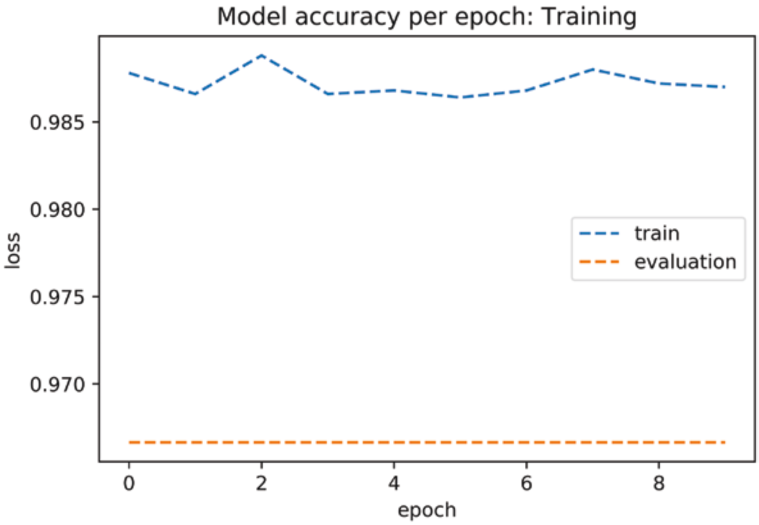


Figure 30-14. *Model accuracy per epoch*

TensorBoard with Keras

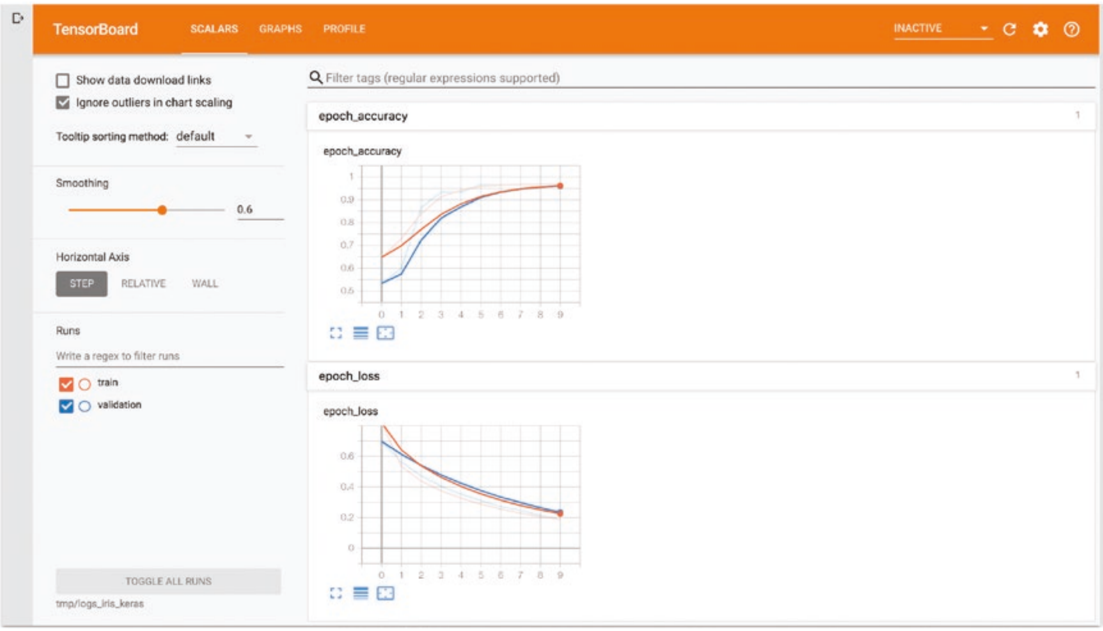
To visualize models with TensorBoard, attach a TensorBoard callback **'tf.keras.callbacks.TensorBoard()'** to the **'model.fit()'** method before training the model. The model graph, scalars, histograms, and other metrics are stored as event files in the log directory.

For this example, we modify the Iris model to use TensorBoard. The TensorBoard output is shown in Figure 30-15.

```
!pip install -q tensorflow==2.0.0-beta0

# import packages
import tensorflow as tf
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# load the TensorBoard notebook extension
%load_ext tensorboard

# dataset url
train_data_url = "https://storage.googleapis.com/download.tensorflow.org/
data/iris_training.csv"
```

```
test_data_url = "https://storage.googleapis.com/download.tensorflow.org/
data/iris_test.csv"

# define column names
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
'species']

# download and load the csv files
train_data = pd.read_csv(tf.keras.utils.get_file('iris_train.csv',
train_data_url),
                               skiprows=1, header=None, names=columns)

test_data = pd.read_csv(tf.keras.utils.get_file('iris_test.csv', test_data_url),
                               skiprows=1, header=None, names=columns)

# separate the features and targets
(X_train, y_train) = (train_data.iloc[:,0:-1], train_data.iloc[:,-1])
(X_test, y_test) = (test_data.iloc[:,0:-1], test_data.iloc[:,-1])

# apply one-hot encoding to targets
y_train=tf.keras.utils.to_categorical(y_train)
y_test=tf.keras.utils.to_categorical(y_test)

# create the functional model
def model_fn():
    # Model input
    model_input = tf.keras.layers.Input(shape=(4,))
    # Adds a densely-connected layer with 32 units to the model:
    x = tf.keras.layers.Dense(32, activation='relu')(model_input)
    # Add a softmax layer with 3 output units:
    predictions = tf.keras.layers.Dense(3, activation='softmax')(x)

    # the model
    model = tf.keras.Model(inputs=model_input,
                           outputs=predictions,
                           name='iris_model')
    # compile the model
    model.compile(optimizer='sgd',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

# parameters
batch_size=50

# use tf.data to batch and shuffle the dataset
train_ds = tf.data.Dataset.from_tensor_slices(
    (X_train.values, y_train)).shuffle(len(X_train)).repeat().batch(batch_size)
test_ds = tf.data.Dataset.from_tensor_slices((X_test.values, y_test)).
batch(batch_size)

# build train model
model = model_fn()

# print train model summary
model.summary()

# tensorboard
tensorboard = tf.keras.callbacks.TensorBoard(log_dir='./tmp/logs_iris_keras',
                                             histogram_freq=0, write_
                                             graph=True,
                                             write_images=True)

# assign callback
callbacks = [tensorboard]

# train the model
history = model.fit(train_ds, epochs=10,
                    steps_per_epoch=100,
                    validation_data=test_ds,
                    callbacks=callbacks)
```

```
# evaluate the model
score = model.evaluate(test_ds)
print('Test loss: {:.2f} \nTest accuracy: {:.2f}%'.format(score[0],
score[1]*100))

# execute the command to run TensorBoard
%tensorboard --logdir tmp/logs_iris_keras
```



□ Show data download links
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

☑ ○ train
☑ ○ validation

tmp/logs_iris_keras

Figure 30-15. TensorBoard output of Iris model

Checkpointing to Select Best Models

Checkpointing makes it possible to save the weights of the neural network model when there is an increase in the validation accuracy metric. This is achieved in Keras using the **'tf.keras.callbacks.ModelCheckpoint()'**. The saved weights can then be loaded back into the model and used to make predictions. Using the Iris dataset, we'll build a model that saves the weights to file only when there is an improvement in the validation set performance. For completeness sake as we have done in the previous segments, we will produce this example within a complete code listing.

```
!pip install -q tensorflow==2.0.0-beta0
```