```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
```

```
import keras
keras.__version__
```

```
    '2.4.3'
```

```
from keras.datasets import imdb

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
    17465344/17464789 [==============================] - 0s 0us/step
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/datasets/imdb.py:159: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
      x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/datasets/imdb.py:160: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
      x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
```

```
max([max(sequence) for sequence in train_data])
```

```
    9999
```

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    # Create an all-zero matrix of shape (len(sequences), dimension)
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # set specific indices of results[i] to 1s
    return results

# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
```

```
x_train[0]
```

```
    array([0., 1., 1., ..., 0., 0., 0.])
```

```
# Our vectorized labels
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

run the original model implementing checkpoint

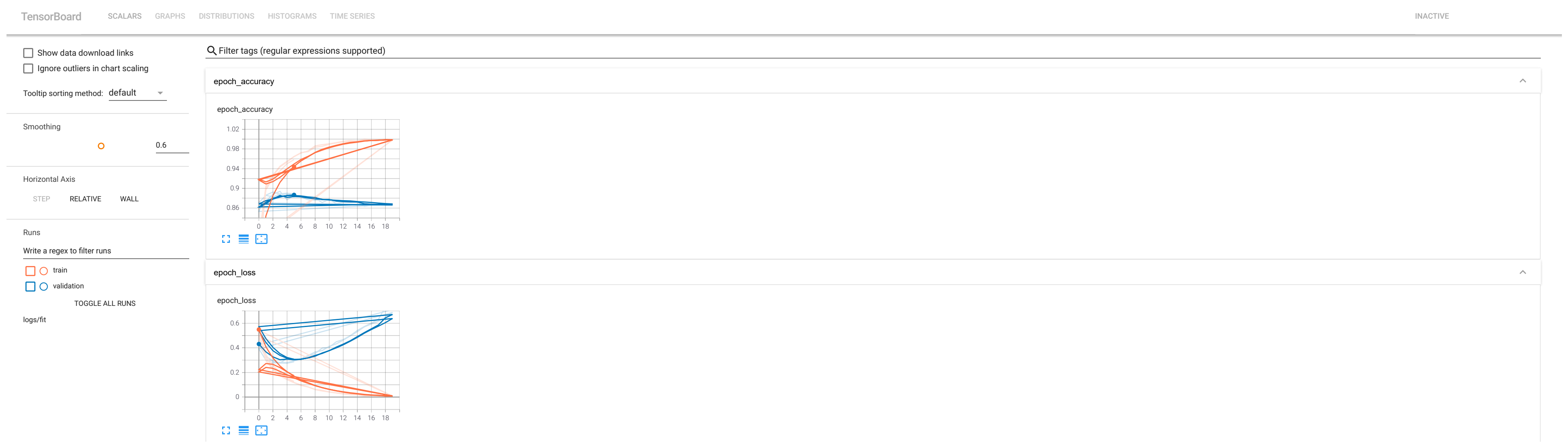
- List item
- List item

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
# checkpoint
filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
log_dir = "logs/fit/"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val),
                    callbacks=callbacks_list)
```

```
    Epoch 1/20
    30/30 [==============================] - 2s 51ms/step - loss: 0.6089 - accuracy: 0.6833 - val_loss: 0.4210 - val_accuracy: 0.8733

    Epoch 00001: val_accuracy improved from -inf to 0.87330, saving model to weights-improvement-01-0.87.hdf5
    Epoch 2/20
    30/30 [==============================] - 1s 35ms/step - loss: 0.3562 - accuracy: 0.8969 - val_loss: 0.3340 - val_accuracy: 0.8815

    Epoch 00002: val_accuracy improved from 0.87330 to 0.88150, saving model to weights-improvement-02-0.88.hdf5
    Epoch 3/20
    30/30 [==============================] - 1s 36ms/step - loss: 0.2488 - accuracy: 0.9330 - val_loss: 0.2923 - val_accuracy: 0.8870

    Epoch 00003: val_accuracy improved from 0.88150 to 0.88700, saving model to weights-improvement-03-0.89.hdf5
    Epoch 4/20
    30/30 [==============================] - 1s 35ms/step - loss: 0.1933 - accuracy: 0.9413 - val_loss: 0.2751 - val_accuracy: 0.8916

    Epoch 00004: val_accuracy improved from 0.88700 to 0.89160, saving model to weights-improvement-04-0.89.hdf5
    Epoch 5/20
    30/30 [==============================] - 1s 35ms/step - loss: 0.1524 - accuracy: 0.9540 - val_loss: 0.2816 - val_accuracy: 0.8881

    Epoch 00005: val_accuracy did not improve from 0.89160
    Epoch 6/20
```

```
30/30 [==============================] - 1s 35ms/step - loss: 0.1273 - accuracy: 0.9630 - val_loss: 0.3261 - val_accuracy: 0.8729

Epoch 00006: val_accuracy did not improve from 0.89160
Epoch 7/20
30/30 [==============================] - 1s 35ms/step - loss: 0.1052 - accuracy: 0.9698 - val_loss: 0.3227 - val_accuracy: 0.8781

Epoch 00007: val_accuracy did not improve from 0.89160
Epoch 8/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0857 - accuracy: 0.9788 - val_loss: 0.3196 - val_accuracy: 0.8812

Epoch 00008: val_accuracy did not improve from 0.89160
Epoch 9/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0736 - accuracy: 0.9814 - val_loss: 0.3408 - val_accuracy: 0.8797

Epoch 00009: val_accuracy did not improve from 0.89160
Epoch 10/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0593 - accuracy: 0.9879 - val_loss: 0.3693 - val_accuracy: 0.8770

Epoch 00010: val_accuracy did not improve from 0.89160
Epoch 11/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0483 - accuracy: 0.9890 - val_loss: 0.4314 - val_accuracy: 0.8677

Epoch 00011: val_accuracy did not improve from 0.89160
Epoch 12/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0391 - accuracy: 0.9921 - val_loss: 0.4176 - val_accuracy: 0.8742

Epoch 00012: val_accuracy did not improve from 0.89160
Epoch 13/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0318 - accuracy: 0.9943 - val_loss: 0.4468 - val_accuracy: 0.8712

Epoch 00013: val_accuracy did not improve from 0.89160
Epoch 14/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0235 - accuracy: 0.9964 - val_loss: 0.4748 - val_accuracy: 0.8724

Epoch 00014: val_accuracy did not improve from 0.89160
Epoch 15/20
30/30 [==============================] - 1s 36ms/step - loss: 0.0174 - accuracy: 0.9976 - val_loss: 0.5069 - val_accuracy: 0.8698
```

```
%tensorboard --logdir logs/fit
```



☐ Show data download links
☐ Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis
STEP RELATIVE WALL

Runs
Write a regex to filter runs
☐ train
☐ validation
TOGGLE ALL RUNS
logs/fit

Filter tags (regular expressions supported)

epoch_accuracy

epoch_loss

With Check point, vaildation accuracy increased form epochs 1,2,3 and 5

epoch 1 - accuracy increased from 0.7792 to 0.8710

epoch 2 - accuracy increased from 0.8710 to 0.8809

epoch 3 - accuracy increased from 0.8758 to 0.8855

epoch 5 - accuracy increased from 0.8881 to 0.8888

0s completed at 7:36 AM