```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
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)
```

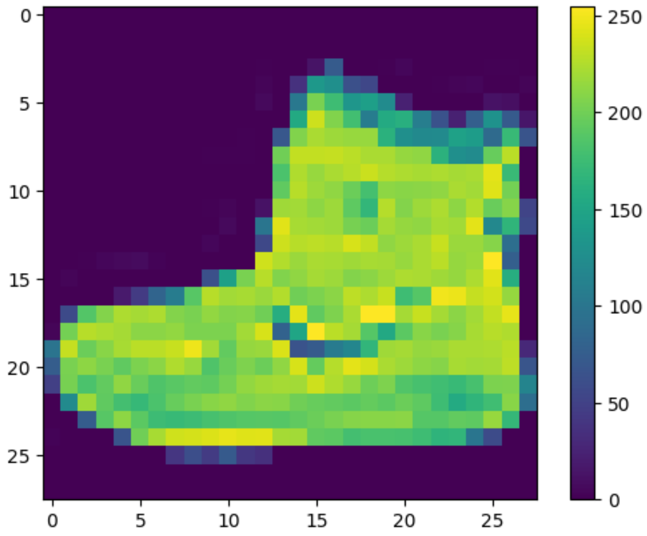
```
test_images.shape
```

```
(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=30)
```

```
Epoch 1/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.4981 - accuracy: 0.8242
Epoch 2/30
1875/1875 [==============================] - 14s 7ms/step - loss: 0.3718 - accuracy: 0.8659
Epoch 3/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3346 - accuracy: 0.8770
Epoch 4/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3131 - accuracy: 0.8857
Epoch 5/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2960 - accuracy: 0.8901
Epoch 6/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2819 - accuracy: 0.8952
Epoch 7/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2697 - accuracy: 0.9001
Epoch 8/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2578 - accuracy: 0.9049
Epoch 9/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2497 - accuracy: 0.9061
Epoch 10/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2390 - accuracy: 0.9115
Epoch 11/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2337 - accuracy: 0.9134
Epoch 12/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2246 - accuracy: 0.9162
Epoch 13/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2185 - accuracy: 0.9180
```

```
Epoch 14/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2103 - accuracy: 0.9220
Epoch 15/30
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2041 - accuracy: 0.9241
Epoch 16/30
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1993 - accuracy: 0.9250
Epoch 17/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1962 - accuracy: 0.9264
Epoch 18/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1905 - accuracy: 0.9286
Epoch 19/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1844 - accuracy: 0.9313
Epoch 20/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1808 - accuracy: 0.9328
Epoch 21/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1777 - accuracy: 0.9331
Epoch 22/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1734 - accuracy: 0.9345
Epoch 23/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1684 - accuracy: 0.9373
Epoch 24/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1624 - accuracy: 0.9388
Epoch 25/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1628 - accuracy: 0.9391
Epoch 26/30
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1562 - accuracy: 0.9411
Epoch 27/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1521 - accuracy: 0.9427
Epoch 28/30
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1505 - accuracy: 0.9437
Epoch 29/30
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1481 - accuracy: 0.9450
```

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

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

```
313/313 - 1s - loss: 0.3979 - accuracy: 0.8849 - 662ms/epoch - 2ms/step

Test accuracy: 0.8848999738693237
```

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

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

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

```
predictions[0]
```

```
array([1.2701780e-10, 2.7280393e-16, 2.1641811e-13, 2.9576729e-13,
       7.6283347e-15, 4.9248447e-05, 1.6708884e-14, 2.1604472e-05,
       6.0403946e-11, 9.9992913e-01], dtype=float32)
```

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

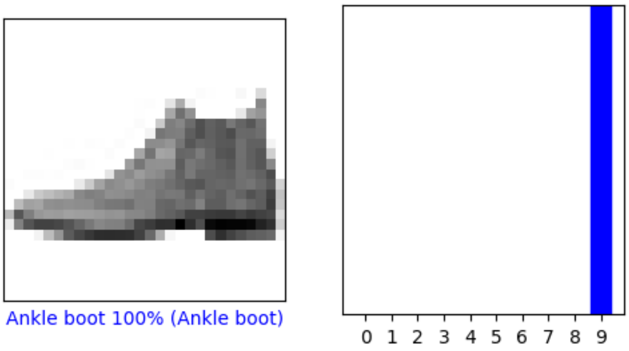
```
9
```

```
test_labels[0]
```

```
9
```

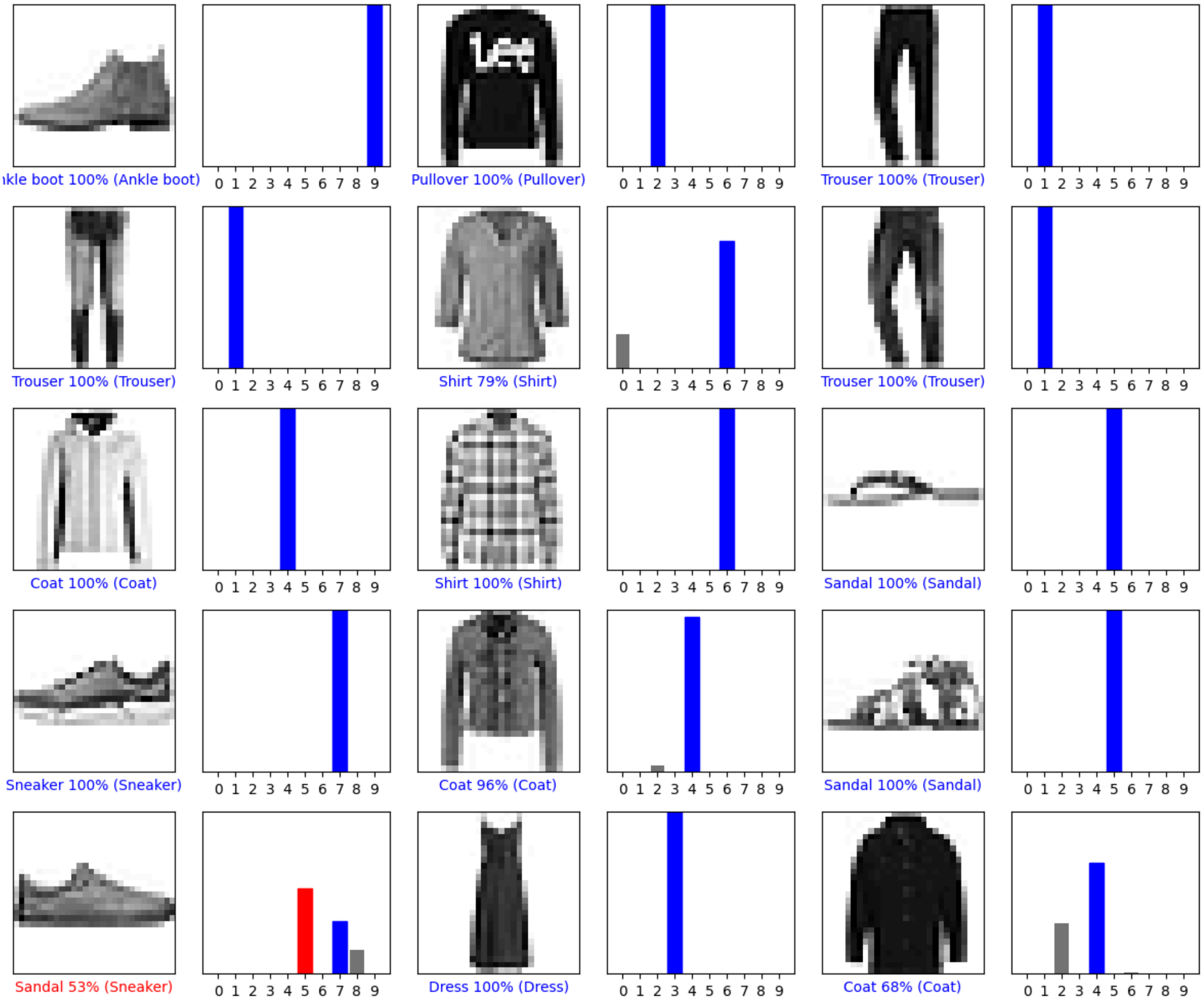
```
def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  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)
```