MNIST_Classification

May 12, 2019

In [ ]:
```
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__)
```

```
1.13.1
```

In [ ]:
```
mnist = keras.datasets.mnist

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

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

In [ ]:
```
train_images.shape
```

Out[ ]:
```
(60000, 28, 28)
```

In [ ]:
```
len(train_labels)
```

Out[ ]:
```
60000
```

In [ ]:
```
train_labels
```

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

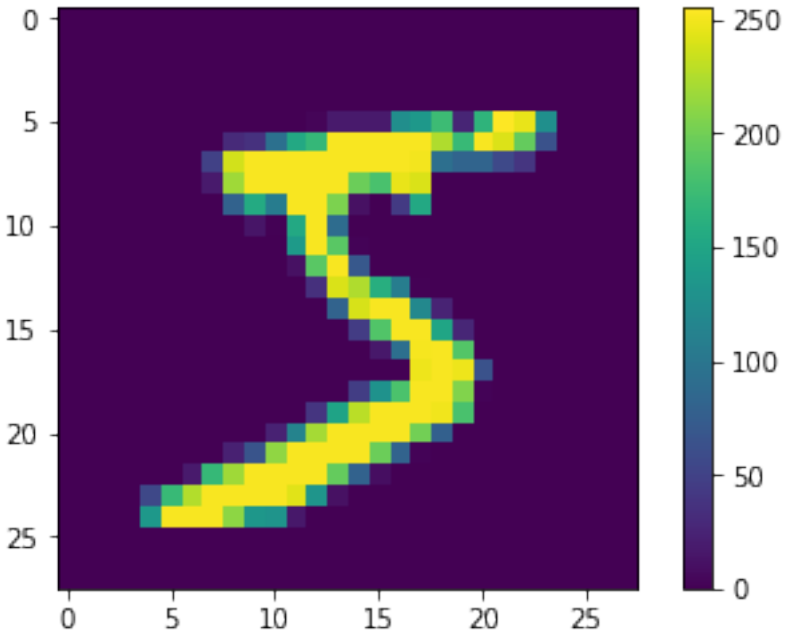
In [ ]:
```
test_images.shape
```

Out[ ]:
```
(10000, 28, 28)
```

In [ ]:
```
len(test_labels)
```

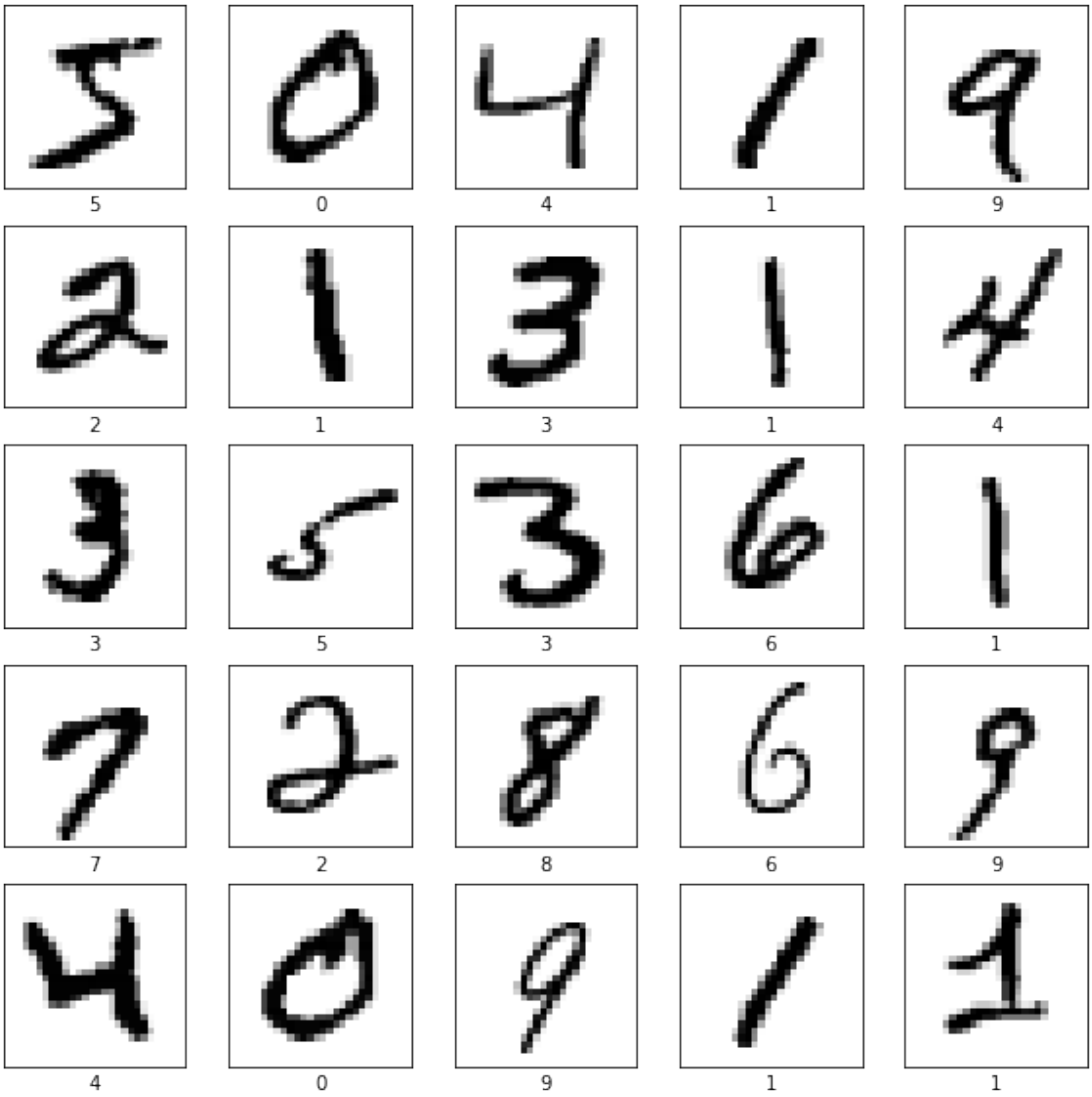
Out[ ]:
```
10000
```

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



In [ ]:
```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

In [ ]:
```
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 [ ]: 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)
        ])
```

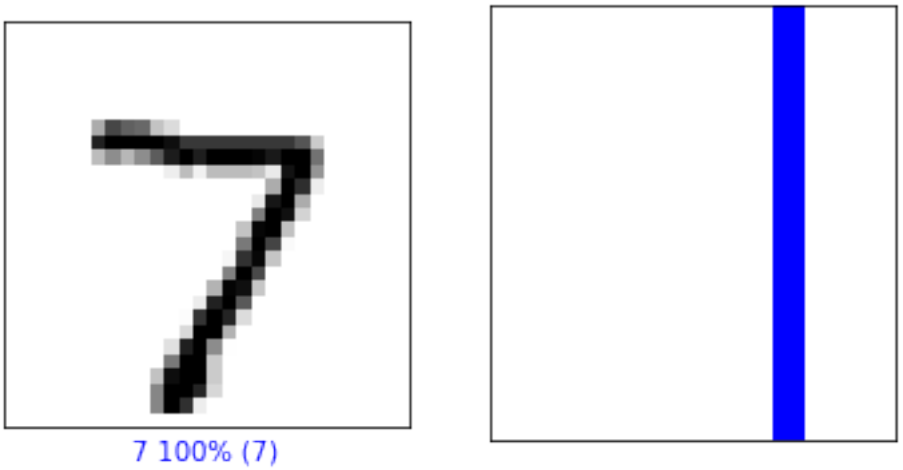
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resource_
Instructions for updating:
Colocations handled automatically by placer.
```

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

```
In [ ]: model.fit(train_images, train_labels, epochs=5)

Epoch 1/5
60000/60000 [==============================] - 8s 139us/sample - loss: 0.2628 - acc: 0.9242
Epoch 2/5
60000/60000 [==============================] - 8s 134us/sample - loss: 0.1166 - acc: 0.9663
Epoch 3/5
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0801 - acc: 0.9758
Epoch 4/5
60000/60000 [==============================] - 8s 137us/sample - loss: 0.0599 - acc: 0.9812
Epoch 5/5
60000/60000 [==============================] - 8s 137us/sample - loss: 0.0459 - acc: 0.9860


Out[ ]: <tensorflow.python.keras.callbacks.History at 0x7f9175bcfa90>

In [ ]: test_loss, test_acc = model.evaluate(test_images, test_labels)

        print('Test accuracy:', test_acc)

10000/10000 [==============================] - 0s 43us/sample - loss: 0.0877 - acc: 0.9741
Test accuracy: 0.9741


In [ ]: predictions = model.predict(test_images)

In [ ]: predictions[0]

Out[ ]: array([2.8120133e-09, 1.5560181e-10, 1.8059081e-06, 5.8903031e-05,
               3.4871784e-14, 2.4939219e-08, 1.3011033e-11, 9.9993920e-01,
               1.1663506e-07, 3.1092583e-08], dtype=float32)

In [ ]: np.argmax(predictions[0])

Out[ ]: 7

In [ ]: test_labels[0]

Out[ ]: 7

In [ ]: 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:
```

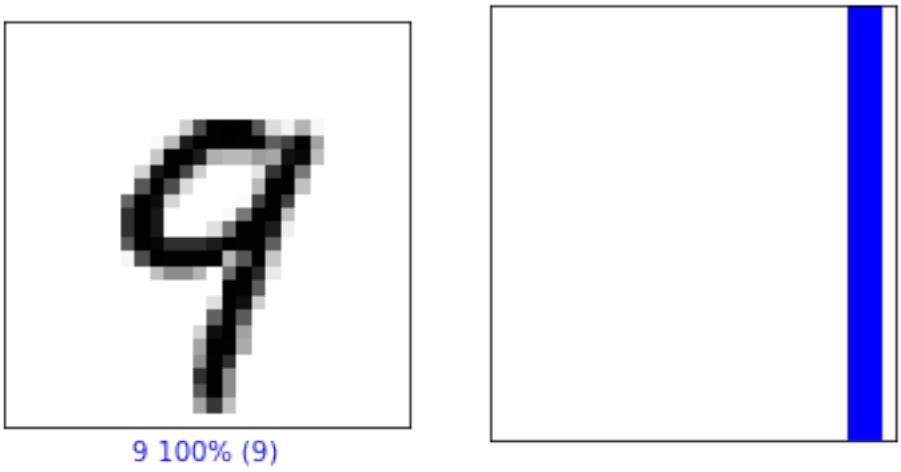
```
    color = 'blue'
  else:
    color = 'red'

  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 [ ]:
```
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()
```

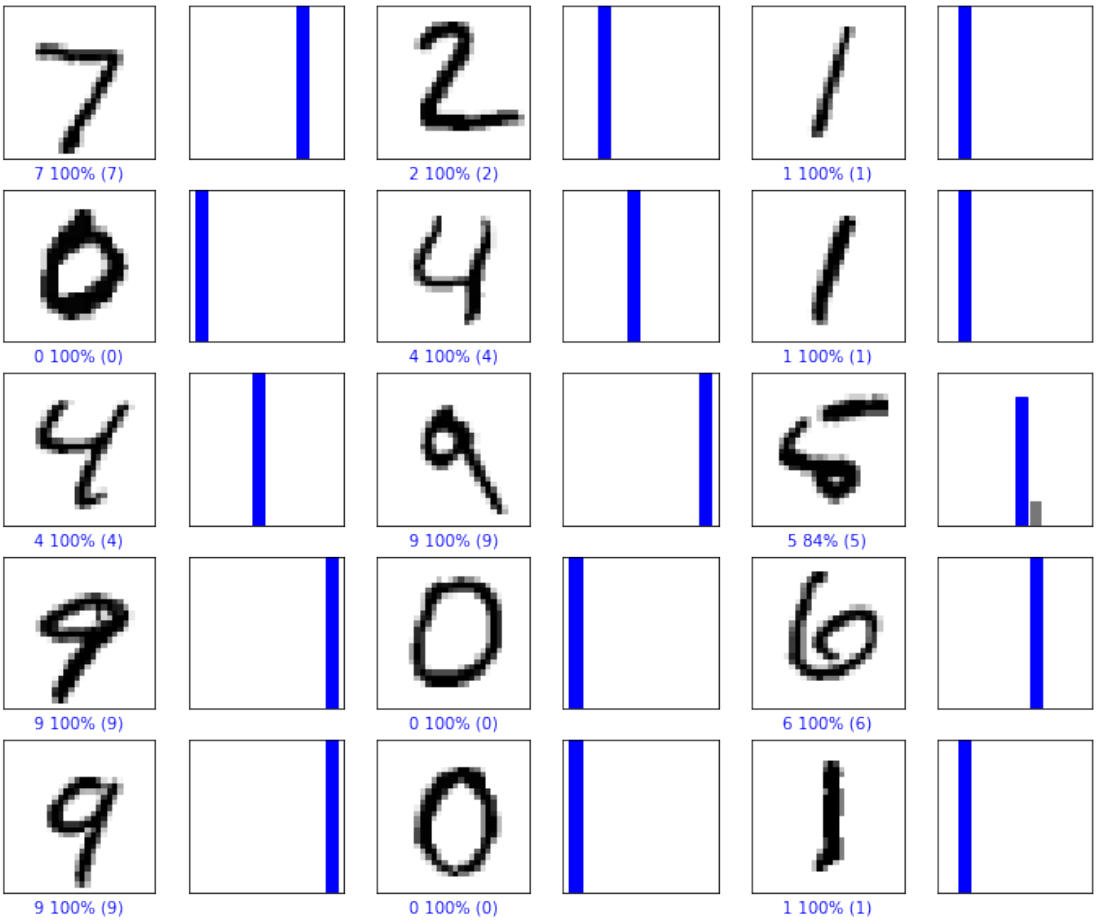


In [ ]:
```
i = 12
plt.figure(figsize=(6,3))
```

```
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 [ ]:
```
# 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 [ ]:
```
# Grab an image from the test dataset
img = test_images[0]

print(img.shape)
```

```
(28, 28)
```

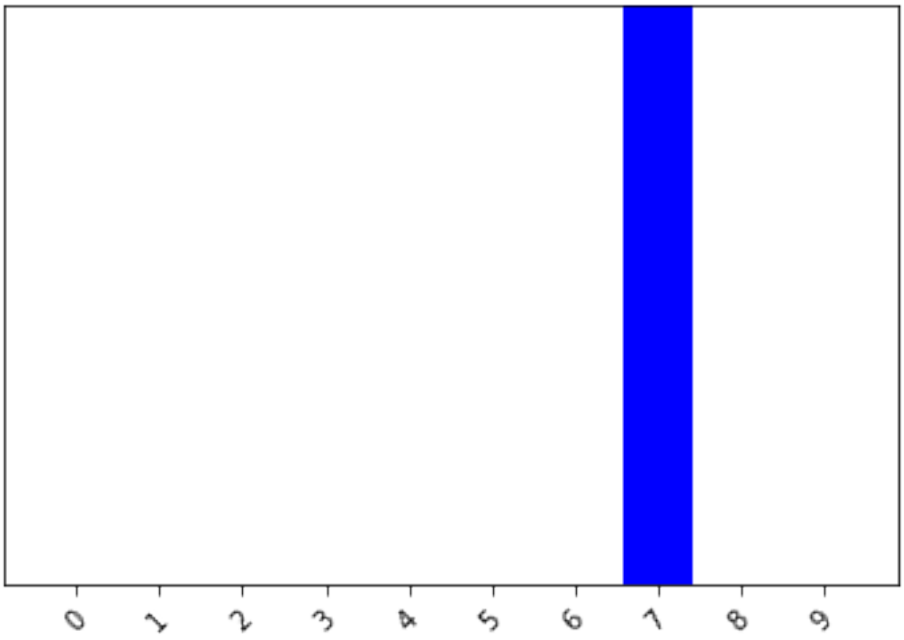
In [ ]:
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

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

In [ ]:
```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[2.81201884e-09 1.55601809e-10 1.80590985e-06 5.89029733e-05
  3.48718484e-14 2.49391707e-08 1.30110332e-11 9.99939203e-01
  1.16635164e-07 3.10925827e-08]]
```

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



In [ ]:
```
prediction_result = np.argmax(predictions_single[0])
print(prediction_result)
```

```
7
```