訓練模型(Train the model)

訓練神經網絡模型需要執行以下步驟：

1. 將訓練數據輸入模型。在這個例子中，訓練數據是在 `train_images` 和 `train_labels` 陣列。
2. 該模型學會副圖像和標籤。
3. 您要求模型對測試集進行預測（在此示例中為test_images數組）。
4. 驗證預測是否與 `test_labels` 陣列中的標籤匹配。

Training the neural network model requires the following steps:

1. Feed the training data to the model. In this example, the training data is in the `train_images` and `train_labels` arrays.
2. The model learns to associate images and labels.
3. You ask the model to make predictions about a test set—in this example, the `test_images` array.
4. Verify that the predictions match the labels from the `test_labels` array.

餵模型(Feed the model)

要開始訓練，請調用該 `model.fit` 方法，之所以這樣稱呼是因為該方法使模型“適合”訓練數據

To start training, call the `model.fit` method—so called because it "fits" the model to the training data:

In [16]:

```
model.fit(train_images, train_labels, epochs=10)
```

```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 4s 60us/sample - loss: 0.4926
- accuracy: 0.8256
Epoch 2/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.3742
- accuracy: 0.8650
Epoch 3/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.3360
- accuracy: 0.8769
Epoch 4/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.3118
- accuracy: 0.8856
Epoch 5/10
60000/60000 [==============================] - 3s 54us/sample - loss: 0.2928
- accuracy: 0.8924
Epoch 6/10
60000/60000 [==============================] - 3s 49us/sample - loss: 0.2808
- accuracy: 0.8963
Epoch 7/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.2673
- accuracy: 0.9010
Epoch 8/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.2565
- accuracy: 0.9051
Epoch 9/10
60000/60000 [==============================] - 3s 49us/sample - loss: 0.2458
- accuracy: 0.9075
Epoch 10/10
60000/60000 [==============================] - 3s 50us/sample - loss: 0.2376
- accuracy: 0.9115
```

Out[16]:

```
<tensorflow.python.keras.callbacks.History at 0x1f5cda4cf48>
```

由於模型火車，顯示損失和精度指標。該模型在訓練數據上達到約0.91（或91%）的精度。

As the model trains, the loss and accuracy metrics are displayed. This model reaches an accuracy of about 0.91 (or 91%) on the training data.

評估準確性(Evaluate accuracy)

接下來，比較模型在測試數據集上的表現：

Next, compare how the model performs on the test dataset:

In [17]:

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
10000/1 - 0s - loss: 0.2209 - accuracy: 0.8838

Test accuracy: 0.8838
```

事實證明，測試數據集的準確性略低於訓練數據集的準確性。訓練準確性和測試準確性之間的差距代表過度擬合。過度擬合是當一個機器學習模型進行比對在訓練數據上新的，以前看不到的投入差。過度擬合的模型“存儲”訓練數據，而測試數據的準確性較低。有關更多信息，請參見以下內容：

- 證明過度擬合 (https://www.tensorflow.org/tutorials/keras/overfit_and_underfit#demonstrate_overfitting)
- 防止過度擬合的策略 (https://www.tensorflow.org/tutorials/keras/overfit_and_underfit#strategies_to_prevent_overfitting)

It turns out that the accuracy on the test dataset is a little less than the accuracy on the training dataset. This gap between training accuracy and test accuracy represents *overfitting*. Overfitting is when a machine learning model performs worse on new, previously unseen inputs than on the training data. An overfitted model "memorizes" the training data—with less accuracy on testing data. For more information, see the following:

- Demonstrate overfitting (https://www.tensorflow.org/tutorials/keras/overfit_and_underfit#demonstrate_overfitting)
- Strategies to prevent overfitting (https://www.tensorflow.org/tutorials/keras/overfit_and_underfit#strategies_to_prevent_overfitting)

作出預測(Make predictions)

通過訓練模型，您可以使用它來預測某些圖像。模型的線性輸出logits。附加一個softmax層，以將logits (https://developers.google.com/machine-learning/glossary#logits)轉換為更容易解釋的概率。

With the model trained, you can use it to make predictions about some images. The model's linear outputs, logits (https://developers.google.com/machine-learning/glossary#logits). Attach a softmax layer to convert the logits to probabilities, which are easier to interpret.

In [18]:

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
```

In [19]:

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

在這裡，模型已經預測了測試集中每個圖像的標籤。讓我們看一下第一個預測：

Here, the model has predicted the label for each image in the testing set. Let's take a look at the first prediction:

In [20]:

```
predictions[0]
```

Out[20]:

```
array([1.8260094e-08, 7.7989943e-09, 9.0727440e-08, 5.6814780e-08,
       9.2827264e-08, 2.1960903e-02, 2.8113501e-08, 2.0588467e-02,
       7.2089841e-09, 9.5745033e-01], dtype=float32)
```

預測是10個數字組成的數組。它們代表模型對圖像對應於10種不同服裝中的每一種的“信心”。您可以看到哪個標籤的置信度最高：

A prediction is an array of 10 numbers. They represent the model's "confidence" that the image corresponds to each of the 10 different articles of clothing. You can see which label has the highest confidence value:

In [21]:

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

Out[21]:

```
9
```

因此，模型最有信心該圖像是踝靴或class_names[9]。檢查測試標籤表明此分類是正確的：

So, the model is most confident that this image is an ankle boot, or `class_names[9]` . Examining the test label shows that this classification is correct:

In [22]:

```
test_labels[0]
```

Out[22]:

```
9
```

以圖形方式查看完整的10個類預測。

Graph this to look at the full set of 10 class predictions.

In [23]:

```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, 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, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  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')
```

驗證預測(Verify predictions)

通過訓練模型，您可以使用它來預測某些圖像。

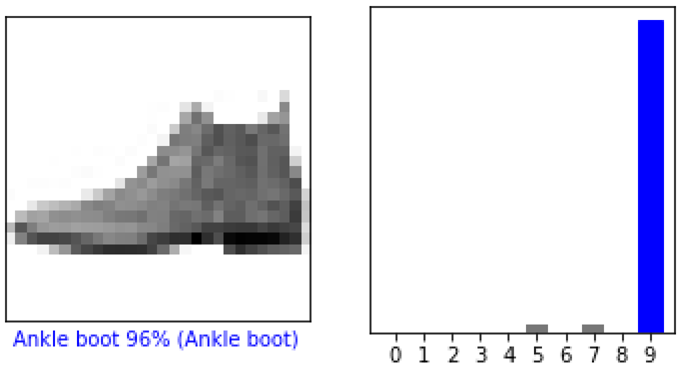
With the model trained, you can use it to make predictions about some images.

讓我們看一下第0張圖像，預測和預測數組。正確的預測標籤為藍色，錯誤的預測標籤為紅色。該數字給出了預測標籤的百分比（滿分為100）。

Let's look at the 0th image, predictions, and prediction array. Correct prediction labels are blue and incorrect prediction labels are red. The number gives the percentage (out of 100) for the predicted label.

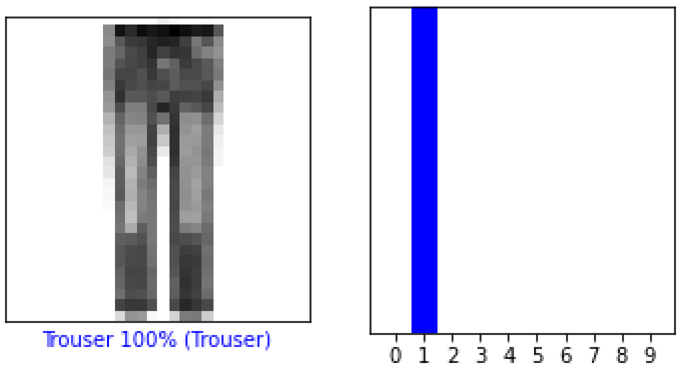
In [24]:

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



In [25]:

```
i = 15
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```

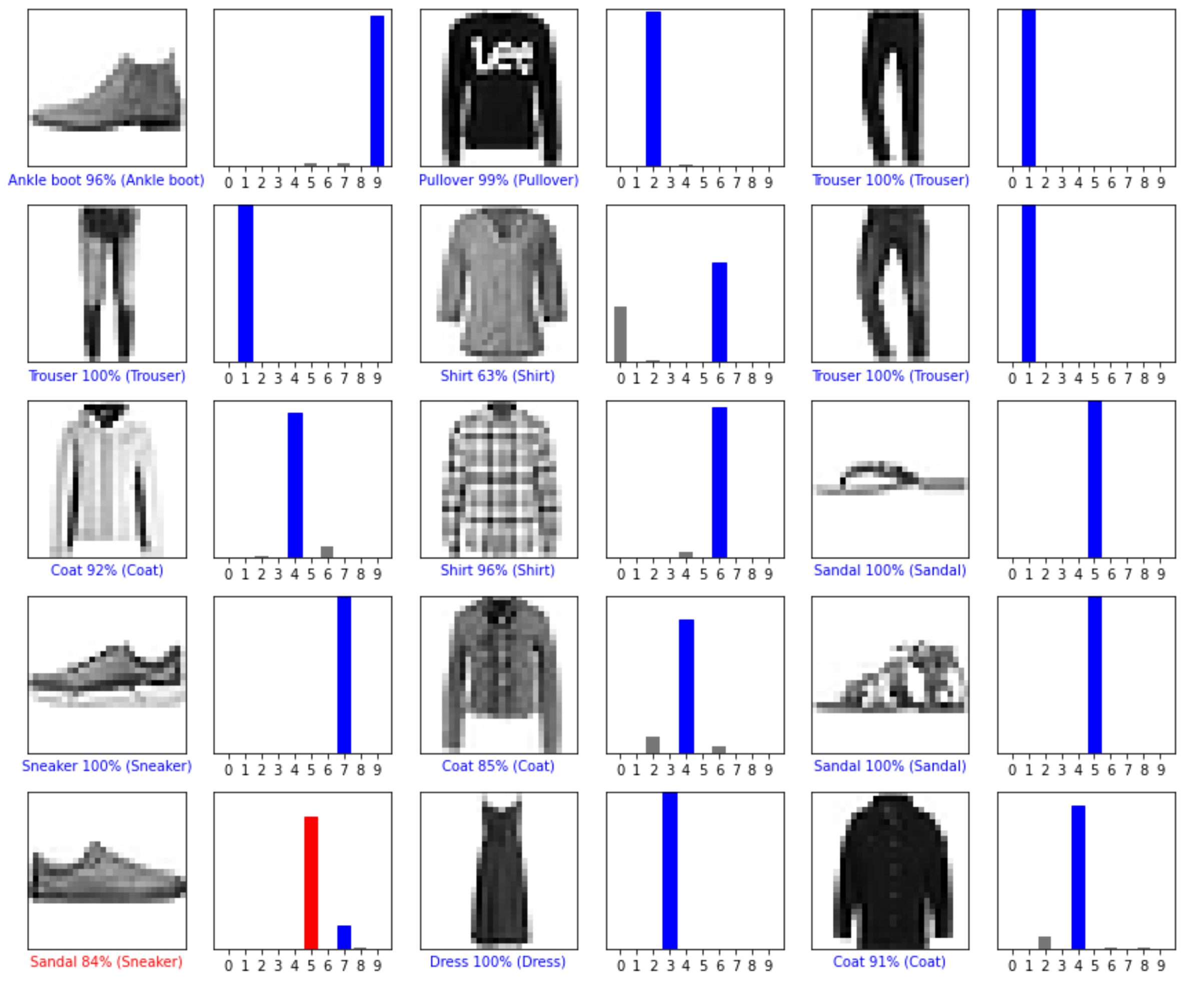


讓我們繪製一些帶有預測的圖像。請注意，即使非常自信，該模型也可能是錯誤的。

Let's plot several images with their predictions. Note that the model can be wrong even when very confident.

In [26]:

```python
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and 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[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```



使用訓練有素的模型(Use the trained model)

最後，使用經過訓練的模型對單個圖像進行預測。

Finally, use the trained model to make a prediction about a single image.

In [27]:

```
# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)
```

```
(28, 28)
```

`tf.keras` 對模型進行了優化，可以一次對一*批*或一組示例進行預測。因此，即使您使用的是單個圖像，也需要將其添加到列表中：

`tf.keras` models are optimized to make predictions on a *batch*, or collection, of examples at once. Accordingly, even though you're using a single image, you need to add it to a list:

In [28]:

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

print(img.shape)
```

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

現在，為該圖像預測正確的標籤：

Now predict the correct label for this image:

In [29]:

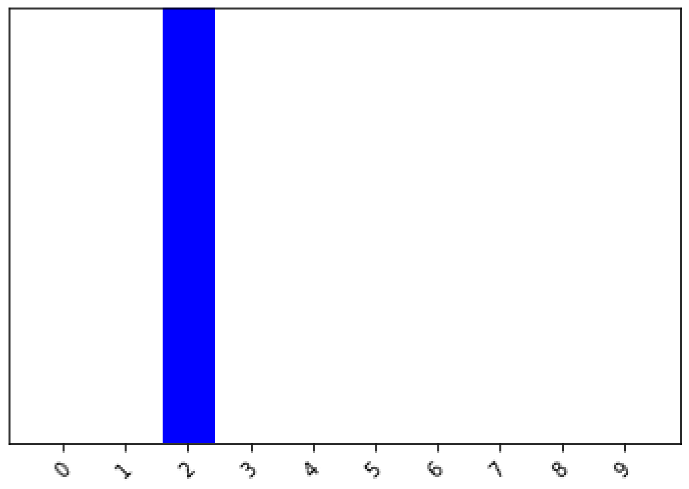
```
predictions_single = probability_model.predict(img)

print(predictions_single)
```

```
[[3.56780474e-05 9.90337695e-13 9.89654541e-01 1.28765471e-11
  9.40908119e-03 6.03479376e-12 9.00745683e-04 1.05521876e-16
  2.13636914e-10 1.37546239e-15]]
```

In [26]:

```
plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



`keras.Model.predic` t返回列表列表-數據批次中每個圖像的一個列表。批量獲取我們（僅）圖像的預測：

`keras.Model.predict` returns a list of lists—one list for each image in the batch of data. Grab the predictions for our (only) image in the batch:

In [30]:

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

Out[30]:

```
2
```

該模型將按預期預測標籤。

And the model predicts a label as expected.