

[딥러닝 4단계] 합성곱 신경망 네트워크(CNN)

합성곱 신경망

개념

윤곽선 검출 `edge detection`

패딩 `padding`

스트라이드 합성곱 `strided convolution`

데이터가 3D일 때

구현

합성곱 층

Pooling 층

Fully connected 층

질문

합성곱 신경망

Convolutional Neural Networks

개념

- 필요성
 - 컴퓨터 비전 `Computer Vision` 분야에서는 다루는 데이터가 이미지라서 크기 ↑
⇒ 합성곱 연산을 사용하여 가중치의 개수를 줄임.
- ▼ 이유

- 변수 공유 → 필터 하나 가지고 이미지 여기저기랑 여러 번 곱함. 그럼 필터 하나만큼의 변수(=가중치)만 있으면 되잖아.
- 희소 연결 → Fully connected와 반대되는 개념. 이미지 픽셀 1개는 필터 원소 1개와 곱해지고, 같은 필터의 다른 원소들이랑은 곱해지지 않음.
- 합성곱을 쓰면 이미지를 몇 픽셀 이동해도 예측 결과가 동일함.
- 사용 분야
 - 컴퓨터 비전
 - 근데 여기서 아이디어 얻어서 음성 인식 같은 다른 분야에서도 쓰임.



[요약]

▼ 합성곱 연산 **convolution operation**

→ 사진 * 필터 = output

여기서 * 연산.

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

*

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

=

-5			

$$3 \times 1 + 0 \times 0 + 1 \times -1 + 1 \times 1 + 5 \times 0 + 8 \times -1 + 2 \times 1 + 7 \times 0 + 2 \times -1 = -5$$

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

*

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

=

-5	-4		

$$0 \times 1 + 1 \times 0 + 2 \times -1 + 5 \times 1 + 8 \times 0 + 9 \times -1 + 7 \times 1 + 2 \times 0 + 5 \times -1 = -4$$

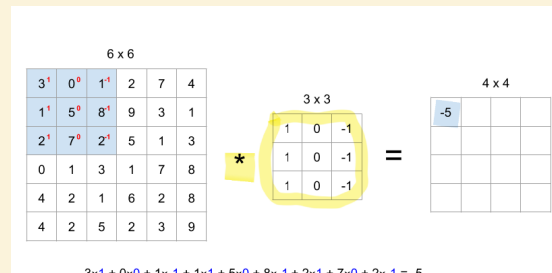
...

▼ tmi

사실 이걸 수학적으로 따지면 **교차상관** **cross correlations** (합성곱은 아니고 약간 다른 것)인데,

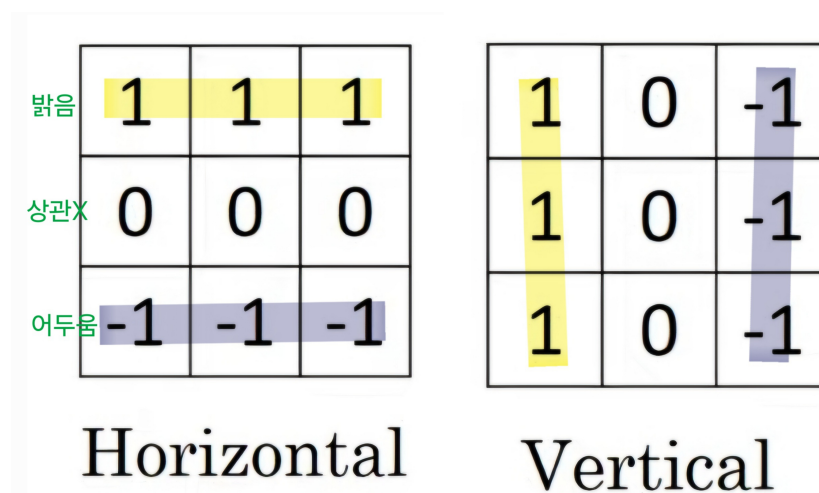
딥러닝 분야에선 관습적으로 합성곱이라고 부름.

▼ 필터 = 커널 **kernel**



윤곽선 검출 **edge detection**

- 필터를 조절해서 사진의 윤곽선(사진의 일부만 나타내는 특징)을 뽑아낼 수 있다
- 용어
 - 수직 윤곽선 → 사진에서 윤곽선 중 수직 방향의 성분
 - 수평 윤곽선
- 필터의 기본 형태



▼ ex) 수직 윤곽선

여기서 수직 윤곽선 → 노란 형광펜

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

 $*$

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

 $=$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

▼ ex) 수직 윤곽선

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

→ 밝은 데서 → 어두운 쪽으로

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

감지된 윤곽선 : 밝음

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

→ 어두운 데서 → 밝은 쪽으로

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

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

감지된 윤곽선 : 어두움

▼ ex) 수평 윤곽선

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

6x6

 $*$

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

 $=$

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

- 초록 : 밝은 쪽(위) → 어두운 쪽(아래)이라 양수
- 보라 : 반대라서 음수
- 노랑 : 1,2번째 열은 양수인데 3번째 열이 음수라서 플마 하면 작은 양수
- 다른 필터들

1	0	-1	3	0	-3
2	0	-2	10	0	-10
1	0	-1	3	0	-3
sobel			scharr		

- 장점 : 가운데 픽셀에 더 큰 가중치를 둠 → 더 선명한 output
- 최근 딥러닝에서는
임의의 숫자로 초기화하고 → 학습시켜서
해당 문제에 적합한 필터를 만들

패딩 padding

- 필요성
 - 패딩이 없으면 합성곱 연산을 할 때마다 이미지가 계속 축소됨
 - 패딩이 없으면 이미지 가장자리의 픽셀은 단 한번만 사용됨. 정보를 버리게 됨.
- 해결방법
 - 이미지 테두리에 숫자 0으로 채운 패딩을 추가함.

스트라이드 합성곱 **strided convolution**

- 스트라이드 크기
 - 필터가 한 번 이동할 때 몇 칸 움직이는지
 - 지금까지의 방법은 $s = 1$ 이었던 것



[요약]

- 최종 이미지 크기
 - $\rightarrow \lfloor \frac{n+2p-f}{s} \rfloor + 1$
 - n : 이미지 크기
 - p : 패딩 크기
 - f : 필터 크기
 - 보통 홀수
 - ▼ 이유
 1. 짝수면 패딩이 비대칭
 2. 홀수여야 중심위치가 존재함.
 - s : 스트라이드
 - 보통은 필터 크기가 정수가 되도록 p, s 를 정함.

데이터가 3D일 때

2D	흑백 이미지
3D	컬러 이미지

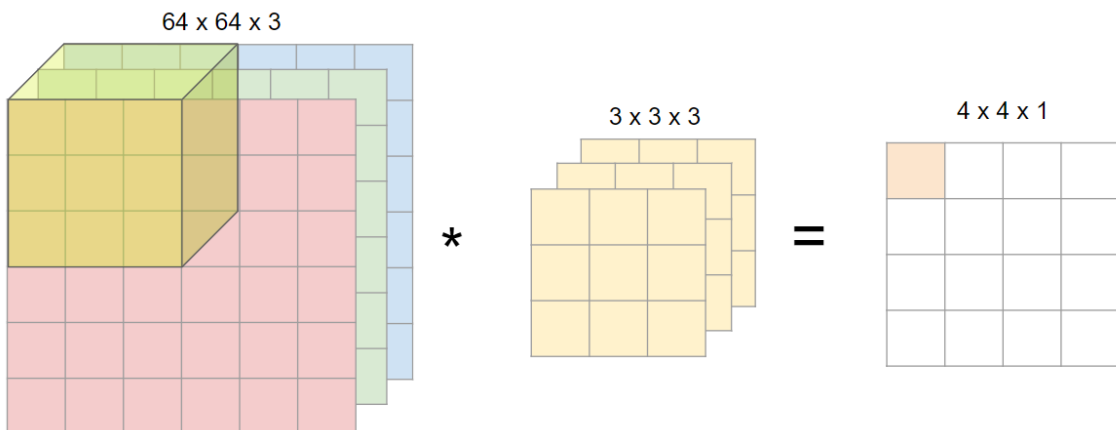
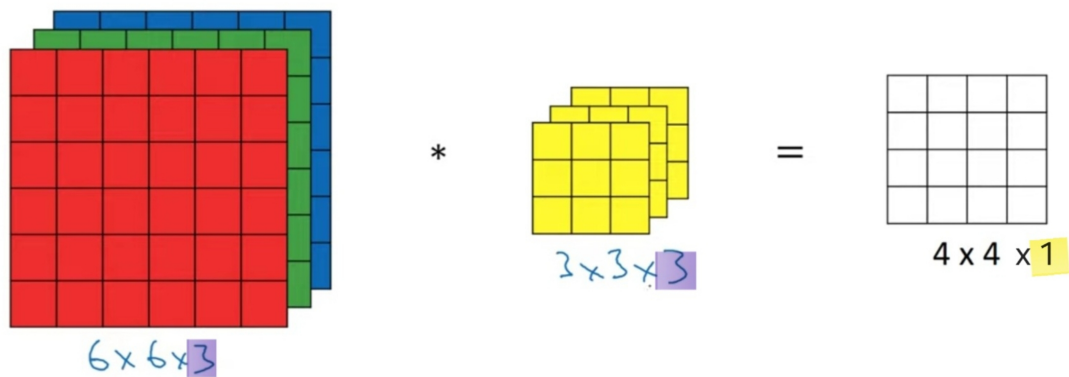
★ [요약]

- 채널 수
 - 채널 = 3차원에서 깊이, 높이

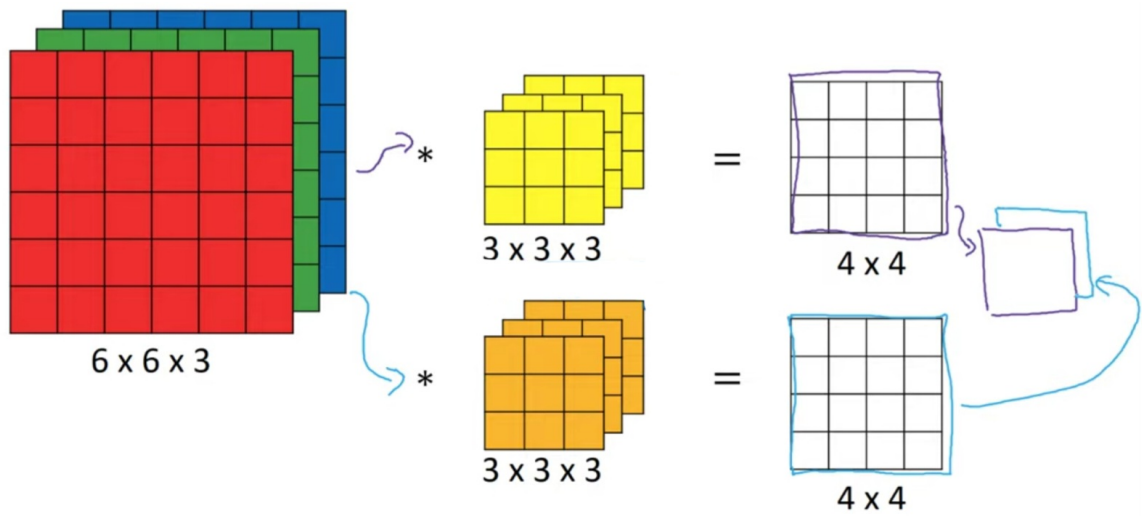
이미지	필터	output
컬러 (RGB → 3)	이미지와 동일	사용한 필터 개수

- 필터 개수
 - = 탐지한 feature 개수
 - = 윤곽선 종류 개수 (ex 수직, 수평, 7°, 36°, ...)

▼ ex) 필터 1개 썼을 때



▼ ex) 필터 여러 개 썼을 때



하나는 수직 윤곽선,
 하나는 수평 윤곽선,
 하나는 대각선
 ...
 이런 식으로 쓸 수 있음.

구현



[요약]

- layer 종류 → 애네를 조립해서 전체 신경망 만들.

Convolution	(CONV)	합성곱 층
Pooling	(POOL)	풀링 층
Fully connected	(FC)	

- 출력층
 - layer 여럿 거치면서 3차원 output이 나왔겠지.
 - 그걸 1차원 벡터로 짝 펴.
 - 그리고 softmax / logistic 회귀 함수에 넣어.
 - 그 결과 → predict한 값 (최종 output)
 - 어떻게 조립할지
 - 직접 정하지 말고 문헌 중에 내 문제에 잘 작동할 것 같은 거 갖다 쓰기
- ▼ 흔한 ex)
- CONV1 - POOL1
 - CONV2 - POOL2
 - FC3 - FC4 등 FC 몇 개
 - SOFTMAX

합성곱 층

- 특징 feature 을 추출하는 역할
- 연산 순서 (layer 1개 기준)
 - 합성곱 연산 → 편향 추가 → 활성화 함수
- 합성곱 층 거듭할수록 이런 경향 有
 - height / width → 작아짐

- #channel → 커짐

▼ 가중치 개수가 정말 줄었을까?

- 표기법
 - l : l 번째 계층
 - $f^{[l]}$: 필터의 크기
 - $p^{[l]}$: 패딩의 양
 - $s^{[l]}$: 스트라이드 크기
 - n_H : 이미지의 높이
 - n_W : 이미지의 넓이
 - n_c : 채널의 수
- 합성곱 썼을 때
 - 한 layer 당 가중치 개수
= 모든 필터의 원소 개수
 - $(f^{[l]} \times f^{[l]} \times n_c^{[l-1]}) \times n_c^{[l]}$
(필터 1개 크기) × (필터 개수)
 - 추가된 편향까지 더하면 $+n_c^{[l]}$
- 합성곱 안 썼을 때
 - 한 layer 당 가중치 개수
= 가중치행렬 $W^{[l]}$ 의 크기
 - $(n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}) \times (n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]})$

Pooling 층

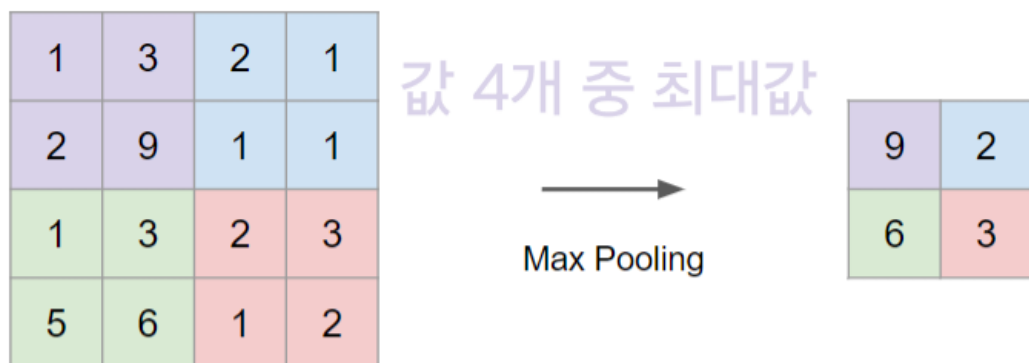
- 가중치 개수를 줄이는 역할

- 효과
 - height / width(표현의 크기)를 줄임
→ 계산 시간을 줄이고, 특징을 더 잘 검출해낼 수 있음.

- 종류 (2가지)

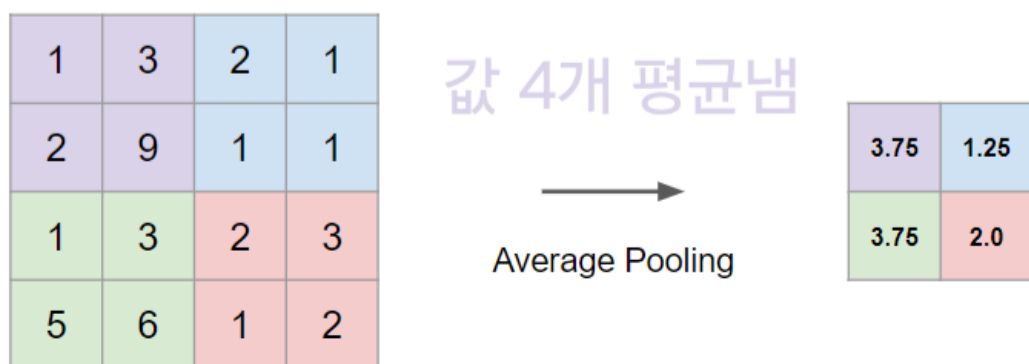
▼ Max Pooling ← 보통 이거 사용

이유 : 최댓값만 남김 → 특징을 더 잘 남김



합성곱 층으로 치면 $f = 2$, $s = 2$, $p = 0$ 인데, 필터를 가지고 합성곱 연산 대신 Max/Average 연산을 하는 셈

▼ Average Pooling



- layer 개수 셀 때
 - 가중치 있는 층만 셀 거라서, Pooling 층은 안 셈.

- 즉, 직전 합성곱 층 + 풀링 층 → 하나의 layer로 간주.

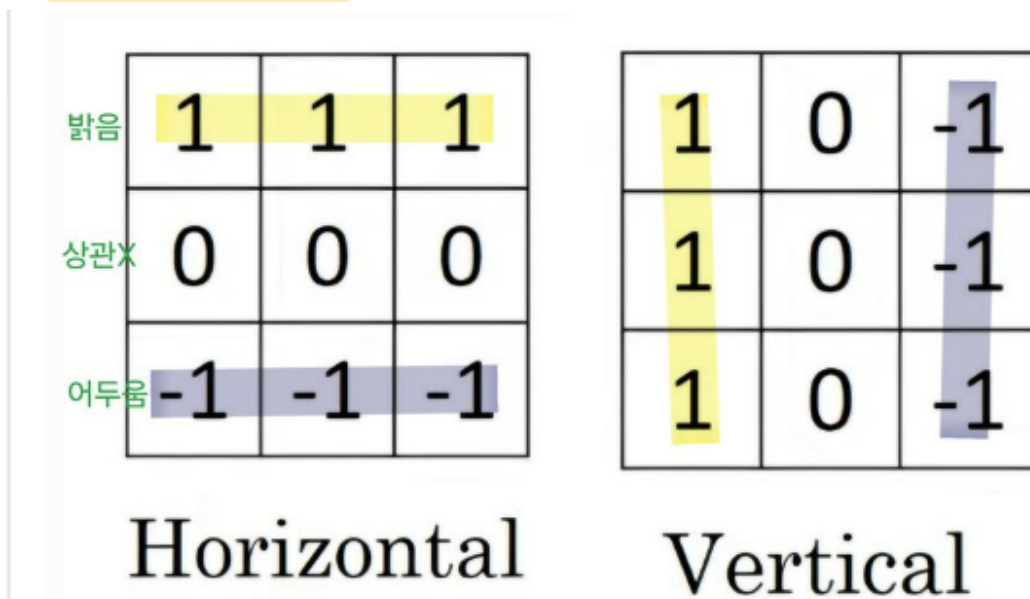
Fully connected 층

- 출력을 분류하는 역할

질문

- 왜 수직이 저런 형태?

- 필터의 기본 형태



- 퀴즈

✕ 4. 합성곱 신경망에서 완전 연결 층의 역할은 무엇인가요? *

0/1

☒ 이미지의 특징을 추출하는 역할

✕

☐ 출력을 분류하는 역할

☐ 필터를 공유하는 역할

정답

☒ 출력을 분류하는 역할