

Machine Learning & Deep Learning

Lecture 24

패션 이미지 분류를 위한 딥러닝

김희숙 (H.S.Kim)

Convolutional Neural Network

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

● 합성곱 신경망

- ▶ 다층 퍼셉트론(MLP; Multi Layer Perceptron) 구조로 설계되며 이미지 인식 및 분류, 의료 영상 처리, 자율 주행 자동차 등의 **컴퓨터 비전(Computer Vision)** 분야에 주로 사용



Convolutional Neural Network

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

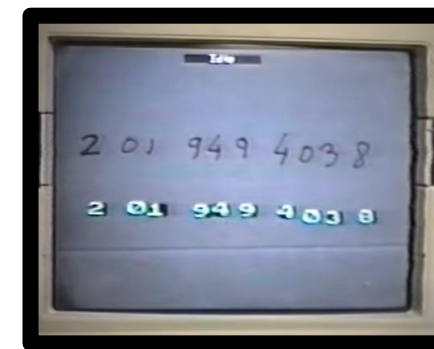
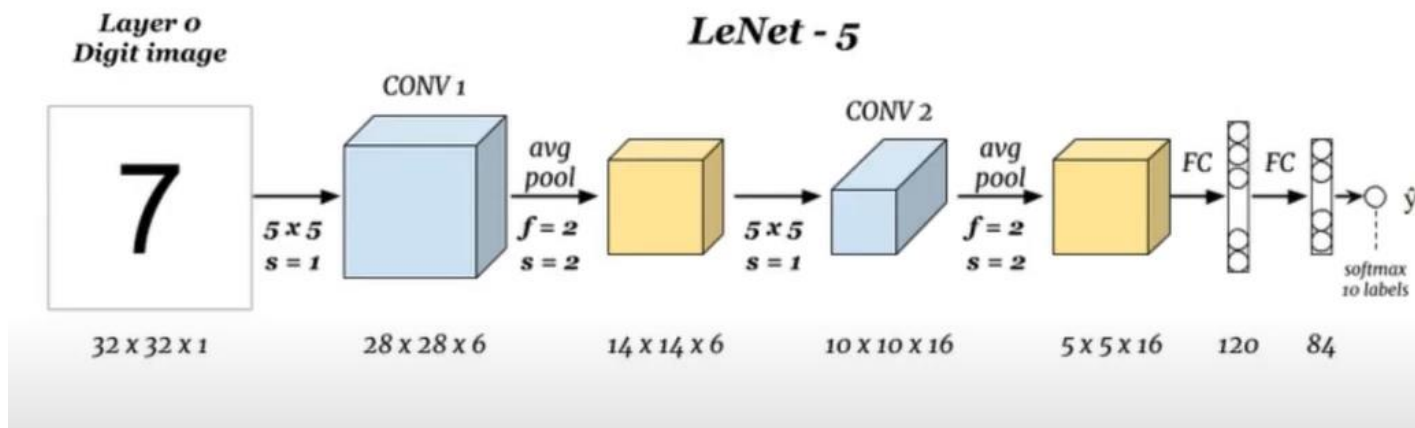


• Yann LeCun(얀 르쿤)

- 1989년, 미국의 통신 회사인 AT&T의 벨 연구소에 있을 때

Convolutional Neural Network 발표

- LeNet5가 은행에서 수표에 적힌 숫자 인식에 사용



Convolutional Neural Network

1. CNN 소개

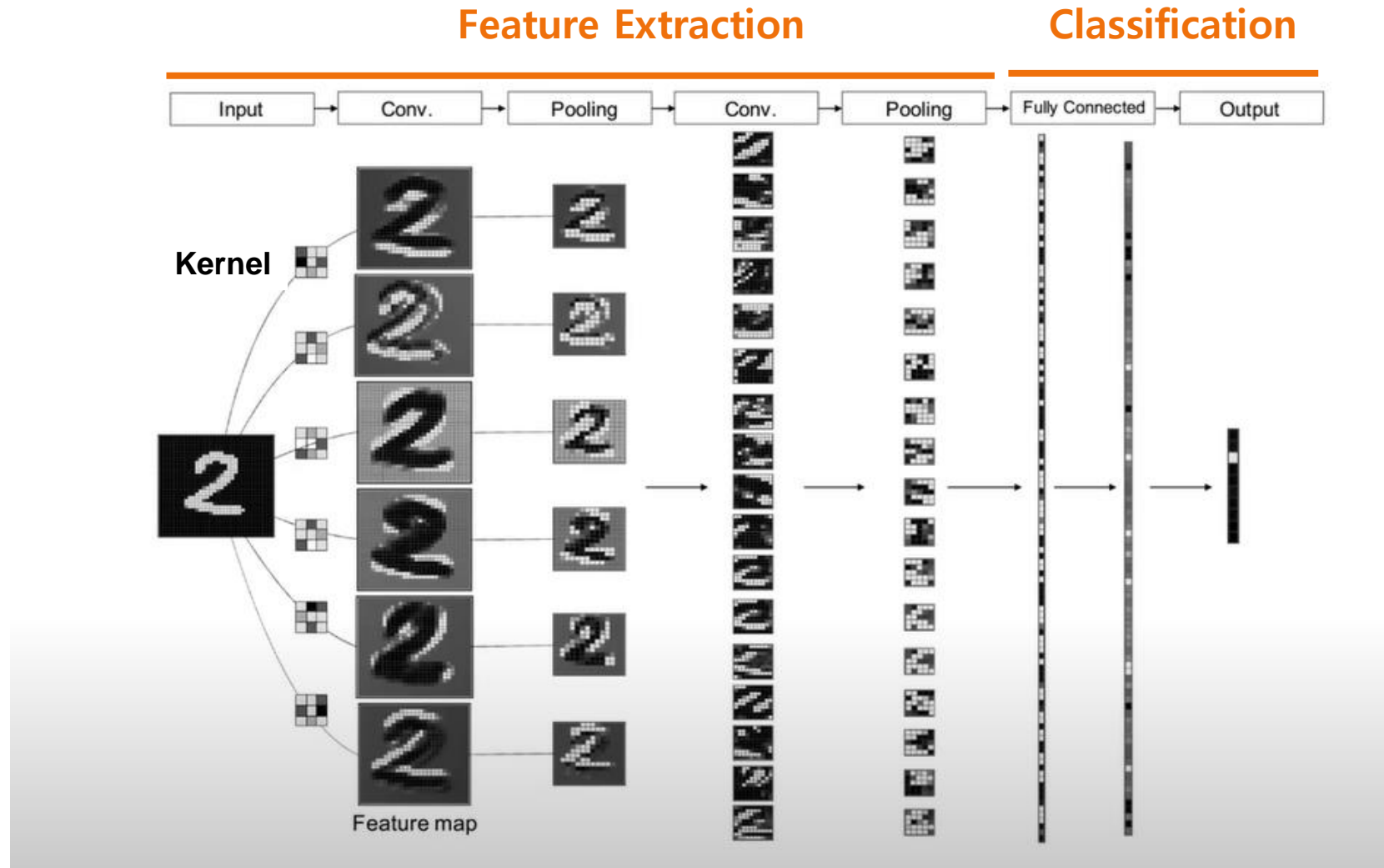
2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가



Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

1 Convolution Layer

- 입력된 이미지에서 특징을 추출하기 위해 Kernel을 활용

1	0	1	0
0	1	1	0
0	0	1	1
0	0	1	0

4×4 Input Image

예) 2×2 Kernel의 경우

- ✓ 각 셀에는 가중치가 들어있음
- ✓ 가중치를 ×1, ×0 라고 가정

×1	×0
×0	×1

2×2 Kernel

Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

- 4×4 Input Image와 2×2 Kernel을 Convolution

- ✓ Input Image에 Kernel의 가중치의 값을 각각 곱하고, 곱한 결과를 더하면 2가 됨

$$(1 \times 1) + (0 \times 0) + (0 \times 0) + (1 \times 1) = 2$$

1	0	1	0
0	1	1	0
0	0	1	1
0	0	1	0

4×4 Input Image

×

1	0
0	1

2×2
Kernel



stride = 1

2 Kernel이 한 칸씩 이동하면서 적용됨

1×1	0×0	1	0	1	0×1	1×0	0	1	0	1×1	0×0
0×0	1×1	1	0	0	1×0	1×1	0	0	1	1×0	0×1
0	0	1	1	0	0	0	1	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0
0×1	1×0	1	0	0	1×1	1×0	0	0	1	1×1	0×0
0×0	0×1	1	1	0	0	0×0	1×1	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0	1	1	0
0×1	0×0	1	1	0	0	0×1	1×0	1	0	0	0
0×0	0×1	1	0	0	0	0×0	1×1	0	0	0	0

Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

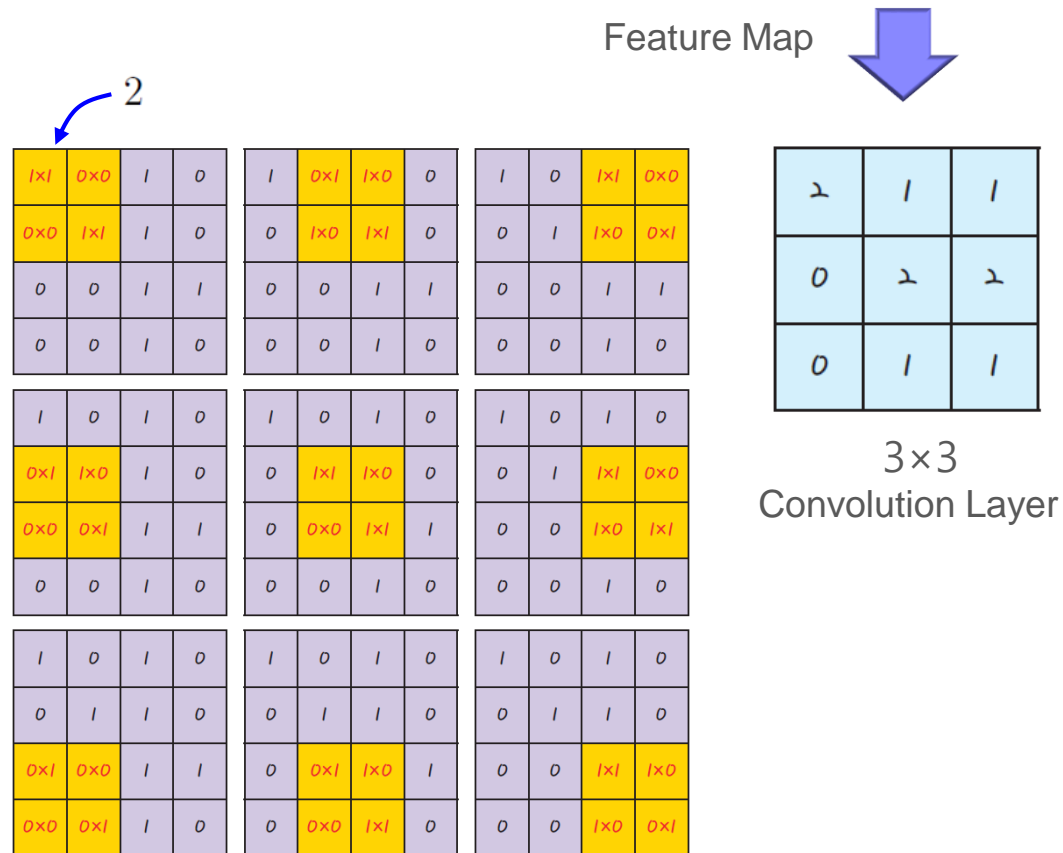
1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

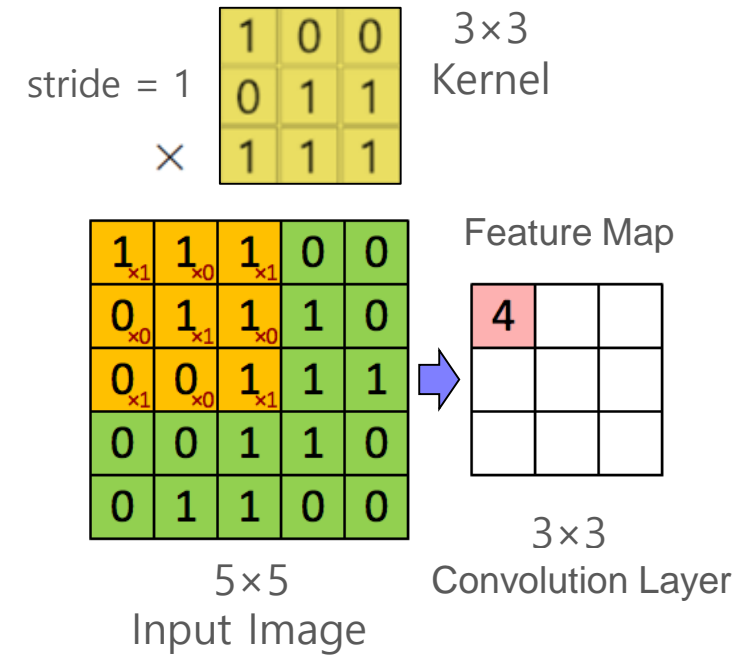
4) CNN 딥러닝 구현 결과에 대한 평가

- 4×4 Input Image와 2×2 Kernel을 Convolution



<참고>

- 5×5 Input Image와 3×3 Kernel을 Convolution



Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

■ Zero Padding

- ✓ Feature Map의 사이즈를 Input Image의 사이즈와 동일하게 생성

0	0	0	0	0	0	0	0
0	3	3	4	4	7	0	0
0	9	7	6	5	8	2	0
0	6	5	5	6	9	2	0
0	7	1	3	2	7	8	0
0	0	3	7	1	8	3	0
0	4	0	4	3	2	2	0
0	0	0	0	0	0	0	0

6×6
Input Image

*

1	0	-1
1	0	-1
1	0	-1

3×3
Kernel

=

-10	-13	1			
-9	3	0			

6×6
Convolution Layer

same

Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

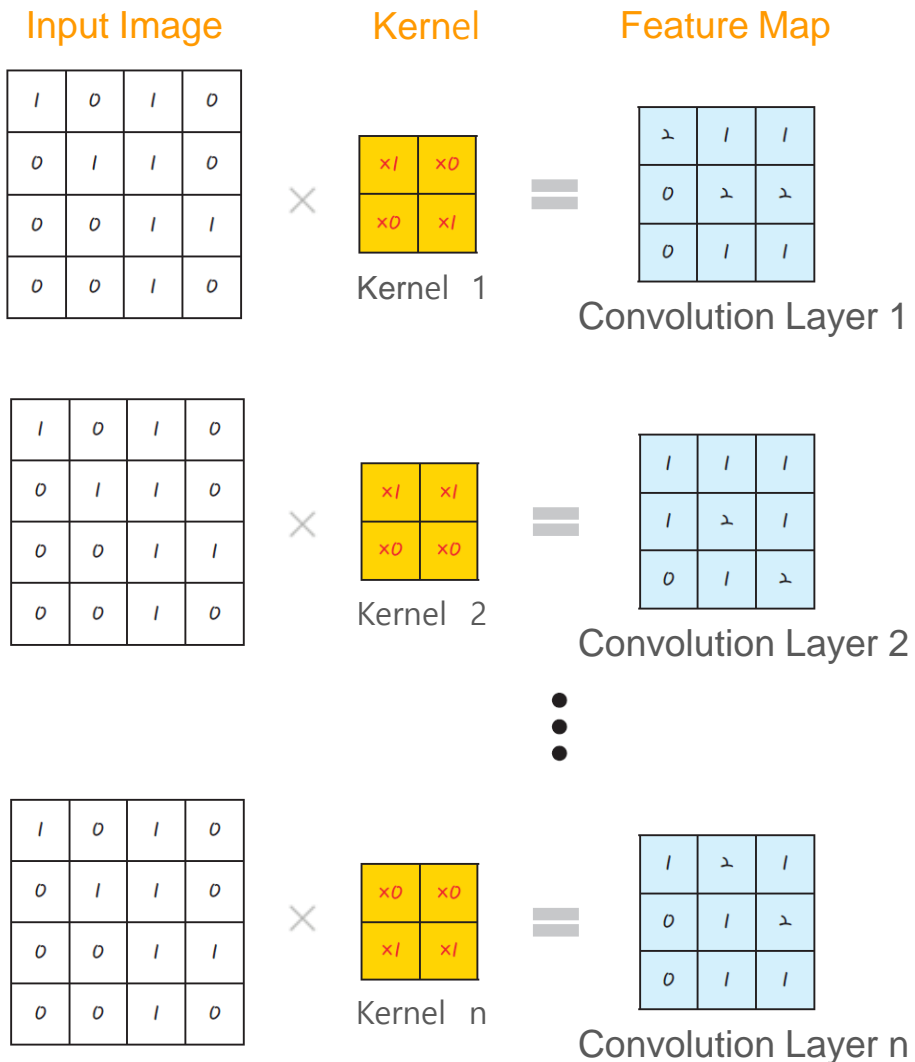
2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

■ 다양한 패턴의 Kernel을 여러 개 사용

- ✓ 여러 개의 Convolution된 이미지가 만들어져서 **다양한 Feature Map**을 획득할 수 있으므로 더욱 **정교한 특징**을 추출할 수 있다.



Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

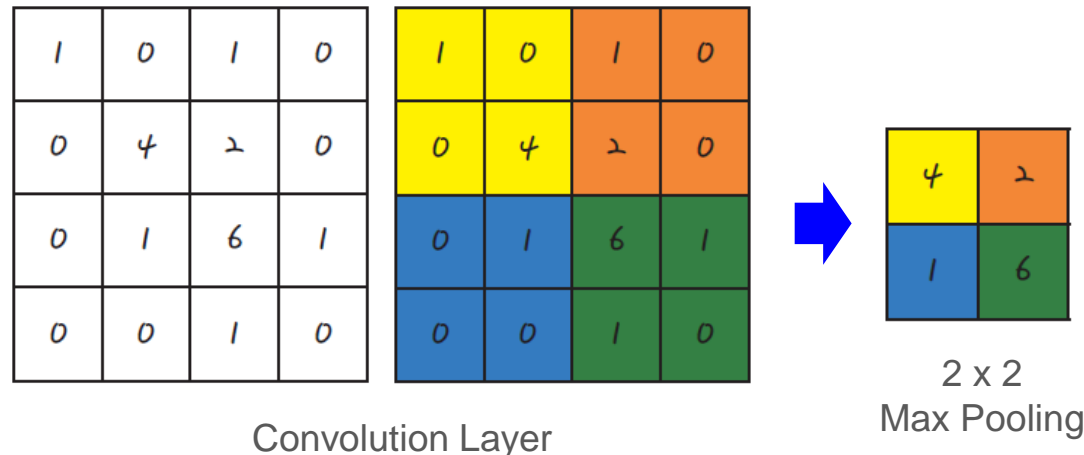
3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2 Pooling

- Convolution Layer의 데이터의 크기를 축소시키면서 이미지의 핵심 특징을 찾는 방식
 - ✓ **Max Pooling** : 정해진 구역 안에서 최대값을 추출하는 방식
 - ✓ **Average Pooling** : 정해진 구역 안에서 평균값을 추출하는 방식

예) 2×2 Max Pooling



Convolution Layer

Convolutional Neural Network (Feature Extraction)

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

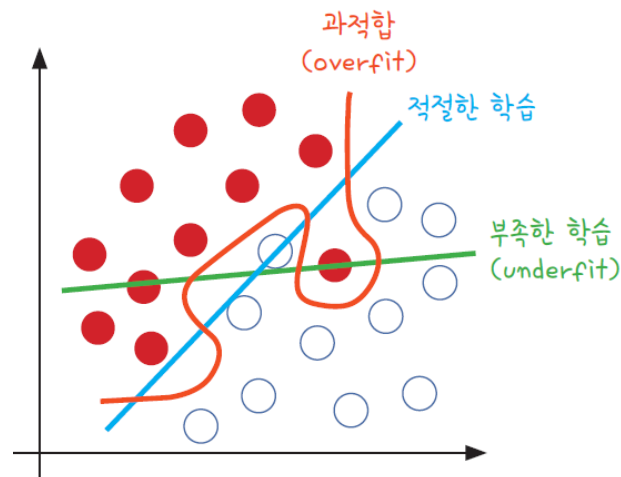
2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

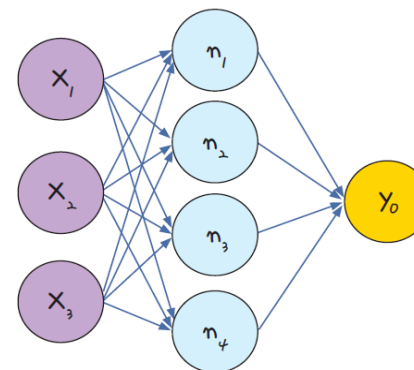
4) CNN 딥러닝 구현 결과에 대한 평가

3 Drop out

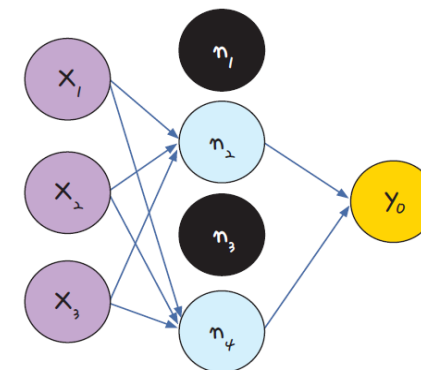
- 은닉층에 배치된 노드 중 일부의 노드를 신경망에서 차단시키는 것으로 Overfitting 해결하기 위해서 사용



Max pooling



Drop out



CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인











2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

CNN을 활용한 Fashion MNIST 딥러닝

Fashion MNIST Data set

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

- 독일 온라인 패션 유통업체 Zalando(잘란도)에서 패션 이미지 데이터 셋을 제공
- Training data set : 60,000 개 / Test data set : 10,000개의 총 70,000개 이미지 데이터 셋으로 구성

각 이미지는 28x28 픽셀(총 784 픽셀)의 그레이 스케일(0 ~ 255 사이의 정수) 이미지로 구성되어 있고, 10개로 레이블링됨

https://www.tensorflow.org/datasets/catalog/fashion_mnist

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

[1] Fashion MNIST 이미지 데이터셋 준비 및 확인

1. 라이브러리 임포트하기

```
In [1]: # 필요한 라이브러리 임포트
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import matplotlib.font_manager

%matplotlib inline
```

2. 이미지 데이터셋 불러오기 및 확인하기

2.1 트레이닝 및 테스트 이미지 데이터셋 불러오기

```
In [2]: fashion_train_df = pd.read_csv('./input/fashionmnist/fashion-mnist_train.csv', sep=',')
fashion_test_df = pd.read_csv('./input/fashionmnist/fashion-mnist_test.csv', sep=',')
```

총 785개의 컬럼으로 구성

label : 0 ~ 9 클래스

pixel1 ~ pixel784 : 0 ~ 255 그레이 스케일 픽셀 값

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2.2 트레이닝 이미지 데이터셋 확인하기

```
In [3]: fashion_train_df.head() # 트레이닝 데이터셋 head 부분
```

```
Out[3]:
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	6	0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	0	0	0	0	0
3	0	0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	1	0	0	0	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

2.3 테스트 이미지 데이터셋 확인하기

```
In [5]: fashion_test_df.head() # 테스트 데이터셋 head 부분
```

```
Out[5]:
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	0	0	0	0	0	0	0	0	9	8	...	103	87	56	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	34	0	0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	14	53	99	...	0	0	0	0	63	53	31	0	0	0
3	2	0	0	0	0	0	0	0	0	0	...	137	126	140	0	133	224	222	56	0	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2.4 트레이닝 및 테스트 이미지 데이터셋 최종 확인하기

In [7]:

```
# 트레이닝 데이터셋
print(fashion_train_df.shape)
print(type(fashion_train_df))

(60000, 785)
<class 'pandas.core.frame.DataFrame'>
```

In [8]:

```
# 테스트 데이터셋
print(fashion_test_df.shape)
print(type(fashion_test_df))

(10000, 785)
<class 'pandas.core.frame.DataFrame'>
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

3. 이미지 데이터셋을 numpy.ndarray 타입으로 변환하기

```
In [9]: train = np.array(fashion_train_df, dtype='float32') # 트레이닝 데이터셋  
        test = np.array(fashion_test_df, dtype='float32')  # 테스트 데이터셋
```

```
In [10]: # 트레이닝 데이터셋  
         print(train.shape)  
         print(type(train))  
  
(60000, 785)  
<class 'numpy.ndarray'>
```

```
In [11]: # 테스트 데이터셋  
         print(test.shape)  
         print(type(test))  
  
(10000, 785)  
<class 'numpy.ndarray'>
```


CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

4. 이미지 데이터셋에서 랜덤으로 이미지를 선택하여 시각화하기

4.1 트레이닝 이미지 데이터셋 중에서 랜덤으로 하나의 이미지를 선택하여 시각화하기

In [14]:

```
print(fashion_train_df["label"]) # 현재 트레이닝 데이터셋 60,000개 : 0 ~ 9에 해당하는 label로 구성되어 있음
```

```
0      2
1      9
2      6
3      0
4      3
```

```
..
59995   9
59996   1
59997   8
59998   8
59999   7
```

```
Name: label, Length: 60000, dtype: int64
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

In [15]:

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

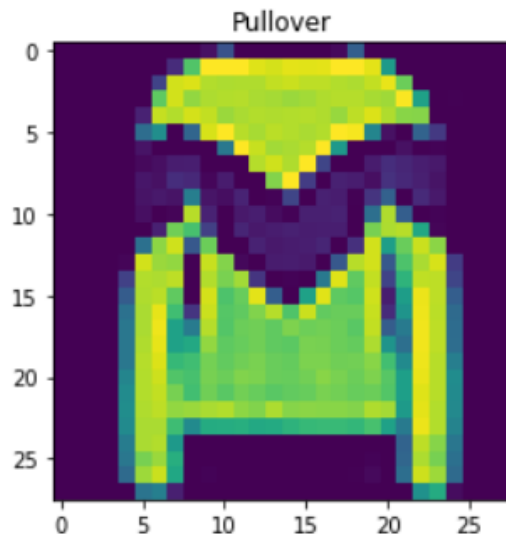
# 이미지와 레이블 확인
i = random.randint(1,60000) # 트레이닝 데이터셋 개수가 60,000개이므로 1 ~ 60,000까지 수 중에서 임의의 수를 선택
plt.imshow(train[i,:].reshape((28,28))) # 준비된 이미지 28x28(784 pixel)에 적합하게 이미지 출력

label_index = fashion_train_df["label"][i] #label
plt.title(f"{class_names[label_index]}") #이미지 출력 시, label을 class_name으로 대체하여 출력

print(i, '번째 이미지 선택')
print("label_index : ", label_index, end='')
print("\t\t class name : ", class_names[label_index])
```

48350 번째 이미지 선택

label_index : 2 class name : Pullover



Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

4.2 트레이닝 이미지 데이터셋 중에서 랜덤으로 다수의 이미지를 선택하여 시각화하기

In [17]:

```
# 그리드 형식으로 이미지 출력
W_grid = 15
L_grid = 15
fig, axes = plt.subplots(W_grid, L_grid, figsize=(17, 17)) # 15행, 15열, 각 이미지 사이즈(17x17)

axes = axes.ravel() # 15x15 구조의 2차원 타일을 1차원 타일로 평면화
n_train = len(train) # 트레이닝 데이터셋의 길이를 n_train에 할당, 60000 이 할당됨

for i in np.arange(0, W_grid * L_grid): # 0에서 224까지 총 255번 반복 수행
    index = np.random.randint(0, n_train) # 60,000개의 트레이닝 데이터셋 중에서 랜덤 값 선택

    # 랜덤으로 선택된 index에 해당하는 이미지 보여주기
    axes[i].imshow( train[index,1:].reshape((28,28)) ) # 준비된 이미지 28x28(784 pixel)에 적합하게 이미지 출력
    label_index = int(train[index,0]) # label
    axes[i].set_title(class_names[label_index], fontsize=12) # 이미지 출력 시, label을 class_name으로 대체하여 출력
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.4, wspace=0.4) # 서브 플롯 레이아웃 너비와 높이의 간격 비율 0.4
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

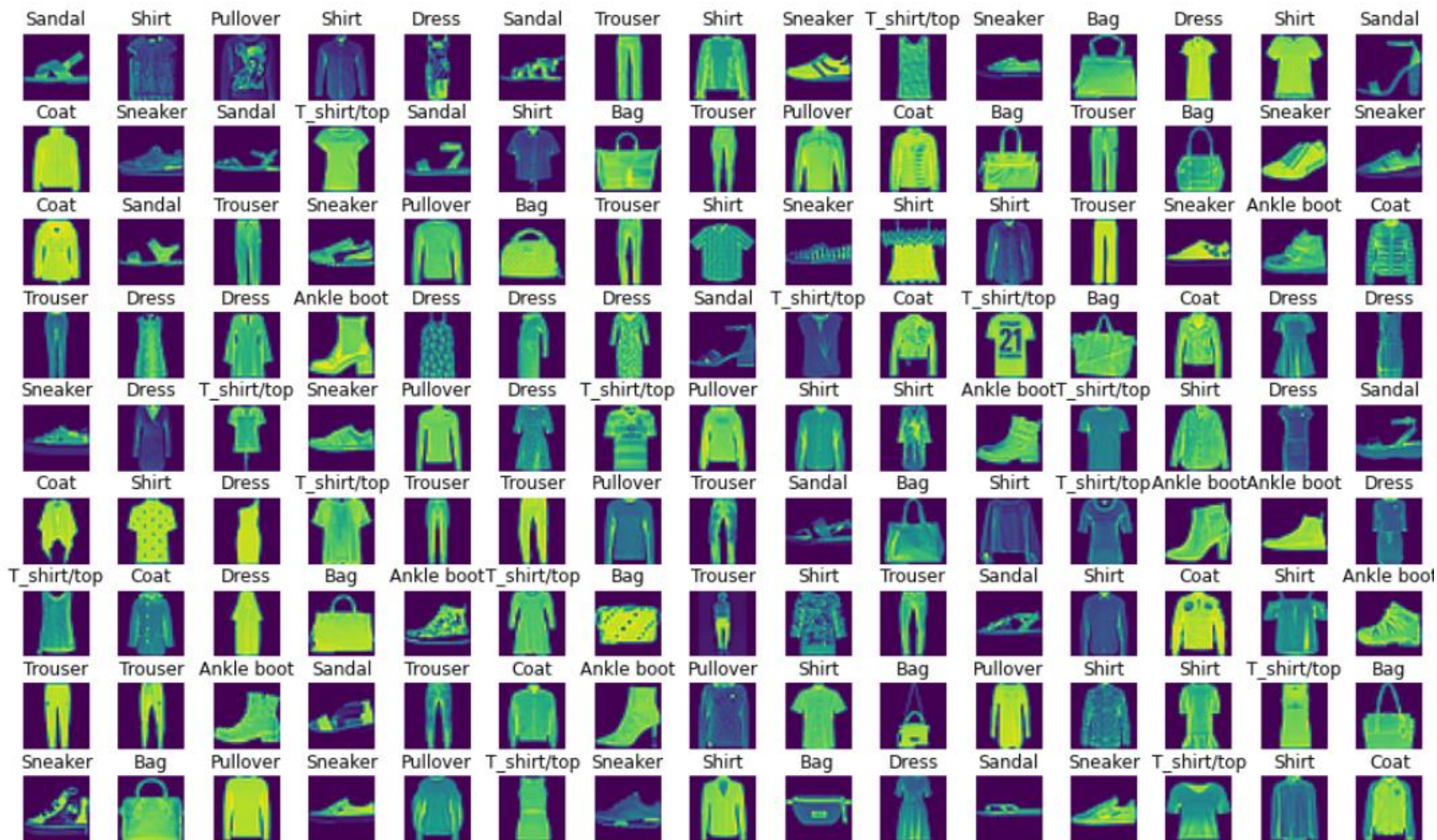
2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가



CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

[2] 이미지 데이터셋 정규화 및 분포 확인

1. 이미지 데이터셋 정규화하기

이미지 데이터셋 값을 딥러닝 신경망 모델에 적용하기 전에 0 ~ 1 사이 값 범위로 정규화 작업 수행
(현재 픽셀 값이 0~255 이므로, 255로 나누면 됨)

```
In [18]: # 트레이닝 및 테스트 데이터셋 정규화
X_train = train[:, 1:] / 255
Y_train = train[:, 0]

X_test = test[:, 1:] / 255
y_test = test[:, 0]
```

정규화 작업 후, 잘 수행되었는지 테스트하기

```
In [19]: plt.figure(figsize=(10, 10))

for i in range(25): # 처음 0번 ~ 24번 이미지까지 출력 테스트
    plt.subplot(5, 5, i+1) # 5행 5열 구조로 출력
    plt.xticks([])
    plt.yticks([])
    plt.imshow(X_train[i].reshape((28,28))) # 트레이닝 데이터셋

    label_index = int(Y_train[i]) # 트레이닝 데이터셋 label에 해당하는 0 ~ 9 사이의 값
    plt.title(class_names[label_index]) # #이미지 출력 시, label_index를 class_name으로 대체하여 출력
plt.show()
```


CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가



Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2. 이미지 데이터셋 분포 확인하기

```
In [20]: font_path = "C:\Windows\Fonts\gulim.ttc" # 한글 폰트 설정
font_name = matplotlib.font_manager.FontProperties(fname=font_path).get_name()
plt.rc('font', family=font_name, size=10)
```

```
In [21]: plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1) # 좌측 이미지
classes, counts = np.unique(Y_train, return_counts=True)
plt.barh(class_names, counts) #수평 막대 플롯
plt.title('트레이닝 데이터 세트의 클래스 분포')
print("트레이닝 데이터 :", class_names)
print("트레이닝 데이터 :", counts)
```

```
plt.subplot(2, 2, 2) # 우측 이미지
classes, counts = np.unique(y_test, return_counts=True)
plt.barh(class_names, counts)
plt.title('테스트 세트의 클래스 분포')
print("테스트 데이터 :", class_names)
print("테스트 데이터 :", counts)

plt.subplots_adjust(wspace=0.4)
```

```
트레이닝 데이터 : ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
트레이닝 데이터 : [6000 6000 6000 6000 6000 6000 6000 6000 6000 6000]
테스트 데이터 : ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
테스트 데이터 : [1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

트레이닝 데이터 : ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
트레이닝 데이터 : [6000 6000 6000 6000 6000 6000 6000 6000 6000 6000]
테스트 데이터 : ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
테스트 데이터 : [1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]

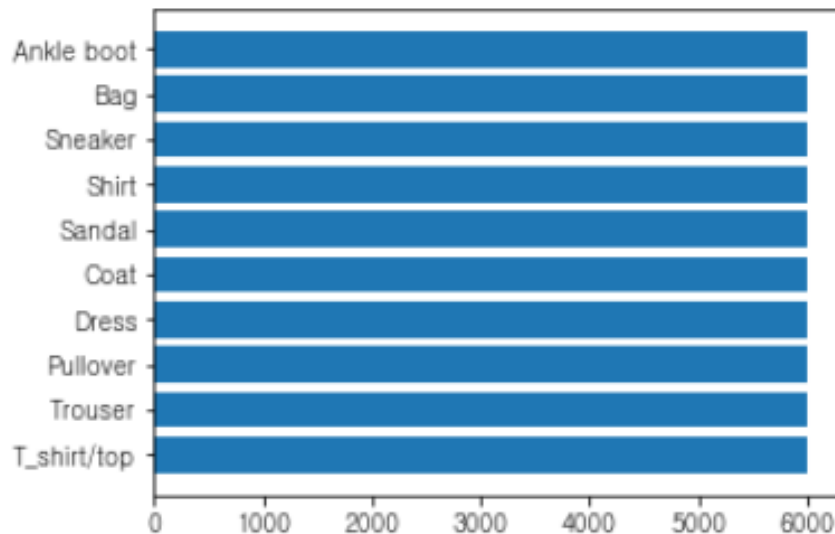
1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

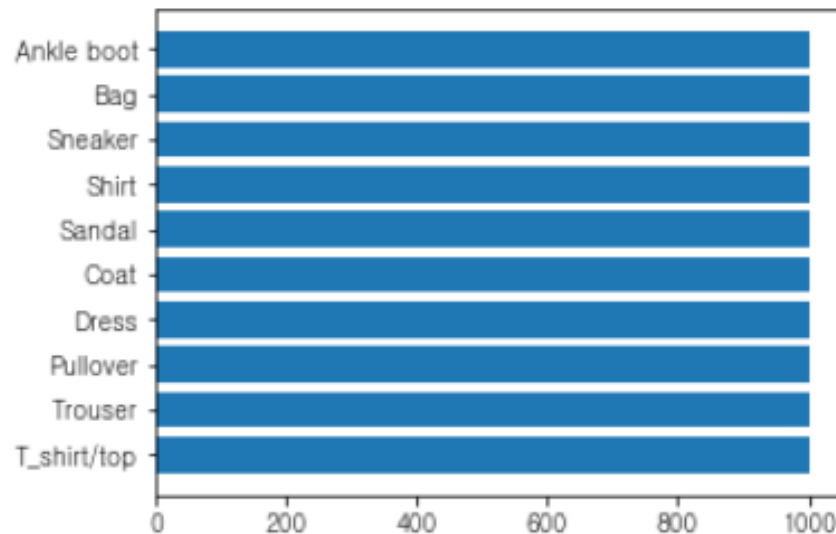
3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

트레이닝 데이터 세트의 클래스 분포



테스트 세트의 클래스 분포



CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

[3] CNN 딥러닝 구현

1. 트레이닝 이미지 데이터셋을 train 과 validate 용도로 분리하기

딥러닝 수행 시, 과적합(overfitting)을 방지하기 위해 트레이닝 및 벨리데이션 세트로 분할

```
In [22]: from sklearn.model_selection import train_test_split

x_train, x_validate, y_train, y_validate = train_test_split(X_train, Y_train, random_state=42)

# random_state=42 사용하여 데이터셋 셔플 과정에서 매번 같은 값을 획득
# train 75%, validate 25%
```

```
In [23]: x_train.shape #75%
```

```
Out [23]: (45000, 784)
```

```
In [24]: x_validate #25%
```

```
Out [24]: array([[0.      , 0.      , 0.      , ..., 0.      , 0.      , 0.      ],
 [0.      , 0.      , 0.      , ..., 0.827451, 0.4      , 0.      ],
 [0.      , 0.      , 0.      , ..., 0.      , 0.      , 0.      ],
 ...,
 [0.      , 0.      , 0.      , ..., 0.      , 0.      , 0.      ],
 [0.      , 0.      , 0.      , ..., 0.      , 0.      , 0.      ],
 [0.      , 0.      , 0.      , ..., 0.      , 0.      , 0.      ]],
 dtype=float32)
```

```
In [25]: print(x_train.shape) # 60,000개 중 75%, 즉 45000개 획득, 784(28*28)개의 픽셀 데이터로 구성
print(y_train.shape) # 60,000개 중 75%, 즉 45000개 획득, label(0~9)에 해당하는 부분
print(x_validate.shape) # 60,000개 중 25%, 즉 15000개 획득, 784(28*28)개의 픽셀 데이터로 구성
print(y_validate.shape) # 60,000개 중 25%, 즉 15000개 획득, label(0~9)에 해당하는 부분

(45000, 784)
(45000,)
(15000, 784)
(15000,)
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2. 이미지 데이터셋 shape 변경하기

2.1 트레이닝 이미지 데이터셋 (train 과 validate 이미지 데이터셋) shape 변경하기

tensorflow 및 keras를 이용한 딥러닝 프레임워크를 효율적으로 사용할 수 있도록 X_train, x_validate 구조를 변경한다.

```
In [26]: x_train = x_train.reshape(x_train.shape[0], *(28, 28, 1))
          x_validate = x_validate.reshape(x_validate.shape[0], *(28, 28, 1))
```

```
In [27]: print(x_train.shape) # 45000, 784 에서 784는 28*28 이미지 픽셀 데이터를 의미함
          print(y_train.shape) # 45000, 는 레이블에 해당하는 부분
          print(x_validate.shape)
          print(y_validate.shape)

(45000, 28, 28, 1)
(45000,)
(15000, 28, 28, 1)
(15000,)
```

2.2 테스트 이미지 데이터셋 shape 변경하기

```
In [28]: X_test.shape
```

```
Out[28]: (10000, 784)
```

tensorflow 및 keras를 이용한 딥러닝 프레임워크를 효율적으로 사용할 수 있도록 X_test 구조를 변경한다.

```
In [29]: X_test = X_test.reshape(X_test.shape[0], *(28, 28, 1))
```

```
In [30]: print(X_test.shape) # 10000, 28, 28, 1 마지막 1은 그레이 이미지를 의미 (3을 사용하면 컬러)

(10000, 28, 28, 1)
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

3. CNN 딥러닝을 위한 라이브러리 임포트하기

```
In [31]: import keras  
         # import tensorflow
```

Using TensorFlow backend.

```
In [32]: from keras.models import Sequential  
         from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization  
         from keras.optimizers import Adam  
         from keras.callbacks import TensorBoard
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

4. CNN 딥러닝을 위한 모델링 및 실행하기

4.1 특징 추출을 위한 모델링 (Feature Extraction Modeling) 설계하기

- `cnn_model.add(BatchNormalization())`
배치 단위로 활성화 함수의 활성화 값 또는 출력값을 정규화(정규분포: 평균 0, 분산 1)하여 학습 속도를 개선
- `cnn_model.add(MaxPooling2D(pool_size=(2, 2)))`
`pool_size=(2, 2)`는 데이터 크기가 가로 세로 각각 절반으로 줄어들게 하고, `MaxPooling2D`는 주어진 이미지를 2차원으로 생성
- `cnn_model.add(Dropout(0.2))`
20% 드롭아웃으로 과적합을 방지

In [33]:

```
cnn_model = Sequential()

# 첫번째 필터 : 32 filters 활용한 이미지 특징 추출
cnn_model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(28,28,1), activation='relu', padding='same'))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.2)) # 20% 드롭아웃으로 과적합을 방지

# 두번째 필터 : 64 filters 활용한 이미지 특징 추출
cnn_model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.2))
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

4.2 분류를 위한 모델링 (Classification Modeling) 설계하기

In [34]:

```
# 1차원으로 이미지를 재구성
cnn_model.add(Flatten()) # Flatten() 은 주어진 이미지를 1차원으로 생성
cnn_model.add(Dense(units=128, activation='relu'))
cnn_model.add(Dropout(0.2))
cnn_model.add(Dense(units=10, activation='softmax'))
```

4.3 모델 컴파일 및 실행하기

모델 컴파일

parse_categorical_crossentropy는 다중 분류 손실함수로,

categorical_crossentropy와 동일하지만 레이블링 결과가 one-hot encoding 형태가 아닌 integer type 클래스일 경우에 사용할 수 있다.

In [35]:

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

모델 실행

x_train, x_validate, y_train, y_validate = train_test_split(X_train, Y_train, random_state=42)

위에서 준비한 Train 그리고 Validate 데이터셋을 기반으로 실행

In [36]:

```
epochs = 15
batch_size = 32
history = cnn_model.fit(x_train, y_train, epochs=epochs, verbose=1, validation_data=(x_validate, y_validate))
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

In [36]:

```
epochs = 15
batch_size = 32
history = cnn_model.fit(x_train, y_train, epochs=epochs, verbose=1, validation_data=(x_validate, y_validate))
```

Train on 45000 samples, validate on 15000 samples

```
Epoch 1/15
45000/45000 [=====] - 54s 1ms/step - loss: 0.4831 - accuracy: 0.8276 - val_loss: 0.3382 - val_accuracy: 0.8711
Epoch 2/15
45000/45000 [=====] - 53s 1ms/step - loss: 0.3382 - accuracy: 0.8761 - val_loss: 0.3234 - val_accuracy: 0.8847
Epoch 3/15
45000/45000 [=====] - 54s 1ms/step - loss: 0.2973 - accuracy: 0.8900 - val_loss: 0.2549 - val_accuracy: 0.9073
Epoch 4/15
45000/45000 [=====] - 54s 1ms/step - loss: 0.2725 - accuracy: 0.8992 - val_loss: 0.2858 - val_accuracy: 0.8923
Epoch 5/15
45000/45000 [=====] - 55s 1ms/step - loss: 0.2496 - accuracy: 0.9076 - val_loss: 0.2806 - val_accuracy: 0.9043
Epoch 6/15
45000/45000 [=====] - 54s 1ms/step - loss: 0.2288 - accuracy: 0.9131 - val_loss: 0.2339 - val_accuracy: 0.9179
Epoch 7/15
45000/45000 [=====] - 54s 1ms/step - loss: 0.2210 - accuracy: 0.9163 - val_loss: 0.2354 - val_accuracy: 0.9188
Epoch 8/15
45000/45000 [=====] - 57s 1ms/step - loss: 0.2098 - accuracy: 0.9217 - val_loss: 0.2955 - val_accuracy: 0.8979
Epoch 9/15
45000/45000 [=====] - 64s 1ms/step - loss: 0.1968 - accuracy: 0.9276 - val_loss: 0.2549 - val_accuracy: 0.9159
Epoch 10/15
45000/45000 [=====] - 71s 2ms/step - loss: 0.1858 - accuracy: 0.9304 - val_loss: 0.2949 - val_accuracy: 0.9034
Epoch 11/15
45000/45000 [=====] - 58s 1ms/step - loss: 0.1777 - accuracy: 0.9330 - val_loss: 0.2601 - val_accuracy: 0.9165
Epoch 12/15
45000/45000 [=====] - 59s 1ms/step - loss: 0.1733 - accuracy: 0.9342 - val_loss: 0.2298 - val_accuracy: 0.9240
Epoch 13/15
45000/45000 [=====] - 57s 1ms/step - loss: 0.1634 - accuracy: 0.9377 - val_loss: 0.2471 - val_accuracy: 0.9195
Epoch 14/15
45000/45000 [=====] - 65s 1ms/step - loss: 0.1571 - accuracy: 0.9410 - val_loss: 0.2674 - val_accuracy: 0.9145
Epoch 15/15
45000/45000 [=====] - 59s 1ms/step - loss: 0.1499 - accuracy: 0.9437 - val_loss: 0.2894 - val_accuracy: 0.9112
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

[4] CNN 딥러닝 구현 결과에 대한 평가

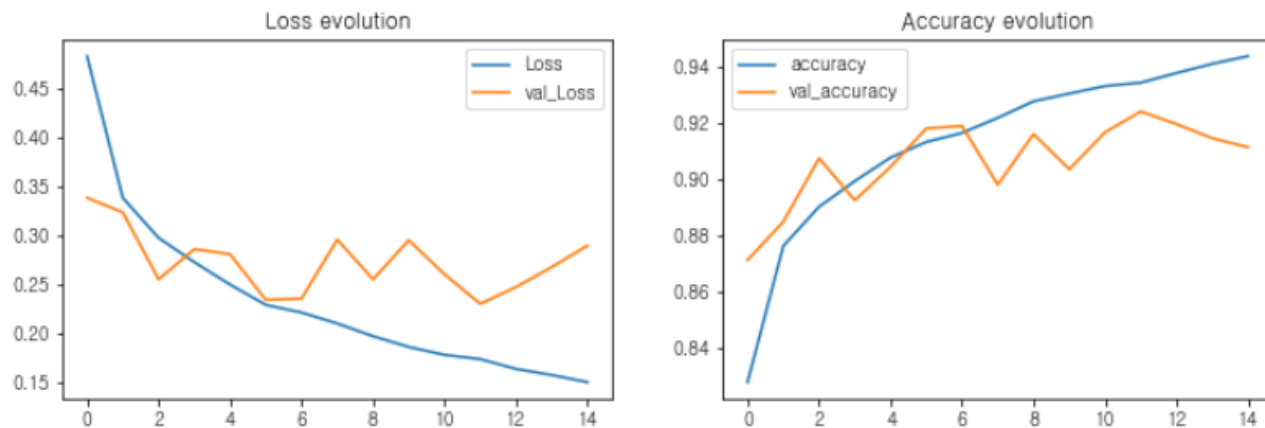
1. CNN 딥러닝 결과 시각화하기

```
In [37]: plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='val_Loss')
plt.legend()
plt.title('Loss evolution')

plt.subplot(2, 2, 2)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.title('Accuracy evolution')
```

Out [37]: Text(0.5, 1.0, 'Accuracy evolution')



CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

2. 테스트 데이터셋을 기반으로 CNN 딥러닝 모델 평가하기

• Accuracy 평가

```
In [38]: evaluation = cnn_model.evaluate(X_test, y_test)
print(f'Test Accuracy : {evaluation[1]:.3f}')
```

```
10000/10000 [=====] - 3s 256us/step
Test Accuracy : 0.922
```

• Test 이미지 데이터셋 1000개를 기반으로 패션 이미지 분류 예측이 정확히 되는지 테스트한 후, 테스트 결과 출력

0	1	2	3	4	5	6	7	8	9
티셔츠 / 상단	바지	풀오버	드레스	코트	샌들	셔츠	운동화	가방	발목 부츠
T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot

```
In [39]: # Test 데이터셋 확인 1000개 확인 및 테스트
prediction = cnn_model.predict(X_test) #예측은 10 개의 숫자로 구성된 배열(가장 큰 값이 선택됨)
print("테스트 데이터 개수 : ", len(prediction))
print(prediction)
```

```
테스트 데이터 개수 : 10000
[[9.9999559e-01 7.6663471e-18 1.4796736e-07 ... 4.9837509e-20
 7.5375338e-11 5.6453076e-19]
 [9.3838801e-25 1.0000000e+00 3.5306345e-24 ... 3.0756988e-36
 1.5641025e-21 1.4492006e-33]
 [1.0321854e-02 8.3469676e-14 9.5732683e-01 ... 2.2206283e-14
 4.4981952e-08 1.0565395e-16]
 ...
 [2.8791287e-22 2.4721660e-29 5.3975073e-20 ... 4.4471039e-21
 1.0000000e+00 5.8756209e-25]
 [8.9250456e-12 3.9015591e-18 1.8753290e-15 ... 3.1736806e-15
 1.0000000e+00 5.2473608e-17]
 [7.4149633e-09 9.9999893e-01 5.9997950e-07 ... 3.3434713e-18
 1.9434274e-10 1.1986846e-13]]
```


CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

- 1000개의 테스트 결과 중에서 첫번째 이미지 결과만 출력

```
In [40]: # Test 데이터셋 0번째 이미지의 예측 결과
print(prediction[0])

# 아래의 10개의 값 중에서 가장 큰 값을 가진 0번 인덱스에 해당하는 "T-shirt/top"으로 예측함
[9.9999559e-01 7.6663471e-18 1.4796736e-07 3.0473291e-10 5.3267658e-11
 3.2567081e-15 4.2422198e-06 4.9837509e-20 7.5375338e-11 5.6453076e-19]
```

- Test 이미지 데이터셋 1000개를 기반으로 패션 이미지 분류 예측이 정확히 되는지 테스트한 후, 클래스로 레이블링된 결과 출력

```
In [41]: predicted_classes = cnn_model.predict_classes(X_test) # Test 데이터셋 1000개 전체를 예측
print(predicted_classes)

[0 1 2 ... 8 8 1]
```

```
In [42]: predicted_classes
```

```
Out[42]: array([0, 1, 2, ..., 8, 8, 1], dtype=int64)
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가

- Test 이미지 데이터셋 1000개가 클래스로 레이블링된 결과 중에서 5x5 (25개)의 결과를 시각화

In [43]:

```
L = 5
W = 5

fig, axes = plt.subplots(L, W, figsize = (12,12))
axes = axes.ravel() #ravel()은 평면화된 배열을 반환

for i in np.arange(0, L * W): # Test 데이터셋 중에서 25개(0~24번) 이미지 출력
    axes[i].imshow(X_test[i].reshape(28,28))
    axes[i].set_title(f"예측 클래스 = {predicted_classes[i]} \n 실제 클래스 = {int(y_test[i])}")
    axes[i].axis('off')
plt.subplots_adjust(wspace=0.5)
```

CNN을 활용한 Fashion MNIST 딥러닝

1. CNN 소개

2. CNN을 활용한 Fashion MNIST 딥러닝

1) Fashion MNIST 이미지 데이터셋 준비 및 확인

2) 이미지 데이터셋 정규화 및 분포 확인

3) CNN 딥러닝 구현

4) CNN 딥러닝 구현 결과에 대한 평가



Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	