제5장: 세가지기본신경망II: CNN

Soyoung Park

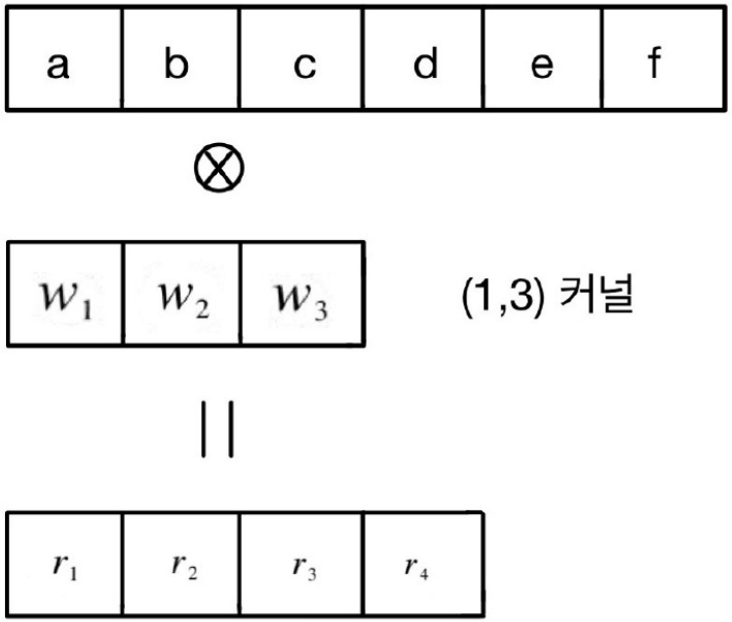
Pusan National University
Department of Statistics

soyoung@pusan.ac.kr

Convolutional Neural Network

- CNN모형의 설계는 MLP의 일반화 모형
-
- CNN도 MLP모형과 마찬가지로 입력층, 은닉층, 출력층을 연결하는 선형결합과 활성함수로 구성됨
-
- 1D, 2D, 3D convolution은 1D, 2D, 3D 선형결합함수

1D convolution



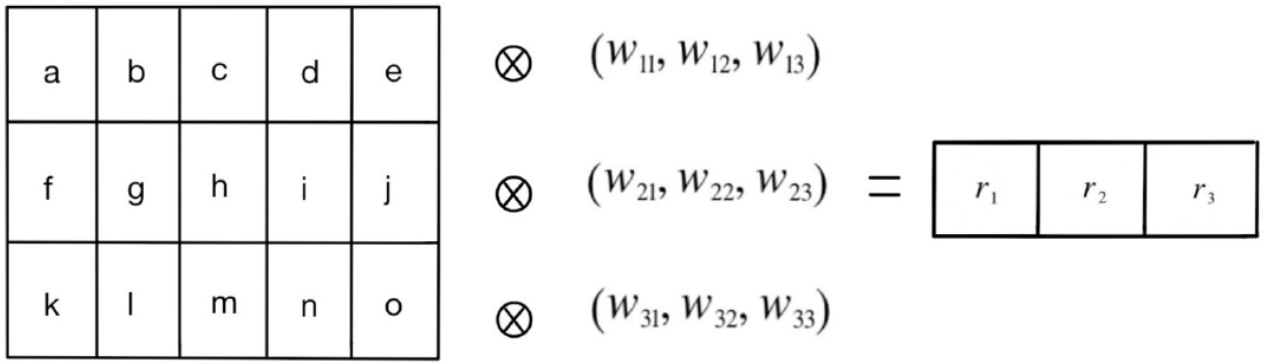
$$r_1 = w_1 a + w_2 b + w_3 c,$$
$$r_2 = w_1 b + w_2 c + w_3 d,$$
$$r_3 = w_1 c + w_2 d + w_3 e,$$
$$r_4 = w_1 d + w_2 e + w_3 f$$

그림 2–6 (1,3) 커널에 의한 1D convolution

1D convolution

- {a,b,c,d,f,e} 6개로 구성된 특성변수 입력
- 모수 w_1, w_2, w_3로 구성된 (1,3) 커널을 한칸씩 오른쪽으로 움직이면서
- r_1, r_2, r_3, r_4로 표기된 선형결합과 크기가 (1,4)인 1D텐서 출력
-
- 6개의 특성변수가 입력되었으므로, 출력도 6개로 만들고 싶다면 입력 양 끝에 0을 추가하면 크기가 (1,6)인 1D텐서를 출력 $\rightarrow$ padding

2D텐서에 1D convolution 적용



$$r_1 = w_{11}a + w_{12}b + w_{13}c + w_{21}f + w_{22}g + w_{23}h + w_{31}k + w_{32}l + w_{33}m$$

$$r_2 = w_{11}b + w_{12}c + w_{13}d + w_{21}g + w_{22}h + w_{23}i + w_{31}l + w_{32}m + w_{33}n$$

$$r_3 = w_{11}c + w_{12}d + w_{13}e + w_{21}h + w_{22}i + w_{23}j + w_{31}m + w_{32}n + w_{33}o$$

그림 2–7 2D텐서 자료에 대한 1D convolution의 적용

2D텐서에 1D convolution 적용

- {a,b,c,d,f,e, .., o} (3,5)인 2D텐서 특성변수 입력
- 9개의 모수 $w_{11}, w_{12}, w_{13}, ..., w_{33}$로 (1,3)커널 3개를 사용
- r_1, r_2, r_3로 표기된 선형결합과 크기가 (1,4)인 1D텐서 출력
- MLP에서는 15개의 모수가 필요했지만, 1D convolution은 9개의 모수만 사용
- MLP모형에서 다음 layer의 노드수만큼 선형결합을 만들었던것 처럼, CNN에서도 1D커널을 노드수 만큼 정의하여 노드수만큼 1D텐서를 출력한다 $\rightarrow$ 결과적으로 2D텐서 출력
-

2D텐서에 1D convolution 적용

- 그림 2-7에서 필터수가 10개이면 크기가 (1,3)인 1D텐서를 10개 출력함. 따라서 9개의 모수가 총 10쌍이 필요하므로 총 90개의 모수가 필요함
- MLP였다면, 특성변수가 15개이므로 15개의 모수가 10쌍이 필요하여 총 150개의 모수가 필요하지만 1D convolution을 사용하여 모수 60개를 줄임.
- 최종적으로 (1,3) 1D텐서 r_1, r_2, r_3에 bias b를 더해주고, 활성함수 적용한 후, 다음 layer로 이동

2D convolution

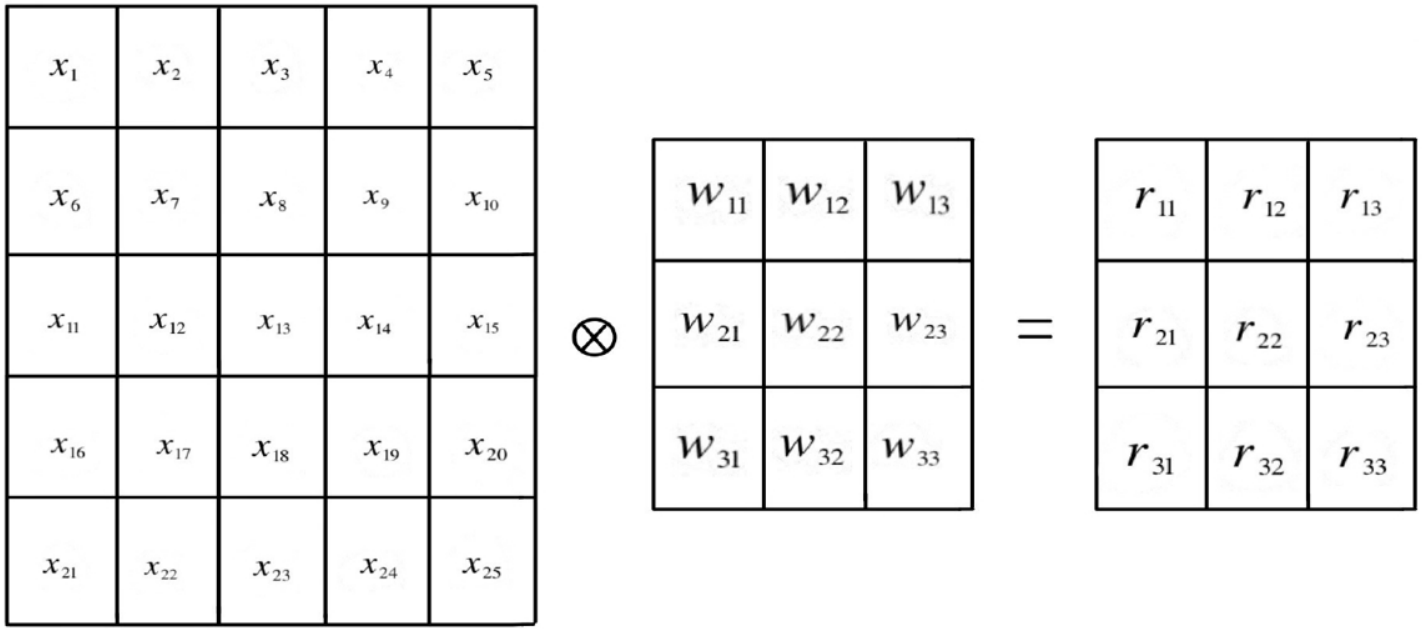


그림 2–8 (3,3) 커널을 이용한 2D convolution의 적용

$$r_{23} = w_{11}x_8 + w_{12}x_9 + w_{13}x_{10} + w_{21}x_{13} + w_{22}x_{14} + w_{23}x_{15} + w_{31}x_{18} + w_{32}x_{19} + w_{33}x_{20}$$

2D convolution

- 입력특성변수는 2D텐서이상, 출력은 2D텐서
- 그림 2-8은 입력특성변수가 5×5 2D텐서이고, (3,3) 커널을 적용한 예시
- 출력인 r_{ij}는 5×5 텐서에 대해 좌에서 우로, 위에서 아래로 한칸씩 이동하면서 (3,3)커널을 곱해 산출한 선형 결합임
- 5×5 2D 입력텐서와 동일한 5×5 선형결합을 출력하기 위해 좌우 끝 열과 상하 끝 행에 0을 padding
- stride를 2로하여 2칸씩 이동하여 선형결합을 구하면, 2×2 2D텐서를 출력
- 일반적으로 $stride = m$을 쓰면 입력자료의 크기를 1/m으로 축소시키는 역할

2D convolution

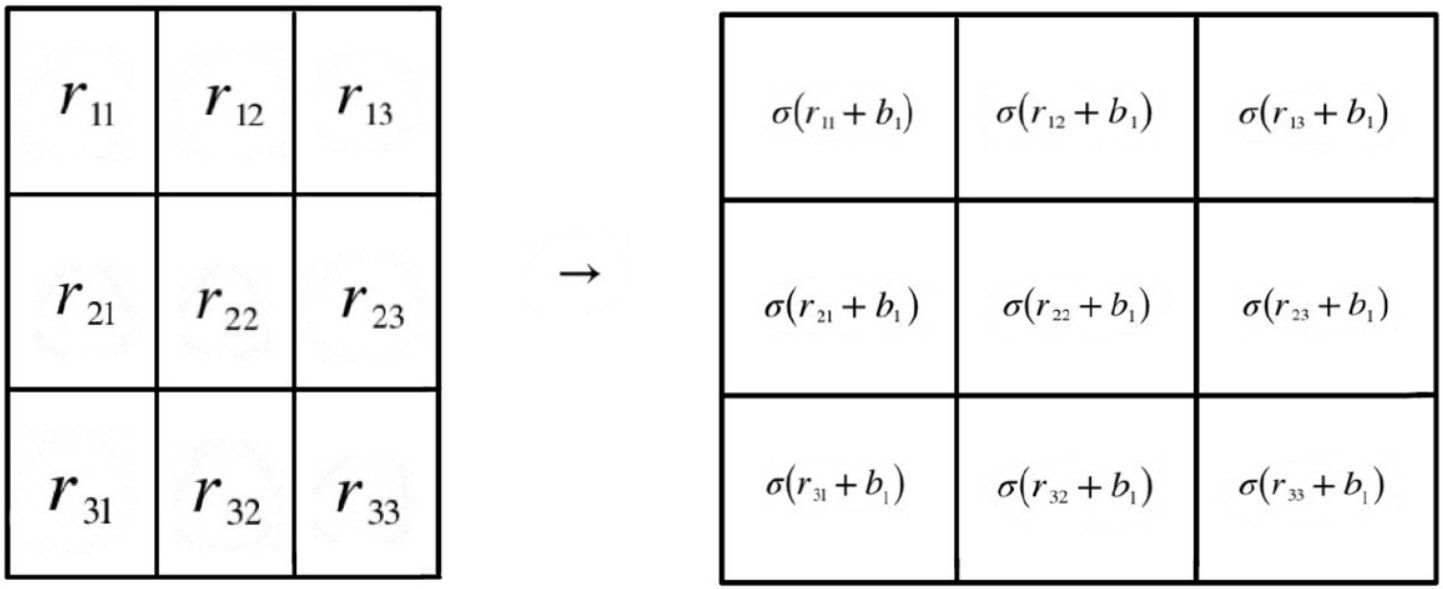


그림 2–9 〈그림 2–8〉의 출력

2D convolution

- 그림 2-8에서 필터를 10개 사용하면, (3,3) 커널이 10개 필요
- 그림 2-9와 같이 bias와 활성함수를 적용한 3×3 2D텐서를 출력
- 이러한 출력이 10개 있으므로, (3,3,10)인 3D텐서가 출력되며 이 3D 텐서가 다음 층의 입력특성변수가 된다.
- 그림 2-8의 예제에서, MLP모형이라면 25개의 모수가 필요하고, 1D convolution은 15개, 2D convolution은 9개의 모수가 필요
- MLP는 하나의 선형결합, 1D convolution은 1D텐서, 2D convolution은 2D텐서를 출력하므로 정보의 압축정도는 텐서가 증가할수록 낮아지는 효과!

3D텐서자료에 대한 2D convolution의 적용

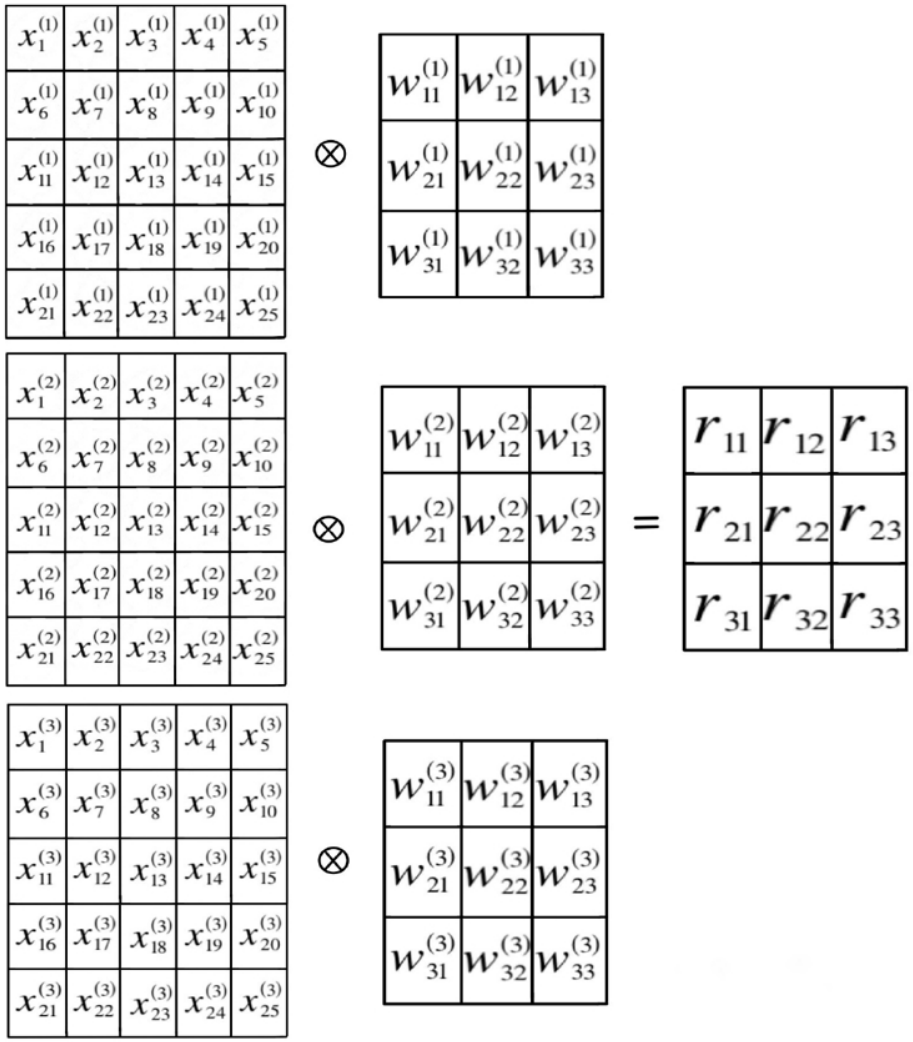


그림 2–10 3D텐서자료에 대한 2D convolution의 적용

3D텐서자료에 대한 2D convolution의 적용

- 입력자료가 컬러이미지와 같이 (h,w,3)인 3D텐서일 때, 2D convolution의 원리는 그림 2-10과 같음

$$r_{31} = \sum_{k=1}^{3} (w_{11}^{(k)} x_{11} + w_{12}^{(k)} x_{12} + w_{13}^{(k)} x_{13} + w_{21}^{(k)} x_{16} + w_{22}^{(k)} x_{17} + w_{23}^{(k)} x_{18} + w_{31}^{(k)} x_{21} + w_{32}^{(k)} x_{22} + w_{33}^{(k)} x_{23})$$

3D convolution

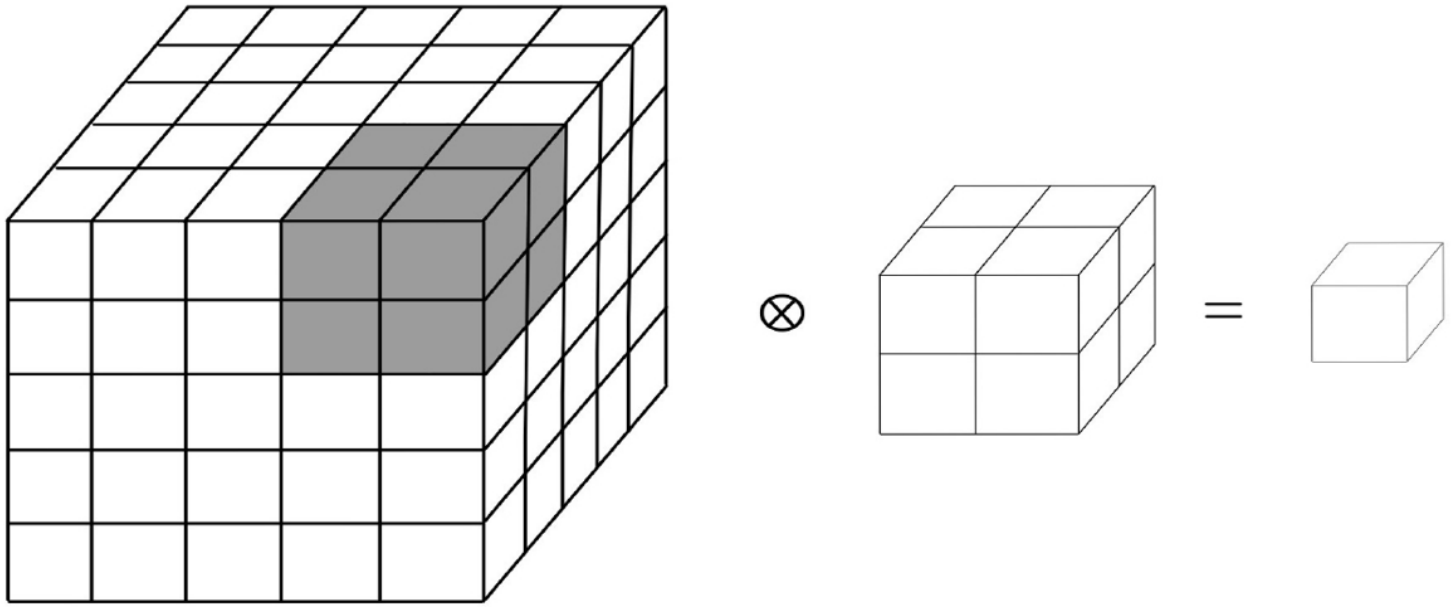


그림 2–12 3D텐서 입력자료에 대한 (2,2,2) 커널을 이용한 3D convolution

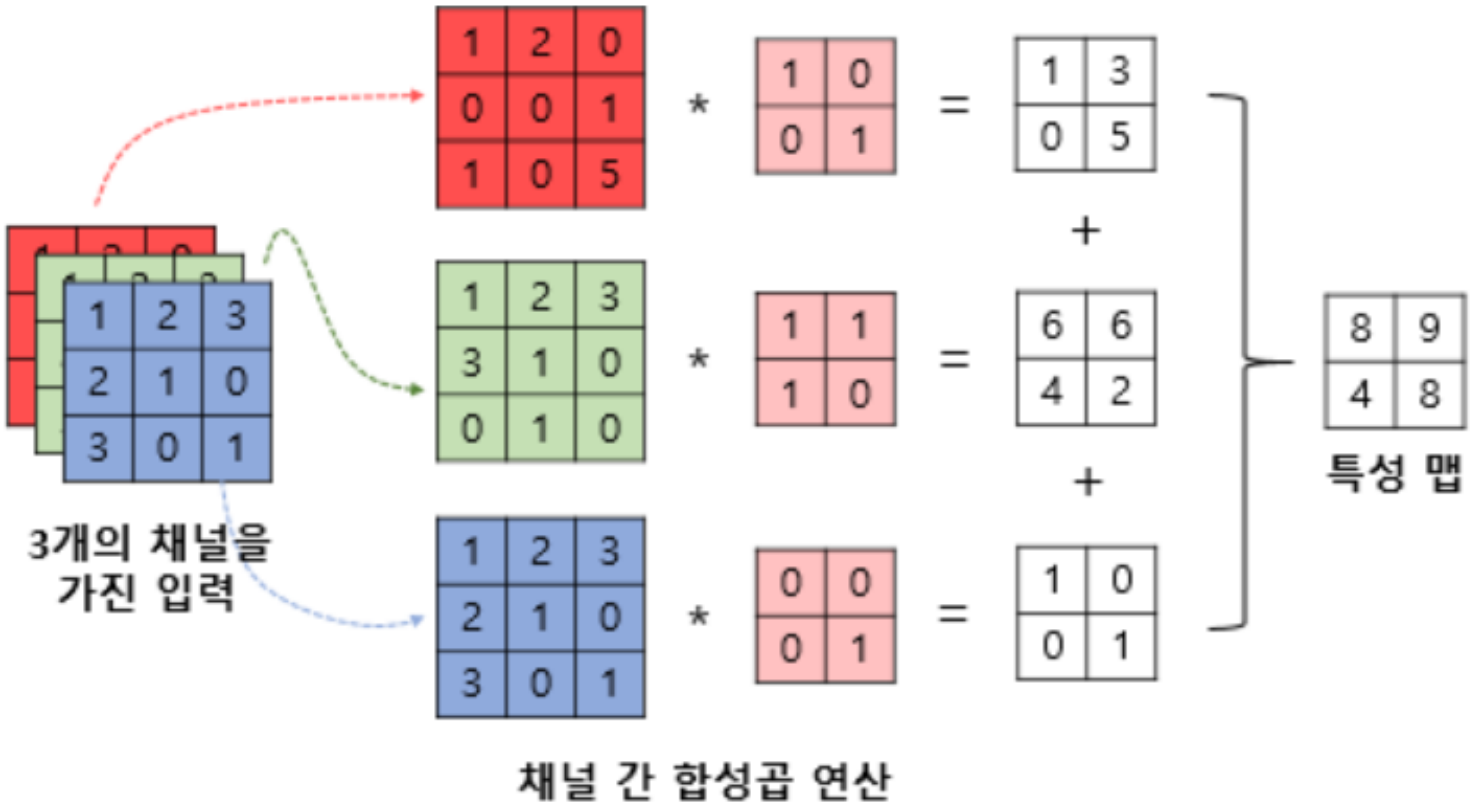
- 입력특성변수는 3D텐서 이상이며 이를 3D텐서 선형결합해야하므로 커널은 3D텐서 모수로 구성됨

3D convolution

- 입력자료가 컬러이미지와 같이 (h,w,3)인 3D텐서일 때, 2D convolution의 원리는 그림 2-10과 같음

$$r_{31} = \sum_{k=1}^{3} (w_{11}^{(k)} x_{11} + w_{12}^{(k)} x_{12} + w_{13}^{(k)} x_{13} + w_{21}^{(k)} x_{16} + w_{22}^{(k)} x_{17} + w_{23}^{(k)} x_{18} + w_{31}^{(k)} x_{21} + w_{32}^{(k)} x_{22} + w_{33}^{(k)} x_{23})$$

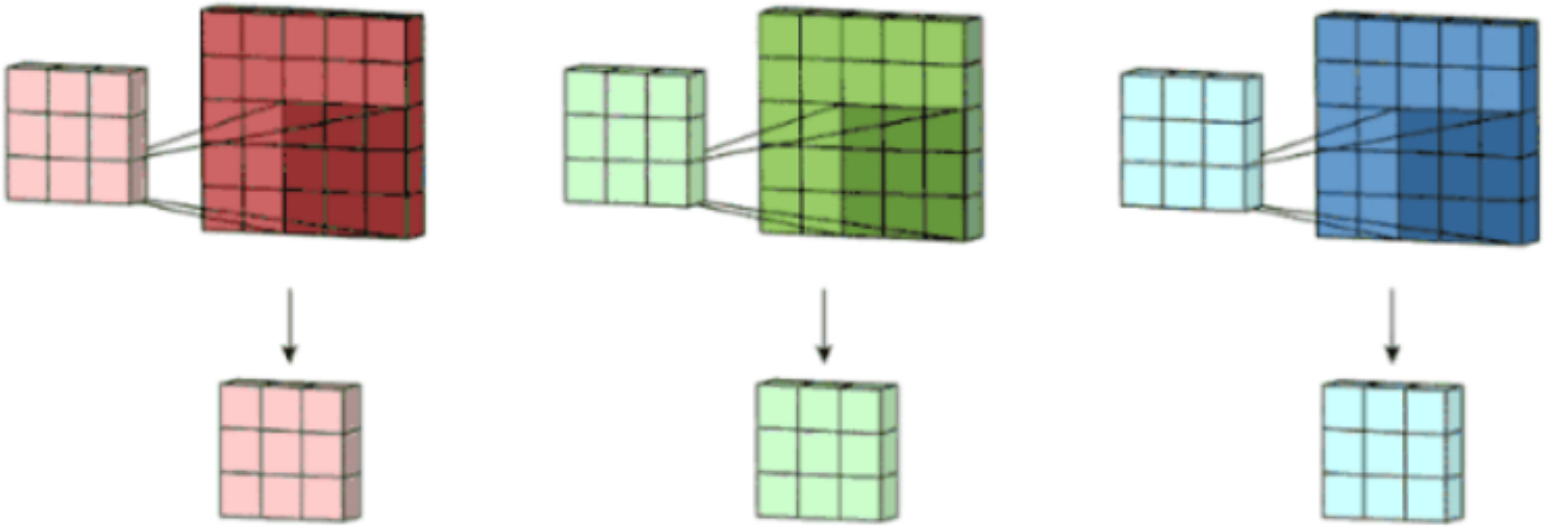
3D convolution



1

[1] https://blog.naver.com/neogates/222434251795

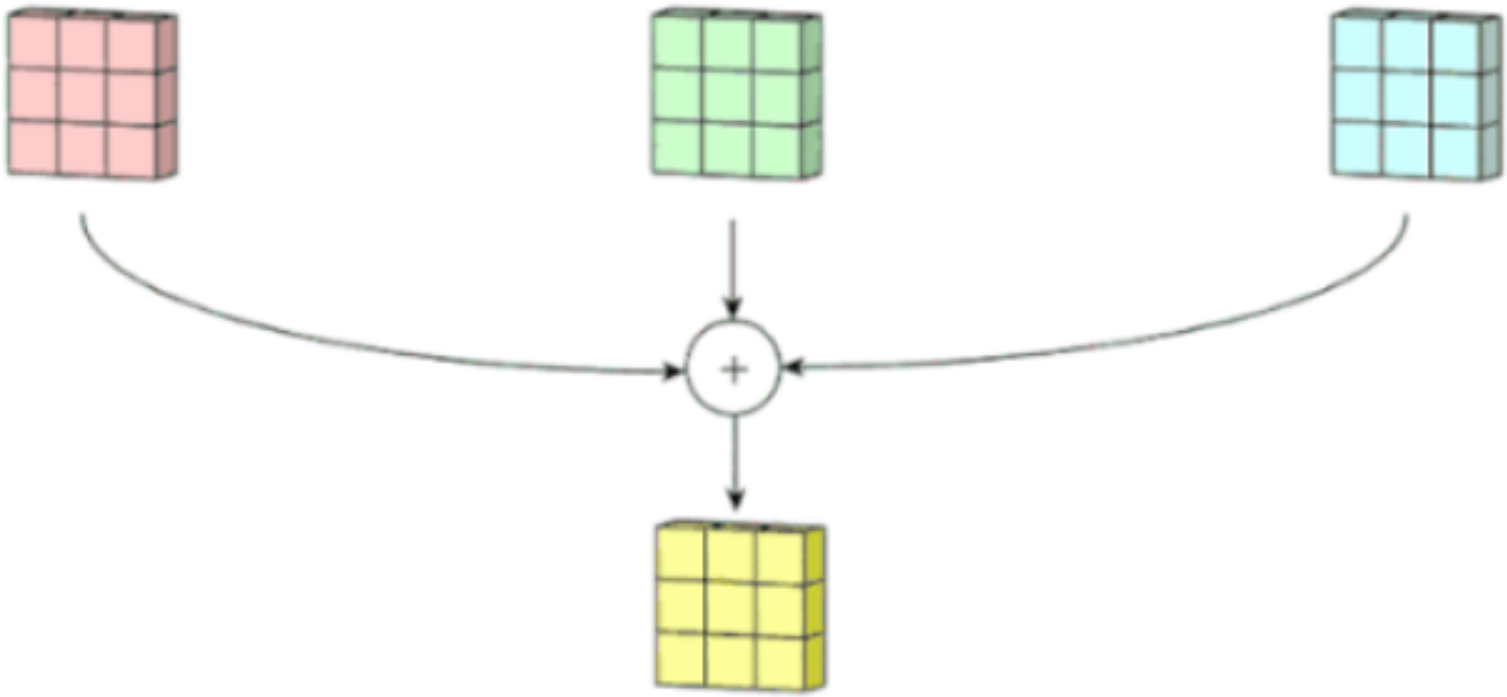
3D convolution



2

[2] https://blog.naver.com/neogates/222434251795

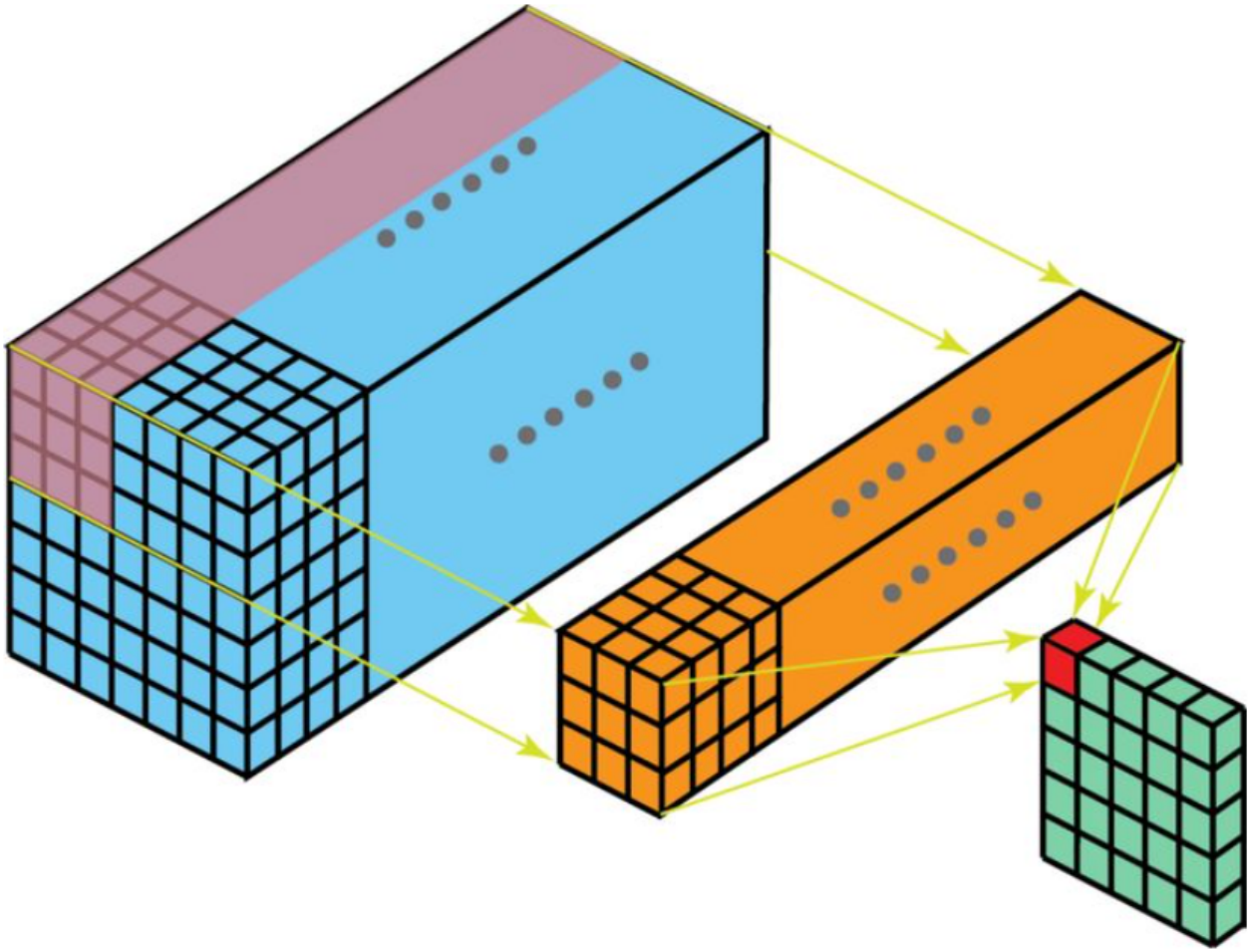
3D convolution



3

[3] https://blog.naver.com/neogates/222434251795

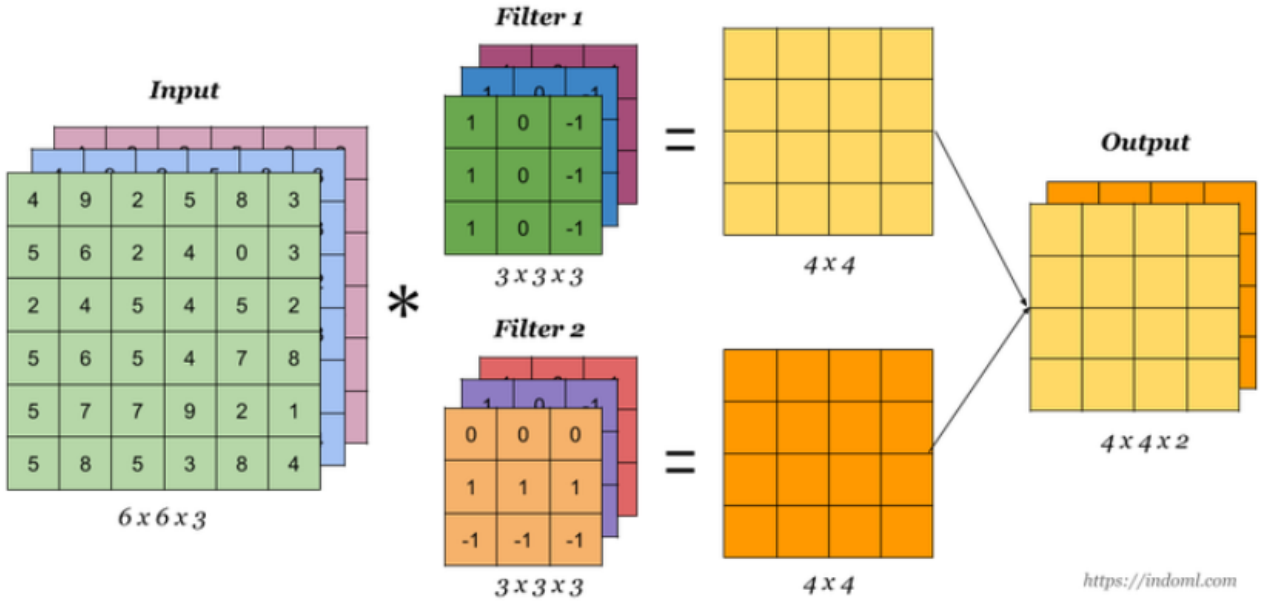
3D convolution



4

3D convolution

- 입력 데이터의 채널이 깊어지면 위의 그림과 같이 filter의 길이도 확장되어 convolution연산을 하는데,
-

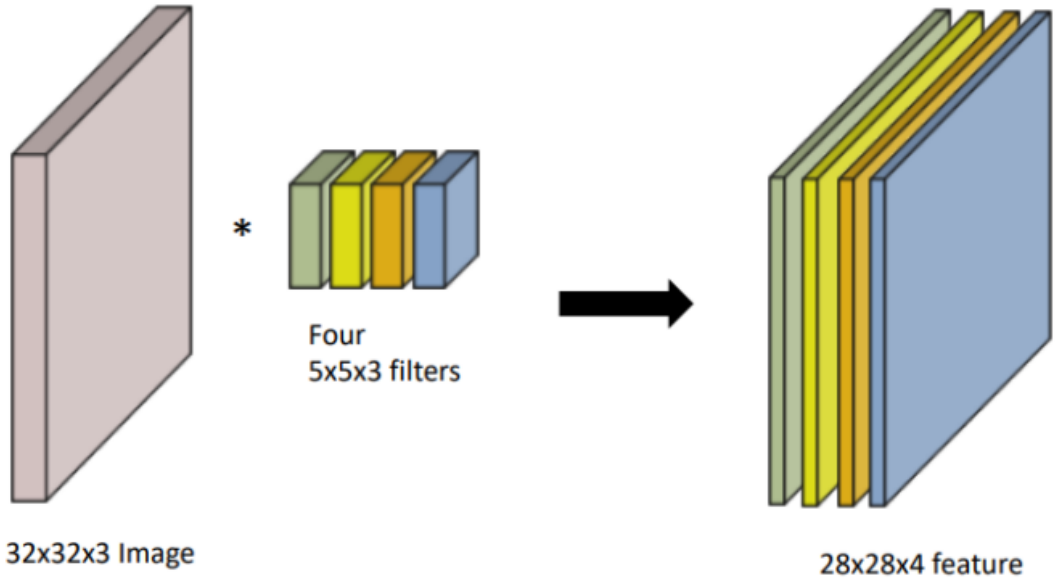


5

[5] https://blog.naver.com/neogates/222434251795

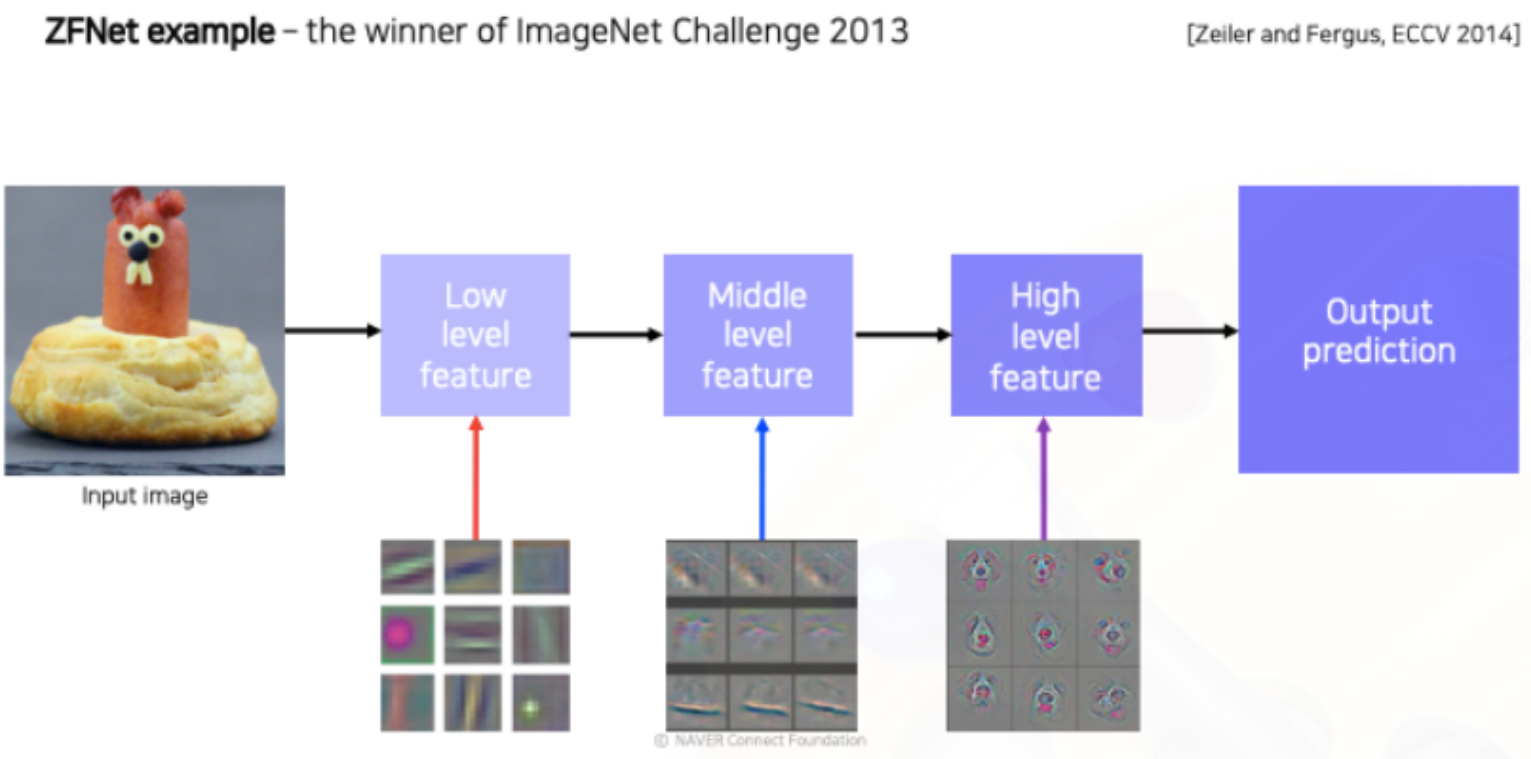
3D convolution

-
- 즉 N 개의 필터는 N 개의 채널을 가진 출력을 만든다. 각 채널의 연산은 독립적으로 이루어지며 그 결과도 독립적이지만, 그 결과물을 적층시켜 $W \times H \times C$ 형태의 3D Tensor 로 생성한다.



6

CNN 연산과정의 시각화



7

[7] https://blog.naver.com/neogates/222434251795

CNN 연산과정의 시각화

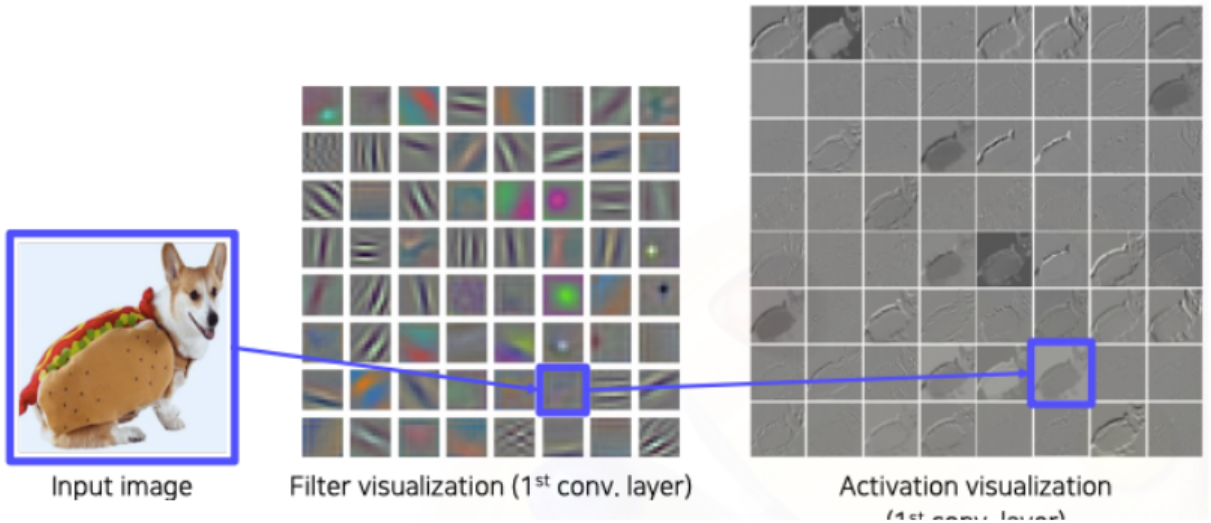
- 중간단계의 tensor를 convolution의 역연산인 deconvolution을 통해 시각화가 가능
-
- 중간계층과 더 나아가 높은 계층으로 갈수록 점점 high level에 의미가 있는 표현을 학습

8

[8] https://blog.naver.com/neogates/222434251795

Filter의 시각화

- Filter 역시 시각화 하여 사람이 직접 설계한 filter가 아닌 학습된 필터는 어떠한 형태로 생성되는지. 해당 필터는 입력 데이터의 어떤 특징을 끄집어 내는지도 확인이 가능



9

[9] https://blog.naver.com/neogates/222434251795

Filter 선택

- 알고리즘 설계자는 이 부분에서 채널의 사이즈를 적합하게 선정해야 하는데, 채널(필터의 갯수)을 너무 작게 잡으면 입력 데이터의 특징을 모두 잡아낼 수 없으며 채널을 너무 많이 잡으면 불필요한 연산을 하게 된다.
-
- convolution 연산에서는 filter의 사이즈를 receptive filed (자극이 특정 감각뉴런의 반응으로 이어지는 영역)로 취급
- 3x3 크기의 filter를 사용한다면 딥러닝 모델을 3x3 크기만큼의 입력만을 이용하여 convolution단위 연산을 수행

10

[10] https://blog.naver.com/neogates/222434251795

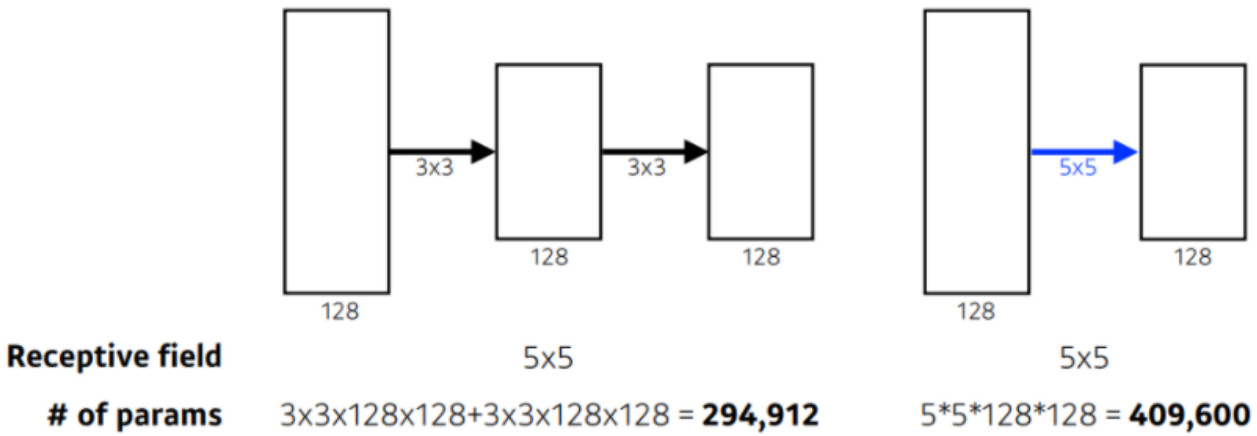
Filter의 선택

- 이러한 receptive filed 는 공간적으로 인접한 정보들간의 관계를 필터를 이용하여 특징을 뽑아내는 크기 단위로도 볼 수 있다.
-

11

[11] https://blog.naver.com/neogates/222434251795

Receptive field의 다양화



(1) filter size 를 다양하게 사용

- 다만 filter size가 커지면 그만큼 연산량이 exponential하게 늘어나 연산량과 시간, 자원 등을 사용해야 한다.
-

12

[12] https://blog.naver.com/neogates/222434251795

Receptive field의 다양화

(2) 입력 데이터의 크기를 늘리고 줄여서 전체 입력 사이즈 대비 상대적인 receptive field 크기를 조절할 수 있다.

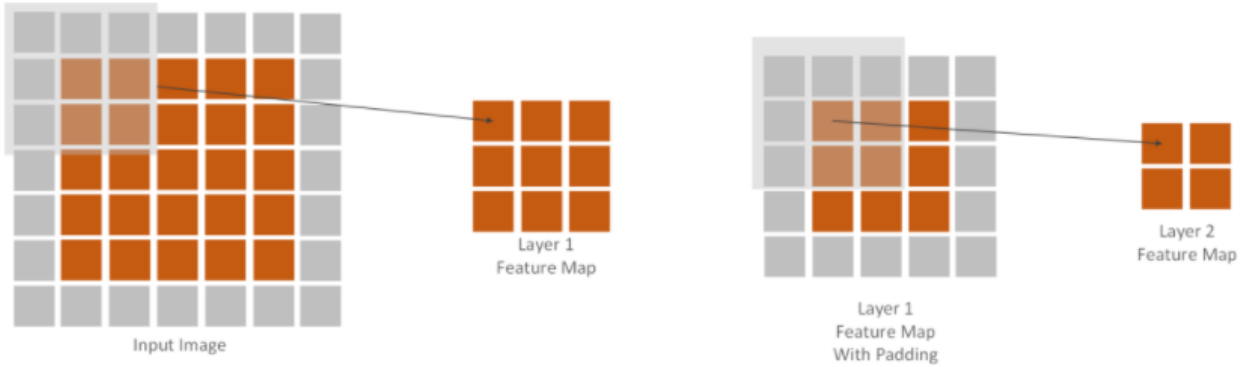
- 가장 일반적으로 사용하는 방법은 같은 filter size라도 커다란 해상도의 입력 데이터에서는 작은 receptive filed로 취급될 수 있지만, 작은 해상도의 입력 데이터에서는 큰 receptive field로 작용할 수 있다.

13

[13] https://blog.naver.com/neogates/222434251795

Receptive field의 다양화

- convolution 연산을 거쳐 입력대비 출력 크기를 줄이거나 pooling 등으로 출력크기를 줄이면 filter size를 그대로 유지하더라도 다양한 receptive filed를 적용하는 효과를 가져올 수 있다.



  - 위의 그림에서 filter size는 3x3에서 2x2로 줄었지만, 입력 해상도가 줄어듦으로서 오히려 receptive field는 늘어난 것을 확인할 수 있다.
  -

14

[14] https://blog.naver.com/neogates/222434251795

Padding and stride

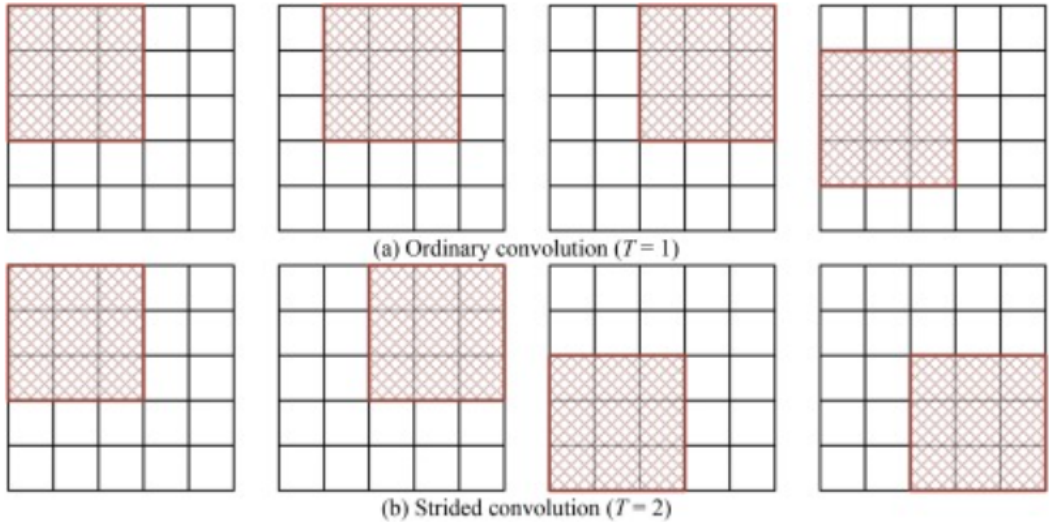
Padding

- 6개의 특성변수가 입력되었으므로, 출력도 6개로 만들고 싶다면 입력 양 끝에 0을 추가하면 크기가 (1,6)인 1D텐서를 출력 $\rightarrow$ padding

Stride

- 그림 2-6은 커널이 한칸씩 움직였지만, 3칸씩 움직인다면 $w_1a + w_2b + w_3c$와 $w_1d + w_2e + w_3f$ 2개의 선형결합을 출력함
- 이처럼 이동하는 칸수를 stride라고 한다.

Stride

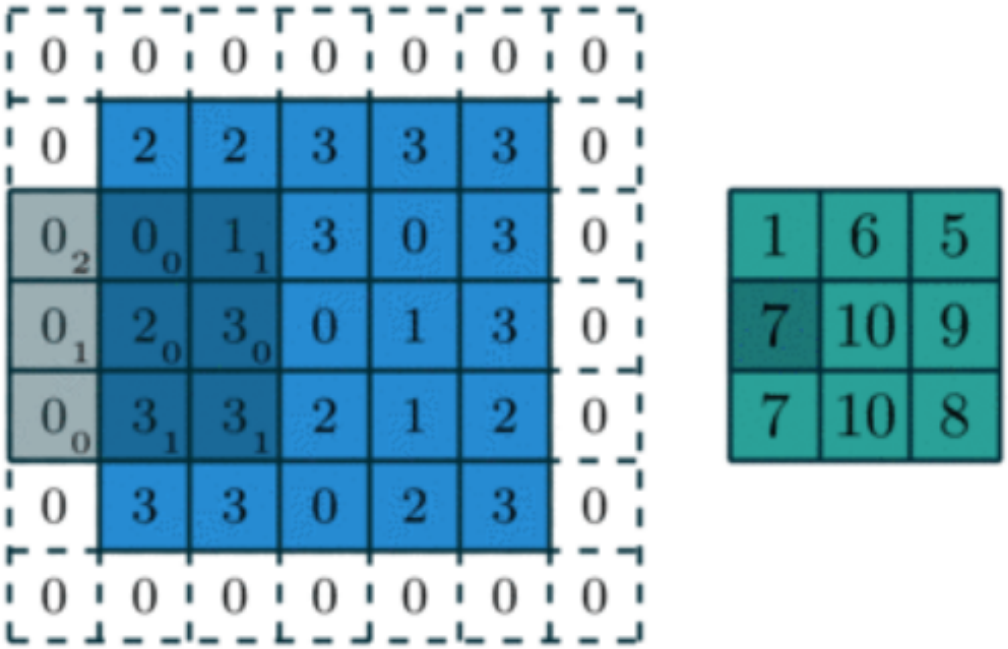


- stride를 2이상으로 잡으면 출력 tensor의 width와 height는 입력 tensor에 비해 2배이상으로 줄어든다.
- 심지어 stride를 1로 잡아도 입력 tensor보다 작아질 수 밖에 없는데. 입력 tensor의 width, height와 동일한 출력 tensor를 구해야 할때 입력 텐서의 외곽에 Padding 을 넣어 출력 tensor의 width, height를 보존할 수 있다.

15

[15] https://blog.naver.com/neogates/222435500734

Padding



16

[16] https://blog.naver.com/neogates/222435500734

Convolution 연산

- 입력 Tensor Width: W
- 입력 Tensor Height: H
- 필터 Width: FW
- 필터 Height: FH
- Stride: S
- Padding: P

$$OutputHeight = OH = \frac{(H + 2P - FH)}{S} + 1$$

$$OutputWidth = OW = \frac{(W + 2P - FW)}{S} +$$

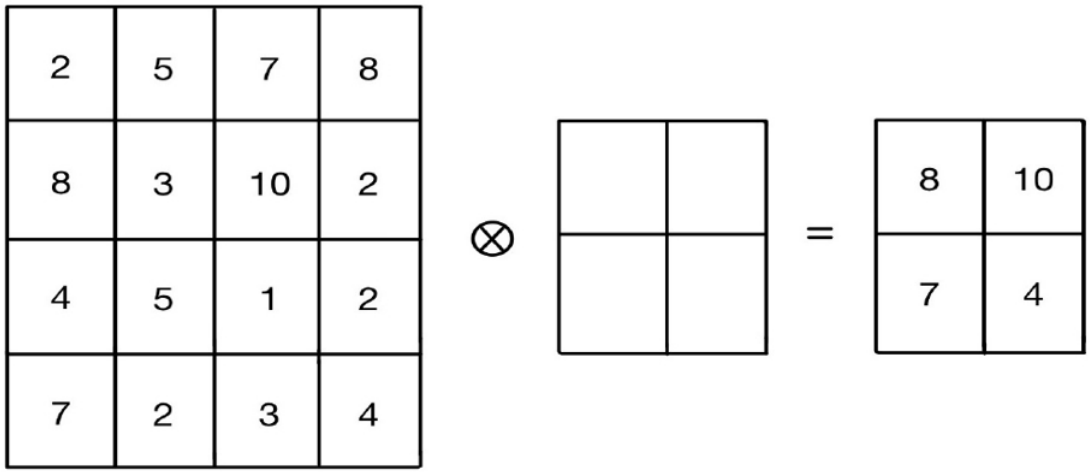
17

[17] https://blog.naver.com/neogates/222435500734

Pooling

- 이미지 자료의 특성을 좀 더 드러나게 하기 위한 커널
- 2D커널인 경우, stride= 2를 가진 (2,2) 또는 stride= 3를 가진 (3,3) pooling 커널을 사용
- stride와 pooling의 크기가 일치하는 이유는 pooling 커널 적용시 겹치는 부분이 없도록 하고, 원래 텐서의 크기를 1/2, 1/3이 되도록 하기 위함
- pooling 커널은 모수의 수를 줄일 뿐만아니라, CNN의 성능을 향상시키는 역할
- pooling 커널은 선형결합이 아니므로 커널 자체의 모수는 없음

Max-pooling



$$8 = \max\left(\begin{array}{|c|c|}\hline 2 & 5 \\ \hline 8 & 3 \\ \hline\end{array}\right) \quad 10 = \max\left(\begin{array}{|c|c|}\hline 7 & 8 \\ \hline 10 & 2 \\ \hline\end{array}\right)$$

$$7 = \max\left(\begin{array}{|c|c|}\hline 4 & 5 \\ \hline 7 & 2 \\ \hline\end{array}\right) \quad 4 = \max\left(\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline\end{array}\right)$$

그림 2–13 stride=2인 (2,2) max–pooling

Max-pooling

- 그림 2-13은 stride= 2인 max-pooling의 작동원리를 보여주고 있음
- 좌에서 우, 위에서 아래로 내려가면서 최댓값으로 8, 10, 7, 4를 구함
- (4,4)인 2D 입력텐서의 크기를 1/2로 줄여 (2,2)인 2D텐서가 됨
- 이미지 자료에서 픽셀의 숫자가 큰것은 밝은색(흰색)을 의미하므로, max-pooling 커널 안에서 가장 밝은 색을 추출하는 역할을 함
- 이미지의 edge를 추출하는 기능

4개의 히든층을 가진 CNN

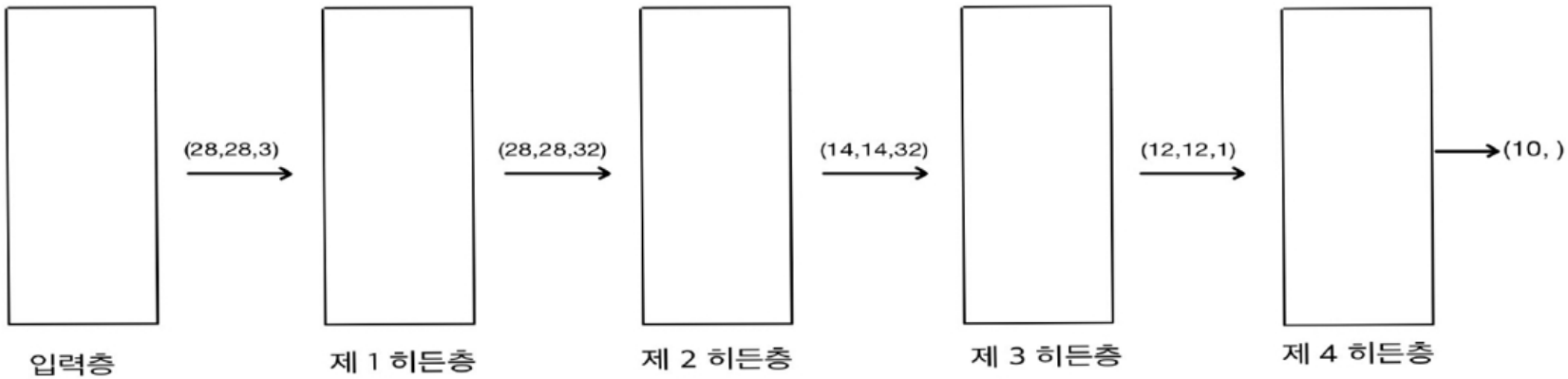


그림 2-14 4개의 히든층을 가진 CNN 아키텍처

4개의 히든층을 가진 CNN

1

2

3

4

5

4개의 히든층을 가진 CNN의 출력

- 입력층에서는
- 제1히든층에서는
- 제2히든층에서는
- 제3히든층에서는
- 제4히든층에서는

4개의 히든층을 가진 CNN의 모수의 개수

- 입력층 $\rightarrow$ 제1층:

- 제1층 $\rightarrow$ 제2층:
- 제2층 $\rightarrow$ 제3층:
- 제3층 $\rightarrow$ 제4층:
- 총