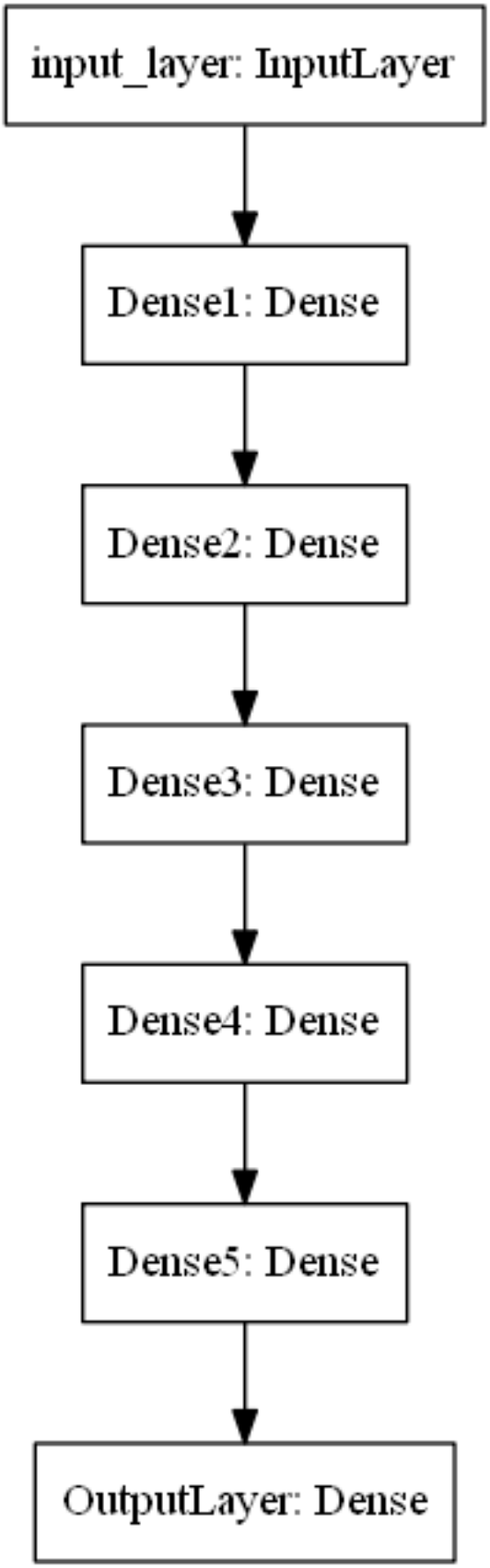
TensorFlow-CallBacks

October 9, 2022

1. Download the data from here. You have to use data.csv file for this assignment
2. Code the model to classify data like below image. You can use any number of units in your Dense layers.


input_layer: InputLayer
Dense1: Dense
Dense2: Dense
Dense3: Dense
Dense4: Dense
Dense5: Dense
OutputLayer: Dense

3. Writing Callbacks

- Write your own callback function, that has to print the micro F1 score and AUC score after each epoch. Do not use tf.keras.metrics for calculating AUC and F1 score.
- Save your model at every epoch if your validation accuracy is improved from previous epoch.
- You have to decay learning based on below conditions
  - Cond1. If your validation accuracy at that epoch is less than previous epoch accuracy, you have to decrease the learning rate by 10%.
  - Cond2. For every 3rd epoch, decay your learning rate by 5%.
- If you are getting any NaN values(either weigths or loss) while training, you have to terminate your training.
- You have to stop the training if your validation accuracy is not increased in last 2 epochs.
- Use tensorboard for every model and analyse your scalar plots and histograms (you need to upload the screenshots and write the observations for each model for evaluation).

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
from google.colab import drive
drive.mount(mountpoint='/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
from IPython.display import display
```

[4]:
```
from matplotlib import pyplot as plt
from matplotlib import style
style.use(style='seaborn-deep')
```

[5]:
```
from sklearn.metrics import auc, f1_score, roc_curve
from sklearn.model_selection import train_test_split
```

[6]:
```
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, LearningRateScheduler,␣
↪ModelCheckpoint, ReduceLROnPlateau
```

[7]:
```
from tqdm import tqdm
```

[8]:
```
import datetime
import numpy as np
import os
import pandas as pd
import pickle
import seaborn as sns
```

```
import tensorflow as tf
```

[9]:
```
data_file = '/content/drive/MyDrive/Applied-AI/Assignment-20/data.csv'
```

[10]:
```
data_df = pd.read_csv(filepath_or_buffer=data_file)
display(data_df.head())
```

```
         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
```

[11]:
```
display(data_df.describe())
```

```
                f1            f2         label
count  20000.000000  20000.000000  20000.000000
mean       0.000630     -0.000745      0.500000
std        0.671165      0.674704      0.500013
min       -1.649781     -1.600645      0.000000
25%       -0.589878     -0.596424      0.000000
50%        0.001795     -0.003113      0.500000
75%        0.586631      0.597803      1.000000
max        1.629722      1.584291      1.000000
```

[12]:
```
display(data_df['label'].value_counts().to_frame())
```

```
     label
0.0  10000
1.0  10000
```

We have a balanced data.

[13]:
```
X = data_df.drop(columns=['label'])
y = data_df['label'].values

display(X.head())
print(y[:5])
```

```
         f1        f2
0  0.450564  1.074305
1  0.085632  0.967682
2  0.117326  0.971521
3  0.982179 -0.380408
4 -0.720352  0.955850

[0. 0. 1. 0. 0.]
```

[14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=0)
```

[15]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

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

Custom callbacks.

[16]:
```
class ScoreCallBack(tf.keras.callbacks.Callback):
    """
    This class is a custom callback for scores.
    """

    def __init__(self, validation_data):
        self.X_test = validation_data[0]
        self.y_test = validation_data[1]

    def on_epoch_end(self, epoch, logs={}):
        """
        This overrides the method present in parent class and prints F1 Score␣
↪and AUC Score on each epoch end.
        """
        y_pred_probs = self.model.predict(x=self.X_test)
        y_pred = np.argmax(a=y_pred_probs, axis=1)

        fpr, tpr, thresholds = roc_curve(y_true=self.y_test, y_score=y_pred)
        auc_score_val = auc(x=fpr, y=tpr)
        f1_score_val = f1_score(y_true=self.y_test, y_pred=y_pred)

        print("\nAUC: {} - F1-Score: {}".format(auc_score_val, f1_score_val))
```

[17]:
```
class TerminateCallBack(tf.keras.callbacks.Callback):
    """
    This class is a custom callback for termination if NaN.
    """

    def on_epoch_end(self, epoch, logs={}):
        """
        This overrides the method present in parent class and terminates if␣
↪loss is NaN.
        """
        loss = logs.get('loss')
        model_weights = self.model.get_weights()
        if ((loss is not None) or
            (model_weights is not None)):
```

```
            if (np.isnan(loss) or
                np.isinf(loss) or
                np.any([np.any(np.isnan(x)) for x in model_weights])):
                print("Invalid loss/weights and terminated at epoch {}.".
 ↪format(epoch))
                self.model.stop_training = True
```

[18]:
```
def scheduler(epoch, lr):
    """
    This function decreases the learning rate for every 3rd epoch by 5%.
    """
    if (epoch + 1) % 3 == 0:
        lr -= (lr * 0.05)
        return lr
    else:
        return lr
```

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.

[19]:
```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

[20]:
```
m1_ki = tf.keras.initializers.RandomUniform(minval=0, maxval=1)
```

[21]:
```
input_layer = Input(shape=(2,))
layer1 = Dense(units=20, activation='tanh',␣
 ↪kernel_initializer=m1_ki)(input_layer)
layer2 = Dense(units=15, activation='tanh', kernel_initializer=m1_ki)(layer1)
layer3 = Dense(units=10, activation='tanh', kernel_initializer=m1_ki)(layer2)
layer4 = Dense(units=5, activation='tanh', kernel_initializer=m1_ki)(layer3)
layer5 = Dense(units=2, activation='tanh', kernel_initializer=m1_ki)(layer4)
output_layer = Dense(units=1, activation='sigmoid',␣
 ↪kernel_initializer=m1_ki)(layer5)
```

[22]:
```
model1 = Model(inputs=input_layer, outputs=output_layer)
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
model1.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy', 'AUC'])
```

[23]:
```
score_callback = ScoreCallBack(validation_data=(X_test, y_test))

termination_callback = TerminateCallBack()
```

```
decay_lr_callback = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,␣
 ↪patience=1, verbose=1, mode='auto')

lr_schedule_callback = LearningRateScheduler(schedule=scheduler, verbose=1)

filepath = "model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
model_save_callback = ModelCheckpoint(filepath=filepath,␣
 ↪monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto')

early_stop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.01,␣
 ↪patience=2, verbose=1)

log_dir = os.path.join('logs', 'fits', datetime.datetime.now().
 ↪strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)

callbacks = [score_callback, termination_callback,
             decay_lr_callback, lr_schedule_callback,
             model_save_callback, early_stop_callback, tensorboard_callback]
```

[24]:
```
model1.fit(x=X_train, y=y_train, epochs=5, validation_data=(X_test, y_test),␣
 ↪batch_size=20, callbacks=callbacks)
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/5
  1/670 […] - ETA: 18:40 - loss: 0.9227 - accuracy:
0.5000 - auc: 0.4835

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0044s vs `on_train_batch_end` time: 0.0134s). Check
your callbacks.

663/670 [============================>.] - ETA: 0s - loss: 0.6998 - accuracy:
0.5020 - auc: 0.4996
AUC: 0.5 - F1-Score: 0.0

Epoch 1: val_accuracy improved from -inf to 0.50485, saving model to
model_save/weights-01-0.5048.hdf5
670/670 [==============================] - 8s 9ms/step - loss: 0.6997 -
accuracy: 0.5026 - auc: 0.5000 - val_loss: 0.6934 - val_accuracy: 0.5048 -
val_auc: 0.5043 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/5
668/670 [============================>.] - ETA: 0s - loss: 0.6940 - accuracy:
```

```
0.5034 - auc: 0.4990
AUC: 0.5 - F1-Score: 0.0

Epoch 2: val_accuracy improved from 0.50485 to 0.50636, saving model to
model_save/weights-02-0.5064.hdf5
670/670 [==============================] - 6s 9ms/step - loss: 0.6940 -
accuracy: 0.5032 - auc: 0.4987 - val_loss: 0.6934 - val_accuracy: 0.5064 -
val_auc: 0.5012 - lr: 0.0100

Epoch 3: LearningRateScheduler setting learning rate to 0.009499999787658453.
Epoch 3/5
666/670 [============================>.] - ETA: 0s - loss: 0.6941 - accuracy:
0.4953 - auc: 0.4908
AUC: 0.5 - F1-Score: 0.0

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0009499999694526196.

Epoch 3: val_accuracy did not improve from 0.50636
670/670 [==============================] - 7s 10ms/step - loss: 0.6941 -
accuracy: 0.4952 - auc: 0.4911 - val_loss: 0.6935 - val_accuracy: 0.4995 -
val_auc: 0.4999 - lr: 9.5000e-04
Epoch 3: early stopping
```

[24]:
```
<keras.callbacks.History at 0x7f0ddf3dae10>
```

[25]:
```
%tensorboard --logdir logs/fits
```

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

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.

[26]:
```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

[27]:
```
m2_ki = tf.keras.initializers.RandomUniform(minval=0, maxval=1)
```

[28]:
```
input_layer = Input(shape=(2,))
layer1 = Dense(units=20, activation='relu',␣
 ↪kernel_initializer=m2_ki)(input_layer)
layer2 = Dense(units=15, activation='relu', kernel_initializer=m2_ki)(layer1)
layer3 = Dense(units=10, activation='relu', kernel_initializer=m2_ki)(layer2)
```

```
layer4 = Dense(units=5, activation='relu', kernel_initializer=m2_ki)(layer3)
layer5 = Dense(units=2, activation='relu', kernel_initializer=m2_ki)(layer4)
output_layer = Dense(units=1, activation='sigmoid',␣
 ↪kernel_initializer=m2_ki)(layer5)
```

[29]:
```
model2 = Model(inputs=input_layer, outputs=output_layer)
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
model2.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy', 'AUC'])
```

[30]:
```
score_callback = ScoreCallBack(validation_data=(X_test, y_test))

termination_callback = TerminateCallBack()

decay_lr_callback = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,␣
 ↪patience=1, verbose=1, mode='auto')

lr_schedule_callback = LearningRateScheduler(schedule=scheduler, verbose=1)

filepath = "model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
model_save_callback = ModelCheckpoint(filepath=filepath,␣
 ↪monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto')

early_stop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.01,␣
 ↪patience=2, verbose=1)

log_dir = os.path.join('logs', 'fits', datetime.datetime.now().
 ↪strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)

callbacks = [score_callback, termination_callback,
             decay_lr_callback, lr_schedule_callback,
             model_save_callback, early_stop_callback, tensorboard_callback]
```

[31]:
```
model2.fit(x=X_train, y=y_train, epochs=5, validation_data=(X_test, y_test),␣
 ↪batch_size=20, callbacks=callbacks)
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/5
  1/670 […] - ETA: 7:46 - loss: 115.2181 -
accuracy: 0.4000 - auc: 0.4495

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0016s vs `on_train_batch_end` time: 0.0035s). Check
your callbacks.
```

```
649/670 [============================>.] - ETA: 0s - loss: 0.9067 - accuracy:
0.4995 - auc: 0.4969
AUC: 0.5 - F1-Score: 0.0

Epoch 1: val_accuracy improved from -inf to 0.50636, saving model to
model_save/weights-01-0.5064.hdf5
670/670 [==============================] - 3s 4ms/step - loss: 0.9001 -
accuracy: 0.4992 - auc: 0.4966 - val_loss: 0.6931 - val_accuracy: 0.5064 -
val_auc: 0.5000 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/5
644/670 [============================>..] - ETA: 0s - loss: 0.6935 - accuracy:
0.5031 - auc: 0.4989
AUC: 0.5 - F1-Score: 0.0

Epoch 2: ReduceLROnPlateau reducing learning rate to 0.0009999999776482583.

Epoch 2: val_accuracy did not improve from 0.50636
670/670 [==============================] - 3s 4ms/step - loss: 0.6935 -
accuracy: 0.5016 - auc: 0.4981 - val_loss: 0.6936 - val_accuracy: 0.4936 -
val_auc: 0.5000 - lr: 1.0000e-03

Epoch 3: LearningRateScheduler setting learning rate to 0.000949999934528023.
Epoch 3/5
667/670 [============================>.] - ETA: 0s - loss: 0.6932 - accuracy:
0.5015 - auc: 0.4992
AUC: 0.5 - F1-Score: 0.0

Epoch 3: ReduceLROnPlateau reducing learning rate to 9.499999578110874e-05.

Epoch 3: val_accuracy did not improve from 0.50636
670/670 [==============================] - 3s 4ms/step - loss: 0.6932 -
accuracy: 0.5012 - auc: 0.4991 - val_loss: 0.6932 - val_accuracy: 0.4936 -
val_auc: 0.5000 - lr: 9.5000e-05
Epoch 3: early stopping
```

[31]:
```
<keras.callbacks.History at 0x7f0ddb7c8750>
```

[32]:
```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 1993), started 0:26:43 ago. (Use '!kill␣
↪1993' to kill it.)
```

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

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.

[33]:
```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

[34]:
```
m3_ki = tf.keras.initializers.HeUniform()
```

[35]:
```
input_layer = Input(shape=(2,))
layer1 = Dense(units=20, activation='relu',␣
 ↪kernel_initializer=m3_ki)(input_layer)
layer2 = Dense(units=15, activation='relu', kernel_initializer=m3_ki)(layer1)
layer3 = Dense(units=10, activation='relu', kernel_initializer=m3_ki)(layer2)
layer4 = Dense(units=5, activation='relu', kernel_initializer=m3_ki)(layer3)
layer5 = Dense(units=2, activation='relu', kernel_initializer=m3_ki)(layer4)
output_layer = Dense(units=1, activation='sigmoid',␣
 ↪kernel_initializer=m3_ki)(layer5)
```

[36]:
```
model3 = Model(inputs=input_layer, outputs=output_layer)
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
model3.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy', 'AUC'])
```

[37]:
```
score_callback = ScoreCallBack(validation_data=(X_test, y_test))

termination_callback = TerminateCallBack()

decay_lr_callback = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,␣
 ↪patience=1, verbose=1, mode='auto')

lr_schedule_callback = LearningRateScheduler(schedule=scheduler, verbose=1)

filepath = "model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
model_save_callback = ModelCheckpoint(filepath=filepath,␣
 ↪monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto')

early_stop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.01,␣
 ↪patience=2, verbose=1)

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

```
callbacks = [score_callback, termination_callback,
             decay_lr_callback, lr_schedule_callback,
             model_save_callback, early_stop_callback, tensorboard_callback]
```

[38]:
```
model3.fit(x=X_train, y=y_train, epochs=5, validation_data=(X_test, y_test),␣
↪batch_size=20, callbacks=callbacks)
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 1/5
  1/670 […] - ETA: 7:36 - loss: 0.7161 - accuracy:
0.4500 - auc: 0.4050

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0024s vs `on_train_batch_end` time: 0.0033s). Check
your callbacks.

644/670 [===========================>..] - ETA: 0s - loss: 0.6489 - accuracy:
0.6180 - auc: 0.6707
AUC: 0.5 - F1-Score: 0.0

Epoch 1: val_accuracy improved from -inf to 0.66500, saving model to
model_save/weights-01-0.6650.hdf5
670/670 [==============================] - 3s 4ms/step - loss: 0.6489 -
accuracy: 0.6184 - auc: 0.6704 - val_loss: 0.6103 - val_accuracy: 0.6650 -
val_auc: 0.7307 - lr: 0.0100

Epoch 2: LearningRateScheduler setting learning rate to 0.009999999776482582.
Epoch 2/5
654/670 [============================>.] - ETA: 0s - loss: 0.6128 - accuracy:
0.6644 - auc: 0.7243
AUC: 0.5 - F1-Score: 0.0

Epoch 2: val_accuracy improved from 0.66500 to 0.66970, saving model to
model_save/weights-02-0.6697.hdf5
670/670 [==============================] - 2s 3ms/step - loss: 0.6128 -
accuracy: 0.6644 - auc: 0.7243 - val_loss: 0.6073 - val_accuracy: 0.6697 -
val_auc: 0.7317 - lr: 0.0100

Epoch 3: LearningRateScheduler setting learning rate to 0.009499999787658453.
Epoch 3/5
656/670 [============================>.] - ETA: 0s - loss: 0.6101 - accuracy:
0.6633 - auc: 0.7290
AUC: 0.5 - F1-Score: 0.0

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0009499999694526196.

Epoch 3: val_accuracy did not improve from 0.66970
670/670 [==============================] - 2s 3ms/step - loss: 0.6099 -
```

```
accuracy: 0.6636 - auc: 0.7294 - val_loss: 0.6098 - val_accuracy: 0.6642 -
val_auc: 0.7318 - lr: 9.5000e-04
Epoch 3: early stopping
```

[38]:
```
<keras.callbacks.History at 0x7f0ddc7f1890>
```

[39]:
```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 1993), started 0:37:00 ago. (Use '!kill␣
 ↪1993' to kill it.)
```

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

Model-4

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

[40]:
```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

[41]:
```
m4_ki = tf.keras.initializers.HeNormal()
```

[42]:
```
input_layer = Input(shape=(2,))
layer1 = Dense(units=20, activation='relu',␣
 ↪kernel_initializer=m4_ki)(input_layer)
layer2 = Dense(units=15, activation='relu', kernel_initializer=m4_ki)(layer1)
layer3 = Dense(units=10, activation='relu', kernel_initializer=m4_ki)(layer2)
layer4 = Dense(units=5, activation='relu', kernel_initializer=m4_ki)(layer3)
layer5 = Dense(units=2, activation='relu', kernel_initializer=m4_ki)(layer4)
output_layer = Dense(units=1, activation='sigmoid',␣
 ↪kernel_initializer=m4_ki)(layer5)
```

[43]:
```
model4 = Model(inputs=input_layer, outputs=output_layer)
optimizer = tf.keras.optimizers.Adam()
model4.compile(optimizer=optimizer, loss='binary_crossentropy',␣
 ↪metrics=['accuracy', 'AUC'])
```

[44]:
```
score_callback = ScoreCallBack(validation_data=(X_test, y_test))

termination_callback = TerminateCallBack()

decay_lr_callback = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,␣
 ↪patience=1, verbose=1, mode='auto')

lr_schedule_callback = LearningRateScheduler(schedule=scheduler, verbose=1)
```

```
filepath = "model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
model_save_callback = ModelCheckpoint(filepath=filepath,␣
 ↪monitor='val_accuracy', verbose=1, save_best_only=True, mode='auto')

early_stop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.01,␣
 ↪patience=2, verbose=1)

log_dir = os.path.join('logs', 'fits', datetime.datetime.now().
 ↪strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)

callbacks = [score_callback, termination_callback,
             decay_lr_callback, lr_schedule_callback,
             model_save_callback, early_stop_callback, tensorboard_callback]
```

[45]:
```
model4.fit(x=X_train, y=y_train, epochs=5, validation_data=(X_test, y_test),␣
 ↪batch_size=20, callbacks=callbacks)
```

```
Epoch 1: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 1/5
  1/670 […] - ETA: 12:10 - loss: 0.6720 - accuracy:
0.5000 - auc: 0.6450

WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0014s vs `on_train_batch_end` time: 0.0044s). Check
your callbacks.

654/670 [============================>.] - ETA: 0s - loss: 0.6732 - accuracy:
0.5880 - auc: 0.6580
AUC: 0.5 - F1-Score: 0.0

Epoch 1: val_accuracy improved from -inf to 0.64303, saving model to
model_save/weights-01-0.6430.hdf5
670/670 [==============================] - 4s 4ms/step - loss: 0.6724 -
accuracy: 0.5900 - auc: 0.6614 - val_loss: 0.6537 - val_accuracy: 0.6430 -
val_auc: 0.6960 - lr: 0.0010

Epoch 2: LearningRateScheduler setting learning rate to 0.0010000000474974513.
Epoch 2/5
654/670 [============================>.] - ETA: 0s - loss: 0.6316 - accuracy:
0.6632 - auc: 0.7222
AUC: 0.5 - F1-Score: 0.0

Epoch 2: val_accuracy improved from 0.64303 to 0.67136, saving model to
model_save/weights-02-0.6714.hdf5
670/670 [==============================] - 2s 3ms/step - loss: 0.6311 -
```

```
accuracy: 0.6637 - auc: 0.7227 - val_loss: 0.6215 - val_accuracy: 0.6714 -
val_auc: 0.7281 - lr: 0.0010

Epoch 3: LearningRateScheduler setting learning rate to 0.0009500000451225787.
Epoch 3/5
660/670 [============================>.] - ETA: 0s - loss: 0.6158 - accuracy:
0.6739 - auc: 0.7308
AUC: 0.5 - F1-Score: 0.0

Epoch 3: ReduceLROnPlateau reducing learning rate to 9.500000160187483e-05.

Epoch 3: val_accuracy did not improve from 0.67136
670/670 [==============================] - 2s 3ms/step - loss: 0.6156 -
accuracy: 0.6746 - auc: 0.7311 - val_loss: 0.6165 - val_accuracy: 0.6689 -
val_auc: 0.7274 - lr: 9.5000e-05

Epoch 4: LearningRateScheduler setting learning rate to 9.500000305706635e-05.
Epoch 4/5
670/670 [==============================] - ETA: 0s - loss: 0.6091 - accuracy:
0.6737 - auc: 0.7377
AUC: 0.5 - F1-Score: 0.0

Epoch 4: ReduceLROnPlateau reducing learning rate to 9.500000305706636e-06.

Epoch 4: val_accuracy did not improve from 0.67136
670/670 [==============================] - 2s 3ms/step - loss: 0.6091 -
accuracy: 0.6737 - auc: 0.7377 - val_loss: 0.6153 - val_accuracy: 0.6689 -
val_auc: 0.7267 - lr: 9.5000e-06
Epoch 4: early stopping
```

[45]:
```
<keras.callbacks.History at 0x7f0ddc502cd0>
```

[46]:
```
%tensorboard --logdir logs/fits
```

```
Reusing TensorBoard on port 6006 (pid 1993), started 0:47:17 ago. (Use '!kill␣
↪1993' to kill it.)
```

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

End of the file.