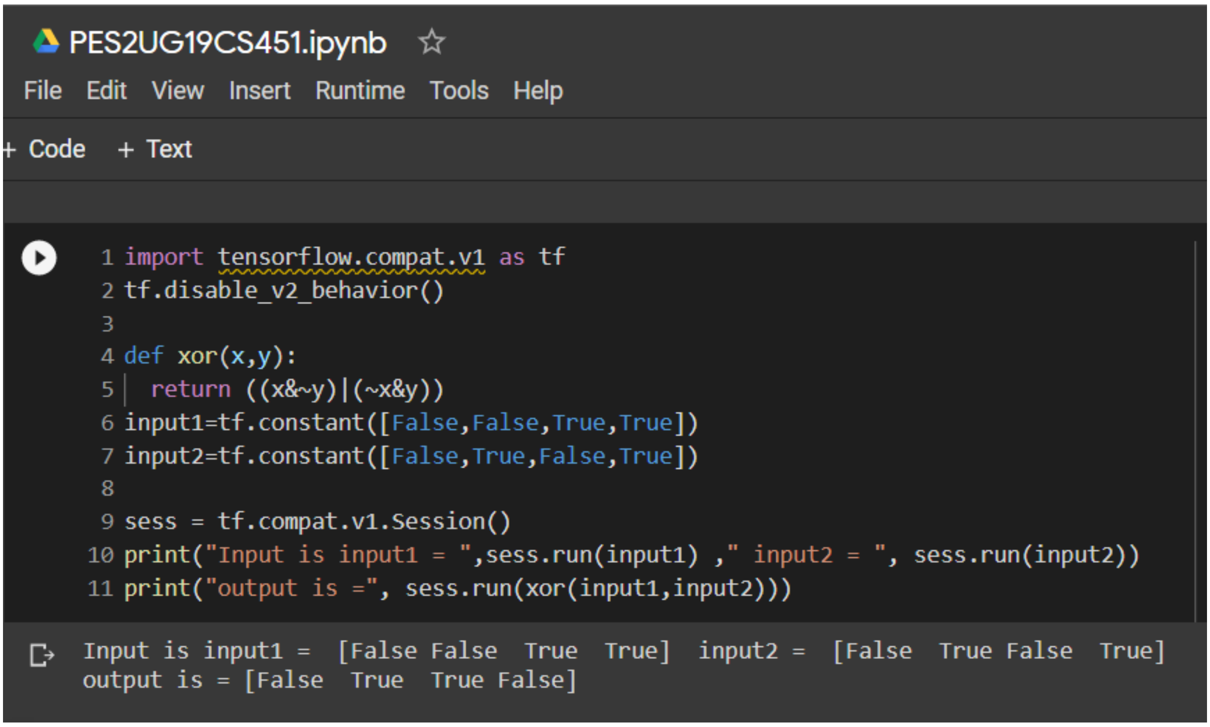
Name: Venkata Krishnarjun Vuppala **Semester**: 6
SRN: PES2UG19CS451 **Section**: G
Subject: Topics in Deep Learning

Assignment 1

Lab programs

1. **Implementation of XOR function using basic gates.**

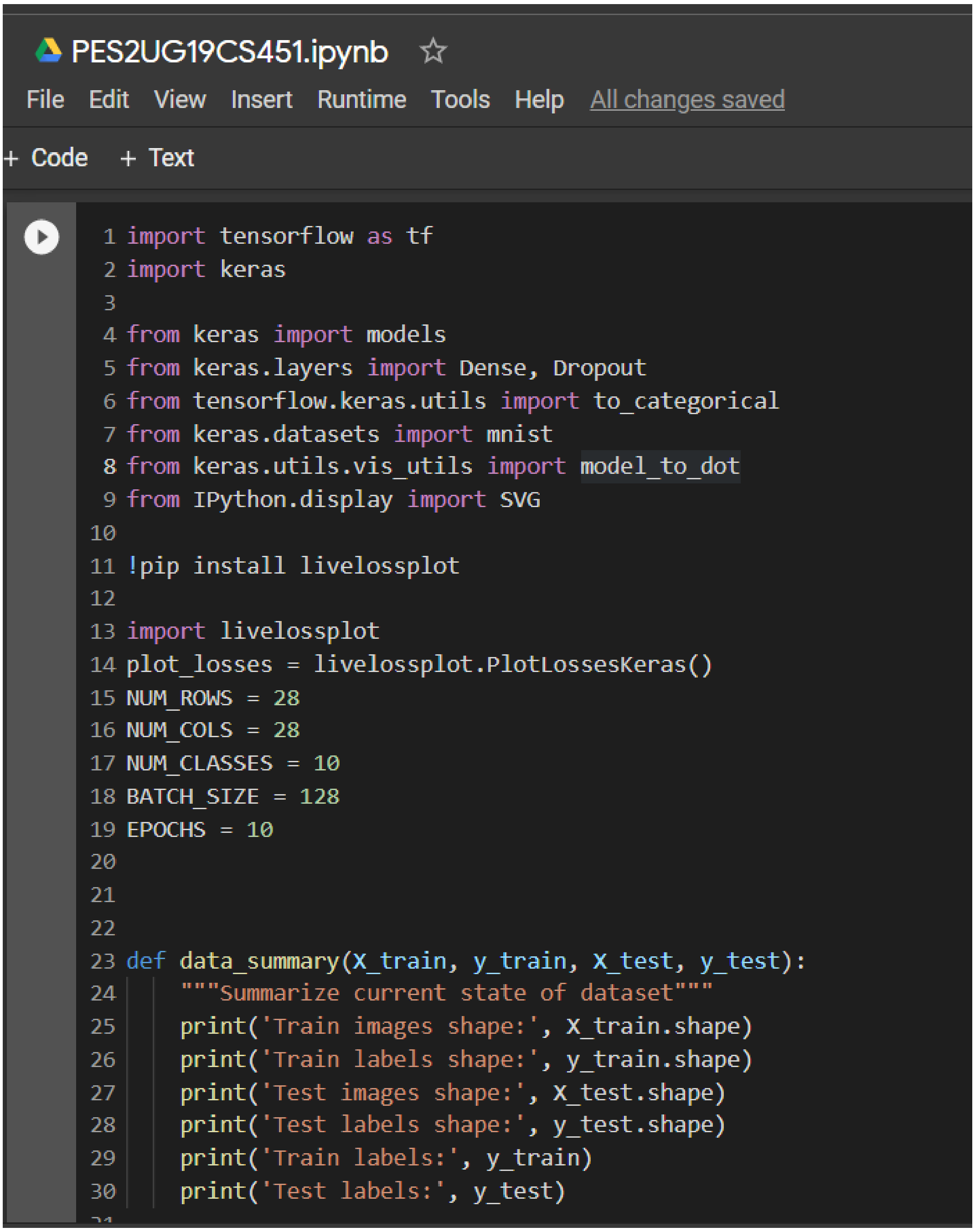


```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

def xor(x,y):
  return ((x&~y)|(~x&y))
input1=tf.constant([False,False,True,True])
input2=tf.constant([False,True,False,True])

sess = tf.compat.v1.Session()
print("Input is input1 = ",sess.run(input1) ," input2 = ", sess.run(input2))
print("output is =", sess.run(xor(input1,input2)))
```

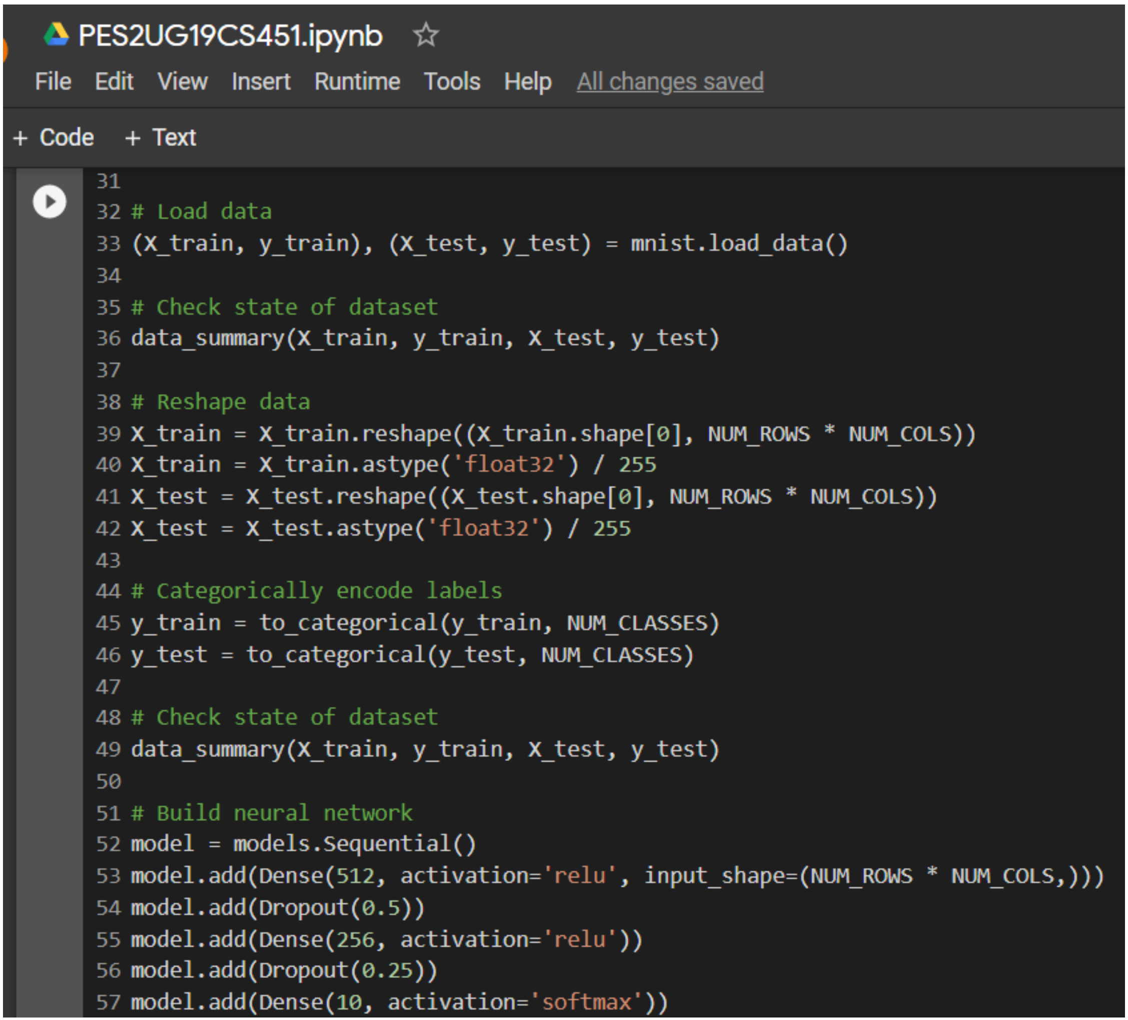
```
Input is input1 =  [False False  True  True]  input2 =  [False  True False  True]
output is = [False  True  True False]
```

2. Implementation of neural network using tensorflow and keras

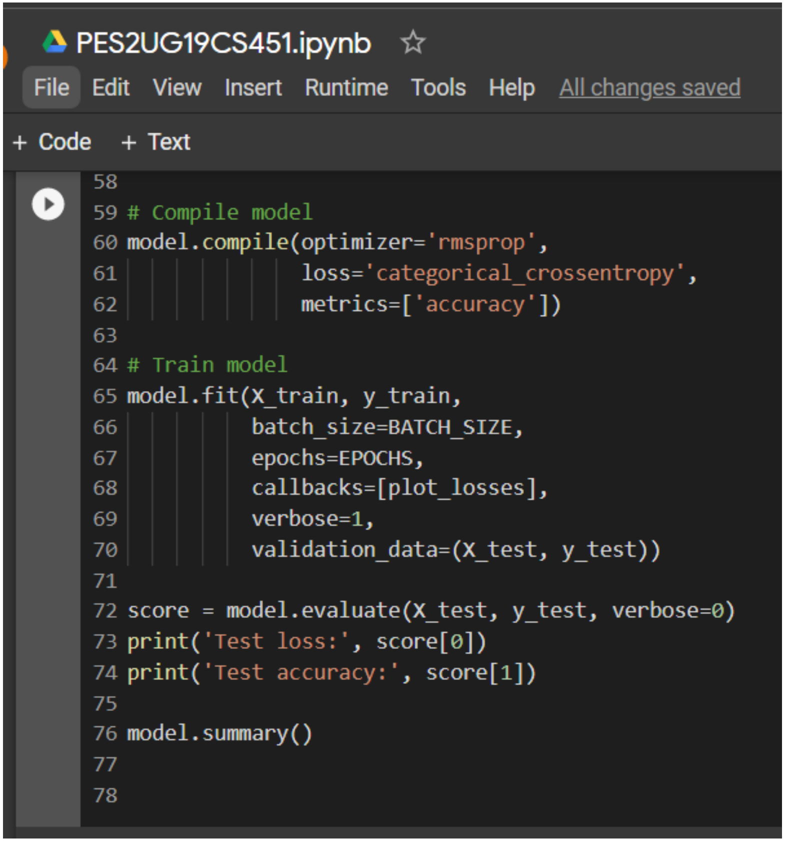
https://keras.io/guides/sequential_model/



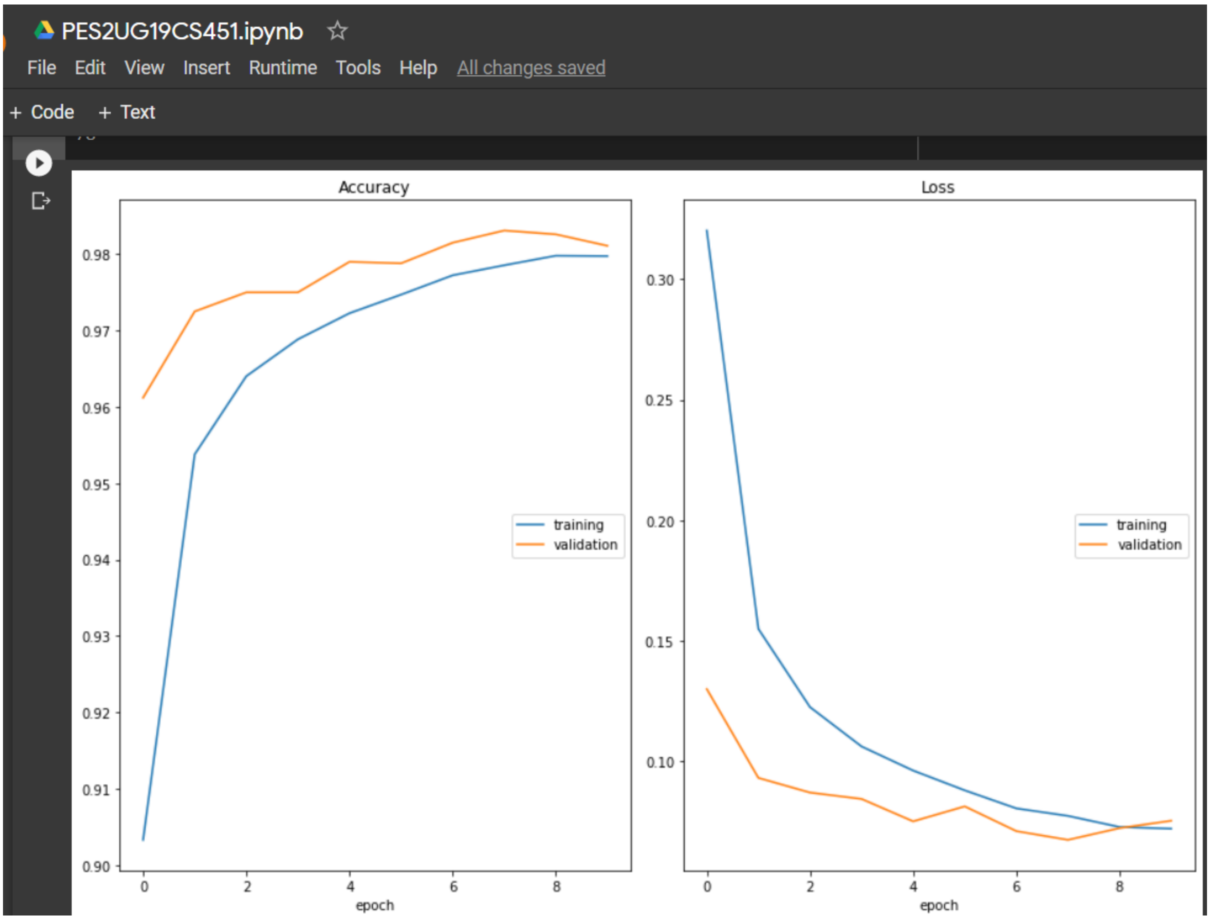
```python
import tensorflow as tf
import keras

from keras import models
from keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
from keras.datasets import mnist
from keras.utils.vis_utils import model_to_dot
from IPython.display import SVG

!pip install livelossplot

import livelossplot
plot_losses = livelossplot.PlotLossesKeras()
NUM_ROWS = 28
NUM_COLS = 28
NUM_CLASSES = 10
BATCH_SIZE = 128
EPOCHS = 10



def data_summary(X_train, y_train, X_test, y_test):
    """Summarize current state of dataset"""
    print('Train images shape:', X_train.shape)
    print('Train labels shape:', y_train.shape)
    print('Test images shape:', X_test.shape)
    print('Test labels shape:', y_test.shape)
    print('Train labels:', y_train)
    print('Test labels:', y_test)
```

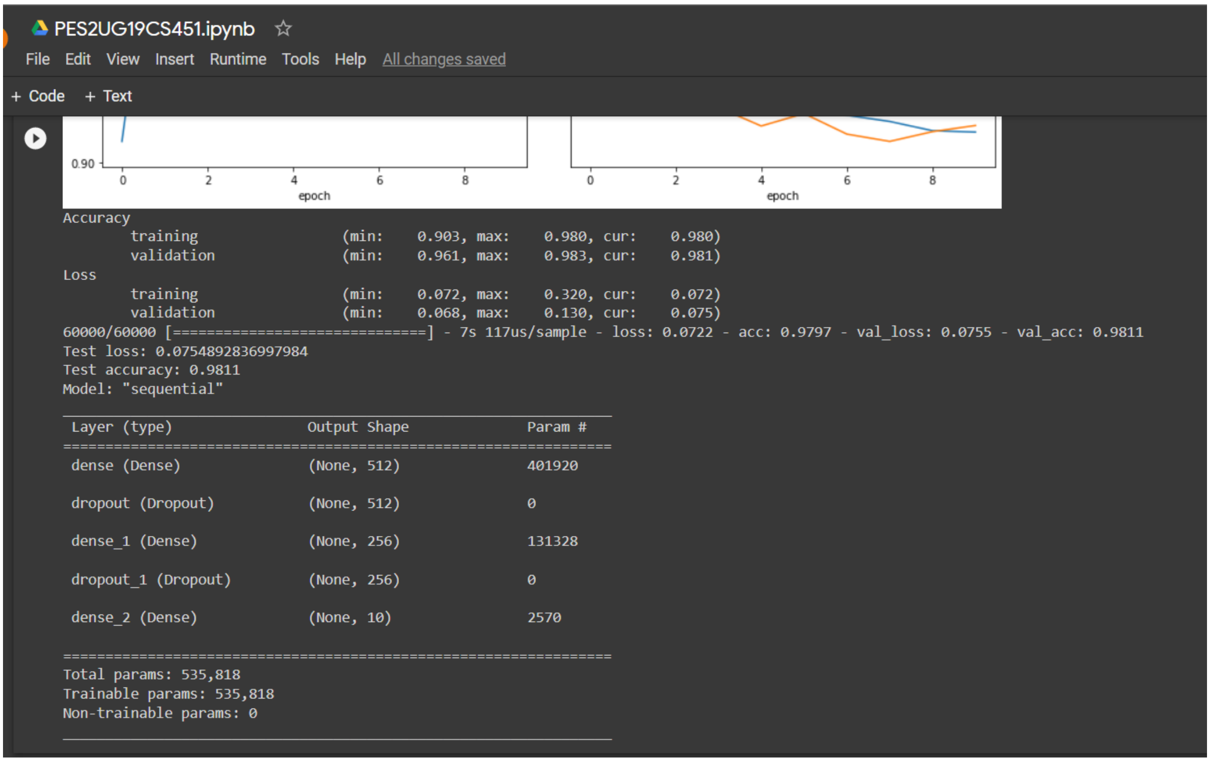


```

# Load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Check state of dataset
data_summary(X_train, y_train, X_test, y_test)

# Reshape data
X_train = X_train.reshape((X_train.shape[0], NUM_ROWS * NUM_COLS))
X_train = X_train.astype('float32') / 255
X_test = X_test.reshape((X_test.shape[0], NUM_ROWS * NUM_COLS))
X_test = X_test.astype('float32') / 255

# Categorically encode labels
y_train = to_categorical(y_train, NUM_CLASSES)
y_test = to_categorical(y_test, NUM_CLASSES)

# Check state of dataset
data_summary(X_train, y_train, X_test, y_test)

# Build neural network
model = models.Sequential()
model.add(Dense(512, activation='relu', input_shape=(NUM_ROWS * NUM_COLS,)))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10, activation='softmax'))
```



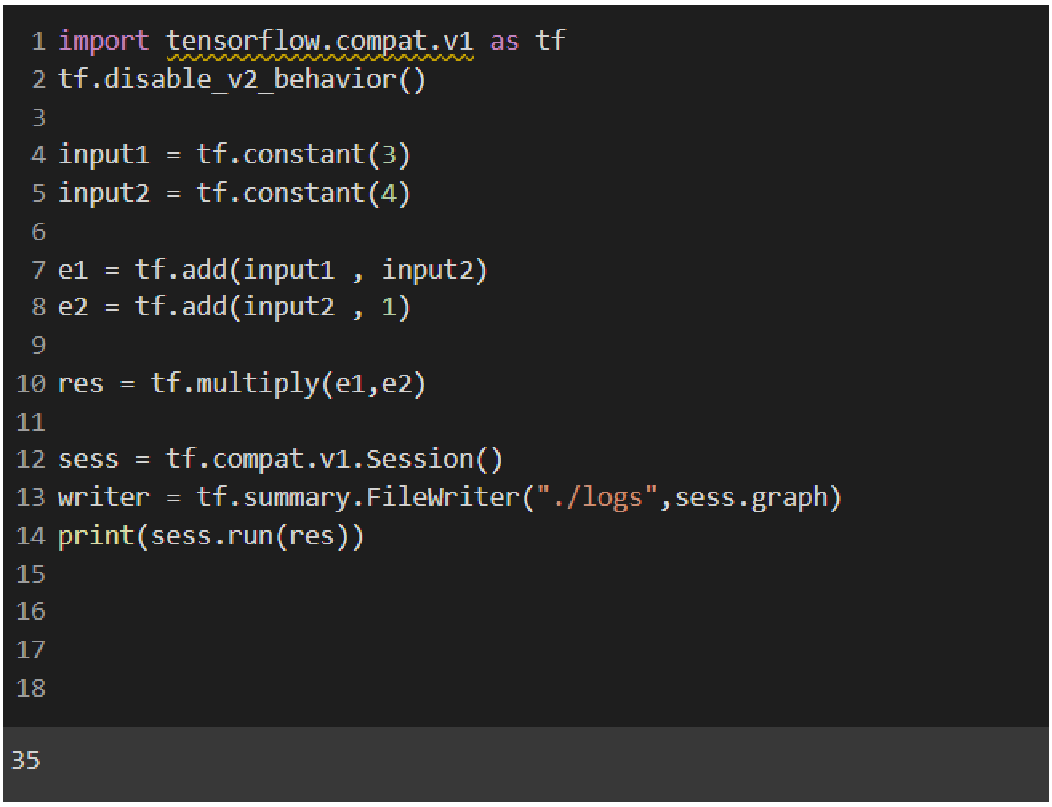
```

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

# Train model
model.fit(X_train, y_train,
          batch_size=BATCH_SIZE,
          epochs=EPOCHS,
          callbacks=[plot_losses],
          verbose=1,
          validation_data=(X_test, y_test))

score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

model.summary()


```





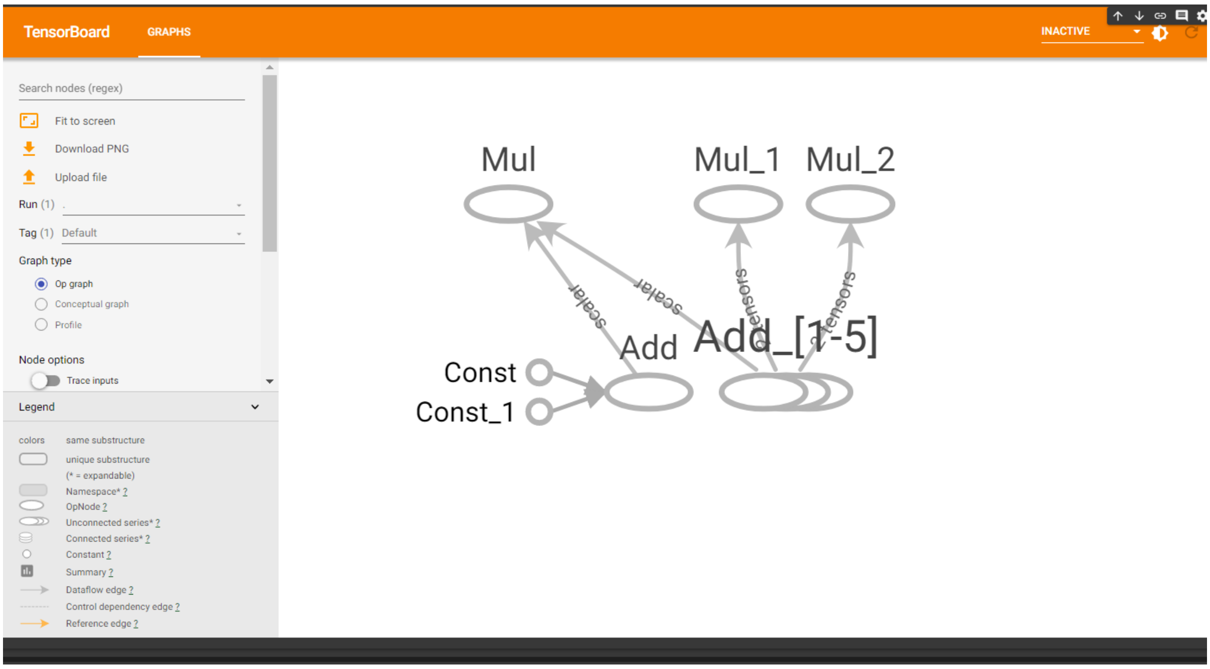
```
Accuracy
        training                 (min:    0.903, max:    0.980, cur:    0.980)
        validation               (min:    0.961, max:    0.983, cur:    0.981)
Loss
        training                 (min:    0.072, max:    0.320, cur:    0.072)
        validation               (min:    0.068, max:    0.130, cur:    0.075)
60000/60000 [==============================] - 7s 117us/sample - loss: 0.0722 - acc: 0.9797 - val_loss: 0.0755 - val_acc: 0.9811
Test loss: 0.07548928369979084
Test accuracy: 0.9811
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 10)                2570

=================================================================
Total params: 535,818
Trainable params: 535,818
Non-trainable params: 0
_________________________________________________________________
```

3. Implementation of computational graphs using tensorflow for simple expressions like e=(a+b)*(b+1).



```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

input1 = tf.constant(3)
input2 = tf.constant(4)

e1 = tf.add(input1 , input2)
e2 = tf.add(input2 , 1)

res = tf.multiply(e1,e2)

sess = tf.compat.v1.Session()
writer = tf.summary.FileWriter("./logs",sess.graph)
print(sess.run(res))
```

```
35
```



TensorBoard
GRAPHS
INACTIVE
Search nodes (regex)
Fit to screen
Download PNG
Upload file
Run (1)
Tag (1) Default
Graph type
● Op graph
○ Conceptual graph
○ Profile
Node options
Trace inputs
Legend
colors
same substructure
unique substructure
(* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
Mul
Mul_1
Mul_2
scalar
scalar
tensors
tensors
Add
Add_[1-5]
Const
Const_1