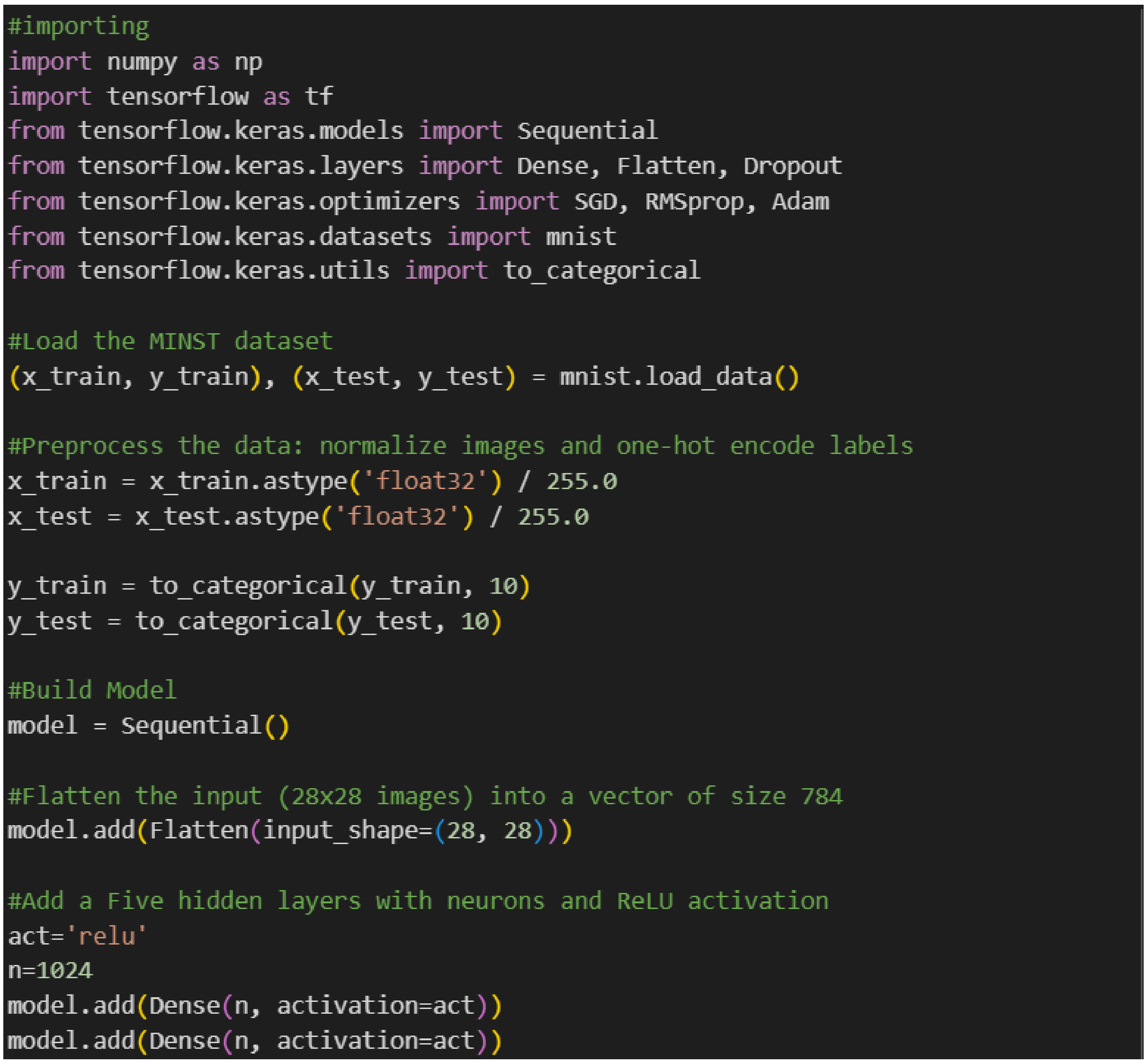
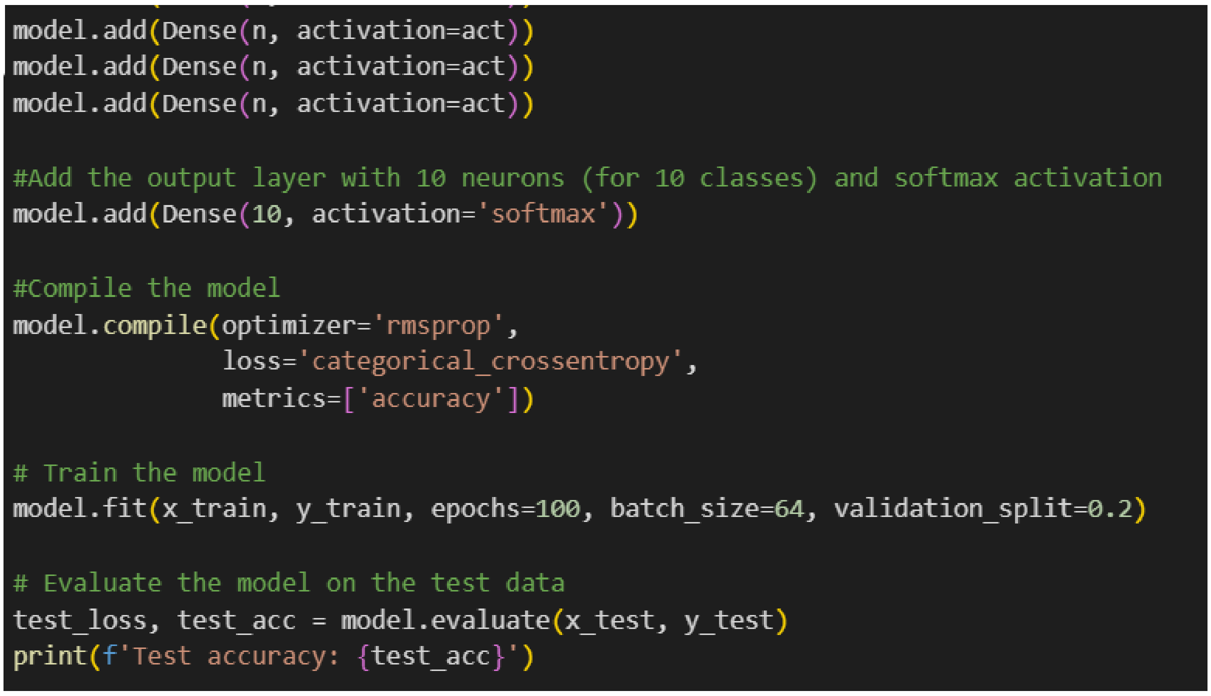
ICP5 Assignment report

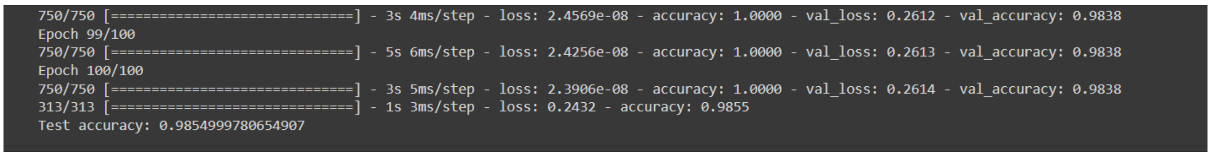


```
#importing
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

#Load the MINST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

#Preprocess the data: normalize images and one-hot encode labels
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)

#Build Model
model = Sequential()

#Flatten the input (28x28 images) into a vector of size 784
model.add(Flatten(input_shape=(28, 28)))

#Add a Five hidden layers with neurons and ReLU activation
act='relu'
n=1024
model.add(Dense(n, activation=act))
model.add(Dense(n, activation=act))
```



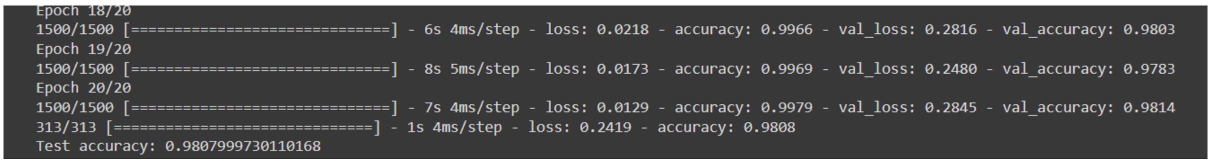
```python
model.add(Dense(n, activation=act))
model.add(Dense(n, activation=act))
model.add(Dense(n, activation=act))

#Add the output layer with 10 neurons (for 10 classes) and softmax activation
model.add(Dense(10, activation='softmax'))

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

# Train the model
model.fit(x_train, y_train, epochs=100, batch_size=64, validation_split=0.2)

# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')
```

Output :-

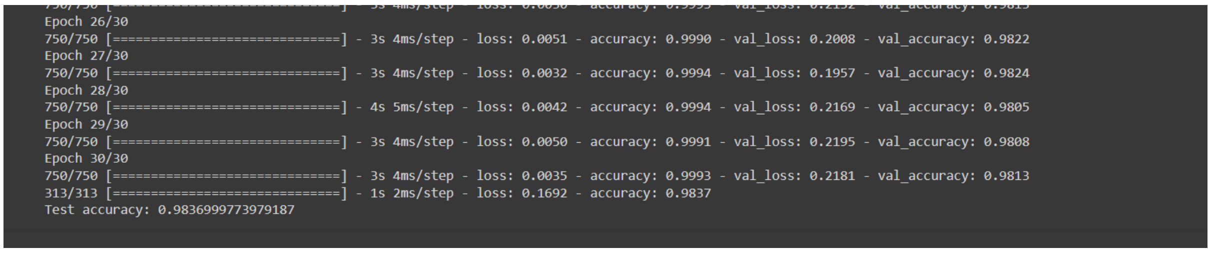


```
750/750 [==============================] - 3s 4ms/step - loss: 2.4569e-08 - accuracy: 1.0000 - val_loss: 0.2612 - val_accuracy: 0.9838
Epoch 99/100
750/750 [==============================] - 5s 6ms/step - loss: 2.4256e-08 - accuracy: 1.0000 - val_loss: 0.2613 - val_accuracy: 0.9838
Epoch 100/100
750/750 [==============================] - 3s 5ms/step - loss: 2.3906e-08 - accuracy: 1.0000 - val_loss: 0.2614 - val_accuracy: 0.9838
313/313 [==============================] - 1s 3ms/step - loss: 0.2432 - accuracy: 0.9855
Test accuracy: 0.9854999780654907
```

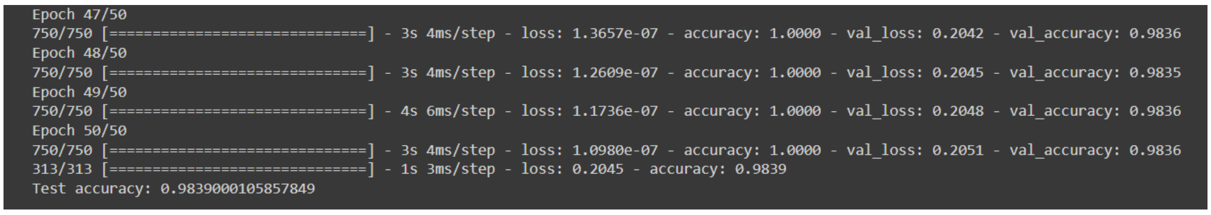
This is closest to 99%



```
Epoch 18/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0218 - accuracy: 0.9966 - val_loss: 0.2816 - val_accuracy: 0.9803
Epoch 19/20
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0173 - accuracy: 0.9969 - val_loss: 0.2480 - val_accuracy: 0.9783
Epoch 20/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0129 - accuracy: 0.9979 - val_loss: 0.2845 - val_accuracy: 0.9814
313/313 [==============================] - 1s 4ms/step - loss: 0.2419 - accuracy: 0.9808
Test accuracy: 0.9807999730110168
```



```
Epoch 26/30
750/750 [==============================] - 3s 4ms/step - loss: 0.0051 - accuracy: 0.9990 - val_loss: 0.2008 - val_accuracy: 0.9822
Epoch 27/30
750/750 [==============================] - 3s 4ms/step - loss: 0.0032 - accuracy: 0.9994 - val_loss: 0.1957 - val_accuracy: 0.9824
Epoch 28/30
750/750 [==============================] - 4s 5ms/step - loss: 0.0042 - accuracy: 0.9994 - val_loss: 0.2169 - val_accuracy: 0.9805
Epoch 29/30
750/750 [==============================] - 3s 4ms/step - loss: 0.0050 - accuracy: 0.9991 - val_loss: 0.2195 - val_accuracy: 0.9808
Epoch 30/30
750/750 [==============================] - 3s 4ms/step - loss: 0.0035 - accuracy: 0.9993 - val_loss: 0.2181 - val_accuracy: 0.9813
313/313 [==============================] - 1s 2ms/step - loss: 0.1692 - accuracy: 0.9837
Test accuracy: 0.9836999773979187
```



```
Epoch 47/50
750/750 [==============================] - 3s 4ms/step - loss: 1.3657e-07 - accuracy: 1.0000 - val_loss: 0.2042 - val_accuracy: 0.9836
Epoch 48/50
750/750 [==============================] - 3s 4ms/step - loss: 1.2609e-07 - accuracy: 1.0000 - val_loss: 0.2045 - val_accuracy: 0.9835
Epoch 49/50
750/750 [==============================] - 4s 6ms/step - loss: 1.1736e-07 - accuracy: 1.0000 - val_loss: 0.2048 - val_accuracy: 0.9836
Epoch 50/50
750/750 [==============================] - 3s 4ms/step - loss: 1.0980e-07 - accuracy: 1.0000 - val_loss: 0.2051 - val_accuracy: 0.9836
313/313 [==============================] - 1s 3ms/step - loss: 0.2045 - accuracy: 0.9839
Test accuracy: 0.9839000105857849
```

- When optimizer is sgd or adam, they have less accuracy compared to rmsprop
- When we increase the neurons, the accuracy will increase
- Accuracy increases when we increase more epochs

YouTube Video Link:- https://www.youtube.com/watch?v=Tun2vWb4Rlo

Github link:- https://github.com/Ksahitha/BDA.git