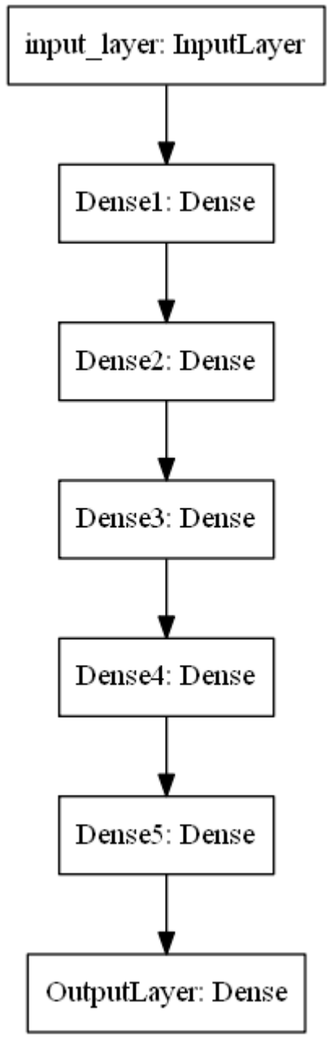
1. Download the data from here

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.

Model **1**

```
import tensorflow as tf
import datetime
import pandas as pd
import pickle
import numpy as np
import math
from tqdm import tqdm
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, recall_score, precision_score,roc_curve,auc
from keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from keras.optimizers import SGD
from keras.initializers import RandomUniform
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import TensorBoard
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
Using TensorFlow backend.
```

```
data = pd.read_csv('data.csv')
y = data['label'].values
X = data.drop(['label'], axis=1)
X.shape
y.shape
# create training and testing vars
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print( X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

```
(16000, 2) (16000,)
(4000, 2) (4000,)
```

```
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': []}

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

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.model1.stop_training = True
        weights1=model1.layers[1].get_weights()
        weights2=model1.layers[2].get_weights()
        weights3=model1.layers[3].get_weights()
        weights4=model1.layers[4].get_weights()
        weights5=model1.layers[5].get_weights()
        weights6=model1.layers[6].get_weights()
        for w1 in weights1:
          if np.nan in w1 or math.inf in w1:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
        for w2 in weights2:
          if np.nan in w2 or math.inf in w2:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
        for w3 in weights3:
          if np.nan in w3 or math.inf in w3:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
        for w4 in weights4:
          if np.nan in w4 or math.inf in w4:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
        for w5 in weights5:
          if np.nan in w5 or math.inf in w5:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
        for w6 in weights6:
          if np.nan in w6 or math.inf in w6:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model1.stop_training = True
history_own=LossHistory()

# Function to change the learning rate every 3 rd epoch
def changeLearningRate(epoch):
  initial_lrate = 0.1
  drop = 0.05
  epochs_drop = 3
  lrate = initial_lrate * math.pow(drop,
           math.floor((1+epoch)/epochs_drop))
  return lrate


# ref https://developpaper.com/tf-keras-implements-f1-score-precision-recall-and-other-metrics/
```

```
# function to calculate the F1 score
class Metrics(tf.keras.callbacks.Callback):
  def __init__(self, valid_data):
    super(Metrics, self).__init__()
    self.validation_data = valid_data
  def on_train_begin(self, logs={}):
    self.val_f1s = []
    self.val_recalls = []
    self.val_precisions = []
    self.val_auc=[]

  def on_epoch_end(self, epoch, logs={}):
    val_predict = (np.asarray(self.model.predict(self.validation_data[0]))).round()
    val_targ = self.validation_data[1]
    _val_f1 = f1_score(val_targ, val_predict)
    _val_recall = recall_score(val_targ, val_predict)
    _val_precision = precision_score(val_targ, val_predict)
    _val_fpr,_val_tpr,_v_tresh= roc_curve(val_targ, val_predict)
    _val_auc=auc(_val_fpr,_val_tpr)
    self.val_f1s.append(_val_f1)
    self.val_recalls.append(_val_recall)
    self.val_precisions.append(_val_precision)
    self.val_auc.append(_val_auc)
    print( " — val_f1: %f  _val_auc: %f " %(_val_f1,_val_auc))
    return

metrics = Metrics(valid_data=(X_test,y_test))
```

Model 1 : F1_score :0.657943 , Accuracy :0.5017

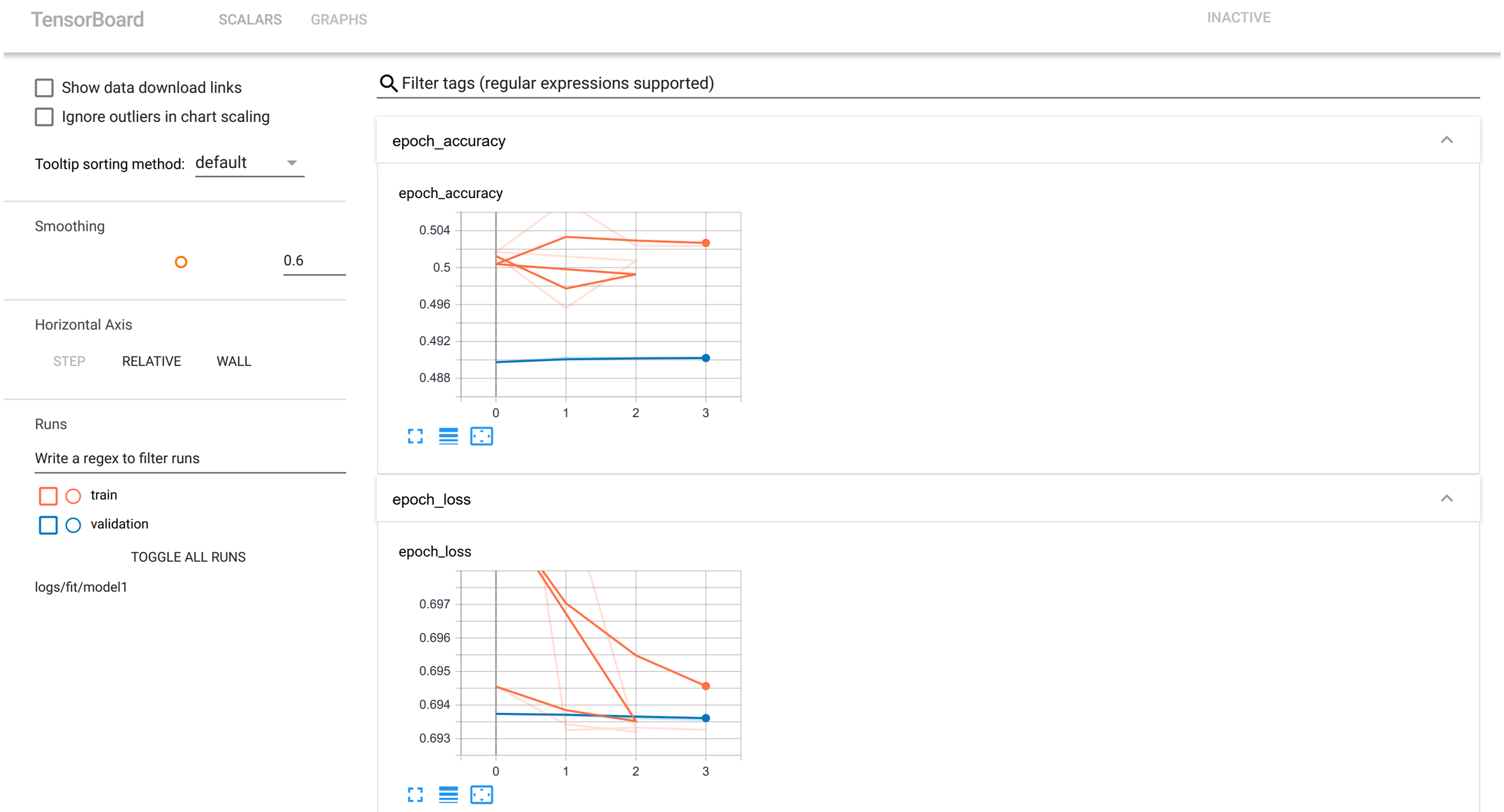
```
# For a single-input model with 2 classes (binary classification):
model1 = tf.keras.models.Sequential()
init=RandomUniform(minval=0, maxval=1, seed=None)
model1.add(Dense(20, activation='tanh', input_dim=2,kernel_initializer=init))
model1.add(Dense(16, activation='tanh',kernel_initializer=init))
model1.add(Dense(12, activation='tanh',kernel_initializer=init))
model1.add(Dense(8, activation='tanh',kernel_initializer=init))
model1.add(Dense(4, activation='tanh',kernel_initializer=init))
model1.add(Dense(2, activation='tanh',kernel_initializer=init))
model1.add(Dense(1, activation='sigmoid',kernel_initializer=init))

sgd=SGD(lr=0.0001,momentum=0.9,nesterov=True)
model1.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

filepath="drive/My Drive/Colab Notebooks/model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')
# earlystop call back monitoring val_accuray
earlystop = EarlyStopping(monitor='val_accuracy', patience=2, verbose=1)
#callback changes the learning rate
lrschedule = LearningRateScheduler(changeLearningRate,verbose=1)
#Callback reduces learning rate by 10%
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,patience=1)
```

```
#callback to terminate the training when value nan or inf encountered in weights or loss
terminate=TerminateNaN()
#tensor board callback
log_dir="logs/fit/model1"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0, write_graph=True,write_grads=Tru

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

# Train the model, iterating on the data in batches of 100 samples
model1.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_test), batch_size=100,callbacks=callback_lst)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/10
152/160 [==========================>..] - ETA: 0s - loss: 0.7082 - accuracy: 0.5011
Epoch 00001: val_accuracy improved from -inf to 0.48975, saving model to drive/My Drive/Colab Notebooks/model_save/weights-01-0.4897.hdf5
 — val_f1: 0.492415  _val_auc: 0.490039
160/160 [==============================] - 1s 9ms/step - loss: 0.7075 - accuracy: 0.5017 - val_loss: 0.6937 - val_accuracy: 0.4897 - lr: 0.1000

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/10
134/160 [=======================>.....] - ETA: 0s - loss: 0.6933 - accuracy: 0.5066
Epoch 00002: val_accuracy improved from 0.48975 to 0.49025, saving model to drive/My Drive/Colab Notebooks/model_save/weights-02-0.4902.hdf5
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5072 - val_loss: 0.6937 - val_accuracy: 0.4902 - lr: 0.1000

Epoch 00003: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 3/10
140/160 [=======================>....] - ETA: 0s - loss: 0.6934 - accuracy: 0.5017
Epoch 00003: val_accuracy did not improve from 0.49025
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5024 - val_loss: 0.6936 - val_accuracy: 0.4902 - lr: 5.0000e-04

Epoch 00004: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 4/10
142/160 [=======================>....] - ETA: 0s - loss: 0.6933 - accuracy: 0.5017
Epoch 00004: val_accuracy did not improve from 0.49025
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 0s 3ms/step - loss: 0.6933 - accuracy: 0.5023 - val_loss: 0.6936 - val_accuracy: 0.4902 - lr: 5.0000e-04
Epoch 00004: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f23ec950f98>
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model1
```

```
Reusing TensorBoard on port 6006 (pid 411), started 0:49:55 ago. (Use '!kill 411' to kill it.)
```



Model 2: F1_score :0.657943 , Accuracy :0.5020

```
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, terminated at epoch {}".format(epoch))
```

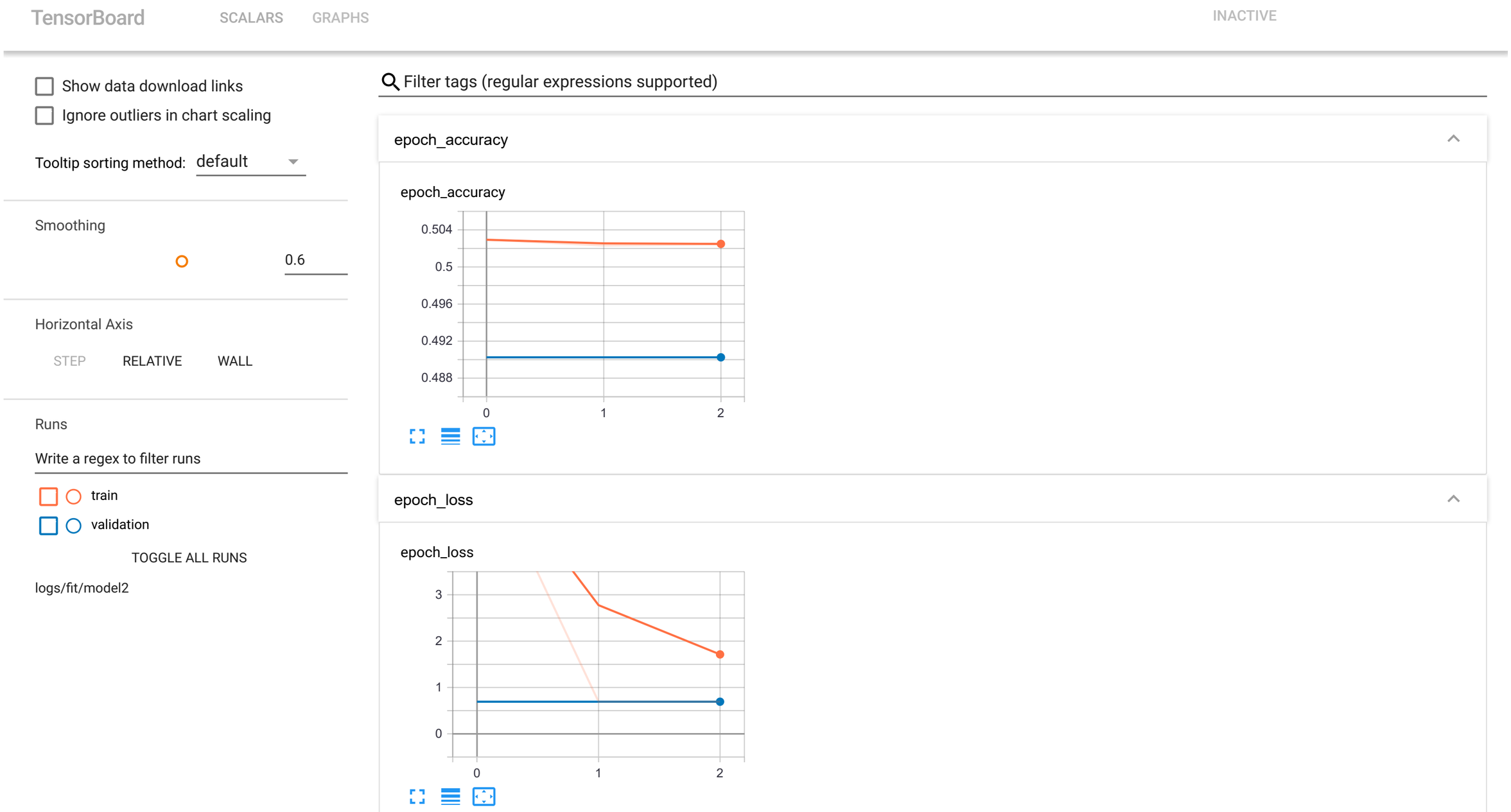
```
        print("Invalid loss and terminated at epoch {}".format(epoch))
        self.model2.stop_training = True
weights1=model2.layers[1].get_weights()
weights2=model2.layers[2].get_weights()
weights3=model2.layers[3].get_weights()
weights4=model2.layers[4].get_weights()
weights5=model2.layers[5].get_weights()
weights6=model2.layers[6].get_weights()
for w1 in weights1:
  if np.nan in w1 or math.inf in w1:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
for w2 in weights2:
  if np.nan in w2 or math.inf in w2:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
for w3 in weights3:
  if np.nan in w3 or math.inf in w3:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
for w4 in weights4:
  if np.nan in w4 or math.inf in w4:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
for w5 in weights5:
  if np.nan in w5 or math.inf in w5:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
for w6 in weights6:
  if np.nan in w6 or math.inf in w6:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model2.stop_training = True
history_own=LossHistory()



# For a single-input model with 2 classes (binary classification):
model2 = tf.keras.models.Sequential()
init=RandomUniform(minval=0, maxval=1, seed=None)
model2.add(Dense(20, activation='relu', input_dim=2,kernel_initializer=init))
model2.add(Dense(16, activation='relu',kernel_initializer=init))
model2.add(Dense(12, activation='relu',kernel_initializer=init))
model2.add(Dense(8, activation='relu',kernel_initializer=init))
model2.add(Dense(4, activation='relu',kernel_initializer=init))
model2.add(Dense(2, activation='relu',kernel_initializer=init))
model2.add(Dense(1, activation='sigmoid',kernel_initializer=init))

sgd=SGD(lr=0.0001,momentum=0.9,nesterov=True)
model2.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

filepath="drive/My Drive/Colab Notebooks/model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')
# earlystop call back monitoring val_accuray
earlystop = EarlyStopping(monitor='val_accuracy', patience=2, verbose=1)
```

```
# earlystop call back monitoring val_accuray
lrschedule = LearningRateScheduler(changeLearningRate,verbose=1)
# earlystop call back monitoring val_accuray
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,patience=1)
# earlystop call back monitoring val_accuray
terminate=TerminateNaN()
#tensor board callback
log_dir="logs/fit/model2"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0, write_graph=True,write_grads=Tru
#callback list
callback_lst=[history_own,checkpoint,earlystop,reduce_lr,lrschedule,terminate,metrics,tensorboard_callback]

# Train the model, iterating on the data in batches of 100 samples
model2.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_test), batch_size=100,callbacks=callback_lst)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/10
140/160 [=========================>....] - ETA: 0s - loss: 7.0440 - accuracy: 0.5034
Epoch 00001: val_accuracy improved from -inf to 0.49025, saving model to drive/My Drive/Colab Notebooks/model_save/weights-01-0.4902.hdf5
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 2s 10ms/step - loss: 6.2502 - accuracy: 0.5029 - val_loss: 0.6932 - val_accuracy: 0.4902 - lr: 0.1000

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/10
136/160 [=========================>.....] - ETA: 0s - loss: 0.6932 - accuracy: 0.5053
Epoch 00002: val_accuracy did not improve from 0.49025
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.5023 - val_loss: 0.6932 - val_accuracy: 0.4902 - lr: 0.0100

Epoch 00003: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 3/10
132/160 [=========================>......] - ETA: 0s - loss: 0.6931 - accuracy: 0.5020
Epoch 00003: val_accuracy did not improve from 0.49025
 — val_f1: 0.657943  _val_auc: 0.500000
160/160 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.5024 - val_loss: 0.6932 - val_accuracy: 0.4902 - lr: 5.0000e-04
Epoch 00003: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f23e798c6a0>
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model2
```

```
Reusing TensorBoard on port 6007 (pid 883), started 0:42:51 ago. (Use '!kill 883' to kill it.)
```



Model 3 : F1_score :0.673799 ,Accuracy :0.6735

```
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, terminated at epoch {}".format(epoch))
```

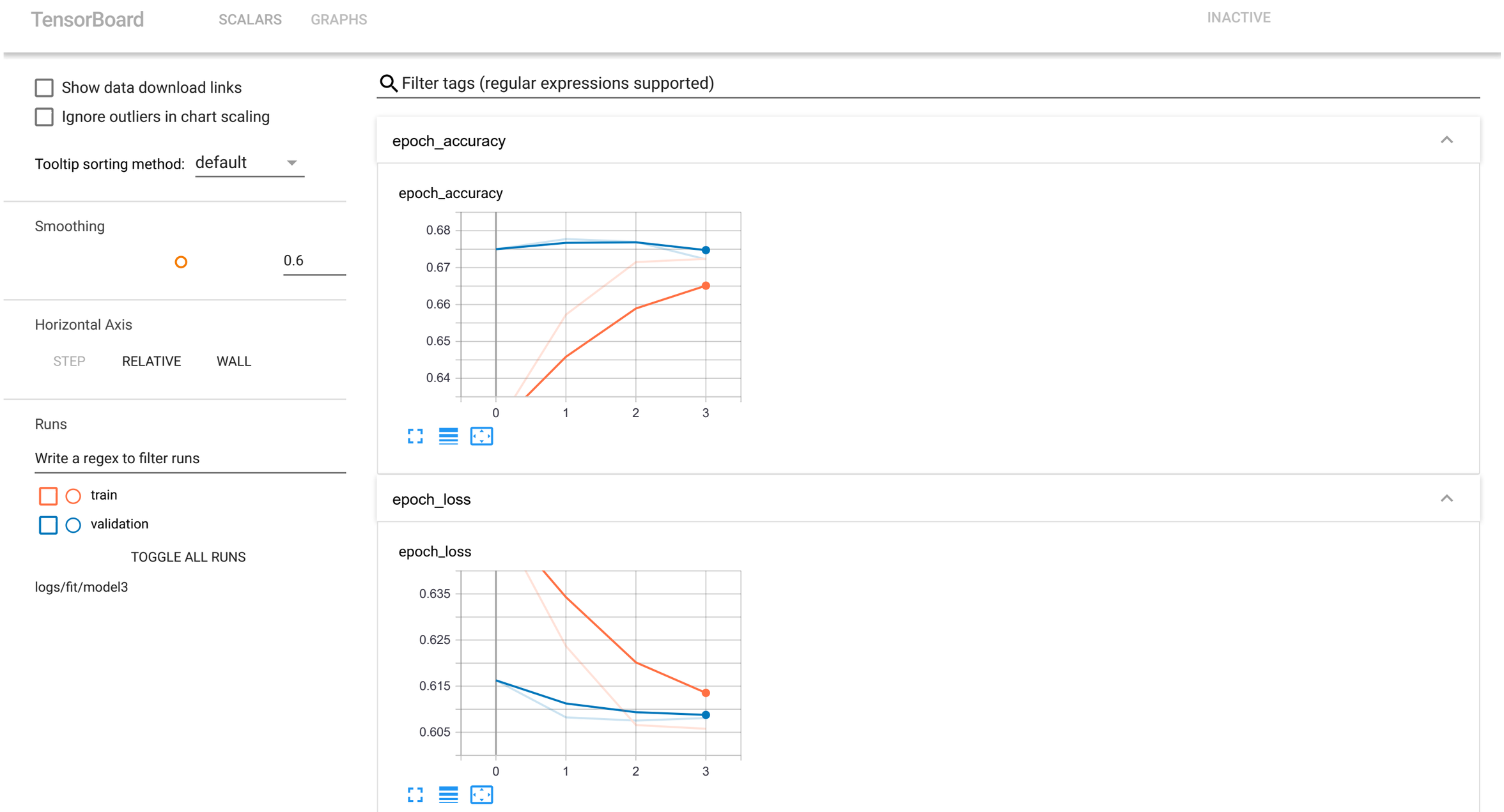
```
        print("Invalid loss and terminated at epoch {}".format(epoch))
        self.model3.stop_training = True
weights1=model3.layers[1].get_weights()
weights2=model3.layers[2].get_weights()
weights3=model3.layers[3].get_weights()
weights4=model3.layers[4].get_weights()
weights5=model3.layers[5].get_weights()
weights6=model3.layers[6].get_weights()
for w1 in weights1:
  if np.nan in w1 or math.inf in w1:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model3.stop_training = True
for w2 in weights2:
  if np.nan in w2 or math.inf in w2:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model1.stop_training = True
for w3 in weights3:
  if np.nan in w3 or math.inf in w3:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model3.stop_training = True
for w4 in weights4:
  if np.nan in w4 or math.inf in w4:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model3.stop_training = True
for w5 in weights5:
  if np.nan in w5 or math.inf in w5:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model3.stop_training = True
for w6 in weights6:
  if np.nan in w6 or math.inf in w6:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model3.stop_training = True
history_own=LossHistory()


# For a single-input model with 2 classes (binary classification):
model3 = tf.keras.models.Sequential()

init=tf.keras.initializers.he_uniform(seed=None)

model3.add(Dense(20, activation='relu', input_dim=2,kernel_initializer=init))
model3.add(Dense(16, activation='relu',kernel_initializer=init))
model3.add(Dense(12, activation='relu',kernel_initializer=init))
model3.add(Dense(8, activation='relu',kernel_initializer=init))
model3.add(Dense(4, activation='relu',kernel_initializer=init))
model3.add(Dense(2, activation='relu',kernel_initializer=init))
model3.add(Dense(1, activation='sigmoid',kernel_initializer=init))

sgd=SGD(lr=0.0001,momentum=0.9,nesterov=True)
model3.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

filepath="drive/My Drive/Colab Notebooks/model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')
# earlystop call back monitoring val_accuray
```

```
earlystop = EarlyStopping(monitor='val_accuracy', patience=2, verbose=1)
#callback changes the learning rate
lrschedule = LearningRateScheduler(changeLearningRate,verbose=1)
#callback changes the learning rate
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,patience=1)
#callback changes the learning rate
terminate=TerminateNaN()
#tensor board callback
log_dir="logs/fit/model3"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0, write_graph=True,write_grads=Tru
#callback list
callback_lst=[history_own,checkpoint,earlystop,reduce_lr,lrschedule,terminate,metrics,tensorboard_callback]

# Train the model, iterating on the data in batches of 100 samples
model3.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_test), batch_size=100,callbacks=callback_lst)
```

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

Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/10
149/160 [==========================>...] - ETA: 0s - loss: 0.6540 - accuracy: 0.6242
Epoch 00001: val_accuracy improved from -inf to 0.67500, saving model to drive/My Drive/Colab Notebooks/model_save/weights-01-0.6750.hdf5
 — val_f1: 0.656811  _val_auc: 0.674223
160/160 [==============================] - 1s 4ms/step - loss: 0.6520 - accuracy: 0.6268 - val_loss: 0.6163 - val_accuracy: 0.6750 - lr: 0.1000

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/10
132/160 [=======================>......] - ETA: 0s - loss: 0.6272 - accuracy: 0.6534
Epoch 00002: val_accuracy improved from 0.67500 to 0.67775, saving model to drive/My Drive/Colab Notebooks/model_save/weights-02-0.6777.hdf5
 — val_f1: 0.661591  _val_auc: 0.677076
160/160 [==============================] - 0s 3ms/step - loss: 0.6237 - accuracy: 0.6572 - val_loss: 0.6082 - val_accuracy: 0.6777 - lr: 0.1000

Epoch 00003: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 3/10
156/160 [===========================>.] - ETA: 0s - loss: 0.6062 - accuracy: 0.6722
Epoch 00003: val_accuracy did not improve from 0.67775
 — val_f1: 0.668888  _val_auc: 0.676780
160/160 [==============================] - 0s 3ms/step - loss: 0.6066 - accuracy: 0.6715 - val_loss: 0.6075 - val_accuracy: 0.6770 - lr: 5.0000e-04

Epoch 00004: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 4/10
133/160 [=======================>......] - ETA: 0s - loss: 0.6052 - accuracy: 0.6735
Epoch 00004: val_accuracy did not improve from 0.67775
 — val_f1: 0.673799  _val_auc: 0.672598
160/160 [==============================] - 0s 3ms/step - loss: 0.6058 - accuracy: 0.6724 - val_loss: 0.6081 - val_accuracy: 0.6722 - lr: 5.0000e-04
Epoch 00004: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f23e908b978>
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model3
```

```
Reusing TensorBoard on port 6008 (pid 1305), started 0:38:34 ago. (Use '!kill 1305' to kill it.)
```



Model 4: F1_score :0.666667,Accuracy :0.6706

```
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, terminated at epoch {}".format(epoch))
```

```
        print("Invalid loss and terminated at epoch {}".format(epoch))
        self.model4.stop_training = True
weights1=model4.layers[1].get_weights()
weights2=model4.layers[2].get_weights()
weights3=model4.layers[3].get_weights()
weights4=model4.layers[4].get_weights()
weights5=model4.layers[5].get_weights()
weights6=model4.layers[6].get_weights()
for w1 in weights1:
  if np.nan in w1 or math.inf in w1:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model4.stop_training = True
for w2 in weights2:
  if np.nan in w2 or math.inf in w2:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model4.stop_training = True
for w3 in weights3:
  if np.nan in w3 or math.inf in w3:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model4.stop_training = True
for w4 in weights4:
  if np.nan in w4 or math.inf in w4:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model4.stop_training = True
for w5 in weights5:
  if np.nan in w5 or math.inf in w5:
    print("Invalid weights and terminated at epoch {}".format(epoch))
    self.model4.stop_training = True
        for w6 in weights6:
          if np.nan in w6 or math.inf in w6:
            print("Invalid weights and terminated at epoch {}".format(epoch))
            self.model4.stop_training = True
history_own=LossHistory()


# For a single-input model with 2 classes (binary classification):
model4 = tf.keras.models.Sequential()
init=tf.keras.initializers.he_uniform(seed=None)

model4.add(Dense(20, activation='relu', input_dim=2,kernel_initializer=init))
model4.add(Dense(16, activation='relu',kernel_initializer=init))
model4.add(Dense(12, activation='relu',kernel_initializer=init))
model4.add(Dense(8, activation='relu',kernel_initializer=init))
model4.add(Dense(4, activation='relu',kernel_initializer=init))
model4.add(Dense(2, activation='relu',kernel_initializer=init))
model4.add(Dense(1, activation='sigmoid',kernel_initializer=init))
sgd=SGD(lr=0.0001,momentum=0.9,nesterov=True)
model4.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

filepath="drive/My Drive/Colab Notebooks/model_save/weights-{epoch:02d}-{val_loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_loss',  verbose=1, save_best_only=True, mode='auto')
# earlystop call back monitoring val_accuray
earlystop = EarlyStopping(monitor='val_loss', patience=3, verbose=1)
# earlystop call back monitoring val_accuray
```

```
lrschedule = LearningRateScheduler(changeLearningRate,verbose=1)
#Callback reduces learning rate by 10%
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.1,patience=1)
#Callback reduces learning rate by 10%
terminate=TerminateNaN()
#tensor board callback
log_dir="logs/fit/model4"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0, write_graph=True,write_grads=True
#callback list
callback_lst=[history_own,checkpoint,earlystop,lrschedule,reduce_lr,terminate,metrics,tensorboard_callback]

# Train the model, iterating on the data in batches of 100 samples
model4.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_test), batch_size=100,callbacks=callback_lst)
```

```
Epoch 00001: LearningRateScheduler reducing learning rate to 0.1.
Epoch 1/10
130/160 [=======================>......] - ETA: 0s - loss: 0.6636 - accuracy: 0.6264
Epoch 00001: val_loss improved from inf to 0.62657, saving model to drive/My Drive/Colab Notebooks/model_save/weights-01-0.6266.hdf5
 — val_f1: 0.673585  _val_auc: 0.670234
160/160 [==============================] - 1s 4ms/step - loss: 0.6577 - accuracy: 0.6313 - val_loss: 0.6266 - val_accuracy: 0.6697 - lr: 0.1000

Epoch 00002: LearningRateScheduler reducing learning rate to 0.1.
Epoch 2/10
156/160 [===========================>.] - ETA: 0s - loss: 0.6200 - accuracy: 0.6601
Epoch 00002: val_loss improved from 0.62657 to 0.61828, saving model to drive/My Drive/Colab Notebooks/model_save/weights-02-0.6183.hdf5
 — val_f1: 0.694215  _val_auc: 0.668990
160/160 [==============================] - 0s 3ms/step - loss: 0.6200 - accuracy: 0.6603 - val_loss: 0.6183 - val_accuracy: 0.6670 - lr: 0.1000

Epoch 00003: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 3/10
144/160 [==========================>...] - ETA: 0s - loss: 0.6076 - accuracy: 0.6695
Epoch 00003: val_loss improved from 0.61828 to 0.60759, saving model to drive/My Drive/Colab Notebooks/model_save/weights-03-0.6076.hdf5
 — val_f1: 0.673382  _val_auc: 0.675866
160/160 [==============================] - 0s 3ms/step - loss: 0.6069 - accuracy: 0.6699 - val_loss: 0.6076 - val_accuracy: 0.6758 - lr: 0.0050

Epoch 00004: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 4/10
136/160 [=======================>.....] - ETA: 0s - loss: 0.6060 - accuracy: 0.6712
Epoch 00004: val_loss improved from 0.60759 to 0.60756, saving model to drive/My Drive/Colab Notebooks/model_save/weights-04-0.6076.hdf5
 — val_f1: 0.675019  _val_auc: 0.675963
160/160 [==============================] - 0s 3ms/step - loss: 0.6056 - accuracy: 0.6716 - val_loss: 0.6076 - val_accuracy: 0.6758 - lr: 0.0050

Epoch 00005: LearningRateScheduler reducing learning rate to 0.005000000000000001.
Epoch 5/10
140/160 [========================>....] - ETA: 0s - loss: 0.6033 - accuracy: 0.6756
Epoch 00005: val_loss improved from 0.60756 to 0.60631, saving model to drive/My Drive/Colab Notebooks/model_save/weights-05-0.6063.hdf5
 — val_f1: 0.664091  _val_auc: 0.673681
160/160 [==============================] - 0s 3ms/step - loss: 0.6052 - accuracy: 0.6727 - val_loss: 0.6063 - val_accuracy: 0.6740 - lr: 0.0050

Epoch 00006: LearningRateScheduler reducing learning rate to 0.00025000000000000006.
Epoch 6/10
136/160 [=======================>.....] - ETA: 0s - loss: 0.6046 - accuracy: 0.6715
Epoch 00006: val_loss did not improve from 0.60631
 — val_f1: 0.664955  _val_auc: 0.673494
160/160 [==============================] - 0s 3ms/step - loss: 0.6047 - accuracy: 0.6707 - val_loss: 0.6063 - val_accuracy: 0.6737 - lr: 2.5000e-04

Epoch 00007: LearningRateScheduler reducing learning rate to 0.00025000000000000006.
Epoch 7/10
136/160 [=======================>.....] - ETA: 0s - loss: 0.6046 - accuracy: 0.6692
Epoch 00007: val_loss did not improve from 0.60631
 — val_f1: 0.667349  _val_auc: 0.674573
160/160 [==============================] - 0s 3ms/step - loss: 0.6046 - accuracy: 0.6709 - val_loss: 0.6064 - val_accuracy: 0.6747 - lr: 2.5000e-04

Epoch 00008: LearningRateScheduler reducing learning rate to 0.00025000000000000006.
Epoch 8/10
129/160 [=======================>......] - ETA: 0s - loss: 0.6053 - accuracy: 0.6706
Epoch 00008: val_loss did not improve from 0.60631
 — val_f1: 0.666667  _val_auc: 0.673592
160/160 [==============================] - 0s 3ms/step - loss: 0.6046 - accuracy: 0.6710 - val_loss: 0.6064 - val_accuracy: 0.6737 - lr: 2.5000e-04
Epoch 00008: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f23e99c9550>
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model4
```

```
Reusing TensorBoard on port 6009 (pid 1408), started 0:38:30 ago. (Use '!kill 1408' to kill it.)
```