Deep Learning

CISC 867

Project 1

By:

Name : Abdelrahman Magdy El-Hamoly

email: 21amme1@queensu.ca

Name : Kareem Gamal Mahmoud Mohamed

email: 21kgmm@queensu.ca

Name : Ahmed Mohamed Gaber

email: 21amga@queensu.ca

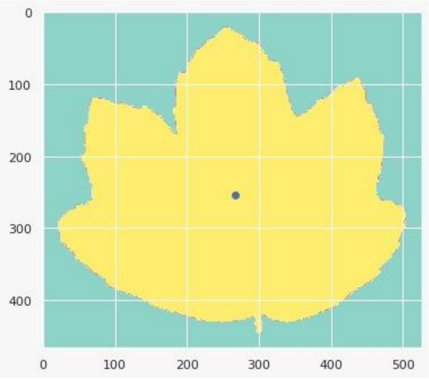
Supervised by\ Dr. Hazem Abbas

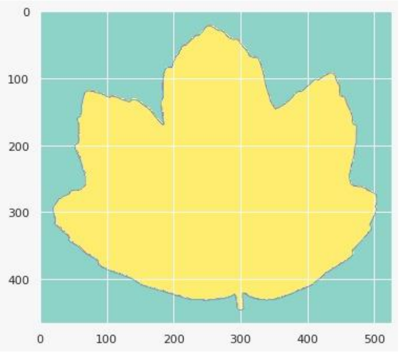
Part 1 :

Description and cleaning:

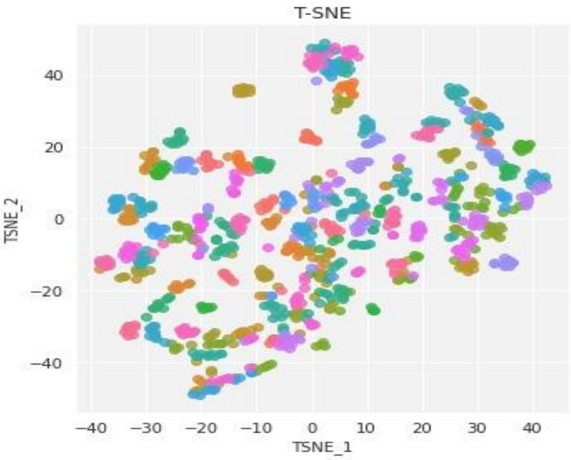
The data is clear ,there are no missing values , there are no outliers, the data is normalized because the range of the values between zero and 1, there is no duplication.

Draw some of the images:





The distribution of the target points:



Part 2 : Simple MLP Model

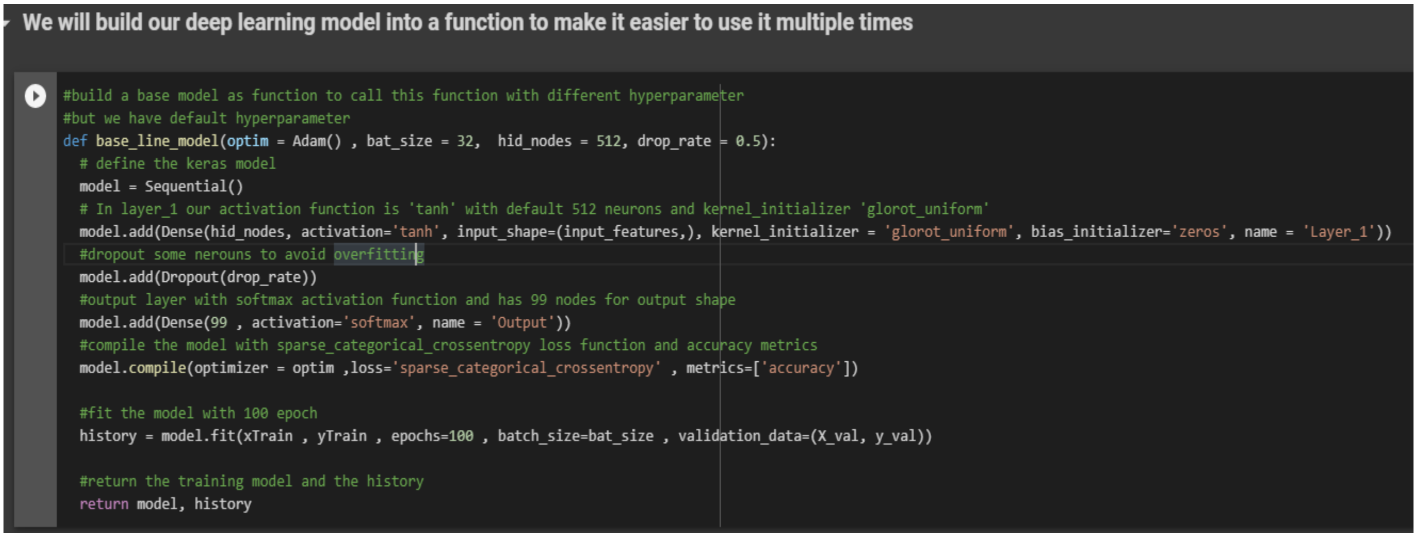
We decided to tune the following hyperparameters:

- Batch Size
- Hidden Nodes Size
- Drop Rate
- Optimizer

In the beginning we had built a function named 'base_line_model' to build several models with different hyperparameters

This function has default hyperparameters which are

[optim = Adam() , bat_size = 32, hid_nodes = 512, drop_rate = 0.5]

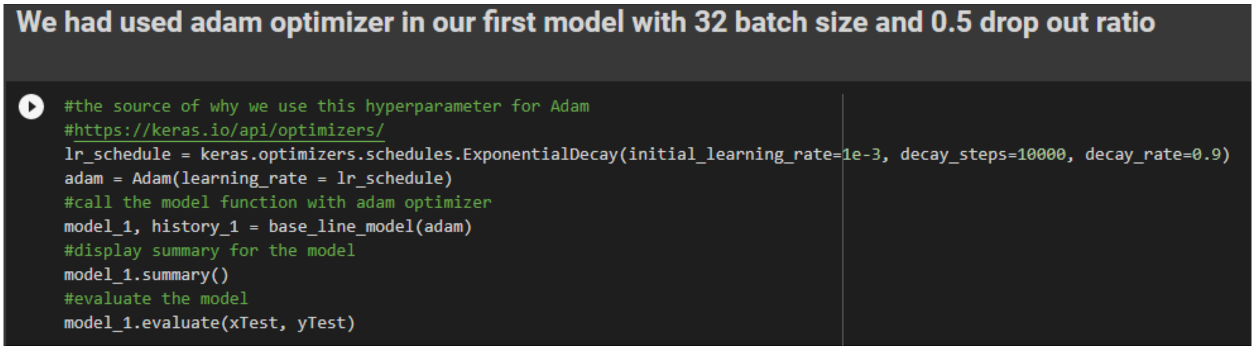


We will build our deep learning model into a function to make it easier to use it multiple times

```
#build a base model as function to call this function with different hyperparameter
#but we have default hyperparameter
def base_line_model(optim = Adam() , bat_size = 32,  hid_nodes = 512, drop_rate = 0.5):
  # define the keras model
  model = Sequential()
  # In layer_1 our activation function is 'tanh' with default 512 neurons and kernel_initializer 'glorot_uniform'
  model.add(Dense(hid_nodes, activation='tanh', input_shape=(input_features,), kernel_initializer = 'glorot_uniform', bias_initializer='zeros', name = 'Layer_1'))
  #dropout some nerouns to avoid overfitting
  model.add(Dropout(drop_rate))
  #output layer with softmax activation function and has 99 nodes for output shape
  model.add(Dense(99 , activation='softmax', name = 'Output'))
  #compile the model with sparse_categorical_crossentropy loss function and accuracy metrics
  model.compile(optimizer = optim ,loss='sparse_categorical_crossentropy' , metrics=['accuracy'])

  #fit the model with 100 epoch
  history = model.fit(xTrain , yTrain , epochs=100 , batch_size=bat_size , validation_data=(X_val, y_val))

  #return the training model and the history
  return model, history
```

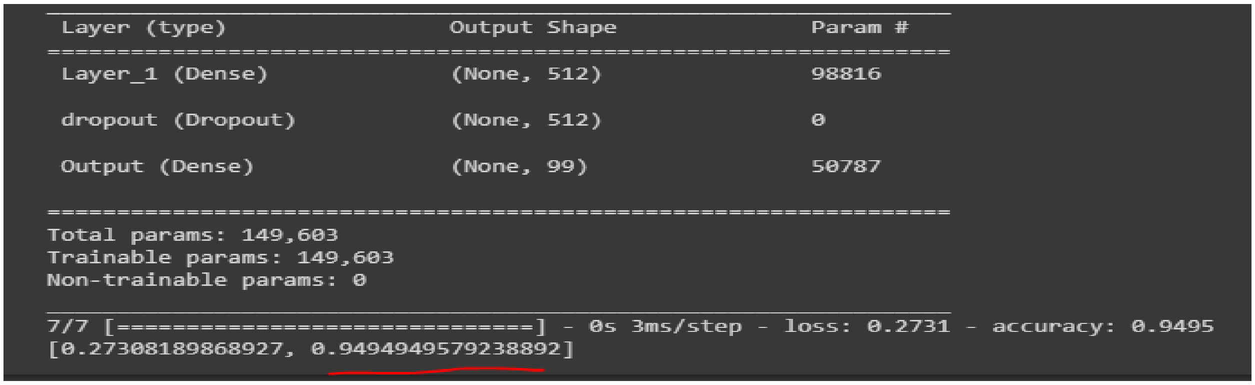
Tune with different optimizers.

Trial 1: Adam optimizer



We had used adam optimizer in our first model with 32 batch size and 0.5 drop out ratio

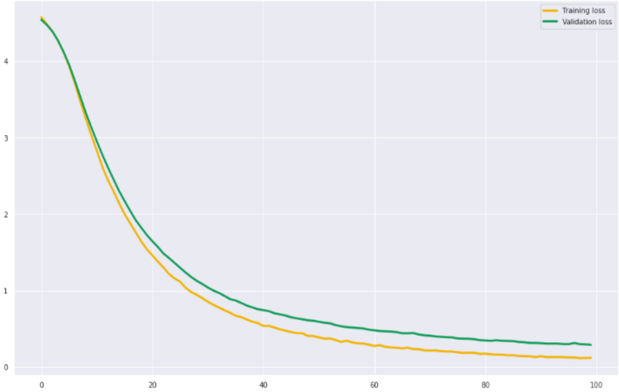
```
#the source of why we use this hyperparameter for Adam
#https://keras.io/api/optimizers/
lr_schedule = keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=1e-3, decay_steps=10000, decay_rate=0.9)
adam = Adam(learning_rate = lr_schedule)
#call the model function with adam optimizer
model_1, history_1 = base_line_model(adam)
#display summary for the model
model_1.summary()
#evaluate the model
model_1.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout (Dropout)           (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 3ms/step - loss: 0.2731 - accuracy: 0.9495
[0.27308189868927, 0.9494949579238892]
```

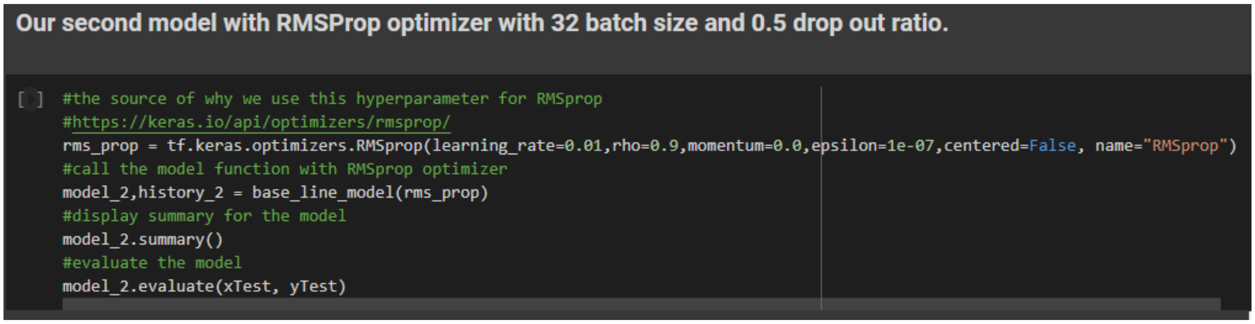
Loss ratio for

validation & training



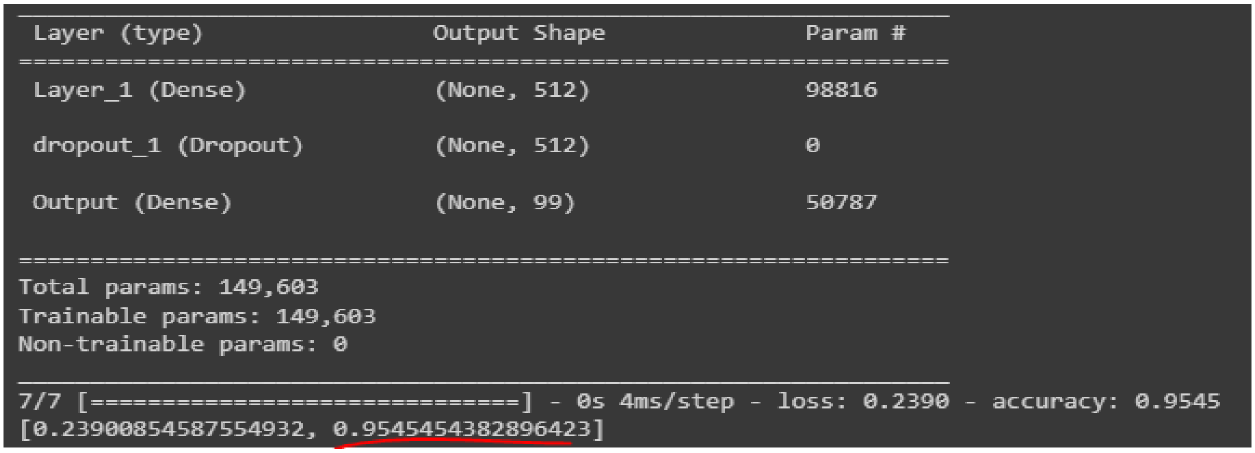
Note : as we can see from this graph we can run only 85 epochs instead of 100 epochs

Trial 2: RMS Prop optimizer



Our second model with RMSProp optimizer with 32 batch size and 0.5 drop out ratio.

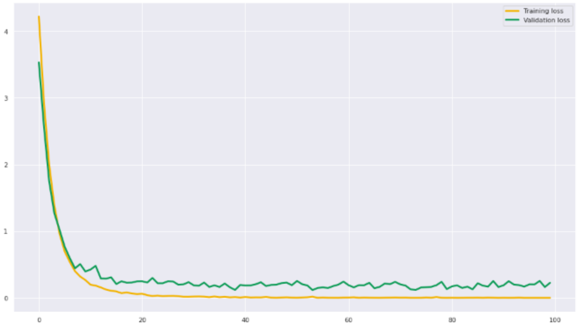
```
[ ] #the source of why we use this hyperparameter for RMSprop
    #https://keras.io/api/optimizers/rmsprop/
    rms_prop = tf.keras.optimizers.RMSprop(learning_rate=0.01,rho=0.9,momentum=0.0,epsilon=1e-07,centered=False, name="RMSprop")
    #call the model function with RMSprop optimizer
    model_2,history_2 = base_line_model(rms_prop)
    #display summary for the model
    model_2.summary()
    #evaluate the model
    model_2.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_1 (Dropout)         (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 4ms/step - loss: 0.2390 - accuracy: 0.9545
[0.23900854587554932, 0.9545454382896423]
```

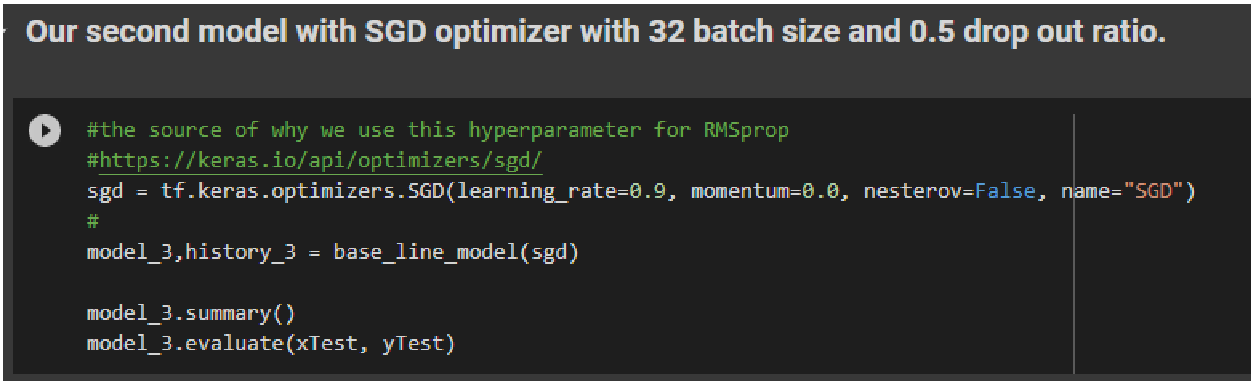
Loss ratio for

validation & training



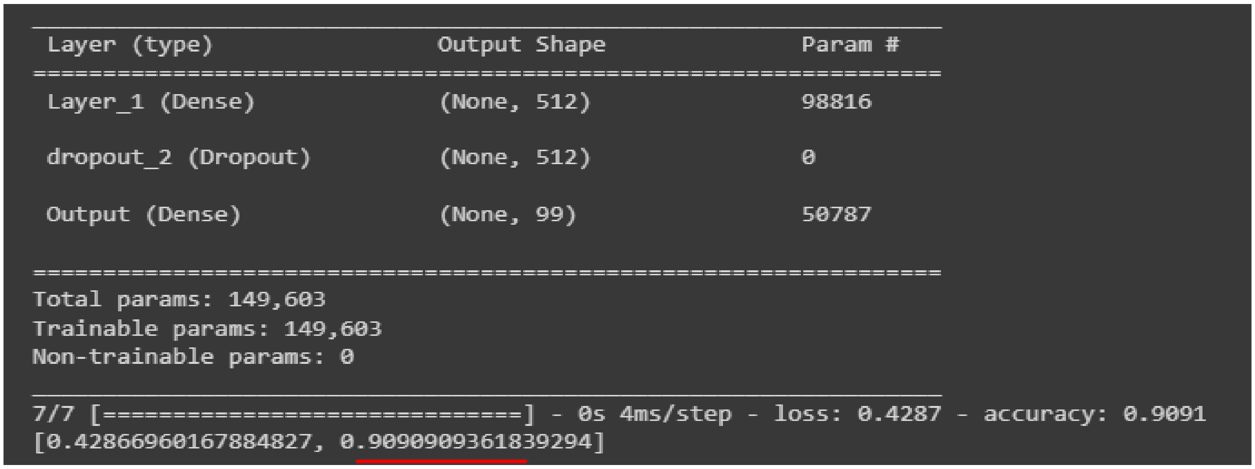
Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 3: SGD optimizer



Our second model with SGD optimizer with 32 batch size and 0.5 drop out ratio.

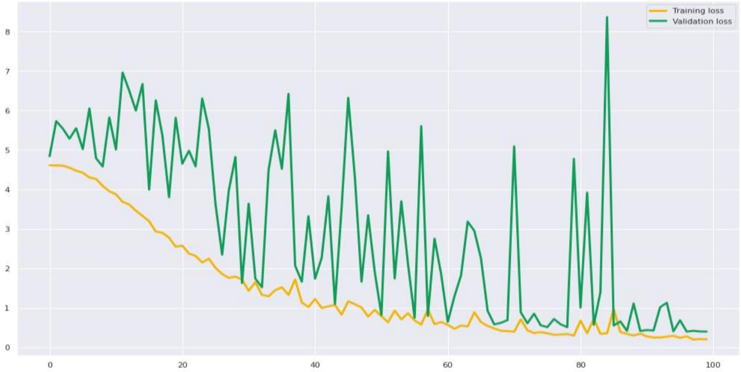
```
#the source of why we use this hyperparameter for RMSprop
#https://keras.io/api/optimizers/sgd/
sgd = tf.keras.optimizers.SGD(learning_rate=0.9, momentum=0.0, nesterov=False, name="SGD")
#
model_3,history_3 = base_line_model(sgd)

model_3.summary()
model_3.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_2 (Dropout)         (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 4ms/step - loss: 0.4287 - accuracy: 0.9091
[0.42866960167884827, 0.9090909361839294]
```

Loss ratio for

validation & training

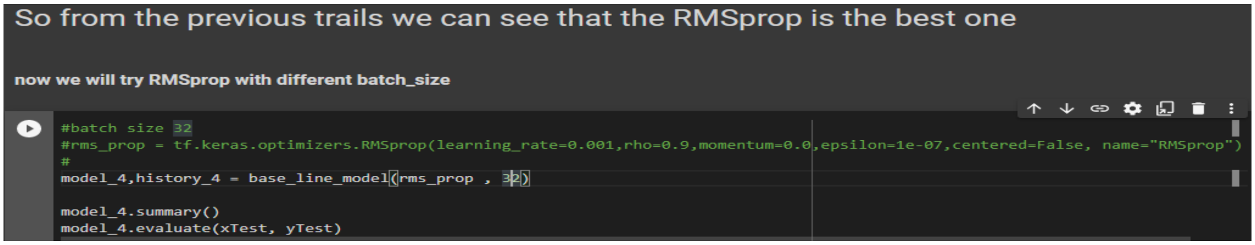


as we can see the SGD is very bad because the curve is not stable at all

So, from all of that the RMSprop is the best one

Now we will use the RMSprop optimizer with different batch size.

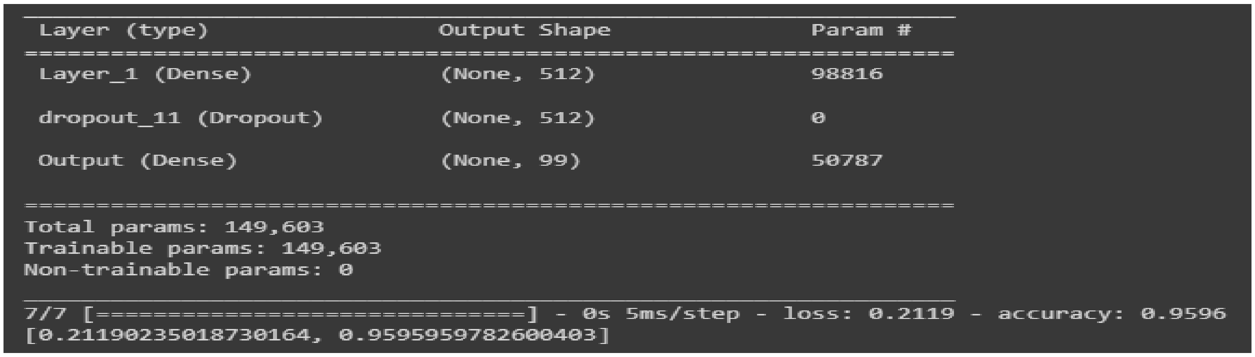
Trial 4: with Batch size = 32



So from the previous trails we can see that the RMSprop is the best one

now we will try RMSprop with different batch_size

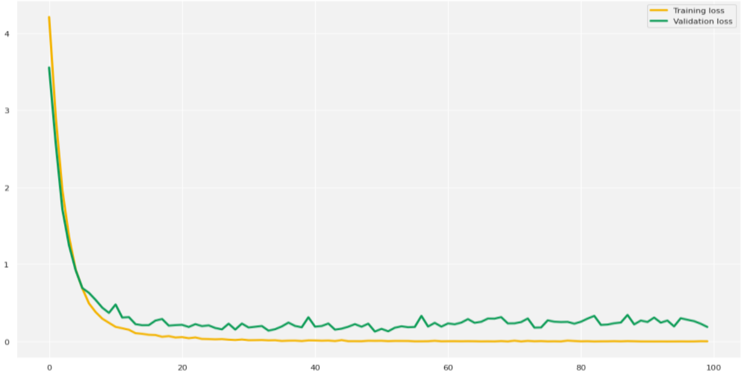
```
#batch size 32
#rms_prop = tf.keras.optimizers.RMSprop(learning_rate=0.001,rho=0.9,momentum=0.0,epsilon=1e-07,centered=False, name="RMSprop")
#
model_4,history_4 = base_line_model(rms_prop , 32)

model_4.summary()
model_4.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_11 (Dropout)        (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 5ms/step - loss: 0.2119 - accuracy: 0.9596
[0.21190235018730164, 0.9595959782600403]
```

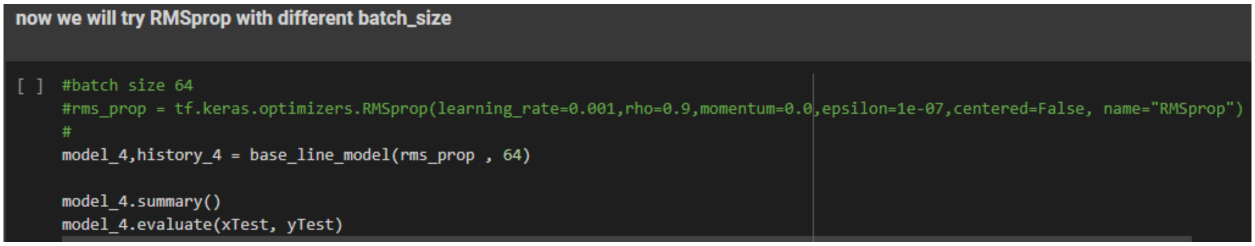
Loss ratio for

validation & training



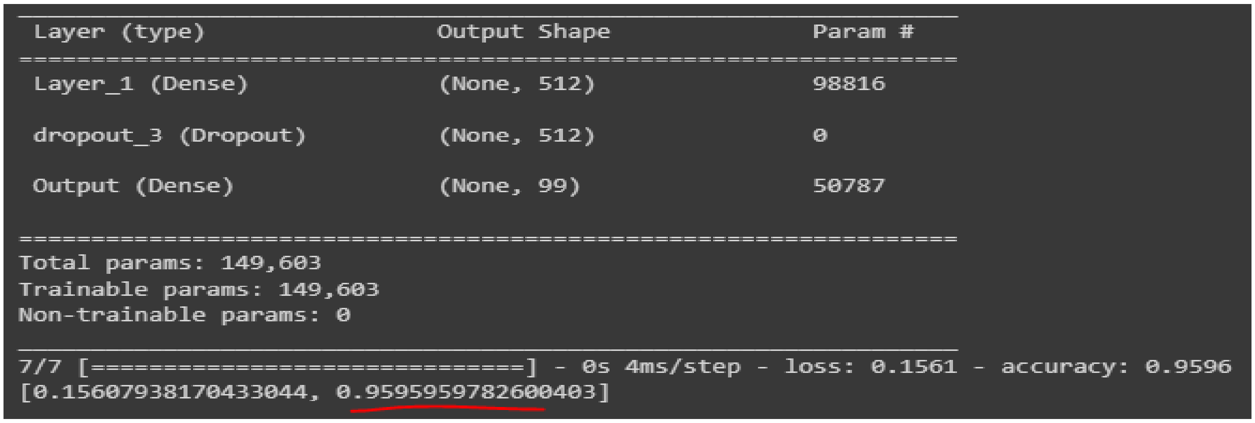
Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs.

Trial 5: with Batch size = 64



now we will try RMSprop with different batch_size

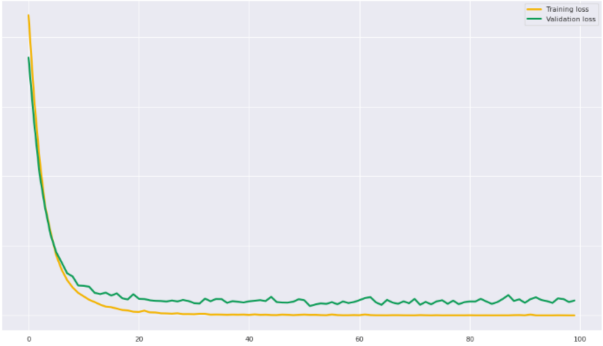
```
#batch size 64
#rms_prop = tf.keras.optimizers.RMSprop(learning_rate=0.001,rho=0.9,momentum=0.0,epsilon=1e-07,centered=False, name="RMSprop")
#
model_4,history_4 = base_line_model(rms_prop , 64)

model_4.summary()
model_4.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_3 (Dropout)         (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 4ms/step - loss: 0.1561 - accuracy: 0.9596
[0.15607938170433044, 0.9595959782600403]
```

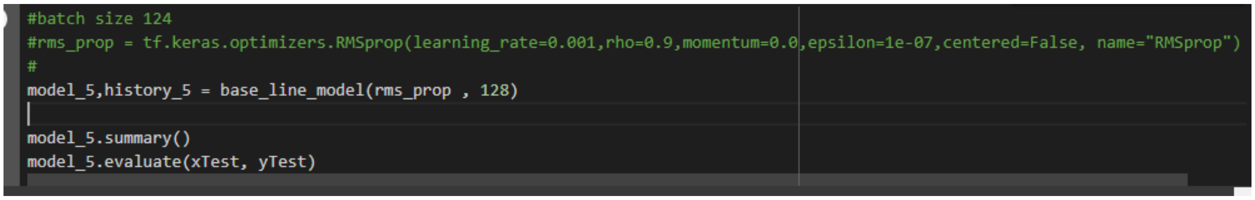
Loss ratio for

validation & training

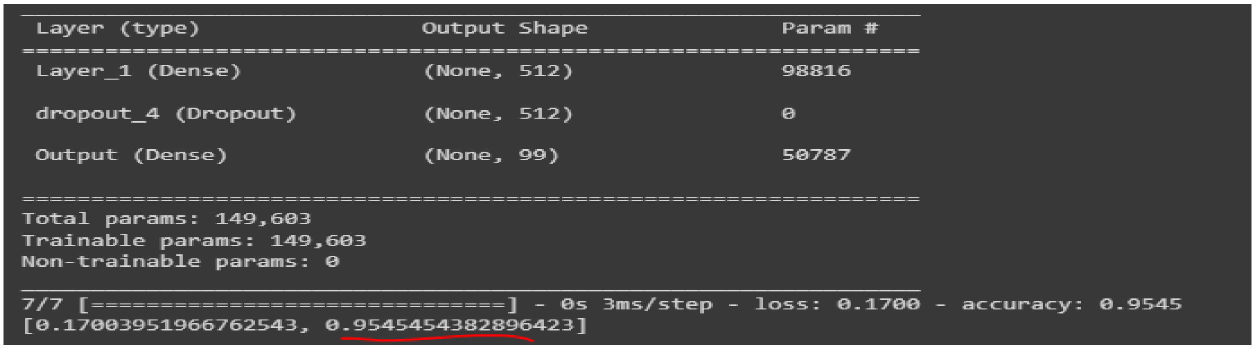


Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 6 : with Batch size = 128



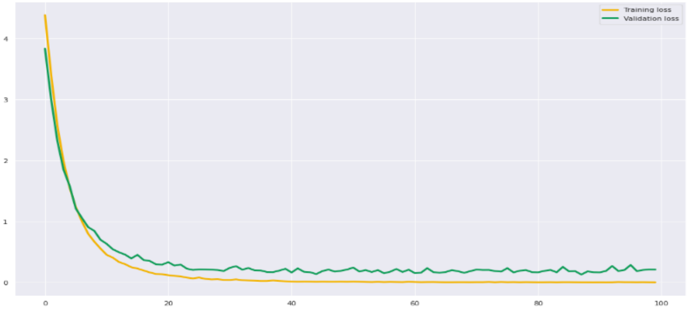
```
#batch size 124
#rms_prop = tf.keras.optimizers.RMSprop(learning_rate=0.001,rho=0.9,momentum=0.0,epsilon=1e-07,centered=False, name="RMSprop")
#
model_5,history_5 = base_line_model(rms_prop , 128)

model_5.summary()
model_5.evaluate(xTest, yTest)
```



```
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_4 (Dropout)         (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 3ms/step - loss: 0.1700 - accuracy: 0.9545
[0.17003951966762543, 0.9545454382896423]
```

Loss ratio for

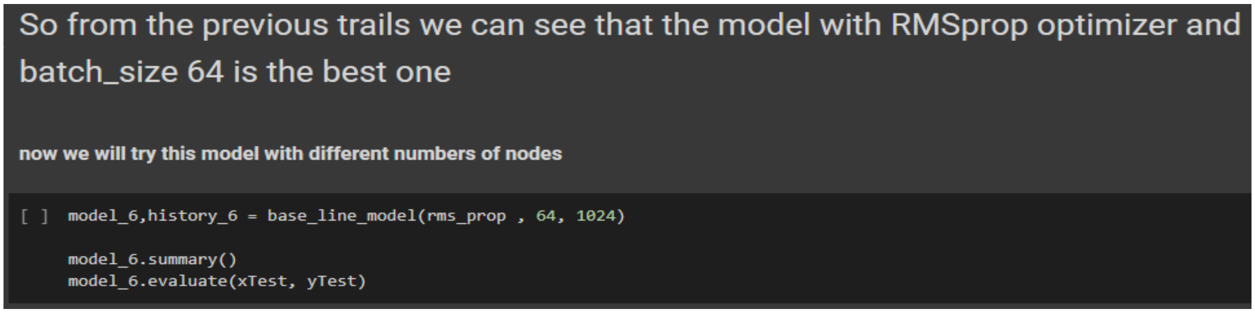
validation & training



Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

So, from all of that the RMSprop is the best optimizer and the batch size = 64 is the best model's hyperparameter Now we will use the RMSprop optimizer and batch size 64 with different number of hidden nodes.

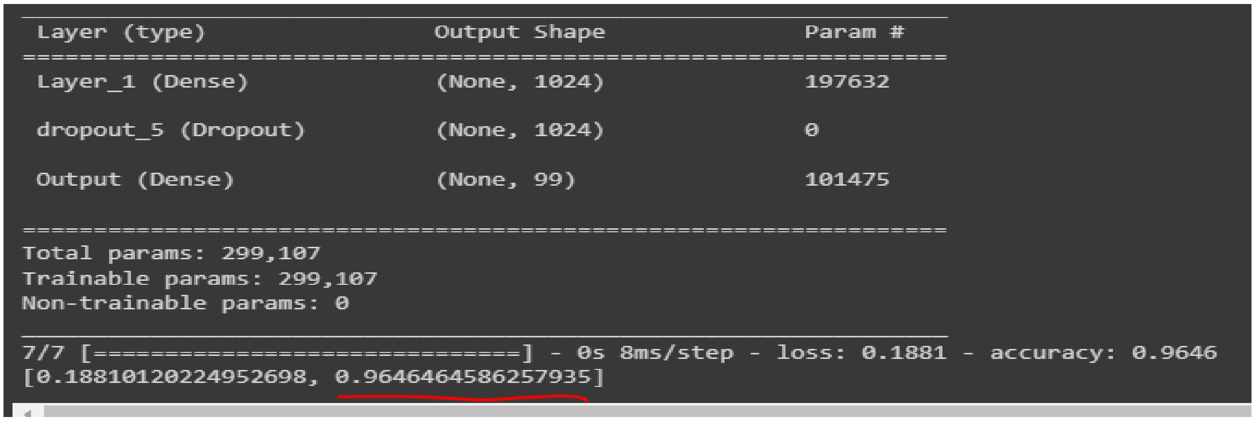
Trial 7 : with 1024 hidden nodes



So from the previous trails we can see that the model with RMSprop optimizer and batch_size 64 is the best one

now we will try this model with different numbers of nodes

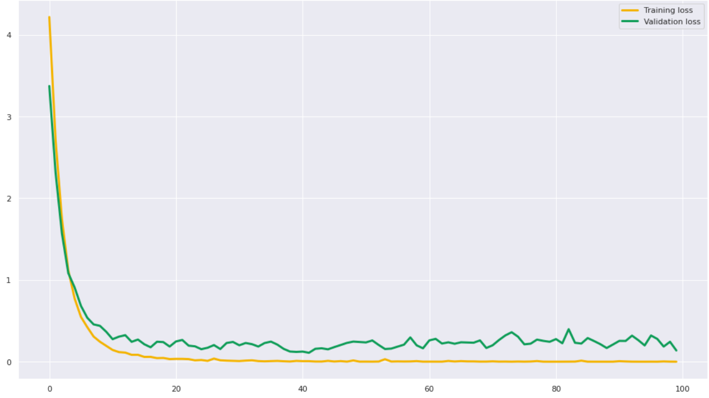
```
[ ]  model_6,history_6 = base_line_model(rms_prop , 64, 1024)

     model_6.summary()
     model_6.evaluate(xTest, yTest)
```



```
 _________________________________________________________________
  Layer (type)                Output Shape              Param #
 =================================================================
  Layer_1 (Dense)             (None, 1024)              197632

  dropout_5 (Dropout)         (None, 1024)              0

  Output (Dense)              (None, 99)                101475

 =================================================================
 Total params: 299,107
 Trainable params: 299,107
 Non-trainable params: 0
 _________________________________________________________________
 7/7 [==============================] - 0s 8ms/step - loss: 0.1881 - accuracy: 0.9646
 [0.18810120224952698, 0.9646464586257935]
```

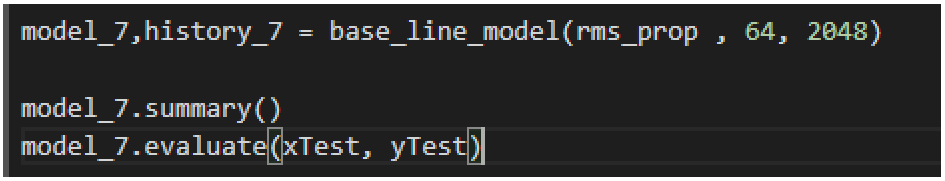
Loss ratio for

validation & training

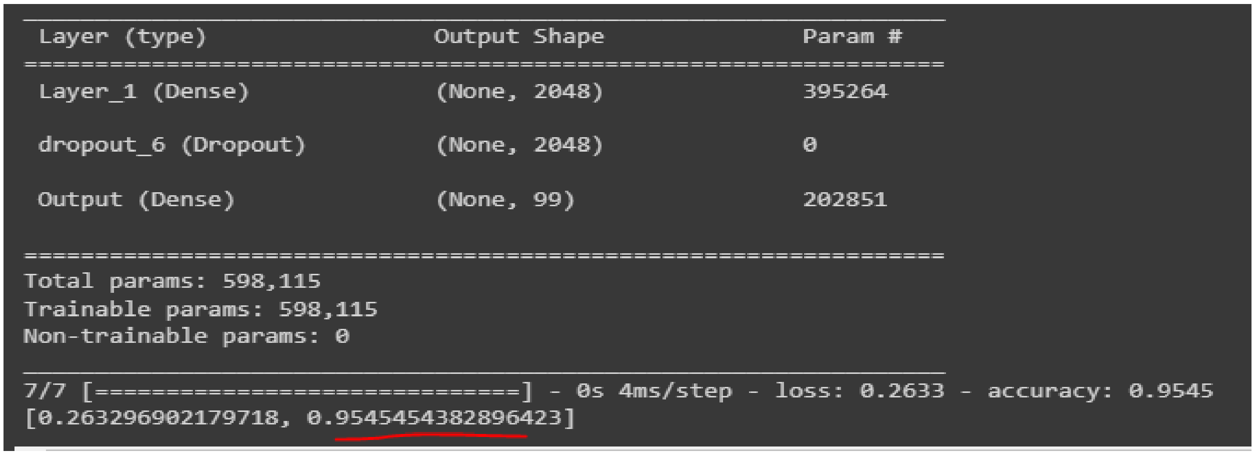


Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 8 : with 2048 hidden nodes



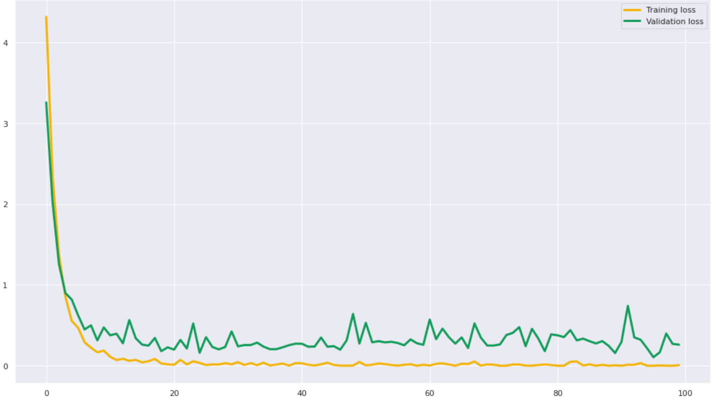
```
model_7,history_7 = base_line_model(rms_prop , 64, 2048)

model_7.summary()
model_7.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 2048)              395264

 dropout_6 (Dropout)         (None, 2048)              0

 Output (Dense)              (None, 99)                202851

=================================================================
Total params: 598,115
Trainable params: 598,115
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 4ms/step - loss: 0.2633 - accuracy: 0.9545
[0.263296902179718, 0.9545454382896423]
```

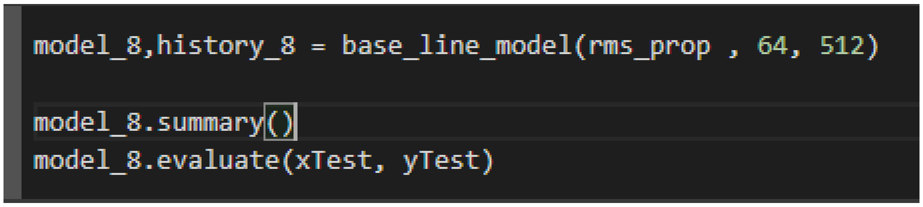
Loss ratio for

validation & training

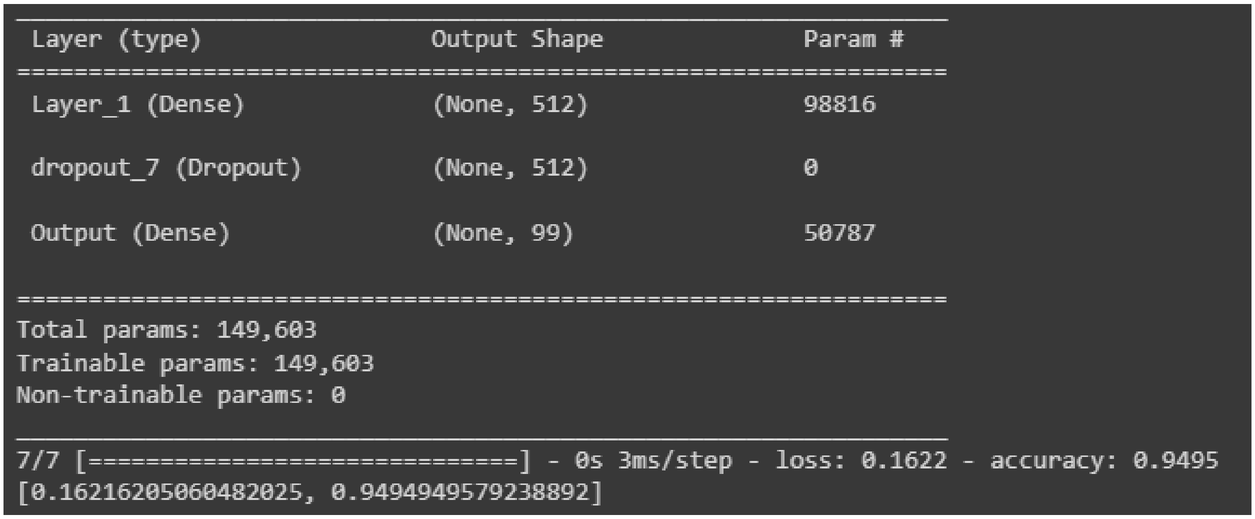


Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 9 : with 512 hidden nodes



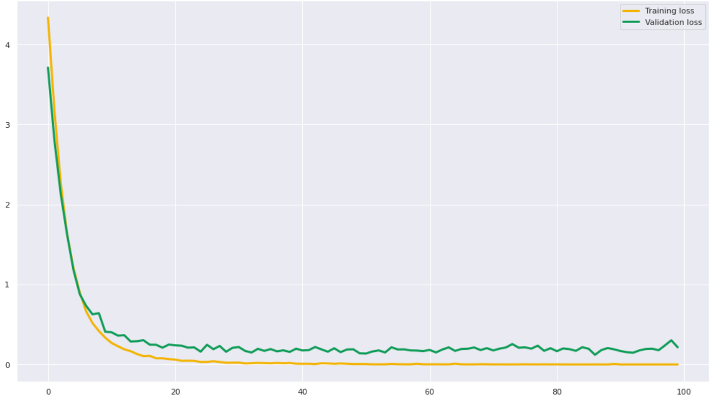
```
model_8,history_8 = base_line_model(rms_prop , 64, 512)

model_8.summary()
model_8.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 512)               98816

 dropout_7 (Dropout)         (None, 512)               0

 Output (Dense)              (None, 99)                50787

=================================================================
Total params: 149,603
Trainable params: 149,603
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 3ms/step - loss: 0.1622 - accuracy: 0.9495
[0.16216205060482025, 0.9494949579238892]
```

Loss ratio for

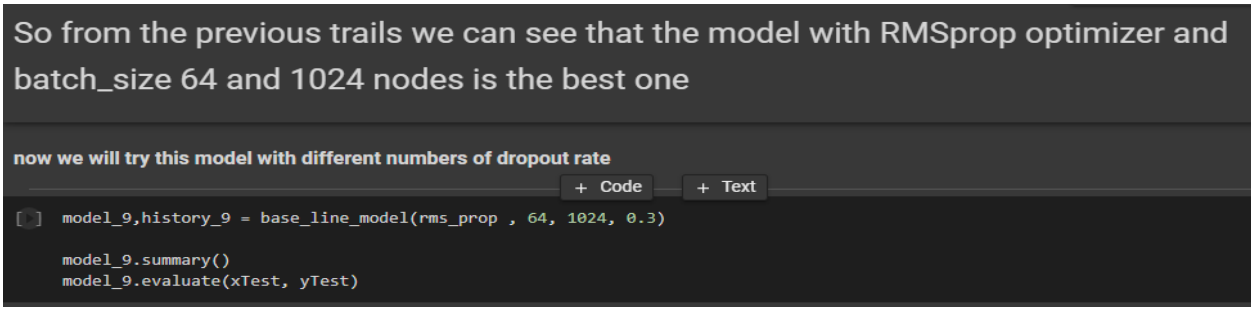
validation & training



Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

So, from all of that the RMSprop is the best optimizer and the batch size = 64 and the number of hidden nodes are 1024 nodes ,that is the best model hyperparameters Now we will use the RMSprop optimizer and batch size 64 with 1024 hidden nodes and with different drop out rate .

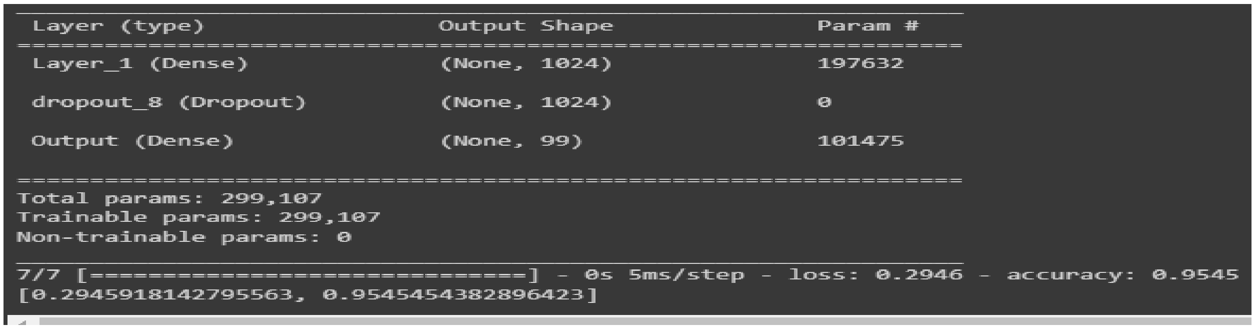
Trial 10 : Drop out rate = 0.3



So from the previous trails we can see that the model with RMSprop optimizer and batch_size 64 and 1024 nodes is the best one

now we will try this model with different numbers of dropout rate

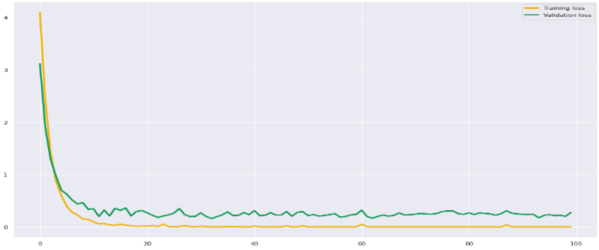
```
model_9,history_9 = base_line_model(rms_prop , 64, 1024, 0.3)

model_9.summary()
model_9.evaluate(xTest, yTest)
```



```
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 1024)              197632

 dropout_8 (Dropout)         (None, 1024)              0

 Output (Dense)              (None, 99)                101475

=================================================================
Total params: 299,107
Trainable params: 299,107
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 5ms/step - loss: 0.2946 - accuracy: 0.9545
[0.2945918142795563, 0.9545454382896423]
```

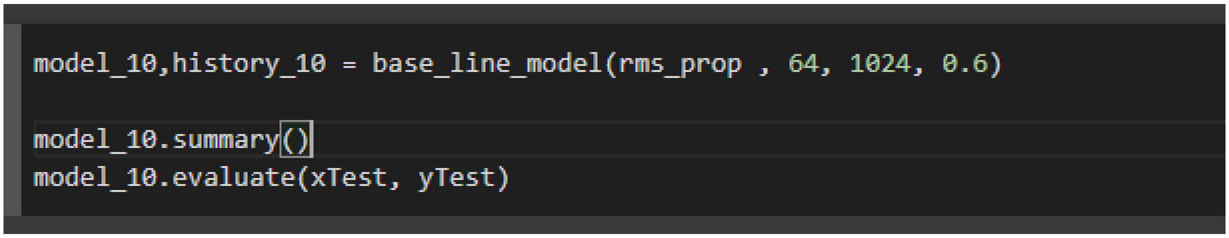
Loss ratio for

validation & training

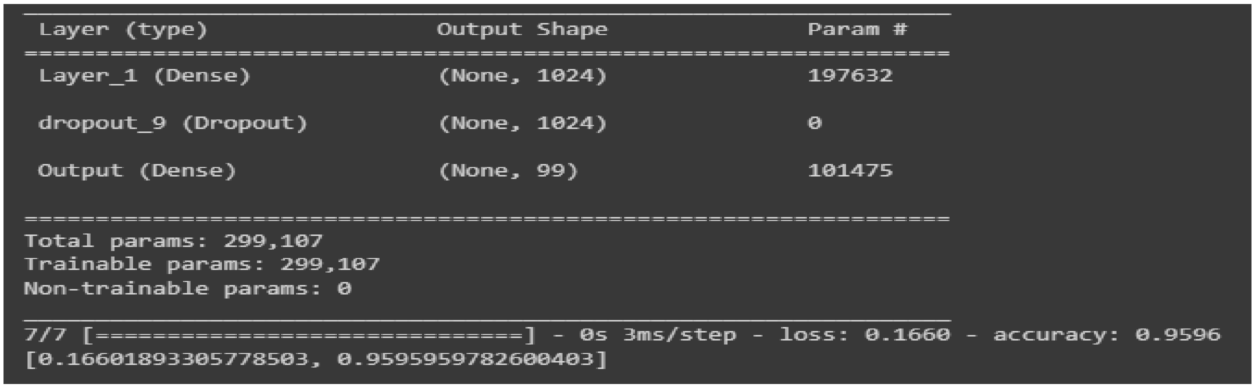


Note : as we can see from this graph we can run only 20 epochs instead of 100 epochs

Trial 11: Drop out rate = 0.6



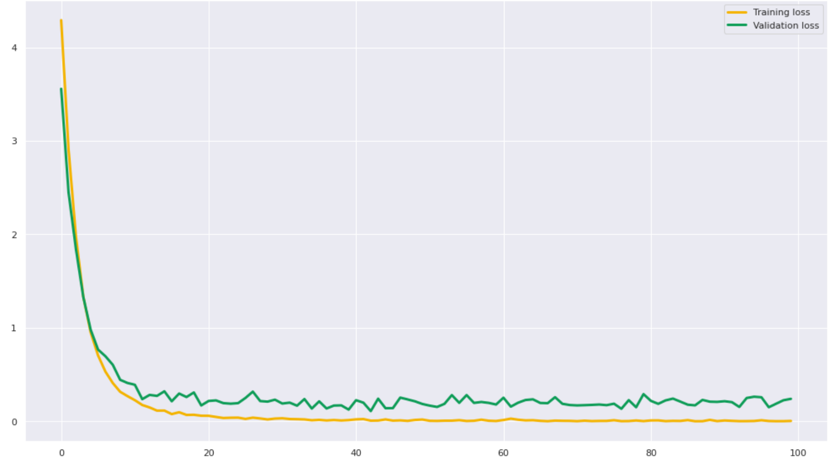
```
model_10,history_10 = base_line_model(rms_prop , 64, 1024, 0.6)

model_10.summary()
model_10.evaluate(xTest, yTest)
```



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 1024)              197632

 dropout_9 (Dropout)         (None, 1024)              0

 Output (Dense)              (None, 99)                101475

=================================================================
Total params: 299,107
Trainable params: 299,107
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 3ms/step - loss: 0.1660 - accuracy: 0.9596
[0.16601893305778503, 0.9595959782600403]
```

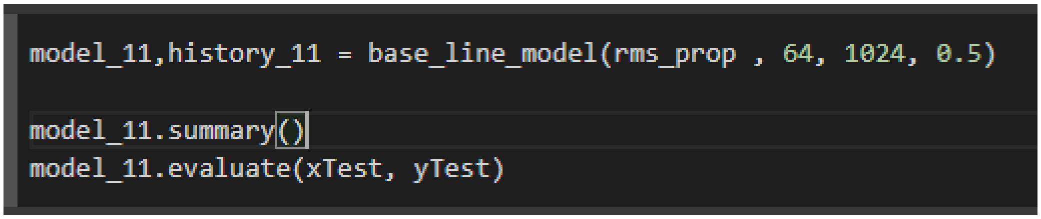
Loss ratio for

validation & training



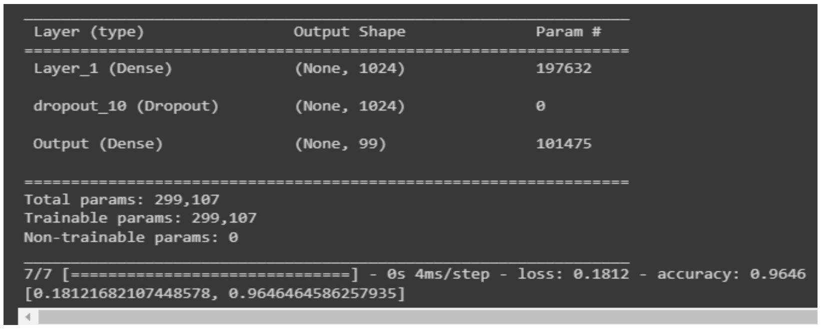
Note : as we can see from this graph we can run only 25 epochs instead of 100 epochs

Trial 12: finally this is the best model with the best hyperparameters :

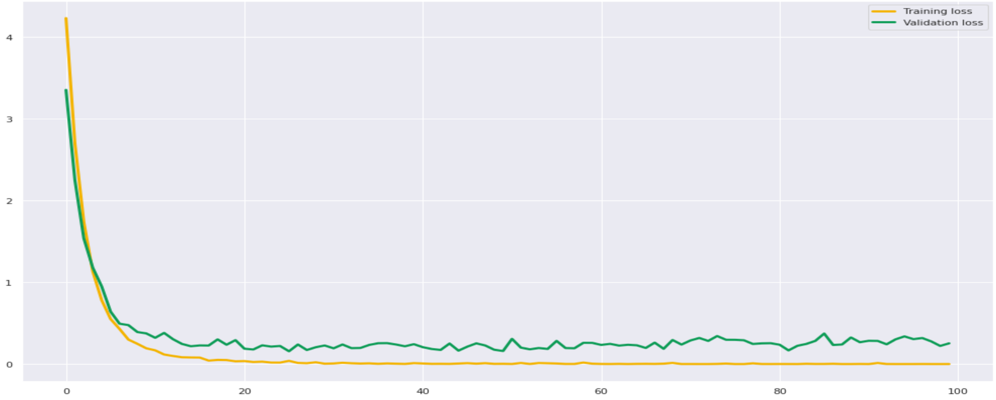


```
model_11,history_11 = base_line_model(rms_prop , 64, 1024, 0.5)

model_11.summary()
model_11.evaluate(xTest, yTest)
```

- Batch Size = 64
- Hidden Nodes Size = 1024
- Drop Rate = 0.5
- Optimizer = RMSprop



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Layer_1 (Dense)             (None, 1024)              197632

 dropout_10 (Dropout)        (None, 1024)              0

 Output (Dense)              (None, 99)                101475

=================================================================
Total params: 299,107
Trainable params: 299,107
Non-trainable params: 0
_________________________________________________________________
7/7 [==============================] - 0s 4ms/step - loss: 0.1812 - accuracy: 0.9646
[0.18121682107448578, 0.9646464586257935]
```



so the best model is model_11 = base_line_model(rms_prop , 64, 1024, 0.5) which has accuracy 0.9646 on test set and 0.1812 loss on test set also and has 100% accuracy on training data

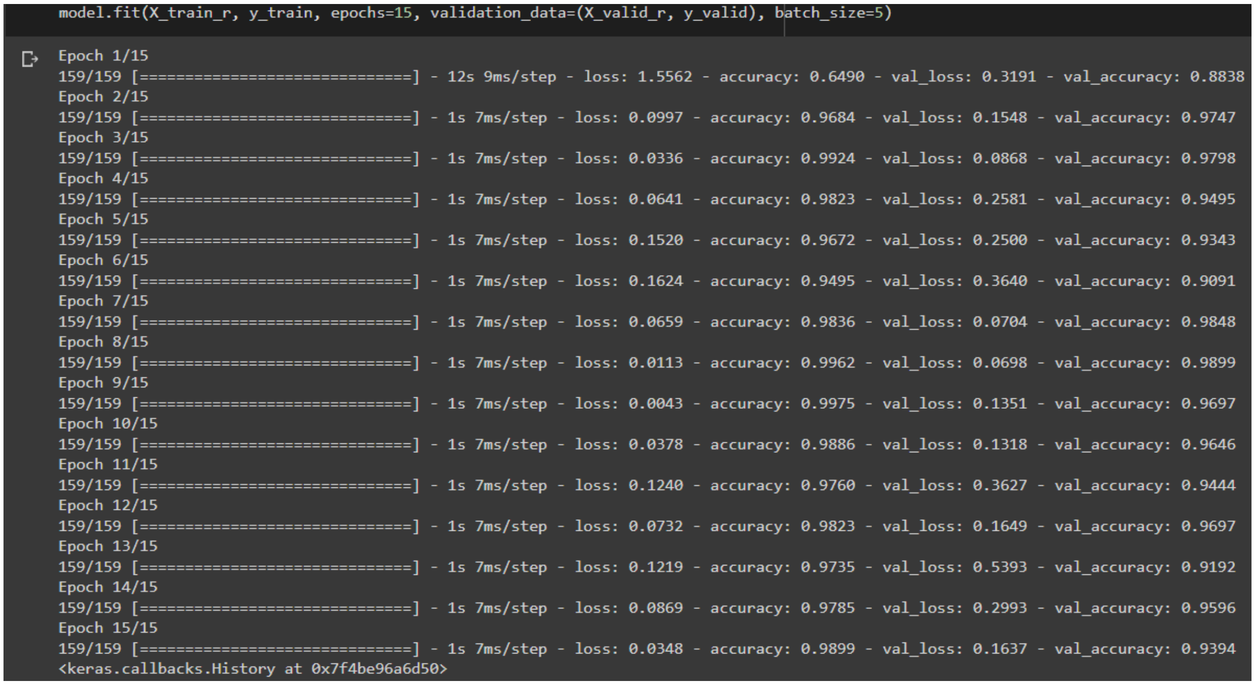
CNN Model

We decided to tune the following hyperparameters:

- Batch Size
- Hidden Nodes Size
- Dropout Rate
- Optimizer

Let’s Start with Batch Size. We Used the following values:

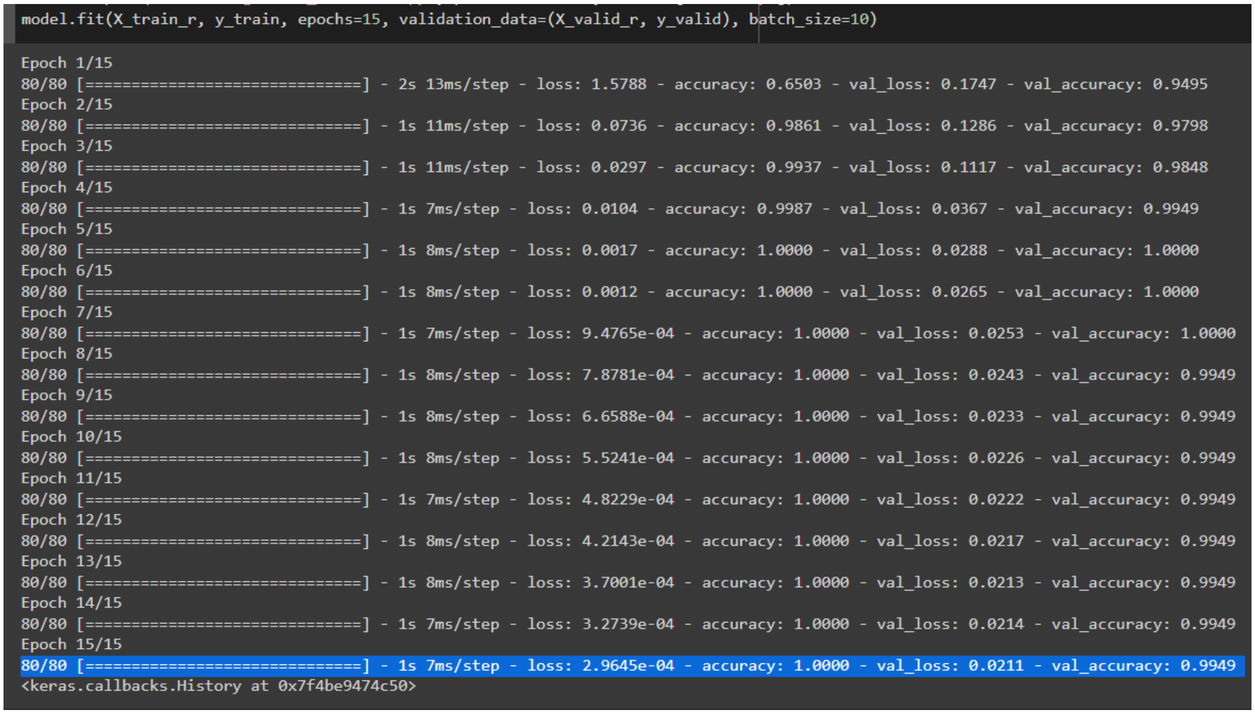
1- Batch_size=5



```
model.fit(X_train_r, y_train, epochs=15, validation_data=(X_valid_r, y_valid), batch_size=5)

Epoch 1/15
159/159 [==============================] - 12s 9ms/step - loss: 1.5562 - accuracy: 0.6490 - val_loss: 0.3191 - val_accuracy: 0.8838
Epoch 2/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0997 - accuracy: 0.9684 - val_loss: 0.1548 - val_accuracy: 0.9747
Epoch 3/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0336 - accuracy: 0.9924 - val_loss: 0.0868 - val_accuracy: 0.9798
Epoch 4/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0641 - accuracy: 0.9823 - val_loss: 0.2581 - val_accuracy: 0.9495
Epoch 5/15
159/159 [==============================] - 1s 7ms/step - loss: 0.1520 - accuracy: 0.9672 - val_loss: 0.2500 - val_accuracy: 0.9343
Epoch 6/15
159/159 [==============================] - 1s 7ms/step - loss: 0.1624 - accuracy: 0.9495 - val_loss: 0.3640 - val_accuracy: 0.9091
Epoch 7/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0659 - accuracy: 0.9836 - val_loss: 0.0704 - val_accuracy: 0.9848
Epoch 8/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0113 - accuracy: 0.9962 - val_loss: 0.0698 - val_accuracy: 0.9899
Epoch 9/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0043 - accuracy: 0.9975 - val_loss: 0.1351 - val_accuracy: 0.9697
Epoch 10/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0378 - accuracy: 0.9886 - val_loss: 0.1318 - val_accuracy: 0.9646
Epoch 11/15
159/159 [==============================] - 1s 7ms/step - loss: 0.1240 - accuracy: 0.9760 - val_loss: 0.3627 - val_accuracy: 0.9444
Epoch 12/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0732 - accuracy: 0.9823 - val_loss: 0.1649 - val_accuracy: 0.9697
Epoch 13/15
159/159 [==============================] - 1s 7ms/step - loss: 0.1219 - accuracy: 0.9735 - val_loss: 0.5393 - val_accuracy: 0.9192
Epoch 14/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0869 - accuracy: 0.9785 - val_loss: 0.2993 - val_accuracy: 0.9596
Epoch 15/15
159/159 [==============================] - 1s 7ms/step - loss: 0.0348 - accuracy: 0.9899 - val_loss: 0.1637 - val_accuracy: 0.9394
<keras.callbacks.History at 0x7f4be96a6d50>
```

```
loss: 0.0348 - accuracy: 0.9899 - val_loss: 0.1637 - val_accuracy: 0.9394
```

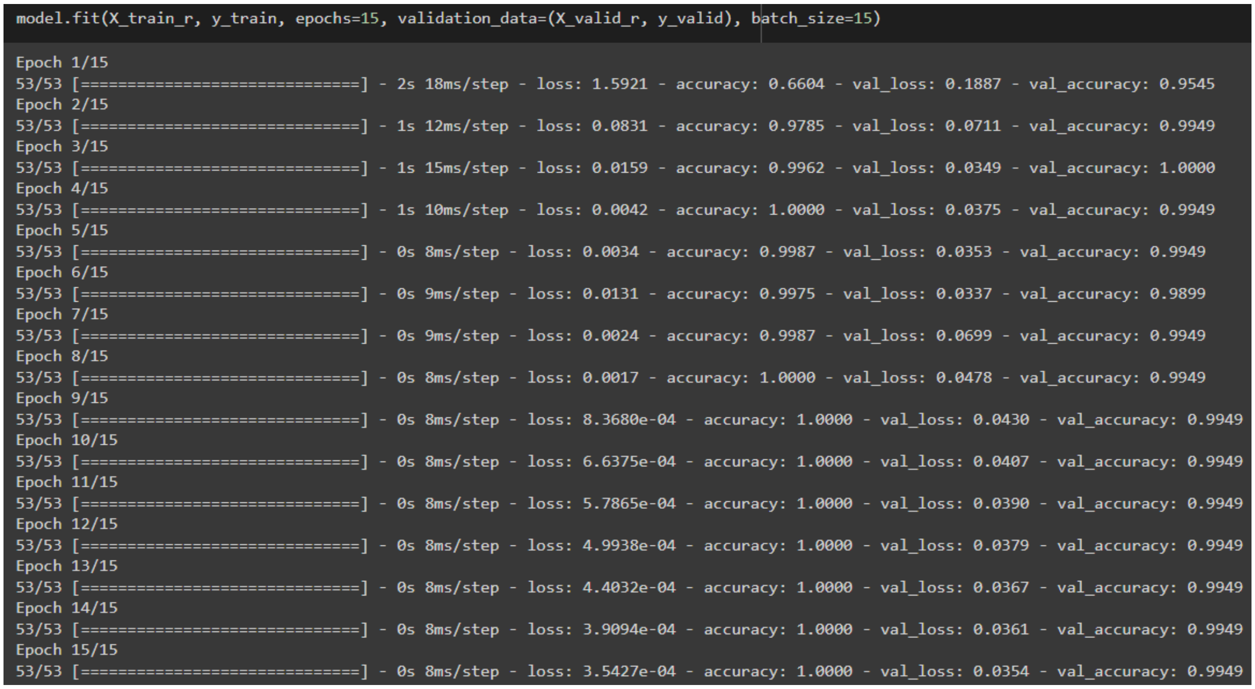
2- Batch_size=10



```
model.fit(X_train_r, y_train, epochs=15, validation_data=(X_valid_r, y_valid), batch_size=10)

Epoch 1/15
80/80 [==============================] - 2s 13ms/step - loss: 1.5788 - accuracy: 0.6503 - val_loss: 0.1747 - val_accuracy: 0.9495
Epoch 2/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0736 - accuracy: 0.9861 - val_loss: 0.1286 - val_accuracy: 0.9798
Epoch 3/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0297 - accuracy: 0.9937 - val_loss: 0.1117 - val_accuracy: 0.9848
Epoch 4/15
80/80 [==============================] - 1s 7ms/step - loss: 0.0104 - accuracy: 0.9987 - val_loss: 0.0367 - val_accuracy: 0.9949
Epoch 5/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0288 - val_accuracy: 1.0000
Epoch 6/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0265 - val_accuracy: 1.0000
Epoch 7/15
80/80 [==============================] - 1s 7ms/step - loss: 9.4765e-04 - accuracy: 1.0000 - val_loss: 0.0253 - val_accuracy: 1.0000
Epoch 8/15
80/80 [==============================] - 1s 8ms/step - loss: 7.8781e-04 - accuracy: 1.0000 - val_loss: 0.0243 - val_accuracy: 0.9949
Epoch 9/15
80/80 [==============================] - 1s 8ms/step - loss: 6.6588e-04 - accuracy: 1.0000 - val_loss: 0.0233 - val_accuracy: 0.9949
Epoch 10/15
80/80 [==============================] - 1s 8ms/step - loss: 5.5241e-04 - accuracy: 1.0000 - val_loss: 0.0226 - val_accuracy: 0.9949
Epoch 11/15
80/80 [==============================] - 1s 7ms/step - loss: 4.8229e-04 - accuracy: 1.0000 - val_loss: 0.0222 - val_accuracy: 0.9949
Epoch 12/15
80/80 [==============================] - 1s 8ms/step - loss: 4.2143e-04 - accuracy: 1.0000 - val_loss: 0.0217 - val_accuracy: 0.9949
Epoch 13/15
80/80 [==============================] - 1s 8ms/step - loss: 3.7001e-04 - accuracy: 1.0000 - val_loss: 0.0213 - val_accuracy: 0.9949
Epoch 14/15
80/80 [==============================] - 1s 7ms/step - loss: 3.2739e-04 - accuracy: 1.0000 - val_loss: 0.0214 - val_accuracy: 0.9949
Epoch 15/15
80/80 [==============================] - 1s 7ms/step - loss: 2.9645e-04 - accuracy: 1.0000 - val_loss: 0.0211 - val_accuracy: 0.9949
<keras.callbacks.History at 0x7f4be9474c50>
```

```
loss: 2.9645e-04 -accuracy: 1.0 -val_loss: 0.0211 -val_accuracy: 0.9949
```

3- Batch_size=15



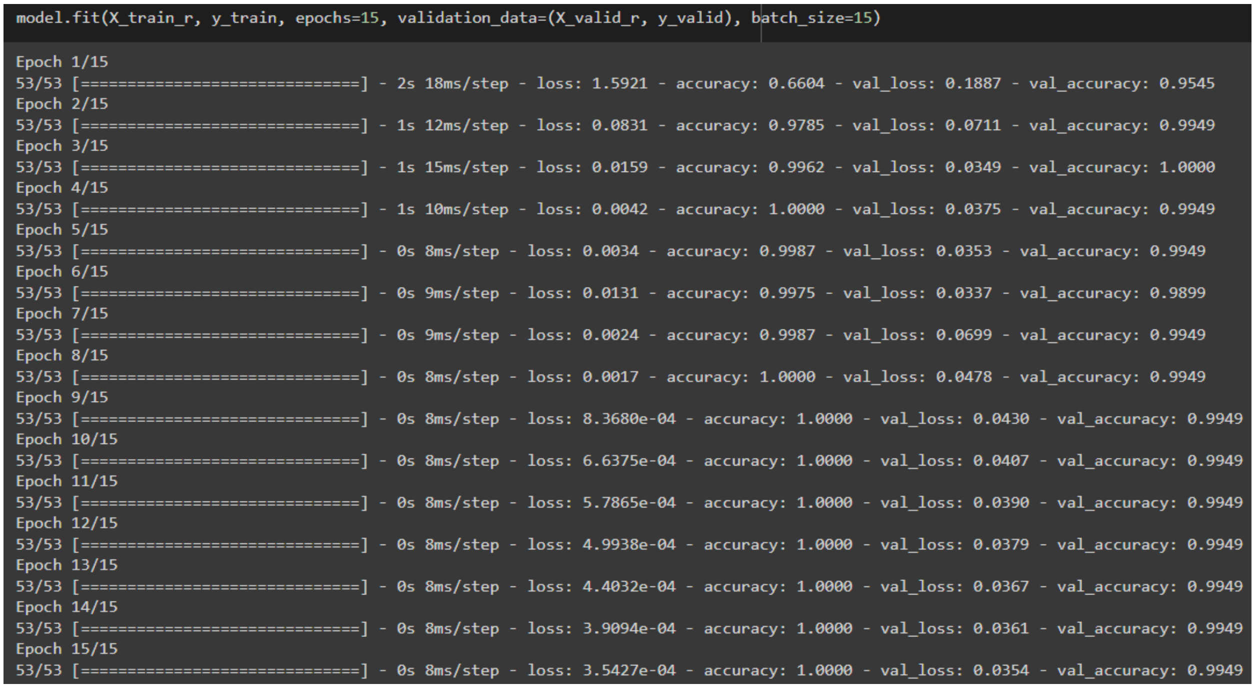
```
model.fit(X_train_r, y_train, epochs=15, validation_data=(X_valid_r, y_valid), batch_size=15)

Epoch 1/15
53/53 [==============================] - 2s 18ms/step - loss: 1.5921 - accuracy: 0.6604 - val_loss: 0.1887 - val_accuracy: 0.9545
Epoch 2/15
53/53 [==============================] - 1s 12ms/step - loss: 0.0831 - accuracy: 0.9785 - val_loss: 0.0711 - val_accuracy: 0.9949
Epoch 3/15
53/53 [==============================] - 1s 15ms/step - loss: 0.0159 - accuracy: 0.9962 - val_loss: 0.0349 - val_accuracy: 1.0000
Epoch 4/15
53/53 [==============================] - 1s 10ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.0375 - val_accuracy: 0.9949
Epoch 5/15
53/53 [==============================] - 0s 8ms/step - loss: 0.0034 - accuracy: 0.9987 - val_loss: 0.0353 - val_accuracy: 0.9949
Epoch 6/15
53/53 [==============================] - 0s 9ms/step - loss: 0.0131 - accuracy: 0.9975 - val_loss: 0.0337 - val_accuracy: 0.9899
Epoch 7/15
53/53 [==============================] - 0s 9ms/step - loss: 0.0024 - accuracy: 0.9987 - val_loss: 0.0699 - val_accuracy: 0.9949
Epoch 8/15
53/53 [==============================] - 0s 8ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0478 - val_accuracy: 0.9949
Epoch 9/15
53/53 [==============================] - 0s 8ms/step - loss: 8.3680e-04 - accuracy: 1.0000 - val_loss: 0.0430 - val_accuracy: 0.9949
Epoch 10/15
53/53 [==============================] - 0s 8ms/step - loss: 6.6375e-04 - accuracy: 1.0000 - val_loss: 0.0407 - val_accuracy: 0.9949
Epoch 11/15
53/53 [==============================] - 0s 8ms/step - loss: 5.7865e-04 - accuracy: 1.0000 - val_loss: 0.0390 - val_accuracy: 0.9949
Epoch 12/15
53/53 [==============================] - 0s 8ms/step - loss: 4.9938e-04 - accuracy: 1.0000 - val_loss: 0.0379 - val_accuracy: 0.9949
Epoch 13/15
53/53 [==============================] - 0s 8ms/step - loss: 4.4032e-04 - accuracy: 1.0000 - val_loss: 0.0367 - val_accuracy: 0.9949
Epoch 14/15
53/53 [==============================] - 0s 8ms/step - loss: 3.9094e-04 - accuracy: 1.0000 - val_loss: 0.0361 - val_accuracy: 0.9949
Epoch 15/15
53/53 [==============================] - 0s 8ms/step - loss: 3.5427e-04 - accuracy: 1.0000 - val_loss: 0.0354 - val_accuracy: 0.9949
```

```
loss: 3.5427e-04 - accuracy: 1.0000 - val_loss: 0.0354 - val_accuracy:
0.9949
```

The best batch size was 10 so we stick with it.

Next we go with hidden nodes Size. We chose the next values:

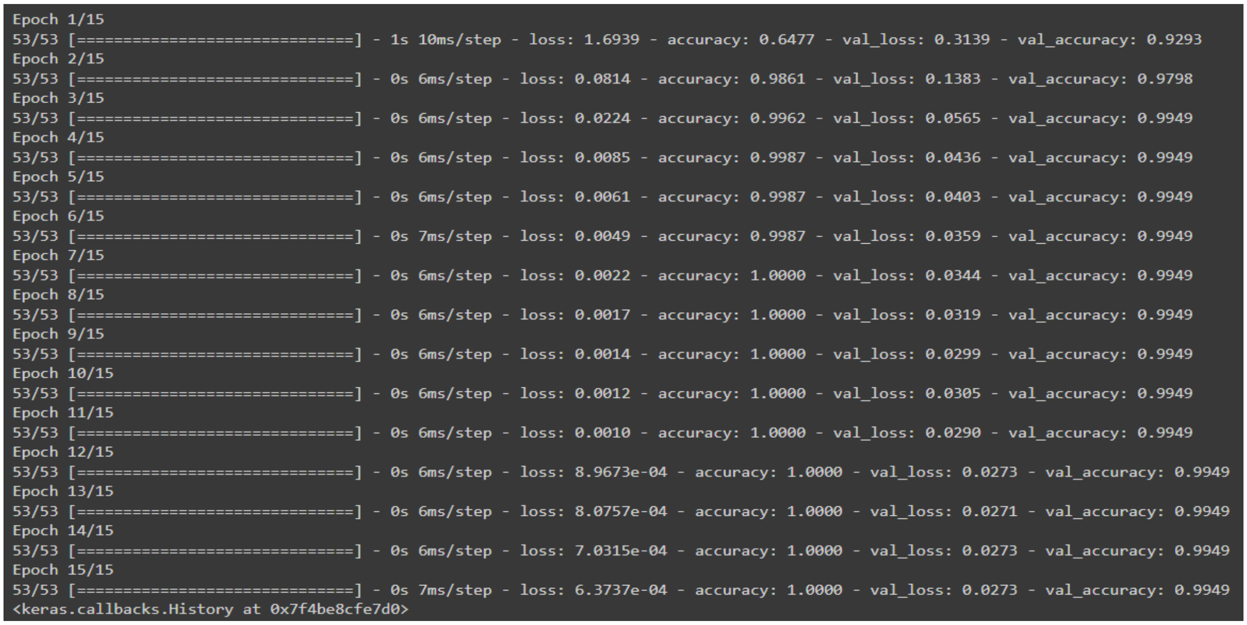
1- Hidden nodes= 512



```
model.fit(X_train_r, y_train, epochs=15, validation_data=(X_valid_r, y_valid), batch_size=15)

Epoch 1/15
53/53 [==============================] - 2s 18ms/step - loss: 1.5921 - accuracy: 0.6604 - val_loss: 0.1887 - val_accuracy: 0.9545
Epoch 2/15
53/53 [==============================] - 1s 12ms/step - loss: 0.0831 - accuracy: 0.9785 - val_loss: 0.0711 - val_accuracy: 0.9949
Epoch 3/15
53/53 [==============================] - 1s 15ms/step - loss: 0.0159 - accuracy: 0.9962 - val_loss: 0.0349 - val_accuracy: 1.0000
Epoch 4/15
53/53 [==============================] - 1s 10ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.0375 - val_accuracy: 0.9949
Epoch 5/15
53/53 [==============================] - 0s 8ms/step - loss: 0.0034 - accuracy: 0.9987 - val_loss: 0.0353 - val_accuracy: 0.9949
Epoch 6/15
53/53 [==============================] - 0s 9ms/step - loss: 0.0131 - accuracy: 0.9975 - val_loss: 0.0337 - val_accuracy: 0.9899
Epoch 7/15
53/53 [==============================] - 0s 9ms/step - loss: 0.0024 - accuracy: 0.9987 - val_loss: 0.0699 - val_accuracy: 0.9949
Epoch 8/15
53/53 [==============================] - 0s 8ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0478 - val_accuracy: 0.9949
Epoch 9/15
53/53 [==============================] - 0s 8ms/step - loss: 8.3680e-04 - accuracy: 1.0000 - val_loss: 0.0430 - val_accuracy: 0.9949
Epoch 10/15
53/53 [==============================] - 0s 8ms/step - loss: 6.6375e-04 - accuracy: 1.0000 - val_loss: 0.0407 - val_accuracy: 0.9949
Epoch 11/15
53/53 [==============================] - 0s 8ms/step - loss: 5.7865e-04 - accuracy: 1.0000 - val_loss: 0.0390 - val_accuracy: 0.9949
Epoch 12/15
53/53 [==============================] - 0s 8ms/step - loss: 4.9938e-04 - accuracy: 1.0000 - val_loss: 0.0379 - val_accuracy: 0.9949
Epoch 13/15
53/53 [==============================] - 0s 8ms/step - loss: 4.4032e-04 - accuracy: 1.0000 - val_loss: 0.0367 - val_accuracy: 0.9949
Epoch 14/15
53/53 [==============================] - 0s 8ms/step - loss: 3.9094e-04 - accuracy: 1.0000 - val_loss: 0.0361 - val_accuracy: 0.9949
Epoch 15/15
53/53 [==============================] - 0s 8ms/step - loss: 3.5427e-04 - accuracy: 1.0000 - val_loss: 0.0354 - val_accuracy: 0.9949
```

```
loss: 3.5427e-04 -accuracy: 1.0000 -val_loss: 0.0354 -val_accuracy: 0.9949
```

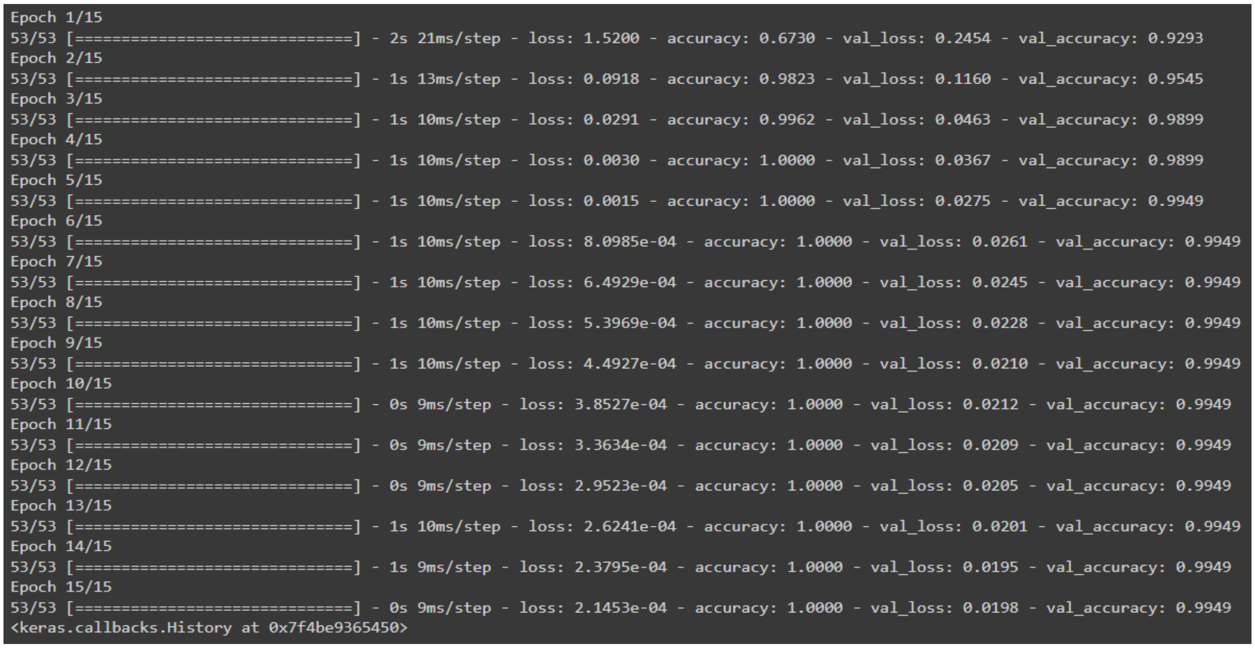
2- Hidden nodes= 256



```
Epoch 1/15
53/53 [==============================] - 1s 10ms/step - loss: 1.6939 - accuracy: 0.6477 - val_loss: 0.3139 - val_accuracy: 0.9293
Epoch 2/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0814 - accuracy: 0.9861 - val_loss: 0.1383 - val_accuracy: 0.9798
Epoch 3/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0224 - accuracy: 0.9962 - val_loss: 0.0565 - val_accuracy: 0.9949
Epoch 4/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0085 - accuracy: 0.9987 - val_loss: 0.0436 - val_accuracy: 0.9949
Epoch 5/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0061 - accuracy: 0.9987 - val_loss: 0.0403 - val_accuracy: 0.9949
Epoch 6/15
53/53 [==============================] - 0s 7ms/step - loss: 0.0049 - accuracy: 0.9987 - val_loss: 0.0359 - val_accuracy: 0.9949
Epoch 7/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.0344 - val_accuracy: 0.9949
Epoch 8/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0319 - val_accuracy: 0.9949
Epoch 9/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0299 - val_accuracy: 0.9949
Epoch 10/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0305 - val_accuracy: 0.9949
Epoch 11/15
53/53 [==============================] - 0s 6ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0290 - val_accuracy: 0.9949
Epoch 12/15
53/53 [==============================] - 0s 6ms/step - loss: 8.9673e-04 - accuracy: 1.0000 - val_loss: 0.0273 - val_accuracy: 0.9949
Epoch 13/15
53/53 [==============================] - 0s 6ms/step - loss: 8.0757e-04 - accuracy: 1.0000 - val_loss: 0.0271 - val_accuracy: 0.9949
Epoch 14/15
53/53 [==============================] - 0s 6ms/step - loss: 7.0315e-04 - accuracy: 1.0000 - val_loss: 0.0273 - val_accuracy: 0.9949
Epoch 15/15
53/53 [==============================] - 0s 7ms/step - loss: 6.3737e-04 - accuracy: 1.0000 - val_loss: 0.0273 - val_accuracy: 0.9949
<keras.callbacks.History at 0x7f4be8cfe7d0>
```

```
loss: 6.3737e-04 -accuracy: 1.00 -val_loss: 0.0273 -val_accuracy:0.9949
```

3- Hidden nodes= 1024



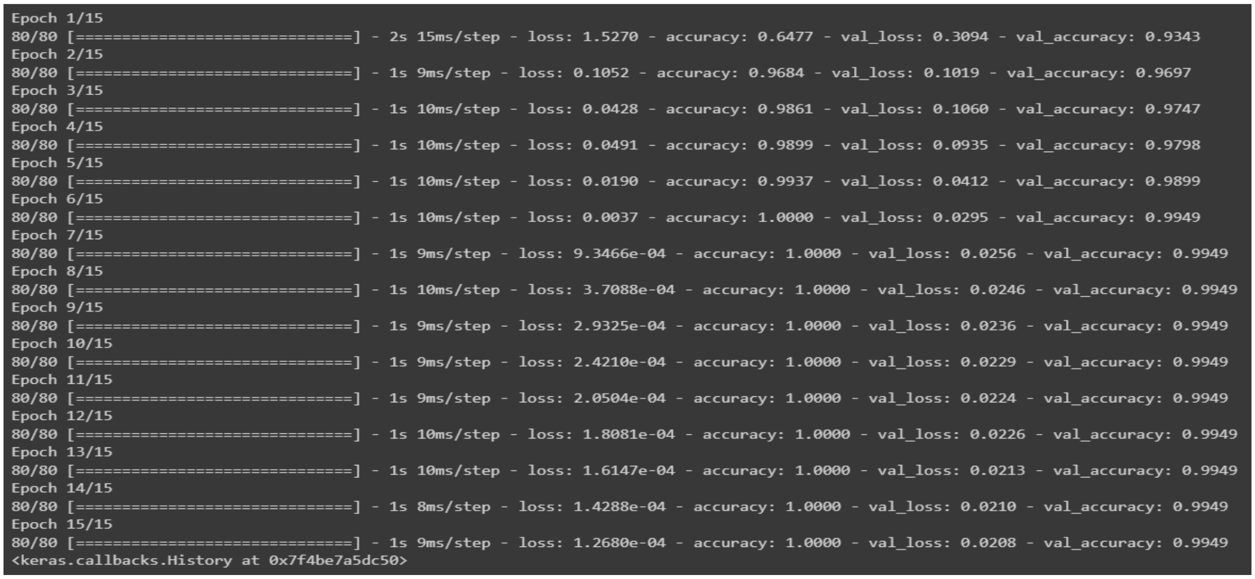
```
Epoch 1/15
53/53 [==============================] - 2s 21ms/step - loss: 1.5200 - accuracy: 0.6730 - val_loss: 0.2454 - val_accuracy: 0.9293
Epoch 2/15
53/53 [==============================] - 1s 13ms/step - loss: 0.0918 - accuracy: 0.9823 - val_loss: 0.1160 - val_accuracy: 0.9545
Epoch 3/15
53/53 [==============================] - 1s 10ms/step - loss: 0.0291 - accuracy: 0.9962 - val_loss: 0.0463 - val_accuracy: 0.9899
Epoch 4/15
53/53 [==============================] - 1s 10ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0367 - val_accuracy: 0.9899
Epoch 5/15
53/53 [==============================] - 1s 10ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0275 - val_accuracy: 0.9949
Epoch 6/15
53/53 [==============================] - 1s 10ms/step - loss: 8.0985e-04 - accuracy: 1.0000 - val_loss: 0.0261 - val_accuracy: 0.9949
Epoch 7/15
53/53 [==============================] - 1s 10ms/step - loss: 6.4929e-04 - accuracy: 1.0000 - val_loss: 0.0245 - val_accuracy: 0.9949
Epoch 8/15
53/53 [==============================] - 1s 10ms/step - loss: 5.3969e-04 - accuracy: 1.0000 - val_loss: 0.0228 - val_accuracy: 0.9949
Epoch 9/15
53/53 [==============================] - 1s 10ms/step - loss: 4.4927e-04 - accuracy: 1.0000 - val_loss: 0.0210 - val_accuracy: 0.9949
Epoch 10/15
53/53 [==============================] - 0s 9ms/step - loss: 3.8527e-04 - accuracy: 1.0000 - val_loss: 0.0212 - val_accuracy: 0.9949
Epoch 11/15
53/53 [==============================] - 0s 9ms/step - loss: 3.3634e-04 - accuracy: 1.0000 - val_loss: 0.0209 - val_accuracy: 0.9949
Epoch 12/15
53/53 [==============================] - 0s 9ms/step - loss: 2.9523e-04 - accuracy: 1.0000 - val_loss: 0.0205 - val_accuracy: 0.9949
Epoch 13/15
53/53 [==============================] - 1s 10ms/step - loss: 2.6241e-04 - accuracy: 1.0000 - val_loss: 0.0201 - val_accuracy: 0.9949
Epoch 14/15
53/53 [==============================] - 1s 9ms/step - loss: 2.3795e-04 - accuracy: 1.0000 - val_loss: 0.0195 - val_accuracy: 0.9949
Epoch 15/15
53/53 [==============================] - 0s 9ms/step - loss: 2.1453e-04 - accuracy: 1.0000 - val_loss: 0.0198 - val_accuracy: 0.9949
<keras.callbacks.History at 0x7f4be9365450>
```

```
loss:2.1453e-04 -accuracy: 1.00 -val_loss: 0.0198 -val_accuracy: 0.9949
```

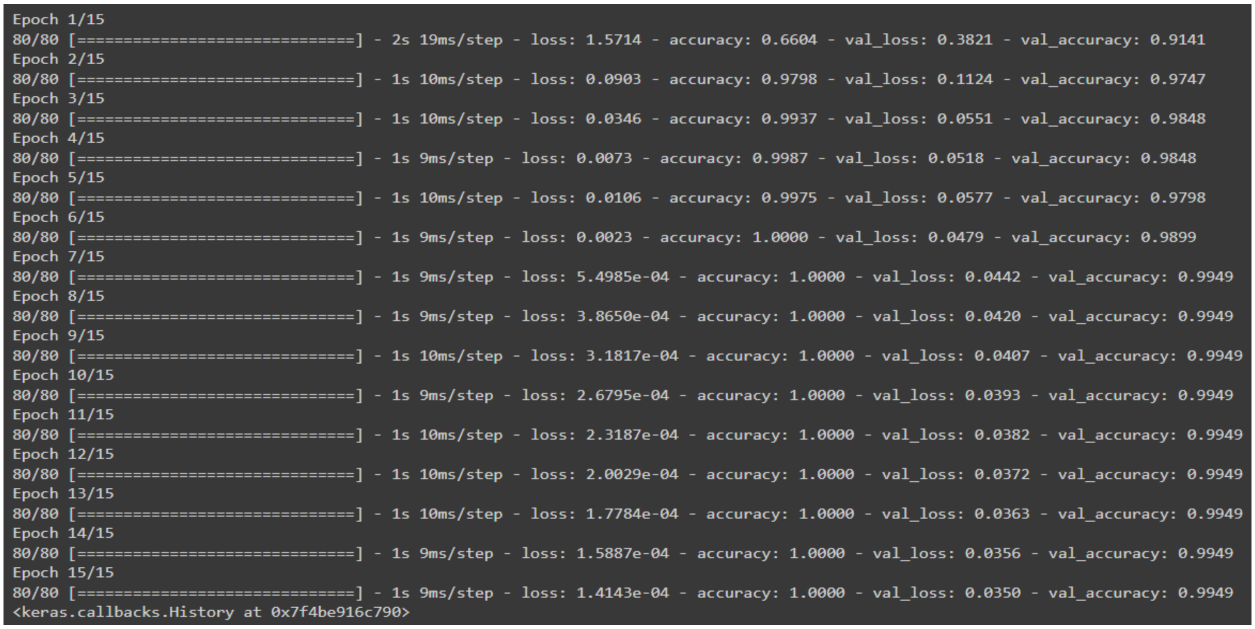
The best hidden nodes number was 1024 so we stick with it.

Next we go with dropout rate. We chose the next values:

1- Dropout= 0.2

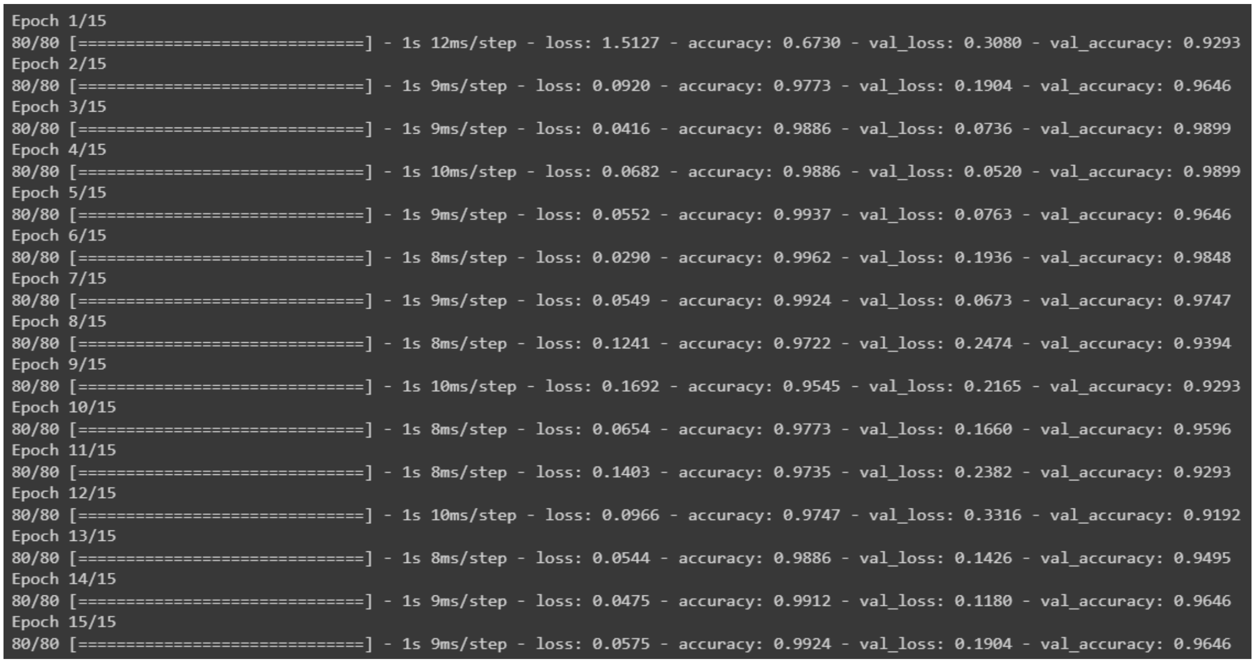


```
Epoch 1/15
80/80 [==============================] - 2s 15ms/step - loss: 1.5270 - accuracy: 0.6477 - val_loss: 0.3094 - val_accuracy: 0.9343
Epoch 2/15
80/80 [==============================] - 1s 9ms/step - loss: 0.1052 - accuracy: 0.9684 - val_loss: 0.1019 - val_accuracy: 0.9697
Epoch 3/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0428 - accuracy: 0.9861 - val_loss: 0.1060 - val_accuracy: 0.9747
Epoch 4/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0491 - accuracy: 0.9899 - val_loss: 0.0935 - val_accuracy: 0.9798
Epoch 5/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0190 - accuracy: 0.9937 - val_loss: 0.0412 - val_accuracy: 0.9899
Epoch 6/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.0295 - val_accuracy: 0.9949
Epoch 7/15
80/80 [==============================] - 1s 9ms/step - loss: 9.3466e-04 - accuracy: 1.0000 - val_loss: 0.0256 - val_accuracy: 0.9949
Epoch 8/15
80/80 [==============================] - 1s 10ms/step - loss: 3.7088e-04 - accuracy: 1.0000 - val_loss: 0.0246 - val_accuracy: 0.9949
Epoch 9/15
80/80 [==============================] - 1s 9ms/step - loss: 2.9325e-04 - accuracy: 1.0000 - val_loss: 0.0236 - val_accuracy: 0.9949
Epoch 10/15
80/80 [==============================] - 1s 9ms/step - loss: 2.4210e-04 - accuracy: 1.0000 - val_loss: 0.0229 - val_accuracy: 0.9949
Epoch 11/15
80/80 [==============================] - 1s 9ms/step - loss: 2.0504e-04 - accuracy: 1.0000 - val_loss: 0.0224 - val_accuracy: 0.9949
Epoch 12/15
80/80 [==============================] - 1s 10ms/step - loss: 1.8081e-04 - accuracy: 1.0000 - val_loss: 0.0226 - val_accuracy: 0.9949
Epoch 13/15
80/80 [==============================] - 1s 10ms/step - loss: 1.6147e-04 - accuracy: 1.0000 - val_loss: 0.0213 - val_accuracy: 0.9949
Epoch 14/15
80/80 [==============================] - 1s 8ms/step - loss: 1.4288e-04 - accuracy: 1.0000 - val_loss: 0.0210 - val_accuracy: 0.9949
Epoch 15/15
80/80 [==============================] - 1s 9ms/step - loss: 1.2680e-04 - accuracy: 1.0000 - val_loss: 0.0208 - val_accuracy: 0.9949
<keras.callbacks.History at 0x7f4be7a5dc50>
```

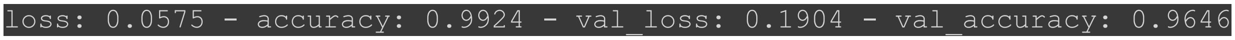


```
Epoch 1/15
80/80 [==============================] - 2s 19ms/step - loss: 1.5714 - accuracy: 0.6604 - val_loss: 0.3821 - val_accuracy: 0.9141
Epoch 2/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0903 - accuracy: 0.9798 - val_loss: 0.1124 - val_accuracy: 0.9747
Epoch 3/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0346 - accuracy: 0.9937 - val_loss: 0.0551 - val_accuracy: 0.9848
Epoch 4/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0073 - accuracy: 0.9987 - val_loss: 0.0518 - val_accuracy: 0.9848
Epoch 5/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0106 - accuracy: 0.9975 - val_loss: 0.0577 - val_accuracy: 0.9798
Epoch 6/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.0479 - val_accuracy: 0.9899
Epoch 7/15
80/80 [==============================] - 1s 9ms/step - loss: 5.4985e-04 - accuracy: 1.0000 - val_loss: 0.0442 - val_accuracy: 0.9949
Epoch 8/15
80/80 [==============================] - 1s 9ms/step - loss: 3.8650e-04 - accuracy: 1.0000 - val_loss: 0.0420 - val_accuracy: 0.9949
Epoch 9/15
80/80 [==============================] - 1s 10ms/step - loss: 3.1817e-04 - accuracy: 1.0000 - val_loss: 0.0407 - val_accuracy: 0.9949
Epoch 10/15
80/80 [==============================] - 1s 9ms/step - loss: 2.6795e-04 - accuracy: 1.0000 - val_loss: 0.0393 - val_accuracy: 0.9949
Epoch 11/15
80/80 [==============================] - 1s 10ms/step - loss: 2.3187e-04 - accuracy: 1.0000 - val_loss: 0.0382 - val_accuracy: 0.9949
Epoch 12/15
80/80 [==============================] - 1s 10ms/step - loss: 2.0029e-04 - accuracy: 1.0000 - val_loss: 0.0372 - val_accuracy: 0.9949
Epoch 13/15
80/80 [==============================] - 1s 10ms/step - loss: 1.7784e-04 - accuracy: 1.0000 - val_loss: 0.0363 - val_accuracy: 0.9949
Epoch 14/15
80/80 [==============================] - 1s 9ms/step - loss: 1.5887e-04 - accuracy: 1.0000 - val_loss: 0.0356 - val_accuracy: 0.9949
Epoch 15/15
80/80 [==============================] - 1s 9ms/step - loss: 1.4143e-04 - accuracy: 1.0000 - val_loss: 0.0350 - val_accuracy: 0.9949
<keras.callbacks.History at 0x7f4be916c790>
```

3- Dropout= 0.1



```
Epoch 1/15
80/80 [==============================] - 1s 12ms/step - loss: 1.5127 - accuracy: 0.6730 - val_loss: 0.3080 - val_accuracy: 0.9293
Epoch 2/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0920 - accuracy: 0.9773 - val_loss: 0.1904 - val_accuracy: 0.9646
Epoch 3/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0416 - accuracy: 0.9886 - val_loss: 0.0736 - val_accuracy: 0.9899
Epoch 4/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0682 - accuracy: 0.9886 - val_loss: 0.0520 - val_accuracy: 0.9899
Epoch 5/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0552 - accuracy: 0.9937 - val_loss: 0.0763 - val_accuracy: 0.9646
Epoch 6/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0290 - accuracy: 0.9962 - val_loss: 0.1936 - val_accuracy: 0.9848
Epoch 7/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0549 - accuracy: 0.9924 - val_loss: 0.0673 - val_accuracy: 0.9747
Epoch 8/15
80/80 [==============================] - 1s 8ms/step - loss: 0.1241 - accuracy: 0.9722 - val_loss: 0.2474 - val_accuracy: 0.9394
Epoch 9/15
80/80 [==============================] - 1s 10ms/step - loss: 0.1692 - accuracy: 0.9545 - val_loss: 0.2165 - val_accuracy: 0.9293
Epoch 10/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0654 - accuracy: 0.9773 - val_loss: 0.1660 - val_accuracy: 0.9596
Epoch 11/15
80/80 [==============================] - 1s 8ms/step - loss: 0.1403 - accuracy: 0.9735 - val_loss: 0.2382 - val_accuracy: 0.9293
Epoch 12/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0966 - accuracy: 0.9747 - val_loss: 0.3316 - val_accuracy: 0.9192
Epoch 13/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0544 - accuracy: 0.9886 - val_loss: 0.1426 - val_accuracy: 0.9495
Epoch 14/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0475 - accuracy: 0.9912 - val_loss: 0.1180 - val_accuracy: 0.9646
Epoch 15/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0575 - accuracy: 0.9924 - val_loss: 0.1904 - val_accuracy: 0.9646
```

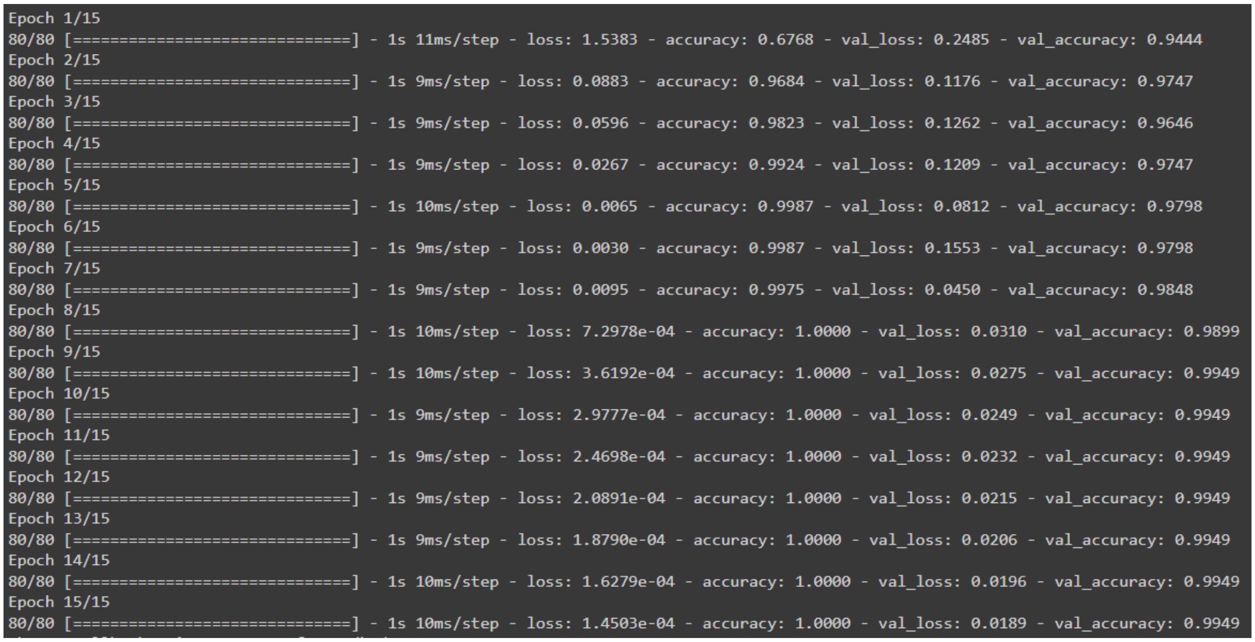


```
loss: 0.0575 - accuracy: 0.9924 - val_loss: 0.1904 - val_accuracy: 0.9646
```

The best Dropout out rate was 0.2 so we stick with it.

Next we go with Optimizers. We chose the next values:

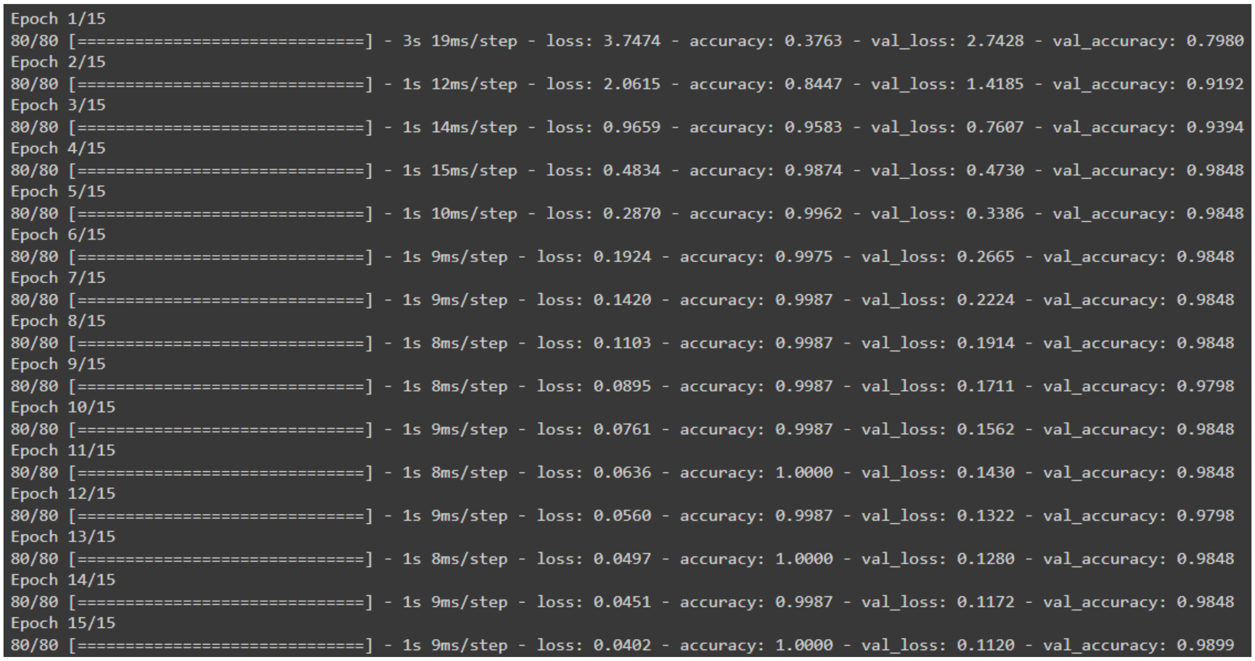
1- Adam



```
Epoch 1/15
80/80 [==============================] - 1s 11ms/step - loss: 1.5383 - accuracy: 0.6768 - val_loss: 0.2485 - val_accuracy: 0.9444
Epoch 2/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0883 - accuracy: 0.9684 - val_loss: 0.1176 - val_accuracy: 0.9747
Epoch 3/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0596 - accuracy: 0.9823 - val_loss: 0.1262 - val_accuracy: 0.9646
Epoch 4/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0267 - accuracy: 0.9924 - val_loss: 0.1209 - val_accuracy: 0.9747
Epoch 5/15
80/80 [==============================] - 1s 10ms/step - loss: 0.0065 - accuracy: 0.9987 - val_loss: 0.0812 - val_accuracy: 0.9798
Epoch 6/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0030 - accuracy: 0.9987 - val_loss: 0.1553 - val_accuracy: 0.9798
Epoch 7/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0095 - accuracy: 0.9975 - val_loss: 0.0450 - val_accuracy: 0.9848
Epoch 8/15
80/80 [==============================] - 1s 10ms/step - loss: 7.2978e-04 - accuracy: 1.0000 - val_loss: 0.0310 - val_accuracy: 0.9899
Epoch 9/15
80/80 [==============================] - 1s 10ms/step - loss: 3.6192e-04 - accuracy: 1.0000 - val_loss: 0.0275 - val_accuracy: 0.9949
Epoch 10/15
80/80 [==============================] - 1s 9ms/step - loss: 2.9777e-04 - accuracy: 1.0000 - val_loss: 0.0249 - val_accuracy: 0.9949
Epoch 11/15
80/80 [==============================] - 1s 9ms/step - loss: 2.4698e-04 - accuracy: 1.0000 - val_loss: 0.0232 - val_accuracy: 0.9949
Epoch 12/15
80/80 [==============================] - 1s 9ms/step - loss: 2.0891e-04 - accuracy: 1.0000 - val_loss: 0.0215 - val_accuracy: 0.9949
Epoch 13/15
80/80 [==============================] - 1s 9ms/step - loss: 1.8790e-04 - accuracy: 1.0000 - val_loss: 0.0206 - val_accuracy: 0.9949
Epoch 14/15
80/80 [==============================] - 1s 10ms/step - loss: 1.6279e-04 - accuracy: 1.0000 - val_loss: 0.0196 - val_accuracy: 0.9949
Epoch 15/15
80/80 [==============================] - 1s 10ms/step - loss: 1.4503e-04 - accuracy: 1.0000 - val_loss: 0.0189 - val_accuracy: 0.9949
```

```
loss: 1.4503e-04 -accuracy: 1.0000 -val_loss: 0.0189 -val_accuracy: 0.9949
```

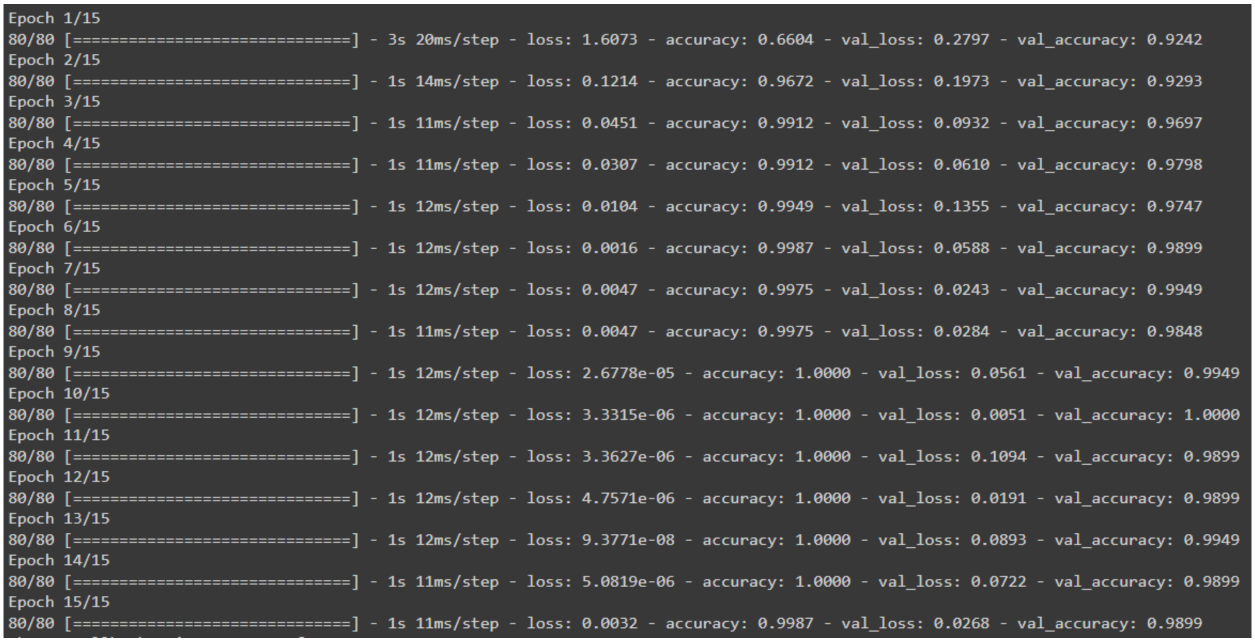
2- SGD



```
Epoch 1/15
80/80 [==============================] - 3s 19ms/step - loss: 3.7474 - accuracy: 0.3763 - val_loss: 2.7428 - val_accuracy: 0.7980
Epoch 2/15
80/80 [==============================] - 1s 12ms/step - loss: 2.0615 - accuracy: 0.8447 - val_loss: 1.4185 - val_accuracy: 0.9192
Epoch 3/15
80/80 [==============================] - 1s 14ms/step - loss: 0.9659 - accuracy: 0.9583 - val_loss: 0.7607 - val_accuracy: 0.9394
Epoch 4/15
80/80 [==============================] - 1s 15ms/step - loss: 0.4834 - accuracy: 0.9874 - val_loss: 0.4730 - val_accuracy: 0.9848
Epoch 5/15
80/80 [==============================] - 1s 10ms/step - loss: 0.2870 - accuracy: 0.9962 - val_loss: 0.3386 - val_accuracy: 0.9848
Epoch 6/15
80/80 [==============================] - 1s 9ms/step - loss: 0.1924 - accuracy: 0.9975 - val_loss: 0.2665 - val_accuracy: 0.9848
Epoch 7/15
80/80 [==============================] - 1s 9ms/step - loss: 0.1420 - accuracy: 0.9987 - val_loss: 0.2224 - val_accuracy: 0.9848
Epoch 8/15
80/80 [==============================] - 1s 8ms/step - loss: 0.1103 - accuracy: 0.9987 - val_loss: 0.1914 - val_accuracy: 0.9848
Epoch 9/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0895 - accuracy: 0.9987 - val_loss: 0.1711 - val_accuracy: 0.9798
Epoch 10/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0761 - accuracy: 0.9987 - val_loss: 0.1562 - val_accuracy: 0.9848
Epoch 11/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0636 - accuracy: 1.0000 - val_loss: 0.1430 - val_accuracy: 0.9848
Epoch 12/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0560 - accuracy: 0.9987 - val_loss: 0.1322 - val_accuracy: 0.9798
Epoch 13/15
80/80 [==============================] - 1s 8ms/step - loss: 0.0497 - accuracy: 1.0000 - val_loss: 0.1280 - val_accuracy: 0.9848
Epoch 14/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0451 - accuracy: 0.9987 - val_loss: 0.1172 - val_accuracy: 0.9848
Epoch 15/15
80/80 [==============================] - 1s 9ms/step - loss: 0.0402 - accuracy: 1.0000 - val_loss: 0.1120 - val_accuracy: 0.9899
```

```
loss: 0.0402 - accuracy: 1.0000 - val_loss: 0.1120 - val_accuracy: 0.9899
```

3- RMSProp



```
Epoch 1/15
80/80 [==============================] - 3s 20ms/step - loss: 1.6073 - accuracy: 0.6604 - val_loss: 0.2797 - val_accuracy: 0.9242
Epoch 2/15
80/80 [==============================] - 1s 14ms/step - loss: 0.1214 - accuracy: 0.9672 - val_loss: 0.1973 - val_accuracy: 0.9293
Epoch 3/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0451 - accuracy: 0.9912 - val_loss: 0.0932 - val_accuracy: 0.9697
Epoch 4/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0307 - accuracy: 0.9912 - val_loss: 0.0610 - val_accuracy: 0.9798
Epoch 5/15
80/80 [==============================] - 1s 12ms/step - loss: 0.0104 - accuracy: 0.9949 - val_loss: 0.1355 - val_accuracy: 0.9747
Epoch 6/15
80/80 [==============================] - 1s 12ms/step - loss: 0.0016 - accuracy: 0.9987 - val_loss: 0.0588 - val_accuracy: 0.9899
Epoch 7/15
80/80 [==============================] - 1s 12ms/step - loss: 0.0047 - accuracy: 0.9975 - val_loss: 0.0243 - val_accuracy: 0.9949
Epoch 8/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0047 - accuracy: 0.9975 - val_loss: 0.0284 - val_accuracy: 0.9848
Epoch 9/15
80/80 [==============================] - 1s 12ms/step - loss: 2.6778e-05 - accuracy: 1.0000 - val_loss: 0.0561 - val_accuracy: 0.9949
Epoch 10/15
80/80 [==============================] - 1s 12ms/step - loss: 3.3315e-06 - accuracy: 1.0000 - val_loss: 0.0051 - val_accuracy: 1.0000
Epoch 11/15
80/80 [==============================] - 1s 12ms/step - loss: 3.3627e-06 - accuracy: 1.0000 - val_loss: 0.1094 - val_accuracy: 0.9899
Epoch 12/15
80/80 [==============================] - 1s 12ms/step - loss: 4.7571e-06 - accuracy: 1.0000 - val_loss: 0.0191 - val_accuracy: 0.9899
Epoch 13/15
80/80 [==============================] - 1s 12ms/step - loss: 9.3771e-08 - accuracy: 1.0000 - val_loss: 0.0893 - val_accuracy: 0.9949
Epoch 14/15
80/80 [==============================] - 1s 11ms/step - loss: 5.0819e-06 - accuracy: 1.0000 - val_loss: 0.0722 - val_accuracy: 0.9899
Epoch 15/15
80/80 [==============================] - 1s 11ms/step - loss: 0.0032 - accuracy: 0.9987 - val_loss: 0.0268 - val_accuracy: 0.9899
```

```
loss: 0.0032 -accuracy: 0.9987 -val_loss: 0.0268 -val_accuracy: 0.9899
```

The best Optimizer was Adam so we stick with it.