Working with Callbacks Assignment

Importing Libraries

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import ReduceLROnPlateau

from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras import *
from tensorflow.keras.utils import *

from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import train_test_split
import os, datetime
```

```
import pandas as pd
from google.colab import files
files = files.upload()
```

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

```
Saving data.csv to data.csv
```

```
df = pd.read_csv("data.csv")
df.head(5)
```

```
         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
```

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

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25)
```

```
def scheduler(epoch,lr):
    if (epoch+1) % 3 == 0:
        lr = 0.95 * lr
        return lr
    else:
        return lr

schedule = tf.keras.callbacks.LearningRateScheduler(scheduler,
verbose=1)

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

    def on_epoch_end(self, epoch, logs={}):
        loss = logs.get('loss')
        model_weights = self.model.get_weights()
        if model_weights is not None:
          if np.any([np.any(np.isnan(x)) for x in model_weights]):
            print("Invalid weights and terminated at epoch
{}".format(epoch))
            self.model.stop_training = True
        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()

class Metrics(tf.keras.callbacks.Callback):
    def __init__(self,x,y):
        self.y=y
        self.x=x
    def on_epoch_end(self,epoch,log={}):
        pred=self.model.predict(self.x)

log['micro_f1_score']=f1_score(self.y,pred.round(0),average='micro')
        log['auc_score']=roc_auc_score(self.y,pred)

metrics = Metrics(x,y)

filepath="{epoch:02d}-{val_loss:.2f}.hdf5"
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=filepath,
monitor='val_loss', verbose=1,
                             save_best_only=True, mode='auto')

Early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
min_delta=0.25, patience=1, verbose=1)

Reduce_Plateau =
tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy',
factor=0.9, patience=1, min_lr=0.0001)

%reload_ext tensorboard
logdir = "/content/logs20221225-10310620221226-051145" +
datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)

print('storing Logs for tensorboard in following directory :
',os.getcwd())
```

```
storing Logs for tensorboard in following directory :  /content
```

Model-1

1. Use tanh as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use RandomUniform(0,1) as initilizer.
4. Analyze your output and training process.

```
inputs=Input(x.shape[1],name='Input_Layer')
#Dense hidden layer
layer0=Dense(10,activation='tanh',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense1")(inputs)
layer1=Dense(10,activation='tanh',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense2")(layer0)
layer2=Dense(10,activation='tanh',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense3")(layer1)
layer3=Dense(10,activation='tanh',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense4")(layer2)
layer4=Dense(10,activation='tanh',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense5")(layer3)
outputs=Dense(1,name='output_layer',kernel_initializer=tf.keras.initia
lizers.RandomUniform(minval=0.,maxval=1.),activation='sigmoid')
(layer4)
model=Model(inputs,outputs)
model.compile(optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.1, nesterov=False,
name="SGD"),loss='binary_crossentropy',metrics=['accuracy'])

model.fit(x,y,callbacks=[checkpoint,Early_stop,metrics,Reduce_Plateau,
terminate,tensorboard,schedule],epochs=30,validation_split=0.2)
```

```
Epoch 1: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 1/30
  1/500 [..............................] - ETA: 6:10 - loss: 1.4149 -
accuracy: 0.6875

WARNING:tensorflow:Callback method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0019s vs
`on_train_batch_end` time: 0.0022s). Check your callbacks.

488/500 [============================>.] - ETA: 0s - loss: 0.8158 -
accuracy: 0.4966
```

```
Epoch 1: val_loss improved from inf to 0.69353, saving model to 01-
0.69.hdf5
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 4s 6ms/step - loss: 0.8128
- accuracy: 0.4971 - val_loss: 0.6935 - val_accuracy: 0.4852 -
micro_f1_score: 0.4999 - auc_score: 0.5046 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 2/30
473/500 [===========================>..] - ETA: 0s - loss: 0.6935 -
accuracy: 0.5020
Epoch 2: val_loss did not improve from 0.69353
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 2s 5ms/step - loss: 0.6935
- accuracy: 0.5019 - val_loss: 0.6937 - val_accuracy: 0.4857 -
micro_f1_score: 0.5003 - auc_score: 0.4876 - lr: 0.0100
Epoch 2: early stopping
```

```
<keras.callbacks.History at 0x7f946a56c790>
```

```
%tensorboard --logdir /content/logs20221225-10310620221226-051145
```

```
Reusing TensorBoard on port 6007 (pid 2760), started 0:42:44 ago. (Use
'!kill 2760' to kill it.)
```

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

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

sc1.png

sc2.png

sc3.png

sc4.png

sc5.png

sc6.png

sc7.png

Observation from Model 1: Using RandomUniform as an initializer, Momentum based SGD as optimizer and tanh as an activation function, We can say

1. Train accuracy is 0.5019 and validation accuracy is 0.4857
2. Train loss is 0.6935 and validation loss is 0.6937.
3. Hence , accuracy is increasing after each opch.

4. Hence, Train Loss decreased sharply from 0.8128 to 0.6935 from first epoch.

5. But validation loss increased after first epoch.

Model-2

1. Use relu as an activation for every layer except output layer.
2. use SGD with momentum as optimizer.
3. use RandomUniform(0,1) as initilizer.
4. Analyze your output and training process.

```
inputs=Input(x.shape[1],name='Input_Layer')
#Dense hidden layer
layer0=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense1")(inputs)
layer1=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense2")(layer0)
layer2=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense3")(layer1)
layer3=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense4")(layer2)
layer4=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.RandomUniform(minval=0., maxval=1.),name="Dense5")(layer3)
outputs=Dense(1,name='output_layer',kernel_initializer=tf.keras.initia
lizers.RandomUniform(minval=0.,maxval=1.),activation='sigmoid')
(layer4)
model=Model(inputs,outputs)
model.compile(optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.1, nesterov=False,
name="SGD"),loss='binary_crossentropy',metrics=['accuracy'])

model.fit(x,y,callbacks=[checkpoint,Early_stop,metrics,Reduce_Plateau,
terminate,tensorboard,schedule],epochs=30,validation_split=0.2)
```

```
Epoch 1: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 1/30
  1/500 [..............................] - ETA: 3:41 - loss: 248.6832
- accuracy: 0.4688

WARNING:tensorflow:Callback method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0015s vs
`on_train_batch_end` time: 0.0022s). Check your callbacks.

490/500 [============================>.] - ETA: 0s - loss: 1.1989 -
accuracy: 0.5403
Epoch 1: val_loss improved from 0.69341 to 0.68288, saving model to
01-0.68.hdf5
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 3s 5ms/step - loss: 1.1886
```

```
- accuracy: 0.5396 - val_loss: 0.6829 - val_accuracy: 0.5565 -
micro_f1_score: 0.5463 - auc_score: 0.5499 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 2/30
491/500 [============================>.] - ETA: 0s - loss: 0.6832 -
accuracy: 0.5425
Epoch 2: val_loss did not improve from 0.68288
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 2s 5ms/step - loss: 0.6834
- accuracy: 0.5429 - val_loss: 0.6838 - val_accuracy: 0.5575 -
micro_f1_score: 0.5469 - auc_score: 0.5499 - lr: 0.0100
Epoch 2: early stopping
```

```
<keras.callbacks.History at 0x7f946a33c610>
```

```
%tensorboard --logdir /content/logs20221225-10310620221226-051145
```

```
Reusing TensorBoard on port 6007 (pid 2760), started 0:44:17 ago. (Use
'!kill 2760' to kill it.)
```

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

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

sc-1.png

sc10.png

sc11.png

sc12.png

sc13.png

sc14.png

Observation from Model 2: Using RandomUniform as an initializer, Momentum based SGD as optimizer and RELU as an activation function, We can say,

1. Train accuracy is 0.5429 and validation accuracy is 0.5575
2. Train loss is 0.6834 and validation loss is 0.6838.
3. Hence , Train and Validation accuracy is increasing after each opch.
4. Hence, Train Loss decreased sharply from 1.1886 to 0.6834 from first epoch.
5. But validation loss increased after first epoch.
6. By using Relu activation instead of Tanh activation, there is improvement in accuracy and loss with same initializer and optimizer.

Model-3

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

```
inputs=Input(x.shape[1],name='Input_Layer')
#Dense hidden layer
layer0=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.he_uniform(),name="Dense1")(inputs)
layer1=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.he_uniform(),name="Dense2")(layer0)
layer2=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.he_uniform(),name="Dense3")(layer1)
layer3=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.he_uniform(),name="Dense4")(layer2)
layer4=Dense(10,activation='relu',kernel_initializer=tf.keras.initiali
zers.he_uniform(),name="Dense5")(layer3)
outputs=Dense(1,name='output_layer',kernel_initializer=tf.keras.initia
lizers.he_uniform(),activation='sigmoid')(layer4)
model=Model(inputs,outputs)
model.compile(optimizer=tf.keras.optimizers.SGD(
    learning_rate=0.01, momentum=0.1, nesterov=False,
name="SGD"),loss='binary_crossentropy',metrics=['accuracy'])

model.fit(x,y,callbacks=[checkpoint,Early_stop,metrics,Reduce_Plateau,
terminate,tensorboard,schedule],epochs=30,validation_split=0.2)
```

```
Epoch 1: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 1/30
  1/500 [..............................] - ETA: 3:53 - loss: 0.6576 -
accuracy: 0.5000

WARNING:tensorflow:Callback method `on_train_batch_end` is slow
compared to the batch time (batch time: 0.0018s vs
`on_train_batch_end` time: 0.0022s). Check your callbacks.

499/500 [============================>.] - ETA: 0s - loss: 0.6780 -
accuracy: 0.5654
Epoch 1: val_loss did not improve from 0.63915
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 3s 6ms/step - loss: 0.6780
- accuracy: 0.5652 - val_loss: 0.6727 - val_accuracy: 0.5922 -
micro_f1_score: 0.5978 - auc_score: 0.6558 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to
0.009999999776482582.
Epoch 2/30
```

```
484/500 [============================>.] - ETA: 0s - loss: 0.6621 -
accuracy: 0.6177
Epoch 2: val_loss did not improve from 0.63915
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 2s 4ms/step - loss: 0.6619
- accuracy: 0.6184 - val_loss: 0.6564 - val_accuracy: 0.6248 -
micro_f1_score: 0.6353 - auc_score: 0.6919 - lr: 0.0100
Epoch 2: early stopping
```

```
<keras.callbacks.History at 0x7f740fb95df0>
```

```
%tensorboard --logdir /content/logs20221225-103106
```

```
Reusing TensorBoard on port 6006 (pid 2115), started 0:55:45 ago. (Use
'!kill 2115' to kill it.)
```

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

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

sc21.png

sc22.png

sc23.png

sc24.png

sc25.png

sc26.png

sc27.png

Observation from Model 3: Using He_Uniform as an initializer, Momentum based SGD as optimizer and RELU as an activation function, We can say

1. Train accuracy is 0.6184 and validation accuracy is 0.6248

2. Train loss is 0.6619 and validation loss is 0.6564.

3. Hence , accuracy is increasing after each opch.

4. Hence, Train Loss decreased sharply from 0.8674 to 0.6934 from first epoch.

5. By using He_uniform instead of RandomUniform, there is high improvement in accuracy and loss.

6. Till now this combination of Relu, Momentum SGD and He_uniform shows best among all above model.

Model-4

1. Try with adam optimizer and relu activation function with he initlization to get better accuracy/f1 score.

```
inputs=Input(x.shape[1],name='Input_Layer')
layer0=Dense(10,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(),name="Dense1")(inputs)
layer1=Dense(10,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(),name='Dense2')(layer0)
layer2=Dense(10,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(),name='Dense3')(layer1)
layer3=Dense(10,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(),name='Dense4')(layer2)
layer4=Dense(10,activation='relu',kernel_initializer=tf.keras.initializers.he_uniform(),name='Dense5')(layer3)
outputs=Dense(1,name='output_layer',kernel_initializer=tf.keras.initializers.he_uniform(),activation='sigmoid')(layer4)
model=Model(inputs,outputs)
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
model.fit(x,y,callbacks=[checkpoint,Early_stop,metrics,Reduce_Plateau,terminate,tensorboard,schedule],epochs=30,validation_split=0.2)
```

```

Epoch 1: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 1/30
497/500 [============================>.] - ETA: 0s - loss: 0.6830 - accuracy: 0.5409
Epoch 1: val_loss did not improve from 0.63915
625/625 [==============================] - 1s 2ms/step
500/500 [==============================] - 5s 8ms/step - loss: 0.6829 - accuracy: 0.5410 - val_loss: 0.6638 - val_accuracy: 0.6280 - micro_f1_score: 0.6154 - auc_score: 0.6677 - lr: 0.0010

Epoch 2: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 2/30
496/500 [============================>.] - ETA: 0s - loss: 0.6391 - accuracy: 0.6368
Epoch 2: val_loss improved from 0.63915 to 0.63027, saving model to 02-0.63.hdf5
625/625 [==============================] - 1s 1ms/step
500/500 [==============================] - 3s 5ms/step - loss: 0.6390 - accuracy: 0.6369 - val_loss: 0.6303 - val_accuracy: 0.6505 - micro_f1_score: 0.6449 - auc_score: 0.7060 - lr: 0.0010
Epoch 2: early stopping
```

```
<keras.callbacks.History at 0x7f740fcd9e80>
```

```
%tensorboard --logdir /content/logs20221225-103106
```

```
Reusing TensorBoard on port 6006 (pid 2115), started 0:58:50 ago. (Use
'!kill 2115' to kill it.)
```

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

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

sc31.png

sc32.png

sc33.png

sc34.png

sc35.png

sc36.png

sc37.png

Observation from Model 4: Using He_Uniform as an initializer, ADAM as optimizer and RELU as an activation function, We can say

1. Train accuracy is 0.6369 and validation accuracy is 0.6505
2. Train loss is 0.6390 and validation loss is 0.6303.
3. Hence , accuracy is increasing after each epoch.
4. Hence, Train Loss decreased sharply from 0.8674 to 0.6934 from first epoch.
5. Till now this combination of Relu, Momentum SGD and He_uniform shows best among all above model.