In [1]:

```
from __future__ import absolute_import, division, print_function
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

In [3]:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

In [4]:

```
train_images.shape
```

Out[4]:

```
(60000, 28, 28)
```

In [5]:

```
len(train_labels)
```

Out[5]:

```
60000
```

In [6]:

```
train_labels
```

Out[6]:

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

In [7]:

```
test_images.shape
```

Out[7]:

```
(10000, 28, 28)
```

In [8]:

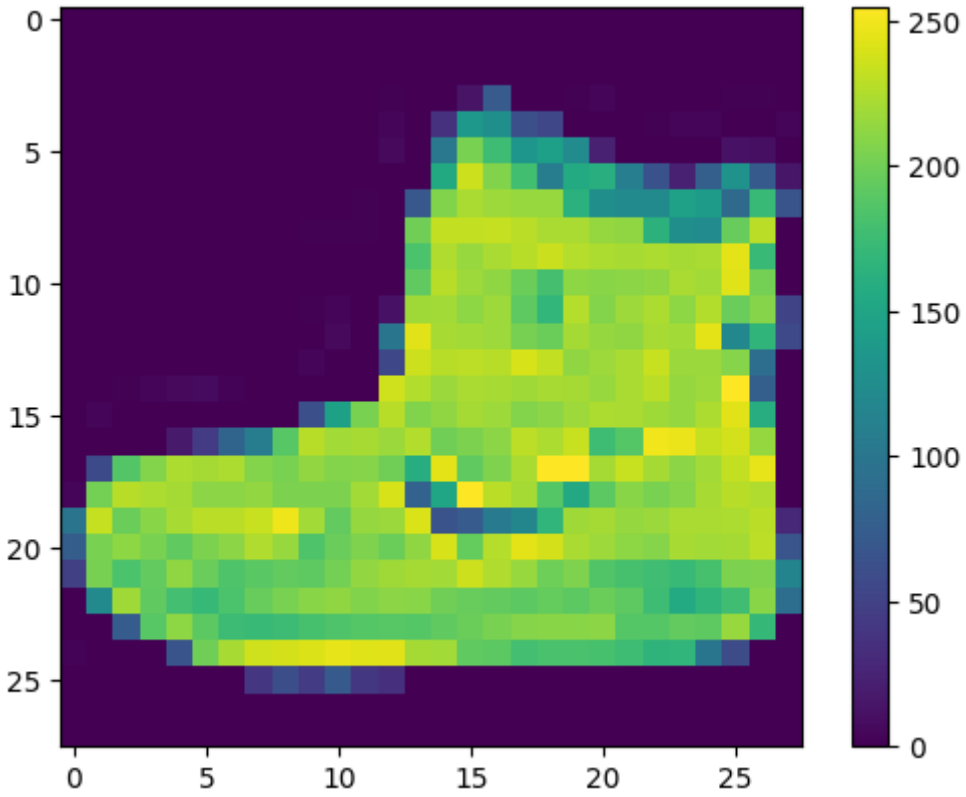
```
len(test_labels)
```

Out[8]:

```
10000
```

In [9]:

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



In [10]:

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

In [11]:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

In [12]:

```
model = keras.Sequential([
keras.layers.Flatten(input_shape=(28, 28)),
keras.layers.Dense(128, activation=tf.nn.relu),
keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

In [13]:

```
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

In [14]:

```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.4977 -
accuracy: 0.8247
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3754 -
accuracy: 0.8654
Epoch 3/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3364 -
accuracy: 0.8775
Epoch 4/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3139 -
accuracy: 0.8850
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2955 -
accuracy: 0.8901
```

Out[14]:

```
<keras.callbacks.History at 0x7fd76ff81a20>
```

In [15]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3584 - ac
curacy: 0.8676
Test accuracy: 0.8676000237464905
```

In [16]:

```
predictions = model.predict(test_images)
```

```
313/313 [==============================] - 1s 2ms/step
```

In [17]:

```
predictions[0]
```

Out[17]:

```
array([1.3383218e-05, 4.0095873e-07, 1.5446010e-06, 3.4709288e-07,
       5.4506000e-07, 4.1185790e-03, 2.7024946e-05, 5.5574376e-02,
       8.2644292e-05, 9.4018114e-01], dtype=float32)
```

In [18]:

```
np.argmax(predictions[0])
```

Out[18]:

9

In [19]:

```
test_labels[0]
```

Out[19]:

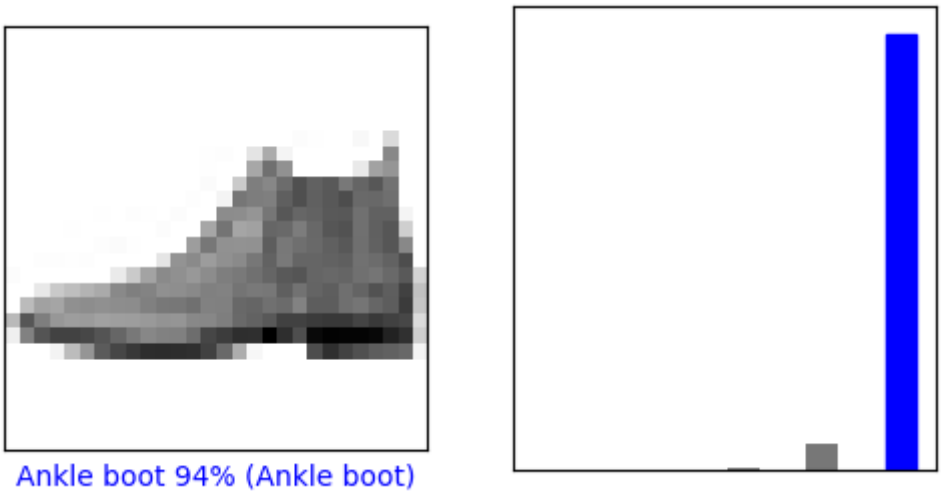
9

In [20]:

```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
   color = 'blue'
  else:
   color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                       100*np.max(predictions_array),
                                       class_names[true_label]),
                                       color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

In [21]:

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
plt.show()
```



In [22]:

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
plt.show()
```

In [23]:

```
# Plot the first X test images, their predicted label, and the true label
# Color correct predictions in blue, incorrect predictions in red
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
   plt.subplot(num_rows, 2*num_cols, 2*i+1)
   plot_image(i, predictions, test_labels, test_images)
   plt.subplot(num_rows, 2*num_cols, 2*i+2)
   plot_value_array(i, predictions, test_labels)
plt.show()
```

In [24]:

```
# Grab an image from the test dataset
img = test_images[0]
print(img.shape)
```

```
(28, 28)
```

In [25]:

```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))
print(img.shape)
```

```
(1, 28, 28)
```

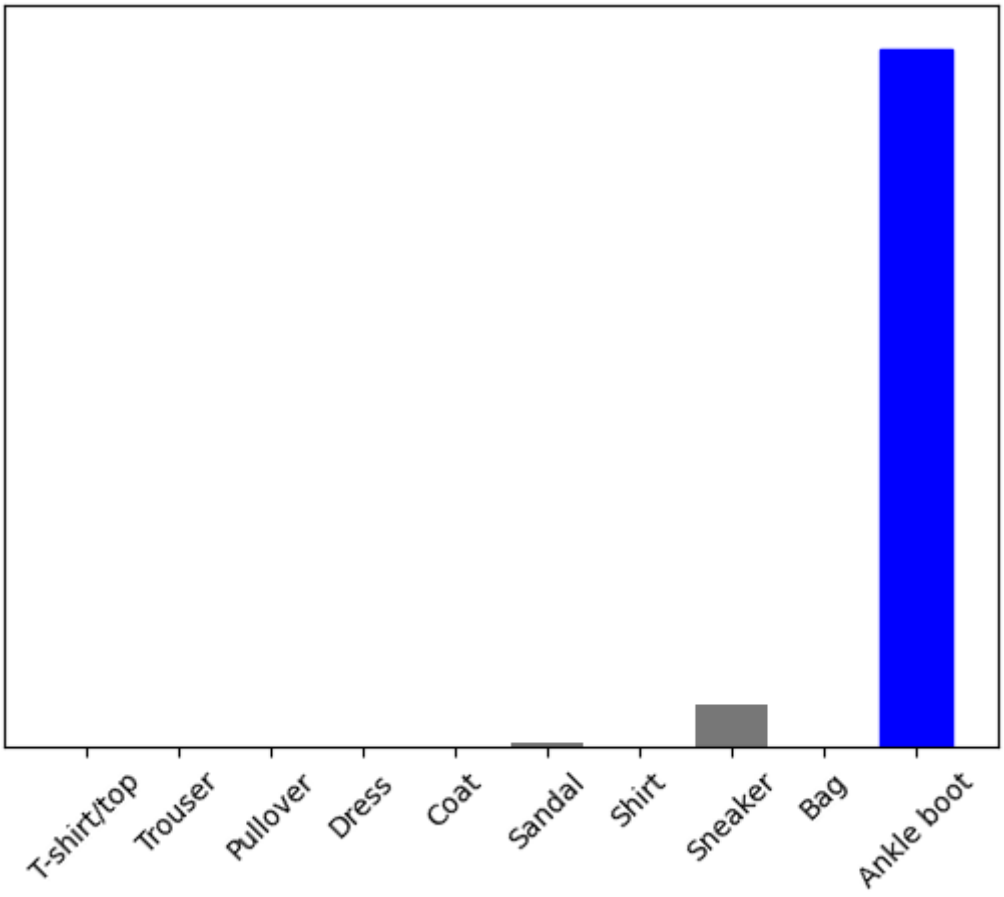
In [26]:

```
predictions_single = model.predict(img)
print(predictions_single)
```

```
1/1 [==============================] - 0s 22ms/step
[[1.3383194e-05 4.0095838e-07 1.5445967e-06 3.4709291e-07 5.4505898e-07
  4.1185752e-03 2.7024947e-05 5.5574380e-02 8.2644219e-05 9.4018120e-01]]
```

In [27]:

```
plot_value_array(0, predictions_single, test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



In [28]:

```
np.argmax(predictions_single[0])
```

Out[28]:

```
9
```

In [ ]:

```

```