아래 내용은 Intel® AI For Youth Program 내용을 참고하여

Brain AI와 Brain AI Coach Network에서 개발한 내용입니다. 상업적 사용은 불가하며,
학교에서 학생들 교육활동에 자유롭게 사용가능합니다.

Project Title: 첫 번째 신경망 훈련하기: 기초적인 분류 문제

다음 소스코드를 제공한 사이트를 이동하여 내용을 이해해가면 아래 소스 코드를 실행해 보세요.
https://www.tensorflow.org/tutorials/keras/classification (https://www.tensorflow.org/tutorials/keras/classification)

Step 1. 문제 찾기

이 튜토리얼에서는 운동화나 셔츠 같은 옷 이미지를 분류하는 신경망 모델을 훈련합니다. 상세 내용을 모두 이해하지 못해도 괜찮습니다. 여기서는 완전한 텐서플로(TensorFlow) 프로그램을 빠르게 살펴 보겠습니다. 자세한 내용은 앞으로 함께 공부합시다.

여기에서는 텐서플로 모델을 만들고 훈련할 수 있는 고수준 API인 tf.keras를 사용합니다.

- 읽을 거리 : https://www.intel.com/content/www/us/en/retail/solutions/ai-in-retail.html (https://www.intel.com/content/www/us/en/retail/solutions/ai-in-retail.html)

Step 2. 데이터 획득

In [2]:

```
# tensorflow와 tf.keras를 임포트합니다
import tensorflow as tf
from tensorflow import keras

# 헬퍼(helper) 라이브러리를 임포트합니다
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

2.3.0

In [3]:

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

In [4]:

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

Step 3. 데이터 탐색

In [5]:

```
1 train_images.shape
```

Out[5]:

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

In [6]:

```
1 len(train_labels)
```

Out[6]:

```
60000
```

In [7]:

```
1 train_labels
```

Out[7]:

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

In [8]:

```
1 test_images.shape
```

Out[8]:

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

In [9]:

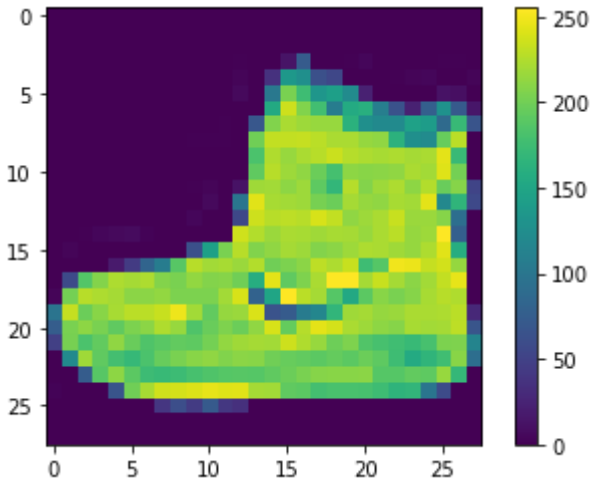
```
1 len(test_labels)
```

Out[9]:

```
10000
```

In [10]:

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



In [11]:

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

In [12]:

```
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()
```

Step 4. 모델링

In [13]:

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

In [14]:

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

In [15]:

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

```
Epoch 1/5
1875/1875 [==============================] - 2s 849us/step - loss: 0.5014 - accurac
y: 0.8244
Epoch 2/5
1875/1875 [==============================] - 2s 929us/step - loss: 0.3749 - accurac
y: 0.8652
Epoch 3/5
1875/1875 [==============================] - 2s 825us/step - loss: 0.3386 - accurac
y: 0.8770
Epoch 4/5
1875/1875 [==============================] - 2s 843us/step - loss: 0.3135 - accurac
y: 0.8849
Epoch 5/5
1875/1875 [==============================] - 2s 806us/step - loss: 0.2951 - accurac
y: 0.8913
```

Out[15]:

```
<tensorflow.python.keras.callbacks.History at 0x2734bdf5f08>
```

Step 5. 모델 평가

In [16]:

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

print('₩n테스트 정확도:', test_acc)
```

```
313/313 - 0s - loss: 0.3625 - accuracy: 0.8681

테스트 정확도: 0.8680999875068665
```

Step 6. 정리 및 배포

In [17]:

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

In [18]:

```
predictions[0]
```

Out[18]:

```
array([1.1074178e-06, 7.1769463e-10, 3.8757616e-08, 2.8852691e-08,
       2.3656273e-06, 1.5141871e-02, 1.1087450e-06, 2.3677288e-02,
       4.1676653e-06, 9.6117204e-01], dtype=float32)
```

In [19]:

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

Out[19]:

```
9
```

In [20]:

```
test_labels[0]
```

Out[20]:

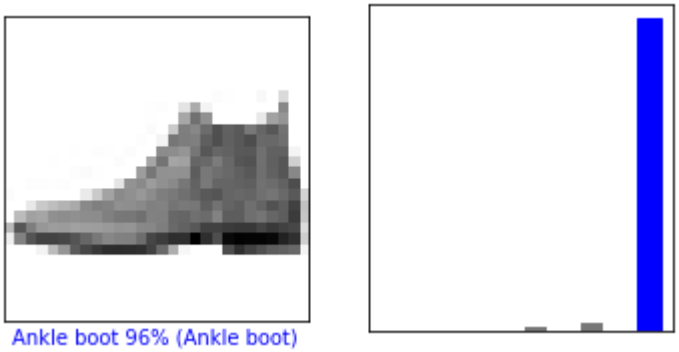
```
9
```

In [21]:

```
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')
```

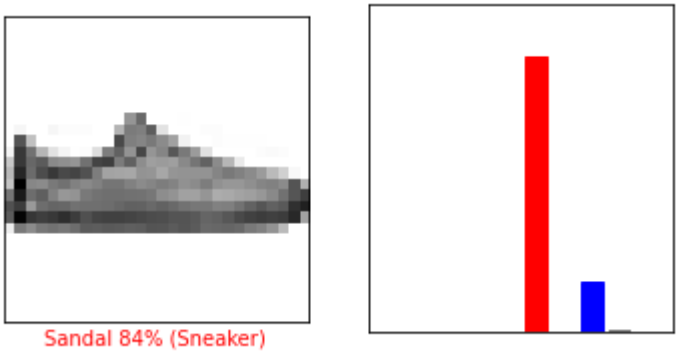
In [22]:

```
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()
```



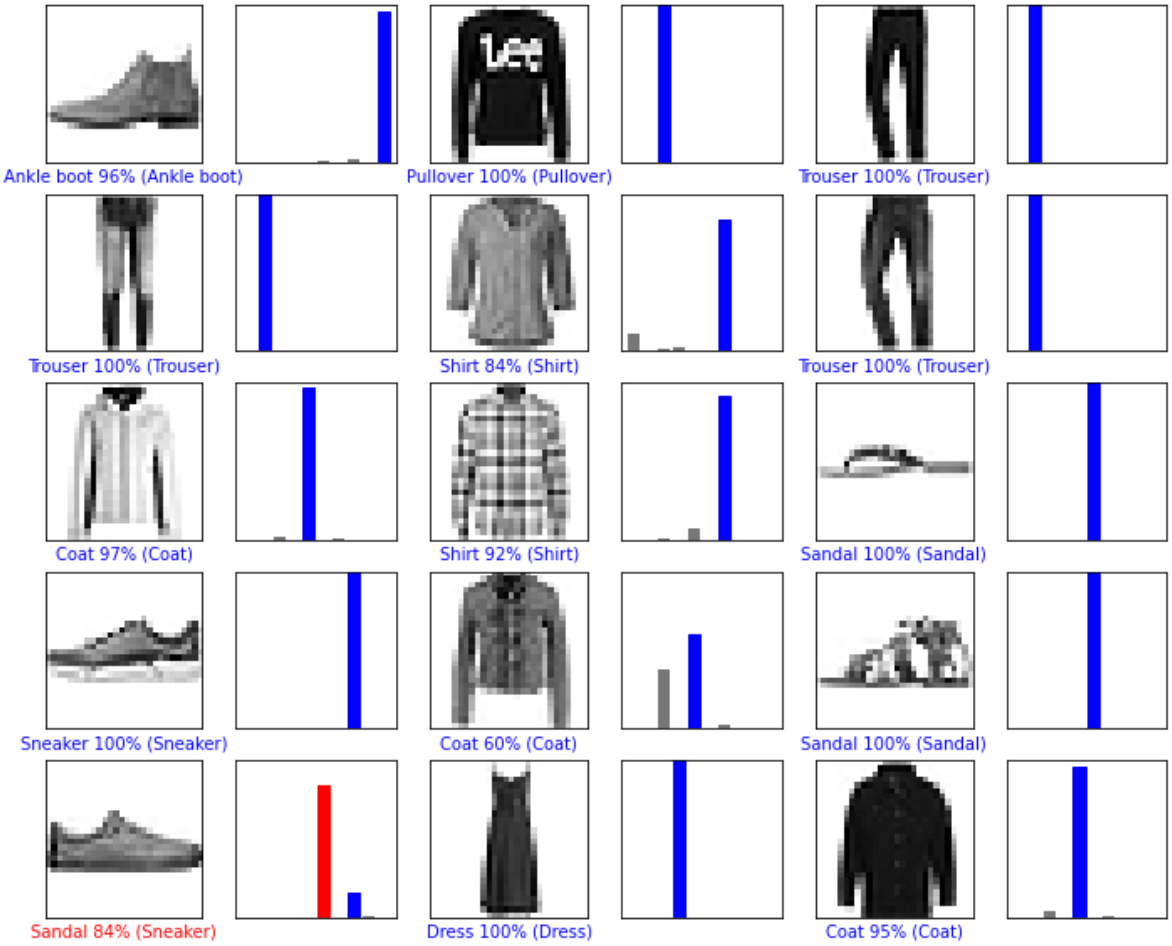
In [23]:

```
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 [30]:

```
# 처음 X 개의 테스트 이미지와 예측 레이블, 진짜 레이블을 출력합니다
# 올바른 예측은 파랑색으로 잘못된 예측은 빨강색으로 나타냅니다
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 [25]:

```
# 테스트 세트에서 이미지 하나를 선택합니다
img = test_images[0]

print(img.shape)
```

```
(28, 28)
```

In [26]:

```
# 이미지 하나만 사용할 때도 배치에 추가합니다
img = (np.expand_dims(img,0))

print(img.shape)
```

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

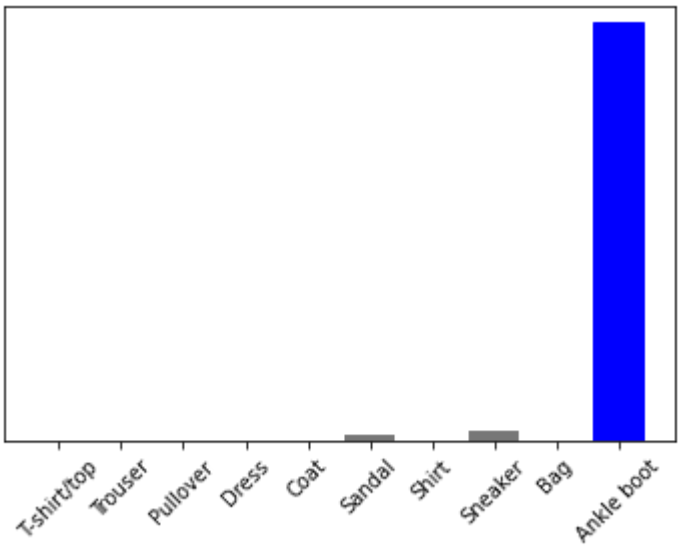
In [27]:

```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[1.1074198e-06 7.1769735e-10 3.8757690e-08 2.8852746e-08 2.3656319e-06
  1.5141877e-02 1.1087493e-06 2.3677299e-02 4.1676731e-06 9.6117204e-01]]
```

In [28]:

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



In [29]:

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

Out[29]:

```
9
```