Name: Rohini Patturaja

Subject: Big Data Analytics

Course ID: DSA 5620

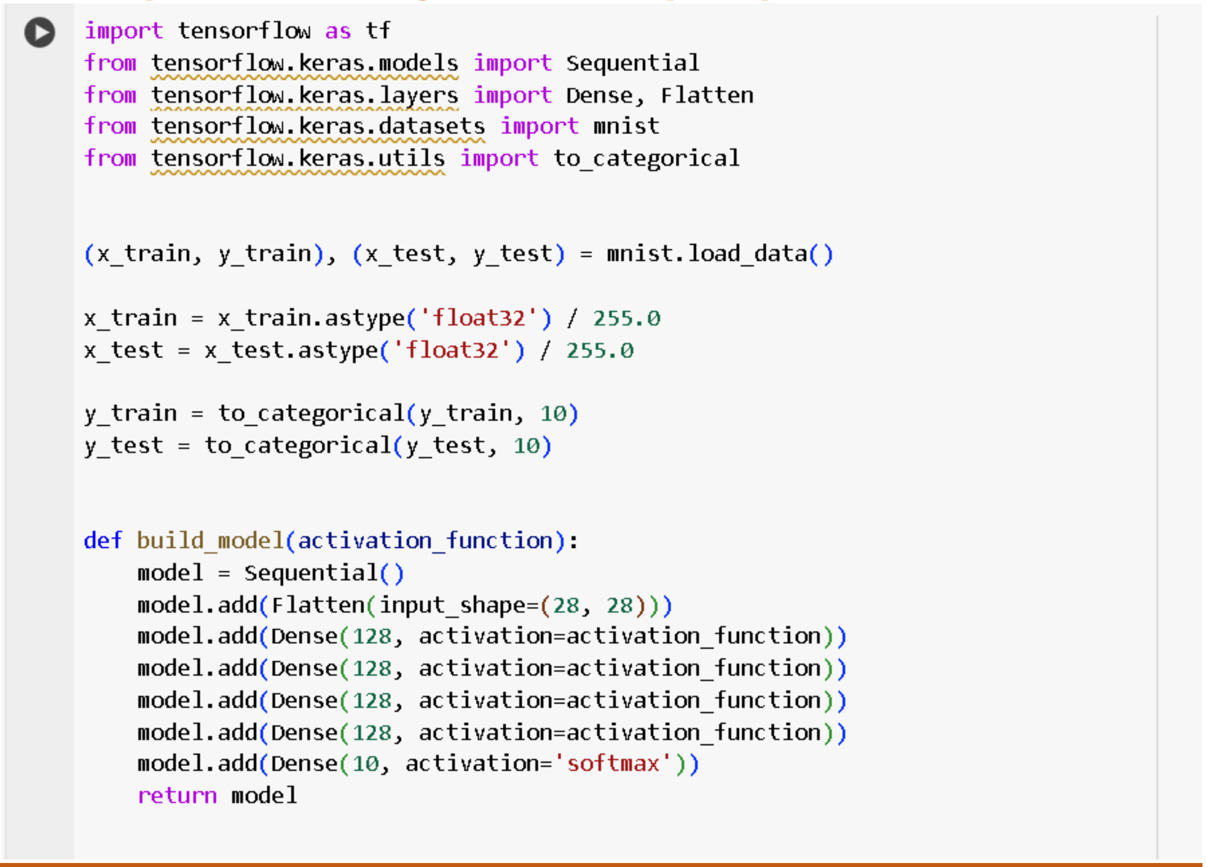
Email: rxp36770@ucmo.edu

Github link: Github ICP4 link

Video link: Rohini_Patturaja_ICP4.mp4

Use the Sequential API to build a simple feedforward neural network. Include 5 hidden layers with ReLU activation and an output layer with Softmax activation. Try adding more layers or using a different number of neurons.

Experiment with different activation functions (e.g., tanh, sigmoid). Compare results with different optimizers like sgd, rmsprop, etc. Your model should achieve a test accuracy above 99%, given the simplicity of the dataset.



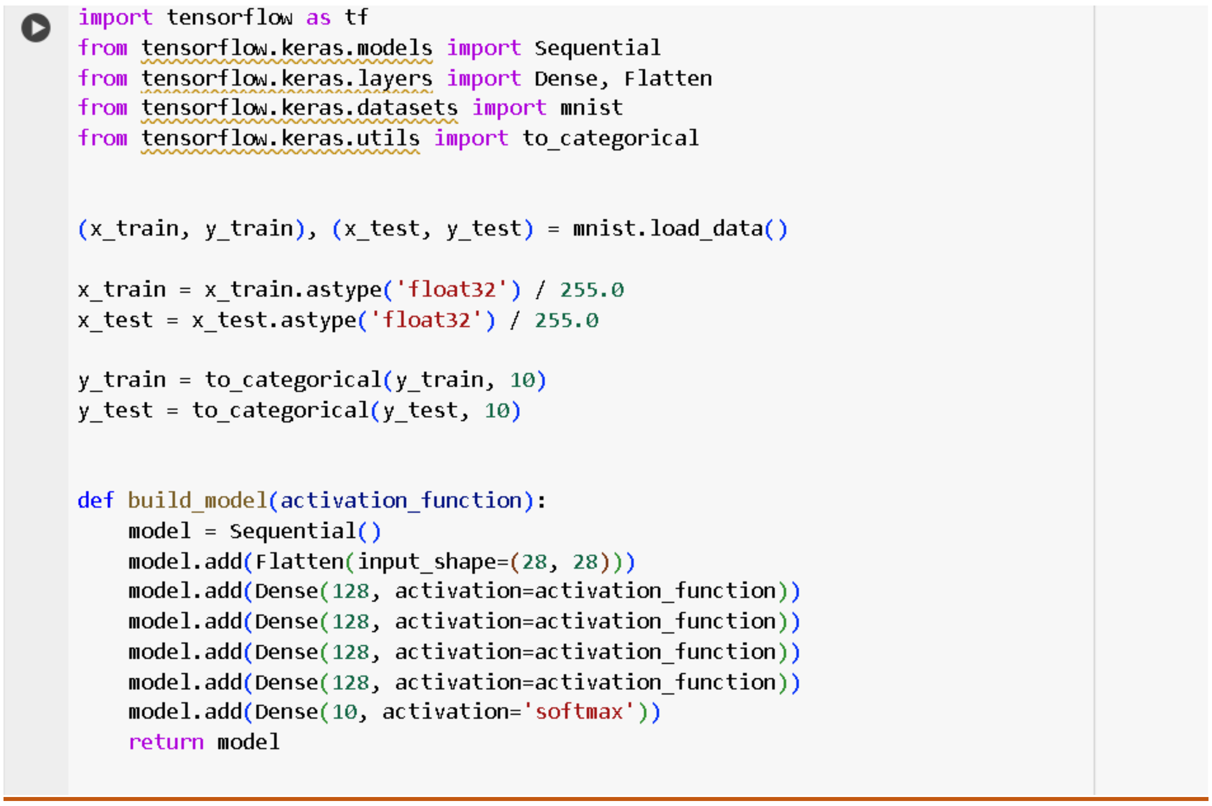
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)


def build_model(activation_function):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(10, activation='softmax'))
    return model
```

```python
def train_and_evaluate_model(activation_function, optimizer='adam'):
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f'Activation Function: {activation_function} - Test accuracy: {test_acc}')

# Test with ReLU
train_and_evaluate_model('relu')

# Test with tanh
train_and_evaluate_model('tanh')

# Test with sigmoid
train_and_evaluate_model('sigmoid')
```

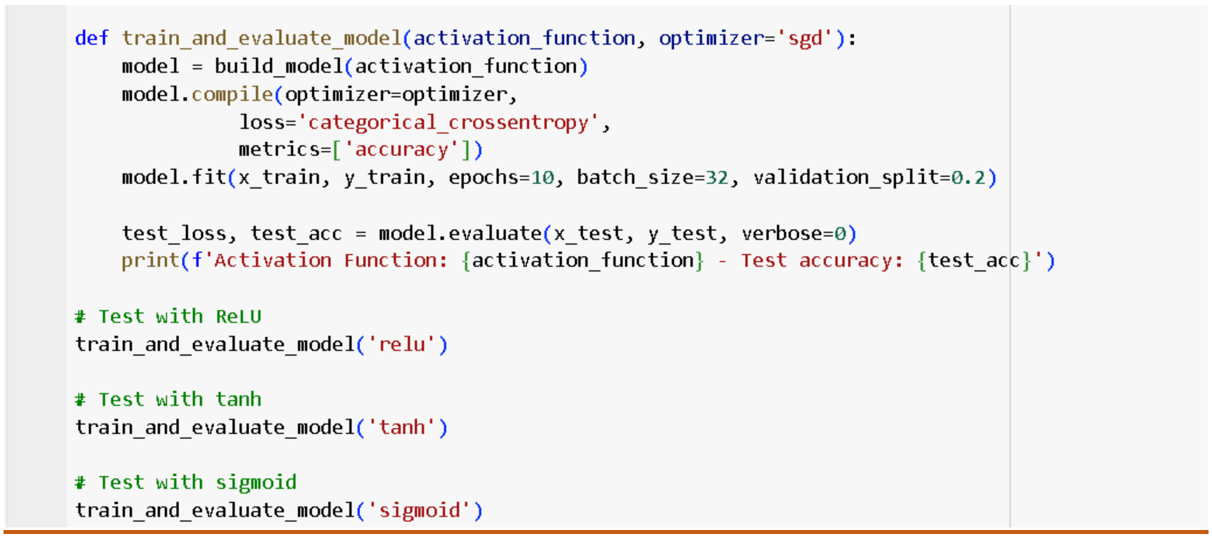
```
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9929 - loss: 0.0225 - val_accuracy: 0.9761 - val_loss: 0.1090
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.9948 - loss: 0.0173 - val_accuracy: 0.9721 - val_loss: 0.1353
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9955 - loss: 0.0160 - val_accuracy: 0.9765 - val_loss: 0.1208
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9956 - loss: 0.0149 - val_accuracy: 0.9743 - val_loss: 0.1312
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9957 - loss: 0.0136 - val_accuracy: 0.9787 - val_loss: 0.1196
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9953 - loss: 0.0150 - val_accuracy: 0.9728 - val_loss: 0.1393
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.9960 - loss: 0.0130 - val_accuracy: 0.9749 - val_loss: 0.1385
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9946 - loss: 0.0181 - val_accuracy: 0.9743 - val_loss: 0.1543
Activation Function: relu - Test accuracy: 0.9747999906539917
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.9959 - loss: 0.0162 - val_accuracy: 0.9777 - val_loss: 0.1464
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9972 - loss: 0.0098 - val_accuracy: 0.9772 - val_loss: 0.1439
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9972 - loss: 0.0112 - val_accuracy: 0.9796 - val_loss: 0.1292
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9957 - loss: 0.0174 - val_accuracy: 0.9772 - val_loss: 0.1566
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9974 - loss: 0.0092 - val_accuracy: 0.9765 - val_loss: 0.1626
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.9970 - loss: 0.0114 - val_accuracy: 0.9741 - val_loss: 0.1604
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9968 - loss: 0.0117 - val_accuracy: 0.9768 - val_loss: 0.1457
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9974 - loss: 0.0088 - val_accuracy: 0.9786 - val_loss: 0.1482
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9964 - loss: 0.0123 - val_accuracy: 0.9778 - val_loss: 0.1577
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9973 - loss: 0.0092 - val_accuracy: 0.9768 - val_loss: 0.1662
Activation Function: tanh - Test accuracy: 0.9797000288963318
```

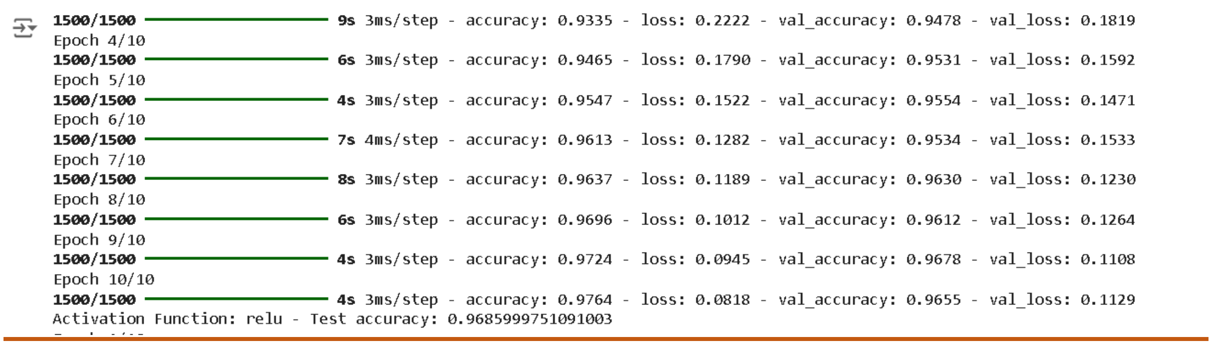
```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9973 - loss: 0.0156 - val_accuracy: 0.9764 - val_loss: 0.2113
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9979 - loss: 0.0076 - val_accuracy: 0.9744 - val_loss: 0.1877
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.9970 - loss: 0.0107 - val_accuracy: 0.9752 - val_loss: 0.2064
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9968 - loss: 0.0124 - val_accuracy: 0.9784 - val_loss: 0.1501
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.9974 - loss: 0.0103 - val_accuracy: 0.9756 - val_loss: 0.2157
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9970 - loss: 0.0142 - val_accuracy: 0.9781 - val_loss: 0.1651
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9977 - loss: 0.0112 - val_accuracy: 0.9744 - val_loss: 0.1748
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9981 - loss: 0.0072 - val_accuracy: 0.9762 - val_loss: 0.1928
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.9985 - loss: 0.0064 - val_accuracy: 0.9770 - val_loss: 0.2159
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9980 - loss: 0.0074 - val_accuracy: 0.9789 - val_loss: 0.1617
Activation Function: sigmoid - Test accuracy: 0.979200005531311
```



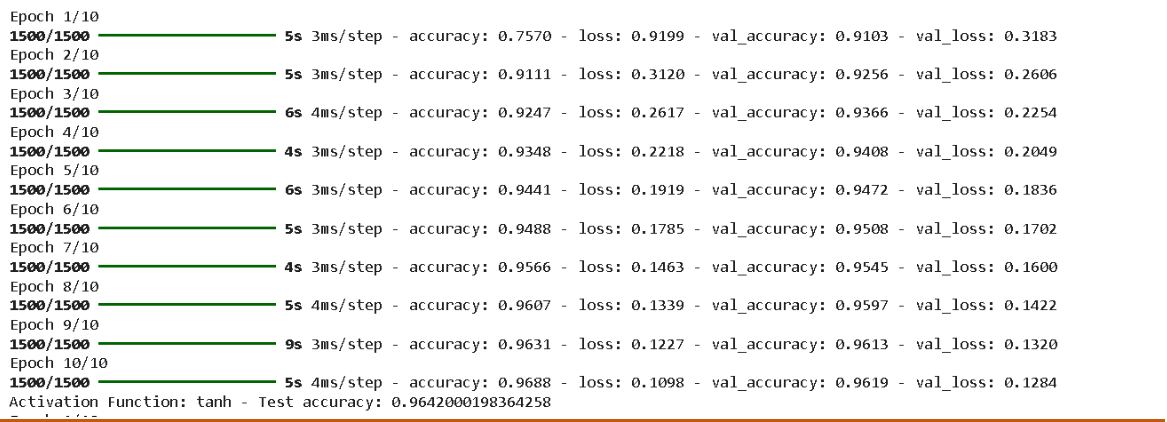
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)


def build_model(activation_function):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(10, activation='softmax'))
    return model
```



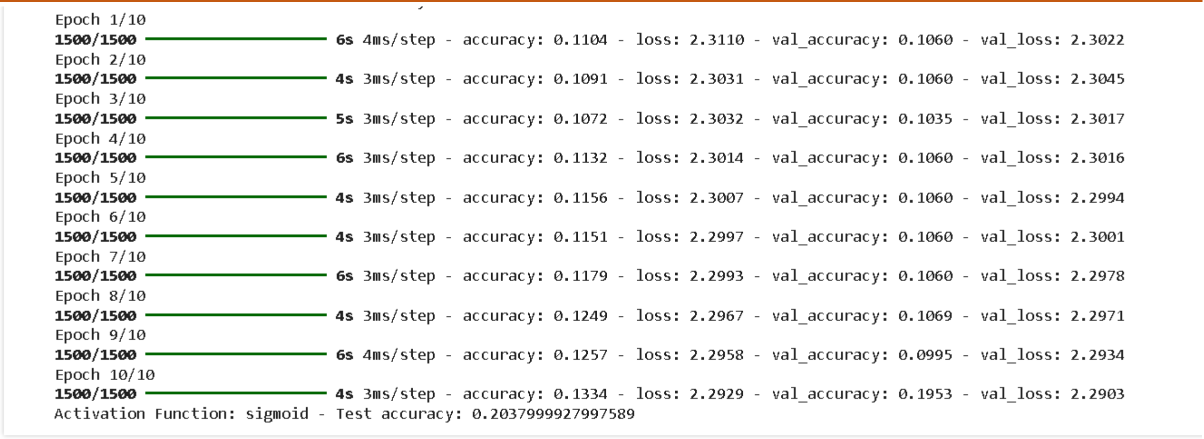
```python
def train_and_evaluate_model(activation_function, optimizer='sgd'):
    model = build_model(activation_function)
    model.compile(optimizer=optimizer,
             loss='categorical_crossentropy',
             metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f'Activation Function: {activation_function} - Test accuracy: {test_acc}')

# Test with ReLU
train_and_evaluate_model('relu')

# Test with tanh
train_and_evaluate_model('tanh')

# Test with sigmoid
train_and_evaluate_model('sigmoid')
```



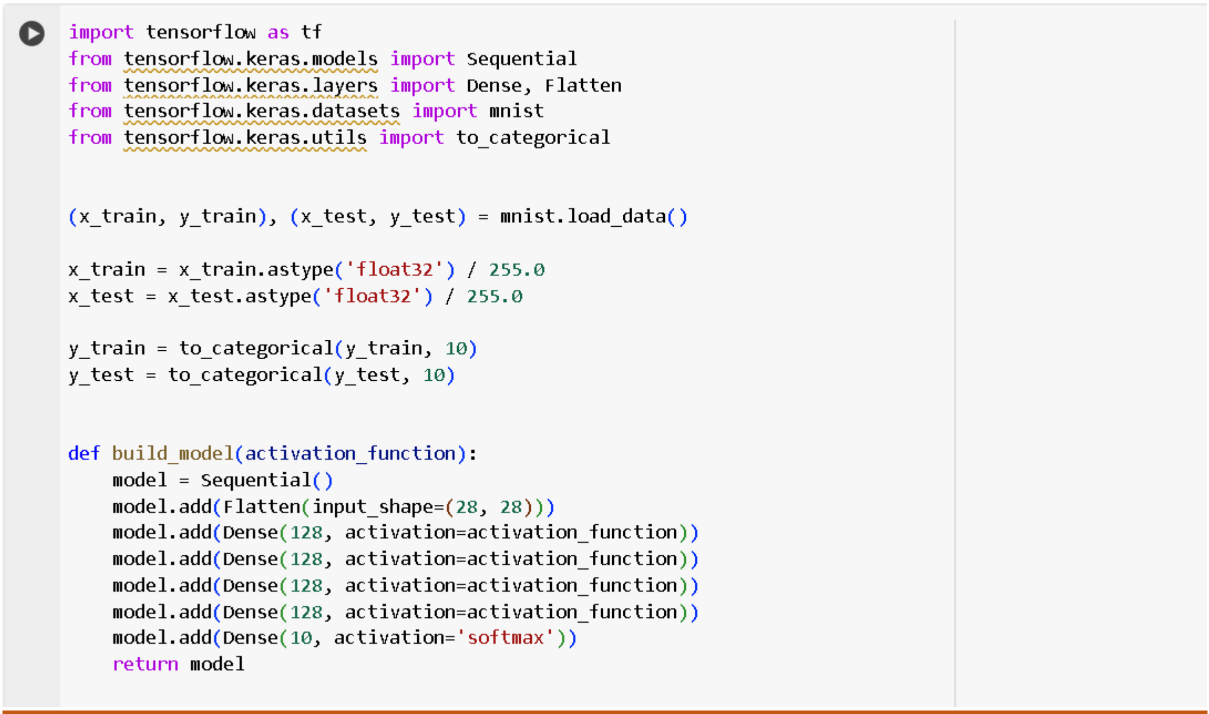
```
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9335 - loss: 0.2222 - val_accuracy: 0.9478 - val_loss: 0.1819
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9465 - loss: 0.1790 - val_accuracy: 0.9531 - val_loss: 0.1592
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9547 - loss: 0.1522 - val_accuracy: 0.9554 - val_loss: 0.1471
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9613 - loss: 0.1282 - val_accuracy: 0.9534 - val_loss: 0.1533
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.9637 - loss: 0.1189 - val_accuracy: 0.9630 - val_loss: 0.1230
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9696 - loss: 0.1012 - val_accuracy: 0.9612 - val_loss: 0.1264
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9724 - loss: 0.0945 - val_accuracy: 0.9678 - val_loss: 0.1108
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9764 - loss: 0.0818 - val_accuracy: 0.9655 - val_loss: 0.1129
Activation Function: relu - Test accuracy: 0.9685999751091003
```



```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7570 - loss: 0.9199 - val_accuracy: 0.9103 - val_loss: 0.3183
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9111 - loss: 0.3120 - val_accuracy: 0.9256 - val_loss: 0.2606
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9247 - loss: 0.2617 - val_accuracy: 0.9366 - val_loss: 0.2254
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9348 - loss: 0.2218 - val_accuracy: 0.9408 - val_loss: 0.2049
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9441 - loss: 0.1919 - val_accuracy: 0.9472 - val_loss: 0.1836
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9488 - loss: 0.1785 - val_accuracy: 0.9508 - val_loss: 0.1702
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9566 - loss: 0.1463 - val_accuracy: 0.9545 - val_loss: 0.1600
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9607 - loss: 0.1339 - val_accuracy: 0.9597 - val_loss: 0.1422
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9631 - loss: 0.1227 - val_accuracy: 0.9613 - val_loss: 0.1320
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9688 - loss: 0.1098 - val_accuracy: 0.9619 - val_loss: 0.1284
Activation Function: tanh - Test accuracy: 0.9642000198364258
```



```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.1104 - loss: 2.3110 - val_accuracy: 0.1060 - val_loss: 2.3022
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1091 - loss: 2.3031 - val_accuracy: 0.1060 - val_loss: 2.3045
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1072 - loss: 2.3032 - val_accuracy: 0.1035 - val_loss: 2.3017
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1132 - loss: 2.3014 - val_accuracy: 0.1060 - val_loss: 2.3016
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1156 - loss: 2.3007 - val_accuracy: 0.1060 - val_loss: 2.2994
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1151 - loss: 2.2997 - val_accuracy: 0.1060 - val_loss: 2.3001
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1179 - loss: 2.2993 - val_accuracy: 0.1060 - val_loss: 2.2978
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1249 - loss: 2.2967 - val_accuracy: 0.1069 - val_loss: 2.2971
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.1257 - loss: 2.2958 - val_accuracy: 0.0995 - val_loss: 2.2934
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1334 - loss: 2.2929 - val_accuracy: 0.1953 - val_loss: 2.2903
Activation Function: sigmoid - Test accuracy: 0.2037999927997589
```



```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical


(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)


def build_model(activation_function):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(128, activation=activation_function))
    model.add(Dense(10, activation='softmax'))
    return model
```

```python
def train_and_evaluate_model(activation_function, optimizer='rmsprop'):
    model = build_model(activation_function)
    model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f'Activation Function: {activation_function} - Test accuracy: {test_acc}')

# Test with ReLU
train_and_evaluate_model('relu')

# Test with tanh
train_and_evaluate_model('tanh')

# Test with sigmoid
train_and_evaluate_model('sigmoid')
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8626 - loss: 0.4443 - val_accuracy: 0.9633 - val_loss: 0.1288
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9624 - loss: 0.1286 - val_accuracy: 0.9652 - val_loss: 0.1242
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9737 - loss: 0.0887 - val_accuracy: 0.9663 - val_loss: 0.1281
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9791 - loss: 0.0752 - val_accuracy: 0.9670 - val_loss: 0.1479
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9824 - loss: 0.0634 - val_accuracy: 0.9653 - val_loss: 0.1590
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9853 - loss: 0.0585 - val_accuracy: 0.9731 - val_loss: 0.1314
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9855 - loss: 0.0523 - val_accuracy: 0.9711 - val_loss: 0.1450
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9888 - loss: 0.0457 - val_accuracy: 0.9747 - val_loss: 0.1864
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9901 - loss: 0.0443 - val_accuracy: 0.9729 - val_loss: 0.1816
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9899 - loss: 0.0433 - val_accuracy: 0.9728 - val_loss: 0.1965
Activation Function: relu - Test accuracy: 0.9717000126838684
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.8702 - loss: 0.4224 - val_accuracy: 0.9522 - val_loss: 0.1622
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9548 - loss: 0.1491 - val_accuracy: 0.9630 - val_loss: 0.1256
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9707 - loss: 0.0985 - val_accuracy: 0.9696 - val_loss: 0.1047
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9786 - loss: 0.0703 - val_accuracy: 0.9664 - val_loss: 0.1221
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9824 - loss: 0.0563 - val_accuracy: 0.9697 - val_loss: 0.1135
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9861 - loss: 0.0450 - val_accuracy: 0.9708 - val_loss: 0.1117
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9894 - loss: 0.0346 - val_accuracy: 0.9715 - val_loss: 0.1148
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9909 - loss: 0.0279 - val_accuracy: 0.9737 - val_loss: 0.1076
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9924 - loss: 0.0219 - val_accuracy: 0.9765 - val_loss: 0.0975
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9946 - loss: 0.0178 - val_accuracy: 0.9730 - val_loss: 0.1173
Activation Function: tanh - Test accuracy: 0.9722999930381775
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.4424 - loss: 1.5188 - val_accuracy: 0.8976 - val_loss: 0.3587
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9096 - loss: 0.3213 - val_accuracy: 0.9323 - val_loss: 0.2278
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9393 - loss: 0.2029 - val_accuracy: 0.9546 - val_loss: 0.1530
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9539 - loss: 0.1541 - val_accuracy: 0.9570 - val_loss: 0.1509
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9625 - loss: 0.1238 - val_accuracy: 0.9644 - val_loss: 0.1226
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9695 - loss: 0.1060 - val_accuracy: 0.9636 - val_loss: 0.1262
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9745 - loss: 0.0874 - val_accuracy: 0.9678 - val_loss: 0.1151
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9778 - loss: 0.0763 - val_accuracy: 0.9683 - val_loss: 0.1187
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9812 - loss: 0.0642 - val_accuracy: 0.9663 - val_loss: 0.1203
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9816 - loss: 0.0607 - val_accuracy: 0.9723 - val_loss: 0.0955
Activation Function: sigmoid - Test accuracy: 0.972000002861023
```