Call_Backs_Assignment

August 18, 2020

Source

https://stackoverflow.com/questions/39779710/setting-up-a-learningratescheduler-in-keras

https://medium.com/@thongonary/how-to-compute-f1-score-for-each-epoch-in-keras-a1acd17715a2

https://stackoverflow.com/questions/41032551/how-to-compute-receiving-operating-characteristic-roc-and-auc-in-keras

How can I get both test accuracy and validation accuracy for each epoch https://github.com/keras-team/keras/issues/2548

Reduce Learning Rate :: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPl

https://stackoverflow.com/questions/51889378/how-to-use-keras-reducelronplateau

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&scope=email%20https%3a%2f%2fwww.googl
eapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%2
0https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2
fwww.googleapis.com%2fauth%2fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

[2]:
```
import os
os.chdir('/content/drive/My Drive/Applied AI Course/Assignments/20. Working␣
 ↪with Callbacks')
!pwd
```

```
/content/drive/My Drive/Applied AI Course/Assignments/20. Working with Callbacks
```

[3]:
```
import pandas
data = pandas.read_csv('data.csv')
```

[4]:
```
data.head()
```

[4]:
```
          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
```

[5]:
```
y = data['label'].values
X = data.drop(['label'], axis=1)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪stratify=y)
```

[6]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(13400, 2)
(13400,)
(6600, 2)
(6600,)
```

[7]:
```
## Importing libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
import datetime
from keras.callbacks import TensorBoard
```

[8]:
```
%load_ext tensorboard
```

[9]:
```
!mkdir Model_save
```

[11]:
```
!mkdir -p  logs/fit/
```

Updated the TerminateNaN class so that it handles weights as well.

[19]:
```
import numpy as np
class TerminateNaN(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    loss = logs.get('loss')
    weight = self.model.get_weights()
    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
    w_list = []
    for w in weight:
      w = np.any(np.isnan(w))
      w_list.append(w)
    nan_W = np.any(np.isnan(w_list))
    if nan_W:
      self.model.stop_training = True
```

Updated the class Metrices So that it takes probabaility of y to calculate the ROAUC

[16]:
```
#https://machinelearningmastery.com/
 ↪how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/
import numpy as np
from keras.callbacks import Callback
from sklearn import metrics
from sklearn.metrics import  f1_score,roc_auc_score
class Metrics(Callback):
  def on_train_begin(self, logs={}):
    self.val_f1s = []
    self.auc=[]

  def on_epoch_end(self, epoch, logs={}):
    yhat_probs = model.predict(X_test, verbose=0)
    # predict crisp classes for test set
    yhat_classes = self.model.predict(X_test, verbose=0).round()
    # reduce to 1d array
    yhat_probs = yhat_probs[:, 0]
    yhat_classes = yhat_classes[:, 0]

    f1 = f1_score(y_test, yhat_classes)
    print('F1 score: %f' % f1)
    # ROC AUC
    auc = roc_auc_score(y_test, yhat_classes.round())
    print('ROC AUC: %f' % auc)
```

[14]:
```
class LossHistory(tf.keras.callbacks.Callback):
  def on_train_begin(self, logs={}):
    self.history={'loss': [],'acc': [],'val_loss': [],'val_accuracy': []}

  def on_epoch_end(self, epoch, logs={}):
    self.history['loss'].append(logs.get('loss'))
    self.history['acc'].append(logs.get('acc'))
    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_accuracy'].append(logs.get('val_accuracy'))
```

```
history_own=LossHistory()
```

[15]:
```
# if the eopch is multiple of 3, then the learning rate should decrease by 5%
def changeLearningRate(epoch,lr):
  if((epoch+1)%3==0):
    lr =lr - (lr*0.05)
  return lr
```

1 Model_1

[17]:
```
!ls
```

```
Call_Backs_Assignment.ipynb  data.csv  Model_save
Call_Backs_Reference.ipynb   logs      tensorboard.ipynb
```

[43]:
```
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler
import keras.backend as K
K.clear_session()

# Input layer
input_layer = Input(shape=(2,))
#Dense hidden layer
layer1 = Dense(50, activation='tanh', kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0., maxval=1.))(input_layer)
layer2 = Dense(50, activation='tanh', kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0., maxval=1.))(layer1)
layer3 = Dense(50, activation='tanh', kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0., maxval=1.))(layer2)
layer4 = Dense(50, activation='tanh', kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0., maxval=1.))(layer3)
layer5 = Dense(50, activation='tanh', kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0., maxval=1.))(layer4)
output = Dense(1, activation='sigmoid', kernel_initializer=tf.keras.
 ↪initializers.RandomUniform(minval=0., maxval=1.))(layer5)

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

# Callbacks
p_metrics = Metrics()
```

```
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.1,␣
 ↪nesterov=False)
reduce_lr1 = ReduceLROnPlateau(monitor='val_loss', factor=0.1)
reduce_lr2 = LearningRateScheduler(changeLearningRate, verbose=1)
es = EarlyStopping(monitor='val_accuracy', patience=2, mode='max')
nan_val = TerminateNaN()

model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])


filepath="/content/drive/My Drive/Applied AI Course/Assignments/20. Working␣
 ↪with Callbacks/Model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', ␣
 ↪verbose=1, save_best_only=True, mode='max')

model.compile(optimizer=optimizer,␣
 ↪loss='binary_crossentropy',metrics=['accuracy'])

tensorboard_callback = tf.keras.callbacks.
 ↪TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,␣
 ↪write_grads=True)


log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

call_back=[history_own,checkpoint,metricss,earlystop,reduce_lr,terminate_nan,cbks,tensorboard_

cb = [p_metrics, reduce_lr1, reduce_lr2, es, nan_val, tensorboard_callback]


model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test),␣
 ↪batch_size=16, callbacks=cb)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 1/10
  2/838 [...] - ETA: 1:11 - loss: 12.7370 - accuracy:
0.4688WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0020s vs `on_train_batch_end` time: 0.1699s).
Check your callbacks.
801/838 [===========================>..] - ETA: 0s - loss: 1.2903 - accuracy:
0.5158F1 score: 0.654944
ROC AUC: 0.542121
```

```
838/838 [==============================] - 2s 2ms/step - loss: 1.2637 -
accuracy: 0.5184 - val_loss: 0.6878 - val_accuracy: 0.5421

Epoch 00002: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 2/10
823/838 [============================>.] - ETA: 0s - loss: 0.6895 - accuracy:
0.5339F1 score: 0.654944
ROC AUC: 0.542121
838/838 [==============================] - 2s 2ms/step - loss: 0.6894 -
accuracy: 0.5348 - val_loss: 0.6936 - val_accuracy: 0.5421

Epoch 00003: LearningRateScheduler reducing learning rate to
0.009499999787658453.
Epoch 3/10
801/838 [===========================>..] - ETA: 0s - loss: 0.6887 - accuracy:
0.5352F1 score: 0.654944
ROC AUC: 0.542121
838/838 [==============================] - 2s 2ms/step - loss: 0.6888 -
accuracy: 0.5351 - val_loss: 0.6871 - val_accuracy: 0.5421
```

[43]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff3744dc588>
```

[45]:
```
!kill 2280
```

[47]:
```
%tensorboard --logdir logs/fit
```

```
Output hidden; open in https://colab.research.google.com to view.
```

2 Model_2

[22]:
```
 %reload_ext tensorboard
```

[23]:
```
K.clear_session()
```

[49]:
```
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler

input_layer = Input(shape=(2,))

#Dense hidden layer
layer1 = Dense(50,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=30))(input_layer)
```

```
layer2 = Dense(50,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=30))(layer1)
layer3 = Dense(50,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=30))(layer2 )
layer4 = Dense(50,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=30))(layer3)
layer5= Dense(50,activation='relu',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=30))(layer4)

#output layer
output = Dense(1,activation='softmax',kernel_initializer=tf.keras.initializers.
 ↪RandomUniform(minval=0,maxval=1,seed=0))(layer5)

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

# Callbacks
p_metrics = Metrics()

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.1,␣
 ↪nesterov=False)
reduce_lr1 = ReduceLROnPlateau(monitor='val_loss', factor=0.1)
reduce_lr2 = LearningRateScheduler(changeLearningRate, verbose=1)
es = EarlyStopping(monitor='val_accuracy', patience=2, mode='max')
nan_val = TerminateNaN()

model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])


filepath="/content/drive/My Drive/Applied AI Course/Assignments/20. Working␣
 ↪with Callbacks/Model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', ␣
 ↪verbose=1, save_best_only=True, mode='max')

model.compile(optimizer=optimizer,␣
 ↪loss='binary_crossentropy',metrics=['accuracy'])

tensorboard_callback = tf.keras.callbacks.
 ↪TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,␣
 ↪write_grads=True)


log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

call_back=[history_own,checkpoint,metricss,earlystop,reduce_lr,terminate_nan,cbks,tensorboard_
```

```
cb = [p_metrics, reduce_lr1, reduce_lr2, es, nan_val, tensorboard_callback]


model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test),␣
 ↪batch_size=16, callbacks=cb)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 1/10
  2/838 [...] - ETA: 45s - loss: 9.0542 - accuracy:
0.4062WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0048s vs `on_train_batch_end` time: 0.1056s).
Check your callbacks.
836/838 [============================>.] - ETA: 0s - loss: 7.6246 - accuracy:
0.5000F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000

Epoch 00002: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 2/10
810/838 [============================>..] - ETA: 0s - loss: 7.6152 - accuracy:
0.5006F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000

Epoch 00003: LearningRateScheduler reducing learning rate to
0.009499999787658453.
Epoch 3/10
831/838 [============================>.] - ETA: 0s - loss: 7.6223 - accuracy:
0.5002F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000
```

[49]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff37363c9b0>
```

[51]:
```
!kill 2280
```

[52]:
```
%tensorboard --logdir logs/fit
```

```
Output hidden; open in https://colab.research.google.com to view.
```

3 Model_3

[28]:
```
K.clear_session()
```

[55]:
```
#Dense hidden layer
layer1 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(input_layer)
layer2 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer1)
layer3 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer2 )
layer4 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer3)
layer5= Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer4)

#output layer
output = Dense(1,activation='softmax',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer5)

# Callbacks
p_metrics = Metrics()

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.1,␣
 ↪nesterov=False)
reduce_lr1 = ReduceLROnPlateau(monitor='val_loss', factor=0.1)
reduce_lr2 = LearningRateScheduler(changeLearningRate, verbose=1)
es = EarlyStopping(monitor='val_accuracy', patience=2, mode='max')
nan_val = TerminateNaN()

model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])


filepath="/content/drive/My Drive/Applied AI Course/Assignments/20. Working␣
 ↪with Callbacks/Model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', ␣
 ↪verbose=1, save_best_only=True, mode='max')

model.compile(optimizer=optimizer,␣
 ↪loss='binary_crossentropy',metrics=['accuracy'])

tensorboard_callback = tf.keras.callbacks.
 ↪TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,␣
 ↪write_grads=True)
```

```
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

call_back=[history_own,checkpoint,metricss,earlystop,reduce_lr,terminate_nan,cbks,tensorboard_

cb = [p_metrics, reduce_lr1, reduce_lr2, es, nan_val, tensorboard_callback]


model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test),␣
 ↪batch_size=16, callbacks=cb)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 1/10
  2/838 [...] - ETA: 54s - loss: 7.1481 - accuracy:
0.5312WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0027s vs `on_train_batch_end` time: 0.1292s).
Check your callbacks.
798/838 [===========================>..] - ETA: 0s - loss: 7.6449 - accuracy:
0.4987F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000

Epoch 00002: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 2/10
816/838 [============================>.] - ETA: 0s - loss: 7.6129 - accuracy:
0.5008F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000

Epoch 00003: LearningRateScheduler reducing learning rate to
0.009949999787658453.
Epoch 3/10
822/838 [============================>.] - ETA: 0s - loss: 7.6304 - accuracy:
0.4996F1 score: 0.666667
ROC AUC: 0.500000
838/838 [==============================] - 2s 2ms/step - loss: 7.6246 -
accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000
```

[55]: <tensorflow.python.keras.callbacks.History at 0x7ff36cf15d30>

[59]:
```
!kill 2664
```

```
/bin/bash: line 0: kill: (2664) - No such process
```

[60]:
```
%tensorboard --logdir logs/fit
```

```
Output hidden; open in https://colab.research.google.com to view.
```

4 Model_4

[66]:
```
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler
import keras.backend as K
K.clear_session()

input_layer = Input(shape=(2,))

#Dense hidden layer
layer1 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(input_layer)
layer2 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer1)
layer3 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer2 )
layer4 = Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer3)
layer5= Dense(50,activation='relu',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer4)


#output layer
output = Dense(1,activation='tanh',kernel_initializer=tf.compat.v2.keras.
 ↪initializers.he_normal(seed=30))(layer5)

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

# Callbacks
p_metrics = Metrics()

optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.1,␣
 ↪nesterov=False)
reduce_lr1 = ReduceLROnPlateau(monitor='val_loss', factor=0.1)
```

```
reduce_lr2 = LearningRateScheduler(changeLearningRate, verbose=1)
es = EarlyStopping(monitor='val_accuracy', patience=2, mode='max')
nan_val = TerminateNaN()

model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])


filepath="/content/drive/My Drive/Applied AI Course/Assignments/20. Working␣
 ↪with Callbacks/Model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', ␣
 ↪verbose=1, save_best_only=True, mode='max')

model.compile(optimizer=optimizer,␣
 ↪loss='binary_crossentropy',metrics=['accuracy'])

tensorboard_callback = tf.keras.callbacks.
 ↪TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True,␣
 ↪write_grads=True)


log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

call_back=[history_own,checkpoint,metricss,earlystop,reduce_lr,terminate_nan,cbks,tensorboard_

cb = [p_metrics, reduce_lr1, reduce_lr2, es, nan_val, tensorboard_callback]


model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test),␣
 ↪batch_size=16, callbacks=cb)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 1/10
  2/838 [...] - ETA: 50s - loss: 5.5204 - accuracy:
0.5938WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0025s vs `on_train_batch_end` time: 0.1169s).
Check your callbacks.
834/838 [============================>.] - ETA: 0s - loss: 0.6949 - accuracy:
0.5989F1 score: 0.688049
ROC AUC: 0.585909
838/838 [==============================] - 2s 2ms/step - loss: 0.6947 -
accuracy: 0.5990 - val_loss: 0.6615 - val_accuracy: 0.5859
```

```
Epoch 00002: LearningRateScheduler reducing learning rate to
0.009999999776482582.
Epoch 2/10
819/838 [============================>.] - ETA: 0s - loss: 0.6706 - accuracy:
0.6371F1 score: 0.638985
ROC AUC: 0.668030
838/838 [==============================] - 2s 2ms/step - loss: 0.6701 -
accuracy: 0.6363 - val_loss: 0.6612 - val_accuracy: 0.6680

Epoch 00003: LearningRateScheduler reducing learning rate to
0.009499999787658453.
Epoch 3/10
795/838 [===========================>..] - ETA: 0s - loss: 0.6784 - accuracy:
0.6272F1 score: 0.638284
ROC AUC: 0.655000
838/838 [==============================] - 2s 2ms/step - loss: 0.6770 -
accuracy: 0.6282 - val_loss: 0.6737 - val_accuracy: 0.6550

Epoch 00004: LearningRateScheduler reducing learning rate to
0.009499999694526196.
Epoch 4/10
831/838 [============================>.] - ETA: 0s - loss: 0.6330 - accuracy:
0.6513F1 score: 0.671544
ROC AUC: 0.619091
838/838 [==============================] - 2s 2ms/step - loss: 0.6334 -
accuracy: 0.6508 - val_loss: 0.6526 - val_accuracy: 0.6191
```

[66]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff36ce21828>
```

[69]:
```
!kill 3300
```

[70]:
```
%tensorboard --logdir logs/fit
```

```
Output hidden; open in https://colab.research.google.com to view.
```

5 Comparing Models

[71]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model number", "Epoch", "Loss", "Accuracy", "Val loss", "Val␣
 ↪accuracy", "f1-score", "auc"]
x.add_row([1, 3, 0.6888, 0.5351,0.6871, 0.5421,0.654944, 0.542121])
x.add_row([2, 3, 7.6246, 0.5000, 7.6246,0.5000,0.666667, 0.500000])
x.add_row([3, 3, 7.6246, 0.5000, 7.6246,0.5000,0.666667, 0.500000])
x.add_row([4, 4, 0.6334, 0.6508, 0.6526,0.6191,0.671544, 0.619091])
print(x)
```

Model number	Epoch	Loss	Accuracy	Val loss	Val accuracy	f1-score	auc
1	3	0.6888	0.5351	0.6871	0.5421	0.654944	0.542121
2	3	7.6246	0.5	7.6246	0.5	0.666667	0.5
3	3	7.6246	0.5	7.6246	0.5	0.666667	0.5
4	4	0.6334	0.6508	0.6526	0.6191	0.671544	0.619091

Observations

In model 4 we have kept number of dense layers and activation units "rule" but just by changing the output layer activation unit to tanh and he_normal kernel, we can observe improvement in accuracy

6 Observations

Model-1

As number of epochs increases train accuracy increases which is a good sign (0.4688 to 0.5351), and for loss, as epoch increases train loss decreases significantly (1.2903 to 0.6888) but validation loss decreases slightly (0.6878 to 0.6871), which is significantly good with tanh activation functions.

But Validation accuracy remains constant - 0.5421

Model -2

As number of epochs increases accuracy and validation loss also doesn't decrease, which is not Good. For this model I used relu as an activation function. loss: 7.6246 - accuracy: 0.5000 - val_loss: 7.6246 - val_accuracy: 0.5000

Model-3

As number of epochs increases accuracy decreases, which is not good. and validation loss also doesn't decrease. For this model I used I used relu activation function with he initializer and SGD with momentum

Model-4

As number of epochs increases accuracy increases and for loss, as epochs increases both the loss decreases significantly. This is best model out of 4 ,in which I used relu activation function with he initializer and SGD with momentum but changes the output layer activation unit to tanh. loss: 0.6334 - accuracy: 0.6508 - val_loss: 0.6526 - val_accuracy: 0.6191