```
import tensorflow as tf
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
print(tf.__version__)
```

```
2.15.0
```

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

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

```
train_images.shape
```

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

```
len(train_labels)
```

```
60000
```

```
train_labels
```

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

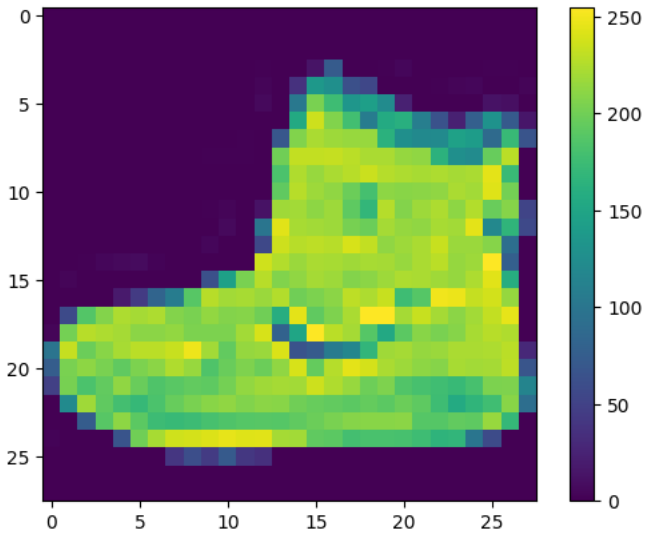
```
test_images.shape
```

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

```
len(test_labels)
```

```
10000
```

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



```
train_images = train_images / 255.0

test_images = test_images / 255.0


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()
```


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


model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])


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

```
Epoch 1/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.4986 - accuracy: 0.8244
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3798 - accuracy: 0.8630
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3371 - accuracy:
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3126 - accura
Epoch 5/10
1875/1875 [==============================] - 14s 8ms/step - loss: 0.2926 - accura
Epoch 6/10
```

```
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2785 - accuracy: 0.8969
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2673 - accuracy: 0.9002
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2556 - accuracy: 0.9038
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2454 - accuracy: 0.9077
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2368 - accuracy: 0.9118
<keras.src.callbacks.History at 0x7ae862afd3c0>
```

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print('\nTest accuracy:', test_acc)
```

```
313/313 - 1s - loss: 0.3340 - accuracy: 0.8805 - 581ms/epoch - 2ms/step

Test accuracy: 0.8805000185966492
```

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

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

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

```
predictions[0]
```

```
array([3.3556644e-06, 9.3383733e-08, 7.3332618e-10, 2.9405577e-08,
       2.5048323e-08, 2.5304945e-04, 1.1252876e-06, 5.2847959e-02,
       4.2172996e-08, 9.4689435e-01], dtype=float32)
```

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

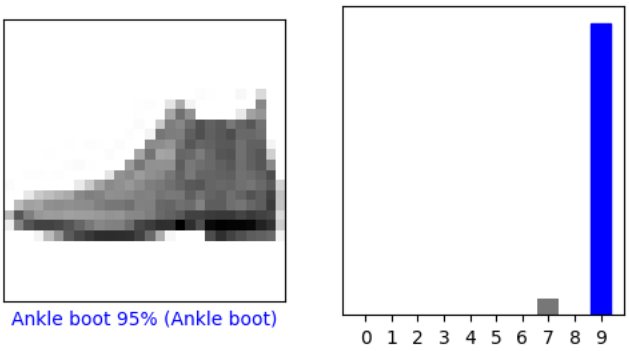
```
9
```

```
test_labels[0]
```

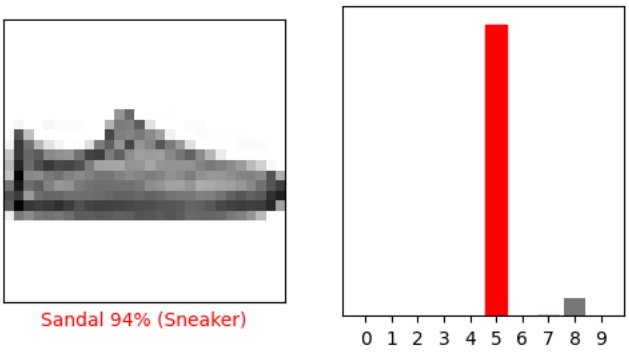
```
9
```

```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = 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):
  true_label = 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')
```

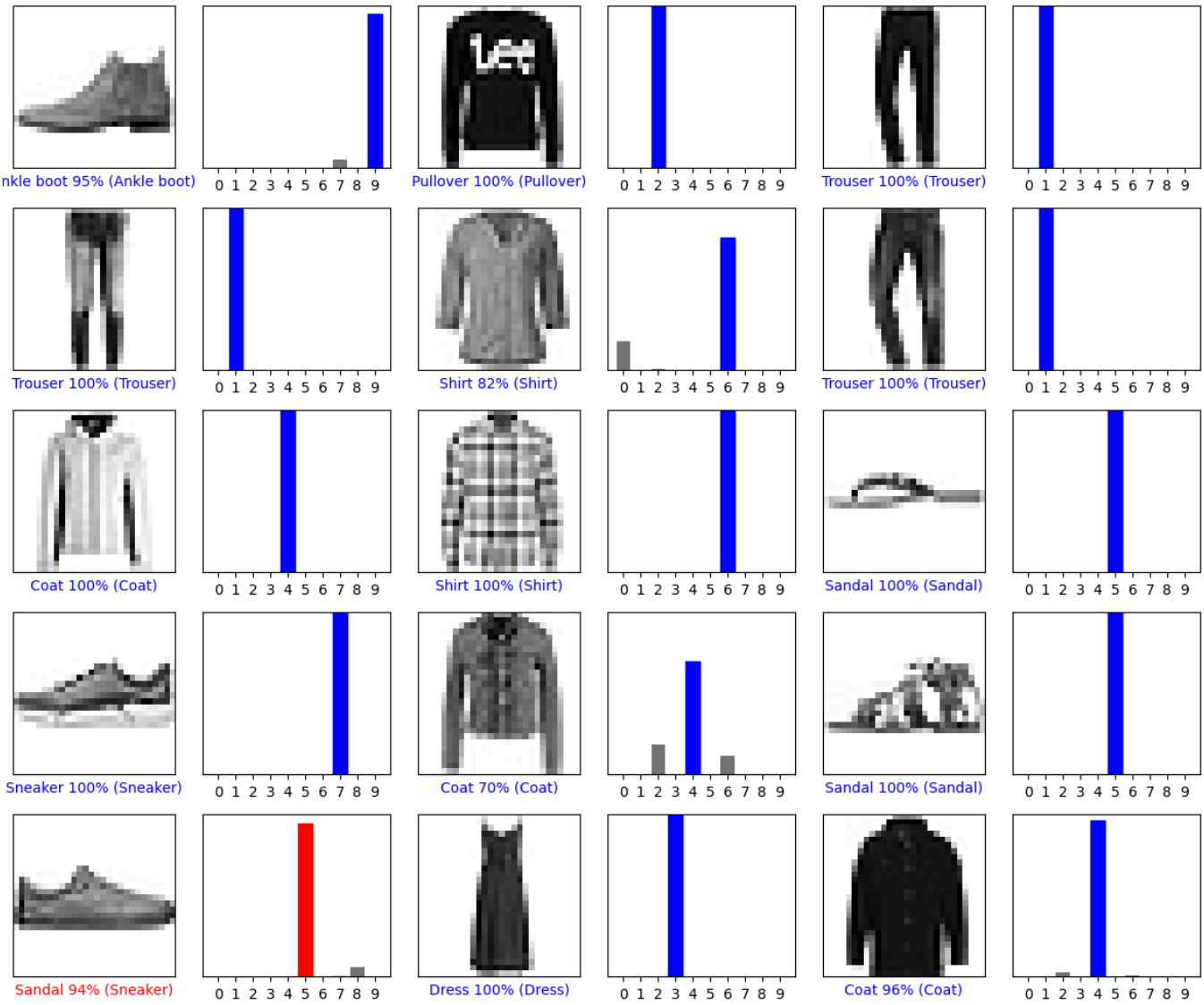
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```




```
i = 12
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()
```




```
# 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()
```




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

```
(28, 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)
```

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

```
1/1 [==============================] - 0s 21ms/step
[[8.8920269e-06 1.0686276e-13 9.9870610e-01 3.4587333e-11 9.5662149e-04
  5.8867157e-14 3.2835340e-04 1.2810071e-19 1.5400181e-10 2.9857014e-18]]
```

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