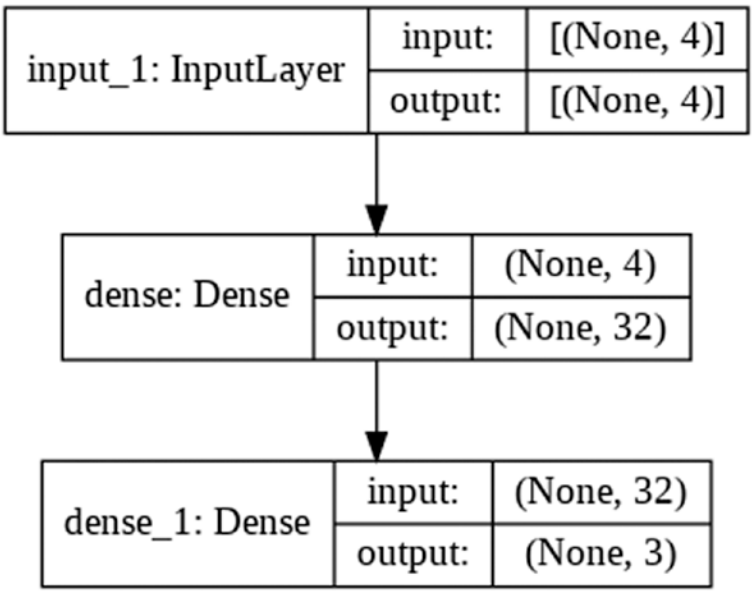

Figure 30-12. *The graph of the model – produced with Keras*

Model Visualization with Keras

With Keras, it is quite easy and straightforward to plot the metrics of the model to have a better graphical perspective as to how the model is performing for every training epoch. This view is also useful for dealing with issues of bias or variance of the model.

A callback function of the 'model.fit()' method returns the loss and evaluation score for each epoch. This information is stored in a variable and plotted.

In this example, we use the same Iris dataset model to illustrate visualization with Keras. The plots of the loss and accuracy of the model at each epoch are shown in Figure 30-13 and Figure 30-14, respectively.

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

```
# import packages
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
```

```
# 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()

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

# list metrics returned from callback function
history.history.keys()

# plot loss metric
plt.figure(1)
plt.plot(history.history['loss'], '--')
plt.plot(history.history['val_loss'], '--')
plt.title('Model loss per epoch: Training')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'evaluation'])
plt.show()
```

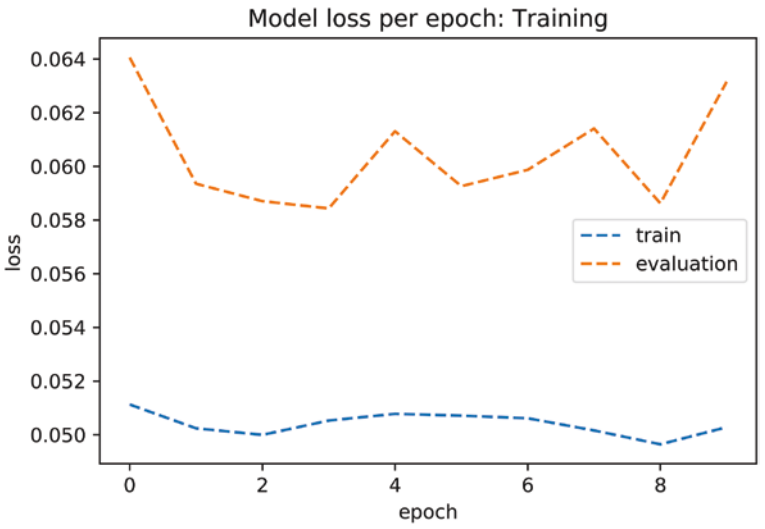

Figure 30-13. *Model loss per epoch*

```
# plot accuracy metric
plt.figure(2)
plt.plot(history.history['accuracy'], '--')
plt.plot(history.history['val_accuracy'], '--')
plt.title('Model accuracy per epoch: Training')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'evaluation'])
plt.show()
```

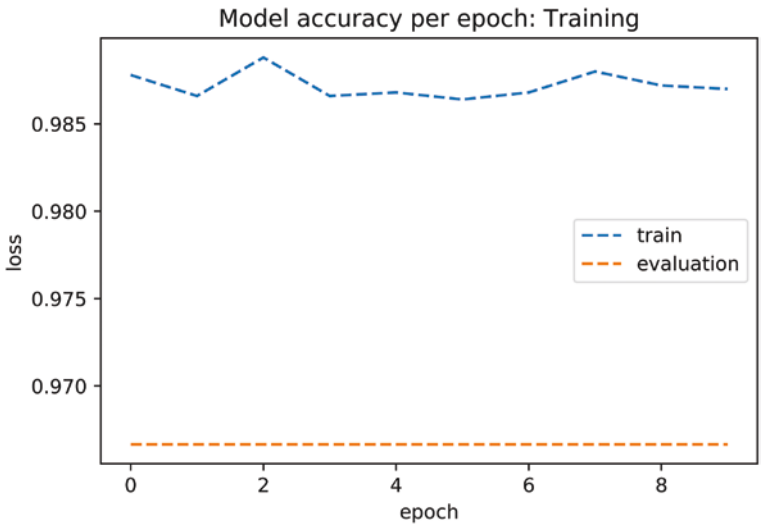

Figure 30-14. *Model accuracy per epoch*

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"
```