In [7]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
import random as rn
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recal
l_score,roc_auc_score
from tensorflow.keras.callbacks import EarlyStopping, LearningRateScheduler, T
erminateOnNaN, ReduceLROnPlateau, ModelCheckpoint
```

In [8]:
```
%load_ext tensorboard
import datetime
```

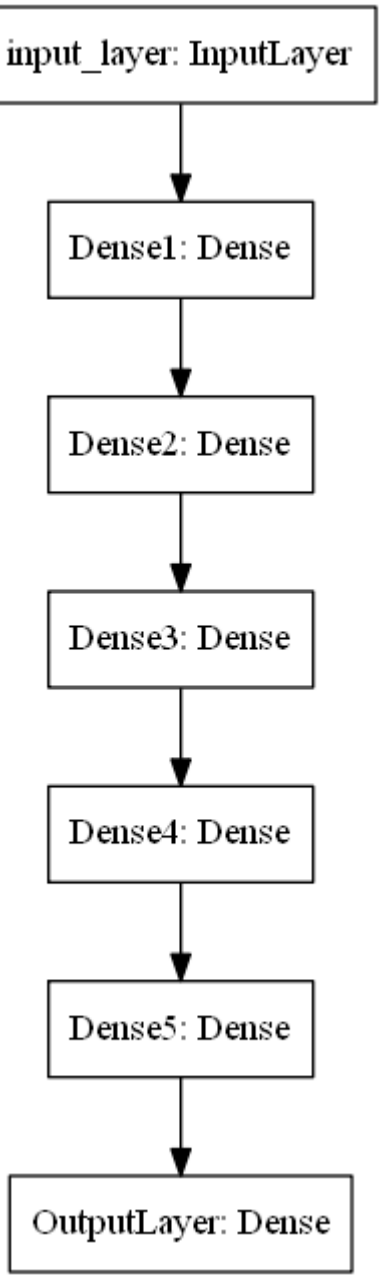
In [9]:
```
# Clear any logs from previous runs
!rm -rf ./logs/
# %tensorboard --logdir logs/fit
```

In [10]:
```
# Hide warnings from Keras
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

Instructions

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 training, you have to terminate your training.

7. You have to stop the training if your validation accuracy is not increased 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.

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.

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.

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.

Model-4

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

Load Data

In [14]:
```
data = pd.read_csv("data.csv")
X = data.iloc[:,:-1].values
y = data.iloc[:,-1].values
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
tate=10)
```

In [16]:
```
y_train = to_categorical(y_train, 2)
y_test = to_categorical(y_test, 2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

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

Callbacks Functions

In [17]:
```
# Monitor MicroF1 and AUC Score
class Metrics_Callback(tf.keras.callbacks.Callback):
  def __init__(self,x_val,y_val):
    self.x_val = x_val
    self.y_val = y_val

  def on_train_begin(self, logs={}):
    self.history = {"auc_score":[],"micro_f1":[]}

  def on_epoch_end(self, epoch, logs={}):
    auc_score = roc_auc_score(self.y_val, model.predict_proba(self.x_val))

    y_true = [0 if x[0]==1.0 else 1 for x in self.y_val]
    f1_s = f1_score(y_true,self.model.predict_classes(self.x_val), average='mi
cro')

    self.history["auc_score"].append(auc_score)
    self.history["micro_f1"].append(f1_s)
```

In [18]:
```
# Change lr on every third epoch
def schedule(epoch,lr):
  if epoch % 3 == 0:
    lr = lr - (lr*.05)
    return lr
  return lr
```

In [19]:
```
!mkdir models
```

In [20]:
```
!rm models/*
```

```
rm: cannot remove 'models/*': No such file or directory
```

In [22]:
```
Metrics = Metrics_Callback(X_test,y_test)
# Stop training is val_accuracy has not increased from last 2 epochs
EarlyStop = EarlyStopping(monitor='val_accuracy', min_delta=0, patience=2,mode
='max')

# Stop training if NaN is encountered
NanStop = TerminateOnNaN()

# Decrease lr by 5% for every 3rd epoch
LrScheduler = LearningRateScheduler(schedule,verbose=1)

# Decrease lr by 10% => lr*lr(1-0.10)
LrValAccuracy = ReduceLROnPlateau(monitor='val_accuracy', patience=1, factor=
0.9, mode='max', verbose=0)

#Save model if val_accuracy increases
filePath = "models/Model1_weights.{epoch:02d}-{val_loss:.2f}.hdf5"
model_checkpoint_callback = ModelCheckpoint(
    filepath=filePath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max')
```

In [23]:
```
# !rm -rf logs/*
```

Model 1

In [ ]:
```
# 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 [69]:
```
model = tf.keras.models.Sequential()
model.add(Input(shape=(2,)))
model.add(Dense(5,activation='tanh',kernel_initializer=tf.keras.initializers.r
andom_uniform(0,1)))
model.add(Dense(4,activation='tanh',kernel_initializer=tf.keras.initializers.r
andom_uniform(0,1)))
model.add(Dense(4,activation='tanh',kernel_initializer=tf.keras.initializers.r
andom_uniform(0,1)))
model.add(Dense(3,activation='tanh',kernel_initializer=tf.keras.initializers.r
andom_uniform(0,1)))
model.add(Dense(3,activation='tanh',kernel_initializer=tf.keras.initializers.r
andom_uniform(0,1)))
model.add(Dense(2,activation="softmax"))
```

In [70]:
```
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.
9),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [71]:
```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogr
am_freq=1, write_graph=True)

model.fit(X_train,y_train,
          epochs=10,
          validation_data=(X_test,y_test),
          callbacks = [tensorboard_callback, Metrics, EarlyStop, NanStop, LrSc
heduler, LrValAccuracy, model_checkpoint_callback])
```

```
Epoch 1/10

Epoch 00001: LearningRateScheduler reducing learning rate to 0.09500000141561
031.
  3/438 [..............................] - ETA: 15s - loss: 0.8104 - accurac
y: 0.4878 WARNING:tensorflow:Callback method `on_train_batch_end` is slow com
pared to the batch time (batch time: 0.0027s vs `on_train_batch_end` time: 0.
0110s). Check your callbacks.
438/438 [==============================] - 2s 4ms/step - loss: 0.7002 - accur
acy: 0.5174 - val_loss: 0.6950 - val_accuracy: 0.5013
Epoch 2/10

Epoch 00002: LearningRateScheduler reducing learning rate to 0.09499999880790
71.
438/438 [==============================] - 2s 4ms/step - loss: 0.6914 - accur
acy: 0.5282 - val_loss: 0.6859 - val_accuracy: 0.5405
Epoch 3/10

Epoch 00003: LearningRateScheduler reducing learning rate to 0.09499999880790
71.
438/438 [==============================] - 2s 4ms/step - loss: 0.6889 - accur
acy: 0.5336 - val_loss: 0.6957 - val_accuracy: 0.5012
Epoch 4/10

Epoch 00004: LearningRateScheduler reducing learning rate to 0.08122500181198
12.
438/438 [==============================] - 2s 4ms/step - loss: 0.6853 - accur
acy: 0.5243 - val_loss: 0.6897 - val_accuracy: 0.5327
```

Out[71]:
```
<tensorflow.python.keras.callbacks.History at 0x7f20e4cb03c8>
```

In [72]:
```
Metrics.history
```

Out[72]:
```
{'auc_score': [0.5142775777334787,
  0.5143046612142665,
  0.528762739930473,
  0.5292529370437684],
 'micro_f1': [0.5013333333333333,
  0.5405,
  0.5011666666666666,
  0.5326666666666666]}
```

In [73]:
```
%tensorboard --logdir logs/fit
```

Model 2

In [ ]:
```
# 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 [46]:
```
model2 = tf.keras.models.Sequential()
model2.add(Input(shape=(2,)))
model2.add(Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.
random_uniform(0,1)))
model2.add(Dense(4,activation='relu',kernel_initializer=tf.keras.initializers.
random_uniform(0,1)))
model2.add(Dense(4,activation='relu',kernel_initializer=tf.keras.initializers.
random_uniform(0,1)))
model2.add(Dense(3,activation='relu',kernel_initializer=tf.keras.initializers.
random_uniform(0,1)))
model2.add(Dense(3,activation='relu',kernel_initializer=tf.keras.initializers.
random_uniform(0,1)))
model2.add(Dense(2,activation="softmax"))
```

In [47]:
```
model2.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1, momentum=
0.9),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [48]:
```
filePath = "models/Model2_weights.{epoch:02d}-{val_loss:.2f}.hdf5"
model_checkpoint_callback = ModelCheckpoint(
    filepath=filePath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max')

!rm -rf logs/*
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogr
am_freq=1, write_graph=True)

model2.fit(X_train,y_train,
          epochs=10,
          validation_data=(X_test,y_test),
          callbacks = [Metrics, EarlyStop, NanStop, LrScheduler, LrValAccuracy
, model_checkpoint_callback, tensorboard_callback])
```

```
Epoch 1/10

Epoch 00001: LearningRateScheduler reducing learning rate to 0.09500000141561
031.
  3/438 [..............................] - ETA: 15s - loss: 0.9888 - accurac
y: 0.5365 WARNING:tensorflow:Callback method `on_train_batch_end` is slow com
pared to the batch time (batch time: 0.0037s vs `on_train_batch_end` time: 0.
0104s). Check your callbacks.
438/438 [==============================] - 2s 4ms/step - loss: 0.7236 - accur
acy: 0.4966 - val_loss: 0.6943 - val_accuracy: 0.5012
Epoch 2/10

Epoch 00002: LearningRateScheduler reducing learning rate to 0.09499999880790
71.
438/438 [==============================] - 2s 4ms/step - loss: 0.6973 - accur
acy: 0.4961 - val_loss: 0.6942 - val_accuracy: 0.5012
Epoch 3/10

Epoch 00003: LearningRateScheduler reducing learning rate to 0.08550000190734
863.
438/438 [==============================] - 2s 4ms/step - loss: 0.6966 - accur
acy: 0.5005 - val_loss: 0.6982 - val_accuracy: 0.5012
```

Out[48]:
```
<tensorflow.python.keras.callbacks.History at 0x7f20e483e828>
```

In [49]:
```
Metrics.history
```

Out[49]:
```
{'auc_score': [0.5071516222699435, 0.5071516222699435, 0.5071516222699435],
 'micro_f1': [0.5011666666666666, 0.5011666666666666, 0.5011666666666666]}
```

In [51]:
```
%tensorboard --logdir logs/fit
```

Model 3

In [ ]:
```
# 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 [ ]:
```
model3 = tf.keras.models.Sequential()
model3.add(Input(shape=(2,)))
model3.add(Dense(10,activation='relu',kernel_initializer=tf.keras.initializers
.he_uniform()))
model3.add(Dense(10,activation='relu',kernel_initializer=tf.keras.initializers
.he_uniform()))
model3.add(Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model3.add(Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model3.add(Dense(3,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model3.add(Dense(2,activation="softmax"))
```

In [ ]:
```
model3.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1, momentum=
0.9),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
filePath = "models/Model3_weights.{epoch:02d}-{val_loss:.2f}.hdf5"
model_checkpoint_callback = ModelCheckpoint(
    filepath=filePath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max')


!rm -rf logs/*
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogr
am_freq=1, write_graph=True)

model3.fit(X_train,y_train,
          epochs=10,
          validation_data=(X_test,y_test),
          callbacks = [Metrics, EarlyStop, NanStop, LrScheduler, LrValAccuracy
, model_checkpoint_callback, tensorboard_callback])
```

```
Epoch 1/10

Epoch 00001: LearningRateScheduler reducing learning rate to 0.09500000141561
031.
  3/438 [..............................] - ETA: 15s - loss: 0.7524 - accurac
y: 0.3420 WARNING:tensorflow:Callback method `on_train_batch_end` is slow com
pared to the batch time (batch time: 0.0031s vs `on_train_batch_end` time: 0.
0112s). Check your callbacks.
438/438 [==============================] - 2s 4ms/step - loss: 0.6862 - accur
acy: 0.5265 - val_loss: 0.6362 - val_accuracy: 0.6400
Epoch 2/10

Epoch 00002: LearningRateScheduler reducing learning rate to 0.09499999880790
71.
438/438 [==============================] - 2s 4ms/step - loss: 0.6373 - accur
acy: 0.6318 - val_loss: 0.6377 - val_accuracy: 0.6540
Epoch 3/10

Epoch 00003: LearningRateScheduler reducing learning rate to 0.09499999880790
71.
438/438 [==============================] - 2s 4ms/step - loss: 0.6369 - accur
acy: 0.6412 - val_loss: 0.6516 - val_accuracy: 0.6372
Epoch 4/10

Epoch 00004: LearningRateScheduler reducing learning rate to 0.08122500181198
12.
438/438 [==============================] - 2s 4ms/step - loss: 0.6328 - accur
acy: 0.6562 - val_loss: 0.6221 - val_accuracy: 0.6583
Epoch 5/10

Epoch 00005: LearningRateScheduler reducing learning rate to 0.08122500032186
508.
438/438 [==============================] - 2s 4ms/step - loss: 0.6257 - accur
acy: 0.6602 - val_loss: 0.6291 - val_accuracy: 0.6418
Epoch 6/10

Epoch 00006: LearningRateScheduler reducing learning rate to 0.07310250401496
887.
438/438 [==============================] - 2s 4ms/step - loss: 0.6258 - accur
acy: 0.6598 - val_loss: 0.6228 - val_accuracy: 0.6543
```

Out[ ]: <tensorflow.python.keras.callbacks.History at 0x7f07e629e358>

In [ ]:
```
Metrics.history
```

Out[ ]:
```
{'auc_score': [0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722],
 'micro_f1': [0.64,
  0.654,
  0.6371666666666667,
  0.6583333333333333,
  0.6418333333333334,
  0.6543333333333333]}
```

In [3]:
```
%tensorboard --logdir logs/fit
```

Model 4

In [ ]:
```
# 1. Used relu as an activation for every layer except output layer.
# 2. used Adam optimizer with learning rate = 0.001
# 3. used he_uniform() as initilizer
```

In [ ]:
```
model4 = tf.keras.models.Sequential()
model4.add(Input(shape=(2,)))
model4.add(Dense(10,activation='relu',kernel_initializer=tf.keras.initializers
.he_uniform()))
model4.add(Dense(10,activation='relu',kernel_initializer=tf.keras.initializers
.he_uniform()))
model4.add(Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model4.add(Dense(5,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model4.add(Dense(3,activation='relu',kernel_initializer=tf.keras.initializers.
he_uniform()))
model4.add(Dense(2,activation="softmax"))
```

In [ ]:
```
model4.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
               loss='categorical_crossentropy',
               metrics=['accuracy'])
```

In [ ]:
```
filePath = "models/Model4_weights.{epoch:02d}-{val_loss:.2f}.hdf5"
model_checkpoint_callback = ModelCheckpoint(
    filepath=filePath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max')

!rm -rf logs/*
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogr
am_freq=1, write_graph=True)

model4.fit(X_train,y_train,
          epochs=10,
          validation_data=(X_test,y_test),
          callbacks = [Metrics, EarlyStop, NanStop, LrScheduler, LrValAccuracy
, model_checkpoint_callback, tensorboard_callback])
```

```
Epoch 1/10

Epoch 00001: LearningRateScheduler reducing learning rate to 0.00095000004512
25787.
  3/438 [..............................] - ETA: 18s - loss: 0.7060 - accurac
y: 0.5417 WARNING:tensorflow:Callback method `on_train_batch_end` is slow com
pared to the batch time (batch time: 0.0037s vs `on_train_batch_end` time: 0.
0131s). Check your callbacks.
438/438 [==============================] - 2s 4ms/step - loss: 0.6840 - accur
acy: 0.5552 - val_loss: 0.6581 - val_accuracy: 0.6200
Epoch 2/10

Epoch 00002: LearningRateScheduler reducing learning rate to 0.00095000001601
87483.
438/438 [==============================] - 2s 4ms/step - loss: 0.6484 - accur
acy: 0.6291 - val_loss: 0.6249 - val_accuracy: 0.6547
Epoch 3/10

Epoch 00003: LearningRateScheduler reducing learning rate to 0.00095000001601
87483.
438/438 [==============================] - 2s 4ms/step - loss: 0.6178 - accur
acy: 0.6638 - val_loss: 0.6134 - val_accuracy: 0.6627
Epoch 4/10

Epoch 00004: LearningRateScheduler reducing learning rate to 0.00090250001521
78108.
438/438 [==============================] - 2s 4ms/step - loss: 0.6033 - accur
acy: 0.6771 - val_loss: 0.6116 - val_accuracy: 0.6638
Epoch 5/10

Epoch 00005: LearningRateScheduler reducing learning rate to 0.00090250000357
62787.
438/438 [==============================] - 2s 4ms/step - loss: 0.6056 - accur
acy: 0.6730 - val_loss: 0.6107 - val_accuracy: 0.6660
Epoch 6/10

Epoch 00006: LearningRateScheduler reducing learning rate to 0.00090250000357
62787.
438/438 [==============================] - 2s 4ms/step - loss: 0.5978 - accur
acy: 0.6720 - val_loss: 0.6121 - val_accuracy: 0.6680
Epoch 7/10

Epoch 00007: LearningRateScheduler reducing learning rate to 0.00085737500339
74648.
438/438 [==============================] - 2s 4ms/step - loss: 0.5991 - accur
acy: 0.6731 - val_loss: 0.6098 - val_accuracy: 0.6653
Epoch 8/10

Epoch 00008: LearningRateScheduler reducing learning rate to 0.00077163748210
29603.
438/438 [==============================] - 2s 4ms/step - loss: 0.5981 - accur
acy: 0.6741 - val_loss: 0.6115 - val_accuracy: 0.6682
Epoch 9/10

Epoch 00009: LearningRateScheduler reducing learning rate to 0.00077163748210
29603.
438/438 [==============================] - 2s 4ms/step - loss: 0.6009 - accur
```

```
acy: 0.6707 - val_loss: 0.6077 - val_accuracy: 0.6692
Epoch 10/10

Epoch 00010: LearningRateScheduler reducing learning rate to 0.00073305560799
78123.
438/438 [==============================] - 2s 4ms/step - loss: 0.5997 - accur
acy: 0.6688 - val_loss: 0.6078 - val_accuracy: 0.6683
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f07f1d148d0>
```

In [ ]:
```
Metrics.history
```

Out[ ]:
```
{'auc_score': [0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722,
  0.4961677846912722],
 'micro_f1': [0.62,
  0.6546666666666666,
  0.6626666666666666,
  0.6638333333333334,
  0.666,
  0.668,
  0.6653333333333333,
  0.6681666666666667,
  0.6691666666666667,
  0.6683333333333333]}
```

In [4]:
```
%tensorboard --logdir logs/fit
```