20기 정규세션

ToBig's 19기 강의자

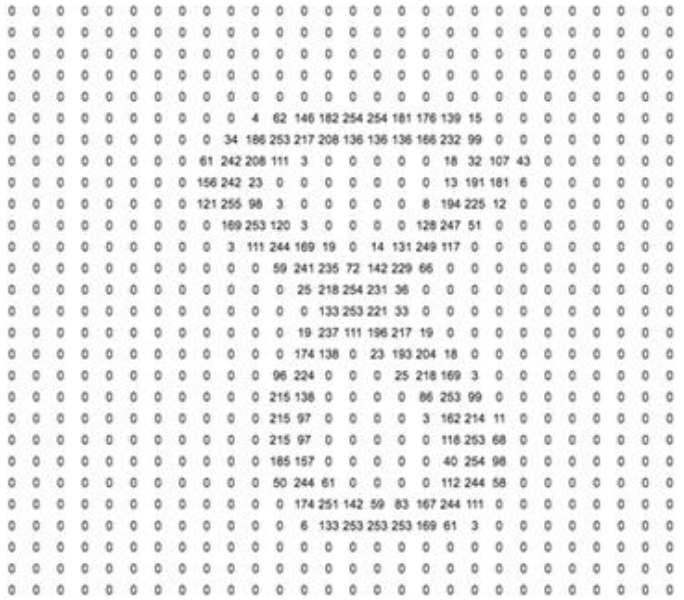
조성민

Vision Basic

Contents

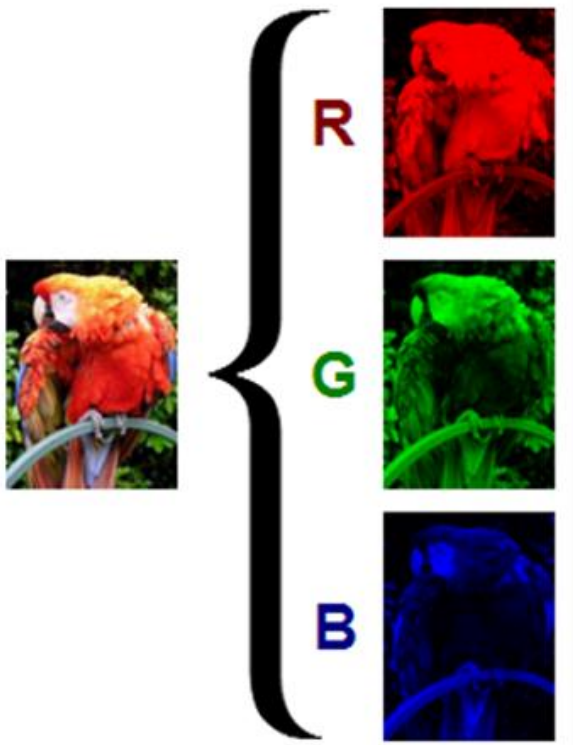
Image와 tensor의 관계

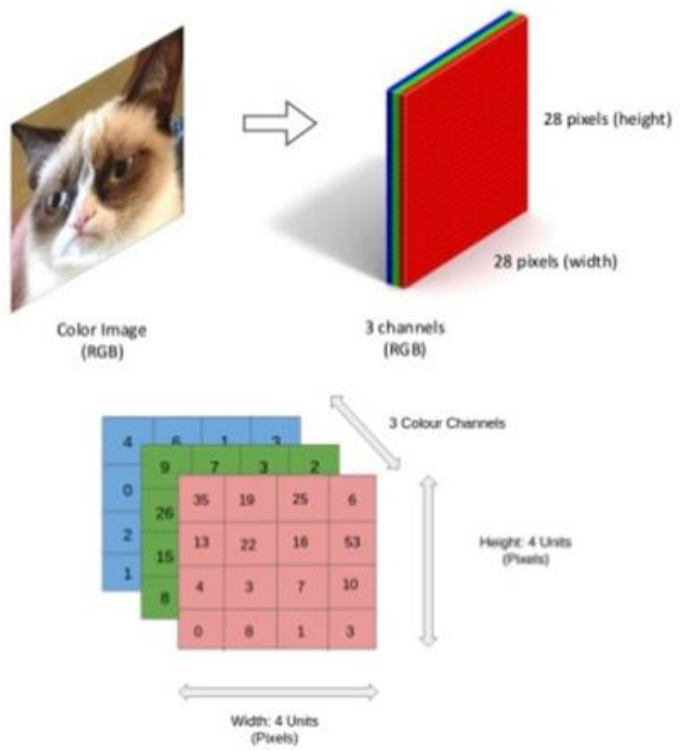
Unit 01 | Image와 tensor의 관계



흑백 이미지는 2차원 tensor로 이루어져 있음 → 수치화 할 필요 X

Unit 01 | Image와 tensor의 관계



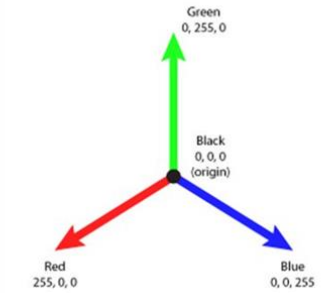


컬러 이미지는 R, G, B 3개의 행렬이 합쳐져 있는 형태로 3차원 tensor로 구성됨

Unit 01 | Image와 tensor의 관계

C×H×W

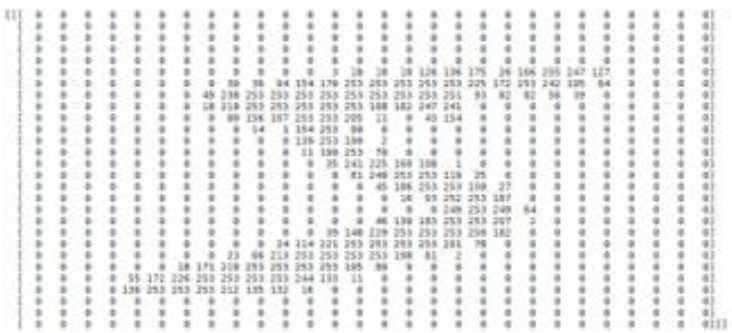
H×W×C

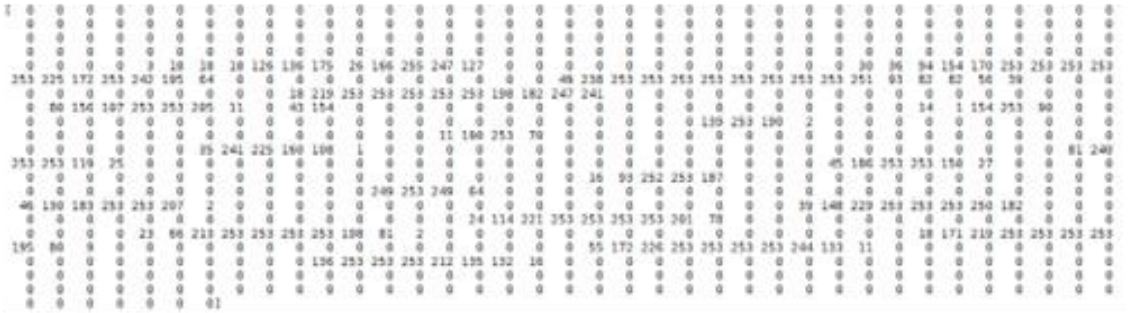


사용하는 프레임워크 및 패키지에 따라 CHW 혹은 HWC 형식으로 맞추어야 함

Convolution

Unit 02 | Convolution

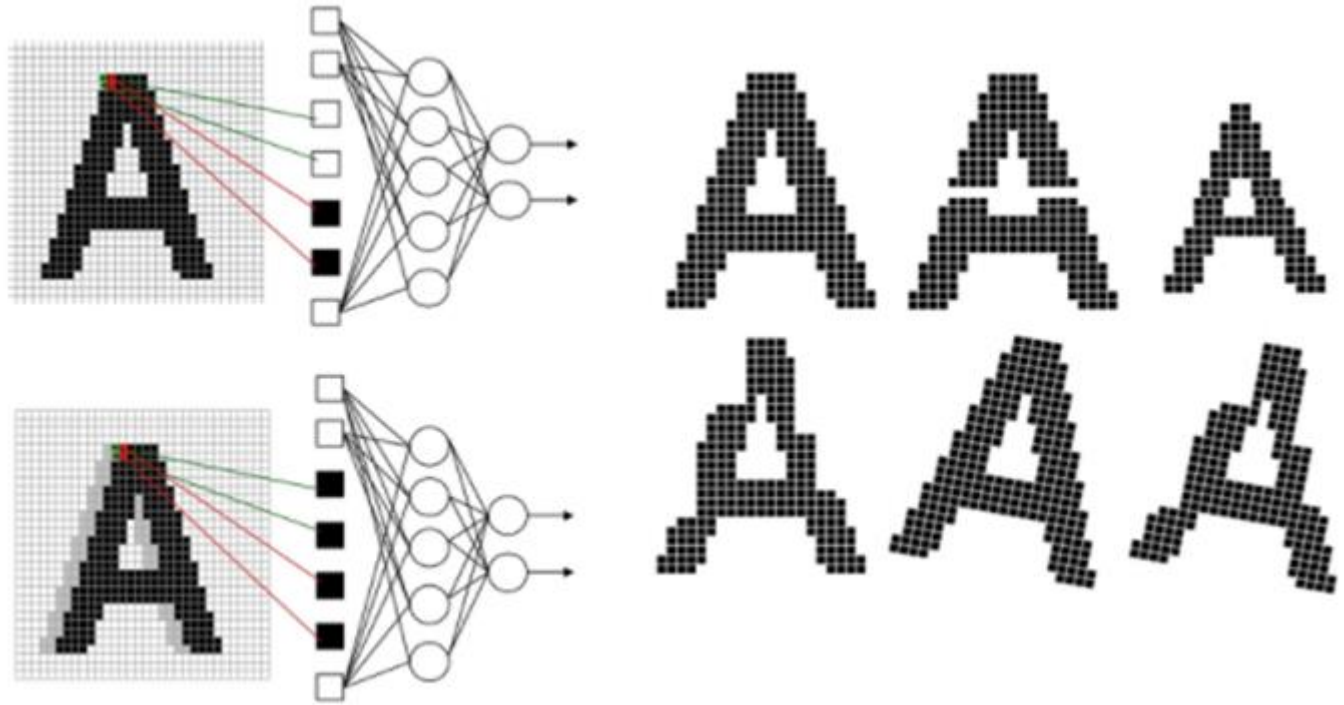




Fully connected layer 는 tensor를 flatten하여 1차원으로 차원을 줄이고 학습을 진행

→ locality한 정보가 모두 사라짐

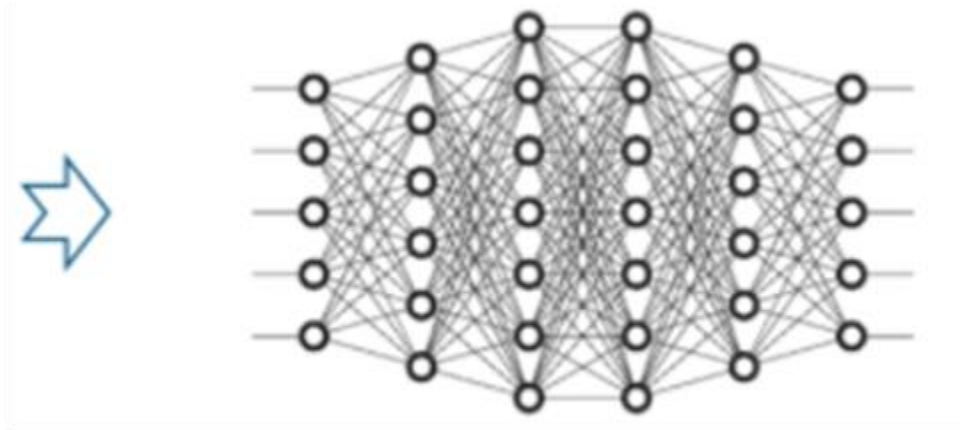
Unit 02 | Convolution



Fully connected layer 으로만 이루어진 DNN의 경우 확대 및 회전과 같은 변형에 취약함

Unit 02 | Convolution

$$3 \times 1{,}920 \times 1{,}080 = 6{,}220{,}800$$



픽셀 수가 많은 고해상도 이미지를 학습할 경우 파라미터의 수가 급격히 많아져
막대한 계산비용과 overfitting문제를 초래함

Unit 02 | Convolution

Fully connected layer의 한계를 해결하기 위해 CNN(Convolutional Neural Network) 등장

$$3 \times 1{,}920 \times 1{,}080 = 6{,}220{,}800$$

픽셀 수가 많은 고해상도 이미지를 학습할 경우 파라미터의 수가 급격히 많아져
막대한 계산비용과 overfitting문제를 초래함

Unit 02 | Convolution

$$[f * g](i) = \sum_{p=1}^{d} f(p)g(i+p) \quad \longleftarrow \text{1D-conv}$$

$$[f * g](i,j) = \sum_{p,q} f(p,q)g(i+p, j+q) \quad \longleftarrow \text{2D-conv}$$

$$[f * g](i,j,k) = \sum_{p,q,r} f(p,q,r)g(i+p, j+q, k+r) \quad \longleftarrow \text{3D-conv}$$

입력 데이터에 맞게 다양한 차원의 convolution 연산 수행 가능

Unit 02 | Convolution

$$\langle A, B \rangle = \sum_{1 \leqslant i,j \leqslant n} a_{i,j} b_{i,j}$$

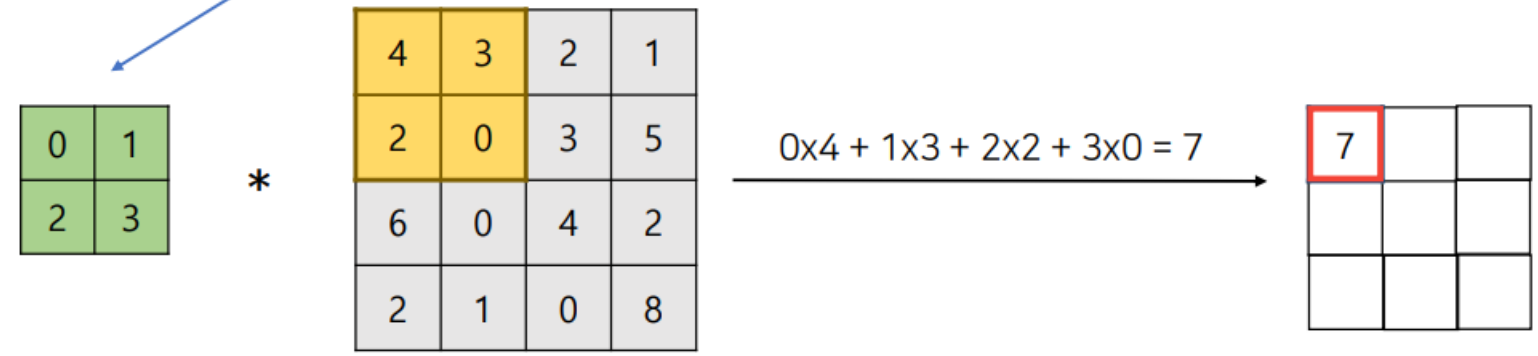
$$\left\langle \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \right\rangle = a_{11}b_{11} + a_{12}b_{12} + a_{21}b_{21} + a_{22}b_{22}$$

합성곱에는 다양한 방법들이 존재하지만 일반적으로 Frobenius inner product 연산 수행

Unit 02 | Convolution

- **2D-Convolution**

$$[f * g](i, j) = \sum_{p,q} f(p, q)g(i + p, j + q)$$

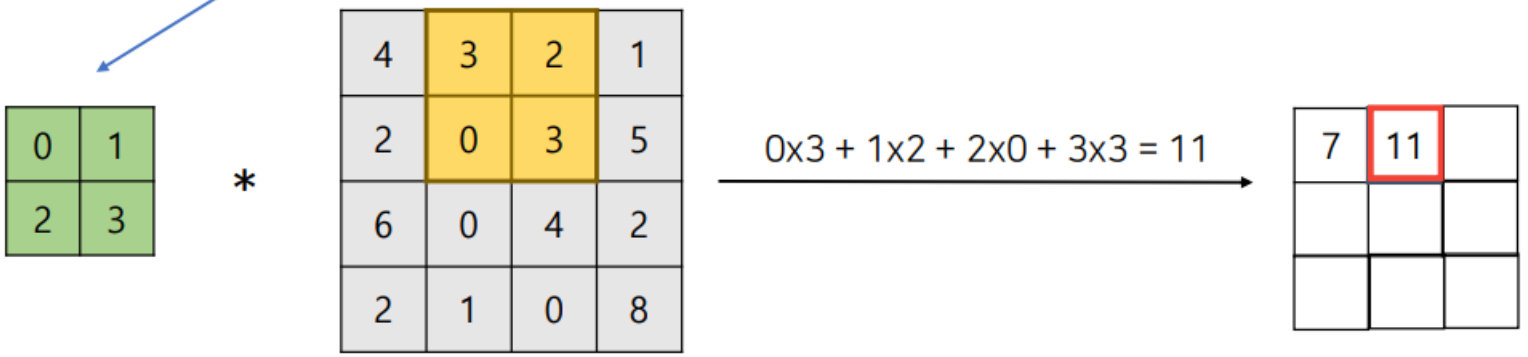


고정된 크기의 커널(kernel)을 일정 크기만큼 이동해가며 convolution 적용

Unit 02 | Convolution

- **2D-Convolution**

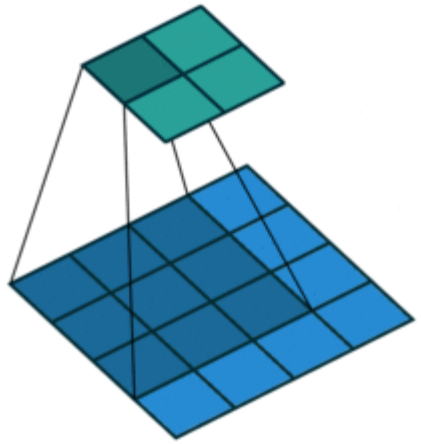
$$[f * g](i, j) = \sum_{p,q} f(p, q)g(i + p, j + q)$$



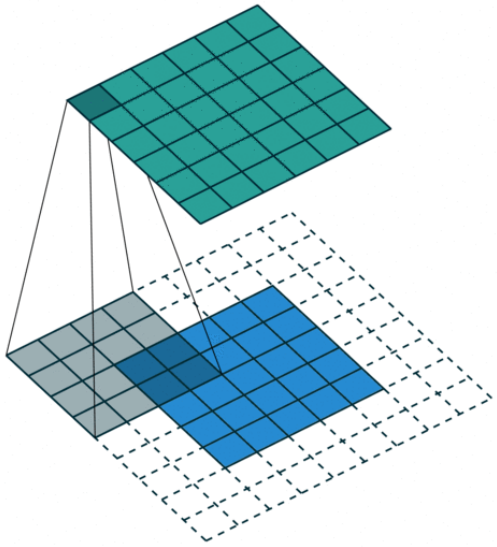
커널 변화 없이 입력만 변함

Unit 02 | Convolution

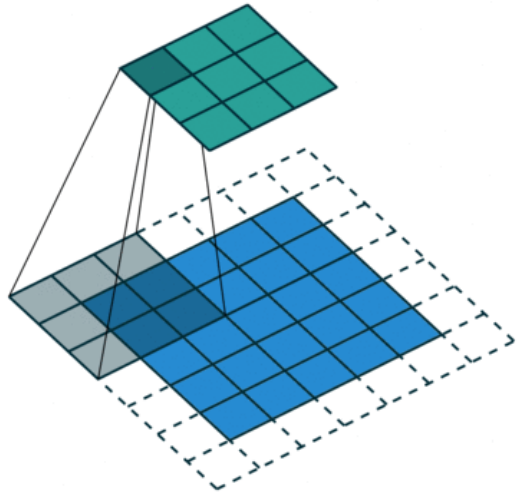
- **2D-Convolution**



- Filter : 3 x 3
- Padding : 0
- stride : 1



- Filter : 4 x 4
- Padding : 2
- stride : 1

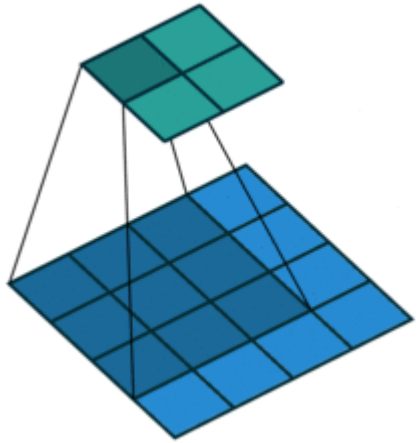


- Filter : 3 x 3
- Padding : 1
- stride : 2

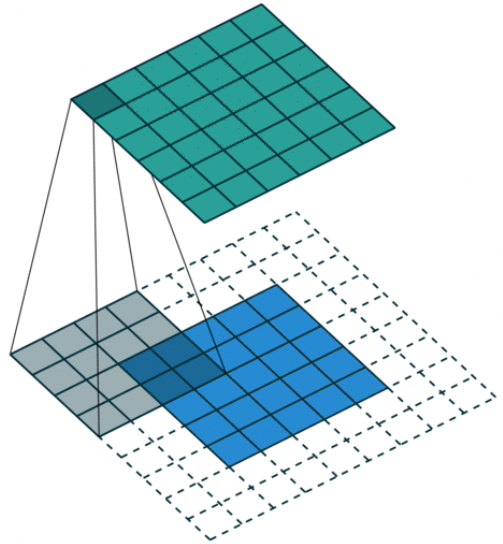
Unit 02 | Convolution

- **Filter size**

  한번의 convolution으로 이해할 수 있는 영역의 크기를 조절
  3x3 filter size가 가장 일반적으로 사용됨



Filter size : 3 x 3

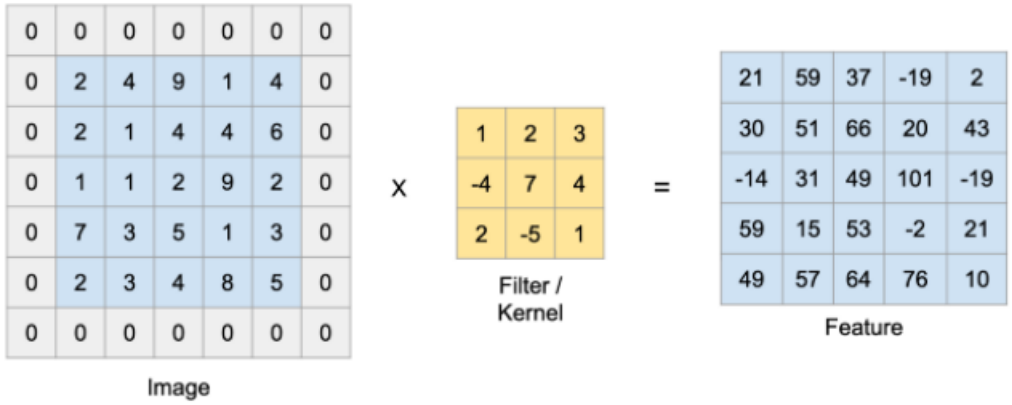


Filter size : 4 x 4

Unit 02 | Convolution

- **Padding**

convolution filter를 통과하면 Input image에 비해 크기가 작아진다.
Input image와 output image의 크기를 유지하기 위해 padding 사용



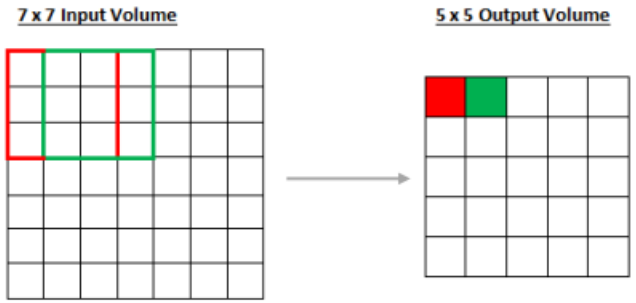
$$OW = \frac{W + 2P - FW}{S} + 1$$

- W(Width) : 입력하는 이미지 행렬의 가로 해상도
- P(Padding) : 주위에 0이 채워지는 줄의 개수
- FW(Filter Width) : Filter의 가로 해상도
- S(Stride) : 필터의 움직이는 칸 수
- OW(Output Width) : 합성곱을 하여 출력된 행렬의 가로 해상도

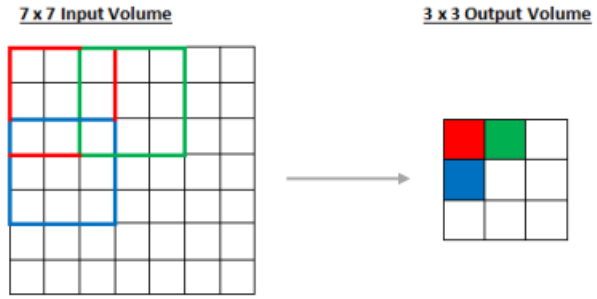
Unit 02 | Convolution

- **stride**

filter의 보폭을 의미하며, 적용되는 간격의 크기를 결정한다.
paddin을 크게 하면 output의 크기가 커지지만, stride를 크게 하면 output의 크기는 작아진다.



Stride : 1

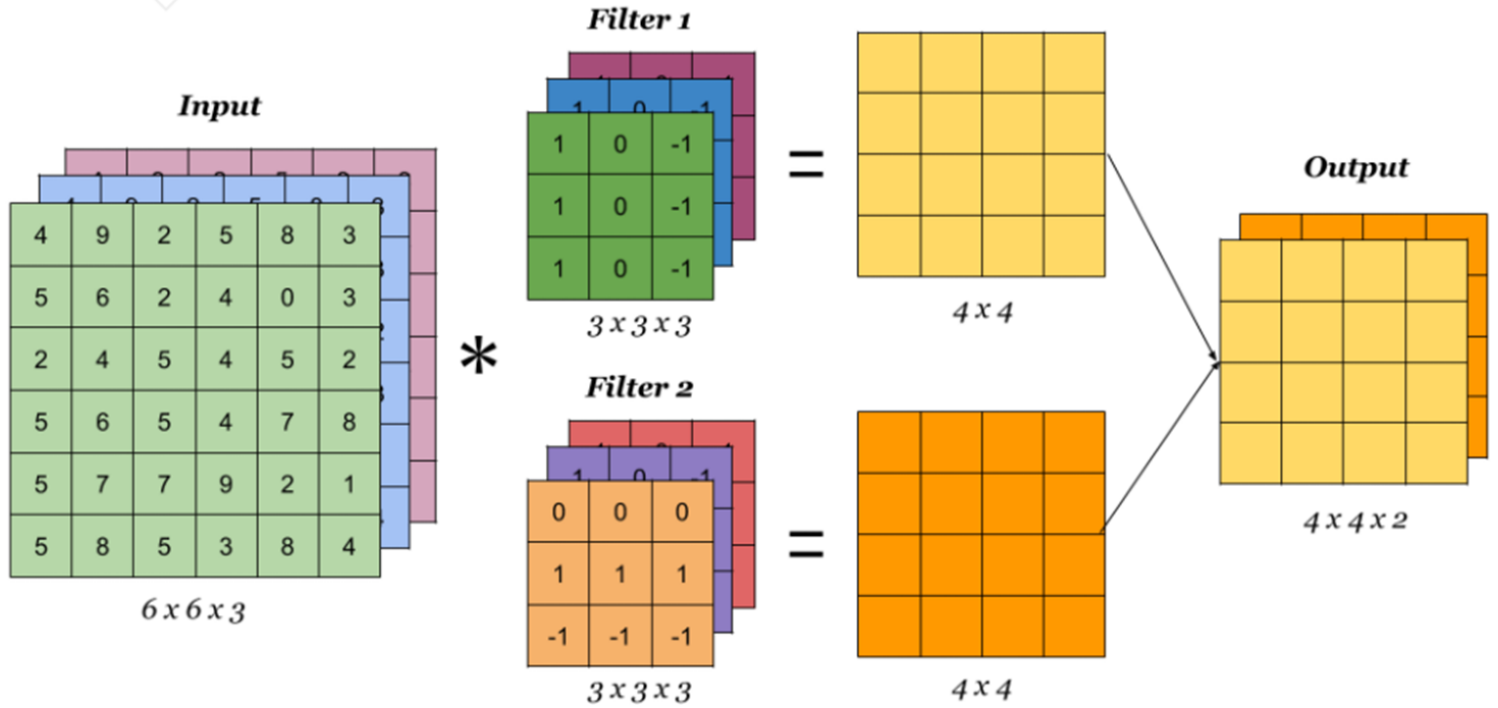


Stride : 2

Unit 02 | Convolution

- **3차원 텐서의 합성곱**

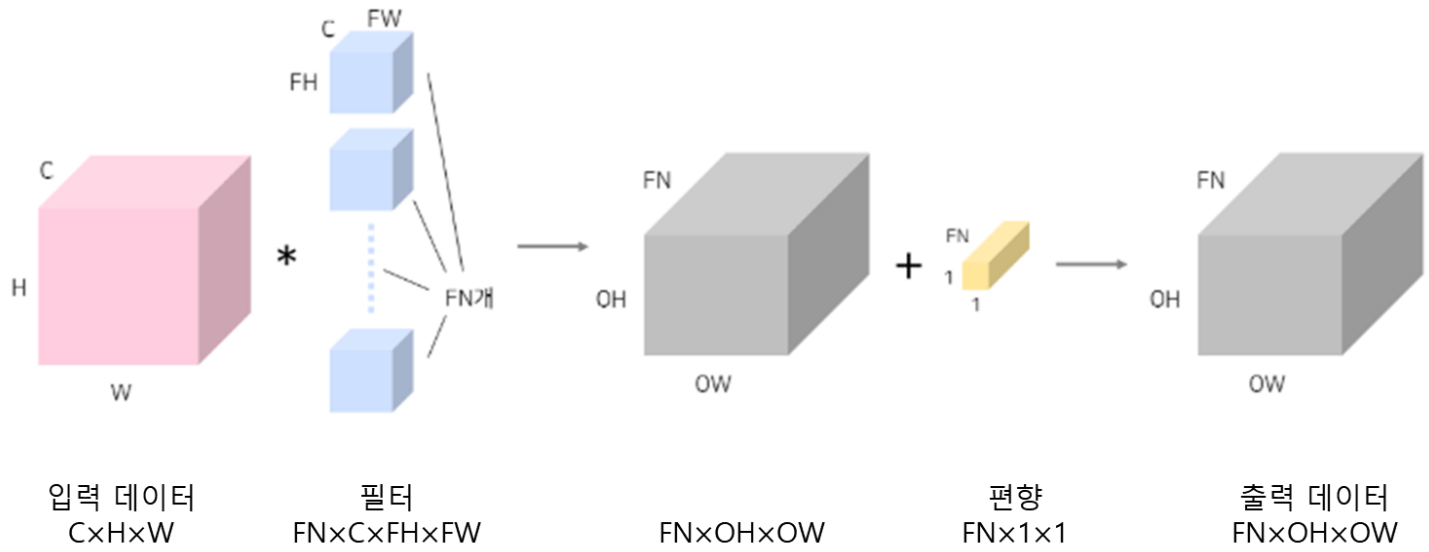
filter와 그 filter에 대응하는 tensor와 내적하여 나온 스칼라 값들로 하나의 행렬 도출
여러 개의 filter를 사용하여 locality한 정보 학습



Unit 02 | Convolution

- **3차원 텐서의 합성곱**

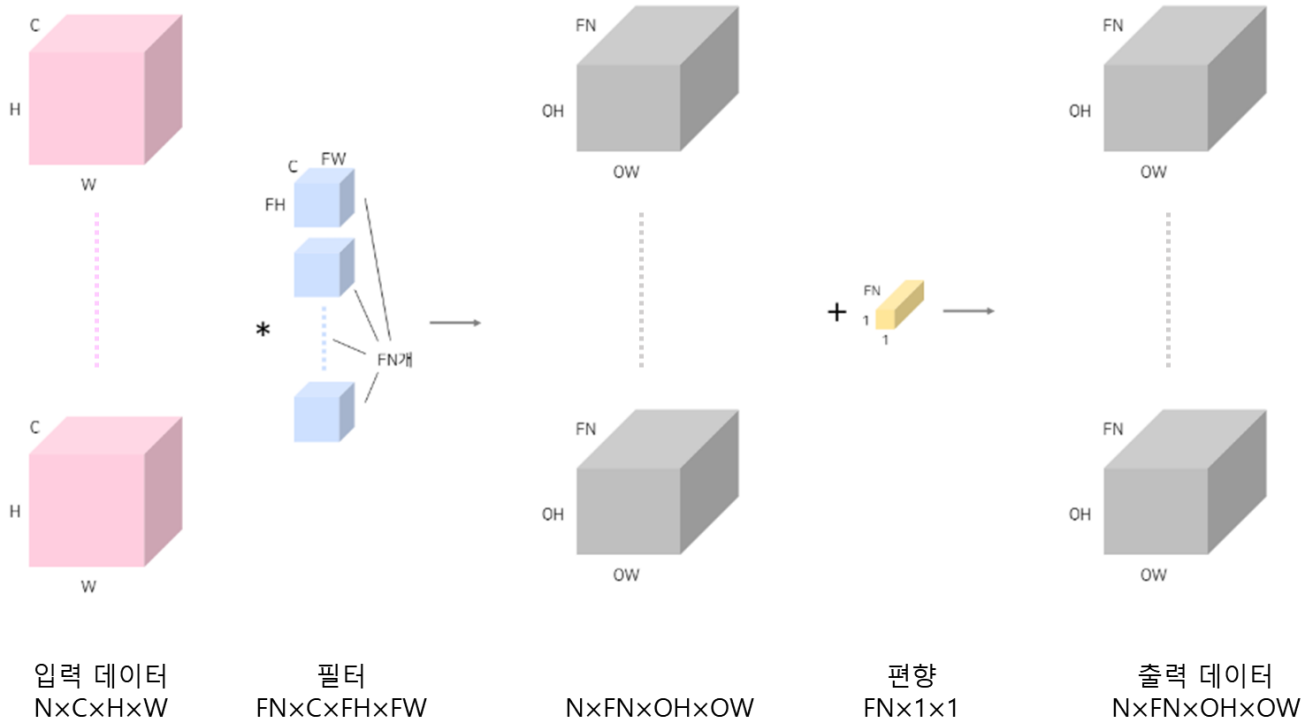
항상 input data의 channel 수와 filter의 차원 수는 같아야 한다.



Unit 02 | Convolution

- **4차원 텐서의 합성곱**

3차원 텐서인 컬러 이미지가 여러장 묶인 batch를 학습하는 것은 4차원 텐서를 입력으로 받는 것과 동일
입력 데이터의 늘어난 차원 수인 N개만큼 합성곱을 한 결과인 출력 데이터 역시 N개 만큼 차원 수 증가

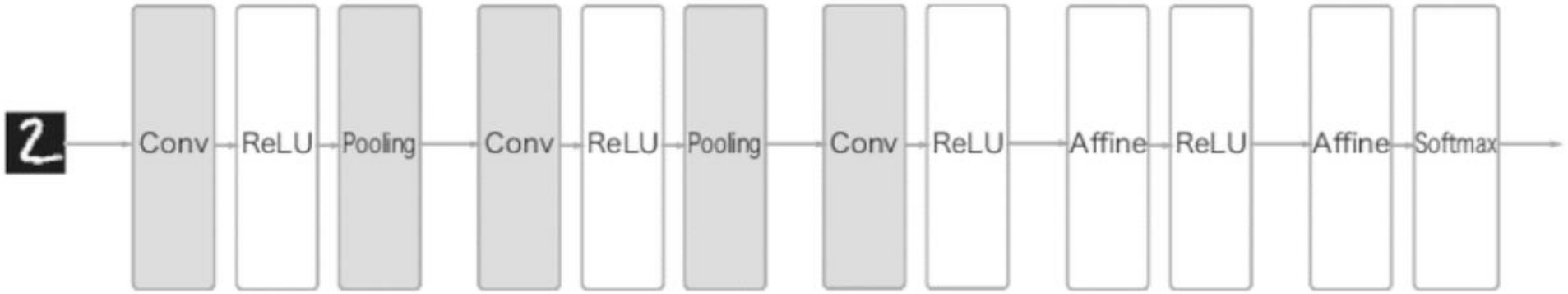


Convolutional Neural Network

Unit 03 | Convolutional Neural Network

- **CNN 구조**

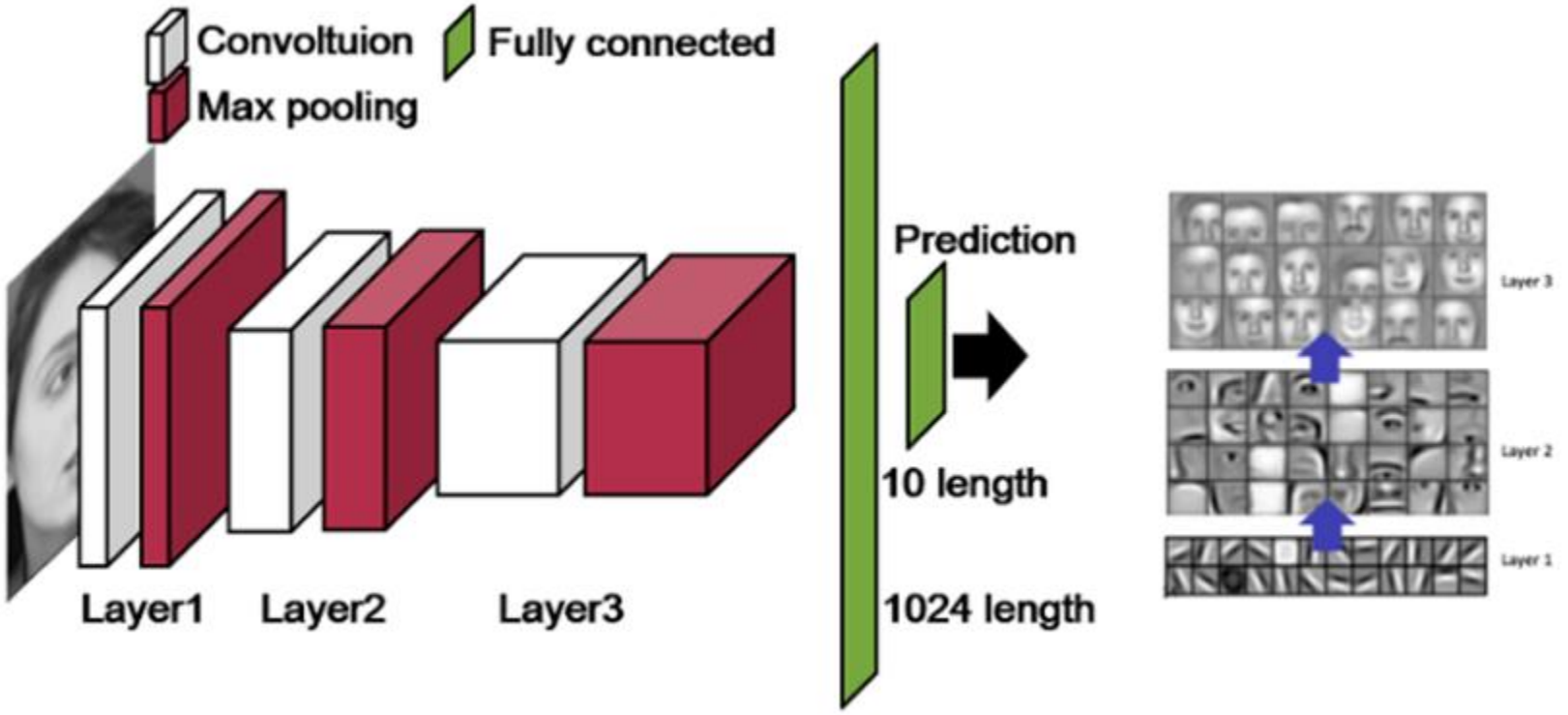
CNN은 기본적으로 "convolution layer", "Pooling layer", "FC layer" 순서로 진행된다.



Unit 03 | Convolutional Neural Network

- **CNN 구조**

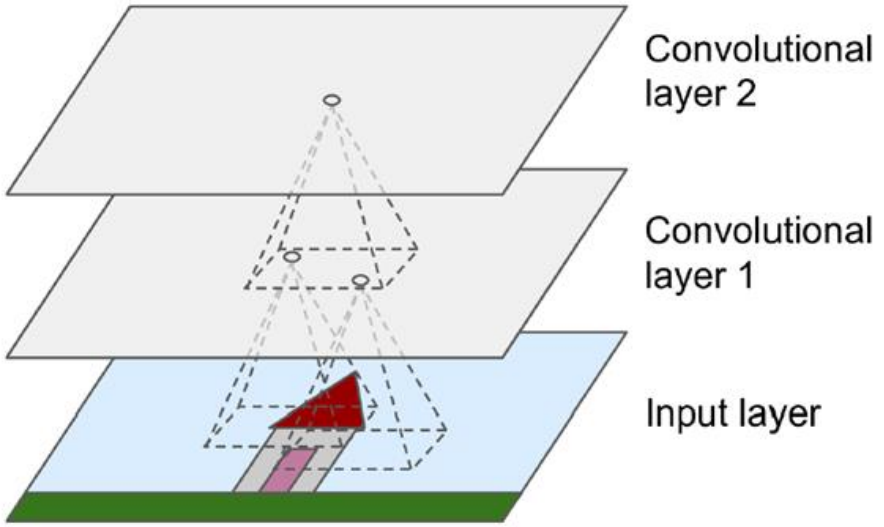
CNN은 convolution layer와 max pooling layer 가 반복된 후 fully-connected layer가 등장하는 구조로 이루어진다.



Unit 03 | Convolutional Neural Network

- **convolution layer**

1. filter와 bias를 학습
2. input data의 channel이 사라진다. (input data: N x C x H x W / output data: N x FN x OH x OW)
3. filter를 훈련시켜 낮은 층의 filter는 저수준의 local한 feature를 추출하고, 높은 층의 filter는 고수준의 global한 feature를 추출하는 것을 목표로 한다.



Unit 03 | Convolutional Neural Network

- **activation function**

  보통 convolution layer와 pooling layer 사이에 적용
  비선형 함수를 사용하여 모델의 layer를 깊게 가져가도록 함

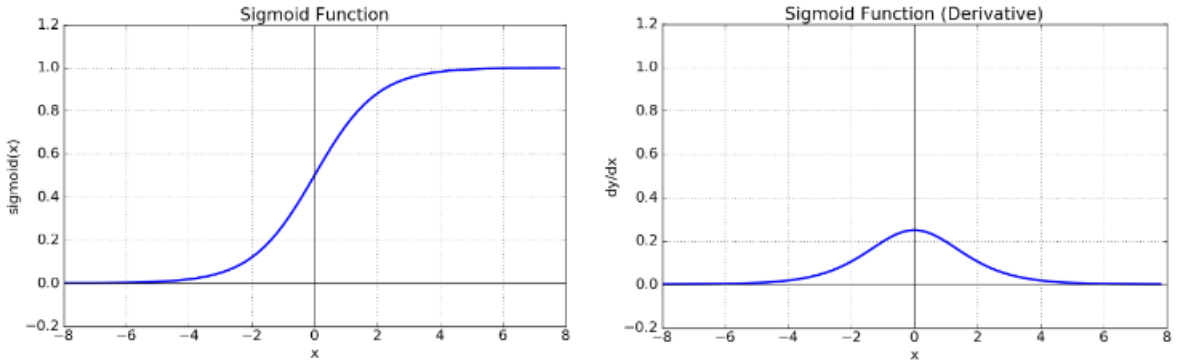
  $h(x) = cx$ 선형함수를 활성화 함수로 사용한 3층 networ를 생각해보자.
  $\rightarrow y(x) = h\left(h\left(h(x)\right)\right)$ 가 되며, 이는 $y(x) = ax$ 와 같은 식이다. $(a = c3)$

  인공신경망에서 활성화 함수는 input data를 다음 layer로 어떻게 출력하는지를 결정한다.
  즉, 입력을 받아 활성화 또는 비활성화를 결정하는 데 사용되는 함수이다.

- **activation function – sigmoid**

x의 값에 따라 0 ~ 1 의 값을 출력하는 S자 형태의 함수이다.

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

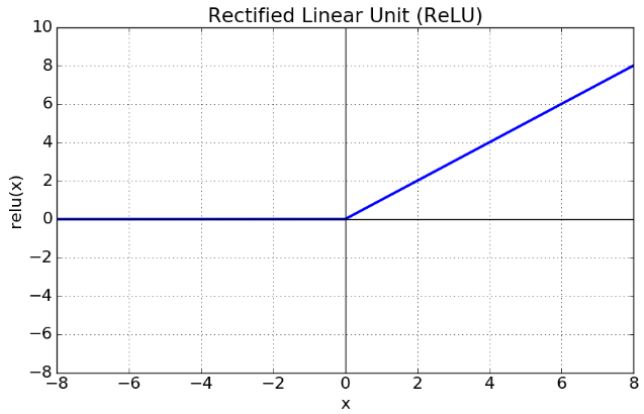


Sigmoid 함수(좌) & Sigmoid 도함수(우)

Sigmoid 도함수 그래프를 보면 미분 계수의 최대값이 0.25 로 1보다 작기에 back-propagation을 계산하는 과정에서 미분값이 계속 곱해져 은닉층의 깊이가 깊어질수록 오차율을 계산하기 어려워지는 문제가 발생하여 Vanishing Gradient Problem이 발생한다.

- **activation function – ReLU**

기존의 sigmoid 의 문제를 해결하기 위해 제안된 활성화 함수
$\mathrm{ReLU}(x) = \max(0, x)$

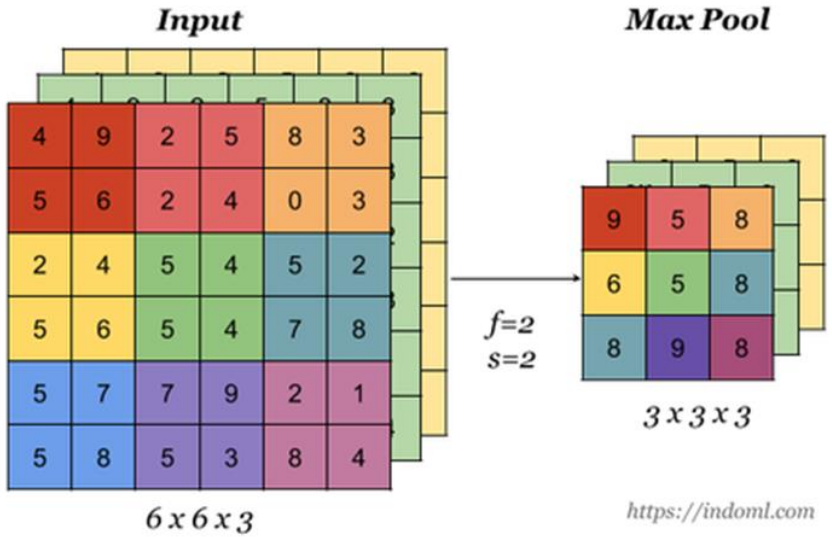


X가 0보다 크면 기울기가 1인 직선, 0보다 작으면 함수값이 0이 되는 직선으로 구성된 함수이다. 0보다 큰 경우 미분값이 1이 되기 때문에 sigmoid에서 발생되는 Vanishing Gradient Problem을 완화할 수 있다. 하지만 0보다 작은 값들에서 뉴런이 죽을 수 있다는 단점이 존재한다.

Unit 03 | Convolutional Neural Network

- **max pooling**

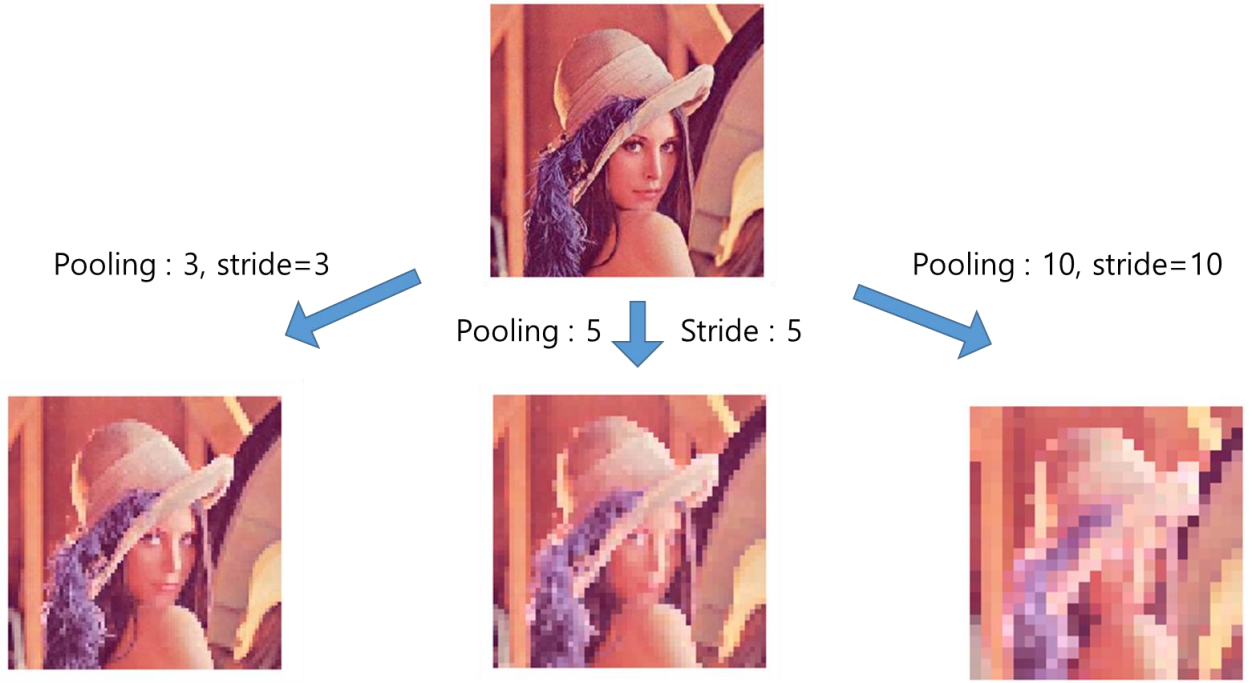
1. 학습시킬 parameter가 존재하지 않는다.
2. channel별로 독립적으로 시행된다. (input data: N x C x H x W / output data: N x C x OH x OW)
3. down samplin을 통해 다음 합성곱층에서 더 빨리 global feature를 찾을 수 있도록 도와주고, parameter 수를 줄여 overfitting을 억제한다.



Unit 03 | Convolutional Neural Network

- **max pooling**

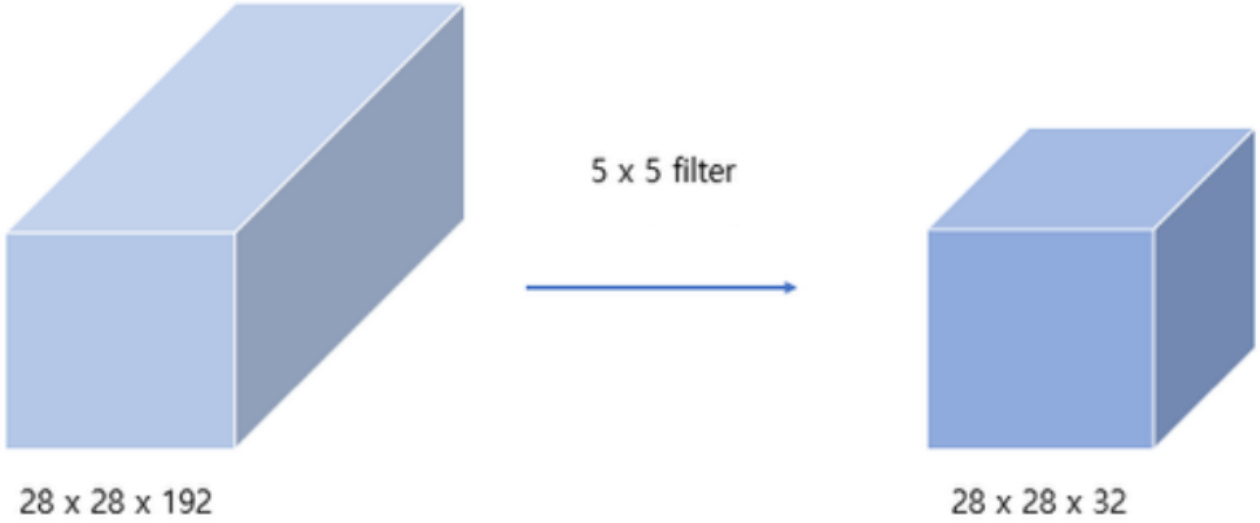
  정해진 filter의 크기에서 해상도를 가장 큰 픽셀값 1개로 줄이기 때문에 마치 input data가 모자이크를 한 것과 같이 출력된다.



Number of parameters on CNN

Unit 04 | Number of parameters on CNN

- CNN의 각 layer는 weight parameter와 bias parameter가 존재.
- 전체 네트워크의 parameter 수는 각 conv layer 파라미터 수의 합
  - { (Kernel size) x (Input channel) + (bias) } x (output channel)



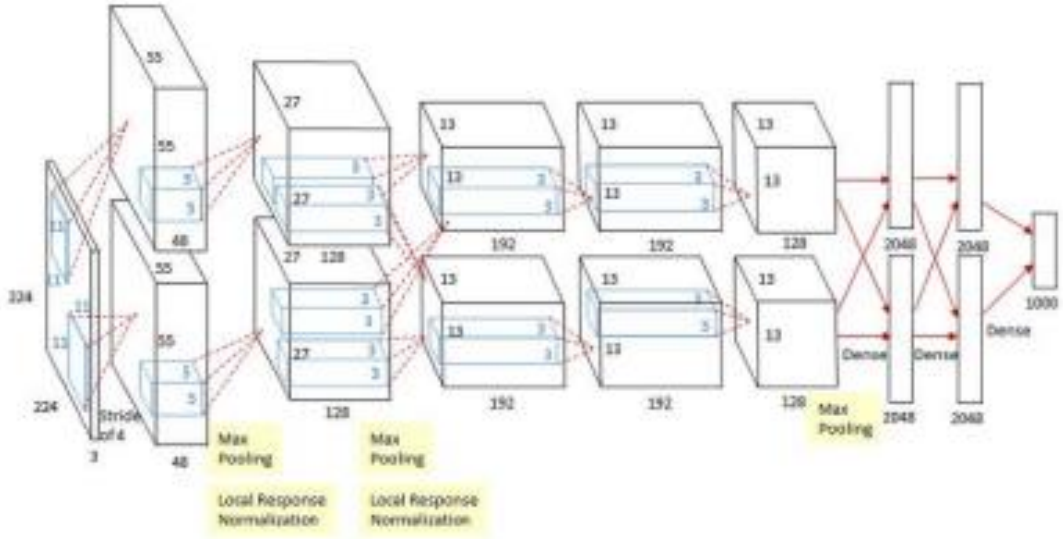
Input channel output channel

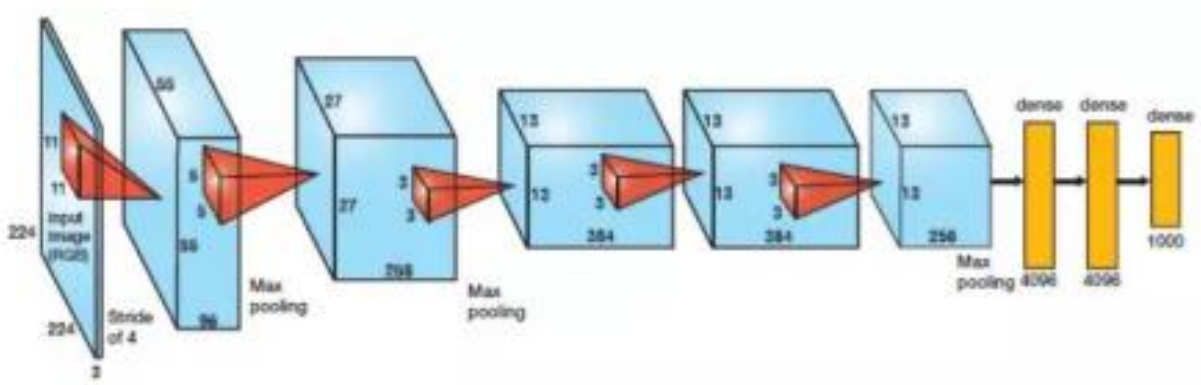
$$(5 \times 5 \times 192 + 1) \times 32 = 153{,}632$$

Kernel size bias

Assignments

과제 – AlexNet model





과제 1. AlexNet의 파라미터 개수 구하기
week7_CNNbasic_AlexNet_parameters.ipynb의 물음표를 채워주세요.

과제 2. AlexNet model의 코드 구현하기
week7_CNNbasic_AlexNet_modeling.ipynb에 모델 구현 후 summary로 전체 모델 구조 보이고 주석을 통해 간단한 설명을 해주세요.

References

- 18기 이선민님 강의
- 한경훈 교수님 딥러닝 기초 강의
  https://sites.google.com/site/kyunghoonhan/deep-learning-i
- 밑바닥부터 시작하는 딥러닝
  https://www.hanbit.co.kr/store/books/look.php?p_code=B8475831198
- https://89douner.tistory.com/57
- https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=handuelly&logNo=221824080339

들어주셔서 감사합니다.