import library

In [1]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
```

Get MNIST Data.

MNIST data loacted in tensorflow > keras > datasets > mnist

Split data to (train images, train labels) and (test images, test labels)

In [2]:

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

There are Total 60000 Train images and Train labels. (6000 images for single class)

Shape of single image is 28 x 28 (pixel)

In [3]:

```
print('Shape of Train images :',train_images.shape)
print('Shape of Train labels : ', train_labels.shape)
print('₩nShape of Test images : ', test_images.shape)
print("Shape of Test labels : ",test_labels.shape)
```

```
Shape of Train images : (60000, 28, 28)
Shape of Train labels :  (60000,)

Shape of Test images :  (10000, 28, 28)
Shape of Test labels :  (10000,)
```

In [4]:

```
print('Train labels : ',train_labels)
```

```
Train labels :  [5 0 4 ... 5 6 8]
```

Plot first train image.

when value is close to 0 : dark

when value is close to 255 : white

In [5]:

```
print(train_images[1])
```

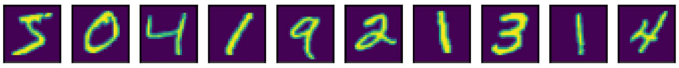
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  51 159 253
  159  50   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  48 238 252 252
  252 237   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  54 227 253 252 239
  233 252  57   6   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  10  60 224 252 253 252 202
   84 252 253 122   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 163 252 252 252 253 252 252
   96 189 253 167   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  51 238 253 253 190 114 253 228
   47  79 255 168   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  48 238 252 252 179  12  75 121  21
    0   0 253 243  50   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  38 165 253 233 208  84   0   0   0   0
    0   0 253 252 165   0   0   0   0   0]
 [  0   0   0   0   0   0   0   7 178 252 240  71  19  28   0   0   0   0
    0   0 253 252 195   0   0   0   0   0]
 [  0   0   0   0   0   0   0  57 252 252  63   0   0   0   0   0   0   0
    0   0 253 252 195   0   0   0   0   0]
 [  0   0   0   0   0   0   0 198 253 190   0   0   0   0   0   0   0   0
    0   0 255 253 196   0   0   0   0   0]
 [  0   0   0   0   0   0  76 246 252 112   0   0   0   0   0   0   0   0
    0   0 253 252 148   0   0   0   0   0]
 [  0   0   0   0   0   0  85 252 230  25   0   0   0   0   0   0   0   0
    7 135 253 186  12   0   0   0   0   0]
 [  0   0   0   0   0   0  85 252 223   0   0   0   0   0   0   0   0   7
  131 252 225  71   0   0   0   0   0   0]
 [  0   0   0   0   0   0  85 252 145   0   0   0   0   0   0   0  48 165
  252 173   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  86 253 225   0   0   0   0   0   0 114 238 253
  162   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  85 252 249 146  48  29  85 178 225 253 223 167
   56   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  85 252 252 252 229 215 252 252 252 196 130   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  28 199 252 252 253 252 252 233 145   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  25 128 252 253 252 141  37   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

Plot First 10 Train images and Corresponding labels

In [6]:

```
print('First 10 Train images in MNIST dataset₩n')
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(train_images[i])
plt.show()
print('₩nTrain labels match with Train label sequentialy₩n',train_labels[:10])
```

```
First 10 Train images in MNIST dataset
```



```
Train labels match with Train label sequentialy
 [5 0 4 1 9 2 1 3 1 4]
```

Important

Change data shape (60000 x 28 x 28) to (60000 x 28 x 28 x 1)

In [7]:

```
train_images = tf.reshape(train_images, [-1, 28, 28, 1])
test_images = tf.reshape(test_images, [-1, 28, 28, 1])
```

Select one convolution model below

There are 3 example models.

3, 5, 7 layer each

MODEL 1 : 3 Layers with 1 Convolution layer

MODEL 2 : 5 Layers with 2 Convolution layer

MODEL 3 : 7 Layers with 4 Convolution layer

In [8]:

```
def select_model(model_number):
    if model_number == 1:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),
                    keras.layers.MaxPool2D((2,2)),
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])

    if model_number == 2:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape=(28,28,1)),
                    keras.layers.MaxPool2D((2,2)),
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),
                    keras.layers.MaxPool2D((2,2)),
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])

    if model_number == 3:
        model = keras.models.Sequential([
                    keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape = (28, 28,1)),
                    keras.layers.MaxPool2D((2,2)),
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),
                    keras.layers.Conv2D(64, (3,3), activation = 'relu'),
                    keras.layers.MaxPool2D((2,2)),
                    keras.layers.Conv2D(128, (3,3), activation = 'relu'),
                    keras.layers.Flatten(),
                    keras.layers.Dense(10, activation = 'softmax')])

    return model
```

In [9]:

```
model = select_model(1)
```

If you want to see information of model, model.summary() will help

summary() is also built in function

In [10]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 flatten (Flatten)           (None, 5408)              0         
                                                                 
 dense (Dense)               (None, 10)                54090     
                                                                 
=================================================================
Total params: 54,410
Trainable params: 54,410
Non-trainable params: 0
_________________________________________________________________
```

Components in training step

Optimizer, Loss function, accuracy metrics

In [11]:

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

Training Step

Training for 5 epochs.

In [12]:

```
model.fit(train_images, train_labels,  epochs = 5)
```

```
Epoch 1/5
1875/1875 [==============================] - 18s 9ms/step - loss: 0.6037 - accuracy:
0.9406
Epoch 2/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0857 - accurac
y: 0.9752
Epoch 3/5
1875/1875 [==============================] - 20s 11ms/step - loss: 0.0685 - accurac
y: 0.9793
Epoch 4/5
1875/1875 [==============================] - 19s 10ms/step - loss: 0.0618 - accurac
y: 0.9815
Epoch 5/5
1875/1875 [==============================] - 19s 10ms/step - loss: 0.0489 - accurac
y: 0.9848
```

Out[12]:

```
<keras.callbacks.History at 0x1a5f0da7e20>
```

Test Step

Perform Test with Test data

In [13]:

```
test_loss, accuracy = model.evaluate(test_images, test_labels, verbose = 2)
print('₩nTest loss : ', test_loss)
print('Test accuracy :', accuracy)
```

```
313/313 - 1s - loss: 0.1184 - accuracy: 0.9744 - 1s/epoch - 3ms/step

Test loss :  0.11840185523033142
Test accuracy : 0.9743999838829041
```

Before prediction, change test image's type to float 32.

In [14]:

```
test_images = tf.cast(test_images, tf.float32)
pred = model.predict(test_images)
Number = [0,1,2,3,4,5,6,7,8,9]
```

```
313/313 [==============================] - 1s 3ms/step
```

In [15]:

```
print('Prediction : ', pred.shape)
print('Test labels : ', test_labels.shape)
```

```
Prediction :  (10000, 10)
Test labels :  (10000,)
```

Functions for plot images, probability

In [16]:

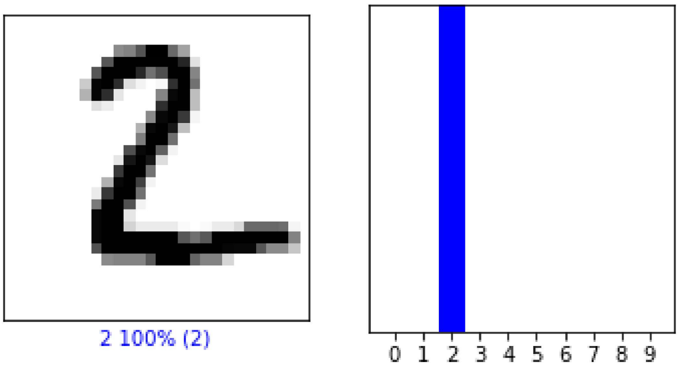
```
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  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(Number[predicted_label],
                                100*np.max(predictions_array),
                                Number[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)
  plt.xticks(Number)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

In [17]:

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

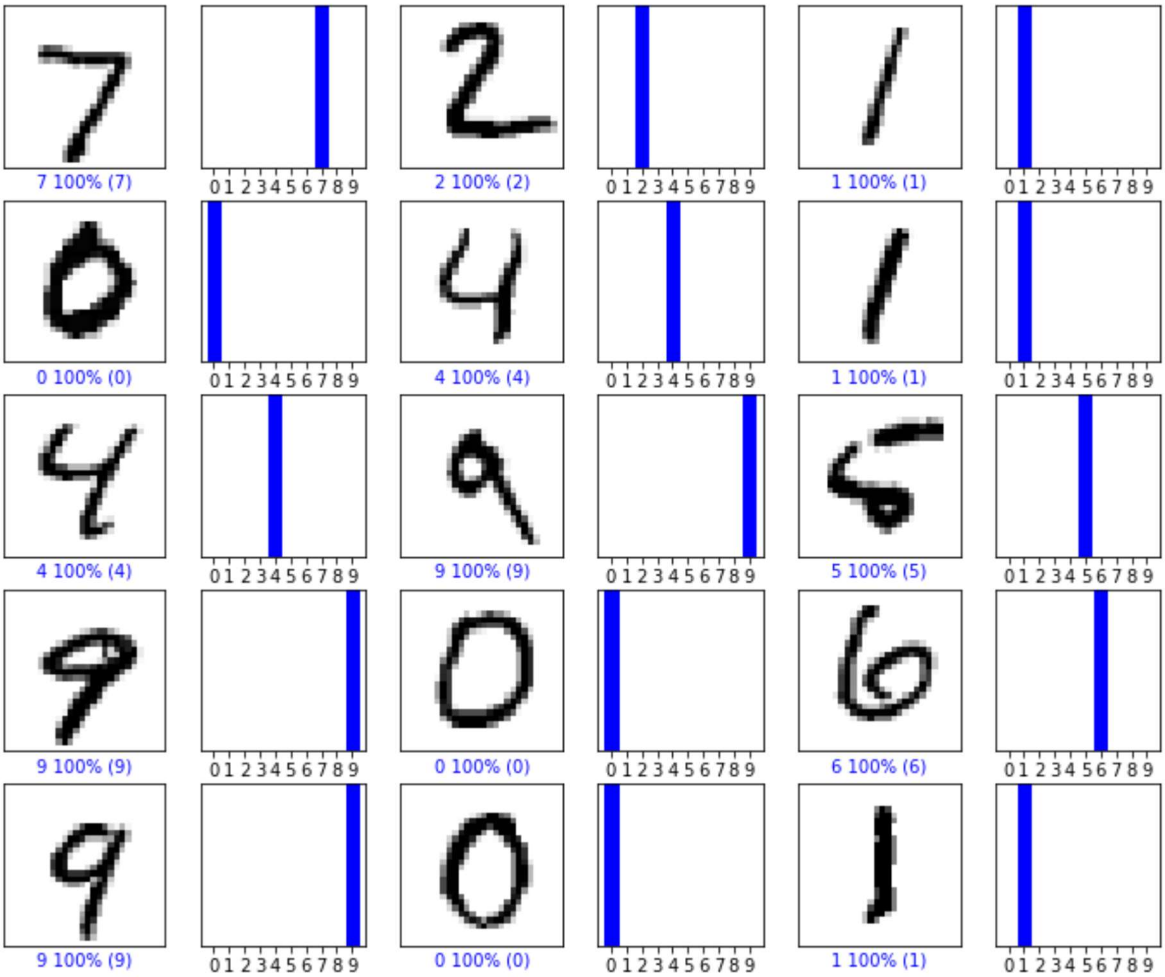
In [18]:

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



In [19]:

```
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, pred, test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, pred, test_labels)
plt.show()
```



Plot images and probability that model predicted wrong

In [20]:

```
def error_mnist(prediction_array, true_label):
    error_index = []

    for i in range(true_label.shape[0]):
        if np.argmax(prediction_array[i]) != true_label[i]:
            error_index.append(i)
    return error_index

# change num_cols, num_rows if you want to see more result.
def plot_error(index, prediction_array, true_label):
    num_cols = 5
    num_rows = 5
    plt.figure(figsize=(2*2*num_cols, 2*num_rows))

    assert len(index) < num_cols * num_rows
    for i in range(len(index)):
        plt.subplot(num_rows, 2*num_cols, 2*i+1)
        idx = index[i]
        plt.imshow(test_images[idx])
        plt.subplot(num_rows, 2*num_cols, 2*i+2)
        plt.bar(range(10), prediction_array[idx])
        plt.xticks(Number)
```

Find index of wrong prediction

Plot first 10 wrong predicted images and probability

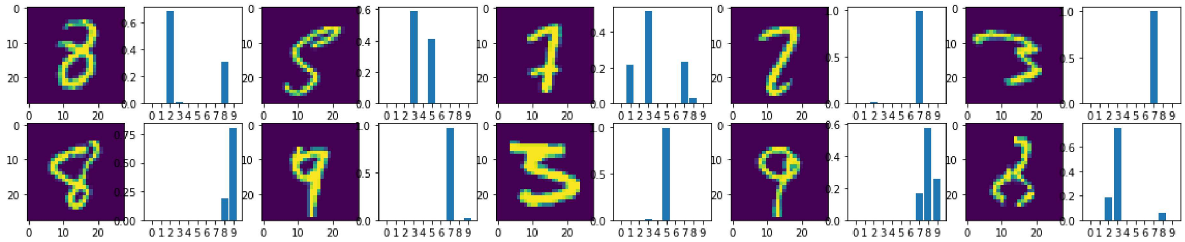
In [21]:

```
index = error_mnist(pred, test_labels)
index_slice = index[:10]
print(index[:10])
```

```
[184, 211, 282, 321, 381, 403, 417, 449, 488, 582]
```

In [22]:

```
plot_error(index_slice, pred, test_labels)
```



In [23]:

```
DONE
```

```
-------------------------------------------------------------------------------------
-
NameError                                  Traceback (most recent call last)
Input In [23], in <cell line: 1>()
----> 1 DONE

NameError: name 'DONE' is not defined
```

In [ ]:

```

```