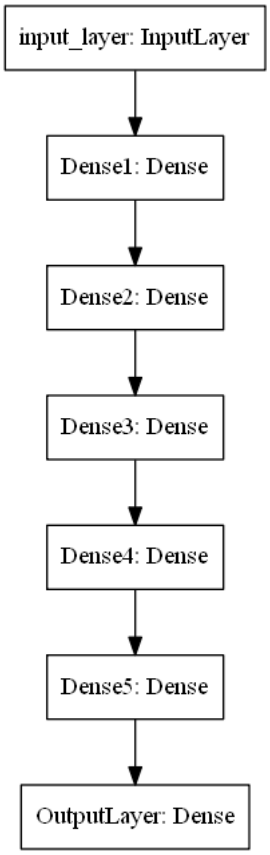
1. Download the data from here

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.

In [ ]:
```
import tensorflow.compat.v1 as tf
tf.compat.v1.disable_eager_execution()
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from keras.utils import np_utils
from keras.datasets import mnist
import seaborn as sns
from keras.initializers import RandomNormal

from keras.models import Sequential
from keras.layers import Dense, Activation
from tensorflow.keras import initializers
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import LearningRateScheduler
```

In [ ]:
```
data = pd.read_csv('data.csv')

print(data.head(5)) # print first five rows of data.

# save the labels into a variable Y.
Y = data['label']

# Drop the label feature and store the remaining feature in X.
X = data.drop("label",axis=1)
```

```
         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 [ ]:
```
print(X.shape)
print(Y.shape)
```

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

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X.values,Y.values, test_size=0.3, random_state=0)
```

In [ ]:
```
(x_train.shape) ,(x_test.shape)
```

Out[ ]:
```
((14000, 2), (6000, 2))
```

In [ ]:
```
y_train = tf.keras.utils.to_categorical(y_train, 2)
y_test = tf.keras.utils.to_categorical(y_test, 2)
```

In [ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Model 1

In [ ]:
```
output_dim = 2
input_dim = x_train.shape[1]
```

In [ ]:
```
input_dim = x_train.shape[1]
model_sigmoid = Sequential()
model_sigmoid.add(Dense(100,activation='tanh',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None),input_shap
model_sigmoid.add(Dense(60,activation='tanh',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(30,activation='tanh',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(20,activation='tanh',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(10,activation='tanh',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(output_dim, activation='softmax'))

model_sigmoid.summary()

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False)
```

```
#model_sigmoid.compile(optimizer='sgd', loss='categorical_crossentropy',metrics=['accuracy'])
model_sigmoid.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               300
_________________________________________________________________
dense_1 (Dense)              (None, 60)                6060
_________________________________________________________________
dense_2 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_3 (Dense)              (None, 20)                620
_________________________________________________________________
dense_4 (Dense)              (None, 10)                210
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 22
=================================================================
Total params: 9,042
Trainable params: 9,042
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
%load_ext tensorboard
```

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

In [ ]:
```
import tensorflow as tf
import datetime

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_freq=1, write_graph=True,write_grads=True)
```

In [ ]:
```
class f1_score_and_auc_Callback(tf.keras.callbacks.Callback):

    def  on_train_begin(self,logs={}):
      self.f1_micro=[]
      self.auc_score=[]

    def on_epoch_end(self, epoch, logs=None):
      y_pred=self.model.predict(x_test).round()
      y_pred_=self.model.predict(x_test)
      y_true=y_test
      score=f1_score(y_true, y_pred, average='micro')
      Auc_score  = roc_auc_score(y_true, y_pred_)

      self.f1_micro.append(score)
      self.auc_score.append(Auc_score)
      print(" F1 micro :",score)
      print(" auc score :",Auc_score)

metrics=f1_score_and_auc_Callback()

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

terminate = TerminateNaN()

from tensorflow.keras.callbacks import EarlyStopping
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.35, patience=5, verbose=1)

class LossHistory(tf.keras.callbacks.Callback):

    def on_train_begin(self, logs={}):
        ## on begin of training, we are creating a instance varible called history
        ## it is a dict with keys [loss, acc, val_loss, val_acc]
        self.history={'loss': [],'acc': [],'val_loss': [],'val_acc': [],'decay' : []}

    def on_epoch_end(self, epoch, logs={}):
        ## on end of each epoch, we will get logs and update the self.history dict
        self.history['loss'].append(logs.get('loss'))
        self.history['acc'].append(logs.get('accuracy'))
        self.history['decay'].append(learn_decay(epoch))

        if logs.get('val_loss', -1) != -1:
            self.history['val_loss'].append(logs.get('val_loss'))
        if logs.get('val_accuracy', -1) != -1:
            self.history['val_acc'].append(logs.get('val_accuracy'))

history_own=LossHistory()

def learn_decay(epoch):
  a = history_own.history
  if epoch == 0:
```

```
    initial_learningrate=0.1
    return initial_learningrate

  if epoch % 3 == 0:
    l_rate = a['decay'][epoch-1]
    changed = l_rate * 0.95
    return changed

  else:
    l_rate = a['decay'][epoch-1]
    return l_rate




lrschedule = LearningRateScheduler(learn_decay, verbose=1)

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy', factor=0.90,patience=0, min_lr=0.000001, verbose=1)

filepath="model_save/model.{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')


callback_list = [metrics,checkpoint,history_own,lrschedule,terminate,tensorboard_callback,reduce_lr,earlystop]
```

In [ ]:
```
model_sigmoid.fit(x_train,y_train,epochs=50, validation_data=(x_test,y_test),callbacks=callback_list)
```

```
Train on 14000 samples, validate on 6000 samples

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/50
   64/14000 [..............................] - ETA: 42s - loss: 0.6942 - accuracy: 0.5000WARNING:tensorflow:Callback method `on_train_bat
ch_begin` is slow compared to the batch time (batch time: 0.0050s vs `on_train_batch_begin` time: 0.0199s). Check your callbacks.
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0050s vs `on_train_batch_end` t
ime: 0.0105s). Check your callbacks.
13536/14000 [============================>.] - ETA: 0s - loss: 0.6938 - accuracy: 0.5041 F1 micro : 0.49466666666666664
 auc score : 0.4833131847445754

Epoch 00001: val_accuracy improved from -inf to 0.49467, saving model to model_save/model.01-0.49.h5
14000/14000 [==============================] - 4s 278us/sample - loss: 0.6938 - accuracy: 0.5037 - val_loss: 0.6935 - val_accuracy: 0.494
7

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/50
13568/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.5032 F1 micro : 0.5053333333333333
 auc score : 0.4831545000231138

Epoch 00002: val_accuracy improved from 0.49467 to 0.50533, saving model to model_save/model.02-0.51.h5
14000/14000 [==============================] - 4s 255us/sample - loss: 0.6938 - accuracy: 0.5030 - val_loss: 0.6931 - val_accuracy: 0.505
3

Epoch 00003: LearningRateScheduler reducing learning rate to 0.1.
Epoch 3/50
13568/14000 [============================>.] - ETA: 0s - loss: 0.6940 - accuracy: 0.4975 F1 micro : 0.49466666666666664
 auc score : 0.48308907591263717

Epoch 00003: val_accuracy did not improve from 0.50533

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 231us/sample - loss: 0.6941 - accuracy: 0.4973 - val_loss: 0.6932 - val_accuracy: 0.494
7

Epoch 00004: LearningRateScheduler reducing learning rate to 0.095.
Epoch 4/50
13792/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4998 F1 micro : 0.5053333333333333
 auc score : 0.4830607671361719

Epoch 00004: val_accuracy did not improve from 0.50533

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 226us/sample - loss: 0.6938 - accuracy: 0.4999 - val_loss: 0.6931 - val_accuracy: 0.505
3

Epoch 00005: LearningRateScheduler reducing learning rate to 0.095.
Epoch 5/50
13824/14000 [============================>.] - ETA: 0s - loss: 0.6940 - accuracy: 0.4946 F1 micro : 0.5053333333333333
 auc score : 0.48302562424880346

Epoch 00005: val_accuracy did not improve from 0.50533

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 236us/sample - loss: 0.6940 - accuracy: 0.4944 - val_loss: 0.6931 - val_accuracy: 0.505
3

Epoch 00006: LearningRateScheduler reducing learning rate to 0.095.
Epoch 6/50
13792/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4992 F1 micro : 0.49466666666666664
 auc score : 0.4830291524280096

Epoch 00006: val_accuracy did not improve from 0.50533

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 249us/sample - loss: 0.6937 - accuracy: 0.4990 - val_loss: 0.6932 - val_accuracy: 0.494
```

```
7
Epoch 00006: early stopping
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f23847b8b70>
```

In [ ]:
```
#Model 1
%tensorboard --logdir '/content/content/MyDrive/callbacks/callback/Callbacks/logs\fit\20210111-133926'
```

In [ ]:
```
ls
```

```
Call_Backs_Assignment.ipynb   data.csv                    model_save/
Call_Backs_Reference.ipynb    'logs\fit\20210111-133926'/   tensorboard.ipynb
```

Model 2

In [ ]:
```
output_dim = 2
input_dim = x_train.shape[1]

nb_epoch = 50

input_dim = x_train.shape[1]
model_sigmoid = Sequential()
model_sigmoid.add(Dense(100,activation='relu',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None),input_shap
model_sigmoid.add(Dense(60,activation='relu',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(30,activation='relu',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(20,activation='relu',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(10,activation='relu',kernel_initializer=initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)))
model_sigmoid.add(Dense(output_dim, activation='softmax'))

model_sigmoid.summary()

model_sigmoid.compile(optimizer='sgd', loss='categorical_crossentropy',metrics=['accuracy'])

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_freq=1, write_graph=True,write_grads=True)

model_sigmoid.fit(x_train,y_train,epochs=50, validation_data=(x_test,y_test),callbacks=callback_list)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               300
_________________________________________________________________
dense_1 (Dense)              (None, 60)                6060
_________________________________________________________________
dense_2 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_3 (Dense)              (None, 20)                620
_________________________________________________________________
dense_4 (Dense)              (None, 10)                210
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 22
=================================================================
Total params: 9,042
Trainable params: 9,042
Non-trainable params: 0
_________________________________________________________________
Train on 14000 samples, validate on 6000 samples

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/50
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be remove
d in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
   64/14000 [..............................] - ETA: 1:33 - loss: 0.6932 - accuracy: 0.4844WARNING:tensorflow:Callback method `on_train_ba
tch_begin` is slow compared to the batch time (batch time: 0.0037s vs `on_train_batch_begin` time: 0.0196s). Check your callbacks.
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0037s vs `on_train_batch_end` t
ime: 0.0113s). Check your callbacks.
13664/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4939 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00001: val_accuracy improved from -inf to 0.49467, saving model to model_save/model.01-0.49.h5
14000/14000 [==============================] - 4s 282us/sample - loss: 0.6937 - accuracy: 0.4936 - val_loss: 0.6946 - val_accuracy: 0.494
7

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/50
13824/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4981 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00002: val_accuracy did not improve from 0.49467

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 211us/sample - loss: 0.6937 - accuracy: 0.4981 - val_loss: 0.6932 - val_accuracy: 0.494
7

Epoch 00003: LearningRateScheduler reducing learning rate to 0.1.
Epoch 3/50
13600/14000 [============================>.] - ETA: 0s - loss: 0.6936 - accuracy: 0.5069 F1 micro : 0.5053333333333333
 auc score : 0.5

Epoch 00003: val_accuracy improved from 0.49467 to 0.50533, saving model to model_save/model.03-0.51.h5
```

```
14000/14000 [==============================] - 4s 289us/sample - loss: 0.6936 - accuracy: 0.5061 - val_loss: 0.6931 - val_accuracy: 0.505
3

Epoch 00004: LearningRateScheduler reducing learning rate to 0.095.
Epoch 4/50
13696/14000 [============================>.] - ETA: 0s - loss: 0.6936 - accuracy: 0.4999 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00004: val_accuracy did not improve from 0.50533

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 219us/sample - loss: 0.6936 - accuracy: 0.4997 - val_loss: 0.6940 - val_accuracy: 0.494
7

Epoch 00005: LearningRateScheduler reducing learning rate to 0.095.
Epoch 5/50
13856/14000 [============================>.] - ETA: 0s - loss: 0.6935 - accuracy: 0.5004 F1 micro : 0.5053333333333333
 auc score : 0.5

Epoch 00005: val_accuracy did not improve from 0.50533

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 219us/sample - loss: 0.6936 - accuracy: 0.4997 - val_loss: 0.6931 - val_accuracy: 0.505
3

Epoch 00006: LearningRateScheduler reducing learning rate to 0.095.
Epoch 6/50
13600/14000 [============================>.] - ETA: 0s - loss: 0.6937 - accuracy: 0.4911 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00006: val_accuracy did not improve from 0.50533

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 216us/sample - loss: 0.6937 - accuracy: 0.4919 - val_loss: 0.6943 - val_accuracy: 0.494
7
Epoch 00006: early stopping
```

Out[ ]: <tensorflow.python.keras.callbacks.History at 0x7f01dfa6d5c0>

In [ ]:
```
ls
```

```
Call_Backs_Assignment.ipynb  'logs\fit\20210111-133926'/   tensorboard.ipynb
Call_Backs_Reference.ipynb   'logs\fit\20210111-135426'/
data.csv                      model_save/
```

In [ ]:
```
#model2
%tensorboard --logdir '/content/content/MyDrive/callbacks/callback/Callbacks/logs\fit\20210111-135426'
```

Model 3

In [ ]:
```
output_dim = 2
input_dim = x_train.shape[1]

nb_epoch = 50

input_dim = x_train.shape[1]
model_sigmoid = Sequential()
model_sigmoid.add(Dense(100,activation='relu',kernel_initializer=initializers.HeUniform(),input_shape=(input_dim,)))
model_sigmoid.add(Dense(60,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(30,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(20,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(10,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(output_dim, activation='softmax'))

model_sigmoid.summary()

model_sigmoid.compile(optimizer='sgd', loss='categorical_crossentropy',metrics=['accuracy'])

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_freq=1, write_graph=True,write_grads=True)

model_sigmoid.fit(x_train,y_train,epochs=50, validation_data=(x_test,y_test),callbacks=callback_list)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 100)               300
_________________________________________________________________
dense_7 (Dense)              (None, 60)                6060
_________________________________________________________________
dense_8 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_9 (Dense)              (None, 20)                620
_________________________________________________________________
dense_10 (Dense)             (None, 10)                210
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 22
=================================================================
Total params: 9,042
Trainable params: 9,042
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 14000 samples, validate on 6000 samples

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/50
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be remove
d in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
   64/14000 [..............................] - ETA: 1:40 - loss: 0.8308 - accuracy: 0.4688WARNING:tensorflow:Callback method `on_train_ba
tch_begin` is slow compared to the batch time (batch time: 0.0044s vs `on_train_batch_begin` time: 0.0199s). Check your callbacks.
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0044s vs `on_train_batch_end` t
ime: 0.0152s). Check your callbacks.
13536/14000 [============================>.] - ETA: 0s - loss: 0.6502 - accuracy: 0.6194 F1 micro : 0.6603333333333333
 auc score : 0.7309752798540634

Epoch 00001: val_accuracy improved from -inf to 0.66033, saving model to model_save/model.01-0.66.h5
14000/14000 [==============================] - 4s 269us/sample - loss: 0.6490 - accuracy: 0.6199 - val_loss: 0.6176 - val_accuracy: 0.660
3

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/50
13696/14000 [============================>.] - ETA: 0s - loss: 0.6200 - accuracy: 0.6557 F1 micro : 0.654
 auc score : 0.7094296895557894

Epoch 00002: val_accuracy did not improve from 0.66033

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 221us/sample - loss: 0.6199 - accuracy: 0.6555 - val_loss: 0.6256 - val_accuracy: 0.654
0

Epoch 00003: LearningRateScheduler reducing learning rate to 0.1.
Epoch 3/50
13824/14000 [============================>.] - ETA: 0s - loss: 0.6145 - accuracy: 0.6536 F1 micro : 0.655
 auc score : 0.7250976111059748

Epoch 00003: val_accuracy did not improve from 0.66033

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 223us/sample - loss: 0.6148 - accuracy: 0.6535 - val_loss: 0.6132 - val_accuracy: 0.655
0

Epoch 00004: LearningRateScheduler reducing learning rate to 0.095.
Epoch 4/50
13536/14000 [============================>.] - ETA: 0s - loss: 0.6106 - accuracy: 0.6653 F1 micro : 0.6705
 auc score : 0.7323616598155169

Epoch 00004: val_accuracy improved from 0.66033 to 0.67050, saving model to model_save/model.04-0.67.h5
14000/14000 [==============================] - 4s 291us/sample - loss: 0.6107 - accuracy: 0.6648 - val_loss: 0.6062 - val_accuracy: 0.670
5

Epoch 00005: LearningRateScheduler reducing learning rate to 0.095.
Epoch 5/50
13536/14000 [============================>.] - ETA: 0s - loss: 0.6071 - accuracy: 0.6659 F1 micro : 0.6368333333333334
 auc score : 0.7180994815410109

Epoch 00005: val_accuracy did not improve from 0.67050

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 221us/sample - loss: 0.6076 - accuracy: 0.6651 - val_loss: 0.6308 - val_accuracy: 0.636
8

Epoch 00006: LearningRateScheduler reducing learning rate to 0.095.
Epoch 6/50
14000/14000 [==============================] - ETA: 0s - loss: 0.6087 - accuracy: 0.6665 F1 micro : 0.6693333333333333
 auc score : 0.7312650906058644

Epoch 00006: val_accuracy did not improve from 0.67050

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 225us/sample - loss: 0.6087 - accuracy: 0.6665 - val_loss: 0.6060 - val_accuracy: 0.669
3
Epoch 00006: early stopping
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7f4ee3d0be10>
```

In [ ]:
```
ls
```

```
Call_Backs_Assignment.ipynb  'logs\fit\20210111-133926'/   model_save/
Call_Backs_Reference.ipynb   'logs\fit\20210111-135426'/   tensorboard.ipynb
data.csv                     'logs\fit\20210111-135903'/
```

In [ ]:
```
#model3
%tensorboard --logdir '/content/content/MyDrive/callbacks/callback/Callbacks/logs\fit\20210111-135903'
```

Model 4

In [ ]:
```
output_dim = 2
input_dim = x_train.shape[1]

nb_epoch = 50

input_dim = x_train.shape[1]
model_sigmoid = Sequential()
model_sigmoid.add(Dense(100,activation='relu',kernel_initializer=initializers.HeUniform(),input_shape=(input_dim,)))
model_sigmoid.add(Dense(60,activation='relu',kernel_initializer=initializers.HeUniform()))
```

```
model_sigmoid.add(Dense(30,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(20,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(10,activation='relu',kernel_initializer=initializers.HeUniform()))
model_sigmoid.add(Dense(output_dim, activation='softmax'))

model_sigmoid.summary()

optimizers =tf.keras.optimizers.Adam(
    learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False,
    name='Adam')

model_sigmoid.compile(optimizer=optimizers, loss='categorical_crossentropy',metrics=['accuracy'])

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_freq=1, write_graph=True,write_grads=True)

model_sigmoid.fit(x_train,y_train,epochs=50, validation_data=(x_test,y_test),callbacks=callback_list)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               300
_________________________________________________________________
dense_1 (Dense)              (None, 60)                6060
_________________________________________________________________
dense_2 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_3 (Dense)              (None, 20)                620
_________________________________________________________________
dense_4 (Dense)              (None, 10)                210
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 22
=================================================================
Total params: 9,042
Trainable params: 9,042
Non-trainable params: 0
_________________________________________________________________
Train on 14000 samples, validate on 6000 samples

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/50
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be remove
d in a future version. This property should not be used in TensorFlow 2.0, as `updates` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
   64/14000 [..............................] - ETA: 1:42 - loss: 10.4953 - accuracy: 0.5312WARNING:tensorflow:Callback method `on_train_b
atch_begin` is slow compared to the batch time (batch time: 0.0054s vs `on_train_batch_begin` time: 0.0201s). Check your callbacks.
WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0054s vs `on_train_batch_end` t
ime: 0.0140s). Check your callbacks.
13600/14000 [============================>.] - ETA: 0s - loss: 0.7009 - accuracy: 0.6232 F1 micro : 0.6556666666666666
 auc score : 0.7199451915417932

Epoch 00001: val_accuracy improved from -inf to 0.65567, saving model to model_save/model.01-0.66.h5
14000/14000 [==============================] - 4s 302us/sample - loss: 0.6991 - accuracy: 0.6239 - val_loss: 0.6281 - val_accuracy: 0.655
7

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/50
13888/14000 [============================>.] - ETA: 0s - loss: 0.6785 - accuracy: 0.5545 F1 micro : 0.5053333333333333
 auc score : 0.5

Epoch 00002: val_accuracy did not improve from 0.65567

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 222us/sample - loss: 0.6787 - accuracy: 0.5542 - val_loss: 0.6986 - val_accuracy: 0.505
3

Epoch 00003: LearningRateScheduler reducing learning rate to 0.1.
Epoch 3/50
13536/14000 [============================>.] - ETA: 0s - loss: 0.6965 - accuracy: 0.4995 F1 micro : 0.5053333333333333
 auc score : 0.5

Epoch 00003: val_accuracy did not improve from 0.65567

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.09000000134110452.
14000/14000 [==============================] - 3s 223us/sample - loss: 0.6965 - accuracy: 0.5006 - val_loss: 0.7012 - val_accuracy: 0.505
3

Epoch 00004: LearningRateScheduler reducing learning rate to 0.095.
Epoch 4/50
13984/14000 [============================>.] - ETA: 0s - loss: 0.6976 - accuracy: 0.4996 F1 micro : 0.5053333333333333
 auc score : 0.5

Epoch 00004: val_accuracy did not improve from 0.65567

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 222us/sample - loss: 0.6976 - accuracy: 0.4996 - val_loss: 0.6966 - val_accuracy: 0.505
3

Epoch 00005: LearningRateScheduler reducing learning rate to 0.095.
Epoch 5/50
13600/14000 [============================>.] - ETA: 0s - loss: 0.6963 - accuracy: 0.5001 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00005: val_accuracy did not improve from 0.65567
```

```
Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 223us/sample - loss: 0.6962 - accuracy: 0.5009 - val_loss: 0.6961 - val_accuracy: 0.494
7

Epoch 00006: LearningRateScheduler reducing learning rate to 0.095.
Epoch 6/50
13760/14000 [============================>.] - ETA: 0s - loss: 0.6964 - accuracy: 0.5059 F1 micro : 0.49466666666666664
 auc score : 0.5

Epoch 00006: val_accuracy did not improve from 0.65567

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.08549999892711639.
14000/14000 [==============================] - 3s 224us/sample - loss: 0.6963 - accuracy: 0.5066 - val_loss: 0.7070 - val_accuracy: 0.494
7
Epoch 00006: early stopping
```

Out[ ]:
```
<tensorflow.python.keras.callbacks.History at 0x7fb41881aac8>
```

In [ ]:
```
ls
```

```
 Call_Backs_Assignment.ipynb  'logs\fit\20210111-135903'/
 Call_Backs_Reference.ipynb   'logs\fit\20210111-140542'/
 data.csv                      model_save/
'logs\fit\20210111-133926'/    tensorboard.ipynb
'logs\fit\20210111-135426'/
```

In [ ]:
```
%tensorboard --logdir '/content/content/MyDrive/callbacks/callback/Callbacks/logs\fit\20210111-140542'
```