Introduction to CNNs

IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

자율주행차는 표지판 유무와 같은 주변에서 일어나는 일을 신속하게 파악해야함.


YIELD

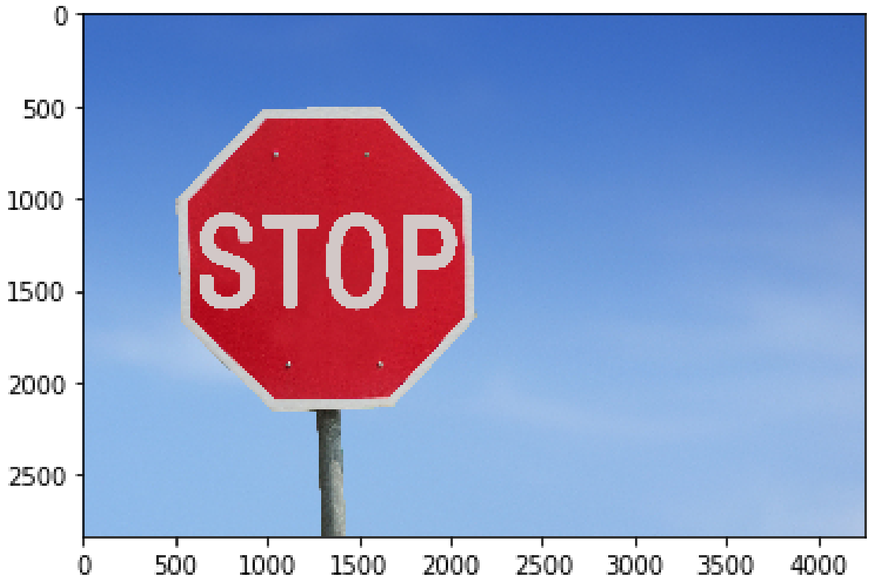
Software and pre-requisites

- **DataCamp's Deep Learning course**
- Machine learning:
  - Overfitting
  - Model evaluation
  - Cross-validation

머신러닝은 과적합과 교차 검증을 통한 모델 평가와 같은 기계학습의 기본 원리를 설명

Images as data

```
import matplotlib.pyplot as plt
data = plt.imread('stop_sign.jpg')
plt.imshow(data)
plt.show()
```



Images as data

```
data.shape
```

컴퓨터는 이미지를 볼 수 없고 컬러 이미지이기 때문에 3차원 배열로 저장됨

```
(2832, 4256, 3)
```

Images as data

```
data[1000, 1500]
```

이미지를 2차원으로 인덱싱을 1000과 1500으로 설정하면 빨간색 채널이 높은 것을 볼 수 있습니다.

```
array([0.73333333, 0.07843137, 0.14509804])
```

Images as data

```
data[250, 3500]
```

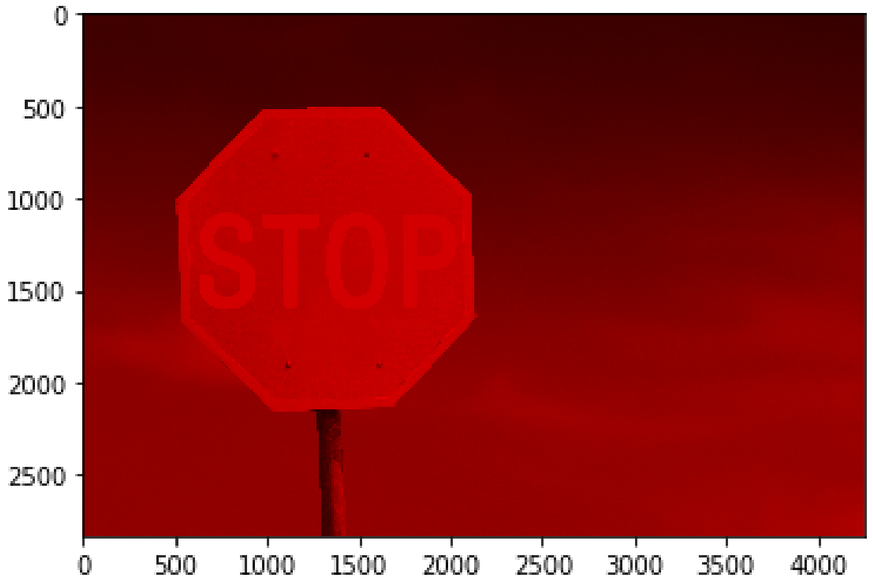
마찬가지로 B 채널이 높고 이미지도 파란색인 것을 확인 할 수 있습니다.

```
array([0.25882353, 0.43921569, 0.77254902])
```

Modifying image data

```
data[:, :, 1] = 0
data[:, :, 2] = 0
plt.imshow(data)
plt.show()
```

녹색과 파란색 값을 0으로 설정하면 이미지에서 빨간색 채널의 정보만 포함된 이미지가 생성됩니다.

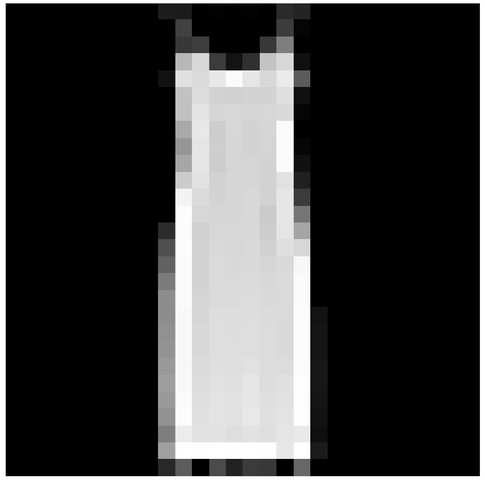


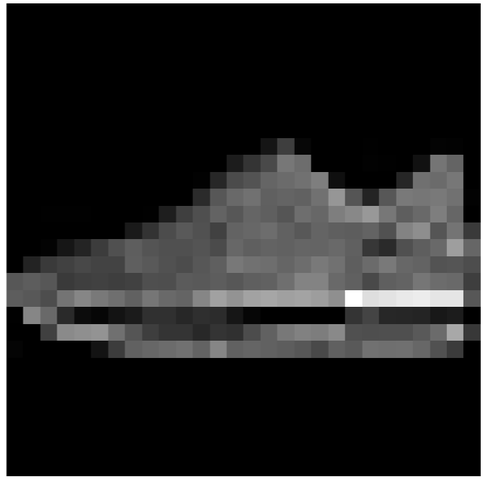
Changing an image

```
data[200:1200, 200:1200, :] = [0, 1, 0]
plt.imshow(data)
plt.show()
```

이미지의 영역을 설정하여 영역에 있는 채널을 모두 녹색으로 설정하면 이미지에서 보는 것처럼 녹색 사각형이 나타나게 됩니다.

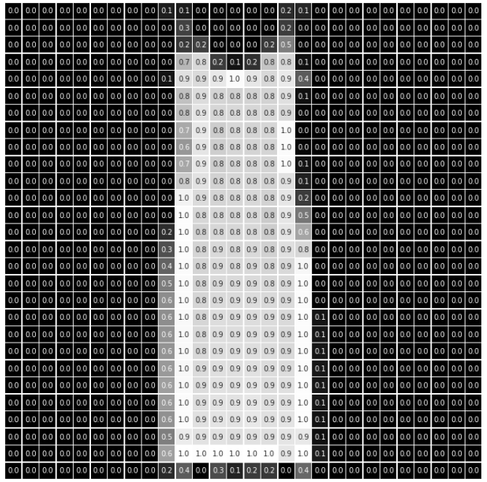
Black and white images

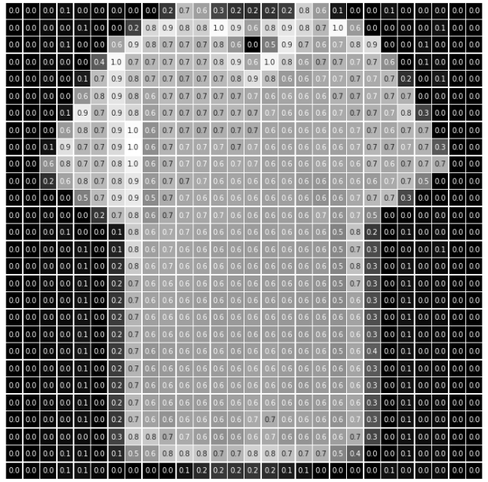


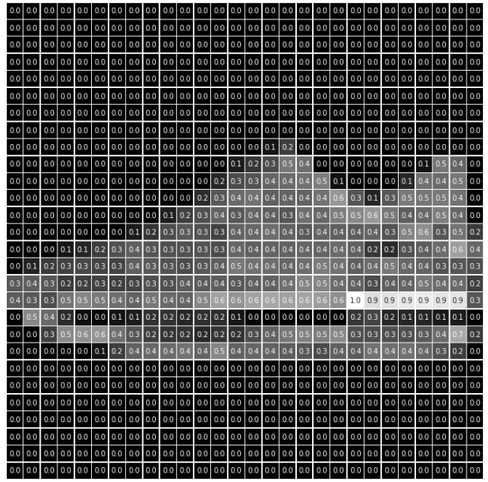


흑백이미지를 살펴보면

Black and white images





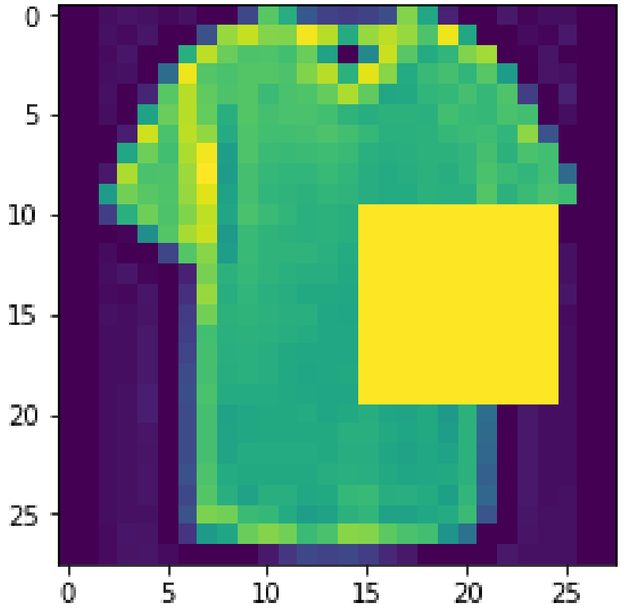


픽셀 숫자가 크면 박고 숫자가 작으면 어두운 부분을 나타냅니다.

Black and white images

```
tshirt[10:20, 15:25] = 1
plt.imshow(tshirt)
plt.show()
```

아까와 마찬가지로 영역을 지정하여 이미지의 일부를 변경할 수 있습니다.



Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON

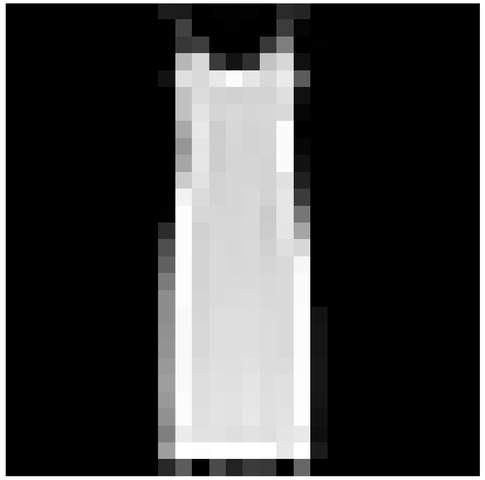
Classifying images

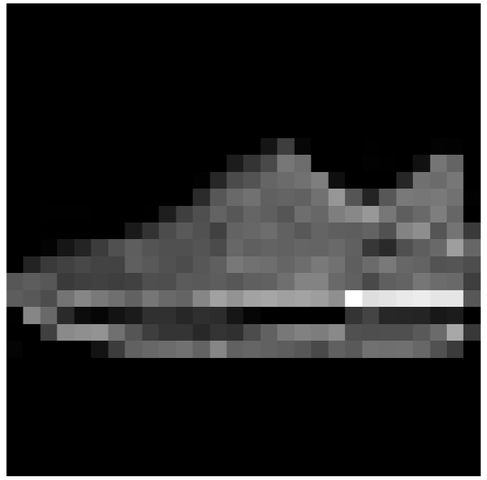
IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

Image classification





드레스, 티셔츠, 신발 등 클래스를 구별하는 알고리즘을 classification라고 합니다.

Image classification: training

train 단계에서 sample를 각 label과 함께 넣습니다.

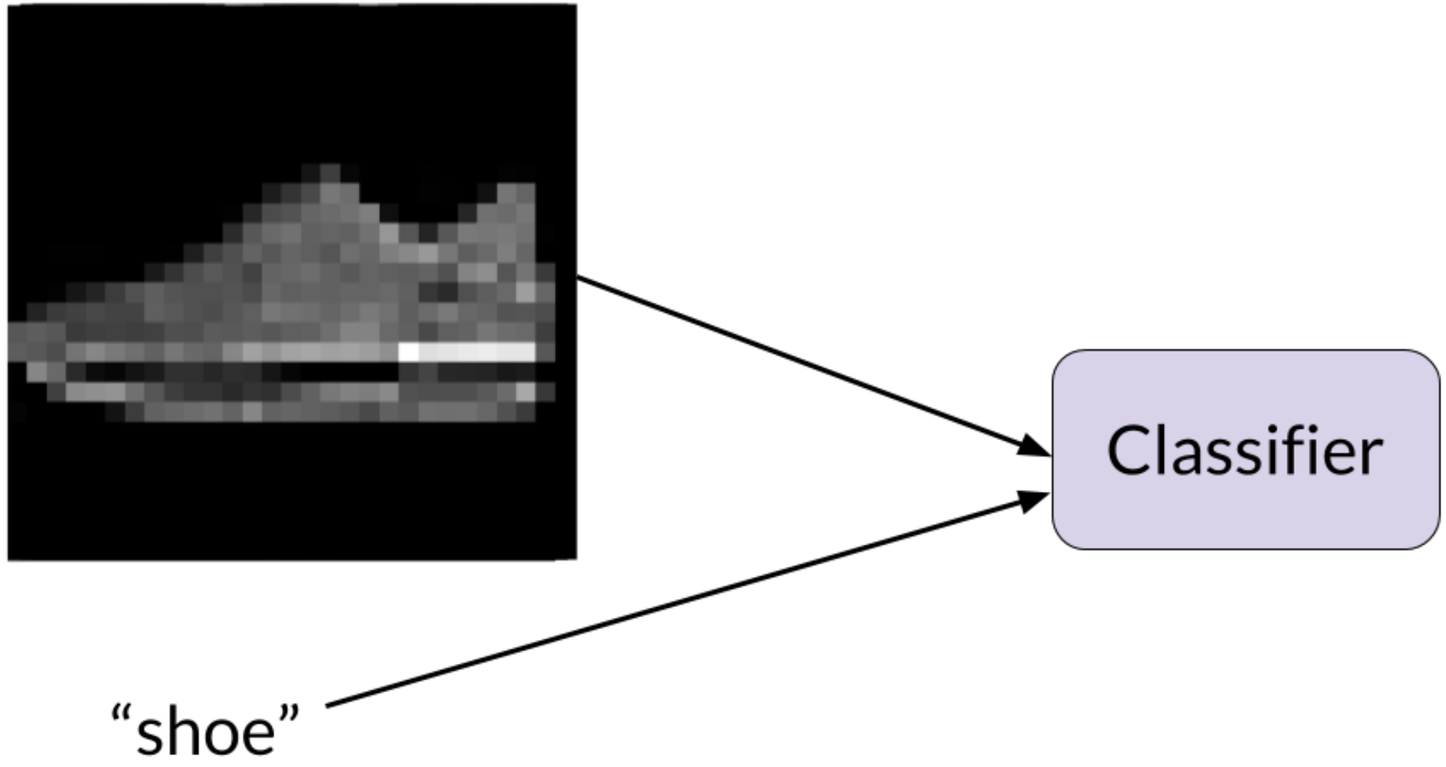


Image classification: training

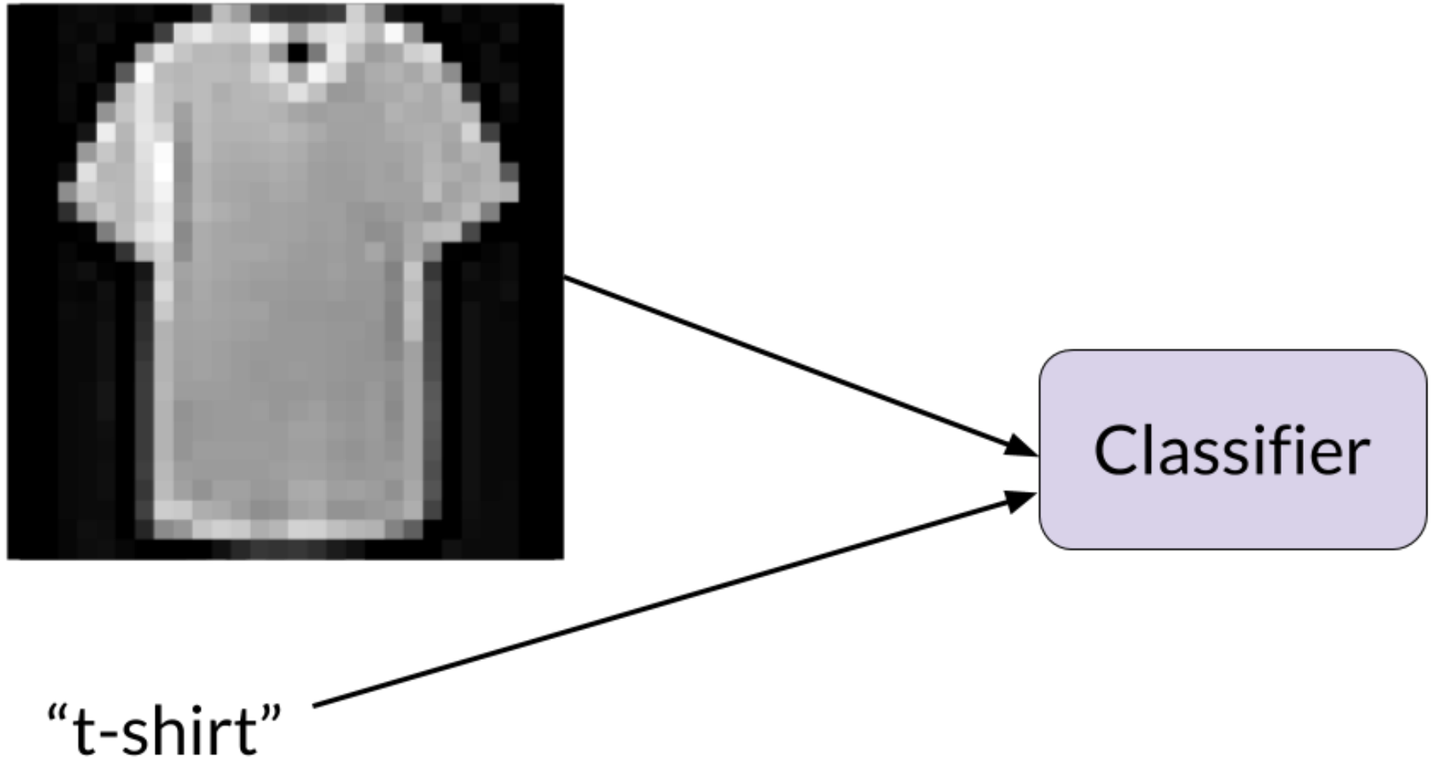


Image classification: training

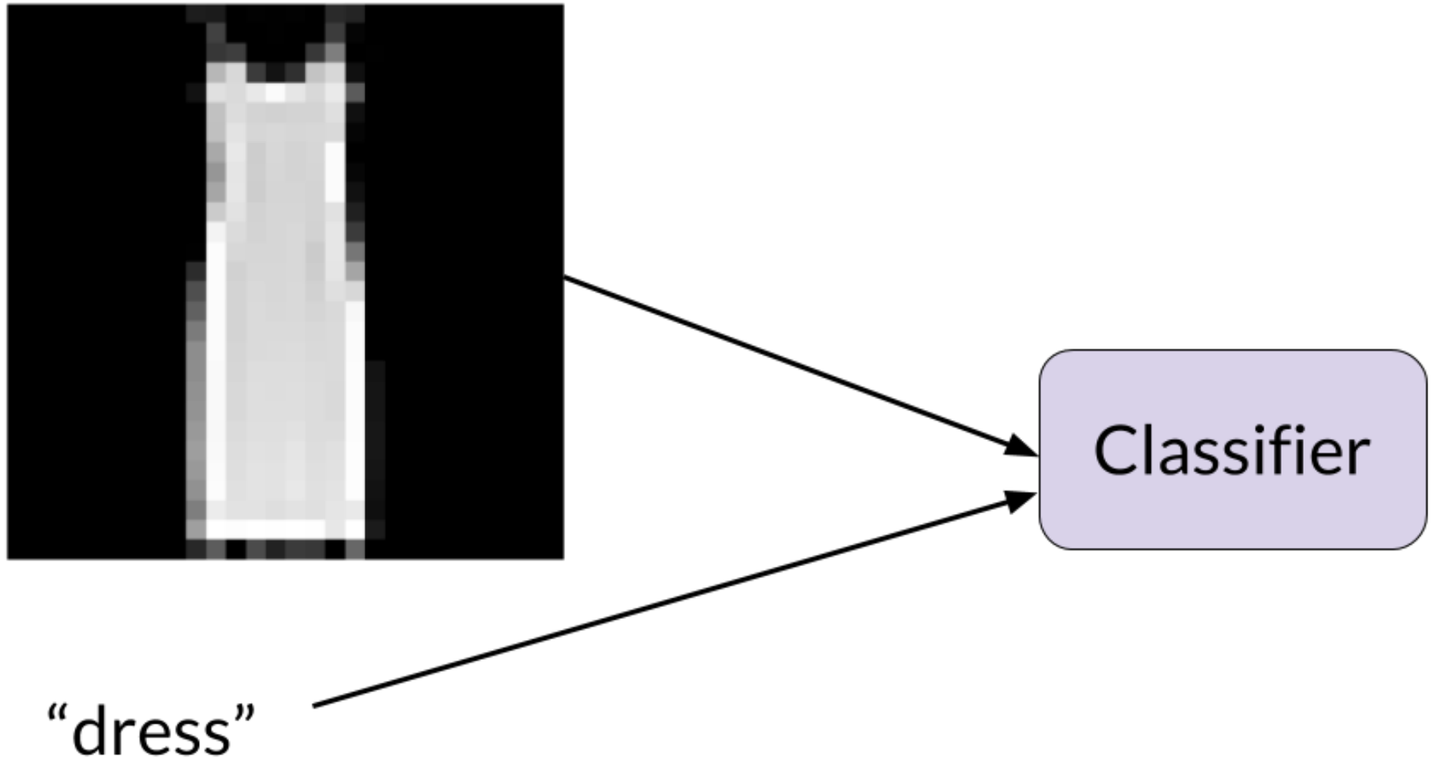
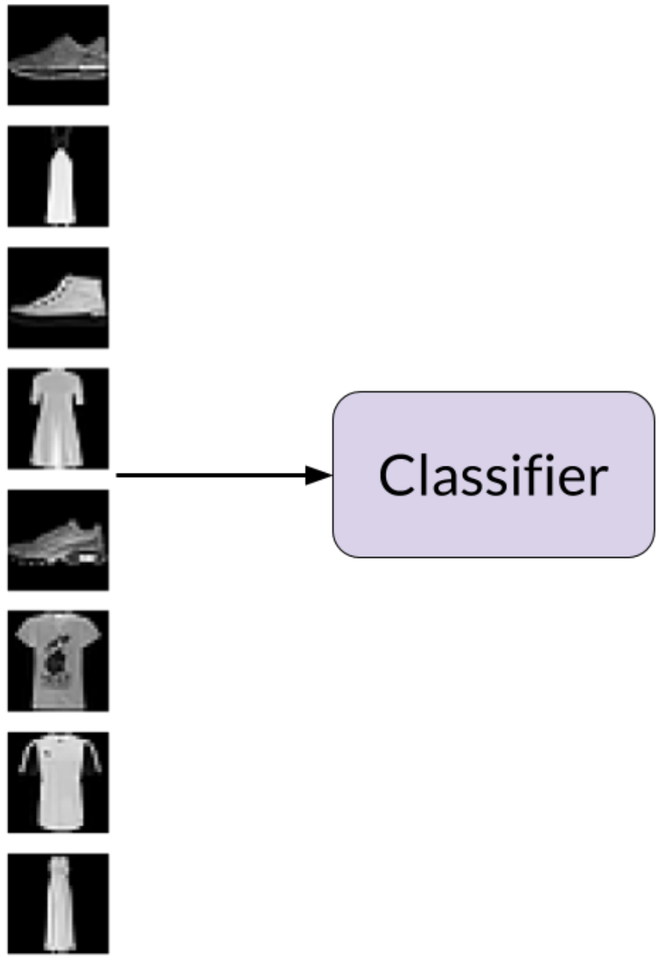


Image classification: evaluation



train이 끝나면 overfitting이 되었는지 데이터의 일부를 가지고 테스트를 합니다.

Image classification: evaluation

Classifier

“shoe”

“dress”

“shoe”

“t-shirt”

”shoe”

“t-shirt”

“shoe”

“dress”

대부분 맞게 분류했지만, 2개는 잘못 분류한 것을 볼 수 있습니다.

Representing class data: one-hot encoding

```
labels = ["shoe", "dress", "shoe", "t-shirt",
          "shoe", "t-shirt", "shoe", "dress"]
```

원핫인코딩을 이용해 label을 표현합니다.

Representing class data: one-hot encoding

```
array([[0., 0., 1.],    <= shoe
       [0., 1., 0.],    <= dress
       [0., 0., 1.],    <= shoe
       [1., 0., 0.],    <= t-shirt
       [0., 0., 1.],    <= shoe
       [1., 0., 0.],    <= t-shirt
       [0., 0., 1.],    <= shoe
       [0., 1., 0.]])   <= dress
```

자신이 아닌 class는 0으로 설정하여 label을 표시합니다.

One-hot encoding

```
categories = np.array(["t-shirt", "dress", "shoe"])
n_categories = 3
ohe_labels = np.zeros((len(labels), n_categories))
for ii in range(len(labels)):
    jj = np.where(categories == labels[ii])
    ohe_labels[ii, jj] = 1
```

배열을 0으로 초기화 한 후 sample에 해당하는 카테고리면 1을 넣습니다.

One-hot encoding: testing predictions

```
test
```

```
array([[0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.]])
```

```
(test * prediction).sum()
```

```
6.0
```

```
prediction
```

```
array([[0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.], <= incorrect
       [0., 0., 1.],
       [1., 0., 0.], <= incorrect
       [0., 0., 1.],
       [0., 1., 0.]])
```

원핫 인코딩을 가지고 예측을 해보니 8개 중 2개의 샘플을 잘못예측하였습니다.

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON

Image classification with Keras

IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

datacamp

Keras for image classification

```
from keras.models import Sequential
model = Sequential()
```

sequntial모델로 초기화합니다.

Keras for image classification

```
from keras.layers import Dense
train_data.shape
```

```
(50, 28, 28, 1)
```

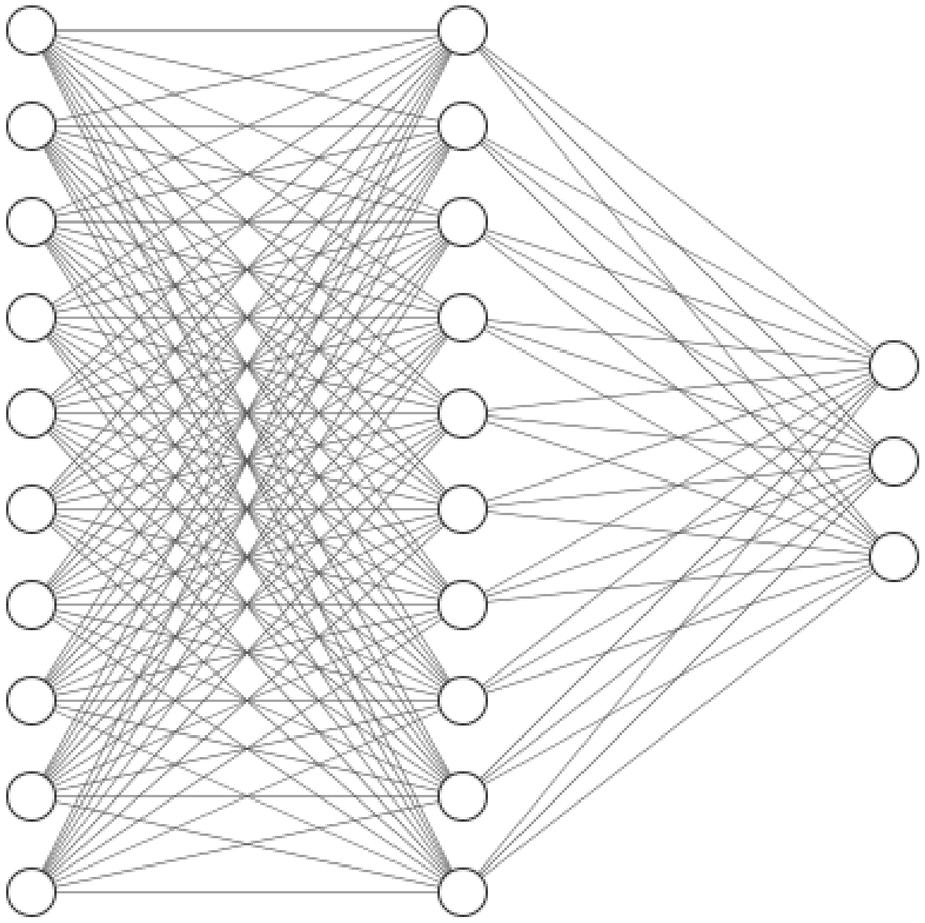
shape을 확인해보면 28X28 픽셀의 50개의 샘플을 확인 할 수 있습니다
(이미지가 흑백이기 때문에 마지막은 1)

Keras for image classification

```
model.add(Dense(10, activation='relu',
          input_shape=(784,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

input shape는 앞에서 이미지의 픽셀인 28X28이고

마지막 dense에 3을 추가하여 3개의 class를 softamax를 활성화 함수로 사용하여 출력층을 만듭니다.



Input Layer $\in \mathbb{R}^{10}$ Hidden Layer $\in \mathbb{R}^{10}$ Output Layer $\in \mathbb{R}^{3}$

Keras for image classification

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

optimizer을 adam으로 사용하고 metric를 정확도를 사용하여 모델을 컴파일 합니다.

Keras for image classification

```
train_data = train_data.reshape((50, 784))
```

model을 fit하기 전에 reshape을 사용하여 2차원으로 변환합니다.

Keras for image classification

```
model.fit(train_data, train_labels,
          validation_split=0.2,
          epochs=3)
```

```
model.fit(train_data, train_labels,
          validation_split=0.2,
          epochs=3)
```

출력해보면 샘플도 적고 epoch을 3밖에 하지 않았기 때문에 정확도가 낮은 것을 확인 할 수 있습니다.

```
Train on 40 samples, validate on 10 samples
Epoch 1/3
32/40 [=========>......] - ETA: 0s - loss: 1.0117 - acc: 0.4688
40/40 [=================] - 0s 4ms/step - loss: 1.0438 - acc: 0.4250
                          - val_loss: 0.9668 - val_acc: 0.4000
Epoch 2/3
32/40 [=========>......] - ETA: 0s - loss: 0.9556 - acc: 0.5312
40/40 [=================] - 0s 195us/step - loss: 0.9404 - acc: 0.5750
                         - val_loss: 0.9068 - val_acc: 0.4000
Epoch 3/3
32/40 [=========>......] - ETA: 0s - loss: 0.9143 - acc: 0.5938
40/40 [=================] - 0s 189us/step - loss: 0.8726 - acc: 0.6750
                         - val_loss: 0.8452 - val_acc: 0.4000
```

Keras for image classification

```
test_data = test_data.reshape((10, 784))
model.evaluate(test_data, test_labels)
```

test data를 가지고 평가해보면 loss는 1.019, accuracy는 40%정도입니다.

```
10/10 [==============================] - 0s 335us/step
[1.0191701650619507, 0.4000000059604645]
```

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON