
SMARCLE 2021 WINTER STUDY

CNN

18 장윤정 , 20 김준수, 20 이유빈, 20 이은지

CONTENTS

1 CNN이란?

2 손글씨 인식 실습

3 CNN 적용하기

연결 구조 간단

합성곱 신경망

시각적 영상을
분석
다층의 피드-
포워드적

CNN이란?

Convolution
Neural Network

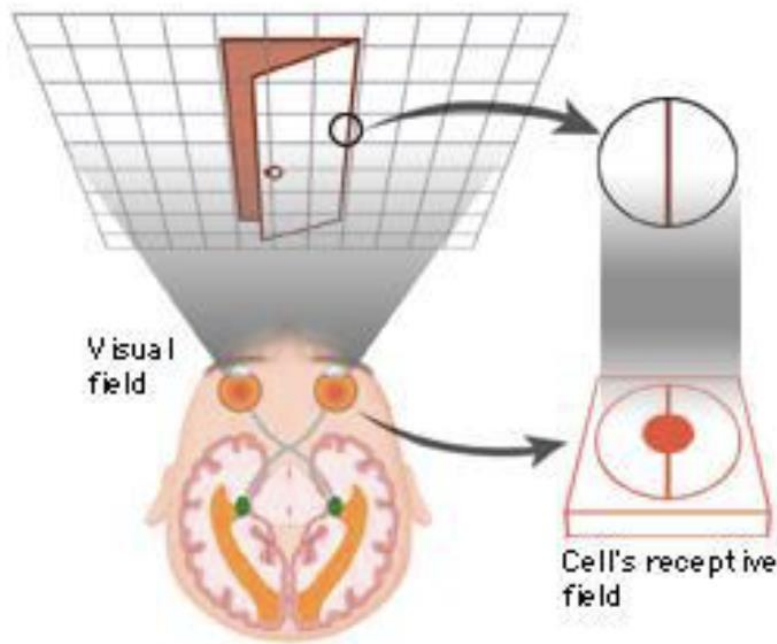
공간 불변 인공 신경망
(SIANN)

"행렬로 표현된 필터의 각 요소가 데이터
처리에 적합하도록 자동으로 학습되게 하자"

딥러닝에서 널리 사용되는 세 가지 이유

- 수동 특징 추출의 필요성 제거
- 고도로 정확한 인식 결과 생성
- 기존 신경망 활용 가능

CNN의 유래



뉴런들이 시야의 일부 범위 안에 있는 시각 자극에만 반응

수직선의 이미지에 반응

다른 각도의 선에 반응

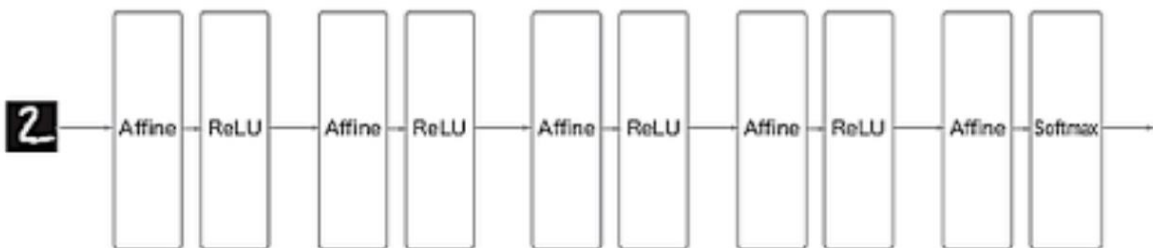
복잡한 패턴에 반응

1958 – 1959 David H. Hubel과 Torsten Wiesel

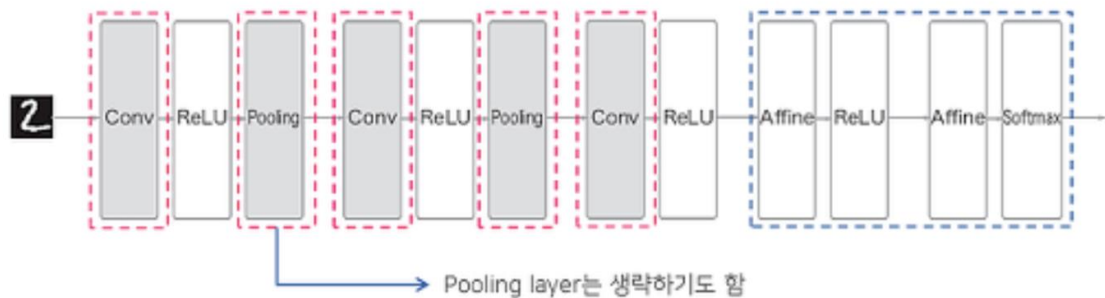


MNIST 이미지 데이터 셋

CNN의 구조

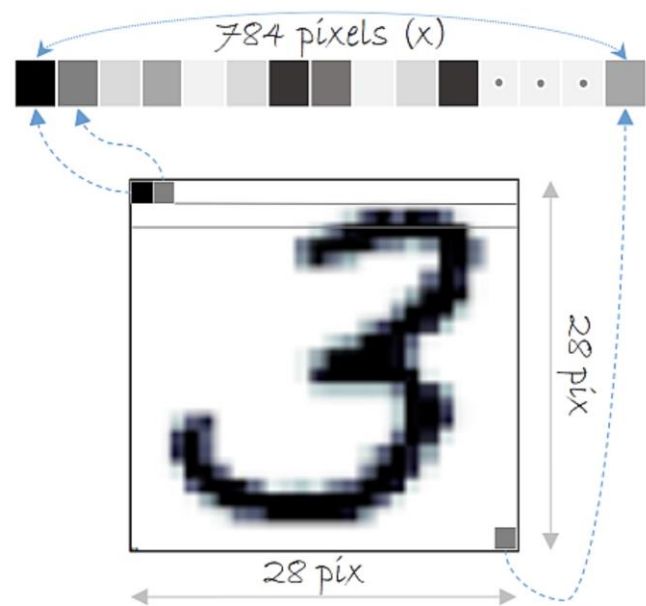


완전연결(fully connected)계층



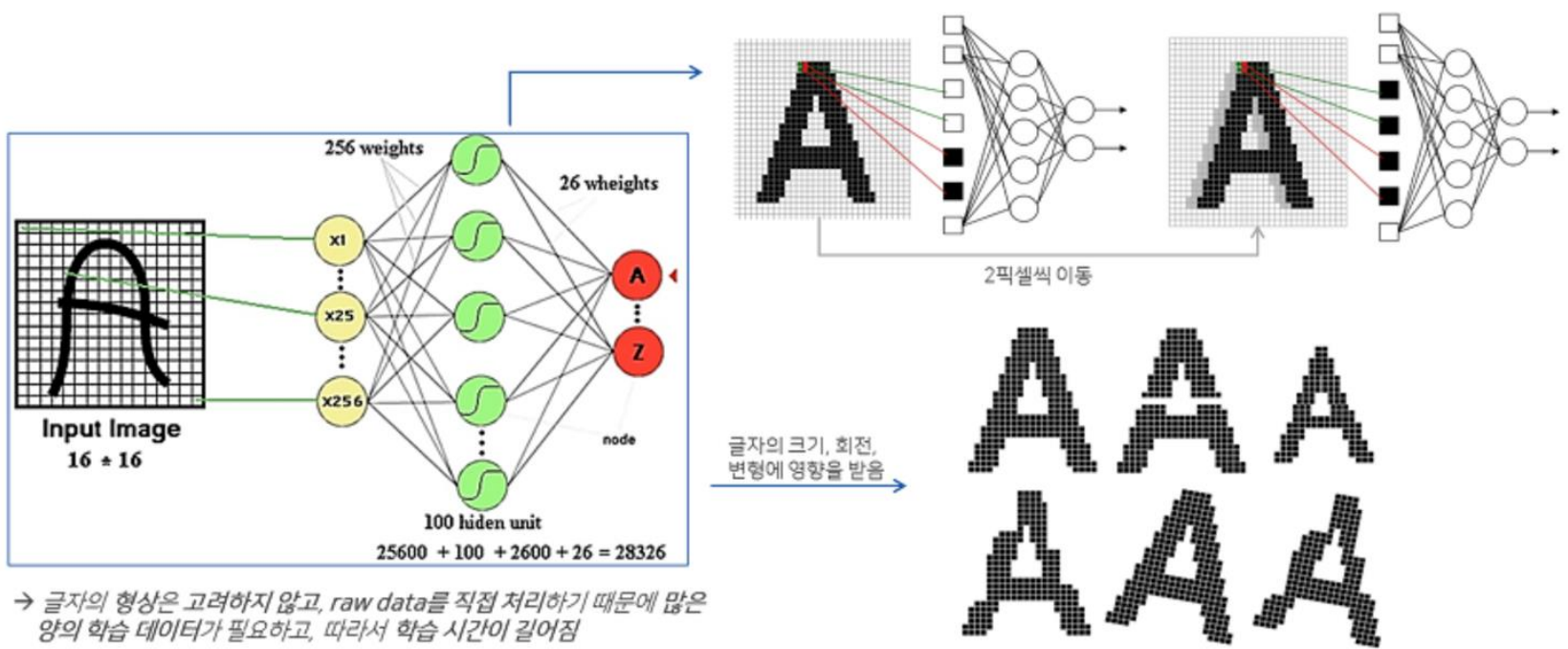
합성곱층(convolutional layer)과 풀링층(pooling layer)

완전연결 계층의 문제점



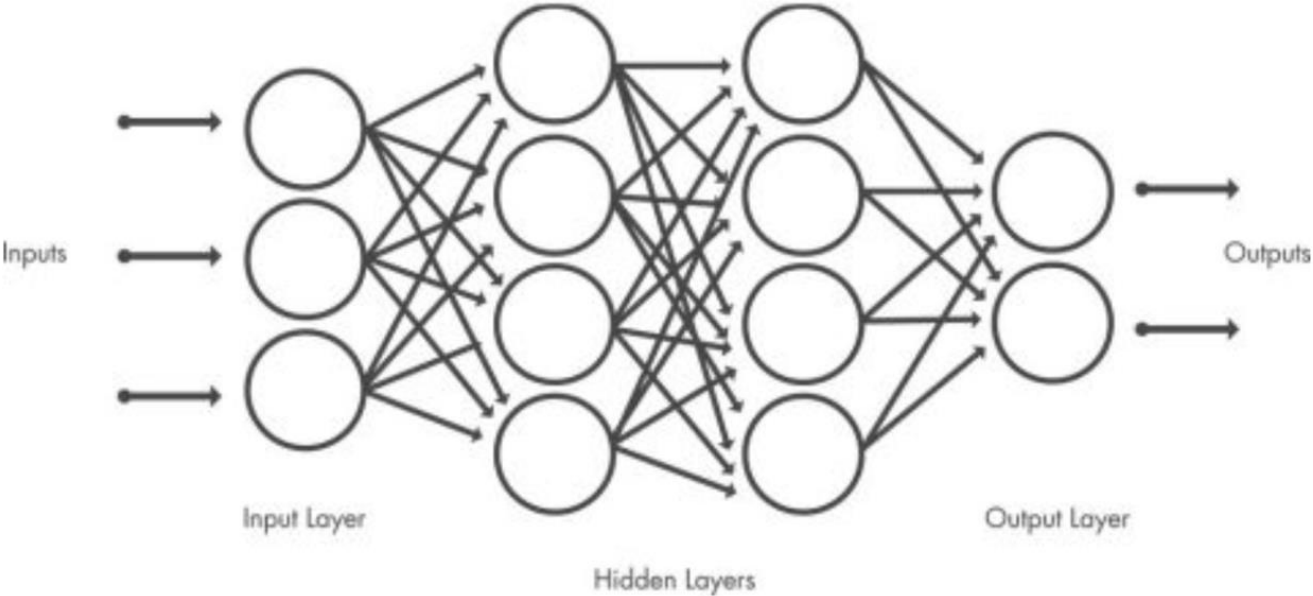
데이터의 형상 무시

MLNN의 문제점



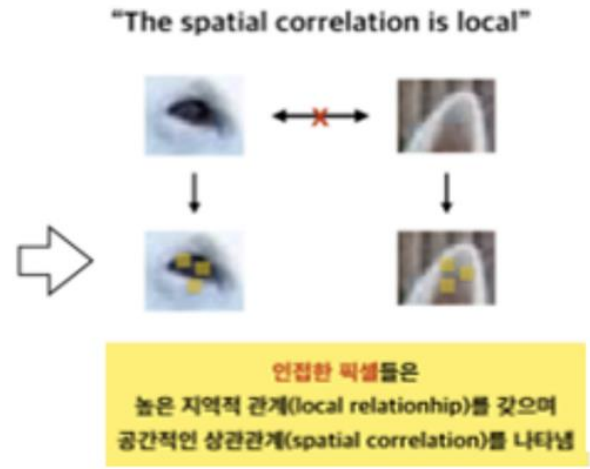
→ 글자의 형상은 고려하지 않고, raw data를 직접 처리하기 때문에 많은 양의 학습 데이터가 필요하고, 따라서 학습 시간이 길어짐

변수의 개수, 네트워크 크기, 학습시간
세가지 문제발생



입력 데이터의
형상을 유지

얼굴분류기



Q: "Which one is in the left picture,
Dog or Cat?"

A: "Dog!!"

 True: [0. 1. 0. 0.]	 True: [0. 0. 1. 0.]	 True: [0. 0. 0. 1.]
 True: [0. 0. 0. 1.]	 True: [0. 0. 0. 1.]	 True: [1. 0. 0. 0.]
 True: [1. 0. 0. 0.]	 True: [1. 0. 0. 0.]	 True: [0. 1. 0. 0.]

이미지넷

Apple

Fruit with red or yellow or green skin and sweet to tart crisp whitish flesh

1319 pictures

94.77% Popularity Percentile

Wordnet IDs

Numbers in brackets: (the number of synsets in the subtree).

ImageNet 2011 Fall Release (32326)

- plant, flora, plant life (4486)
 - geological formation, formation (17)
 - natural object (1112)
 - rock, stone (30)
 - asterism (0)
 - carpet (0)
 - black body, blackbody, full radiator (1)
 - radiator (1)
 - consolidation (0)
 - mechanism (12)
 - body, organic structure, physical (7)
 - nest (7)
 - plant part, plant structure (681)
 - corona (0)
 - button (0)
 - veil, velum (3)
 - pollen tube (0)
 - stock (1)
 - lobe (0)
 - mycelium (1)
 - domatium (0)
 - tendrils (1)

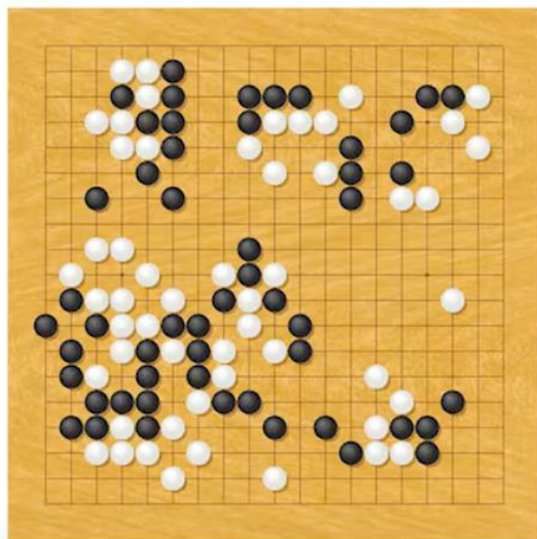
Treemap Visualization

Images of the Synset

Downloads

01

알파고



Neural
Network



Next move
(19 x 19 positions)

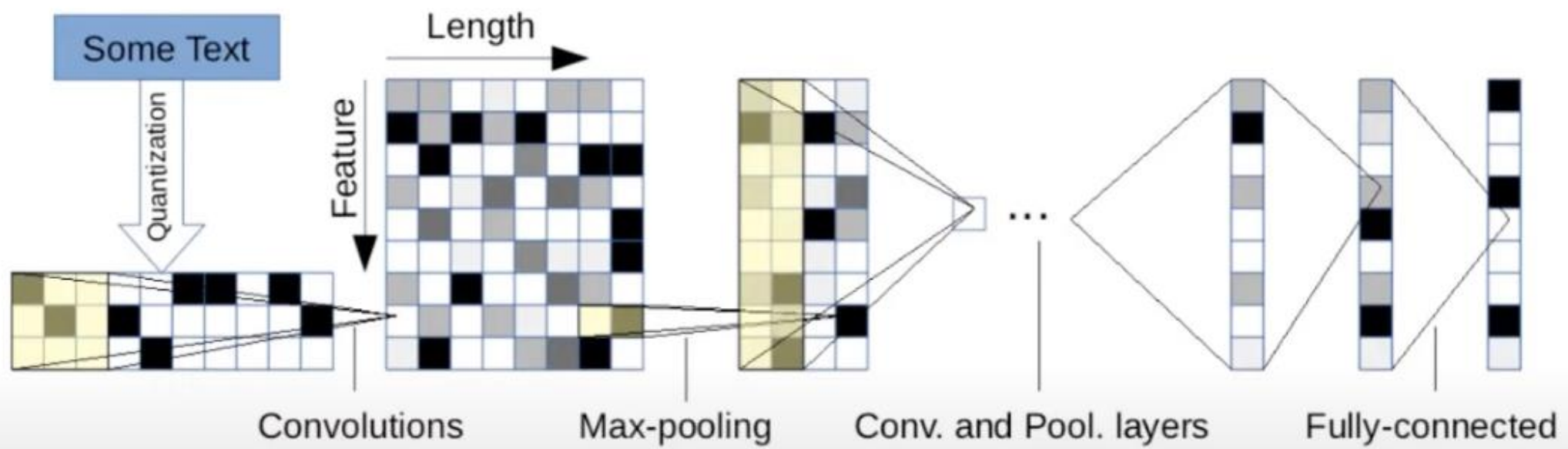
19 x 19 matrix

Black: 1

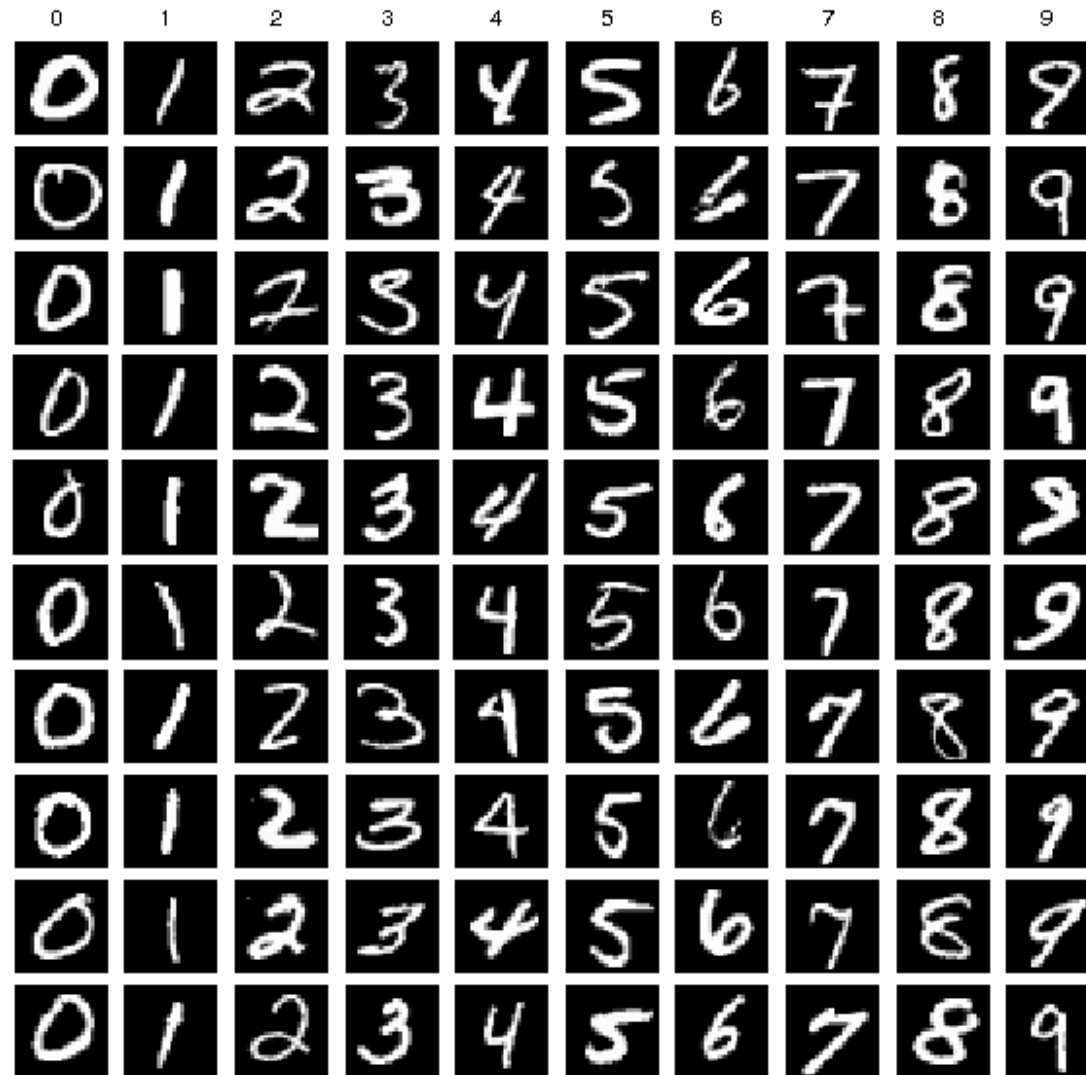
white: -1

none: 0

문서종류 분류



MNIST



03 CNN 적용하기

CNN(컨볼루션 신경망) 모델의 설정

```
model = Sequential()  
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=2))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(10, activation='softmax'))
```


03 CNN 적용하기

< 입력된 이미지 >

[illegible]

< 3 X 3 필터(=커널) >

x1	x0	x0
x0	x1	x0
x0	x0	x1

03 CNN 적용하기

0x1	1x1	1x1	1x0	0x0
1x1	0x1	0x1	1x0	0x0
0x1	0x1	1x1	0x0	0x0
0x0	1x0	0x0	0x1	0x0
1x0	1x0	1x0	1x0	1x1

컨볼루션(Convolution)

< 새롭게 만들어진 층 = 피쳐맵(Feature Map) >




1	1	2
1	1	0
2	1	2

03 CNN 적용하기


< 여러가지 필터 >


1	1	2
---	---	---



수직선 검출


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






수평선 검출


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






Box blur

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11





Sharpen

0	-1	0
-1	5	-1
0	-1	0




그림 6.3 다양한 필터를 적용했을 때의 컨볼루션 연산의 결과³

2	2	1
---	---	---

03 CNN 적용하기

Conv2D() 함수 : 케라스에서 컨볼루션 층을 추가하는 함수

```
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))
```

- 첫 번째 인자 : 필터를 몇 개 적용할지 지정
- Kernel_size = (행, 열) : 필터의 크기 지정
- input_shape = (행, 열, 색상 또는 흑백) : 맨 처음 층에 입력되는 값을 알려주는 역할
- activation : 활성화 함수 지정

03 CNN 적용하기

CNN(컨볼루션 신경망) 모델의 설정

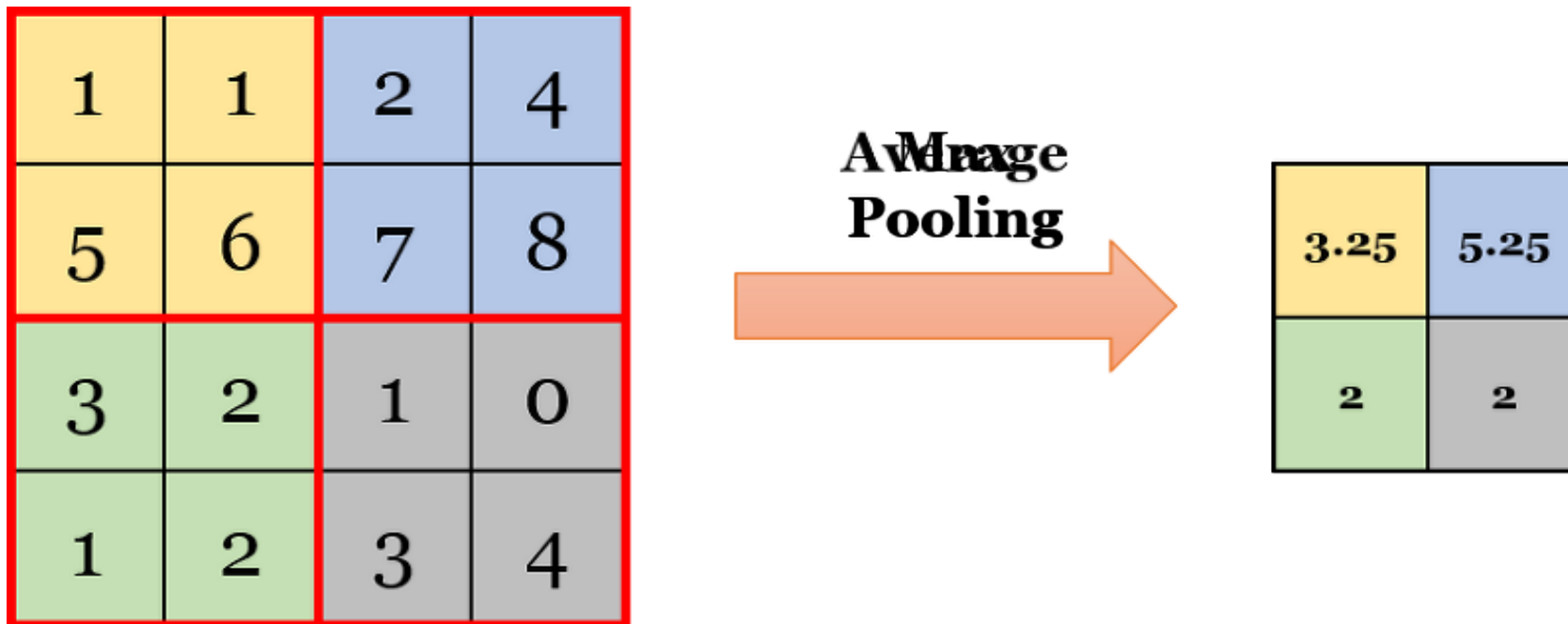
```
model = Sequential()  
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=2))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(10, activation='softmax'))
```

03 CNN 적용하기

풀링(=서브 샘플링) : 컨볼루션의 결과가 여전히 클 때 한 번 더 축소해주는 과정

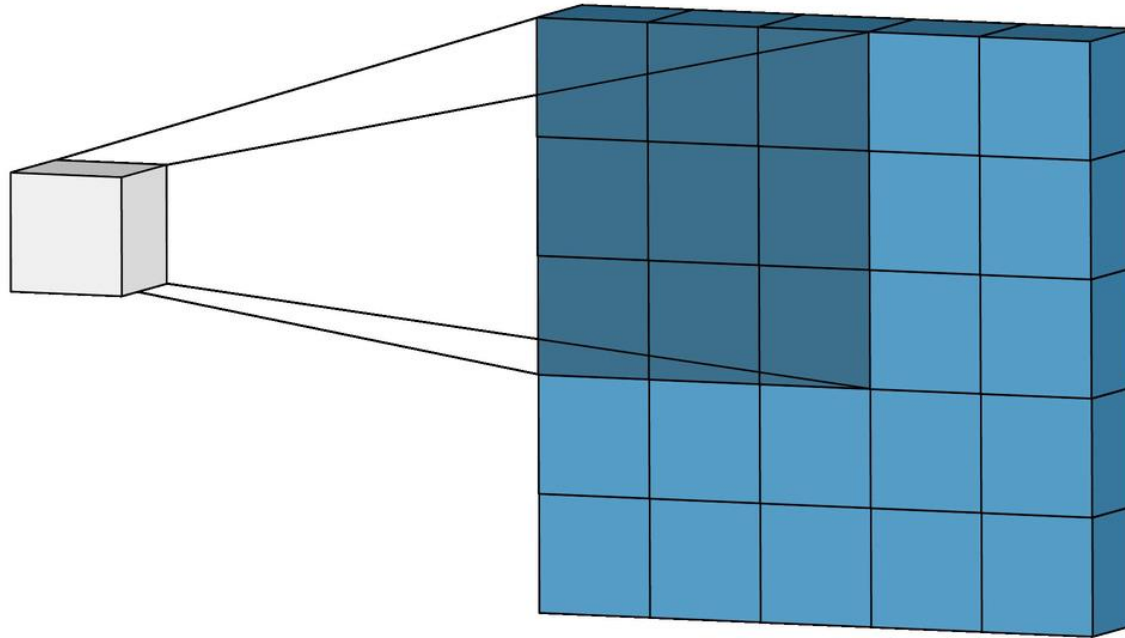
```
model.add(MaxPooling2D(pool_size=2))
```

- pool_size : 풀링 창의 크기 지정, 2로 설정 시 2X2 사각형으로 연산 -> 결과값이 절반으로 축소
- 풀링레이어는 가중치가 존재하지 않으므로 학습되지 않으며 생략 되기도 함



03 CNN 적용하기

스트라이드(stride) : 컨볼루션 필터가 합성곱 연산을 수행하기 위해 이동하는 일정한 이동량



03 CNN 적용하기

스트라이드(stride) : 컨볼루션 필터가 합성곱 연산을 수행하기 위해 이동하는 일정한 이동량

stride 2

0x1	1x0	1x1	1x0	0x0
1x0	0x1	0x0	1x1	0x0
0x1	0x0	1x1	0x0	0x0
0x0	1x1	0x0	0x1	0x0
1x0	1x0	1x0	1x0	1x1



1	2
2	2

03 CNN 적용하기

패딩(Padding) : 출력 데이터의 크기를 조정하기 위해 사용

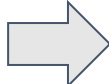
5 X 5 입력된 이미지 -> 제로패딩 7 x 7

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

+

< 3 X 3 필터 >

x1	x0	x0
x0	x1	x0
x0	x0	x1



5 X 5 출력 데이터(=피쳐맵)

0	1	2	1	0
1	1	1	2	1
1	1	1	0	1
1	2	1	2	0
1	1	2	1	1

03 CNN 적용하기

CNN(컨볼루션 신경망) 모델의 설정

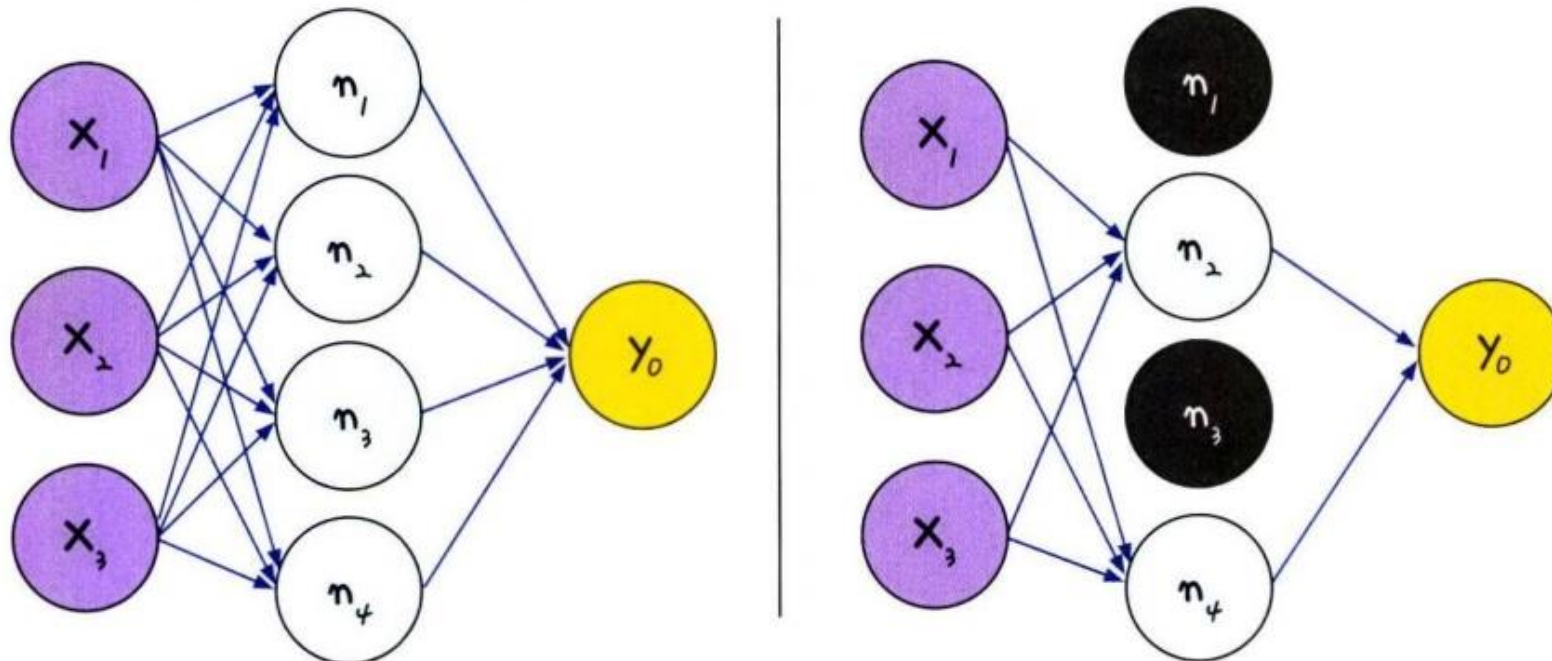
```
model = Sequential()  
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=2))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(10, activation='softmax'))
```

03 CNN 적용하기

드롭아웃 : 은닉층에 배치된 노드 중 일부를 임의로 끄므로써 과적합 방지

```
model.add(Dropout(0.25))
```

● 25%의 노드를 끄

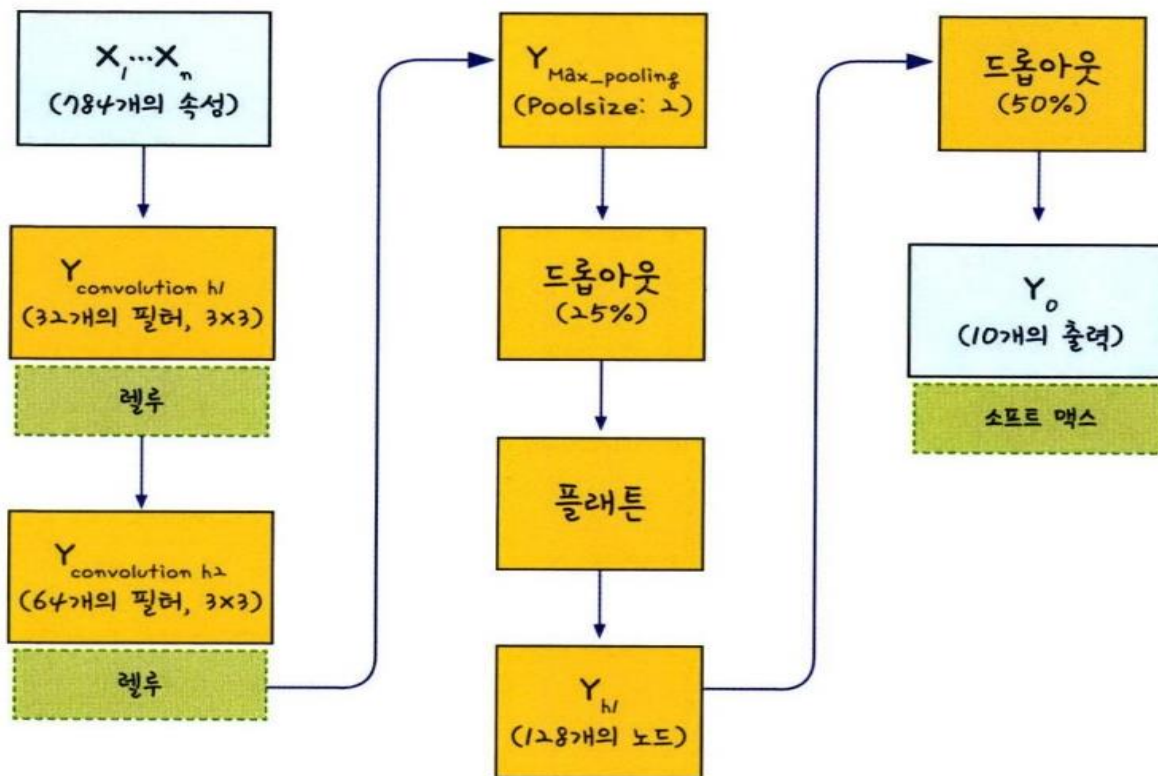


03 CNN 적용하기

플래튼 : 2차원 배열을 1차원 배열로 바꿔주는 함수

```
model.add(Flatten())
```

- 컨볼루션 층, 맥스풀링 층은 2차원 배열이기 때문에
추출된 특징들을 기본층에 넣어서 분류하려면 1차원으로 바꿔 줄 필요성



03 CNN 적용하기

CNN(컨볼루션 신경망) 모델의 설정

```
model = Sequential()  
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(28, 28, 1), activation='relu'))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=2))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(10, activation='softmax'))
```

03 CNN 적용하기

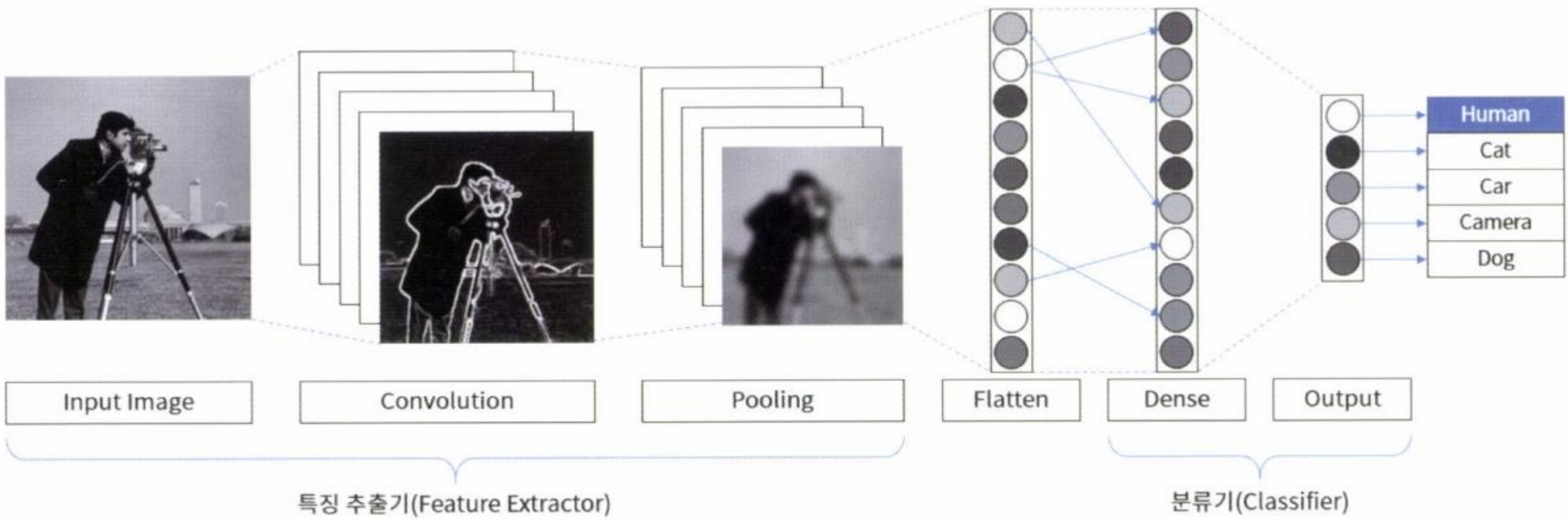


그림 6.5 이미지 분류에 사용되는 컨볼루션 신경망의 구조⁶

Thank you
