

텐서플로우 이미지 분류

(<https://www.tensorflow.org/tutorials/keras/classification>, <https://www.youtube.com/watch?v=uTqgzJVKeKI>)

. 필요한 lib import

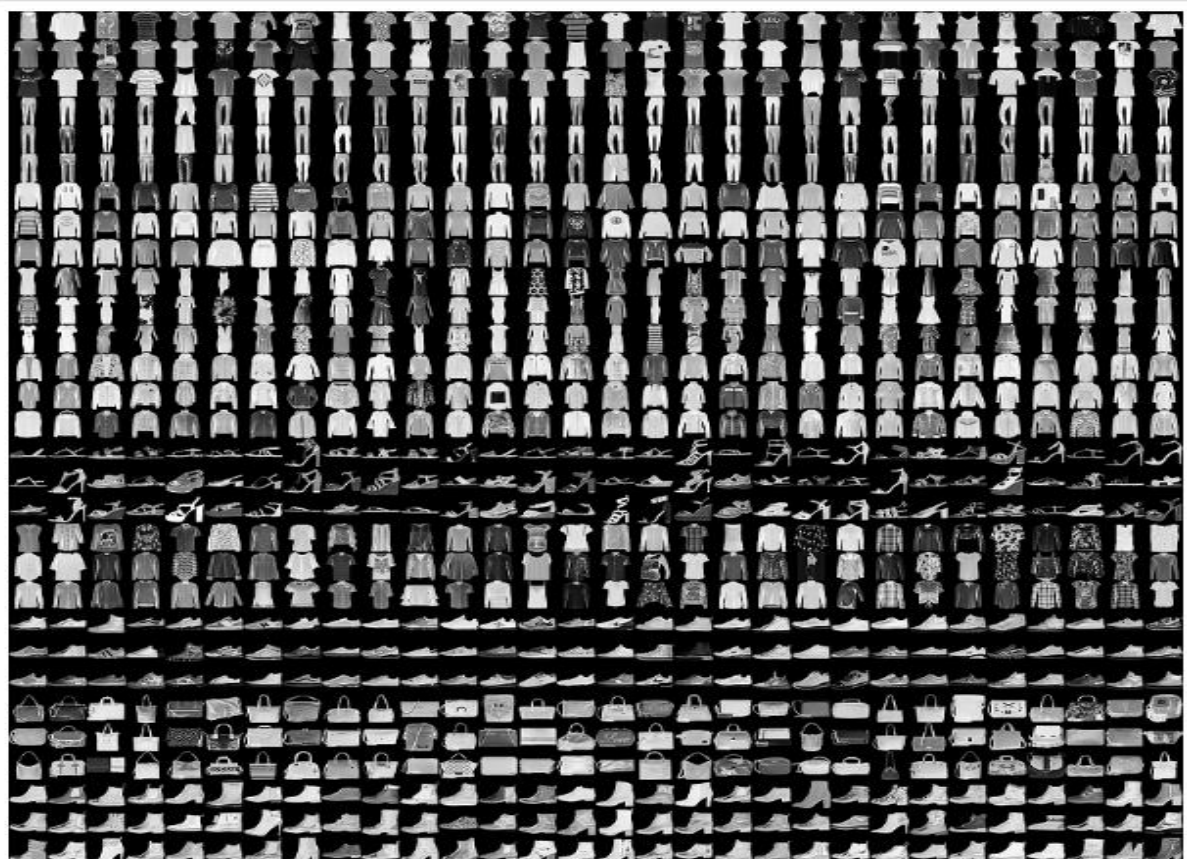
```
# TensorFlow and tf.keras
import tensorflow as tf

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

print(tf.__version__)
```

```
[1] 1 # TensorFlow and tf.keras
    2 import tensorflow as tf
    3
    4 # Helper libraries
    5 import numpy as np
    6 import matplotlib.pyplot as plt
    7
    8 print(tf.__version__)
2.9.2
```

패션 MNIST 데이터셋 임포트하기 (해상도 28x28, 개별 옷 품목, 60000개 이미지로 학습, 10000개로 학습 평가)




```

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

```



```

1 fashion_mnist = tf.keras.datasets.fashion_mnist
2
3 (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [=====] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [=====] - 0s 0us/step

```

(train_***는 학습에, test_***는 테스트에 사용. 이미지는 28x28 numpy 배열이고, 픽셀은 0~255 값, label은 0~9이며, 옷의 클래스)

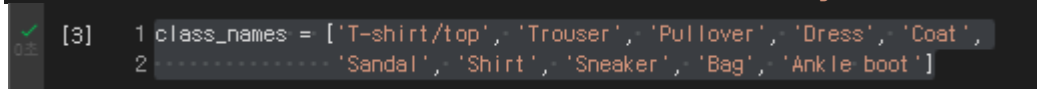
레이블	클래스
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

. class 이름 정해 주기

```

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

```



```

[3] 1 class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
2     'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

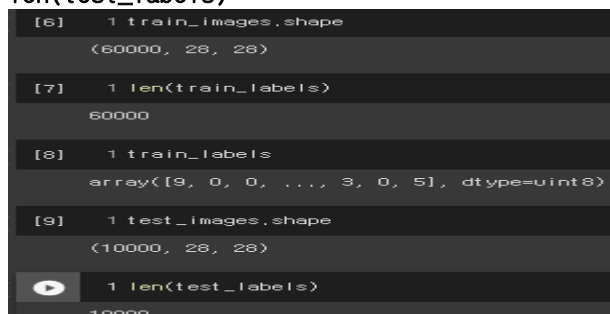
```

. 데이터 탐색

```

train_images.shape # 60000 개
len(train_labels)
train_labels # 60000 개 이미지의 0~9 까지의 라벨
test_images.shape # 10000 개
len(test_labels)

```



```

[6] 1 train_images.shape
    (60000, 28, 28)

[7] 1 len(train_labels)
    60000

[8] 1 train_labels
    array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)

[9] 1 test_images.shape
    (10000, 28, 28)

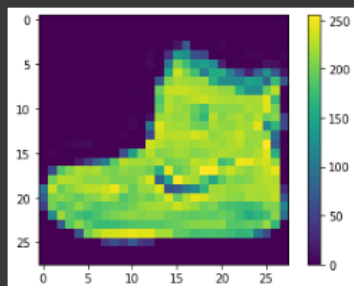
1 len(test_labels)
    10000

```

. 데이터 전처리: 픽셀값 정규화

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

```
[12]: 1 plt.figure()
      2 plt.imshow(train_images[0])
      3 plt.colorbar()
      4 plt.grid(False)
      5 plt.show()
```



```
train_images = train_images / 255.0 # 정규화
```

```
test_images = test_images / 255.0 # 정규화
```

```
[13]: 1 train_images = train_images / 255.0
      2
      3 test_images = test_images / 255.0
```

```
plt.figure(figsize=(10,10)) # 그림 크기
for i in range(25): # 25개 (0~24)를 대상으로
    plt.subplot(5,5,i+1) # 5행 5열로 배열. 첫번째는 1 (0+1=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()
```

```
[14]: 1 plt.figure(figsize=(10,10))
      2 for i in range(25):
      3     plt.subplot(5,5,i+1)
      4     plt.xticks([])
      5     plt.yticks([])
      6     plt.grid(False)
      7     plt.imshow(train_images[i], cmap=plt.cm.binary)
      8     plt.xlabel(class_names[train_labels[i]])
      9 plt.show()
```



. 모델 구성 (모델의 층(레이어) 구성한 후 컴파일. 층들간의 가중치는 훈련 중 학습)

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)), #입력층.
    28x28=784 픽셀의 1 차원 배열로 변환
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10) # 10 개의 확률을 반환
])
```

```
1 model = tf.keras.Sequential([
2     tf.keras.layers.Flatten(input_shape=(28, 28)),
3     tf.keras.layers.Dense(128, activation='relu'),
4     tf.keras.layers.Dense(10)
5 ])
```

. 모델 컴파일 (손실함수: 정확도 평가(최소화 필요), 옵티마이저(모델 업데이트), 메트릭(모니터링))

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
[28] 1 model.compile(optimizer='adam',
2 ..... loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
3 ..... metrics=['accuracy'])
```

. 모델 훈련

```
model.fit(train_images, train_labels, epochs=10) # 훈련 이미지 & 라벨 입력
```

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

Epoch 1/10
1875/1875 [=====] - 7s 2ms/step - loss: 0.5042 - accuracy: 0.8238
Epoch 2/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.3823 - accuracy: 0.8628
Epoch 3/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.3413 - accuracy: 0.8741
Epoch 4/10
1875/1875 [=====] - 4s 2ms/step - loss: 0.3164 - accuracy: 0.8846
Epoch 5/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.2990 - accuracy: 0.8895
Epoch 6/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.2842 - accuracy: 0.8942
Epoch 7/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.2714 - accuracy: 0.8983
Epoch 8/10
1875/1875 [=====] - 5s 3ms/step - loss: 0.2598 - accuracy: 0.9034
Epoch 9/10
1875/1875 [=====] - 5s 3ms/step - loss: 0.2512 - accuracy: 0.9060
Epoch 10/10
1875/1875 [=====] - 5s 2ms/step - loss: 0.2427 - accuracy: 0.9093
<keras.callbacks.History at 0x7f13f663edd0>
```

. 정확도 평가

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

```
[30] 1 test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
      2
      3 print('\nTest accuracy:', test_acc)
```

313/313 - 1s - loss: 0.3343 - accuracy: 0.8831 - 739ms/epoch - 2ms/step

Test accuracy: 0.8830999732017517

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

```
[31] 1 probability_model = tf.keras.Sequential([model,
2 ..... tf.keras.layers.Softmax()])
```

```
[32] 1 predictions = probability_model.predict(test_images)
```

```
313/313 [=====] - 1s 2ms/step
```

```
predictions[0]
[33] 1 predictions[0]
      2
      array([9.7829243e-06, 7.4694366e-09, 3.4943383e-08, 1.7650848e-08,
              2.0011221e-08, 5.2157193e-03, 8.0395657e-06, 1.3987904e-02,
              5.9899509e-07, 9.8077786e-01], dtype=float32)
```

10개 클래스에 대한 예측을 그래프로 표현

[illegible]

```

                                color=color)

def plot_value_array(i, predictions_array, true_label):
    true_label = 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')

```

```

3초
▶ 1 def plot_image(i, predictions_array, true_label, img):
2   true_label, img = true_label[i], img[i]
3   plt.grid(False)
4   plt.xticks([])
5   plt.yticks([])
6
7   plt.imshow(img, cmap=plt.cm.binary)
8
9   predicted_label = np.argmax(predictions_array)
10  if predicted_label == true_label:
11      color = 'blue'
12  else:
13      color = 'red'
14
15  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
16  .....:                             100*np.max(predictions_array),
17  .....:                             class_names[true_label]),
18  .....:                             color=color)
19
20 def plot_value_array(i, predictions_array, true_label):
21  true_label = true_label[i]
22  plt.grid(False)
23  plt.xticks(range(10))
24  plt.yticks([])
25  thisplot = plt.bar(range(10), predictions_array, color="#777777")
26  plt.ylim([0, 1])
27  predicted_label = np.argmax(predictions_array)
28
29  thisplot[predicted_label].set_color('red')
30  thisplot[true_label].set_color('blue')

```

. 예측 확인 (파란색이 올바르게 예측. 적색은 오류)

```

i = 12 # 13 번째 테스트 데이터
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()

```

✓
0.2

```
[36] 1 i = 12  
2 plt.figure(figsize=(6,3))  
3 plt.subplot(1,2,1)  
4 plot_image(i, predictions[i], test_labels, test_images)  
5 plt.subplot(1,2,2)  
6 plot_value_array(i, predictions[i], test_labels)  
7 plt.show()
```

