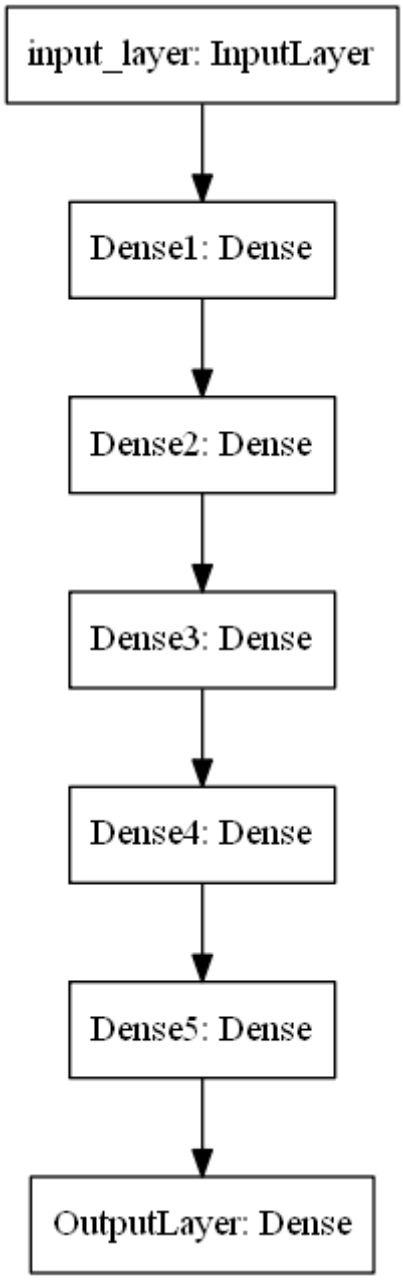
1. Download the data from here (https://drive.google.com/file/d/15dCNcmKskcFVjs7R0ElQkR61Ex53uJpM/view?usp=sharing). You have to use data.csv file for this assignment

2. Code the model to classify data like below image. You can use any number of units in your Dense layers.



3. Writing Callbacks

You have to implement the following callbacks

- Write your own callback function, that has to print the micro F1 score and AUC score after each epoch.Do not use tf.keras.metrics for calculating AUC and F1 score.
- Save your model at every epoch if your validation accuracy is improved from previous epoch.
- 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%.
```

- If you are getting any NaN values(either weigths or loss) while training, you have to terminate your training.
- You have to stop the training if your validation accuracy is not increased in last 2 epochs.
- Use tensorboard for every model and analyse your scalar plots and histograms. (you need to upload the screenshots and write the observations for each model for evaluation)

In [1]:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Dense,Activation,Input
from tensorflow.keras.models import Model
import random as rn
from sklearn.metrics import roc_auc_score,f1_score
from sklearn.model_selection import train_test_split
import datetime ,os
from tensorflow import keras
```

In [2]:

```
data = pd.read_csv('data.csv')
```

In [3]:

```
data.head(5)
```

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]:

```
x = data[['f1','f2']].values
y = data['label'].values
x
```

Out[4]:

```
array([[ 0.45056359,  1.07430486],
       [ 0.08563156,  0.96768173],
       [ 0.11732585,  0.97152066],
       ...,
       [-0.01059418,  0.13878973],
       [ 0.67182701,  0.80430563],
       [-0.8548654 , -0.58882579]])
```

In [5]:

```
x_train,x_test ,y_train ,y_test = train_test_split(x,y,test_size=0.30)
```

In [6]:

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(14000, 2)
(14000,)
(6000, 2)
(6000,)
```

#Custom callback

In [7]:

```
class F1andAuc(tf.keras.callbacks.Callback):

  def __init__(self,validation_data): #__init__ function is called everytime an object is c
    self.x_test = validation_data[0]
    self.y_test = validation_data[1]

  def on_train_begin(self,logs={}):
    self.scores = {'F1_score':[],'AUC_score':[]} #on begining of the training we are creati

  def on_epoch_end(self,epoch,logs={}): #at the end of each epoch we will get the logs and
    y_pred = self.model.predict(self.x_test)
    f1_value = f1_score(self.y_test,y_pred,average='micro') #calculating f1_score
    self.scores['F1_score'].append(f1_value)
    auc_value = roc_auc_score(self.y_test,y_pred) #calculating auc_score
    self.scores['AUC_score'].append(auc_value)

    print('F1_score:',f1_value,'AUC_score:',auc_value)
```

In [8]:

```
class terminateNaN(tf.keras.callbacks.Callback):

  def on_each_epoch_end(self,epoch,logs={}):
    loss = logs.get('loss')
    if loss is not None:
      if np.isnan(loss) or np.isinf(loss): #terminate the training if loss is nan or infini
        print("invalid loss and terminate at epoch {}".format(epoch))
        self.model.stop_training = True

  def on_each_epoch_end(self,epoch,logs={}):
    weights = self.model.get_weights()
    if weights is not None:
      if np.any([np.any(np.isnan(x)) for x in weights]): #terminate the training if weights
        print("invalid weights and terminate epoch {}".format(epoch))
        self.model.stop_training = True
```

In [9]:

```
def changelearningrate(epoch ,lr):
  decay_rate =0.95 # 5% of decay rate for every 3 epochs
  decay_st =3
  if (epoch+1) % decay_st==0:
    return lr * decay_rate
  return lr
```

In [10]:

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

In [11]:

```
%load_ext tensorboard
```

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.
3. Analyze your output and training process.

In [12]:

```
#create a model with ip,5dense layers and op
#train a model with activation function of tanh,for output softmax and initializer as rando

def createmodel_1():
  return tf.keras.models.Sequential(
      [tf.keras.layers.Dense(3,activation='tanh',input_shape=(2,),kernel_initializer = kera
       tf.keras.layers.Dense(12,activation='tanh', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='tanh', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='tanh', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='tanh', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='tanh', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(1,activation='softmax', kernel_initializer= keras.initializers
      ]
  )
```

In [13]:

```
file_p        = "model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5" #saving the model
lr_reduce    = ReduceLROnPlateau(monitor='val_accuracy',factor=0.9,patience =1,min_lr=0.0001
lr_scheduler= LearningRateScheduler(changelearningrate,verbose=1)
checkpoint   = ModelCheckpoint(filepath=file_p,monitor = 'val_accuracy',verbose=1,save_best_
earlystop    = EarlyStopping(monitor = 'val_accuracy',min_delta = 0.35 , patience =2 ,verbos
terminate    = terminateNaN()
scores       = F1andAuc(validation_data=[x_test,y_test])

logdir = os.path.join("logs",datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir,histogram_freq=1,write_graph =
```

In [14]:

```
model1 = createmodel_1()
optimizer = tf.keras.optimizers.SGD(learning_rate = 0.01,momentum = 0.9) #optimizer
model1.compile(optimizer,loss = 'BinaryCrossentropy',metrics = ['accuracy'])
model1.fit(x=x_train,y=y_train,epochs =10,validation_data=[x_test,y_test],
           callbacks = [scores,terminate,earlystop,checkpoint,lr_scheduler,lr_reduce,tensor
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 1/10
  1/438 [..............................] - ETA: 6:28 - loss: 1.8132 - accura
cy: 0.5312WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.0023s vs `on_train_batch_end` time:
0.0033s). Check your callbacks.
433/438 [============================>.] - ETA: 0s - loss: 0.7761 - accurac
y: 0.5012F1_score: 0.497 AUC_score: 0.5

Epoch 1: val_accuracy improved from -inf to 0.49700, saving model to model_s
ave/weights-01-0.4970.hdf5
438/438 [==============================] - 3s 4ms/step - loss: 0.7753 - accu
racy: 0.5013 - val_loss: 0.6939 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 2/10
433/438 [============================>.] - ETA: 0s - loss: 0.6958 - accurac
y: 0.5017F1_score: 0.497 AUC_score: 0.5

Epoch 2: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 3ms/step - loss: 0.6958 - accu
racy: 0.5013 - val_loss: 0.6934 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 3: LearningRateScheduler setting learning rate to 0.00854999963194131
8.
Epoch 3/10
418/438 [============================>..] - ETA: 0s - loss: 0.6947 - accurac
y: 0.5012F1_score: 0.497 AUC_score: 0.5

Epoch 3: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 3ms/step - loss: 0.6947 - accu
racy: 0.5013 - val_loss: 0.6937 - val_accuracy: 0.4970 - lr: 0.0085
Epoch 3: early stopping
```

Out[14]:

```
<keras.callbacks.History at 0x7f0551b1a150>
```

In [16]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 179), started 0:03:07 ago. (Use '!kill
179' to kill it.)
```

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

In [17]:

```
!rm -rf ./logs/
```

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.
3. Analyze your output and training process.

In [18]:

```
#create a model with ip,5dense layers and op
#train a model with activation function of relu,for output softmax and initializer as rando

def createmodel_2():
  return tf.keras.models.Sequential(
      [tf.keras.layers.Dense(3,activation='relu',input_shape=(2,),kernel_initializer = kera
       tf.keras.layers.Dense(12,activation='relu', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='relu', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='relu', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='relu', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(12,activation='relu', kernel_initializer= keras.initializers.R
       tf.keras.layers.Dense(1,activation='softmax', kernel_initializer= keras.initializers
      ]
  )
```

In [19]:

```
model2 = createmodel_2()
optimizer = tf.keras.optimizers.SGD(learning_rate = 0.01,momentum = 0.9) #SGD optimizer wit
model2.compile(optimizer,loss = 'BinaryCrossentropy',metrics = ['accuracy'])
model2.fit(x=x_train,y=y_train,epochs =10,validation_data=[x_test,y_test],
           callbacks = [scores,terminate,earlystop,checkpoint,lr_scheduler,lr_reduce,tensor
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 1/10
  1/438 [..............................] - ETA: 4:30 - loss: 591.9548 - accu
racy: 0.5625WARNING:tensorflow:Callback method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0018s vs `on_train_batch_end` tim
e: 0.0034s). Check your callbacks.
414/438 [===========================>..] - ETA: 0s - loss: 2.1215 - accurac
y: 0.5002F1_score: 0.497 AUC_score: 0.5

Epoch 1: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 2.0448 - accu
racy: 0.5013 - val_loss: 0.6932 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 2/10
421/438 [===========================>..] - ETA: 0s - loss: 0.6934 - accurac
y: 0.5019F1_score: 0.497 AUC_score: 0.5

Epoch 2: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6934 - accu
racy: 0.5013 - val_loss: 0.6931 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 3: LearningRateScheduler setting learning rate to 0.00854999963194131
8.
Epoch 3/10
415/438 [===========================>..] - ETA: 0s - loss: 0.6933 - accurac
y: 0.5013F1_score: 0.497 AUC_score: 0.5

Epoch 3: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6933 - accu
racy: 0.5013 - val_loss: 0.6932 - val_accuracy: 0.4970 - lr: 0.0085
Epoch 3: early stopping
```

Out[19]:

```
<keras.callbacks.History at 0x7f0579cb1110>
```

In [20]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 179), started 0:10:20 ago. (Use '!kill
179' to kill it.)

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

In [21]:

```
!rm -rf ./logs/
```

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.
3. Analyze your output and training process.

In [31]:

```
#create a model with ip,5dense layers and op
#train a model with activation function of relu,for output softmax and initializer as he_un

def createmodel_3():
  return tf.keras.models.Sequential(
      [
       tf.keras.layers.Dense(3,activation='relu' , input_shape=(2,),kernel_initializer =ker
       tf.keras.layers.Dense(12,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(12,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(12,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(12,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(12,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(1,activation='softmax',kernel_initializer=keras.initializers.h
      ]
      )
```

In [32]:

```
model3 = createmodel_3()
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01,momentum=0.9) #sgd optimizer with mo
model3.compile(optimizer , loss = 'BinaryCrossentropy', metrics=['accuracy'])
model3.fit(x=x_train,y=y_train,epochs=10,validation_data=[x_test,y_test],
           callbacks=[scores,terminate,earlystop,checkpoint,lr_scheduler,lr_reduce,tensorbo
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 1/10
  1/438 [..............................] - ETA: 4:18 - loss: 1.1065 - accura
cy: 0.5000WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.0016s vs `on_train_batch_end` time:
0.0026s). Check your callbacks.
435/438 [============================>.] - ETA: 0s - loss: 0.6764 - accurac
y: 0.5014F1_score: 0.497 AUC_score: 0.5

Epoch 1: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6761 - accu
racy: 0.5013 - val_loss: 0.6669 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.00999999977648258
2.
Epoch 2/10
420/438 [===========================>..] - ETA: 0s - loss: 0.6546 - accurac
y: 0.5025F1_score: 0.497 AUC_score: 0.5

Epoch 2: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 3ms/step - loss: 0.6548 - accu
racy: 0.5013 - val_loss: 0.6521 - val_accuracy: 0.4970 - lr: 0.0100

Epoch 3: LearningRateScheduler setting learning rate to 0.00854999963194131
8.
Epoch 3/10
416/438 [===========================>..] - ETA: 0s - loss: 0.6487 - accurac
y: 0.5010F1_score: 0.497 AUC_score: 0.5

Epoch 3: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6479 - accu
racy: 0.5013 - val_loss: 0.6510 - val_accuracy: 0.4970 - lr: 0.0085
Epoch 3: early stopping
```

Out[32]:

```
<keras.callbacks.History at 0x7f054e9f18d0>
```

In [33]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 179), started 0:31:56 ago. (Use '!kill
179' to kill it.)

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

In [26]:

```
!rm -rf ./logs/
```

Model-4

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

In [27]:

```
#create a model with ip,5dense layers and op
#train a model with activation function of relu,for output softmax and initializer as rando

def createmodel_4():
  return tf.keras.models.Sequential(
      [
       tf.keras.layers.Dense(5,activation='relu' , input_shape=(2,),kernel_initializer =ker
       tf.keras.layers.Dense(20,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(20,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(20,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(20,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(20,activation='relu',kernel_initializer=keras.initializers.he_
       tf.keras.layers.Dense(1,activation='softmax',kernel_initializer=keras.initializers.h
      ]
      )
```

In [28]:

```
model4 = createmodel_4()
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001) #adam optimizer
model4.compile(optimizer , loss = 'BinaryCrossentropy', metrics=['accuracy'])
model4.fit(x=x_train,y=y_train,epochs=10,validation_data=[x_test,y_test],
           callbacks=[scores,terminate,earlystop,checkpoint,lr_scheduler,lr_reduce,tensorbo
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.001000000047497451
3.
Epoch 1/10
  1/438 [..............................] - ETA: 4:54 - loss: 0.7233 - accura
cy: 0.5000WARNING:tensorflow:Callback method `on_train_batch_end` is slow co
mpared to the batch time (batch time: 0.0023s vs `on_train_batch_end` time:
0.0034s). Check your callbacks.
423/438 [===========================>..] - ETA: 0s - loss: 0.6734 - accurac
y: 0.5013F1_score: 0.497 AUC_score: 0.5

Epoch 1: val_accuracy did not improve from 0.49700
438/438 [==============================] - 3s 5ms/step - loss: 0.6730 - accu
racy: 0.5013 - val_loss: 0.6518 - val_accuracy: 0.4970 - lr: 0.0010

Epoch 2: LearningRateScheduler setting learning rate to 0.001000000047497451
3.
Epoch 2/10
417/438 [===========================>..] - ETA: 0s - loss: 0.6444 - accurac
y: 0.5002F1_score: 0.497 AUC_score: 0.5

Epoch 2: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6431 - accu
racy: 0.5013 - val_loss: 0.6318 - val_accuracy: 0.4970 - lr: 0.0010

Epoch 3: LearningRateScheduler setting learning rate to 0.000855000040610320
8.
Epoch 3/10
415/438 [===========================>..] - ETA: 0s - loss: 0.6312 - accurac
y: 0.5001F1_score: 0.497 AUC_score: 0.5

Epoch 3: val_accuracy did not improve from 0.49700
438/438 [==============================] - 2s 4ms/step - loss: 0.6311 - accu
racy: 0.5013 - val_loss: 0.6322 - val_accuracy: 0.4970 - lr: 8.5500e-04
Epoch 3: early stopping
```

Out[28]:

```
<keras.callbacks.History at 0x7f0551b1ae90>
```

In [29]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 179), started 0:23:27 ago. (Use '!kill
179' to kill it.)

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

In [30]:

```
!rm -rf ./logs/
```

Note

Make sure that you are plotting tensorboard plots either in your notebook or you can try to create a pdf file with all the tensorboard screenshots.Please write your analysis of tensorboard results for each model.

In [ ]:

```

```