TF2.0 신경망 만들기

- fashion2.0 데이터 셋을 이용한 신경망 만들기
- 개발 환경 : tf 버전 2.x (2020/12)

학습내용

- dd

In [1]:

```
import tensorflow as tf
```

In [2]:

```
print(tf.__version__)
```

```
2.4.0
```

In [3]:

```
# !pip install -q tensorflow-gpu==2.0.0-rc1
```

In [4]:

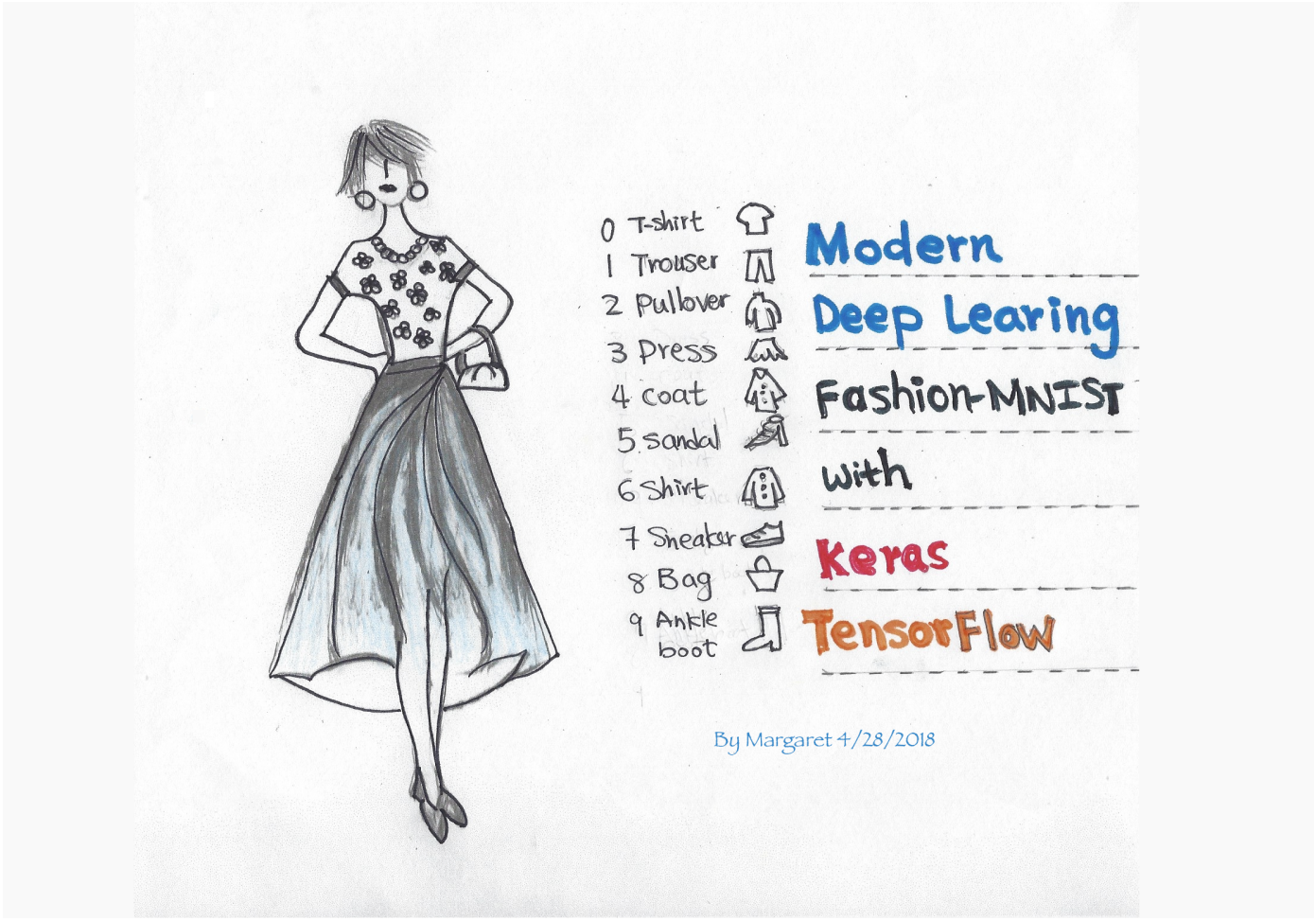
```
# tensorflow와 tf.keras를 임포트합니다
import tensorflow as tf
from tensorflow import keras

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

print(tf.__version__)
print(np.__version__)
```

```
2.4.0
1.19.4
```

Fashion MNIST DataSet



In [5]:

```
fashion_mnist = keras.datasets.fashion_mnist


# 4개의 데이터 셋 반환(numpy 배열)
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-labels-idx1-ubyte.gz)
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-images-idx3-ubyte.gz)
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-labels-idx1-ubyte.gz)
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-images-idx3-ubyte.gz)
4423680/4422102 [==============================] - 0s 0us/step
```

In [6]:

```
print("학습용 데이터 : x: {}, y:{}".format(train_images.shape, train_labels.shape) )
print("테스트 데이터 : x: {}, y:{}".format(test_images.shape, test_labels.shape) )
```

```
학습용 데이터 : x: (60000, 28, 28), y:(60000,)
테스트 데이터 : x: (10000, 28, 28), y:(10000,)
```

In [7]:

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

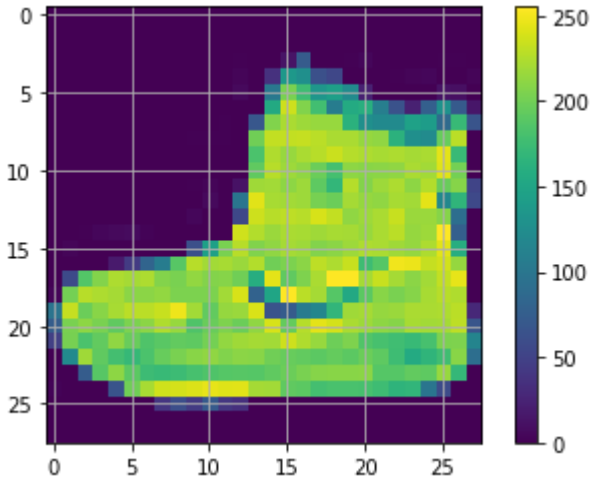
In [8]:

```
print("학습용 데이터의 레이블 ", np.unique(train_labels) )
```

```
학습용 데이터의 레이블  [0 1 2 3 4 5 6 7 8 9]
```

In [9]:

```
plt.figure()
plt.imshow(train_images[0])   # 첫번째 이미지 데이터
plt.colorbar()     # 색깔 표시바
plt.grid(True)   # grid 선
plt.show()
```



In [10]:

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

이미지 확인

In [11]:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)  # 그래프의 표시 위치
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)  # 그리드선
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

모델 생성

In [12]:

```
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'])
```

In [13]:

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

```
Epoch 1/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.6246 - accuracy:
0.7832
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3859 - accuracy:
0.8607
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3396 - accuracy:
0.8778
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3088 - accuracy:
0.8866
Epoch 5/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2950 - accuracy:
0.8920
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2742 - accuracy:
0.8994
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2701 - accuracy:
0.9000
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2501 - accuracy:
0.9071
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2476 - accuracy:
0.9070
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2379 - accuracy:
0.9120
```

Out[13]:

```
<tensorflow.python.keras.callbacks.History at 0x7f1500708860>
```

In [14]:

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

print('Wn테스트 정확도:', test_acc)
```

```
313/313 - 0s - loss: 0.3319 - accuracy: 0.8838

테스트 정확도: 0.8838000297546387
```

예측하기

- 훈련된 모델을 사용하여 이미지에 대한 예측 해보기
- 테스트 세트에 대한 각 이미지의 레이블을 예측. 10개의 숫자배열로 나타난다.

In [15]:

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

In [16]:

```
predictions[0]
```

Out[16]:

```
array([3.6626059e-06, 4.0913672e-10, 1.2223150e-08, 1.6621777e-09,
       2.3146008e-08, 3.2466167e-04, 1.0686205e-07, 2.2366486e-02,
       1.4484547e-07, 9.7730494e-01], dtype=float32)
```

In [17]:

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

Out[17]:

```
9
```

In [18]:

```
test_labels[0]
```

Out[18]:

```
9
```

In [19]:

```
### 10개의 데이터에 대해 확인
np.argmax(predictions, axis=1)[0:10]
```

Out[19]:

```
array([9, 2, 1, 1, 6, 1, 4, 6, 5, 7])
```

In [20]:

```
### 10개의 데이터에 대한 실제값
test_labels[0:10]
```

Out[20]:

```
array([9, 2, 1, 1, 6, 1, 4, 6, 5, 7], dtype=uint8)
```

이미지 데이터 시각화

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)  # 이미지 표시

  # 정확하게 맞춰을 경우, blue(파란), 아니면 red(적색)으로 표시
  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)
```

막대 그래프로 표시

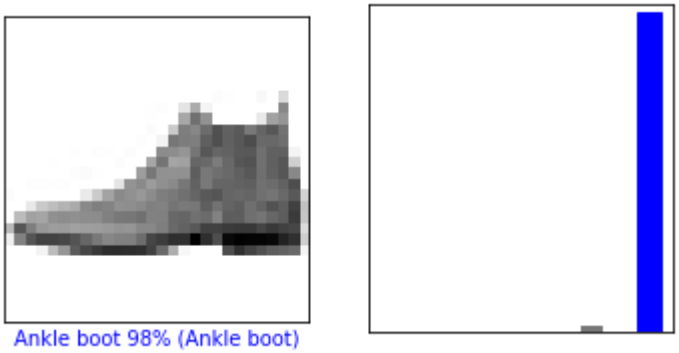
In [22]:

```
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')
```

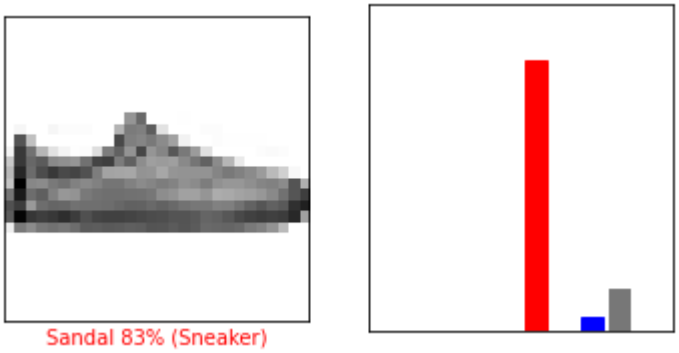
In [23]:

```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)  # 이미지 표시
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)        # 막대 그래프 표시
plt.show()
```



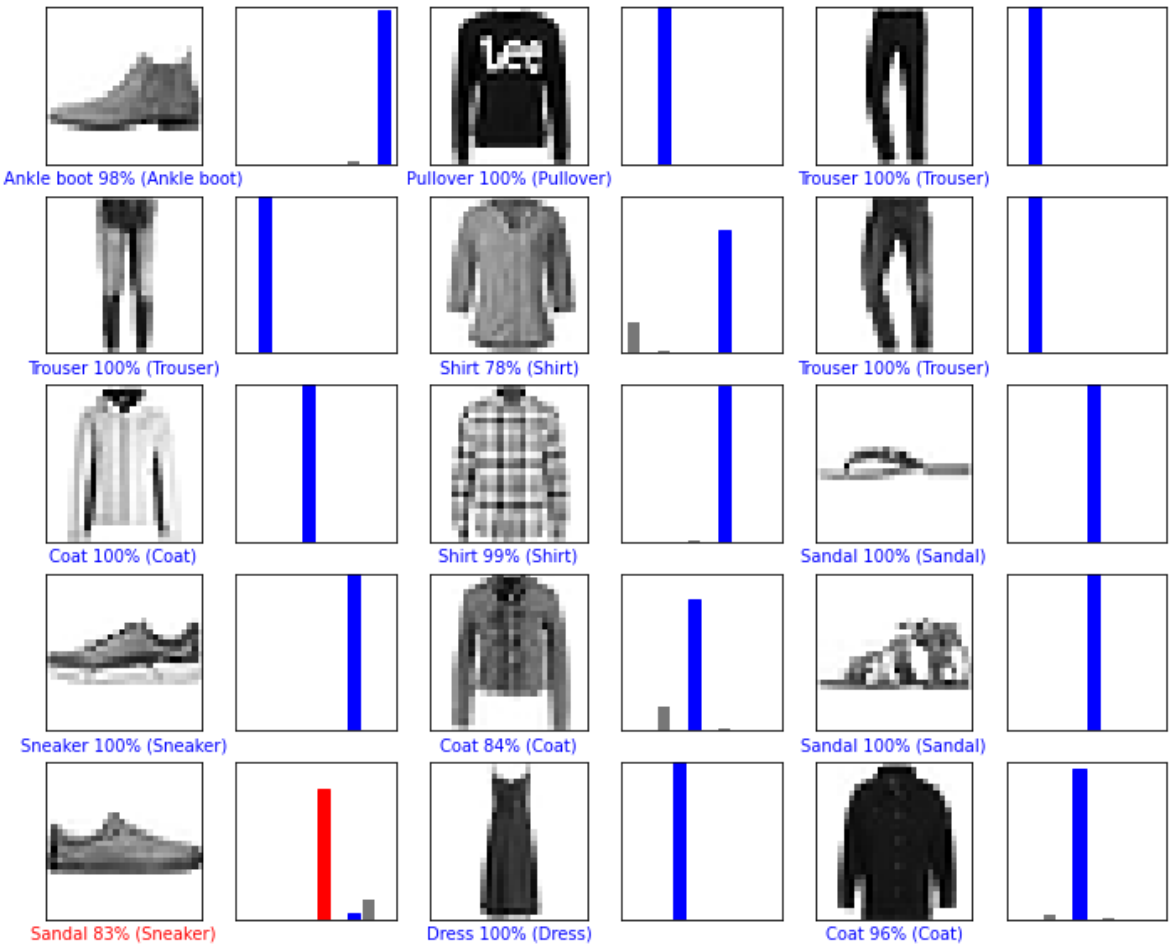
In [24]:

```
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 [25]:

```
# 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()
```



이미지 하나 예측해 보기 예측

In [26]:

```
# 테스트 세트에서 이미지 하나를 선택합니다
img = test_images[0]
print(img.shape)

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

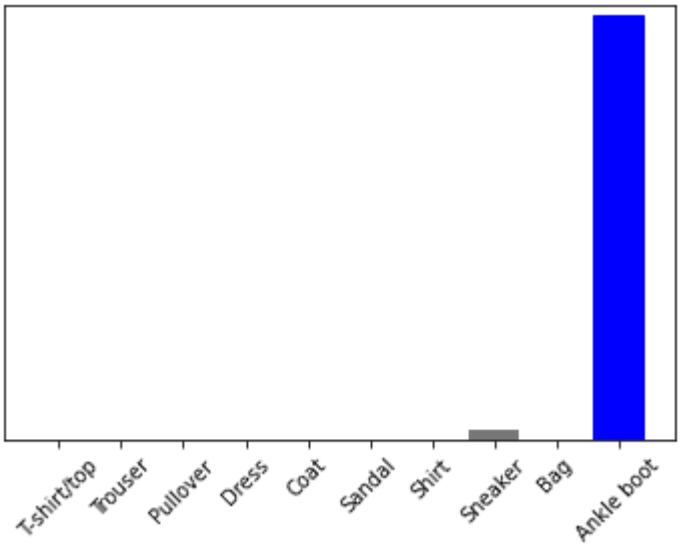
```
(28, 28)
(1, 28, 28)
```

In [27]:

```
predictions_single = model.predict(img)
print(predictions_single)

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

```
[[3.6626132e-06 4.0913672e-10 1.2223173e-08 1.6621777e-09 2.3146008e-08
  3.2466228e-04 1.0686205e-07 2.2366498e-02 1.4484547e-07 9.7730494e-01]]
```



In [28]:

```
idx = np.argmax(predictions_single[0])
print(idx)
print(class_names[idx])
```

```
9
Ankle boot
```

```
    #@title MIT License
    #
    # Copyright (c) 2017 François Chollet
    #
    # Permission is hereby granted, free of charge, to any person obtaining a
```

```
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.
```

REF

- fashion 2.0 TF : https://www.tensorflow.org/tutorials/keras/classification (https://www.tensorflow.org/tutorials/keras/classification)

History

- 2020/12/28 tf 2.x (ver 1.1)