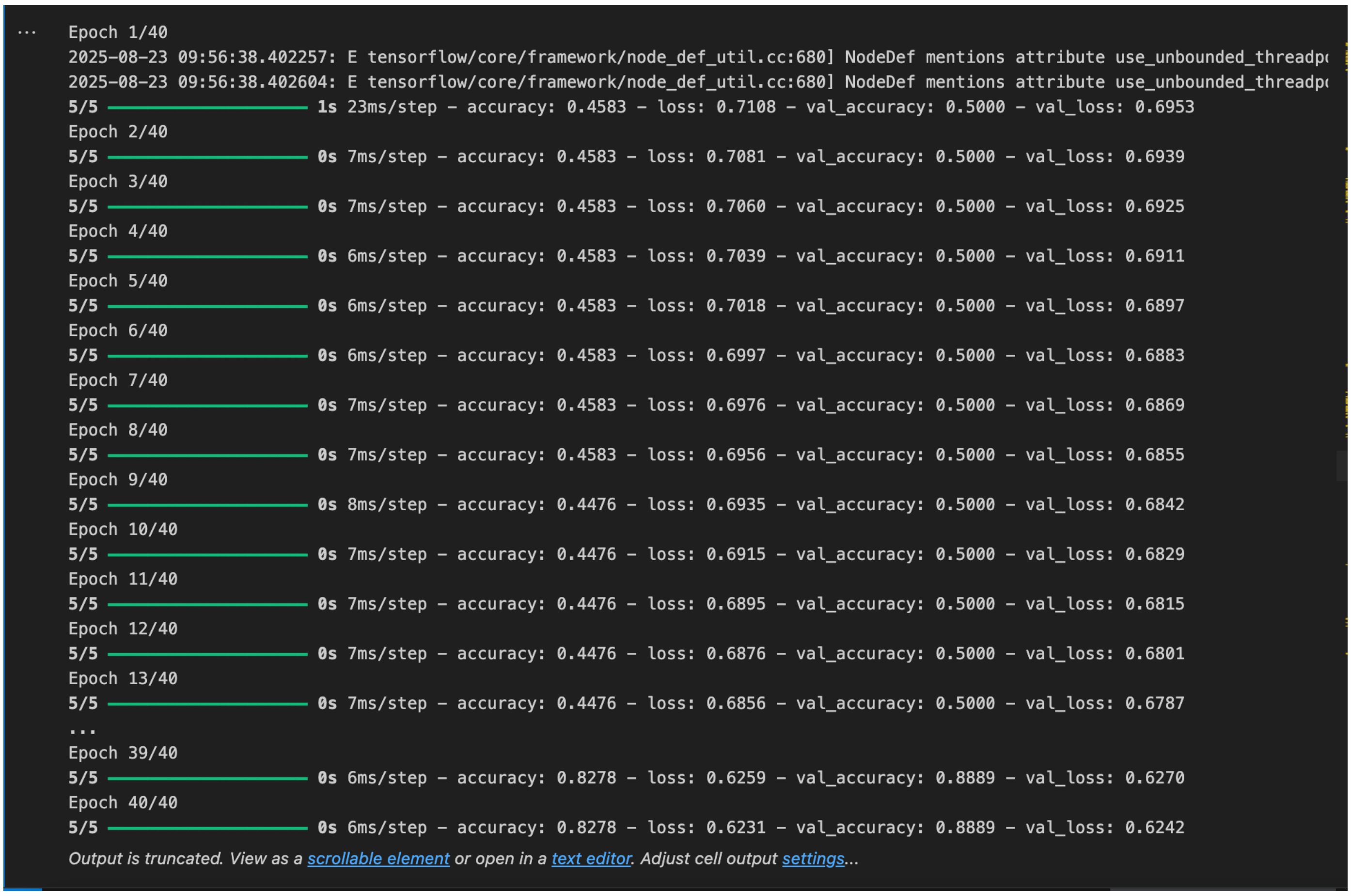
   I think in this problem, no previous DNN was implemented(?)

Adam

Console output:

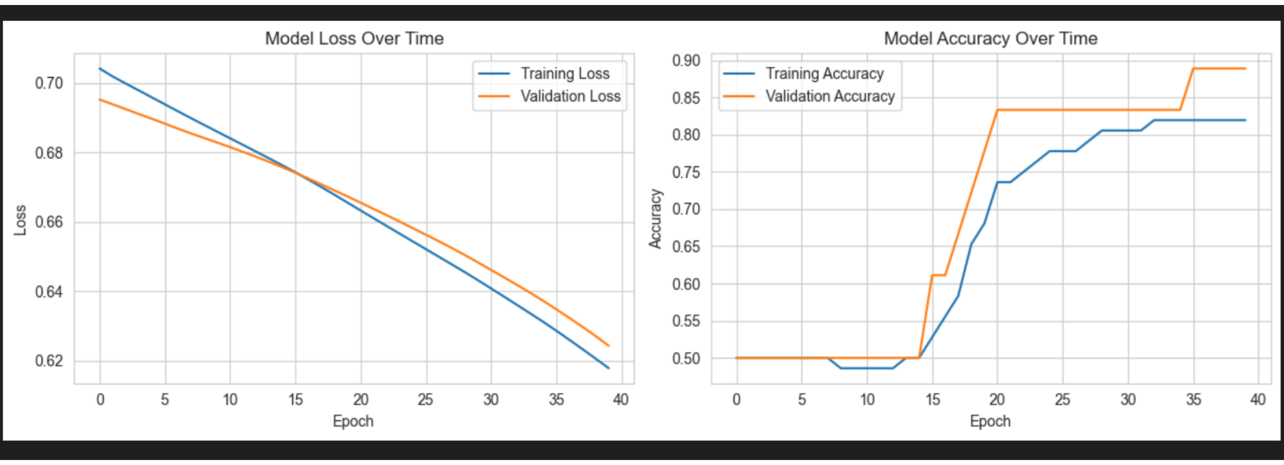


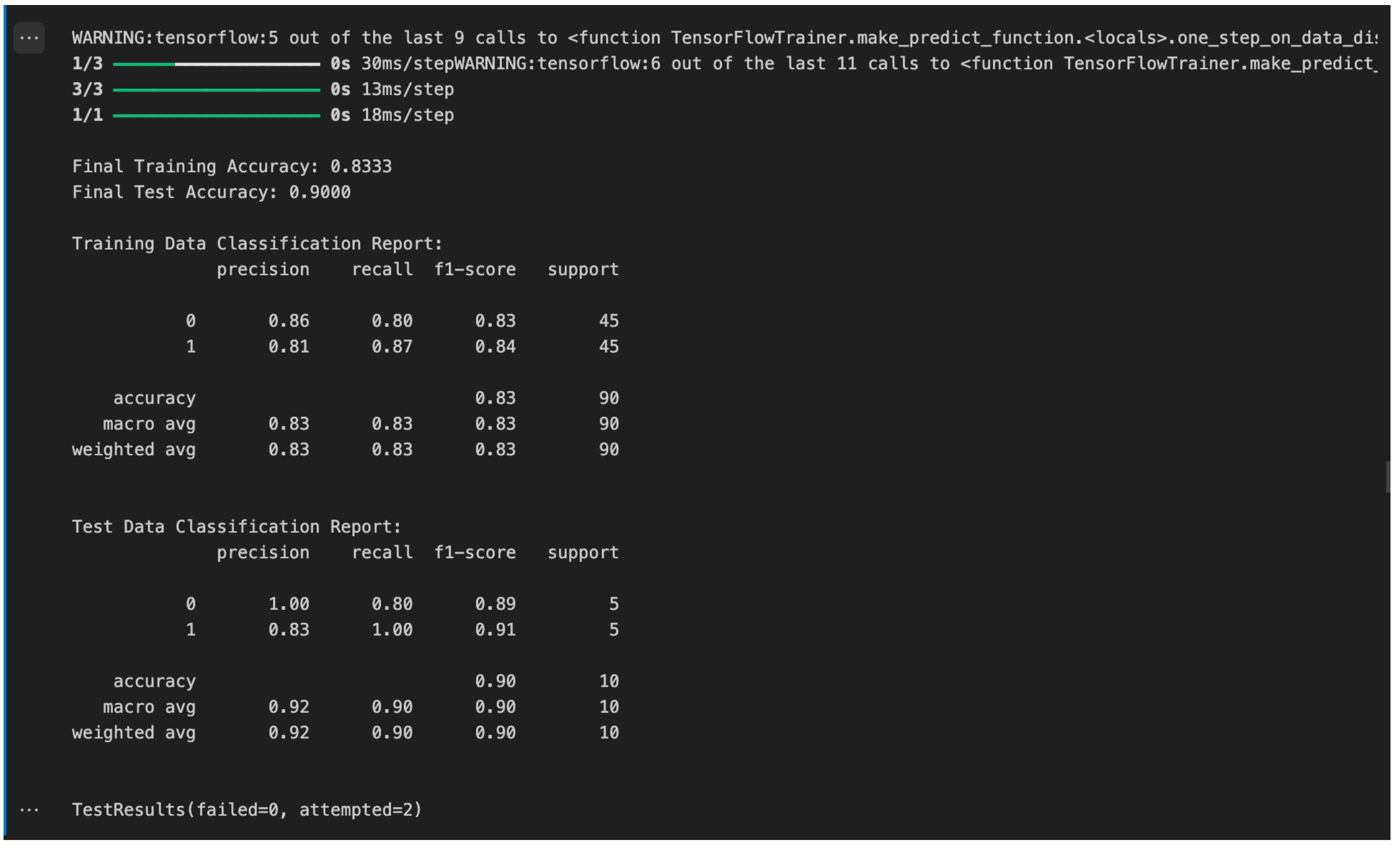
```
...
    Model: "sequential_2"

...
     ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
     ┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
     ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
     │ dense_10 (Dense)                 │ (None, 25)             │            75 │
     ├──────────────────────────────────┼────────────────────────┼───────────────┤
     │ dense_11 (Dense)                 │ (None, 50)             │         1,300 │
     ├──────────────────────────────────┼────────────────────────┼───────────────┤
     │ dense_12 (Dense)                 │ (None, 50)             │         2,550 │
     ├──────────────────────────────────┼────────────────────────┼───────────────┤
     │ dense_13 (Dense)                 │ (None, 25)             │         1,275 │
     ├──────────────────────────────────┼────────────────────────┼───────────────┤
     │ dense_14 (Dense)                 │ (None, 1)              │            26 │
     └──────────────────────────────────┴────────────────────────┴───────────────┘

...
     Total params: 5,226 (20.41 KB)

...
     Trainable params: 5,226 (20.41 KB)

...
     Non-trainable params: 0 (0.00 B)
```



```
...   Epoch 1/40
      2025-08-23 09:56:38.402257: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
      2025-08-23 09:56:38.402604: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
      5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.4583 - loss: 0.7108 - val_accuracy: 0.5000 - val_loss: 0.6953
      Epoch 2/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4583 - loss: 0.7081 - val_accuracy: 0.5000 - val_loss: 0.6939
      Epoch 3/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4583 - loss: 0.7060 - val_accuracy: 0.5000 - val_loss: 0.6925
      Epoch 4/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4583 - loss: 0.7039 - val_accuracy: 0.5000 - val_loss: 0.6911
      Epoch 5/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4583 - loss: 0.7018 - val_accuracy: 0.5000 - val_loss: 0.6897
      Epoch 6/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4583 - loss: 0.6997 - val_accuracy: 0.5000 - val_loss: 0.6883
      Epoch 7/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4583 - loss: 0.6976 - val_accuracy: 0.5000 - val_loss: 0.6869
      Epoch 8/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4583 - loss: 0.6956 - val_accuracy: 0.5000 - val_loss: 0.6855
      Epoch 9/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.4476 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6842
      Epoch 10/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4476 - loss: 0.6915 - val_accuracy: 0.5000 - val_loss: 0.6829
      Epoch 11/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4476 - loss: 0.6895 - val_accuracy: 0.5000 - val_loss: 0.6815
      Epoch 12/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4476 - loss: 0.6876 - val_accuracy: 0.5000 - val_loss: 0.6801
      Epoch 13/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.4476 - loss: 0.6856 - val_accuracy: 0.5000 - val_loss: 0.6787
      ...
      Epoch 39/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8278 - loss: 0.6259 - val_accuracy: 0.8889 - val_loss: 0.6270
      Epoch 40/40
      5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8278 - loss: 0.6231 - val_accuracy: 0.8889 - val_loss: 0.6242
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...





```
WARNING:tensorflow:5 out of the last 9 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_di
1/3 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/stepWARNING:tensorflow:6 out of the last 11 calls to <function TensorFlowTrainer.make_predict_
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step

Final Training Accuracy: 0.8333
Final Test Accuracy: 0.9000

Training Data Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.80      0.83        45
           1       0.81      0.87      0.84        45

    accuracy                           0.83        90
   macro avg       0.83      0.83      0.83        90
weighted avg       0.83      0.83      0.83        90


Test Data Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.80      0.89         5
           1       0.83      1.00      0.91         5

    accuracy                           0.90        10
   macro avg       0.92      0.90      0.90        10
weighted avg       0.92      0.90      0.90        10


TestResults(failed=0, attempted=2)
```

Reflection:

1. How did your model perform as compared with the DNN framework model that you implemented previously?

   Compared to the previous DNN model, that is, the RMSprop, I think the performance is similar(?). Both have relatively high training accuracy (0.8+) and high testing accuracy (0.9+)

2. How many epochs did you run the training for? Why did you choose this?

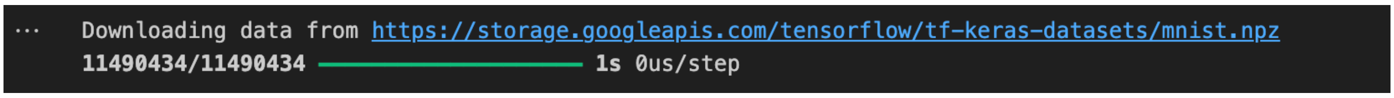
   I ran 40 epochs. Because I found that it is a good choice in terms of the training accuracy and testing accuracy.

3. How did your model speed compare with the DNN framework model that you implemented previously?

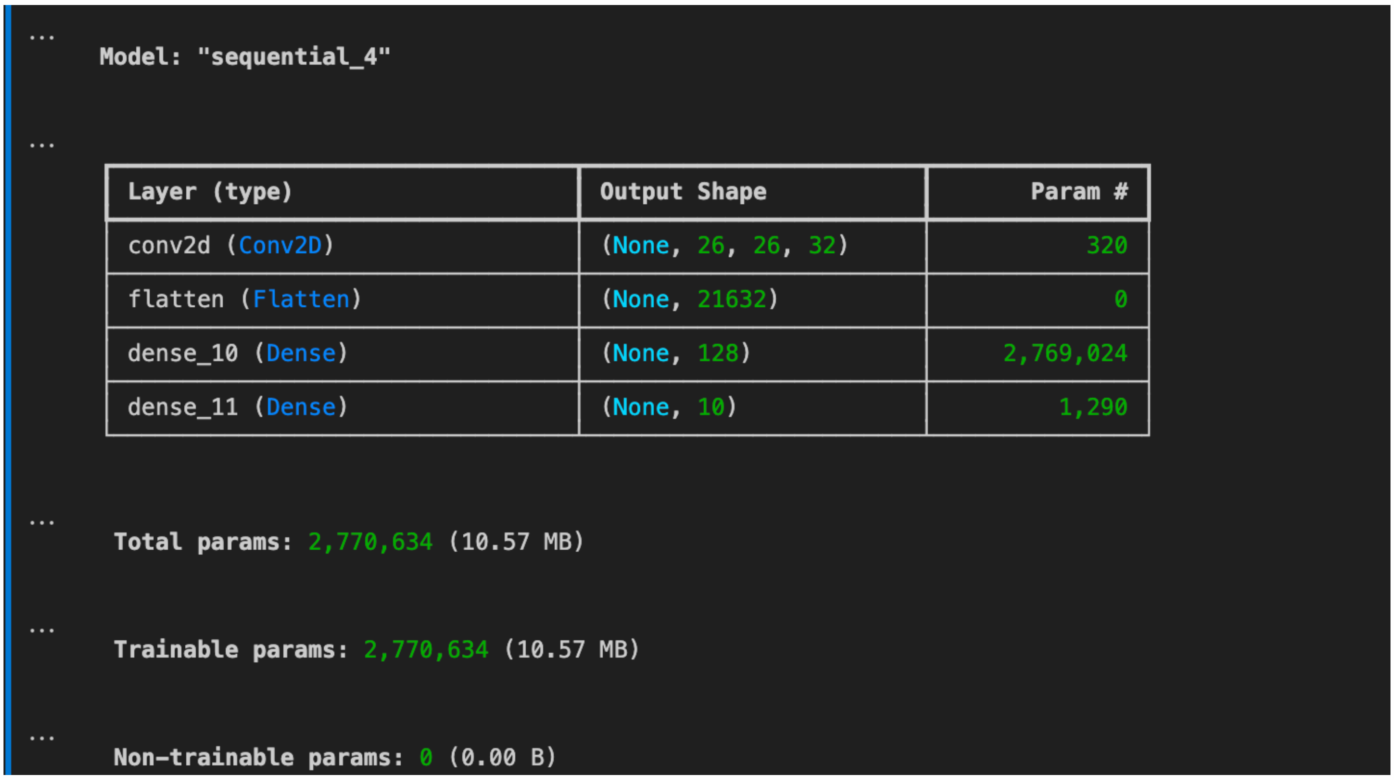
   I think the two models have similar speed in this small-epochs, but when I increased the epoch numbers, I think Adam would be a bit quicklier than the RMSprop.

Part 2

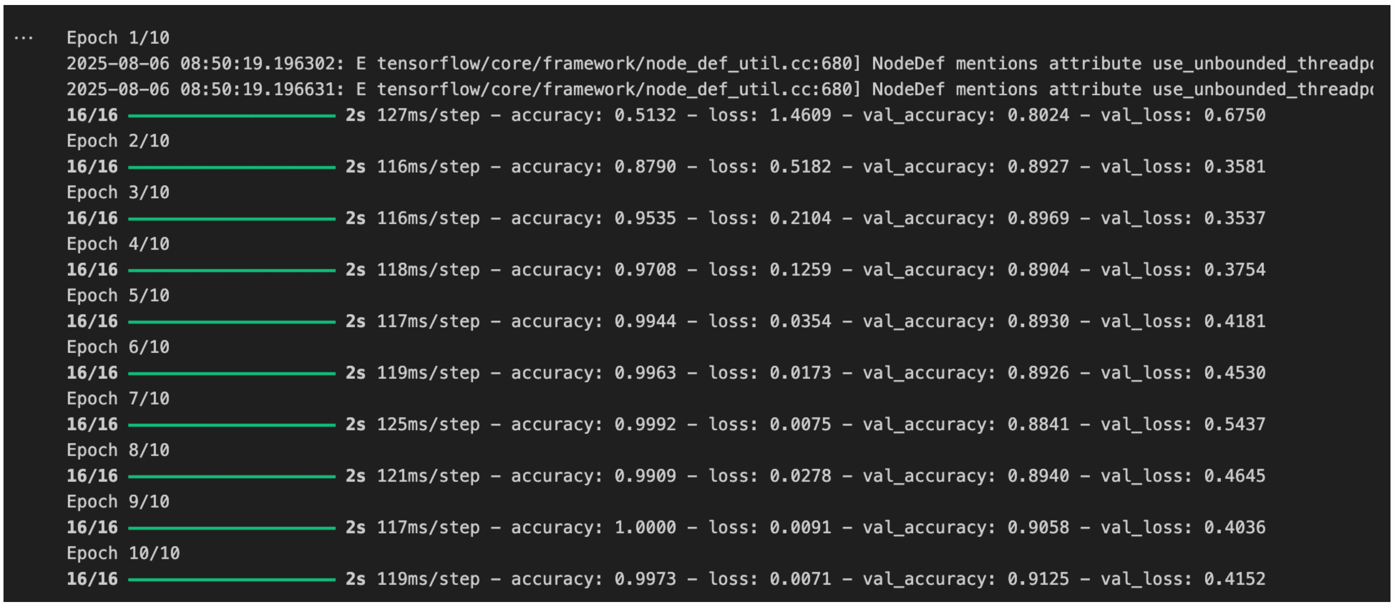
Console output:



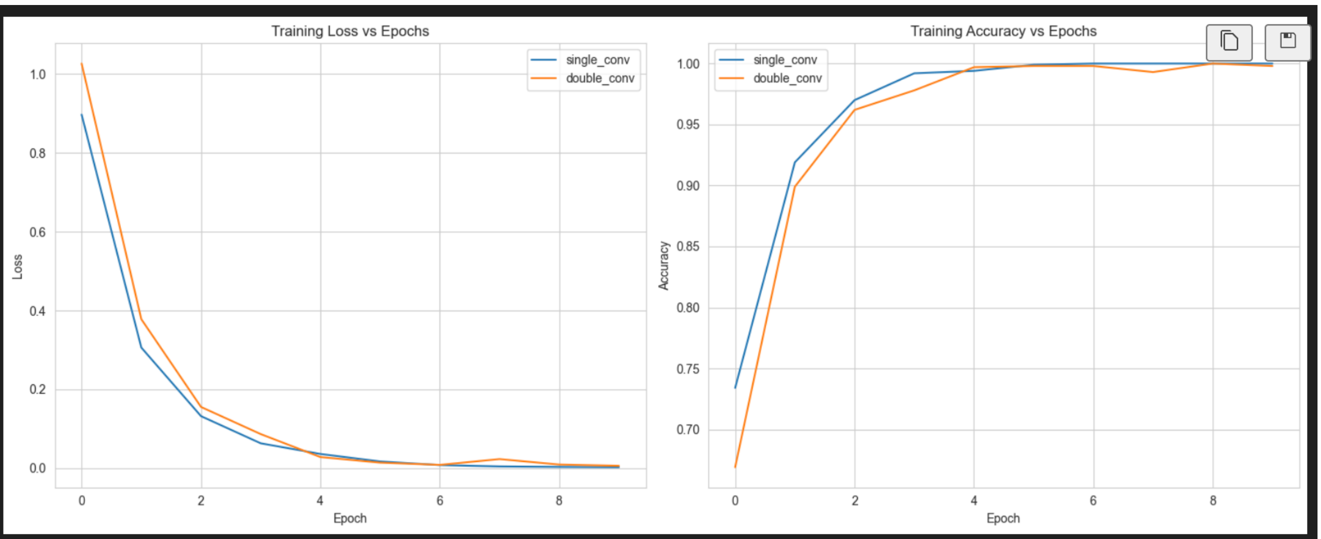
```
...   Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
      11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

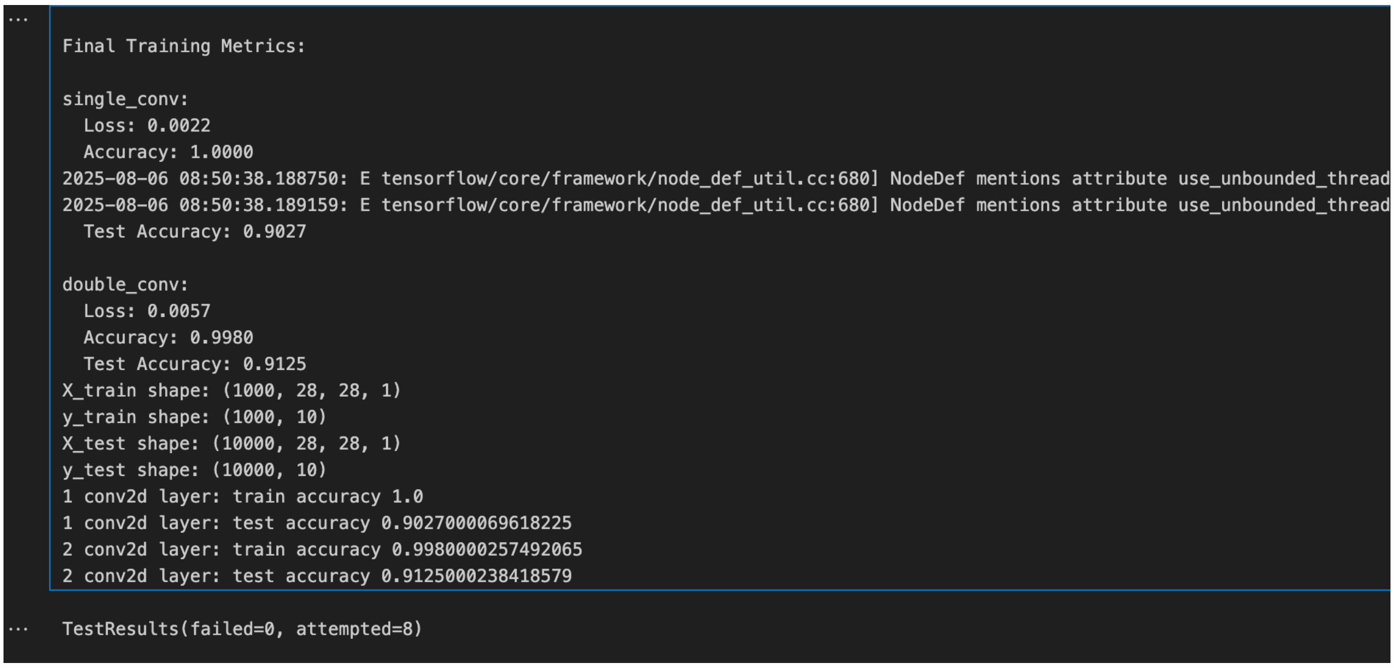


```
...
      Model: "sequential_4"

...
      ┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
      ┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
      ┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
      │ conv2d (Conv2D)                 │ (None, 26, 26, 32)     │           320 │
      ├─────────────────────────────────┼────────────────────────┼───────────────┤
      │ flatten (Flatten)               │ (None, 21632)          │             0 │
      ├─────────────────────────────────┼────────────────────────┼───────────────┤
      │ dense_10 (Dense)                │ (None, 128)            │     2,769,024 │
      ├─────────────────────────────────┼────────────────────────┼───────────────┤
      │ dense_11 (Dense)                │ (None, 10)             │         1,290 │
      └─────────────────────────────────┴────────────────────────┴───────────────┘

...
       Total params: 2,770,634 (10.57 MB)

...
       Trainable params: 2,770,634 (10.57 MB)

...
       Non-trainable params: 0 (0.00 B)
```



```
...   Epoch 1/10
      2025-08-06 08:50:19.196302: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
      2025-08-06 08:50:19.196631: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_threadp
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 127ms/step - accuracy: 0.5132 - loss: 1.4609 - val_accuracy: 0.8024 - val_loss: 0.6750
      Epoch 2/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 116ms/step - accuracy: 0.8790 - loss: 0.5182 - val_accuracy: 0.8927 - val_loss: 0.3581
      Epoch 3/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 116ms/step - accuracy: 0.9535 - loss: 0.2104 - val_accuracy: 0.8969 - val_loss: 0.3537
      Epoch 4/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 118ms/step - accuracy: 0.9708 - loss: 0.1259 - val_accuracy: 0.8904 - val_loss: 0.3754
      Epoch 5/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 117ms/step - accuracy: 0.9944 - loss: 0.0354 - val_accuracy: 0.8930 - val_loss: 0.4181
      Epoch 6/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 119ms/step - accuracy: 0.9963 - loss: 0.0173 - val_accuracy: 0.8926 - val_loss: 0.4530
      Epoch 7/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 125ms/step - accuracy: 0.9992 - loss: 0.0075 - val_accuracy: 0.8841 - val_loss: 0.5437
      Epoch 8/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 121ms/step - accuracy: 0.9909 - loss: 0.0278 - val_accuracy: 0.8940 - val_loss: 0.4645
      Epoch 9/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 117ms/step - accuracy: 1.0000 - loss: 0.0091 - val_accuracy: 0.9058 - val_loss: 0.4036
      Epoch 10/10
      16/16 ━━━━━━━━━━━━━━━━━━━━ 2s 119ms/step - accuracy: 0.9973 - loss: 0.0071 - val_accuracy: 0.9125 - val_loss: 0.4152
```





```
Final Training Metrics:

single_conv:
  Loss: 0.0022
  Accuracy: 1.0000
2025-08-06 08:50:38.188750: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_thread
2025-08-06 08:50:38.189159: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentions attribute use_unbounded_thread
  Test Accuracy: 0.9027

double_conv:
  Loss: 0.0057
  Accuracy: 0.9980
  Test Accuracy: 0.9125
X_train shape: (1000, 28, 28, 1)
y_train shape: (1000, 10)
X_test shape: (10000, 28, 28, 1)
y_test shape: (10000, 10)
1 conv2d layer: train accuracy 1.0
1 conv2d layer: test accuracy 0.9027000069618225
2 conv2d layer: train accuracy 0.9980000257492065
2 conv2d layer: test accuracy 0.9125000238418579

TestResults(failed=0, attempted=8)
```

CNN Reflections:

1. How did your model perform as compared with the CNN+DNN framework model that you implemented previously?

   I think this model outperforms the DNN models I implemented previously. Because it has higher train and test accuracy, and it takes fewer epochs to get there.

2. How many epochs did you run the training for? Why did you choose this?

   I chose 10. It is a small number, but from the graphs we can see that the curves quickly become flat (quite steep at the first few epochs), and the accuracy looks good.

3. How did your model speed compare with the DNN framework model that you implemented previously?

It is more slowly in each epoch (about 100 ms for this model, but less than 10 ms for the DNN models); however, it takes much fewer epochs to get to a satisfactory result.