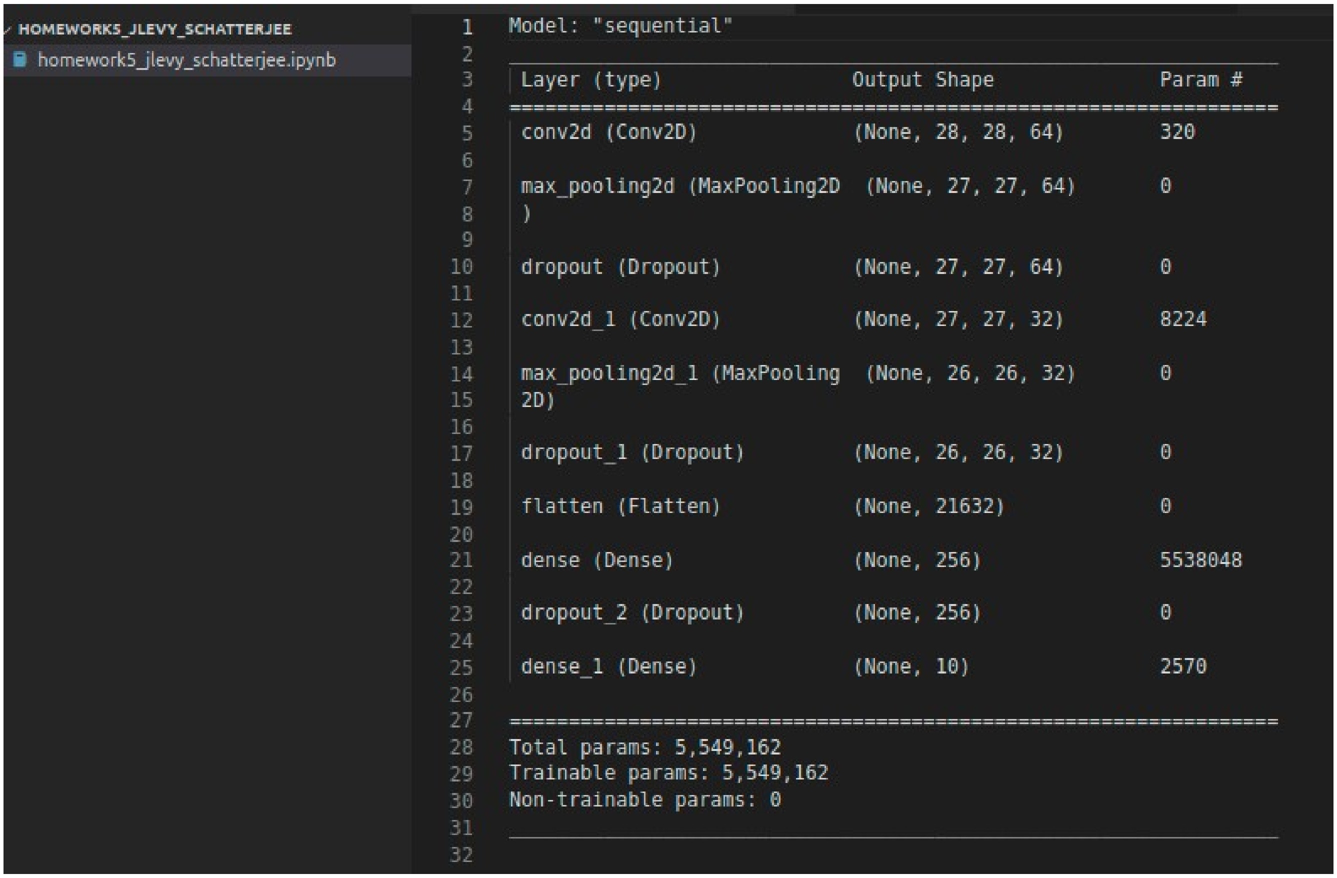
CS/DS541 Deep Learning

Joshua Levy
Sreejani Chatterjee

Part1:

Trained for 10 Epochs, and batch size of 64


HOMEWORK5_JLEVY_SCHATTERJEE
homework5_jlevy_schatterjee.ipynb

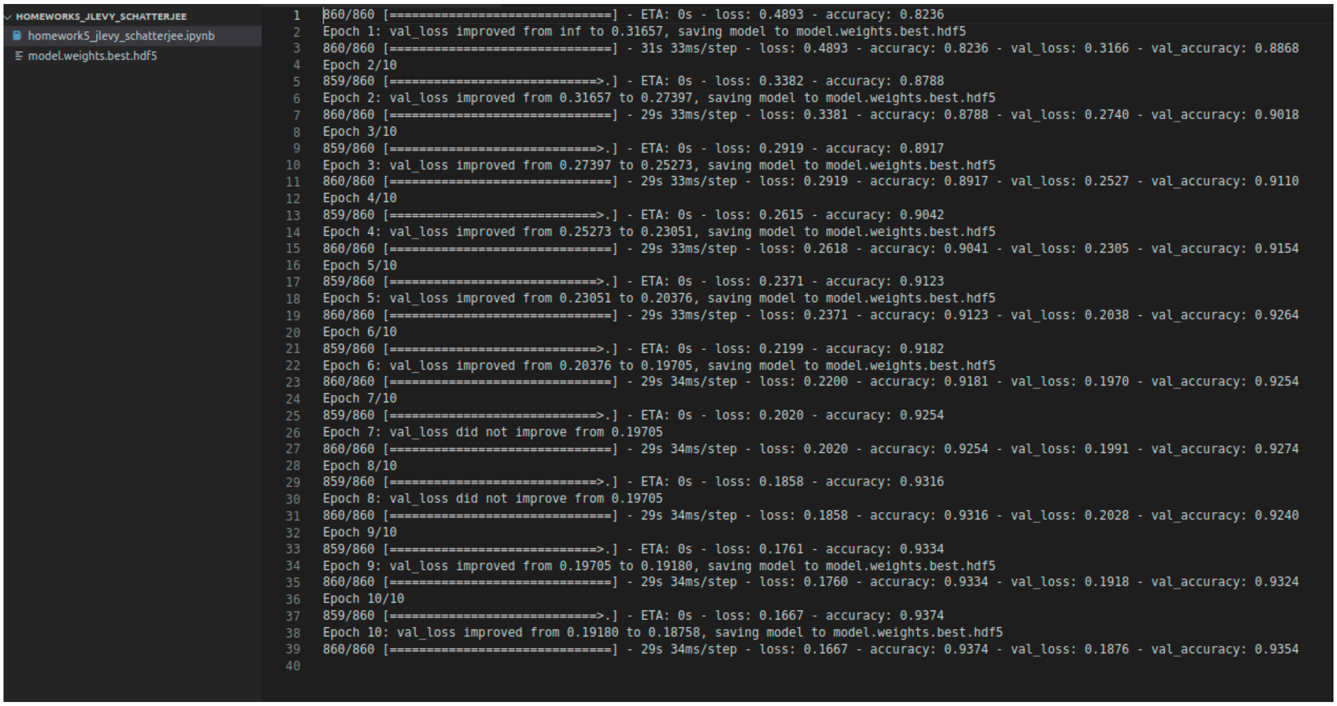
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        320

 max_pooling2d (MaxPooling2D  (None, 27, 27, 64)       0
 )

 dropout (Dropout)           (None, 27, 27, 64)        0

 conv2d_1 (Conv2D)           (None, 27, 27, 32)        8224

 max_pooling2d_1 (MaxPooling  (None, 26, 26, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 26, 26, 32)        0

 flatten (Flatten)           (None, 21632)             0

 dense (Dense)               (None, 256)               5538048

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 5,549,162
Trainable params: 5,549,162
Non-trainable params: 0
_________________________________________________________________
```

<u>Architecture used:</u>

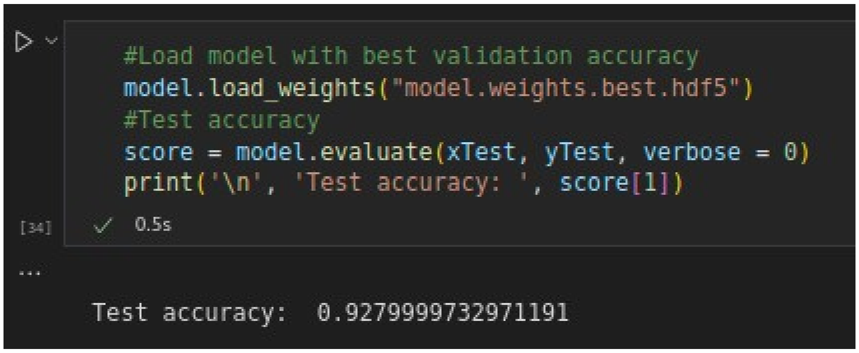
Convolution layer with 64 filters, each 2x2, stride of 1, without padding activation relu
Max pool with a pooling width of 2x2
Dropout of 0.3, Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting
Convolution layer with 32 filters, each 2x2, stride of 1, without padding activation relu
Max pool with a pooling width of 2x2
Dropout of 0.3,
Flattened all feature maps into one long vector.
Fully-connected layer to map into a 256-dimensional vector
Relu
Dropout of 0.5
Fully-connected layer to map into a 10-dimensional vector.
Softmax.

<u>Results:</u>

Loss and accuracy values over the epochs


HOMEWORKS_JLEVY_SCHATTERJEE
homework5_jlevy_schatterjee.ipynb
model.weights.best.hdf5

```
860/860 [==============================] - ETA: 0s - loss: 0.4893 - accuracy: 0.8236
Epoch 1: val_loss improved from inf to 0.31657, saving model to model.weights.best.hdf5
860/860 [==============================] - 31s 33ms/step - loss: 0.4893 - accuracy: 0.8236 - val_loss: 0.3166 - val_accuracy: 0.8868
Epoch 2/10
859/860 [============================>.] - ETA: 0s - loss: 0.3382 - accuracy: 0.8788
Epoch 2: val_loss improved from 0.31657 to 0.27397, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 33ms/step - loss: 0.3381 - accuracy: 0.8788 - val_loss: 0.2740 - val_accuracy: 0.9018
Epoch 3/10
859/860 [============================>.] - ETA: 0s - loss: 0.2919 - accuracy: 0.8917
Epoch 3: val_loss improved from 0.27397 to 0.25273, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 33ms/step - loss: 0.2919 - accuracy: 0.8917 - val_loss: 0.2527 - val_accuracy: 0.9110
Epoch 4/10
859/860 [============================>.] - ETA: 0s - loss: 0.2615 - accuracy: 0.9042
Epoch 4: val_loss improved from 0.25273 to 0.23051, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 33ms/step - loss: 0.2618 - accuracy: 0.9041 - val_loss: 0.2305 - val_accuracy: 0.9154
Epoch 5/10
859/860 [============================>.] - ETA: 0s - loss: 0.2371 - accuracy: 0.9123
Epoch 5: val_loss improved from 0.23051 to 0.20376, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 33ms/step - loss: 0.2371 - accuracy: 0.9123 - val_loss: 0.2038 - val_accuracy: 0.9264
Epoch 6/10
859/860 [============================>.] - ETA: 0s - loss: 0.2199 - accuracy: 0.9182
Epoch 6: val_loss improved from 0.20376 to 0.19705, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 34ms/step - loss: 0.2200 - accuracy: 0.9181 - val_loss: 0.1970 - val_accuracy: 0.9254
Epoch 7/10
859/860 [============================>.] - ETA: 0s - loss: 0.2020 - accuracy: 0.9254
Epoch 7: val_loss did not improve from 0.19705
860/860 [==============================] - 29s 34ms/step - loss: 0.2020 - accuracy: 0.9254 - val_loss: 0.1991 - val_accuracy: 0.9274
Epoch 8/10
859/860 [============================>.] - ETA: 0s - loss: 0.1858 - accuracy: 0.9316
Epoch 8: val_loss did not improve from 0.19705
860/860 [==============================] - 29s 34ms/step - loss: 0.1858 - accuracy: 0.9316 - val_loss: 0.2028 - val_accuracy: 0.9240
Epoch 9/10
859/860 [============================>.] - ETA: 0s - loss: 0.1761 - accuracy: 0.9334
Epoch 9: val_loss improved from 0.19705 to 0.19180, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 34ms/step - loss: 0.1760 - accuracy: 0.9334 - val_loss: 0.1918 - val_accuracy: 0.9324
Epoch 10/10
859/860 [============================>.] - ETA: 0s - loss: 0.1667 - accuracy: 0.9374
Epoch 10: val_loss improved from 0.19180 to 0.18758, saving model to model.weights.best.hdf5
860/860 [==============================] - 29s 34ms/step - loss: 0.1667 - accuracy: 0.9374 - val_loss: 0.1876 - val_accuracy: 0.9354
```



```
#Load model with best validation accuracy
model.load_weights("model.weights.best.hdf5")
#Test accuracy
score = model.evaluate(xTest, yTest, verbose = 0)
print('\n', 'Test accuracy: ', score[1])
```

```
Test accuracy:  0.9279999732971191
```

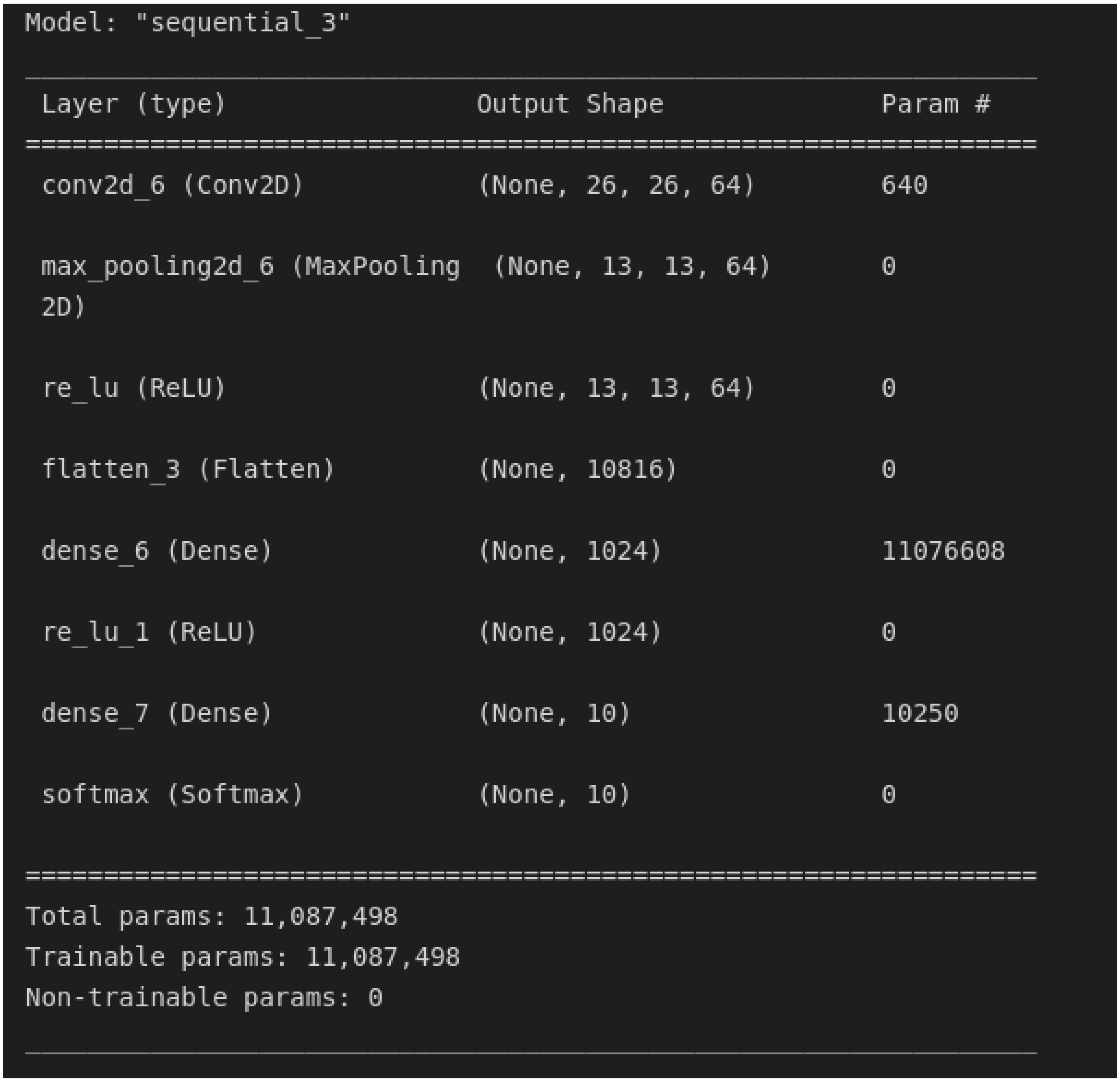
Test Accuracy Obtained: 92.79%

Part2:

Trained for 5 Epochs, and batch size of 64

Architecture used:

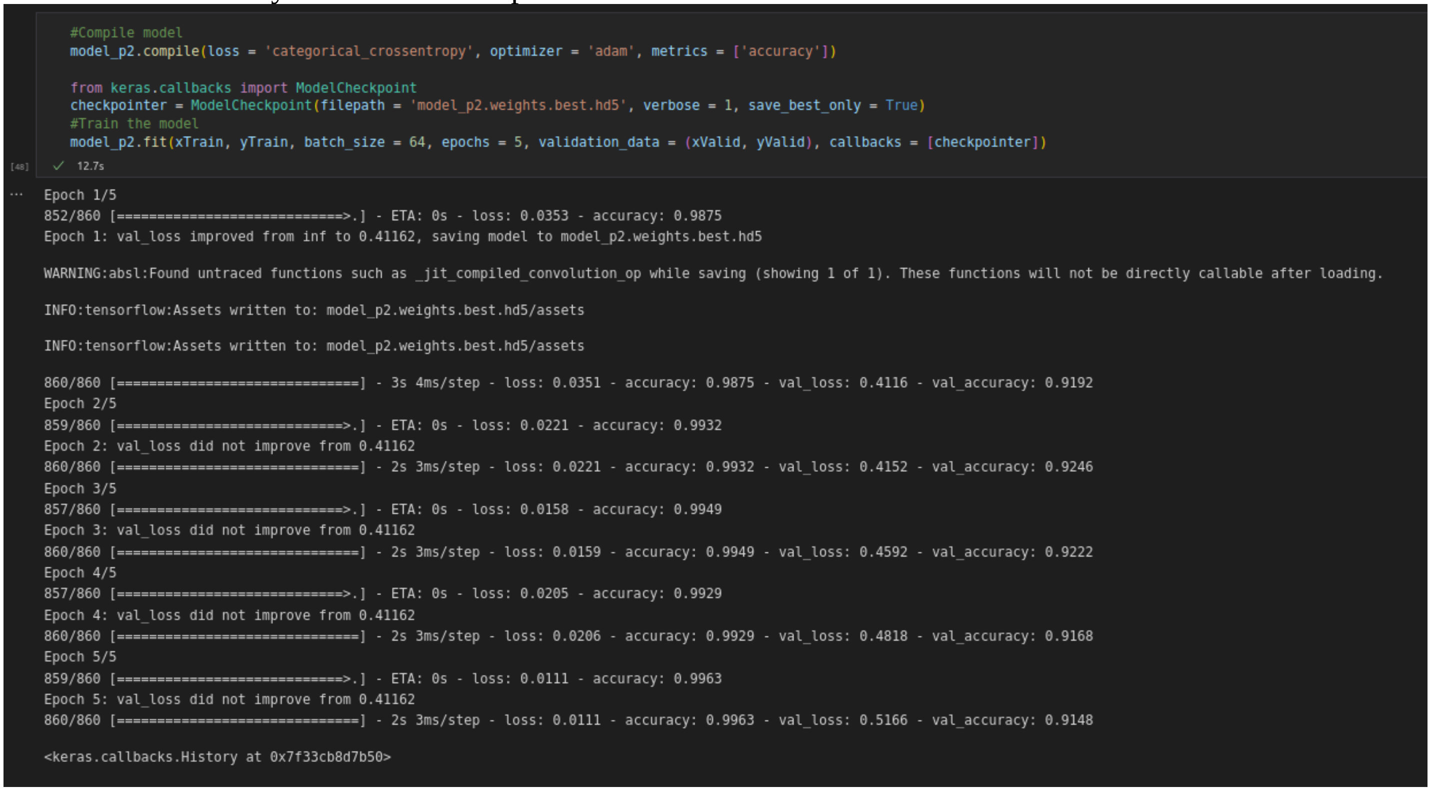
Convolution layer with 64 filters, each 3x3, stride of 1 no padding.
Max pool with a pooling width of 2x2, stride of 2, no padding.
ReLU.
Flatten the 64 feature maps into one long vector.
Fully-connected layer to map into a 1024-dimensional vector.
ReLU.
Fully-connected layer to map into a 10-dimensional vector.
Softmax.



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 64)        640

 max_pooling2d_6 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 re_lu (ReLU)                (None, 13, 13, 64)        0

 flatten_3 (Flatten)         (None, 10816)             0

 dense_6 (Dense)             (None, 1024)              11076608

 re_lu_1 (ReLU)              (None, 1024)              0

 dense_7 (Dense)             (None, 10)                10250

 softmax (Softmax)           (None, 10)                0

=================================================================
Total params: 11,087,498
Trainable params: 11,087,498
Non-trainable params: 0
_________________________________________________________________
```

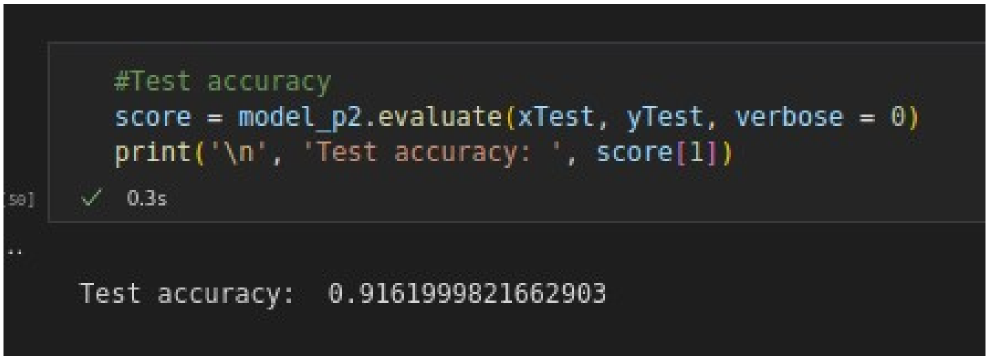
Results:

Loss and accuracy values over the epochs



```
#Compile model
model_p2.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

from keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath = 'model_p2.weights.best.hd5', verbose = 1, save_best_only = True)
#Train the model
model_p2.fit(xTrain, yTrain, batch_size = 64, epochs = 5, validation_data = (xValid, yValid), callbacks = [checkpointer])
```

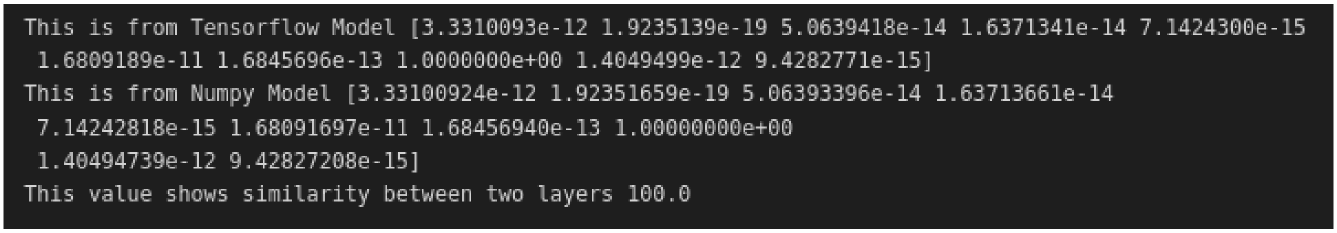
```
Epoch 1/5
852/860 [============================>.] - ETA: 0s - loss: 0.0353 - accuracy: 0.9875
Epoch 1: val_loss improved from inf to 0.41162, saving model to model_p2.weights.best.hd5

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op while saving (showing 1 of 1). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: model_p2.weights.best.hd5/assets

INFO:tensorflow:Assets written to: model_p2.weights.best.hd5/assets

860/860 [==============================] - 3s 4ms/step - loss: 0.0351 - accuracy: 0.9875 - val_loss: 0.4116 - val_accuracy: 0.9192
Epoch 2/5
859/860 [============================>.] - ETA: 0s - loss: 0.0221 - accuracy: 0.9932
Epoch 2: val_loss did not improve from 0.41162
860/860 [==============================] - 2s 3ms/step - loss: 0.0221 - accuracy: 0.9932 - val_loss: 0.4152 - val_accuracy: 0.9246
Epoch 3/5
857/860 [============================>.] - ETA: 0s - loss: 0.0158 - accuracy: 0.9949
Epoch 3: val_loss did not improve from 0.41162
860/860 [==============================] - 2s 3ms/step - loss: 0.0159 - accuracy: 0.9949 - val_loss: 0.4592 - val_accuracy: 0.9222
Epoch 4/5
857/860 [============================>.] - ETA: 0s - loss: 0.0205 - accuracy: 0.9929
Epoch 4: val_loss did not improve from 0.41162
860/860 [==============================] - 2s 3ms/step - loss: 0.0206 - accuracy: 0.9929 - val_loss: 0.4818 - val_accuracy: 0.9168
Epoch 5/5
859/860 [============================>.] - ETA: 0s - loss: 0.0111 - accuracy: 0.9963
Epoch 5: val_loss did not improve from 0.41162
860/860 [==============================] - 2s 3ms/step - loss: 0.0111 - accuracy: 0.9963 - val_loss: 0.5166 - val_accuracy: 0.9148

<keras.callbacks.History at 0x7f33cb8d7b50>
```



```
#Test accuracy
score = model_p2.evaluate(xTest, yTest, verbose = 0)
print('\n', 'Test accuracy: ', score[1])
```

```
Test accuracy:  0.9161999821662903
```

Test Accuracy Obtained: 91.62%

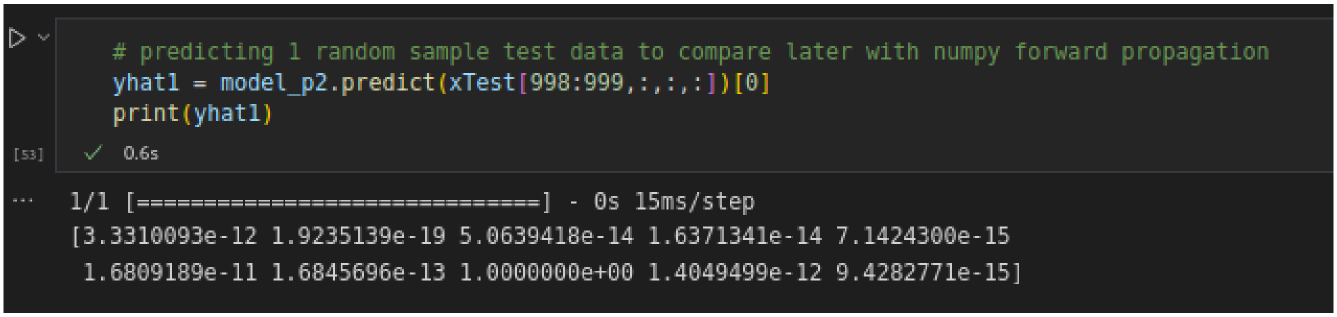


```
This is from Tensorflow Model [3.3310093e-12 1.9235139e-19 5.0639418e-14 1.6371341e-14 7.1424300e-15
 1.6809189e-11 1.6845696e-13 1.0000000e+00 1.4049499e-12 9.4282771e-15]
This is from Numpy Model [3.33100924e-12 1.92351659e-19 5.06393396e-14 1.63713661e-14
 7.14242818e-15 1.68091697e-11 1.68456940e-13 1.00000000e+00
 1.40494739e-12 9.42827208e-15]
This value shows similarity between two layers 100.0
```

Comparison of results from Tensorflow and Numpy code:

We verify the accuracy by predicting a random test data for both the Tensorflow and Numpy model.

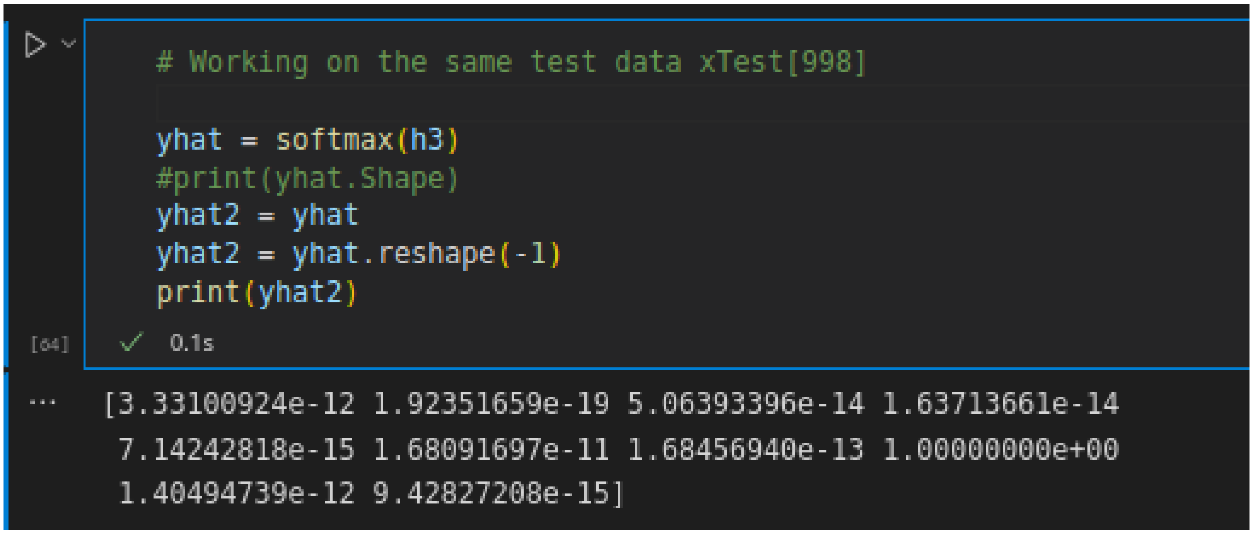
<u>Tensorflow:</u>



```
# predicting 1 random sample test data to compare later with numpy forward propagation
yhat1 = model_p2.predict(xTest[998:999,:,:,:])[0]
print(yhat1)
```

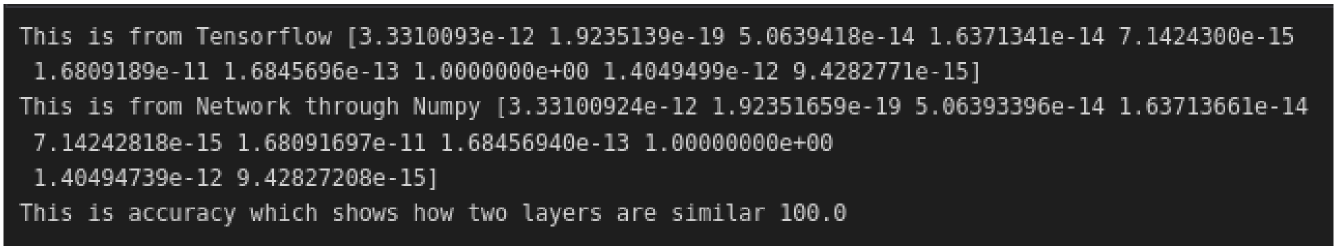
```
1/1 [==============================] - 0s 15ms/step
[3.3310093e-12 1.9235139e-19 5.0639418e-14 1.6371341e-14 7.1424300e-15
 1.6809189e-11 1.6845696e-13 1.0000000e+00 1.4049499e-12 9.4282771e-15]
```

<u>Numpy:</u>



```
# Working on the same test data xTest[998]

yhat = softmax(h3)
#print(yhat.Shape)
yhat2 = yhat
yhat2 = yhat.reshape(-1)
print(yhat2)
```

```
[3.33100924e-12 1.92351659e-19 5.06393396e-14 1.63713661e-14
 7.14242818e-15 1.68091697e-11 1.68456940e-13 1.00000000e+00
 1.40494739e-12 9.42827208e-15]
```

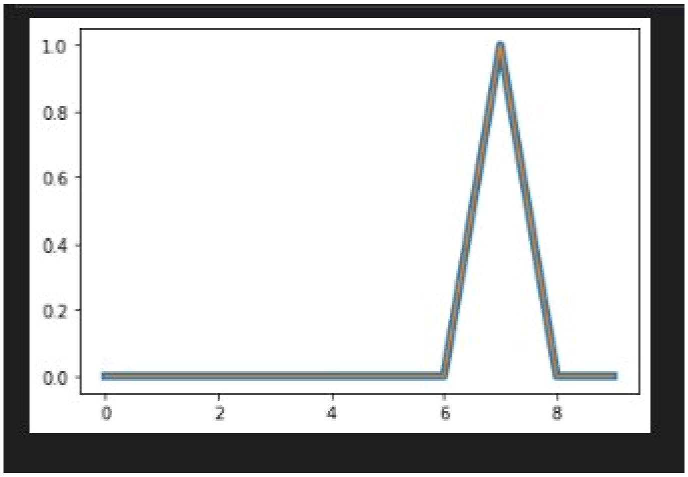


```
This is from Tensorflow [3.3310093e-12 1.9235139e-19 5.0639418e-14 1.6371341e-14 7.1424300e-15
 1.6809189e-11 1.6845696e-13 1.0000000e+00 1.4049499e-12 9.4282771e-15]
This is from Network through Numpy [3.33100924e-12 1.92351659e-19 5.06393396e-14 1.63713661e-14
 7.14242818e-15 1.68091697e-11 1.68456940e-13 1.00000000e+00
 1.40494739e-12 9.42827208e-15]
This is accuracy which shows how two layers are similar 100.0
```

<u>Results:</u>

Both outputs have same values leading to a 100% similarity in results

The following plot shows the same.



1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8