```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
```

```
TensorFlow 2.x selected.
```

```
from __future__ import absolute_import, division, print_function, unicode_literals

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

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.0.0
```

```
mnist = keras.datasets.mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

```
class_names = ['0', '1', '2', '3', '4',
               '5', '6', '7', '8', '9']


train_images.shape
```

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

```
len(train_labels)
```

```
60000
```

```
train_labels
```

```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

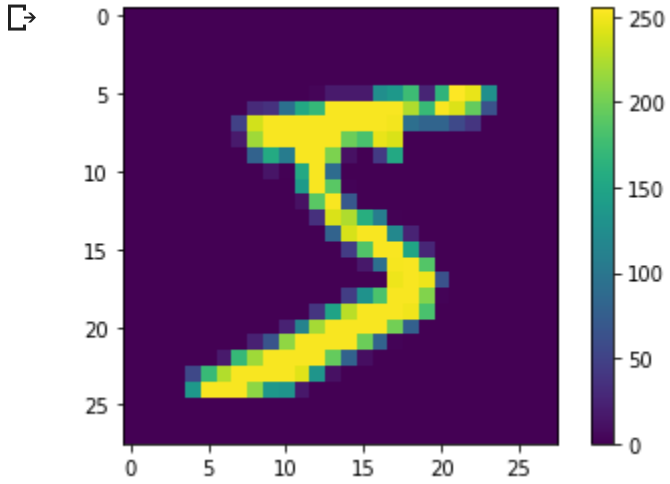
```
test_images.shape
```

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

```
len(test_labels)
```

```
10000
```

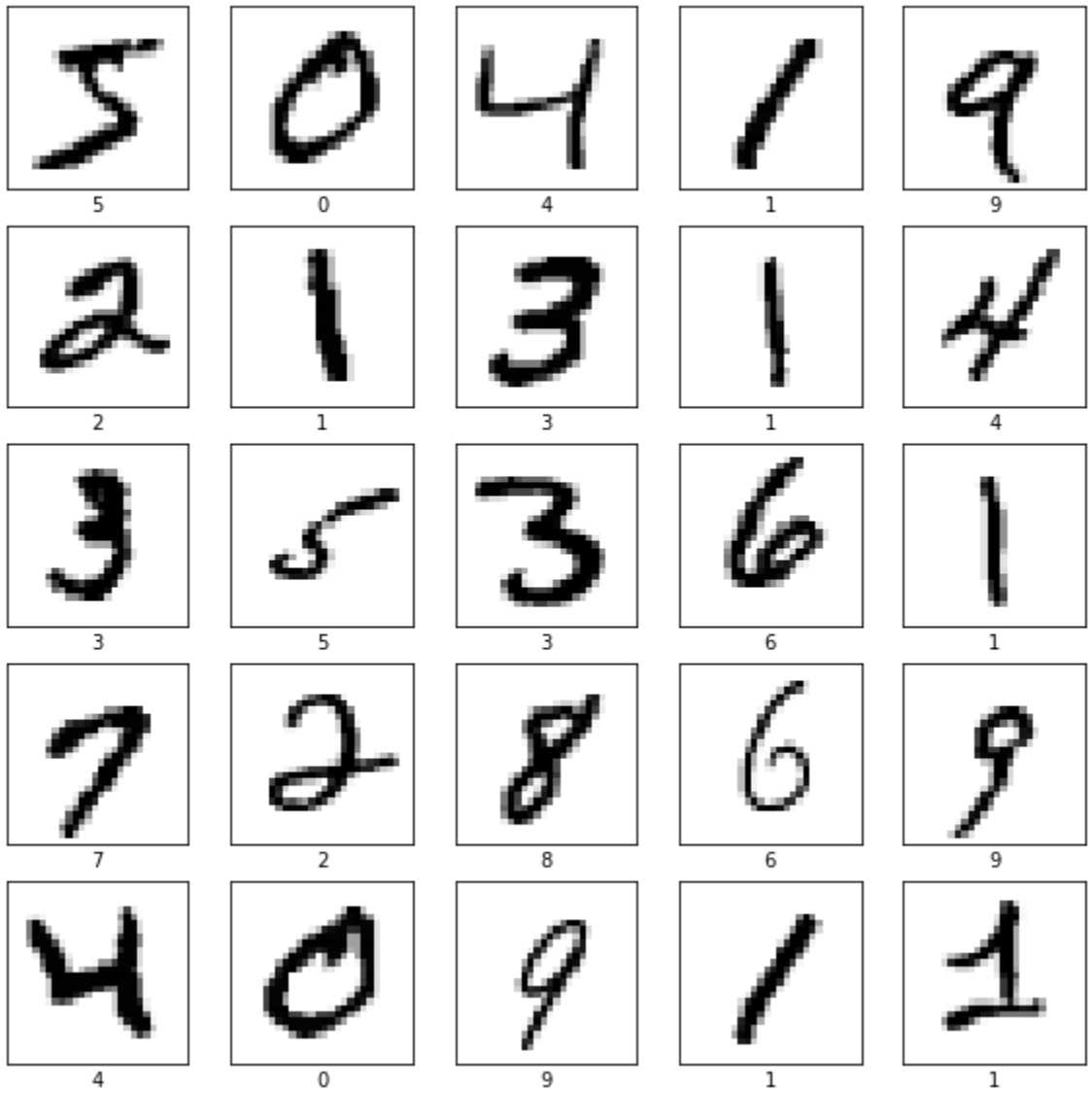
```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```




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

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


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

```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 5s 87us/sample - loss: 0.2583 - accuracy: 0.9259
Epoch 2/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.1129 - accuracy: 0.9661
Epoch 3/10
60000/60000 [==============================] - 5s 78us/sample - loss: 0.0777 - accuracy: 0.9761
Epoch 4/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0600 - accuracy: 0.9817
Epoch 5/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0465 - accuracy: 0.9851
Epoch 6/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0361 - accuracy: 0.9879
Epoch 7/10
60000/60000 [==============================] - 5s 78us/sample - loss: 0.0302 - accuracy: 0.9905
Epoch 8/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.0238 - accuracy: 0.9926
Epoch 9/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.0209 - accuracy: 0.9938
Epoch 10/10
60000/60000 [==============================] - 5s 82us/sample - loss: 0.0155 - accuracy: 0.9952
<tensorflow.python.keras.callbacks.History at 0x7f3b8bb5ae80>
```

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

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

```
10000/1 - 0s - loss: 0.0437 - accuracy: 0.9747

Test accuracy: 0.9747
```

```
predictions = model.predict(test_images)


predictions[0]
```

```
    array([1.27739180e-10, 1.11742626e-09, 5.13923162e-08, 1.19029595e-04,
           1.60261768e-14, 3.84478316e-09, 3.48356434e-16, 9.99879956e-01,
           2.62993805e-09, 1.01028832e-06], dtype=float32)
```

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

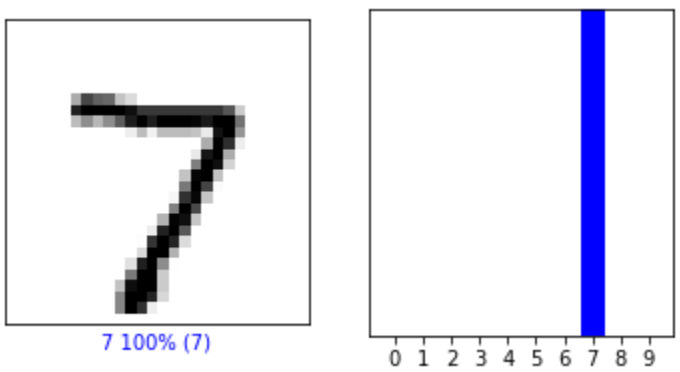
```
7
```

```
test_labels[0]
```

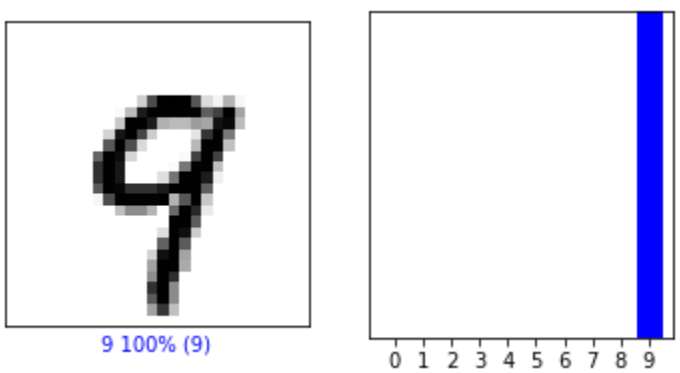
```
7
```

```
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')


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()
```

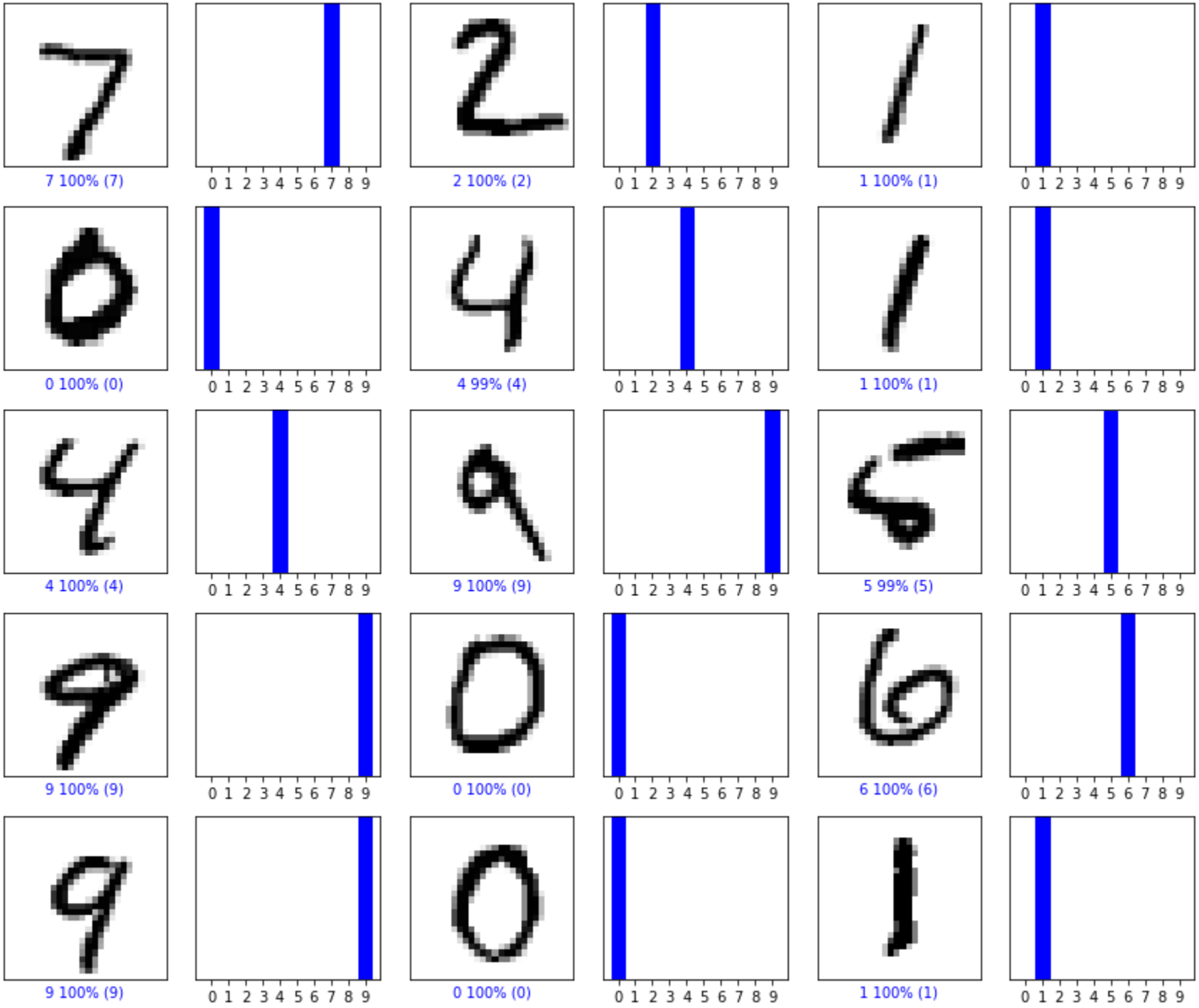



```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```




```
# 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)
```

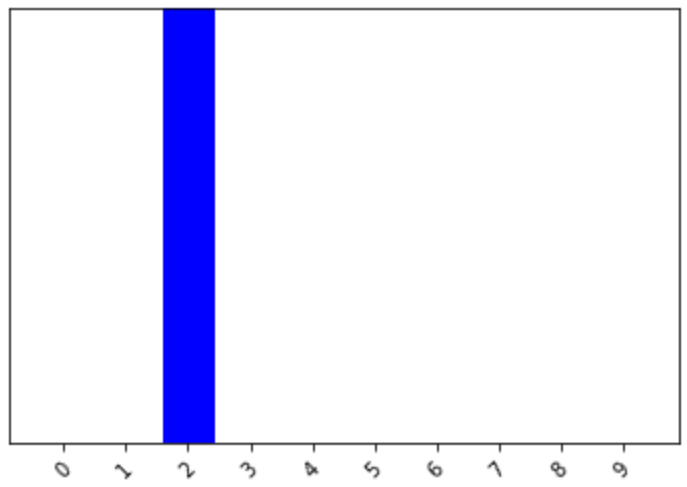
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

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

```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[7.8300815e-09 1.7448054e-06 9.9991965e-01 7.2086579e-05 1.5026756e-19
  4.1991325e-06 3.4572611e-07 3.2661133e-16 1.8941996e-06 5.0799174e-12]]
```

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




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

```
2
```