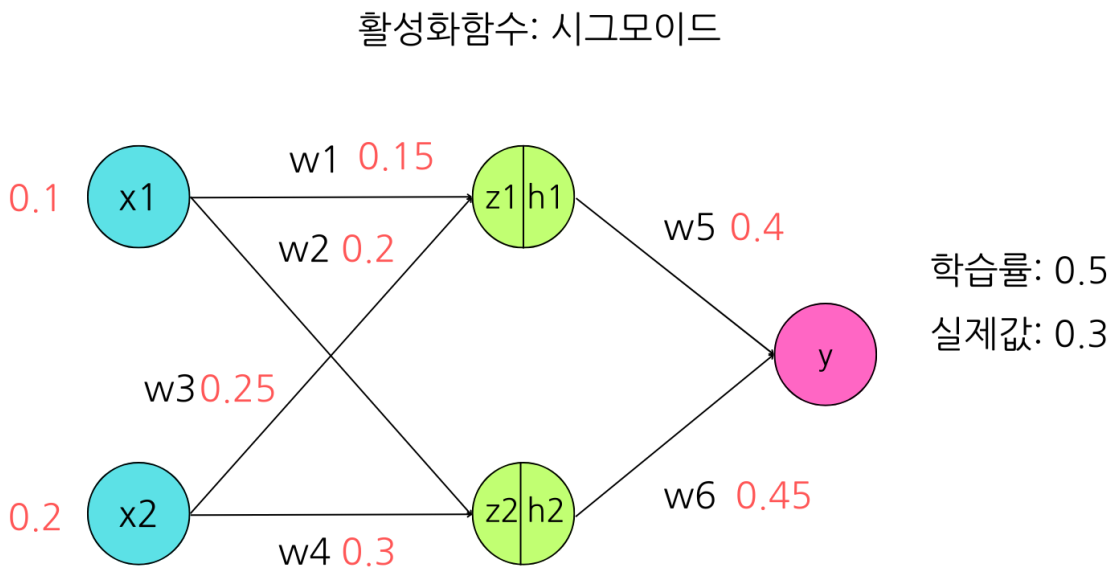
Gradient

Gradient Vanishing



최적의 W값 찾기 위해

순전파, 역전파 과정 계속 거침.

시그모이드 함수 성질 때문에

계산 과정 반복하면 점점 0에 가까워 지는 값 출력.

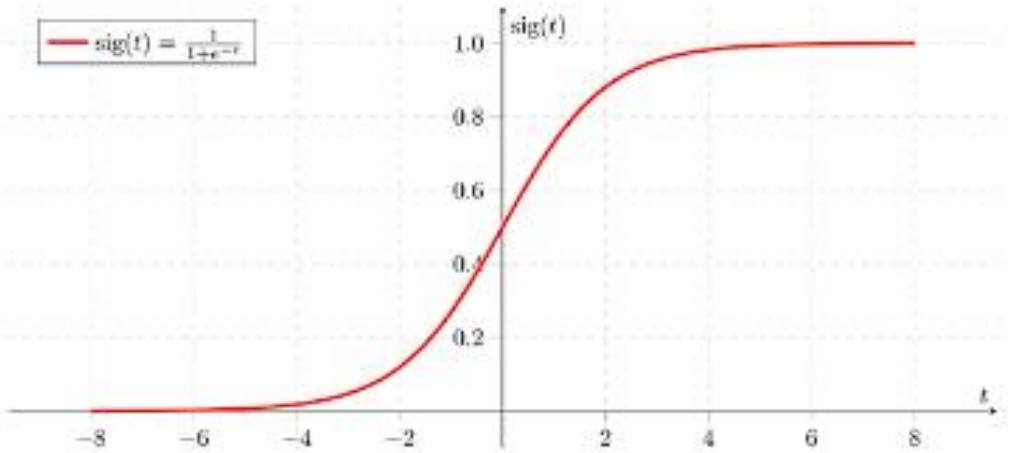
$$w \leftarrow w - \eta \cdot \text{Gradient}$$

n: 학습률

AI명예학회
SKHU

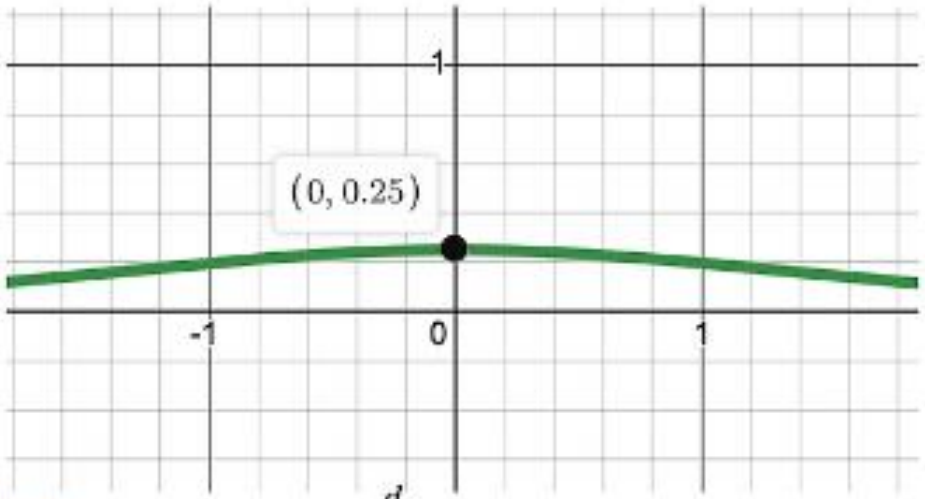
Gradient vanishing: Sigmoid

기울기 크거나 작아짐에 따라 미분 시 0 되는 형태.



sigmoid

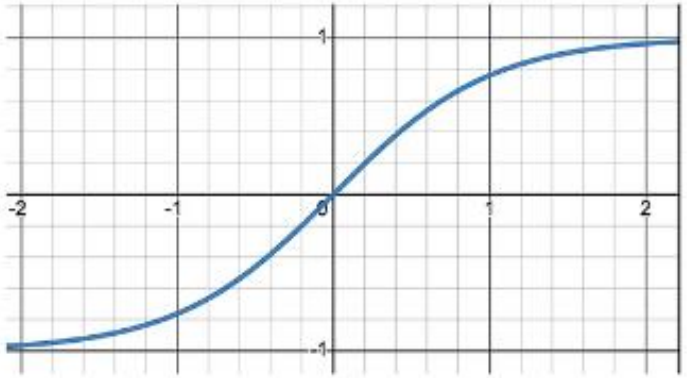
https://heytech.tistory.com/



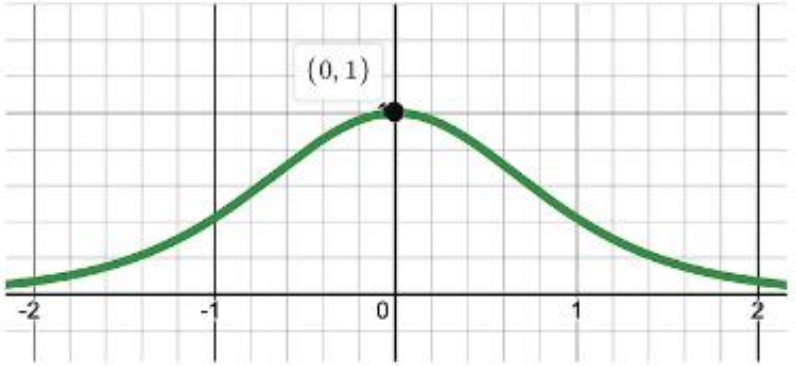
$\frac{d}{dx}sigmoid$

AI명예학회
SKHU

Gradient vanishing: tanh



$tanh$



$\frac{d}{dx}tanh$

https://heytech.tistory.com/

개선 버전.

but.

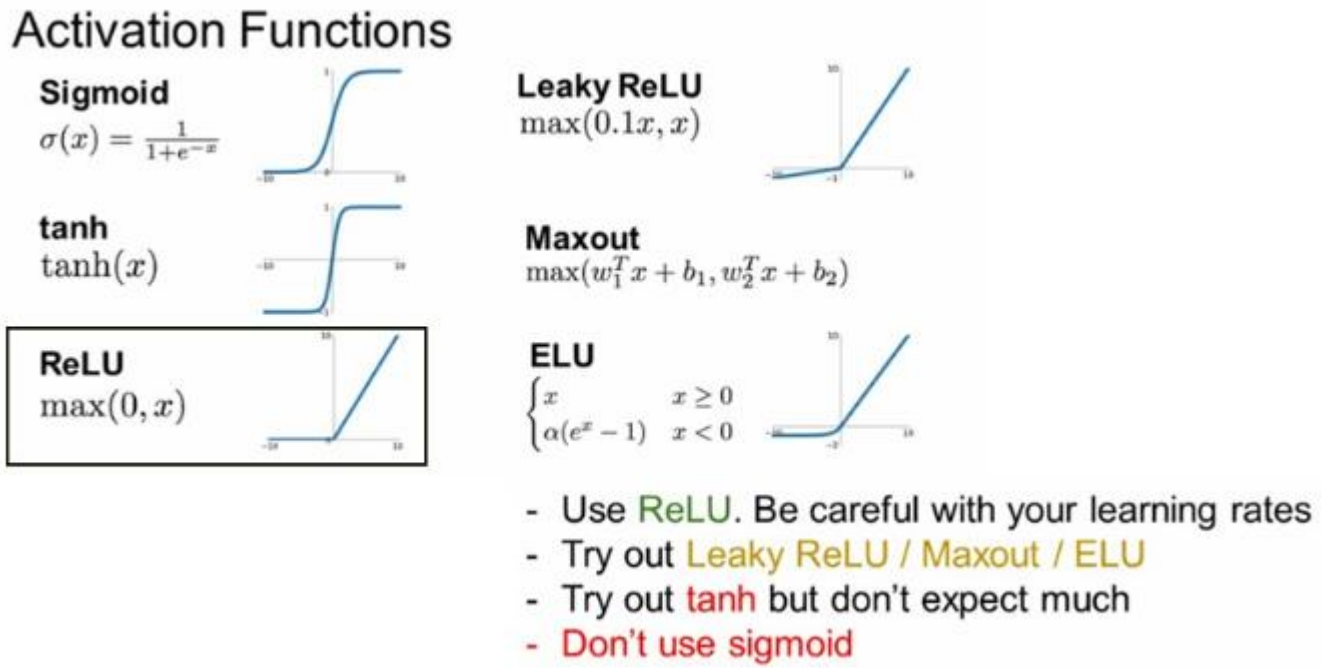
0 범위 제외하고
결국 0에 수렴

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

$$\tanh'(x) = 1 - \tanh(x)^2$$

AI명예학회
SKHU

Gradient vanishing



이와 같이 여러 개선 activation F 존재.

하지만 완전한 기울기 소실 문제는 해결 X

AI명예학회
SKHU

Optimization(경사 하강법 연장선): Momentum

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

기존 GD

x_t: 현재 위치

η: 학습률

∇f(x_t): 기울기

➔ 결국 어디로 가야하는지의 의미

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

Momentum

v_t: 현재 속도

p: 파라미터.(관성값. 보통은 0.9)

α: 학습률

∇f(x_t): 기울기

➔ 이전단계의 속도도 함께 고려.
진동 줄이면서 좀 더 빠르게

AI명예학회
SKHU

Optimization(경사 하강법 연장선): Nesterov Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t + \rho v_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

좀 더 세밀한 조정 가능.

결국 Momentum의 개선책.

v_t: 현재 속도

p: 파라미터.(관성값. 보통은 0.9)

α: 학습률

x_t + pv_t: 현재 위치에서 관성 방향으로 움직인거

AI명예학회
SKHU

Optimization: Adagrad(Adaptive Gradient Algorithm)

$$g_t = \nabla f(x_t)$$

$$G_t = G_{t-1} + g_t^2$$

$$x_{t+1} = x_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \cdot g_t$$

g_t: 현재 시간t 에서의 기울기

G_t: 각 시간마다의 기울기 제곱 누적값

α: 초기 학습률

ϵ: 어쩔 수 없는 오차

결국 G_t값 커질 수록 학습률 점차적 감소.

고정된 학습률이 아닌, 바뀌는 학습률.

➔ 결국 점점 가중치 변화 줄어듦

오래 학습할 수록 기울기 0 수렴.

AI명예학회
SKHU

Optimization: RMSProp(Root Mean Square Propagation

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

EMA(Exponential Moving Average)

$$x_{t+1} = x_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

g_t: 현재 시간t 에서의 기울기

γ: 감쇠율(이전의 기울기 정보 얼마나 넣을지. 보통 0.9로 둠)

x_t: 현재 단계 가중치 값

α: 초기 학습률

$$E[g^2]_t = (1-\gamma) g_t^2 + \gamma(1-\gamma) g_{t-1}^2 + \gamma^2(1-\gamma) g_{t-2}^2 + \dots$$

결국은 현재 기울기에 더 큰 가중치 부여.

최근의 기울기 더 반영. 학습률 너무 빠르게 감소하지 않도록 작동.

➔ Adagrad 개선책

AI명예학회
SKHU

Optimization: Adam(Adaptive Moment Estimation)

1. $$g_t = \nabla f(x_t)$$

2. $$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

3. $$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

4. $$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

5. $$x_{t+1} = x_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

g_t: 현재 시간t 에서의 기울기

m_t: 기울기의 평균. 여러 단계의 기울기들 고려 가능 , 여기서는 Momentum 역할

v_t: 기울기 변화량. 여기서는 RMSProp 역할

B_1, B_2: 파라미터 값, 보통 0.9, 0.99 로 설정

x_t: 현재 단계 가중치 값

α: 초기 학습률

AI명예학회
SKHU

Optimization: Adam(Adaptive Moment Estimation)

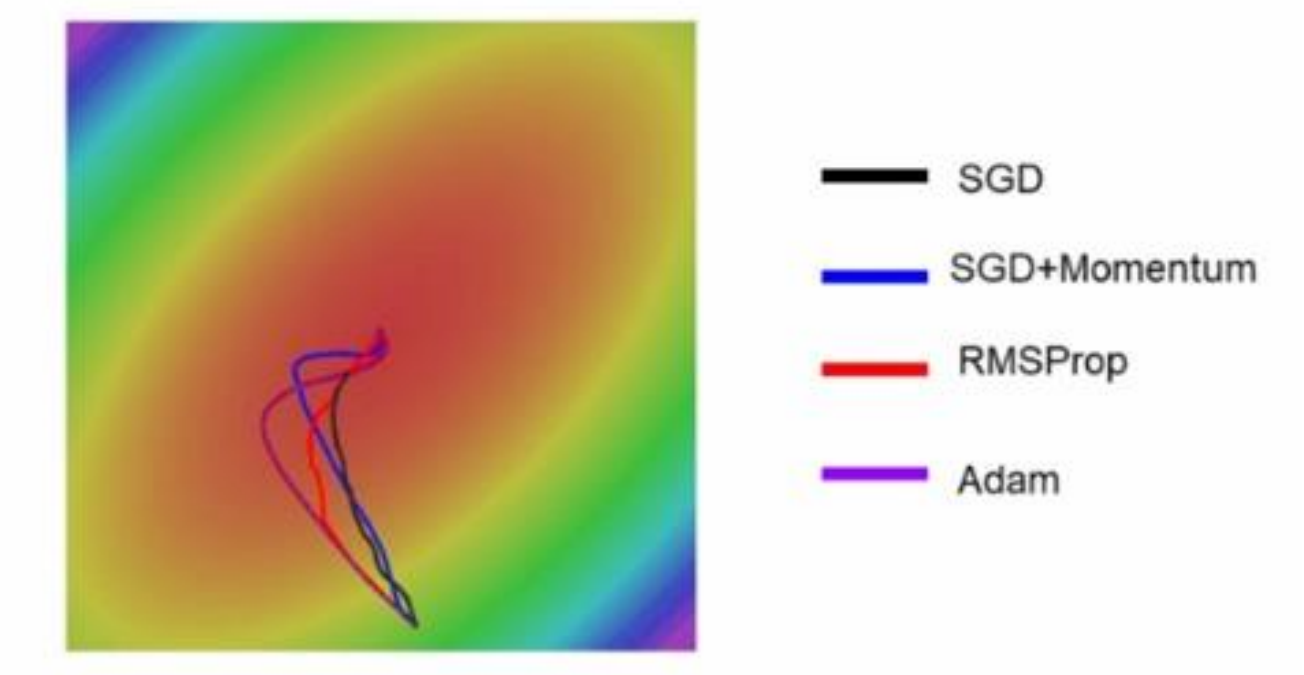
각 단계마다 학습률 자동 조정 좀 더 빡세게 됨.

다만,

초기 설정값들이 생각보다 많아서 감이 좋아야 한다고 하는데 그냥 default 값 넣고 해도 잘 됨.

그래서 딥러닝에서 많이 씀.

AI명예학회
SKHU



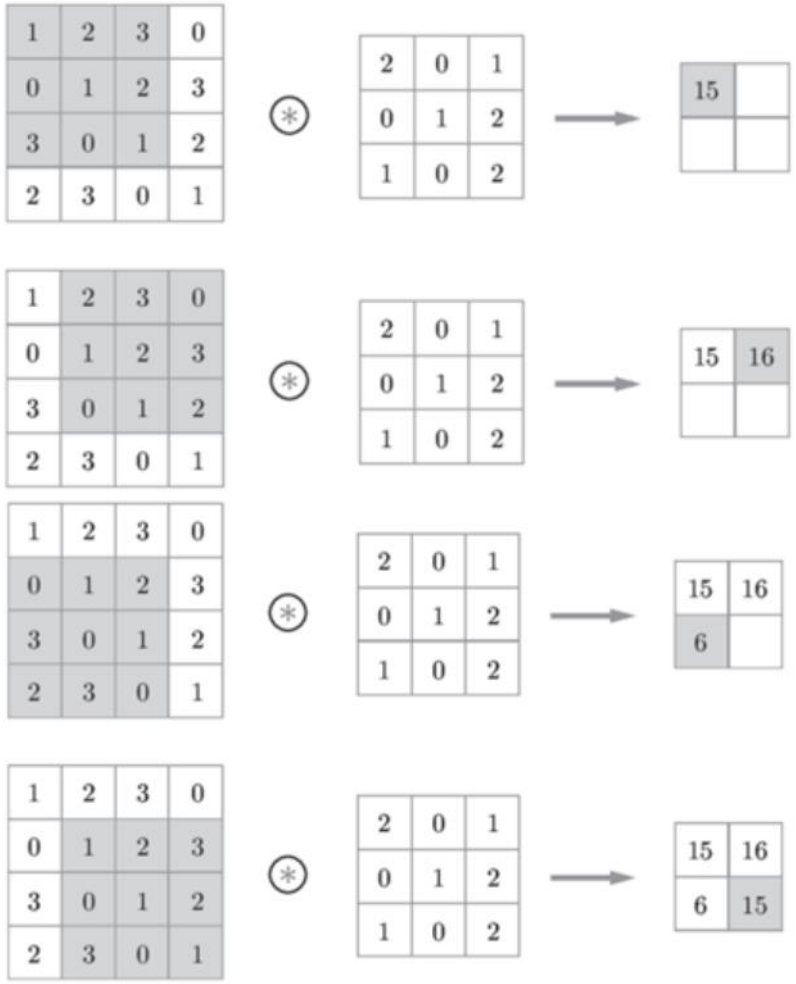
https://www.youtube.com/watch?v=_JB0AO7QxSA&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=7&ab_channel=StanfordUniversitySchoolofEngineering

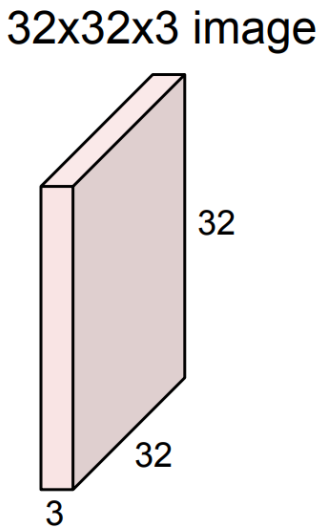
AI명예학회
SKHU

CNN(Convolution Neural Network)

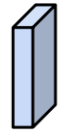
AI명예학회
SKHU

Conv 계층 연산





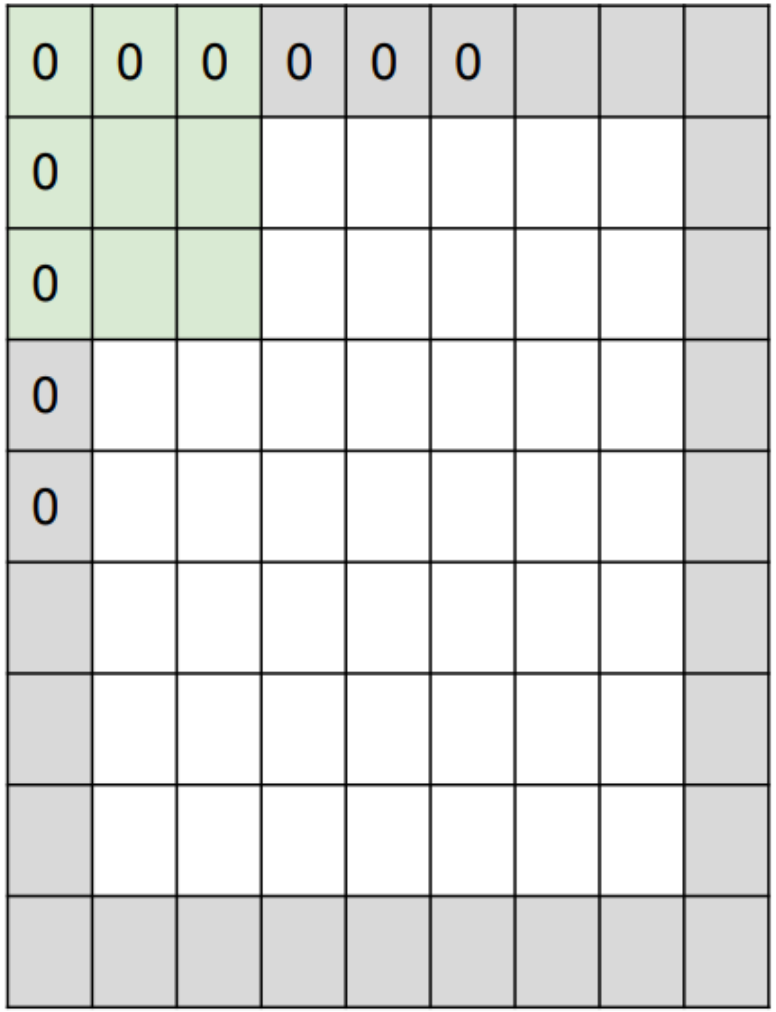
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

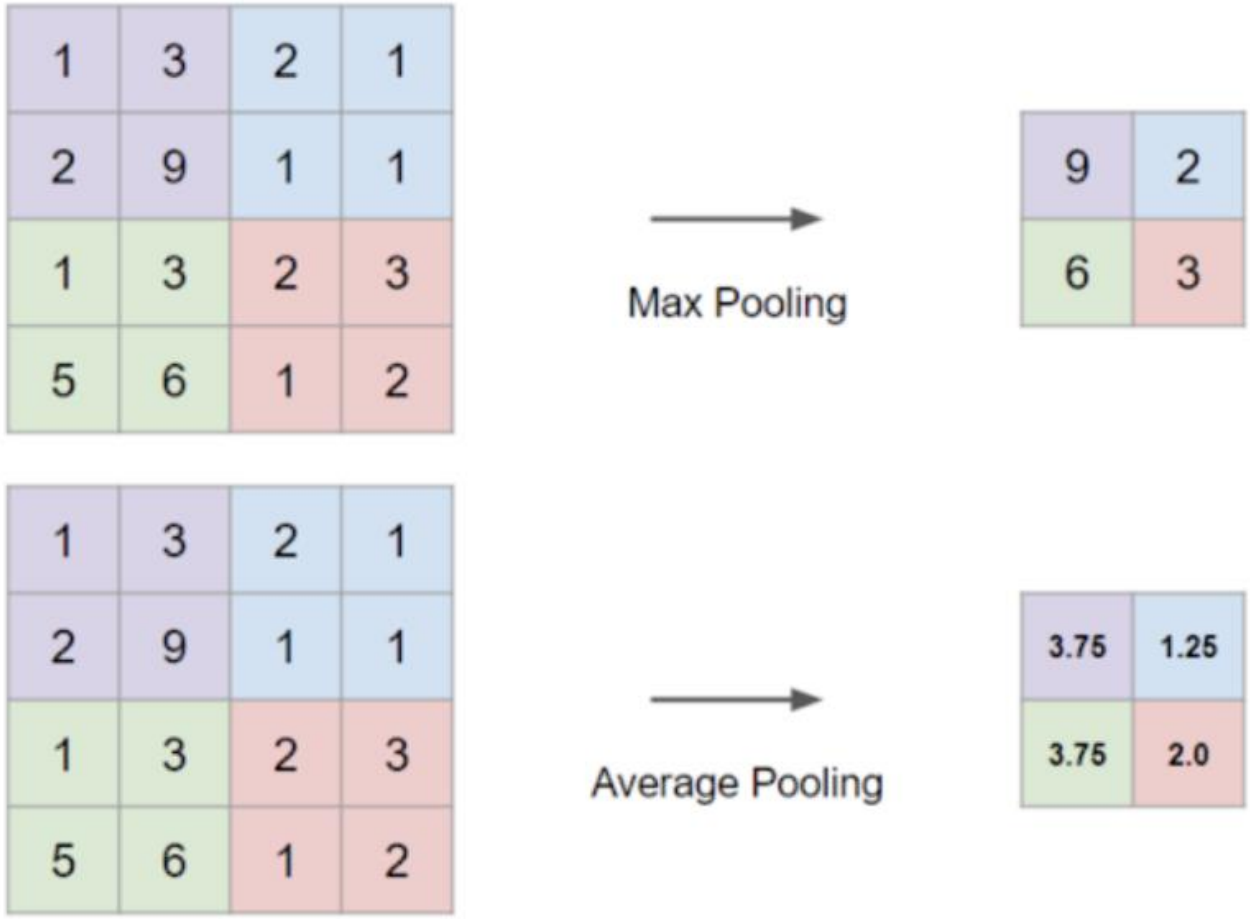
AI명예학회
SKHU

Zero padding



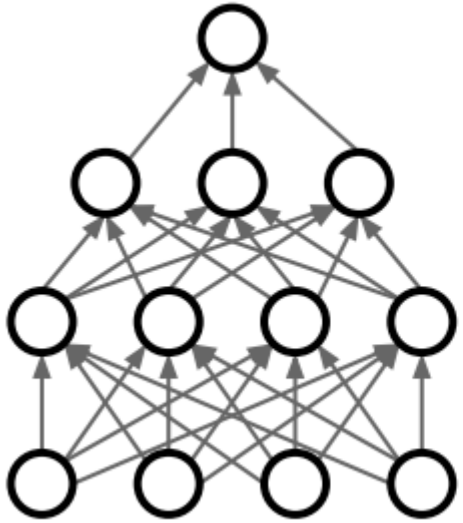
AI명예학회
SKHU

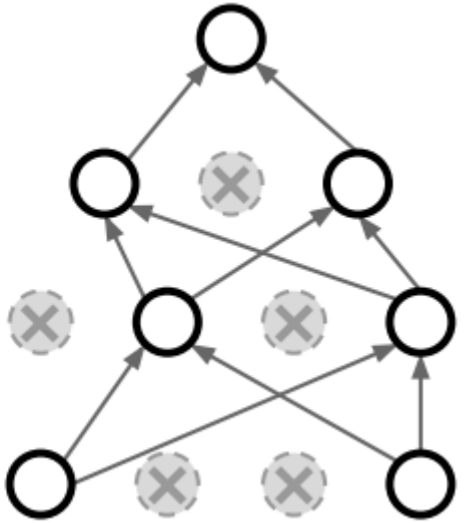
Pooling



AI명예학회
SKHU

Dropout





AI명예학회
SKHU

Assignment

수업 자료 정리

Conv 계층 7개 이상 사용

FC 계층 1~3개 사용

BatchNormalization, maxPooling, Dropout 필수 사용

Train, test 정확도 65% 이상 확보

모델 학습 시간 10분 이하

AI명예학회
SKHU