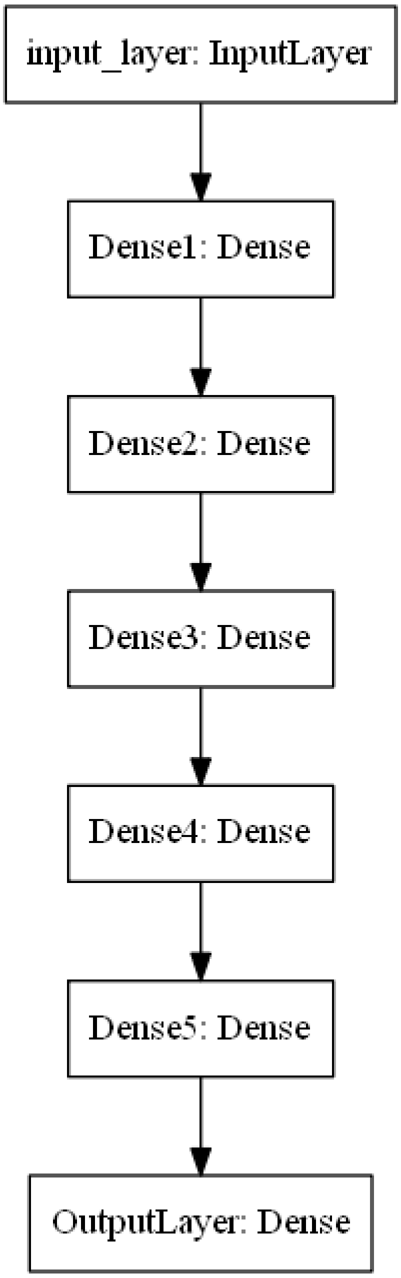
1. Download the data from here

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 a

4. Save your model at every epoch if your validation accuracy is improved from previous e

5. you have to decay learning based on below conditions
   Cond1. If your validation accuracy at that epoch is less than previous epoch accu
   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

7. You have to stop the training if your validation accuracy is not increased in last 2 e

8. Use tensorboard for every model and analyse your gradients. (you need to upload the sc

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.

importing libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.layers import Dense,Input,Activation
from tensorflow.keras.models import Model
import random as rn
from tensorflow import keras
import datetime, os

from keras.callbacks import Callback
from sklearn.metrics import roc_auc_score, f1_score
```

```
Using TensorFlow backend.
```

```
2802988797180/15dCNcmKskcFVjs7R0ElQkR61Ex53uJpM?e=download&authuser=0&nonce=3995mllt1ld
```

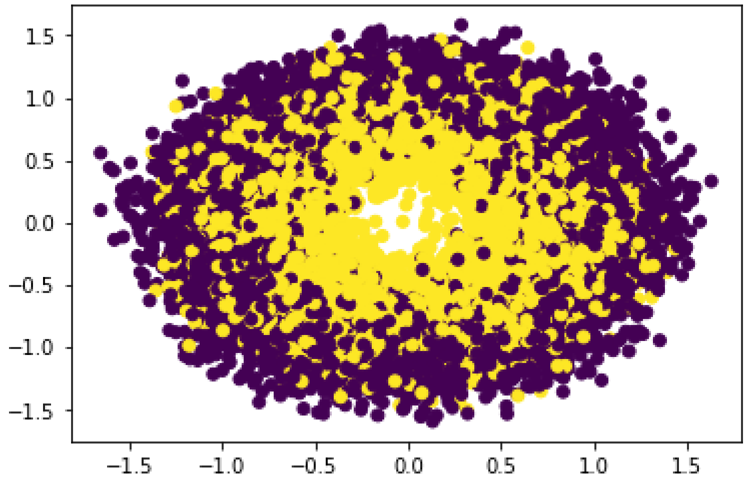
```
--2020-06-06 03:28:14--  https://doc-08-3k-docs.googleusercontent.com/docs/securesc/8
Resolving doc-08-3k-docs.googleusercontent.com (doc-08-3k-docs.googleusercontent.com)
Connecting to doc-08-3k-docs.googleusercontent.com (doc-08-3k-docs.googleusercontent.
HTTP request sent, awaiting response... 200 OK
Length: 886913 (866K) [text/csv]
Saving to: ‘data.csv’

data.csv            100%[===================>] 866.13K  --.-KB/s    in 0.006s

2020-06-06 03:28:14 (132 MB/s) - ‘data.csv’ saved [886913/886913]
```

```
data=pd.read_csv("data.csv")
data.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

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

```
import matplotlib.pyplot as plt
plt.scatter(data['f1'], data['f2'], c=y)
plt.show()

```



```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

```
class Metrics(tf.keras.callbacks.Callback):
   # callback that print the micro F1 score and AUC score after each epoch.
  def __init__(self):
    self.validation_data=(x_test,y_test)
  def on_train_begin(self, logs={}):
    self.val_f1s = []
    self.val_aucs=[]
  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.round())
     roc_val=roc_auc_score(val_targ, val_predict)
     self.val_f1s.append(val_f1)
     self.val_aucs.append(roc_val)
     print("-f1 score :",val_f1,"-ROCValue :", roc_val)

```

```
class TerminateNaN(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
        loss = logs.get('loss')
        if loss is not None:
            if np.isnan(loss) or np.isinf(loss):
                print("Invalid loss and terminated at epoch {}".format(epoch))
                self.model.stop_training = True

```

```
  File "<ipython-input-31-d0860c6b82cc>", line 3
    loss = logs.get('loss')
```

```
def lr_scheduler(epoch, lr):
  #reduce learning rate after every 3 epoch
    decay_rate = .95
    decay_step = 3
    if (epoch+1) % decay_step == 0 :
        return lr * decay_rate
    return lr
```

```
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import LearningRateScheduler

```

```
# Load the TensorBoard notebook extension
%load_ext tensorboard
```

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

```

def create_model_1():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="tanh",input_shape=(2,),kernel_initializer=keras
    tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.R
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.R
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.
    tf. keras.layers.Dense(16, activation="tanh",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers
  ])
```

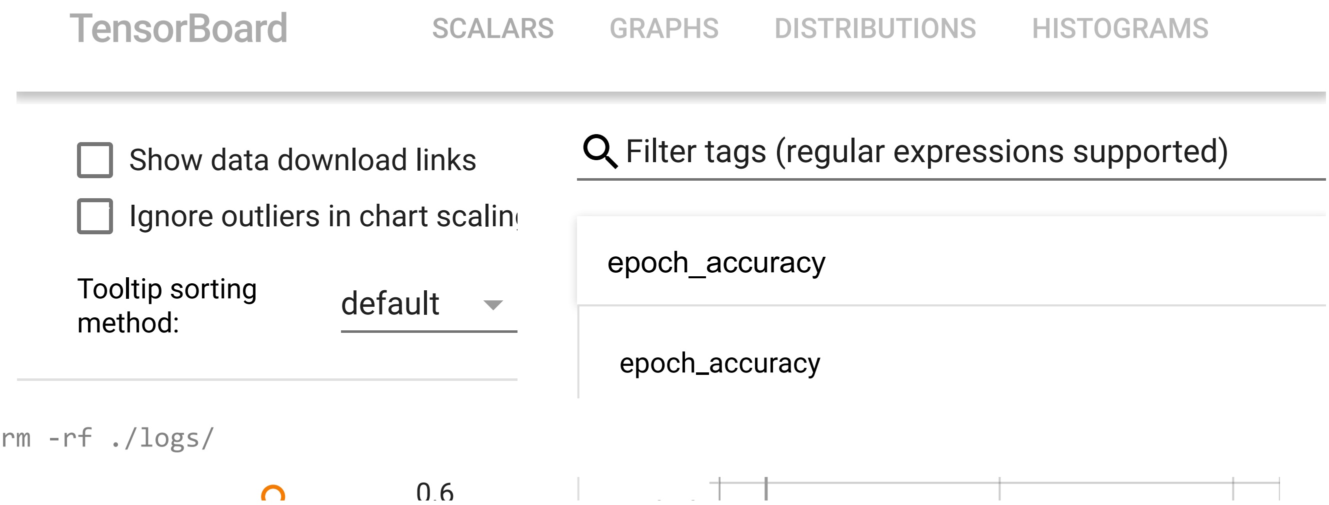
```
filepath="model_save/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=1, min_lr=0.
lrschedule = LearningRateScheduler(lr_scheduler, verbose=0)
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, sav
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.35, patience=2, verbose=1
terminate= TerminateNaN()
metrics=Metrics()
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1)
```

```
model_1=create_model_1()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, nam
model_1.compile(optimizer,
              loss='BinaryCrossentropy',
              metrics=['accuracy'])
model_1.fit(x=x_train,
          y=y_train,
          epochs=15
          epochs=15,
          validation_data=(x_test, y_test),callbacks=[metrics,checkpoint,terminate,lrsc
          )
```

```
Epoch 1/15
458/469 [============================>.] - ETA: 0s - loss: 7.6048 - accuracy: 0.5013-

Epoch 00001: val_accuracy improved from -inf to 0.49360, saving model to model_save/w
469/469 [==============================] - 1s 3ms/step - loss: 7.5921 - accuracy: 0.5
Epoch 2/15
452/469 [===========================>..] - ETA: 0s - loss: 7.5824 - accuracy: 0.5028-

Epoch 00002: val_accuracy did not improve from 0.49360
469/469 [==============================] - 1s 2ms/step - loss: 7.5921 - accuracy: 0.5
Epoch 3/15
458/469 [============================>.] - ETA: 0s - loss: 7.5892 - accuracy: 0.5023-

Epoch 00003: val_accuracy did not improve from 0.49360
469/469 [==============================] - 1s 2ms/step - loss: 7.5921 - accuracy: 0.5
Epoch 00003: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f4b808f8470>
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 331), started 0:18:56 ago. (Use '!kill 331' to
```



□ Show data download links

□ Ignore outliers in chart scalin

Tooltip sorting method: default

Filter tags (regular expressions supported)

epoch_accuracy

epoch_accuracy

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

```

def create_model_2():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.R
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.R
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers
  ])
```

Write a regex to filter runs

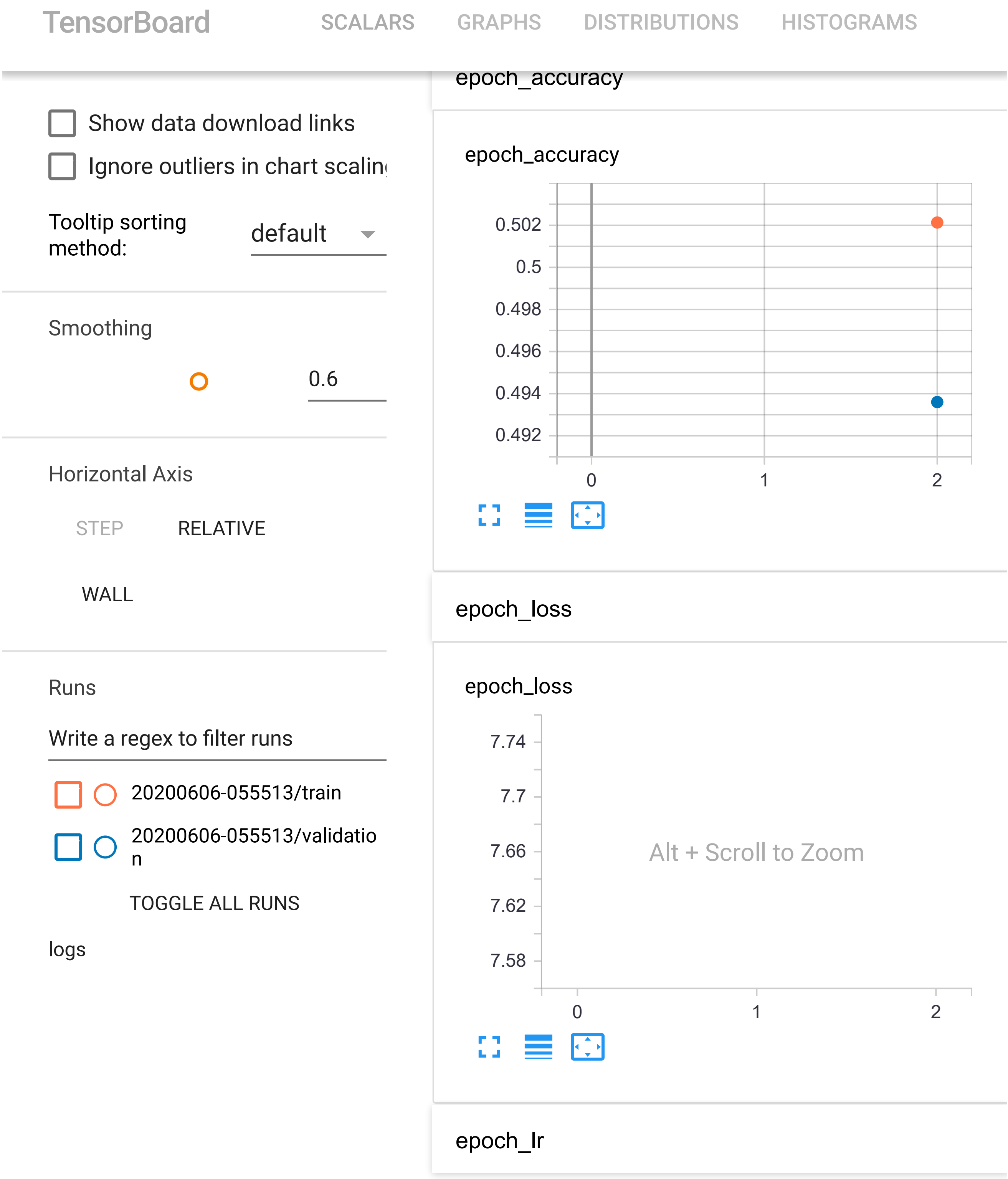
epoch_loss

```
model_2=create_model_2()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, nam
model_2.compile(optimizer,
            loss='BinaryCrossentropy',
            metrics=['accuracy'])
model_2.fit(x=x_train,
        y=y_train,
        epochs=15,
        validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,lr
        )
```

```
Epoch 1/15
449/469 [===========================>..] - ETA: 0s - loss: 7.5896 - accuracy: 0.5023
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 331), started 0:20:01 ago. (Use '!kill 331' to
```



TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS

□ Show data download links

□ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis STEP RELATIVE WALL

Runs

Write a regex to filter runs

□ ○ 20200606-055513/train

□ ○ 20200606-055513/validation

TOGGLE ALL RUNS

logs

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

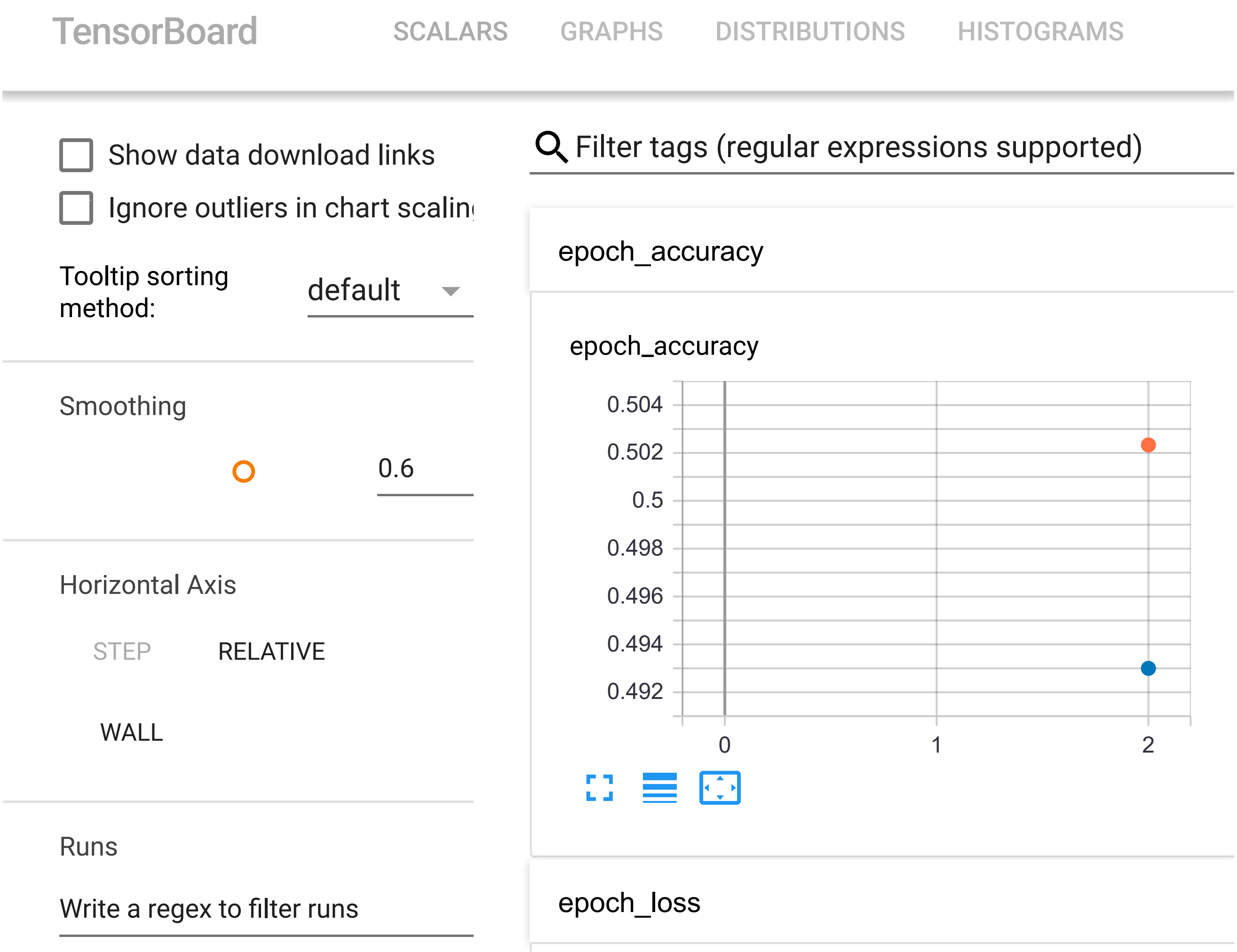
```
def create_model_3():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(2,activation="relu",input_shape=(2,),kernel_initializer=keras
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.h
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.h
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf. keras.layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.
    tf.keras.layers.Dense(1, activation='softmax',kernel_initializer=keras.initializers
  ])
```

```
model_3=create_model_3()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, nam
model_3.compile(optimizer,
              loss='BinaryCrossentropy',
              metrics=['accuracy'])
model_3.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,lr
          )
```

```
Epoch 1/5
450/469 [===========================>..] - ETA: 0s - loss: 7.6490 - accuracy: 0.4984
Epoch 00001: val_accuracy did not improve from 0.62120
-f1 score : 0.673036093418259 -ROCValue : 0.5
469/469 [==============================] - 1s 3ms/step - loss: 7.6612 - accuracy: 0.4
Epoch 2/5
458/469 [============================>.] - ETA: 0s - loss: 7.6683 - accuracy: 0.4971
Epoch 00002: val_accuracy did not improve from 0.62120
-f1 score : 0.673036093418259 -ROCValue : 0.5
469/469 [==============================] - 1s 2ms/step - loss: 7.6612 - accuracy: 0.4
Epoch 3/5
437/469 [==========================>...] - ETA: 0s - loss: 7.6595 - accuracy: 0.4977
Epoch 00003: val_accuracy did not improve from 0.62120
-f1 score : 0.673036093418259 -ROCValue : 0.5
469/469 [==============================] - 1s 2ms/step - loss: 7.6612 - accuracy: 0.4
Epoch 00003: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f4b808da9e8>
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 215), started 0:02:41 ago. (Use '!kill 215' to
```



TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS

□ Show data download links

□ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis STEP RELATIVE WALL

Runs

Write a regex to filter runs

Filter tags (regular expressions supported)

epoch_accuracy

epoch_loss

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

```
def create_model_4():
  return tf.keras.models.Sequential([
    tf.keras.layers.Dense(4,activation="relu",input_shape=(2,),kernel_initializer=keras
    tf.keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.h
    tf.keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.he_
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.h
    tf. keras.layers.Dense(8, activation="relu",kernel_initializer=keras.initializers.h
    tf.keras.layers.Dense(1, activation='sigmoid',kernel_initializer=keras.initializers
  ])
```

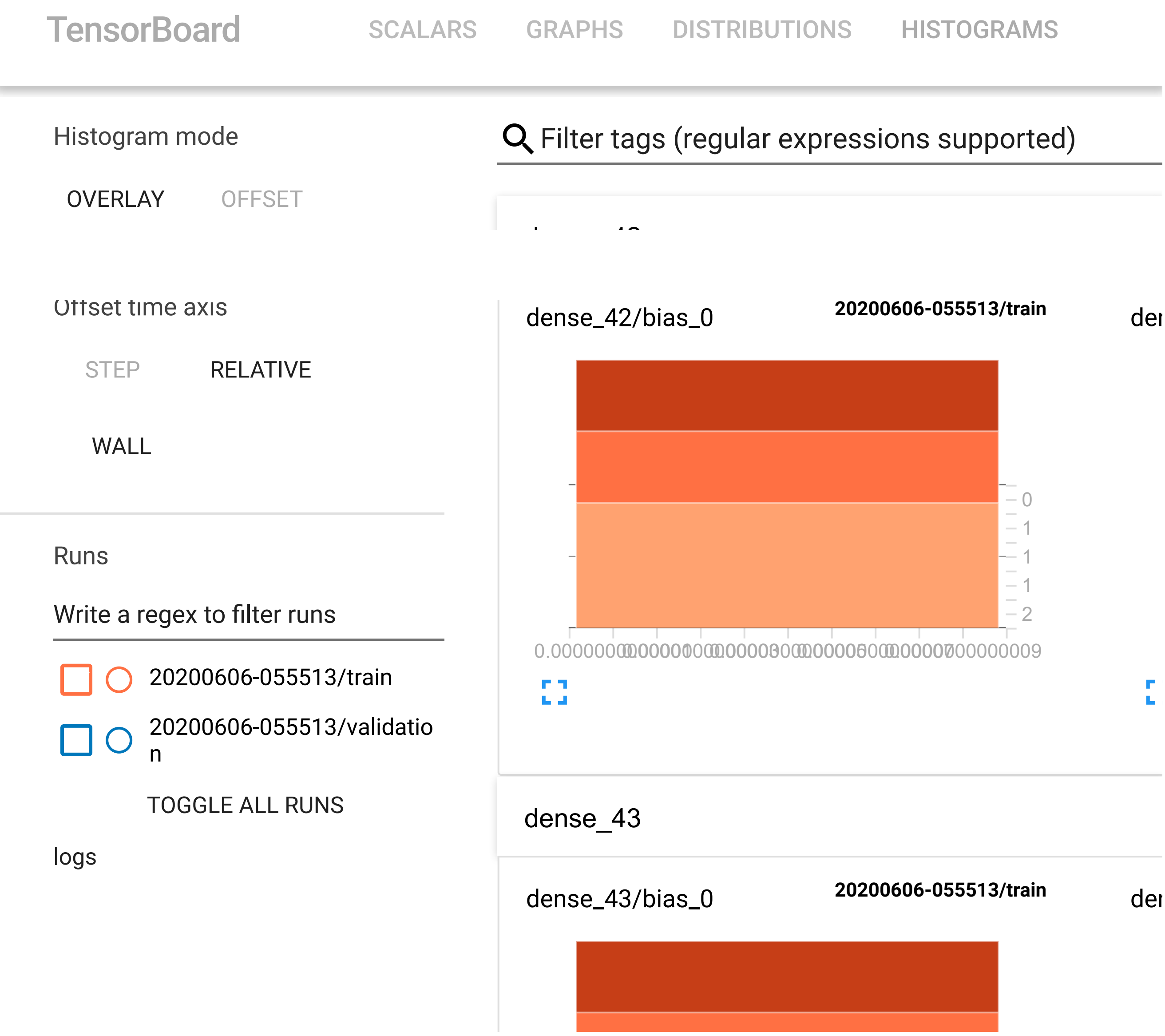
```
model_4=create_model_4()
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, nesterov=False, nam
model_4.compile(optimizer,
             loss='BinaryCrossentropy',
             metrics=['accuracy'])
model_4.fit(x=x_train,
         y=y_train,
         epochs=5,
         validation_data=(x_test, y_test),callbacks=[checkpoint,earlystop,terminate,lr
         )
```

```
Epoch 1/5
466/469 [============================>.] - ETA: 0s - loss: 0.7219 - accuracy: 0.4568
Epoch 00001: val_accuracy improved from 0.49360 to 0.49660, saving model to model_sav
-f1 score : 0.2466327446872194 -ROCValue : 0.4924336403276312
469/469 [==============================] - 1s 2ms/step - loss: 0.7217 - accuracy: 0.4
Epoch 2/5
465/469 [============================>.] - ETA: 0s - loss: 0.6917 - accuracy: 0.5119
Epoch 00002: val_accuracy improved from 0.49660 to 0.52500, saving model to model_sav
-f1 score : 0.43465841466317545 -ROCValue : 0.5230402549153653
469/469 [==============================] - 1s 2ms/step - loss: 0.6917 - accuracy: 0.5
Epoch 3/5
440/469 [===========================>..] - ETA: 0s - loss: 0.6874 - accuracy: 0.5347
Epoch 00003: val_accuracy improved from 0.52500 to 0.54320, saving model to model_sav
-f1 score : 0.5487949427103912 -ROCValue : 0.5434477584807494
469/469 [==============================] - 1s 2ms/step - loss: 0.6872 - accuracy: 0.5
Epoch 00003: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f4b779674e0>
```

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 331), started 0:21:08 ago. (Use '!kill 331' to
```

1



TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS

Histogram mode

OVERLAY OFFSET

Offset time axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

□ ○ 20200606-055513/train

□ ○ 20200606-055513/validation

TOGGLE ALL RUNS

logs

Filter tags (regular expressions supported)

dense_42/bias_0 **20200606-055513/train**

dense_43

dense_43/bias_0 **20200606-055513/train**