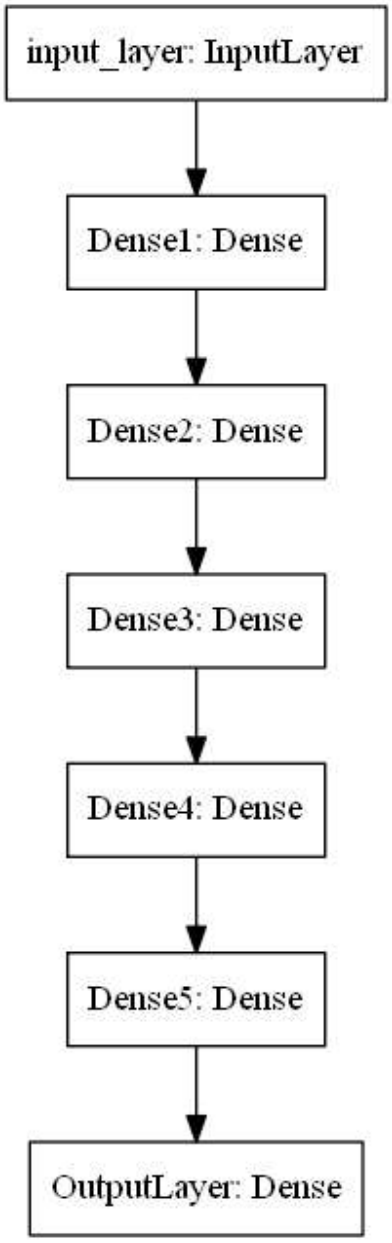
1. Download the data from here (https://drive.google.com/file/d/15dCNcmKskcFVjs7R0ElQkR61Ex53uJpM/view?usp=sharing)

2. Code the model to classify data like below image



3. Write your own callback function, that has to print the micro F1 score and AUC score after each epoch.

4. Save your model at every epoch if your validation accuracy is improved from previous epoch.

5. you have to decay learning based on below conditions
        Cond1. If your validation accuracy at that epoch is less than previous epoch accuracy, you have to decrese the
                learning rate by 10%.
        Cond2. For every 3rd epoch, decay your learning rate by 5%.

6. If you are getting any NaN values(either weigths or loss) while traini
ng, you have to terminate your training.

7. You have to stop the training if your validation accuracy is not incre
ased in last 2 epochs.

8. Use tensorboard for every model and analyse your gradients. (you need
 to upload the screenshots for each model for evaluation)

9. use cross entropy as loss function

10. Try the architecture params as given below.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import datetime

from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score

from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model

from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ReduceLROnPlateau

%load_ext tensorboard
```

Loading Data

In [2]:
```
from google.colab import files
data = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving data_cb.csv to data_cb (6).csv
```

In [3]:
```
df = pd.read_csv('data_cb.csv')
df.head()
```

Out[3]:

	f1	f2	label
0	0.450564	1.074305	0.0
1	0.085632	0.967682	0.0
2	0.117326	0.971521	1.0
3	0.982179	-0.380408	0.0
4	-0.720352	0.955850	0.0

In [4]:
```
df.describe()
```

Out[4]:

	f1	f2	label
count	20000.000000	20000.000000	20000.000000
mean	0.000630	-0.000745	0.500000
std	0.671165	0.674704	0.500013
min	-1.649781	-1.600645	0.000000
25%	-0.589878	-0.596424	0.000000
50%	0.001795	-0.003113	0.500000
75%	0.586631	0.597803	1.000000
max	1.629722	1.584291	1.000000

In [5]:
```
df.shape # Shape of the whole dataset
```

Out[5]:
```
(20000, 3)
```

In [6]:
```
# Arranging the dataset
X = df.iloc[:,0:2]
y = df.iloc[:,2:]
print(X.shape)
print(y.shape)
```

```
(20000, 2)
(20000, 1)
```

Splitting data into Train and Test

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[7]:
```
((16000, 2), (4000, 2), (16000, 1), (4000, 1))
```

In [8]:
```
class LossHistory(tf.keras.callbacks.Callback):

    def on_train_begin(self, logs={}):
        self.f1_score = []

    def on_epoch_end(self, epoch, logs={}):
        pred_y = (np.asarray(self.model.predict(X_test))).round()
        f1_ = f1_score(y_test, pred_y)
        self.f1_score.append(f1_)

        auc_ = roc_auc_score(y_test, pred_y)
        print('- F1_Score: ',f1_, '- AUC: ', auc_)
```

In [9]:
```
history_own = LossHistory()
```

In [10]:
```
class TerminateNaN(tf.keras.callbacks.Callback):

    def on_epoch_end(self, epoch, logs={}):
        loss = logs.get('loss')
        if loss is not None:
            if np.isnan(loss) or np.isinf(loss):
                print("Invalid loss and terminated at epoch {}".format(epoch))
                self.model.stop_training = True
```

In [11]:
```
terminate = TerminateNaN()
```

In [12]:
```
def changeLearningRate(epoch, lr):
    if (epoch+1) % 3 == 0 :
        return lr * (1-0.05)**(epoch+1)
    return lr
```

Model-1

1. Use tanh as an activation for every layer except output layer.

2. use SGD with momentum as optimizer.

3. use RandomUniform(0,1) as initilizer.

4. Analyze your output and training process.

In [13]:
```
# Clear any logs from previous runs
!rm -rf ./logs/
```

In [14]:
```
# Input layer
input_layer = Input(shape=(2,))

# Dense hidden layer
layer1 = Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.Rando

# output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.Ra

# Creating a model
model = Model(inputs=input_layer,outputs=output)


#create a call back list
filepath="model_save/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_acc', verbose=1, sav
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.1, patience=1, verbose=
lrschedule = LearningRateScheduler(changeLearningRate, verbose=0.1)
earlystop = EarlyStopping(monitor='val_acc', patience=2, verbose=0.1)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_



# here we are creating a list with all the callbacks we want
callback_list = [history_own, terminate, checkpoint, reduce_lr, lrschedule, early

optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.07, nesterov=Fal

model.compile(optimizer, loss='BinaryCrossentropy',metrics=['acc'])
model.fit(X_train, y_train, validation_split=0.1, epochs=15, validation_data=(X_t
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `Ten
sorBoard` Callback.


Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/15
  1/900 [..............................] - ETA: 7:08 - loss: 1.0905 - acc: 0.37
50

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0013s vs `on_train_batch_end` time: 0.0029s). Check y
our callbacks.

895/900 [============================>.] - ETA: 0s - loss: 0.7343 - acc: 0.5038
- F1_Score:  0.5339409930438954 - AUC:  0.5146719384270126

Epoch 1: val_acc improved from -inf to 0.51500, saving model to model_save/weig
hts-01-0.5150.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.7341 - acc: 0.
5040 - val_loss: 0.6945 - val_acc: 0.5150 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/15
876/900 [============================>.] - ETA: 0s - loss: 0.6924 - acc: 0.5337
- F1_Score:  0.5375513161072204 - AUC:  0.52160950036953

Epoch 2: val_acc did not improve from 0.51500

Epoch 2: ReduceLROnPlateau reducing learning rate to 0.0009999999776482583.
900/900 [==============================] - 2s 2ms/step - loss: 0.6924 - acc: 0.
```

```
5328 - val_loss: 0.6920 - val_acc: 0.5106 - lr: 1.0000e-03

Epoch 3: LearningRateScheduler setting learning rate to 0.0008573749409115407.
Epoch 3/15
872/900 [============================>.] - ETA: 0s - loss: 0.6911 - acc: 0.5243
- F1_Score:  0.5625866050808315 - AUC:  0.5272852101020182

Epoch 3: val_acc improved from 0.51500 to 0.52062, saving model to model_save/w
eights-03-0.5206.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6912 - acc: 0.
5238 - val_loss: 0.6919 - val_acc: 0.5206 - lr: 8.5737e-04

Epoch 4: LearningRateScheduler setting learning rate to 0.0008573749219067395.
Epoch 4/15
883/900 [============================>.] - ETA: 0s - loss: 0.6911 - acc: 0.5311
- F1_Score:  0.5776775648252537 - AUC:  0.5327719045242665

Epoch 4: val_acc improved from 0.52062 to 0.52375, saving model to model_save/w
eights-04-0.5238.hdf5
900/900 [==============================] - 2s 3ms/step - loss: 0.6910 - acc: 0.
5308 - val_loss: 0.6918 - val_acc: 0.5238 - lr: 8.5737e-04

Epoch 5: LearningRateScheduler setting learning rate to 0.0008573749219067395.
Epoch 5/15
871/900 [============================>.] - ETA: 0s - loss: 0.6909 - acc: 0.5349
- F1_Score:  0.5820058335203051 - AUC:  0.535321611050495

Epoch 5: val_acc did not improve from 0.52375

Epoch 5: ReduceLROnPlateau reducing learning rate to 8.573749219067396e-05.
900/900 [==============================] - 2s 2ms/step - loss: 0.6909 - acc: 0.
5350 - val_loss: 0.6917 - val_acc: 0.5219 - lr: 8.5737e-05

Epoch 6: LearningRateScheduler setting learning rate to 6.302493416218918e-05.
Epoch 6/15
888/900 [============================>.] - ETA: 0s - loss: 0.6908 - acc: 0.5343
- F1_Score:  0.5825112107623318 - AUC:  0.5355783818489297

Epoch 6: val_acc did not improve from 0.52375

Epoch 6: ReduceLROnPlateau reducing learning rate to 6.302493420662359e-06.
900/900 [==============================] - 2s 2ms/step - loss: 0.6908 - acc: 0.
5347 - val_loss: 0.6917 - val_acc: 0.5213 - lr: 6.3025e-06
Epoch 6: early stopping
```

Out[14]: <keras.callbacks.History at 0x7feb4cefc650>

In [15]:
```
%tensorboard --logdir logs/fit
```

```
<IPython.core.display.Javascript object>
```

Model-2

1. Use relu as an activation for every layer except output layer.

2. use SGD with momentum as optimizer.

3. use RandomUniform(0,1) as initilizer.

4. Analyze your output and training process.

In [16]:
```
# Clear any logs from previous runs
!rm -rf ./logs/
```

In [17]:
```
# Input Layer
input_layer = Input(shape=(2,))

# Dense hidden layer
layer1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.Rando

# output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.Ra

# Creating a model
model = Model(inputs=input_layer,outputs=output)


#create a call back list
filepath="model_save/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_acc', verbose=1, sav
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.1, patience=1, verbose=
lrschedule = LearningRateScheduler(changeLearningRate, verbose=0.1)
earlystop = EarlyStopping(monitor='val_acc', patience=2, verbose=0.1)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_


# here we are creating a list with all the callbacks we want
callback_list = [history_own, terminate, checkpoint, reduce_lr, lrschedule, early

optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.4, nesterov=Fals

model.compile(optimizer, loss='BinaryCrossentropy',metrics=['acc'])
model.fit(X_train, y_train, validation_split=0.1, epochs=15, validation_data=(X_t
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `Ten
sorBoard` Callback.


Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/15
  1/900 [..............................] - ETA: 5:27 - loss: 0.7618 - acc: 0.43
75

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0016s vs `on_train_batch_end` time: 0.0023s). Check y
our callbacks.

881/900 [============================>.] - ETA: 0s - loss: 0.7007 - acc: 0.4884
- F1_Score:  0.2345773038842346 - AUC:  0.49444855033257695

Epoch 1: val_acc improved from -inf to 0.48313, saving model to model_save/weig
hts-01-0.4831.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.7004 - acc: 0.
4891 - val_loss: 0.6939 - val_acc: 0.4831 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/15
883/900 [============================>.] - ETA: 0s - loss: 0.6918 - acc: 0.5038
- F1_Score:  0.5825838999780654 - AUC:  0.5255503195758856

Epoch 2: val_acc improved from 0.48313 to 0.51125, saving model to model_save/w
eights-02-0.5113.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6917 - acc: 0.
5039 - val_loss: 0.6912 - val_acc: 0.5113 - lr: 0.0100
```

```
Epoch 3: LearningRateScheduler setting learning rate to 0.008573749808361753.
Epoch 3/15
887/900 [============================>.] - ETA: 0s - loss: 0.6899 - acc: 0.5130
- F1_Score:  0.5834434169969176 - AUC:  0.528262289245429

Epoch 3: val_acc did not improve from 0.51125

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0008573750033974648.
900/900 [==============================] - 2s 2ms/step - loss: 0.6900 - acc: 0.
5125 - val_loss: 0.6905 - val_acc: 0.5069 - lr: 8.5737e-04

Epoch 4: LearningRateScheduler setting learning rate to 0.0008573749801144004.
Epoch 4/15
886/900 [============================>.] - ETA: 0s - loss: 0.6892 - acc: 0.5178
- F1_Score:  0.5826910372164721 - AUC:  0.527509978308243

Epoch 4: val_acc did not improve from 0.51125

Epoch 4: ReduceLROnPlateau reducing learning rate to 8.573749801144004e-05.
900/900 [==============================] - 2s 2ms/step - loss: 0.6893 - acc: 0.
5164 - val_loss: 0.6905 - val_acc: 0.5075 - lr: 8.5737e-05
Epoch 4: early stopping
```

Out[17]:
```
<keras.callbacks.History at 0x7feb4ce90e10>
```

In [18]:
```
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6006 (pid 4421), started 0:00:44 ago. (Use '!kill 4
421' to kill it.)

<IPython.core.display.Javascript object>
```

Model-3

1. Use relu as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use he_uniform() as initilizer.
4. Analyze your output and training process.

In [19]:
```
# Clear any logs from previous runs
!rm -rf ./logs/
```

In [20]:

```
# Input Layer
input_layer = Input(shape=(2,))

# Dense hidden layer
layer1 = Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.he_un

# output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_

# Creating a model
model = Model(inputs=input_layer,outputs=output)


#create a call back list
filepath="model_save/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_acc', verbose=1, sav
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.1, patience=1, verbose=
lrschedule = LearningRateScheduler(changeLearningRate, verbose=0.1)
earlystop = EarlyStopping(monitor='val_acc', patience=2, verbose=0.1)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_


# here we are creating a list with all the callbacks we want
callback_list = [history_own, terminate, checkpoint, reduce_lr, lrschedule, early

optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.4, nesterov=Fals

model.compile(optimizer, loss='BinaryCrossentropy',metrics=['acc'])
model.fit(X_train, y_train, validation_split=0.1, epochs=15, validation_data=(X_t
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `Ten
sorBoard` Callback.


Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/15
  1/900 [..............................] - ETA: 5:27 - loss: 0.7872 - acc: 0.43
75

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0012s vs `on_train_batch_end` time: 0.0026s). Check y
our callbacks.

874/900 [============================>.] - ETA: 0s - loss: 0.6904 - acc: 0.53
31- F1_Score:  0.5263715875960774 - AUC:  0.552784025506066

Epoch 1: val_acc improved from -inf to 0.55063, saving model to model_save/we
ights-01-0.5506.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6901 - acc:
0.5328 - val_loss: 0.6840 - val_acc: 0.5506 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/15
900/900 [==============================] - ETA: 0s - loss: 0.6785 - acc: 0.56
38- F1_Score:  0.5578544061302683 - AUC:  0.5671046854795239

Epoch 2: val_acc improved from 0.55063 to 0.55313, saving model to model_sav
e/weights-02-0.5531.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6785 - acc:
0.5638 - val_loss: 0.6806 - val_acc: 0.5531 - lr: 0.0100
```

```
Epoch 3: LearningRateScheduler setting learning rate to 0.008573749808361753.
Epoch 3/15
869/900 [===========================>..] - ETA: 0s - loss: 0.6755 - acc: 0.57
35- F1_Score:  0.5587786259541985 - AUC:  0.5663883774585741

Epoch 3: val_acc improved from 0.55313 to 0.55875, saving model to model_sav
e/weights-03-0.5587.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6755 - acc:
0.5728 - val_loss: 0.6797 - val_acc: 0.5587 - lr: 0.0086

Epoch 4: LearningRateScheduler setting learning rate to 0.008573750033974648.
Epoch 4/15
890/900 [============================>.] - ETA: 0s - loss: 0.6740 - acc: 0.57
40- F1_Score:  0.5760277365032193 - AUC:  0.572131842679257

Epoch 4: val_acc improved from 0.55875 to 0.56437, saving model to model_sav
e/weights-04-0.5644.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6742 - acc:
0.5740 - val_loss: 0.6789 - val_acc: 0.5644 - lr: 0.0086

Epoch 5: LearningRateScheduler setting learning rate to 0.008573750033974648.
Epoch 5/15
888/900 [============================>.] - ETA: 0s - loss: 0.6728 - acc: 0.57
75- F1_Score:  0.5898862811517058 - AUC:  0.5765959542722962

Epoch 5: val_acc did not improve from 0.56437

Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0008573750033974648.
900/900 [==============================] - 2s 2ms/step - loss: 0.6731 - acc:
0.5767 - val_loss: 0.6782 - val_acc: 0.5631 - lr: 8.5737e-04

Epoch 6: LearningRateScheduler setting learning rate to 0.000630249395106866
2.
Epoch 6/15
892/900 [============================>.] - ETA: 0s - loss: 0.6724 - acc: 0.58
00- F1_Score:  0.5872630043753038 - AUC:  0.5758031400543444

Epoch 6: val_acc improved from 0.56437 to 0.56687, saving model to model_sav
e/weights-06-0.5669.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6724 - acc:
0.5801 - val_loss: 0.6781 - val_acc: 0.5669 - lr: 6.3025e-04

Epoch 7: LearningRateScheduler setting learning rate to 0.000630249385721981
5.
Epoch 7/15
886/900 [============================>.] - ETA: 0s - loss: 0.6722 - acc: 0.57
84- F1_Score:  0.5853896897141461 - AUC:  0.5760059064784248

Epoch 7: val_acc improved from 0.56687 to 0.56813, saving model to model_sav
e/weights-07-0.5681.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6723 - acc:
0.5779 - val_loss: 0.6781 - val_acc: 0.5681 - lr: 6.3025e-04

Epoch 8: LearningRateScheduler setting learning rate to 0.000630249385721981
5.
Epoch 8/15
885/900 [============================>.] - ETA: 0s - loss: 0.6724 - acc: 0.57
61- F1_Score:  0.5827003185493752 - AUC:  0.5744787827814053

Epoch 8: val_acc did not improve from 0.56813

Epoch 8: ReduceLROnPlateau reducing learning rate to 6.302493857219815e-05.
900/900 [==============================] - 2s 2ms/step - loss: 0.6722 - acc:
```

```
0.5773 - val_loss: 0.6780 - val_acc: 0.5650 - lr: 6.3025e-05

Epoch 9: LearningRateScheduler setting learning rate to 3.972143216732483e-0
5.
Epoch 9/15
872/900 [============================>.] - ETA: 0s - loss: 0.6722 - acc: 0.57
72- F1_Score:  0.5829860259867614 - AUC:  0.5749743229201565

Epoch 9: val_acc did not improve from 0.56813

Epoch 9: ReduceLROnPlateau reducing learning rate to 3.9721431676298385e-06.
900/900 [==============================] - 2s 2ms/step - loss: 0.6722 - acc:
0.5773 - val_loss: 0.6780 - val_acc: 0.5650 - lr: 3.9721e-06
Epoch 9: early stopping
```

Out[20]: <keras.callbacks.History at 0x7feb4ce1ad10>

In [21]:
```
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6006 (pid 4421), started 0:01:36 ago. (Use '!kill 4
421' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

Model-4

1. Try with any values to get better accuracy/f1 score.

In [22]:
```
# Clear any logs from previous runs
!rm -rf ./logs/
```

In [23]:
```
# Input layer
input_layer = Input(shape=(2,))

# Dense hidden layer
layer1 = Dense(5,activation='elu',kernel_initializer=tf.keras.initializers.he_uni

# output layer
output = Dense(1,activation='sigmoid',kernel_initializer=tf.keras.initializers.he_

# Creating a model
model = Model(inputs=input_layer,outputs=output)


#create a call back list
filepath="model_save/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_acc', verbose=1, sav
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.1, patience=1, verbose=
lrschedule = LearningRateScheduler(changeLearningRate, verbose=0.1)
earlystop = EarlyStopping(monitor='val_acc', patience=2, verbose=0.1)
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_


# here we are creating a list with all the callbacks we want
callback_list = [history_own, terminate, checkpoint, reduce_lr, lrschedule, early

optimizer = tf.keras.optimizers.Adam(0.01)


model.compile(optimizer, loss='BinaryCrossentropy',metrics=['acc'])
model.fit(X_train, y_train, validation_split=0.1, epochs=15, validation_data=(X_t
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `Ten
sorBoard` Callback.


Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/15
  1/900 [..............................] - ETA: 6:00 - loss: 0.8019 - acc: 0.50
00

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0014s vs `on_train_batch_end` time: 0.0020s). Check y
our callbacks.

886/900 [============================>.] - ETA: 0s - loss: 0.6804 - acc: 0.5634
- F1_Score:  0.6696606786427146 - AUC:  0.6690721948477827

Epoch 1: val_acc improved from -inf to 0.65375, saving model to model_save/weig
hts-01-0.6538.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6794 - acc: 0.
5653 - val_loss: 0.6281 - val_acc: 0.6538 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/15
899/900 [============================>.] - ETA: 0s - loss: 0.6063 - acc: 0.6687
- F1_Score:  0.6383783783783783 - AUC:  0.6648788551872702

Epoch 2: val_acc improved from 0.65375 to 0.66312, saving model to model_save/w
eights-02-0.6631.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.6063 - acc: 0.
6686 - val_loss: 0.6225 - val_acc: 0.6631 - lr: 0.0100
```

```
Epoch 3: LearningRateScheduler setting learning rate to 0.008573749808361753.
Epoch 3/15
888/900 [============================>.] - ETA: 0s - loss: 0.6015 - acc: 0.6707
- F1_Score:  0.6327744726879394 - AUC:  0.659873949789933

Epoch 3: val_acc did not improve from 0.66312

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0008573750033974648.
900/900 [==============================] - 2s 2ms/step - loss: 0.6014 - acc: 0.
6709 - val_loss: 0.6212 - val_acc: 0.6562 - lr: 8.5737e-04

Epoch 4: LearningRateScheduler setting learning rate to 0.0008573749801144004.
Epoch 4/15
885/900 [============================>.] - ETA: 0s - loss: 0.5987 - acc: 0.6727
- F1_Score:  0.6573033707865169 - AUC:  0.6643648135498975

Epoch 4: val_acc improved from 0.66312 to 0.66625, saving model to model_save/w
eights-04-0.6662.hdf5
900/900 [==============================] - 2s 2ms/step - loss: 0.5985 - acc: 0.
6731 - val_loss: 0.6168 - val_acc: 0.6662 - lr: 8.5737e-04

Epoch 5: LearningRateScheduler setting learning rate to 0.0008573749801144004.
Epoch 5/15
868/900 [===========================>..] - ETA: 0s - loss: 0.5990 - acc: 0.6730
- F1_Score:  0.6555952989269289 - AUC:  0.6628601916755257

Epoch 5: val_acc did not improve from 0.66625

Epoch 5: ReduceLROnPlateau reducing learning rate to 8.573749801144004e-05.
900/900 [==============================] - 2s 2ms/step - loss: 0.5980 - acc: 0.
6743 - val_loss: 0.6168 - val_acc: 0.6637 - lr: 8.5737e-05

Epoch 6: LearningRateScheduler setting learning rate to 6.302493951068661e-05.
Epoch 6/15
869/900 [===========================>..] - ETA: 0s - loss: 0.5972 - acc: 0.6750
- F1_Score:  0.6562978072412035 - AUC:  0.6628781931336437

Epoch 6: val_acc did not improve from 0.66625

Epoch 6: ReduceLROnPlateau reducing learning rate to 6.30249414825812e-06.
900/900 [==============================] - 2s 2ms/step - loss: 0.5976 - acc: 0.
6750 - val_loss: 0.6166 - val_acc: 0.6625 - lr: 6.3025e-06
Epoch 6: early stopping
```

Out[23]: <keras.callbacks.History at 0x7feb48727790>

In [24]:
```
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6006 (pid 4421), started 0:03:04 ago. (Use '!kill 4
421' to kill it.)

<IPython.core.display.Javascript object>
```