하이테크 단기과정(2022)

케라스 기초

김찬수

진행 순서

1. 케라스(Keras)

2. 딥러닝 예제 : Fashion-MNIST

케라스(Keras) : 모델 생성 방법

Sequential model

- 단일 입력, 단일 출력에 적합
- 레이어를 쌓는 방식으로 모델 표현 (레이어 리스트)

Functional API

- 다중 입출력 가능 (추천 방식)
- 임의의 모델 구조 표현 가능

Model subclassing

- 모델을 직접 구현
- 복잡도 높거나, 최신 연구 등에 추천

케라스(Keras) : 모델 생성 방법

Sequential model

```
# 3개 레이어를 가진 sequential model 선언
model1 = keras.Sequential(
    [
        layers.Dense(2, activation="relu", name="layer1"),
        layers.Dense(3, activation="relu", name="layer2"),
        layers.Dense(4, name="layer3"),
    ],
    name = "test1"
)
```

```
# 레이어를 순차적으로 작성
model2 = keras.Sequential(name="test2")
model2.add(layers.Dense(2, activation="relu", name="layer1"))
model2.add(layers.Dense(2, activation="relu", name="layer2"))
model2.add(layers.Dense(2, activation="relu", name="layer3"))
```

케라스(Keras) : 모델 생성 방법

🎯 Sequential model

```
layers.Dense(2,
             activation="relu",
             name="layer1")
```

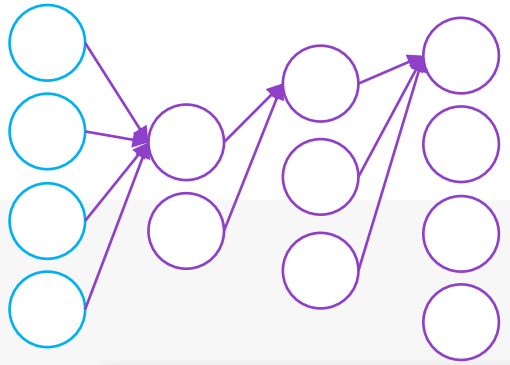
노드 개수

활성화 함수 종류

```
tf.keras.layers.Dense(
    units,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

케라스(Keras) : 모델 생성 방법

Sequential model



```
[32]  1 # 가중치는 입력이 투입되었을때 결정됨
      2 x = tf.ones((1,4))
      3 y = model1(x)
```

```
1 model1.weights
```

```
[<tf.Variable 'layer1/kernel:0' shape=(4, 2) dtype=float32, numpy=
 array([[-0.52915287,  0.22660589],
        [ 0.8745594 ,  0.80093503],
        [-0.7902229 ,  0.24711752],
        [ 0.3313632 ,  0.3962903 ]], dtype=float32)>,
 <tf.Variable 'layer1/bias:0' shape=(2,) dtype=float32, numpy=array([0., 0.], dtype=float32)>,
 <tf.Variable 'layer2/kernel:0' shape=(2, 3) dtype=float32, numpy=
 array([[-0.03583682,  0.57095385,  0.71487474],
        [-1.0009463 ,  0.9685893 , -0.5376901 ]], dtype=float32)>,
 <tf.Variable 'layer2/bias:0' shape=(3,) dtype=float32, numpy=array([0., 0., 0.], dtype=float32)>,
 <tf.Variable 'layer3/kernel:0' shape=(3, 4) dtype=float32, numpy=
 array([[ 0.7985482 ,  0.8817365 ,  0.8037225 , -0.89394474],
        [ 0.81184876,  0.51987123,  0.5344151 ,  0.6427982 ],
        [ 0.27057898, -0.18896312, -0.32519627, -0.7200339 ]],
       dtype=float32)>,
 <tf.Variable 'layer3/bias:0' shape=(4,) dtype=float32, numpy=array([0., 0., 0., 0.], dtype=float32)>]
```

케라스(Keras) : 모델 생성 방법

Sequential model

```
[37]  1 model1.summary()

Model: "test1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (1, 2)                    10
_________________________________________________________________
layer2 (Dense)               (1, 3)                    9
_________________________________________________________________
layer3 (Dense)               (1, 4)                    16
=================================================================
Total params: 35
Trainable params: 35
Non-trainable params: 0
_________________________________________________________________
```

케라스(Keras) : 모델 생성 방법

Functional API

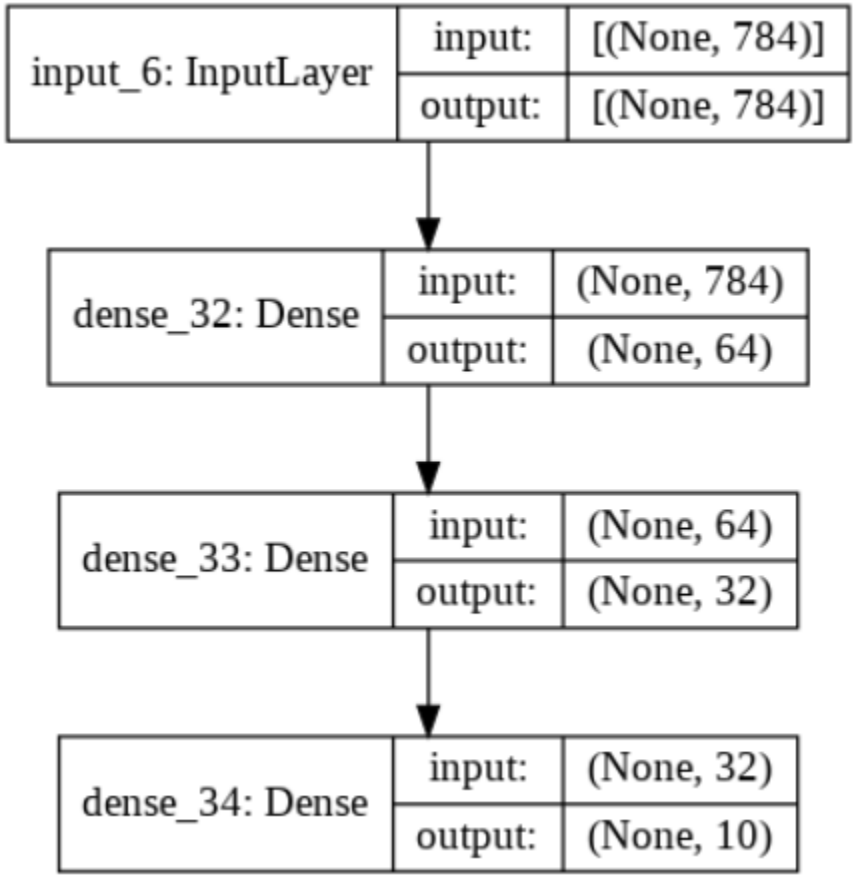
```
# 3개 레이어를 가진 sequential model 선언
dense = layers.Dense(64, activation="relu")
x1 = dense(inputs)
x2 = layers.Dense(32, activation = "relu")(x1)
outputs = layers.Dense(10)(x2)
model = keras.Model(inputs=inputs, outputs=outputs, name="test")

# 모델 확인
model.summary()
keras.utils.plot_model(model, "test_model.png", show_shapes=True)
```

케라스(Keras) : 모델 생성 방법

🎯 Functional API

```
Model: "test"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 784)]             0
_________________________________________________________________
dense_32 (Dense)             (None, 64)                50240
_________________________________________________________________
dense_33 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_34 (Dense)             (None, 10)                330
=================================================================
Total params: 52,650
Trainable params: 52,650
Non-trainable params: 0
```

model.summary()



keras.utils.plot_model()

케라스(Keras) : 모델 생성 방법

🎯 Model subclassing

```
# 퍼셉트론 예
class Linear(keras.layers.Layer):
    def __init__(self, units=32, input_dim=32):
        super(Linear, self).__init__()
        w_init = tf.random_normal_initializer()
        self.w = tf.Variable(
            initial_value=w_init(shape=(input_dim, units), dtype="float32"),
            trainable=True,
        )
        b_init = tf.zeros_initializer()
        self.b = tf.Variable(
            initial_value=b_init(shape=(units,), dtype="float32"), trainable=True
        )

    def call(self, inputs):
        return tf.matmul(inputs, self.w) + self.b
```

케라스(Keras) : 학습 설정 및 진행

```
Model.compile(
    optimizer="rmsprop",
    loss=None,
    metrics=None,
    loss_weights=None,
    weighted_metrics=None,
    run_eagerly=None,
    steps_per_execution=None,
    **kwargs
)
```

- optimizer : 최적화 방식 지정
- loss : 손실함수, 최적화하고자 하는 대상
- metrics : 모델 평가 기준 (accuracy)
- loss_weights : loss에 대한 가중치
- weighted_metrics : 모델 평가시의 가중치
- run_eagerly : tf.function 내부에서 모델을 실행할지 여부
- steps_per_execution : tf.function 실행 중 실행할 배치의 수

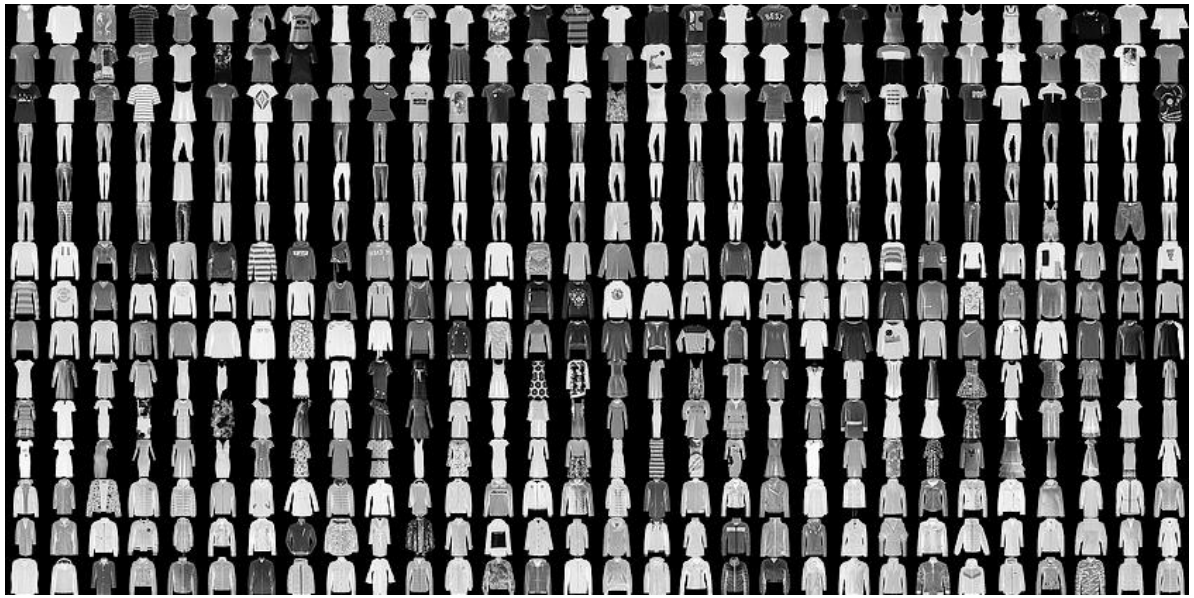
케라스(Keras) : 학습 설정 및 진행

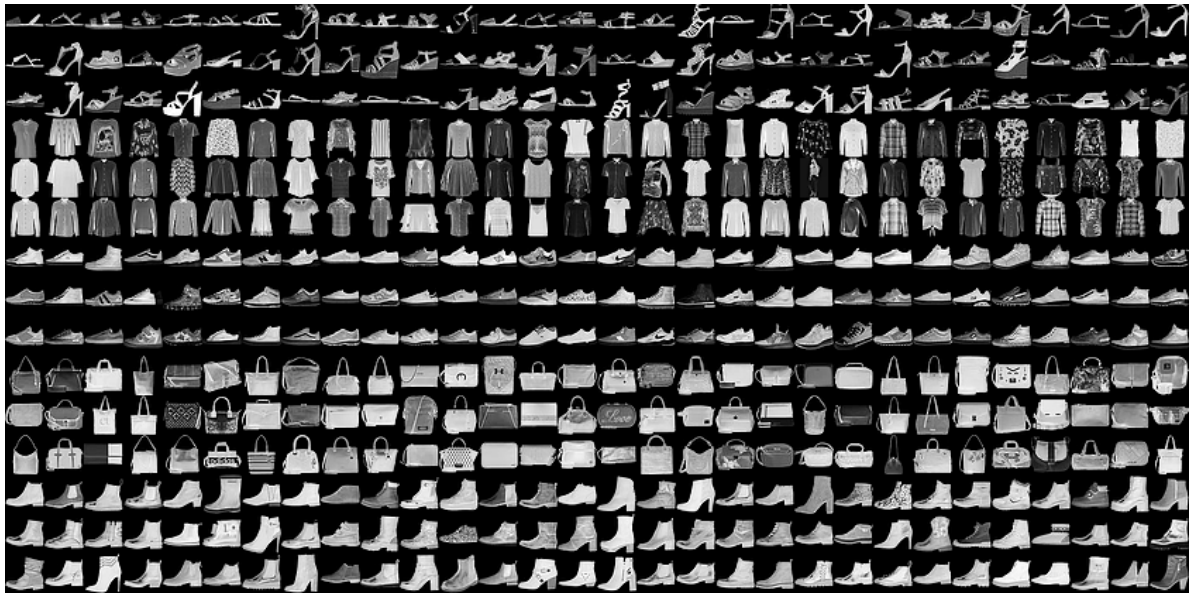
```
Model.fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose="auto",
    callbacks=None,
    validation_split=0.0,
    validation_data=None,
    shuffle=True,
    class_weight=None,
    sample_weight=None,
    initial_epoch=0,
    steps_per_epoch=None,
    validation_steps=None,
    validation_batch_size=None,
    validation_freq=1,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
)
```

- x : 입력데이터
- y : 입력데이터에 대한 정답데이터
- batch_size : 가중치 업데이트 위해 사용하는 샘플 수
- epochs : x, y를 사용하는 반복 횟수
- verbose : 진행 상태 표시 옵션
- callback : 인스턴스 목록
- validation_split : 평가 데이터 비율
- validation_data : 평가 데이터
- shuffle : 각 에포트마다 훈련 데이터를 섞을지 여부
- class_weight : 각 클래스의 손실함수에 대한 가중치
- sample_weight : 각 샘플의 손실함수에 대한 가중치
- initial_epoch : 학습을 시작할 epoch
- steps_per_epoch : epoch 간격
- …

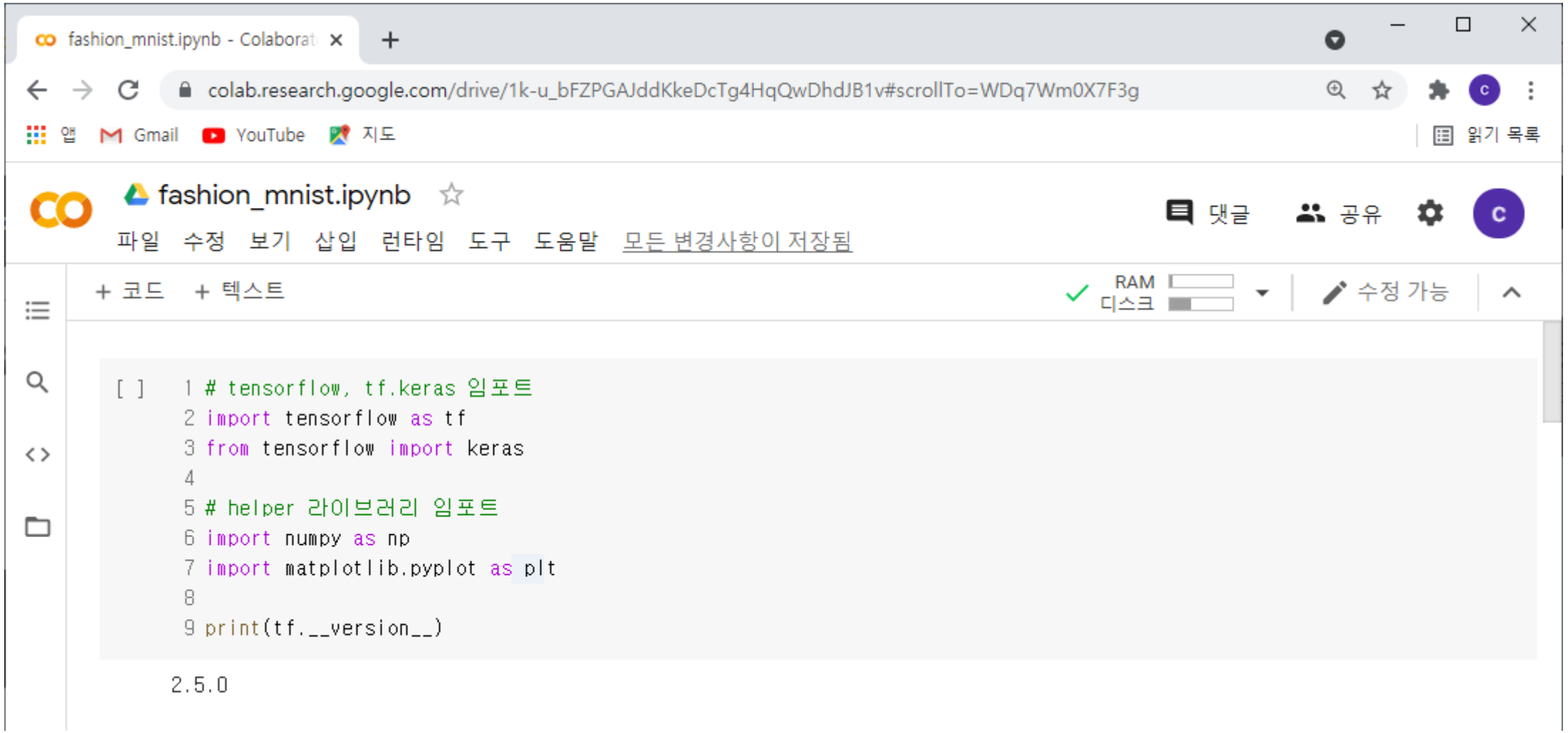
Colab 실습 : 이미지 분류

- 학습 DB : Fashion-MNIST
  (10개 범주, 70,000개 흑백 이미지 (28×28)
- 범주 :
  T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankel boot

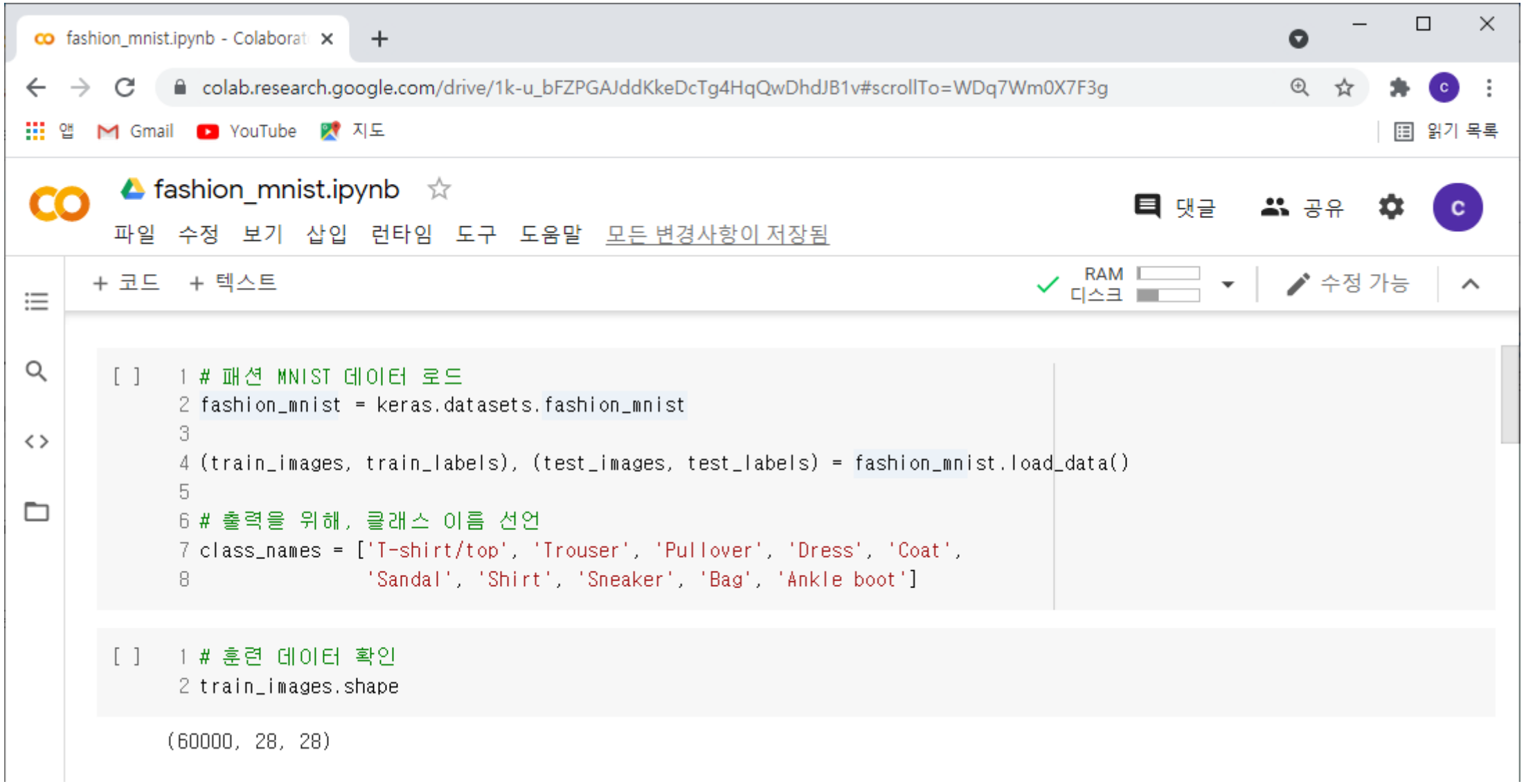




Colab 실습 : 이미지 분류

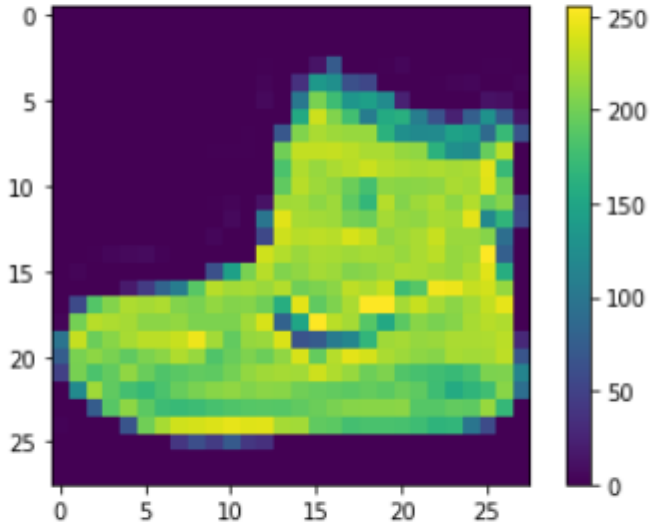


Colab 실습 : 이미지 분류



Colab 실습 : 이미지 분류

```
[ ]  1 # 데이터 이미지 확인
     2 plt.figure()
     3 plt.imshow(train_images[0])
     4 plt.colorbar()
     5 plt.grid(False)
     6 plt.show()
```




```
[ ]  1 # 데이터 전처리
     2 train_images = train_images / 255.0
     3
     4 test_images = test_images / 255.0
```

Colab 실습 : 이미지 분류

```
[ ]  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()
```

Colab 실습 : 이미지 분류

```
[ ]  1 # 모델 설계
     2 model = keras.Sequential([
     3     keras.layers.Flatten(input_shape=(28, 28)),
     4     keras.layers.Dense(64, activation='relu'),
     5     keras.layers.Dense(10, activation='softmax')
     6 ])
     7 # keras.layers.Flatten : 픽셀의 1차원 배열 변환
     8 # keras.layers.Dense : densely-connected/fully-connected 노드(뉴런)
```

```
[ ]  1 # 훈련 설정
     2 model.compile(optimizer='adam',
     3               loss='sparse_categorical_crossentropy',
     4               metrics=['accuracy'])
```

```
[ ]  1 # 학습
     2 model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.5133 - accuracy: 0.8204
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3899 - accuracy: 0.8601
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3528 - accuracy: 0.8722
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3307 - accuracy: 0.8799
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3110 - accuracy: 0.8860
<tensorflow.python.keras.callbacks.History at 0x7f22fa4ed450>
```

Colab 실습 : 이미지 분류

```
[ ]  1 # 정확도 평가
     2 test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
     3
     4 print('₩n테스트 정확도:', test_acc)
```

```
313/313 - 1s - loss: 0.3576 - accuracy: 0.8719

테스트 정확도: 0.8719000220298767
```

```
[ ]  1 #훈련된 모델활용한 예측
     2 predictions = model.predict(test_images)
```

```
[ ]  1 #예측 결과
     2 predictions[0]
```

```
array([7.1314716e-08, 1.4381334e-08, 3.1415359e-09, 1.1036498e-09,
       1.4995673e-07, 5.3564906e-03, 4.2783643e-09, 1.6525986e-02,
       3.1077241e-05, 9.7808617e-01], dtype=float32)
```

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

```
9
```

```
[ ]  1 test_labels[0]
```

```
9
```

Colab 실습 : 이미지 분류

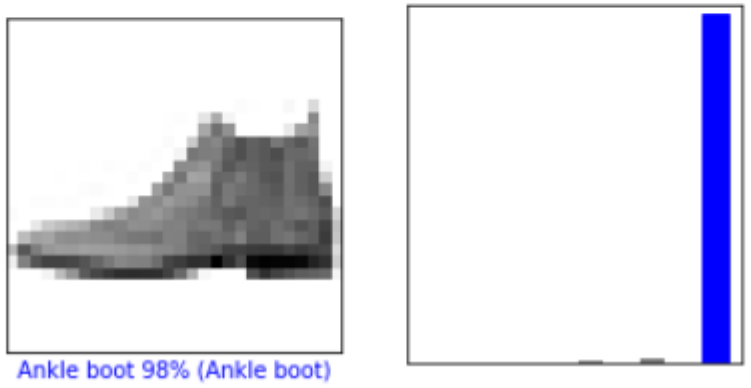
```
[ ]
# 각 클래스에 대한 예측을 표시하기 위한 함수
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  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)
```

Colab 실습 : 이미지 분류

```
21 def plot_value_array(i, predictions_array, true_label):
22   predictions_array, true_label = predictions_array[i], true_label[i]
23   plt.grid(False)
24   plt.xticks([])
25   plt.yticks([])
26   thisplot = plt.bar(range(10), predictions_array, color="#777777")
27   plt.ylim([0, 1])
28   predicted_label = np.argmax(predictions_array)
29
30   thisplot[predicted_label].set_color('red')
31   thisplot[true_label].set_color('blue')
```

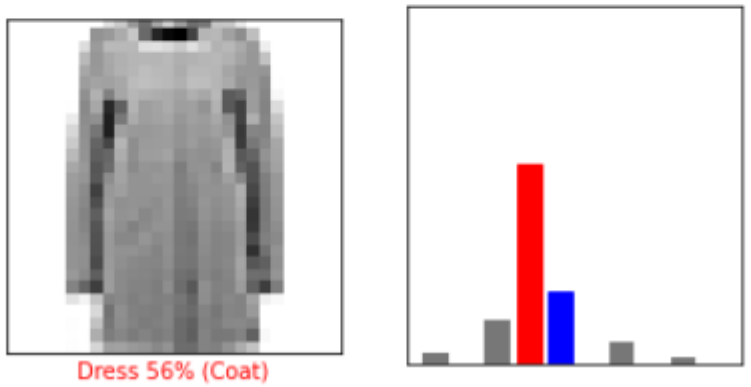
Colab 실습 : 이미지 분류

```
[ ]  1 i = 0
     2 plt.figure(figsize=(6,3))
     3 plt.subplot(1,2,1)
     4 plot_image(i, predictions, test_labels, test_images)
     5 plt.subplot(1,2,2)
     6 plot_value_array(i, predictions,  test_labels)
     7 plt.show()
```



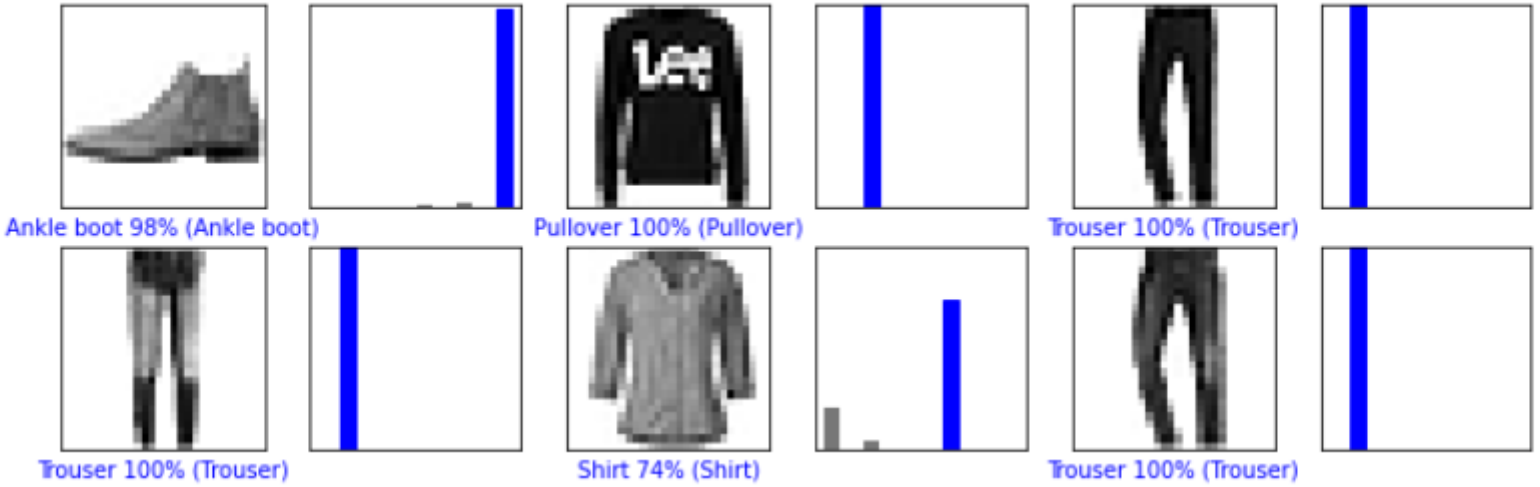
Colab 실습 : 이미지 분류

```
[ ]   1 i = 150
      2 plt.figure(figsize=(6,3))
      3 plt.subplot(1,2,1)
      4 plot_image(i, predictions, test_labels, test_images)
      5 plt.subplot(1,2,2)
      6 plot_value_array(i, predictions,  test_labels)
      7 plt.show()
```



Colab 실습 : 이미지 분류

```
[ ]  1 num_rows = 10
     2 num_cols = 3
     3 num_images = num_rows*num_cols
     4 plt.figure(figsize=(2*2*num_cols, 2*num_rows))
     5 for i in range(num_images):
     6   plt.subplot(num_rows, 2*num_cols, 2*i+1)
     7   plot_image(i, predictions, test_labels, test_images)
     8   plt.subplot(num_rows, 2*num_cols, 2*i+2)
     9   plot_value_array(i, predictions, test_labels)
    10 plt.show()
```



Colab 실습 : 이미지 분류

```
[ ]  1 # 이미지로 테스트
     2 img = test_images[0]; img = (np.expand_dims(img,0))
     3 #img = test_images[0:3]
     4 print(img.shape)
```

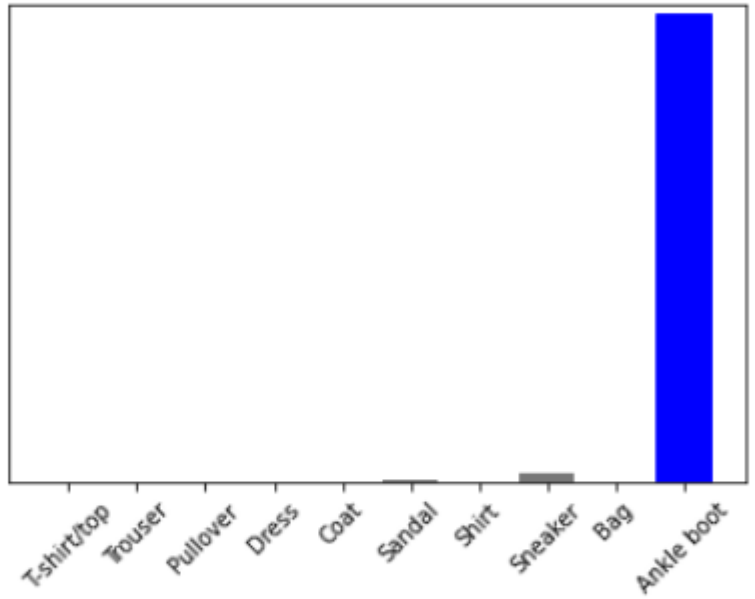
```
(1, 28, 28)
```

```
[ ]  1 predictions_single = model.predict(img)
     2
     3 print(predictions_single)
```

```
[[7.1314716e-08 1.4381306e-08 3.1415359e-09 1.1036498e-09 1.4995688e-07
  5.3564957e-03 4.2783723e-09 1.6525986e-02 3.1077241e-05 9.7808617e-01]]
```

Colab 실습 : 이미지 분류

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




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

```
9
```