〈데이터분석과기계학습 11주차〉

인공신경망 (2)

인공지능융합공학부 데이터사이언스전공

곽찬희

- **역전파의 이해**
  - ✓ https://www.youtube.com/watch?v=tleHLnjs5U8
  - ✓ https://www.youtube.com/watch?v=llg3gGewQ5U

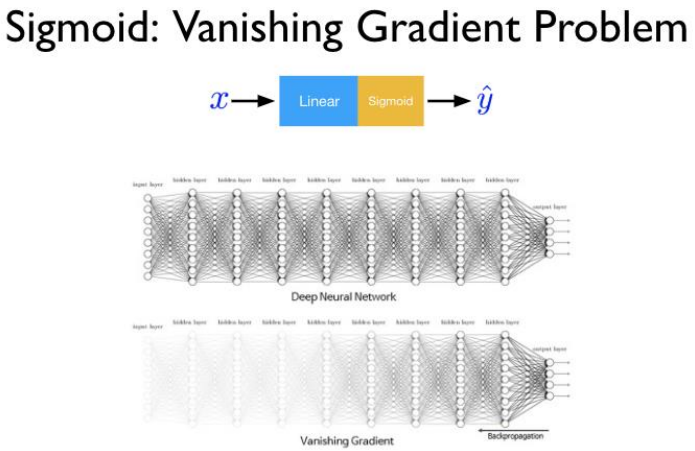
딥러닝을 제대로 사용하려면

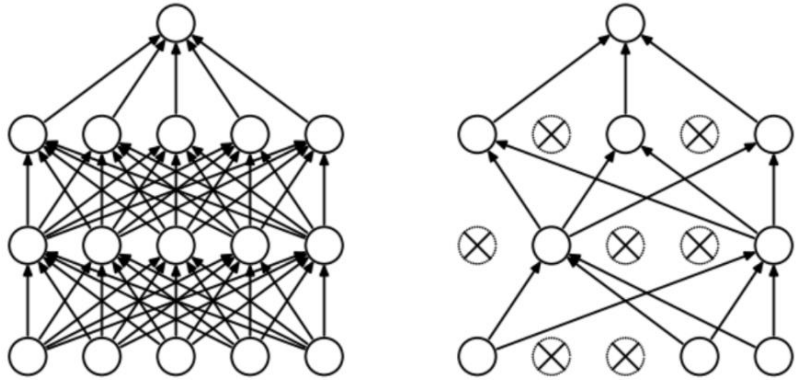
- 기존 모델과 기법을 이해함
- 각 문제별로 잘 작동하는 모델들을 이해함
- 내 문제와 유사한 모델들을 선정하여 시도함
  - ✓ 내가 풀고자 하는 문제가 무엇인지 정확하게 정의 필요
  - ✓ 내 데이터와 유사한 학습 데이터를 활용한 것이 있는지
  - ✓ 거대 모델로 대신 활용할 수 있는지 (특히 NLP)
- 어떤 기법을 추가하면 성능 향상이 있을지 고민함

- 왜 딥러닝 관련 직무에 대학원 출신을 요구할까?

Gradient Vanishing/Exploding

- Gradient Vanishing
  - ✓ Gradient 를 이용해 가중치를 업데이트할 때 앞쪽 layer 에 제대로 업데이트가 안되는 현상
  - ✓ 반대는 Gradient Exploding
- Dropout
  - ✓ VG를 해결하는 대표적 방법. Unit 을 확률적으로 선택함
  - ✓ L2규제와 유사 (why?)





(a) Standard Neural Net

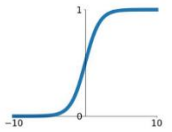
(b) After applying dropout.

Activation Functions

- 활성화 함수에는 다양한 것이 존재함
- 각각의 특징이 있음
  - ✓ Tanh: sigmoid 보다 빠른 학습
  - ✓ ReLU: V.G. 해결. 빠른 계산. Dying Neuron
  - ✓ Leaky ReLU: Dying Neuron 해결 노력
  - ✓ ELU: Exponential 요소 추가 (비용!)
  - ✓ Maxout: 두 가지 경우를 계산해 max값 도출. 계산 복잡

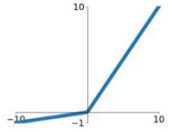
Sigmoid

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



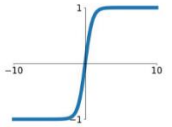
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$

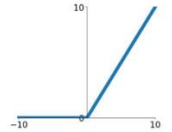


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

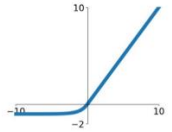
ReLU

$\max(0, x)$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

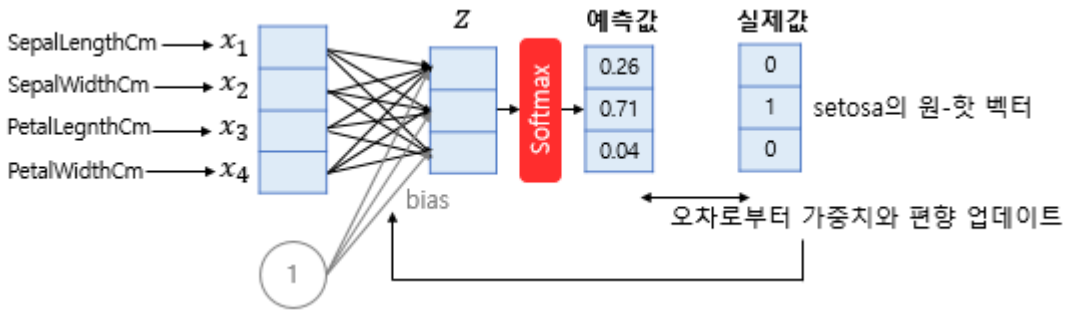


- 문제 유형마다 적합한 함수가 있음
  - ✓ 많은 문제를 다뤄봐야 하는 이유

Loss Functions

- 목적함수는 어떻게 설정할까?
  - ✓ 회귀(수치형 값의 예측): MSE, MAE…
  - ✓ 분류: (Binary)CrossEntropy
    - Softmax: 각 분류별 확률의 합이 1이 되도록 설정한 뒤 가장 큰 값을 보는 기법
  - ✓ 기타 문제 별 loss function 설정 가능

$$y_k = \frac{e^{a_k}}{\sum_{i=1}^{n} e^{a_i}}$$



https://wikidocs.net/35476

SGD보다 빠른 Optimizer!

- Momentum: 가속도를 붙여서 이동해보자
- AdaGrad: 학습률을 적응형으로 만들어보자 (Learning Rate Decay)

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$
$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

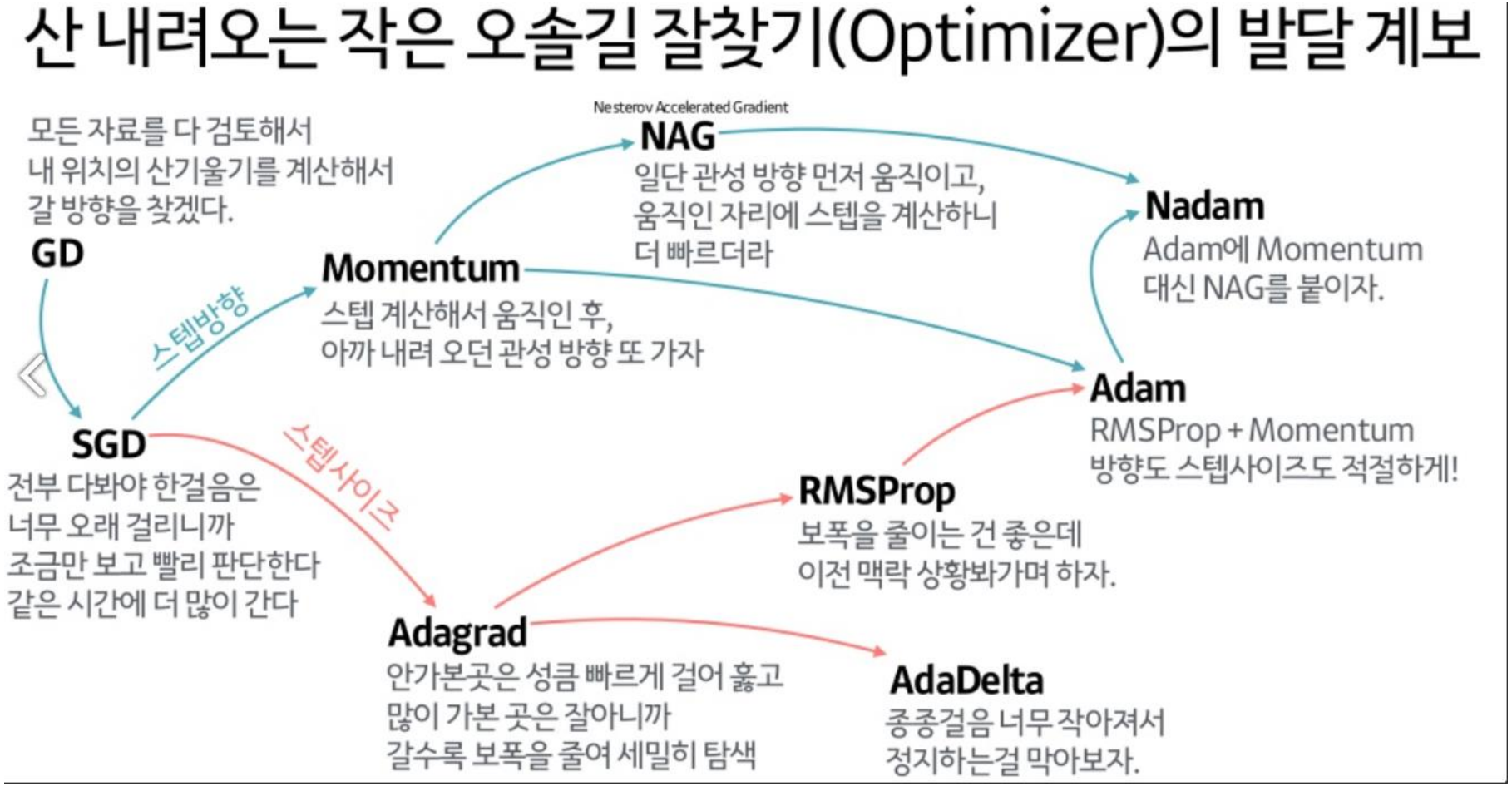
- RMSProp: AdaGrad 가 더 잘 작동할 수 있도록 수정

$$h_i \leftarrow \rho h_{i-1} + (1-\rho) \frac{\partial L_i}{\partial W} \odot \frac{\partial L_i}{\partial W}$$

- ADAM: RMSProp + Momentum

https://www.slideshare.net/yongho/ss-79607172

SGD보다 빠른 Optimizer!



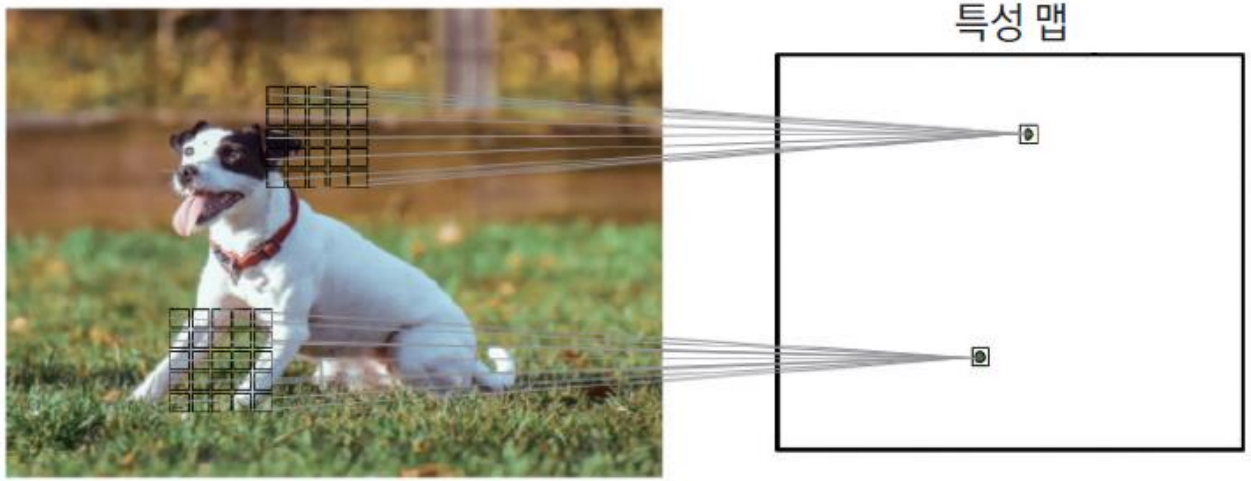
https://www.slideshare.net/yongho/ss-79607172

그 외에 찾아보면 좋은 것들

- 데이터 전처리 with Batch Normalization
- Early Stopping
- Data Augmentation
- 등등…

CNN 의 이해

- Convolutional Neural Network
  - ✓ 합성곱 신경망(CNN)은 뇌의 시각 피질이 물체를 인식할 때 동작하는 방식에서 영감을 얻은 모델
  - ✓ 이미지 분류 작업에서 CNN이 탁월한 성능을 내기 때문에 이 특별한 종류의 피드포워드 신경망은 크게 주목받았고 컴퓨터 비전을 위한 머신 러닝 분야를 크게 발전시켰음
  - ✓ (관련이 높은) 핵심 특징을 올바르게 추출하는 것은 모든 머신 러닝 알고리즘의 성능에서 아주 중요한 요소



합성곱

- 합성곱: 입력과 필터(혹은 커널) 의 곱으로 CNN 계산
  - ✓ 여기서 대괄호 []는 벡터 원소의 인덱스를 나타내는 데 사용
  - ✓ 인덱스 i는 출력 벡터 y의 각 원소에 대응함
  - ✓ -∞에서 +∞까지의 인덱스와 x의 음수 인덱싱

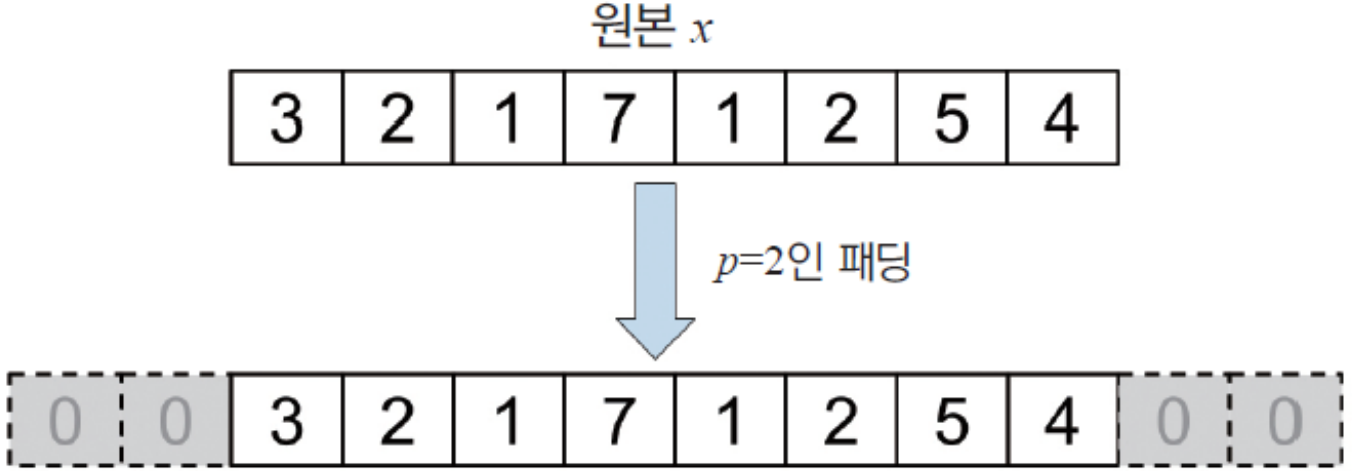
$$y = x * w$$

$$y = x * w \rightarrow y[i] = \sum_{k=-\infty}^{+\infty} x[i-k]w[k]$$

패딩(Padding)

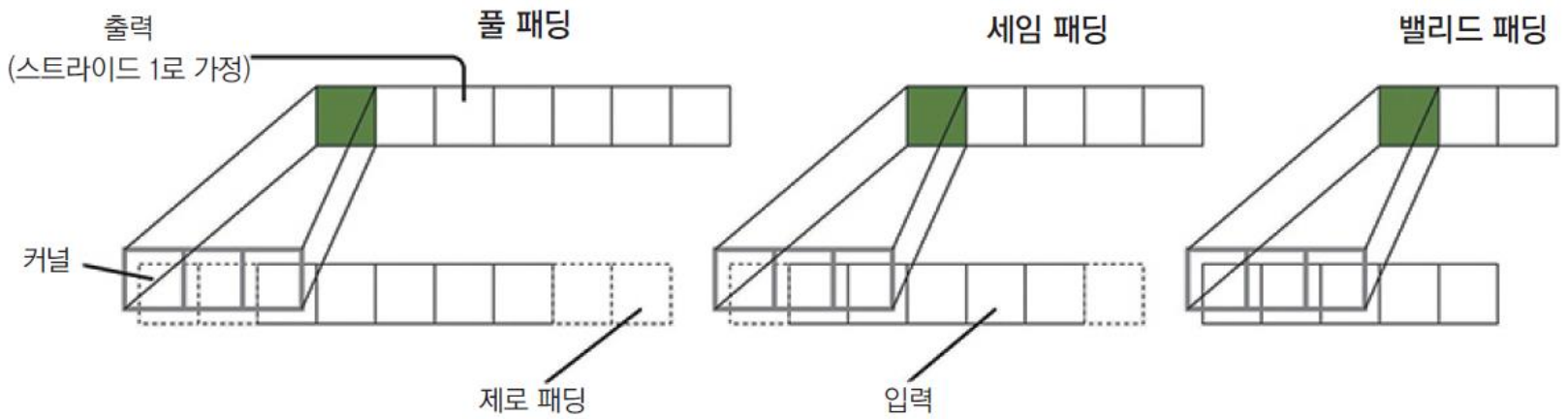
$$y = x * w \rightarrow y[i] = \sum_{k=-\infty}^{+\infty} x[i-k]w[k]$$

- ✓ 첫째 인덱스 -∞부터 +∞까지 합은 특히 이상하게 보임
- ✓ 이전 공식에 있는 덧셈을 올바르게 계산하려면 x와 w가 0으로 채워져 있다고 가정해야 함
- ✓ 또한, 출력 벡터 y도 0으로 채워진 무한 크기가 됨
- ✓ 이는 실제 상황에서는 유용하지 않기 때문에 유한한 개수의 0으로 x가 패딩



패딩(Padding)

- Full Padding : 한 칸부터 시작(출력사이즈 > 입력사이즈)
- Same Padding: 입력과 출력의 사이즈가 같도록
- Valid Padding: No Padding
- 패딩된 벡터 xp크기는 n + 2p



패딩(Padding)

- 합성곱 신경망에서 가장 많이 사용되는 패딩 방법은 세임 패딩
  - ✓ 장점: 벡터의 크기 유지
- 컴퓨터 비전 분야의 이미지 관련된 작업이라면 입력 이미지의 높이와 너비가 유지됨
- 때문에 네트워크 구조를 설계하기 쉬움
- 풀 패딩이나 세임 패딩에 비해 밸리드 패딩의 단점은 신경망에 층이 추가될수록 점진적으로 텐서 크기가 줄어 신경망 성능을 나쁘게 만들 수 있음

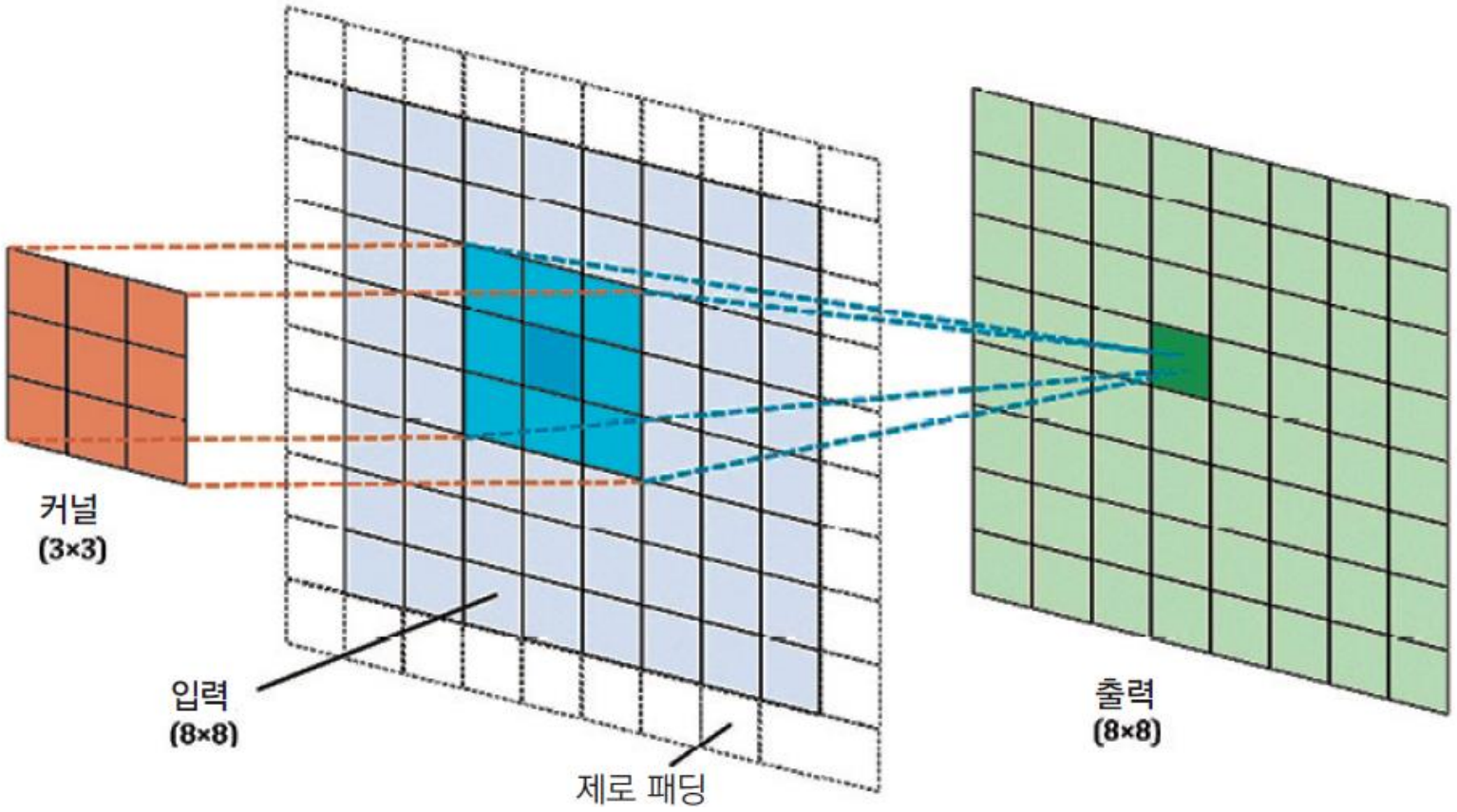
합성곱의 출력 크기

- 합성곱 출력 크기는 입력 벡터 위를 필터 w가 이동하는 전체 횟수로 결정
- 입력 벡터의 크기는 n이고 필터 크기는 m이라고 가정해 보자
- 패딩이 p이고 스트라이드가 s인 $x * w$ 출력 크기는 다음과 같이 계산

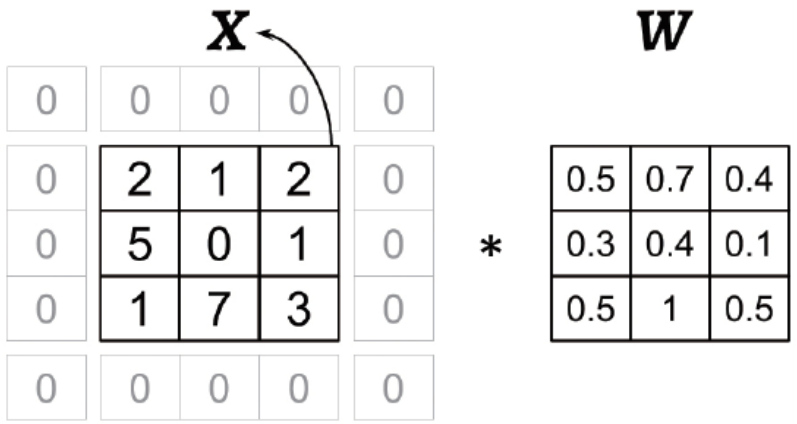
$$o = \left\lfloor \frac{n + 2p - m}{s} \right\rfloor + 1$$

- 여기서 $\lfloor . \rfloor$ 는 버림 연산을 나타냄

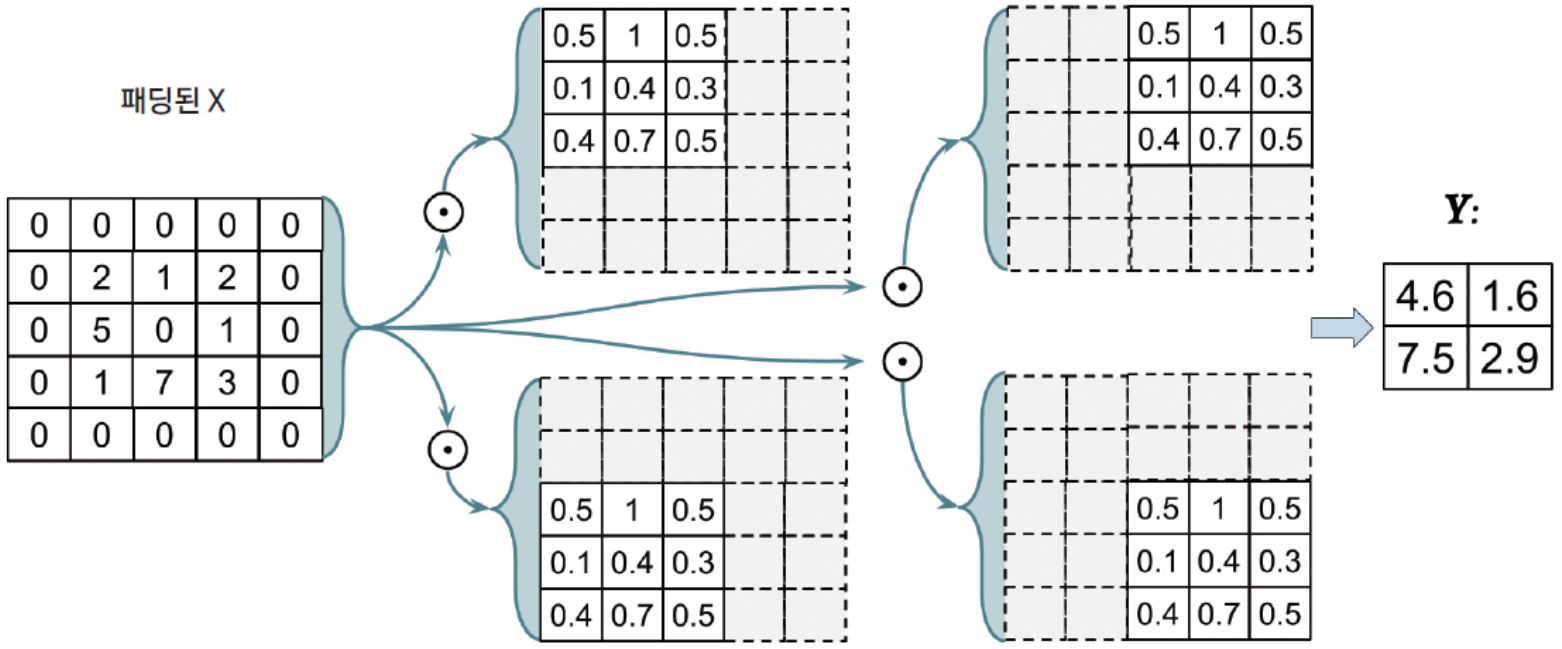
합성곱



합성곱



$$W^r = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 0.1 & 0.4 & 0.3 \\ 0.4 & 0.7 & 0.5 \end{bmatrix}$$



풀링(Pooling)

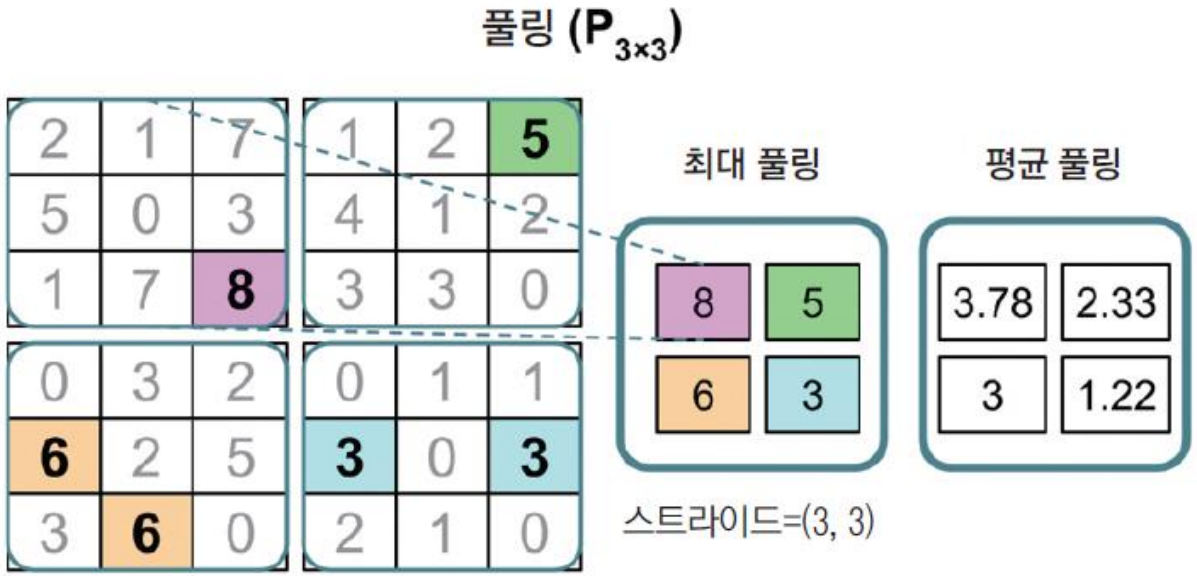
- 서브샘플링
  - ✓ 정보의 압축 (눈에 띄는 정보만 추출)
  - ✓ Max / mean pooling
  - ✓ 지역불변성(구역에 잡음이 있어도 처리 가능)
  - ✓ 계산효율성, 과대적합 감소

$$X_1 = \begin{bmatrix} 10 & 255 & 125 & 0 & 170 & 100 \\ 70 & 255 & 105 & 25 & 25 & 70 \\ 255 & 0 & 150 & 0 & 10 & 10 \\ 0 & 255 & 10 & 10 & 150 & 20 \\ 70 & 15 & 200 & 100 & 95 & 0 \\ 25 & 25 & 100 & 20 & 0 & 60 \end{bmatrix}$$

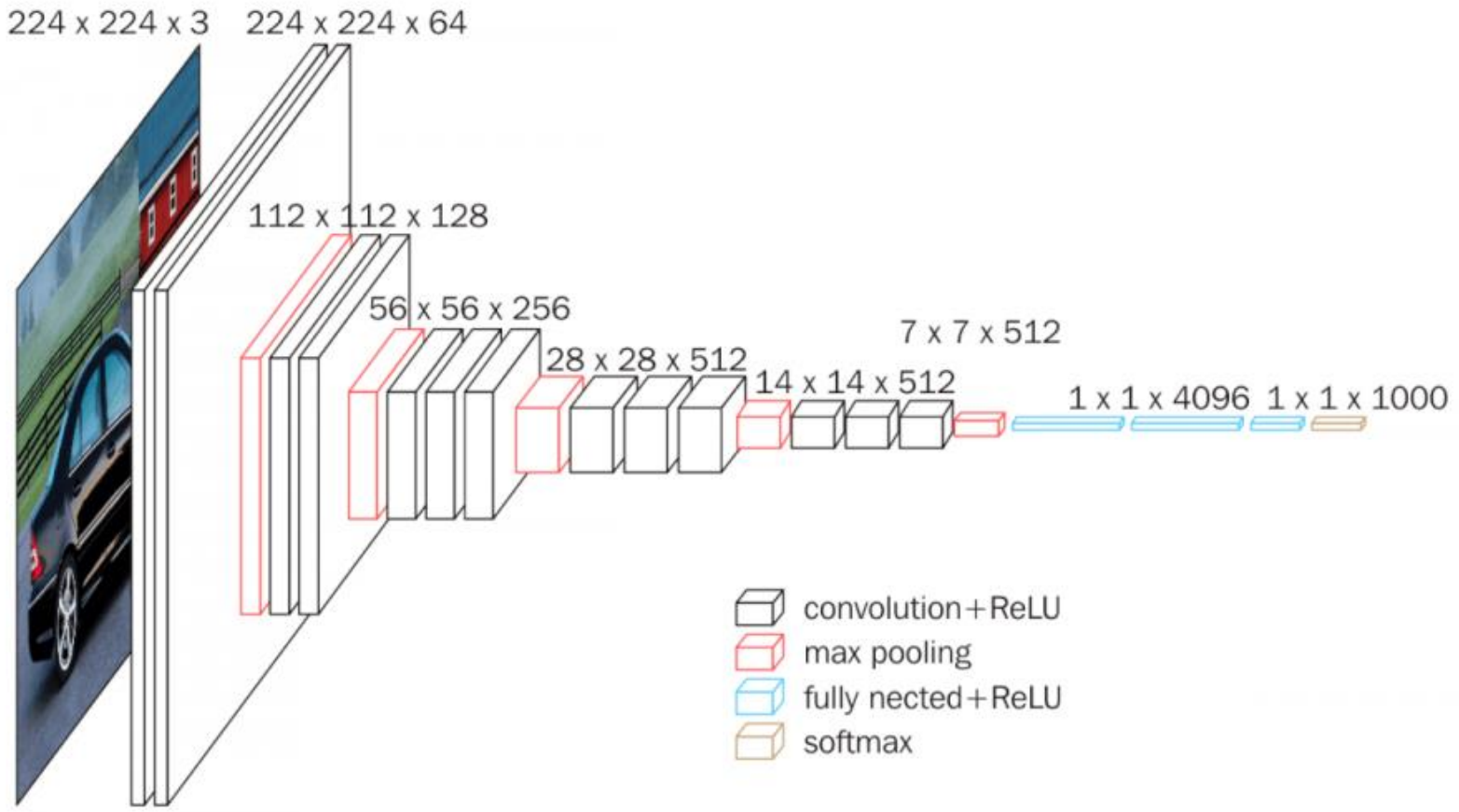
$$X_2 = \begin{bmatrix} 100 & 100 & 100 & 50 & 100 & 50 \\ 95 & 255 & 100 & 125 & 125 & 170 \\ 80 & 40 & 10 & 10 & 125 & 150 \\ 255 & 30 & 150 & 20 & 120 & 125 \\ 30 & 30 & 150 & 100 & 70 & 70 \\ 70 & 30 & 100 & 200 & 70 & 95 \end{bmatrix}$$

$\xrightarrow{\text{최대 풀링 } P_{2\times 2}}$

$$\begin{bmatrix} 255 & 125 & 170 \\ 255 & 150 & 150 \\ 70 & 200 & 95 \end{bmatrix}$$



그래서 이런 모델이…


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax