ImageNet Classification with Deep Convolutional Neural Networks

SKT Fellowship

LEE JINKYU

Department of Civil, Environmental and Architectural Engineering, Korea University

November 19, 2022

SKT AI Fellowship

ABOUT AlexNET

History of ILSVRC competition

2010 - NEC-UIUC (Lin et al.)
2011 - XRCE (Florent Perronnin, Jorge Sanchez)
2012 - **AlexNet**
2013 - ZFNet
2014 – 1st **GoogLeNet**, 2st **VGGNet** (VGGNet이 준우승을 하긴 했지만,
구조의 간결함과 사용의 편이성으로 인해 GoogLeNet보다 더 각광받았다.)
2015 - **ResNet**
2016 - GoogLeNet-v4
2017 - SENet

ILSVRC-2012 대회 우승 >> **이미지 인식에 있어 CNN을 본격적으로 사용하게 됨**

이전 ILSVRC 우승 모델은 모두 **얕은 구조**(얕은 신경망, 단순히 INPUT, HIDDEN, OUTPUT으로 구성,
대표적으로 2011에는 BOV, SVM 등을 사용)로 개발자들이 유용할 것 같은 특성을 결정해 도출하는 것이었음

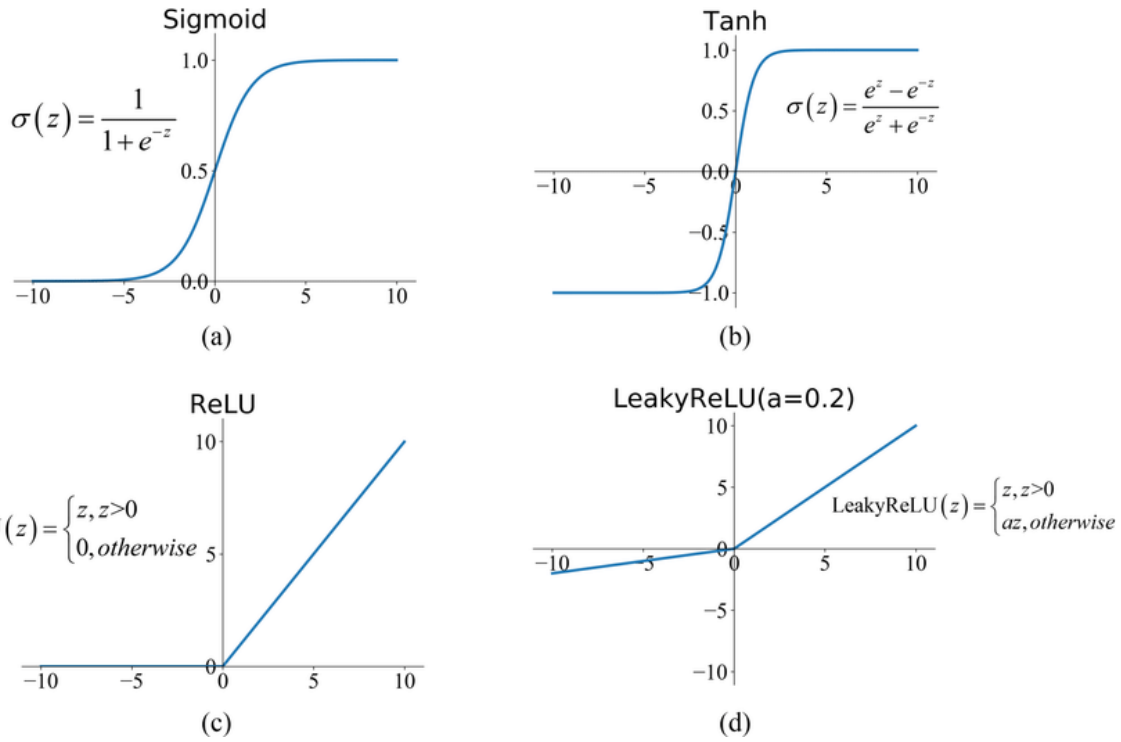
>> **CNN은 매우 깊은 구조**로 0.1% 오류를 낮추는 것도 힘들었던 얕은 모델에 비해 26%에서 16%로 오류를 낮춤

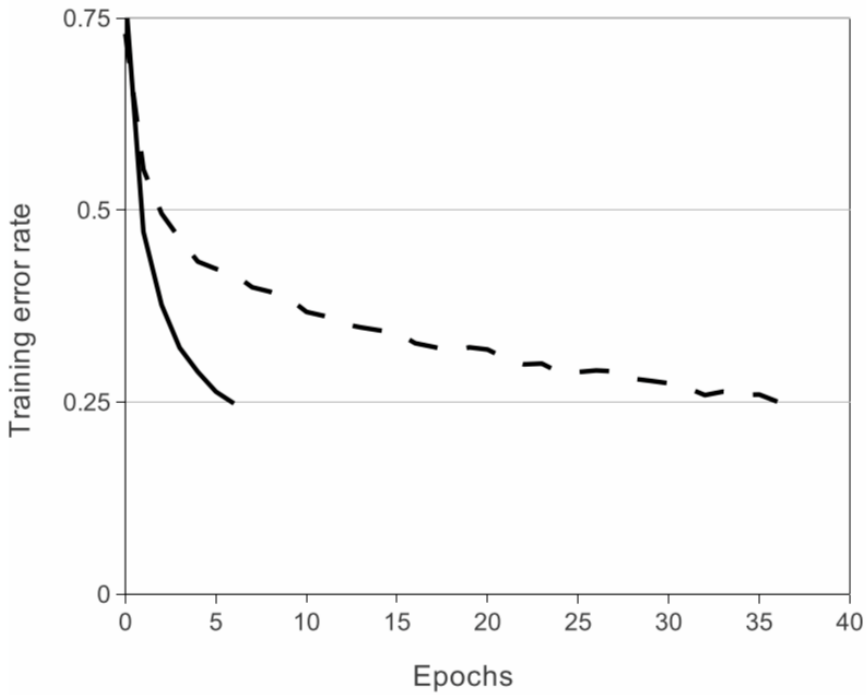
Architecture - ReLU

Why Alexnet use ReLU Activation Fuction ?

기존 CNN의 활성화 함수 : Sigmoid, tanh

Relu
- max(0,x) 기본적으로 단순해 순전파, 역전파 시 **계산이 빠름**
- 모델 실행 시간을 단축시켜 **깊은 모델을 만들 수 있게 함**



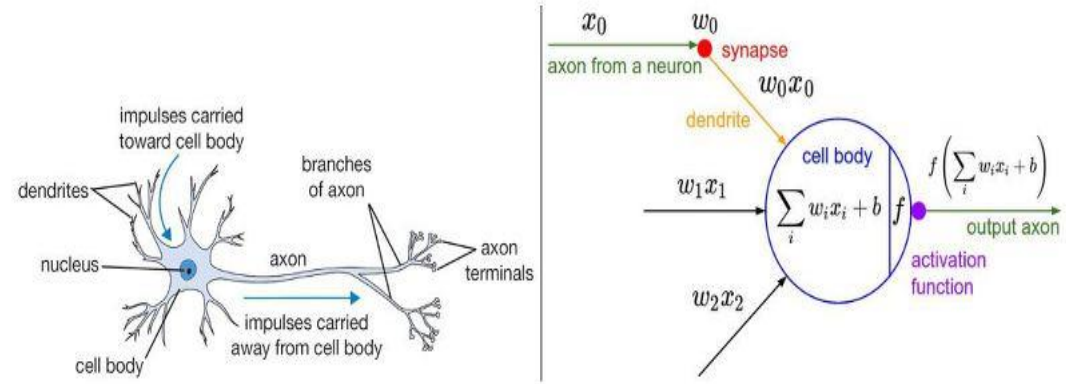


Architecture - ReLU

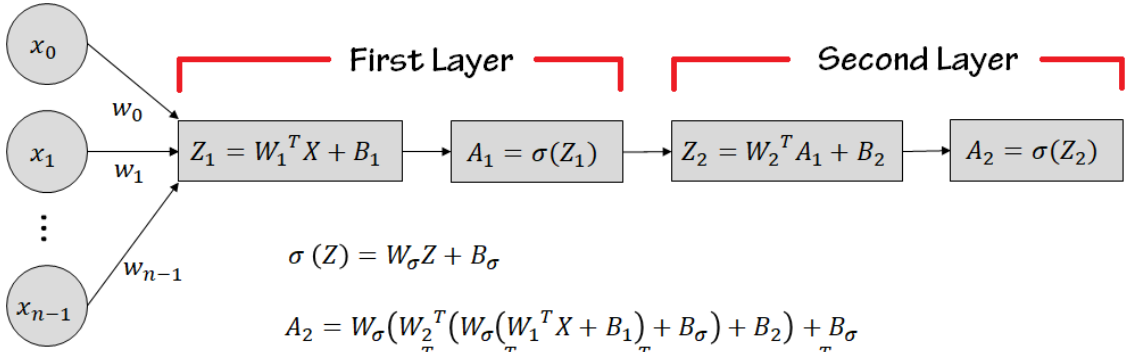
What is Activation Function ?

***활성화 함수란 ? :**

노드에서 계산된 값을 다음 레이어로 넘겨주기 전에 처리하는 함수
보통 비선형 함수를 사용하고, 이를 통해 깊은 네트워크 사용 장점을 극대화 함



A cartoon drawing of a biological neuron (left) and its mathematical model (right).



$$\sigma(Z) = W_\sigma Z + B_\sigma$$

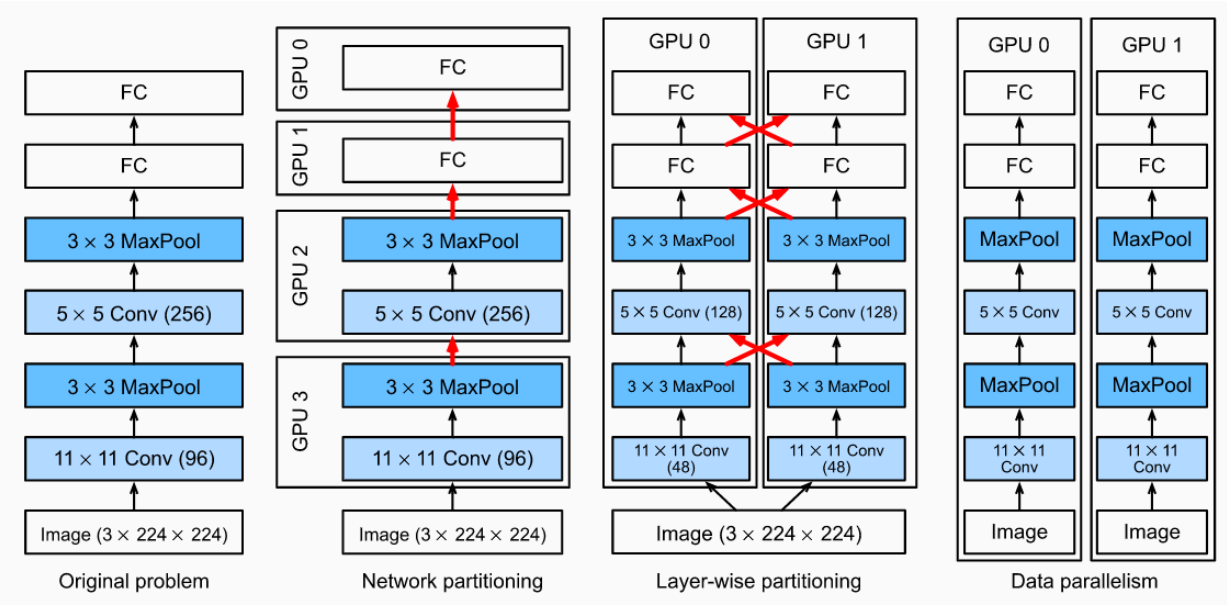
$$\begin{aligned} A_2 &= W_\sigma\left(W_2{}^T\left(W_\sigma\left(W_1{}^T X + B_1\right) + B_\sigma\right) + B_2\right) + B_\sigma \\ &= \underbrace{W_\sigma W_2{}^T W_\sigma W_1{}^T}_{W} X + \underbrace{W_\sigma W_2{}^T W_\sigma B_1 + W_\sigma W_2{}^T B_\sigma + W_\sigma B_2 + B_\sigma}_{B} \\ &= WX + B \end{aligned}$$

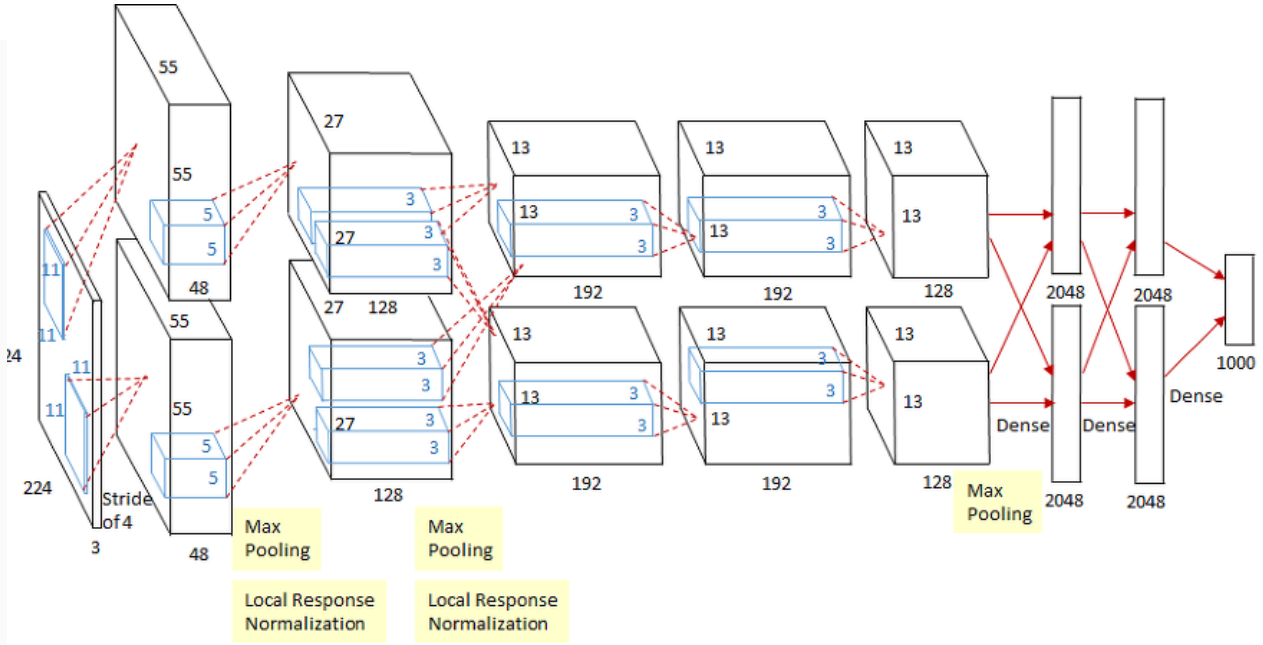
Architecture - Multi GPUs

To overcome GPU memory

GPU Parallelization을 통해 실행 가능

- channel size를 분할해 각 GPU에 적용
- Synchronization이 필요함
- GPU 성능에 비해 훨씬 큰 네트워크 사용 가능
- 각 GPU를 독립적으로 실행하되 몇몇 layer들을 연결함





Architecture - LRN (local response normalization)

Lateral inhibition to use ReLU

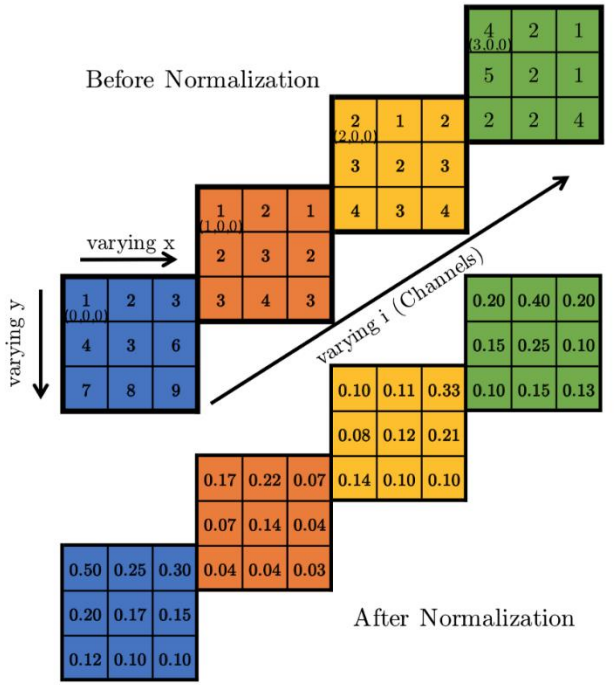
Lateral inhibition(측면 억제)

한 영역에 있는 신경 세포가 상호간 연결되어 있을 때 그 자신의 축색이나 자신과 이웃 신경세포를 매개하는 중간 신경세포를 통해 이웃에 있는 신경 세포를 억제하려는 경향

Alexnet에서는 sigmoid, tanh와는 다르게 양수 방향의 입력값을 그대로 반영하기 때문에 하나의 큰 픽셀값이 주변에 영향을 미칠 수 있음

>> LRN을 통해 같은 위치의 픽셀끼리 정규화

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$



Architecture - Overlapping Pooling

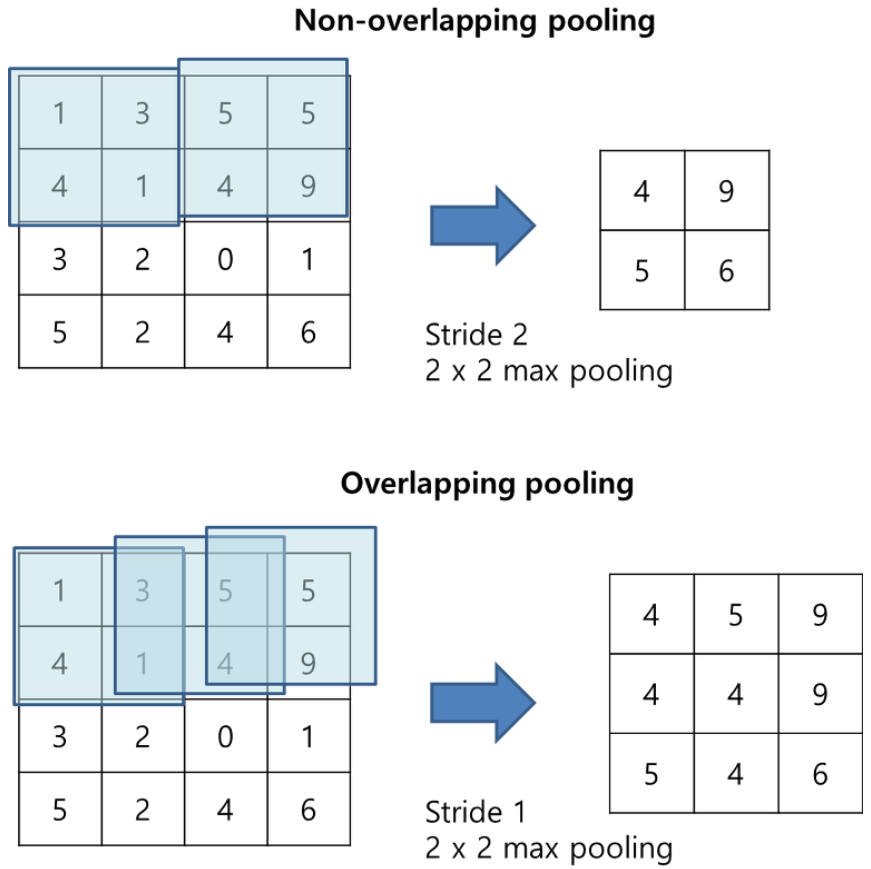
Make Pooling captures as many features as possible

Pooling (풀링)

데이터의 주요 특성만을 반영하는 것 (Sub sampling)
과대적합을 방지

In Alexnet

Overlapping pooling을 통해 정확도는 향상
(당연히 계산량은 늘었다)
Non Overlapping에 비해 과대적합 방지
(why?? 파라미터가 줄어들어야 과대적합 방지 아닌가..? -> 과대적합 방지로서의 역할은 거의 없는 듯 + 정확도도 size에 따라 천차만별..! 확인해봐야할 듯)

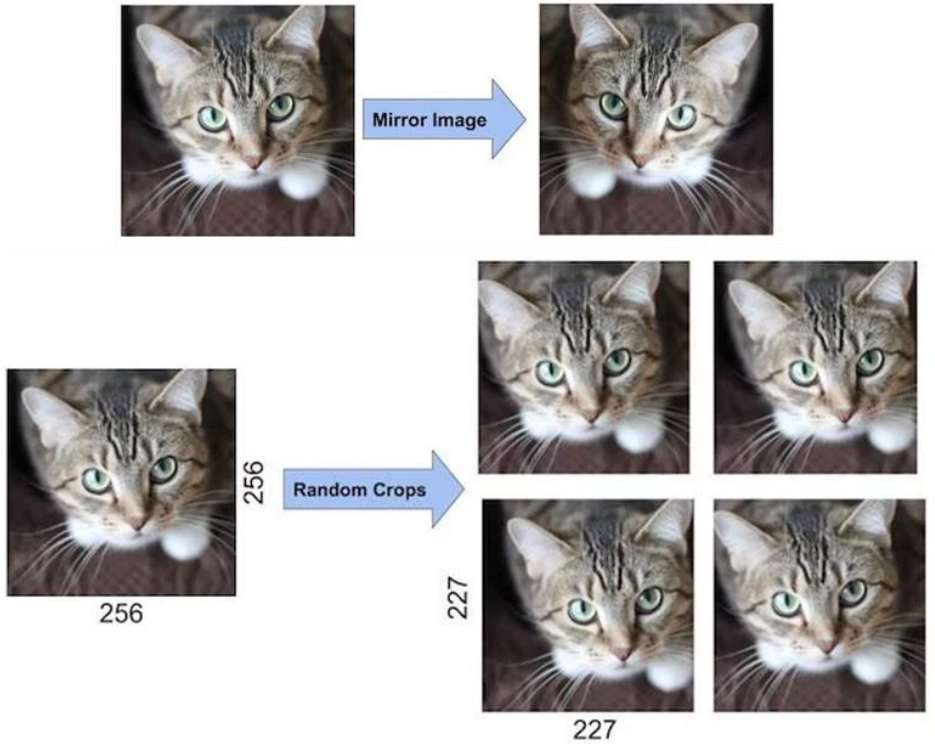


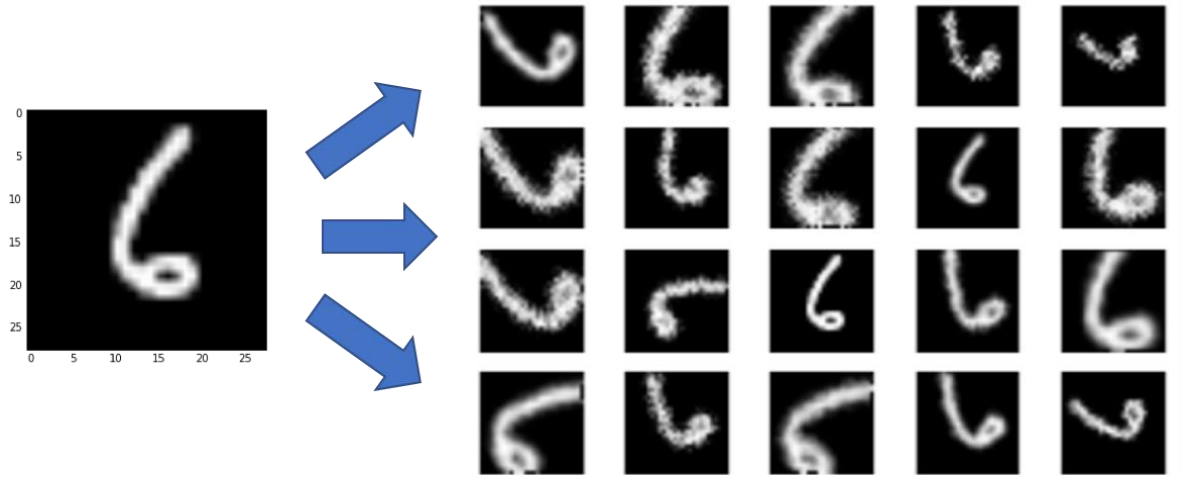
Reduce Overfitting – Data Augmentation_Crop

Prevent Overfitting with mirror, flip, rotate, crop

Mirror & Crop

이미지를 좌우 반전한 후 반전 이미지에 대해 작은 사이즈의 이미지로 Crop
Mirror, Rotate에 의해 의미가 아예 변해버리는 이미지도 존재, label preserved transformation 필요
>> 그럼 mnist의 6이나 9는 augmentation 했을 때 성능이 오히려 떨어질까?





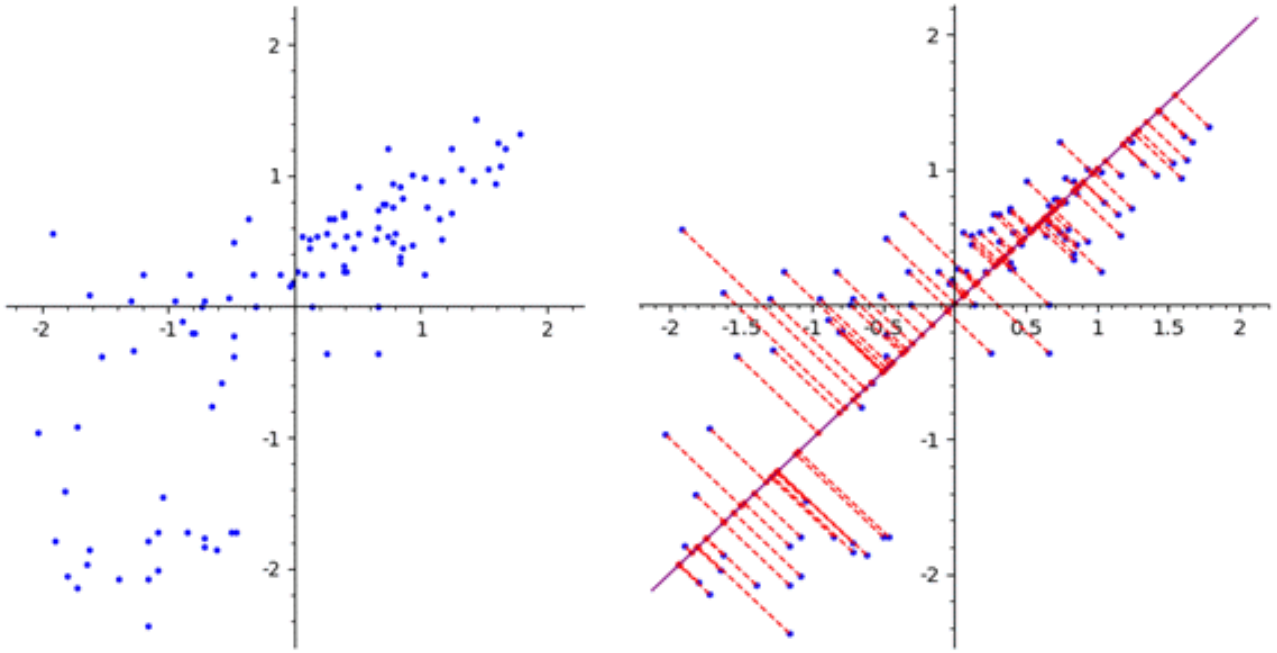
꼭 구현해보자 !

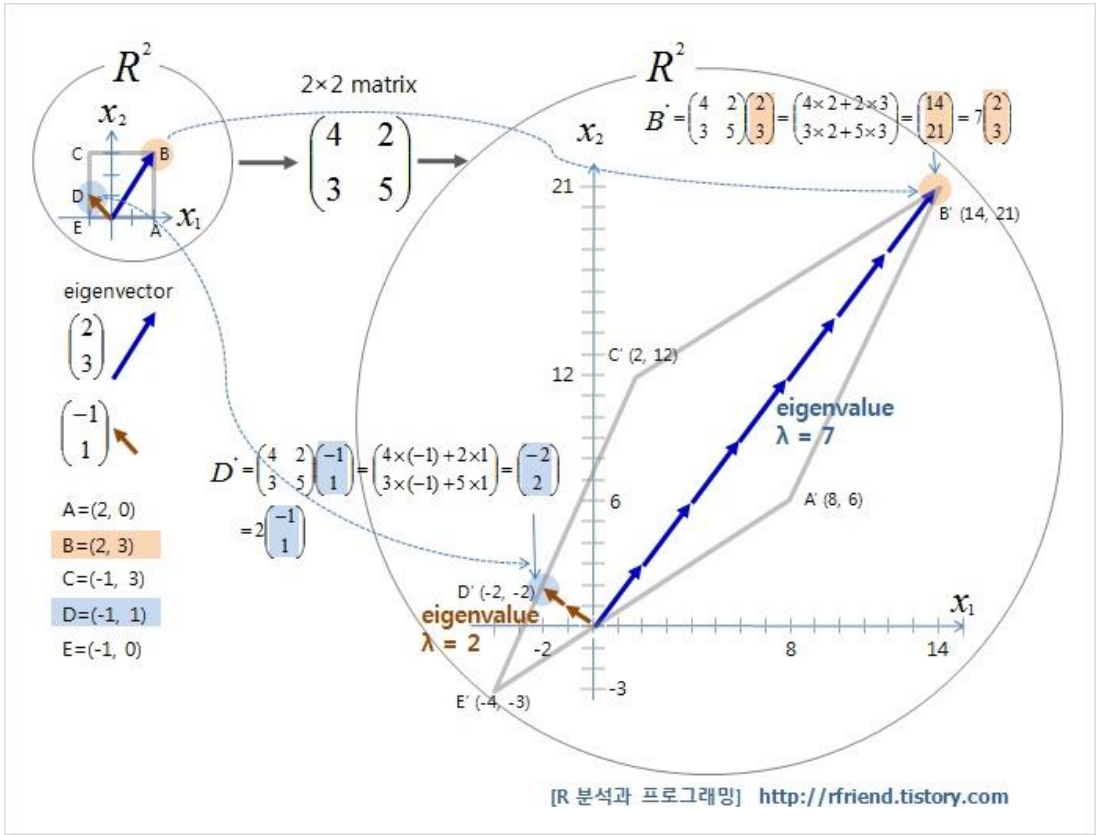
Reduce Overfitting – Data Augmentation_PCA

Eigenvector, Eigenvalue from PCA

PCA

주성분 추출, 원 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터를 저차원으로 변환
이때, 고유값 분해를 통해 고유값과 고유 벡터를 구하여 저차원 변환
(고유 벡터 : 정방 행렬 A를 통한 왜곡에도 방향을 유지하는 벡터, 고유 값 : 고유 벡터으로의 분산)





꼭 구현해보자 !

Reduce Overfitting – Data Augmentation_PCA

Eigenvector, Eigenvalue from PCA

PCA의 고유값과 고유 벡터를 통해 주성분을 유지하면서 색상을 변환시키는 증강 기법

$$I_{xy} = [I_{xy}^R,\ I_{xy}^G,\ I_{xy}^B]^T + [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

$$\alpha_i \sim N(0,\ 0.1)$$

original

Eigenvectors (column vectors)

vec1	vec2	vec3
0.581	-0.152	-0.799
0.613	0.729	0.306
0.536	-0.668	0.517

Eigenvalues (relative to maximum)

val1	val2	val3
1	0.136	0.028

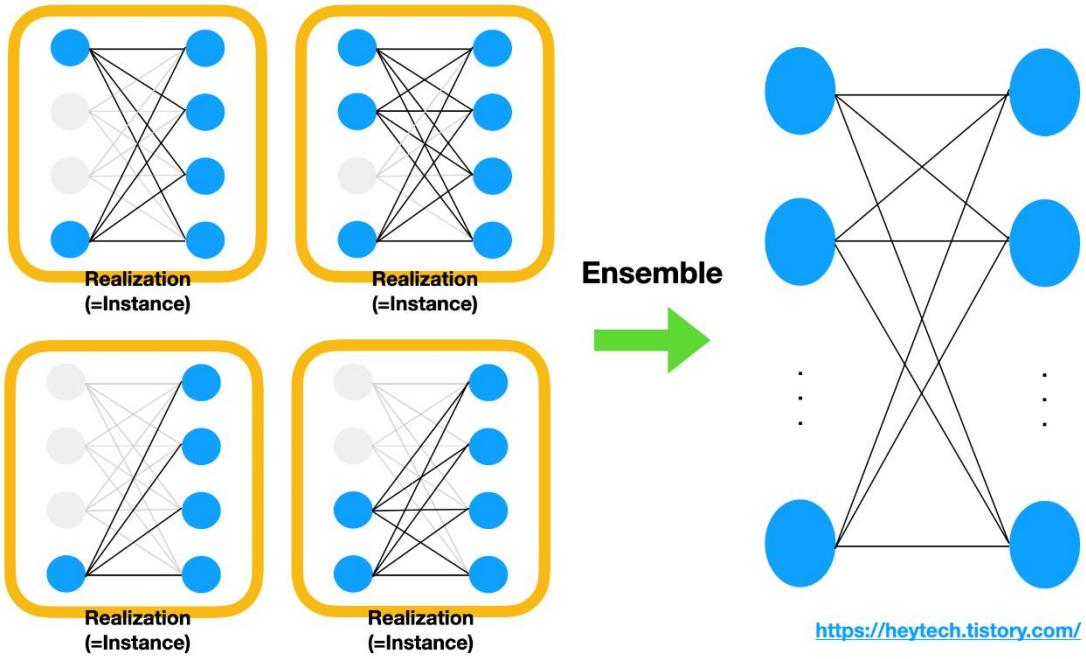
<사진9>

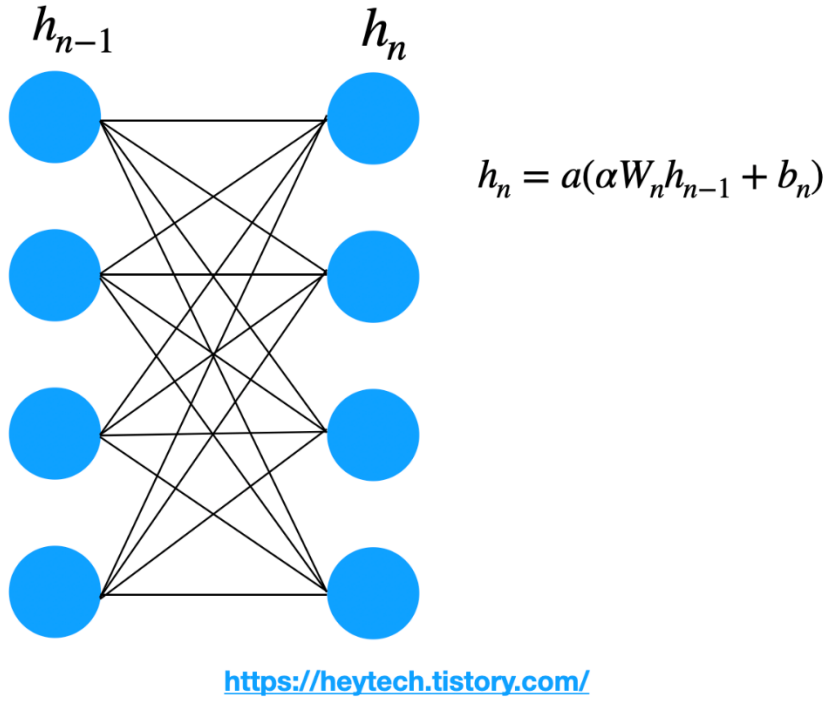
Reduce Overfitting – Dropout

Strong method to prevent overfitting

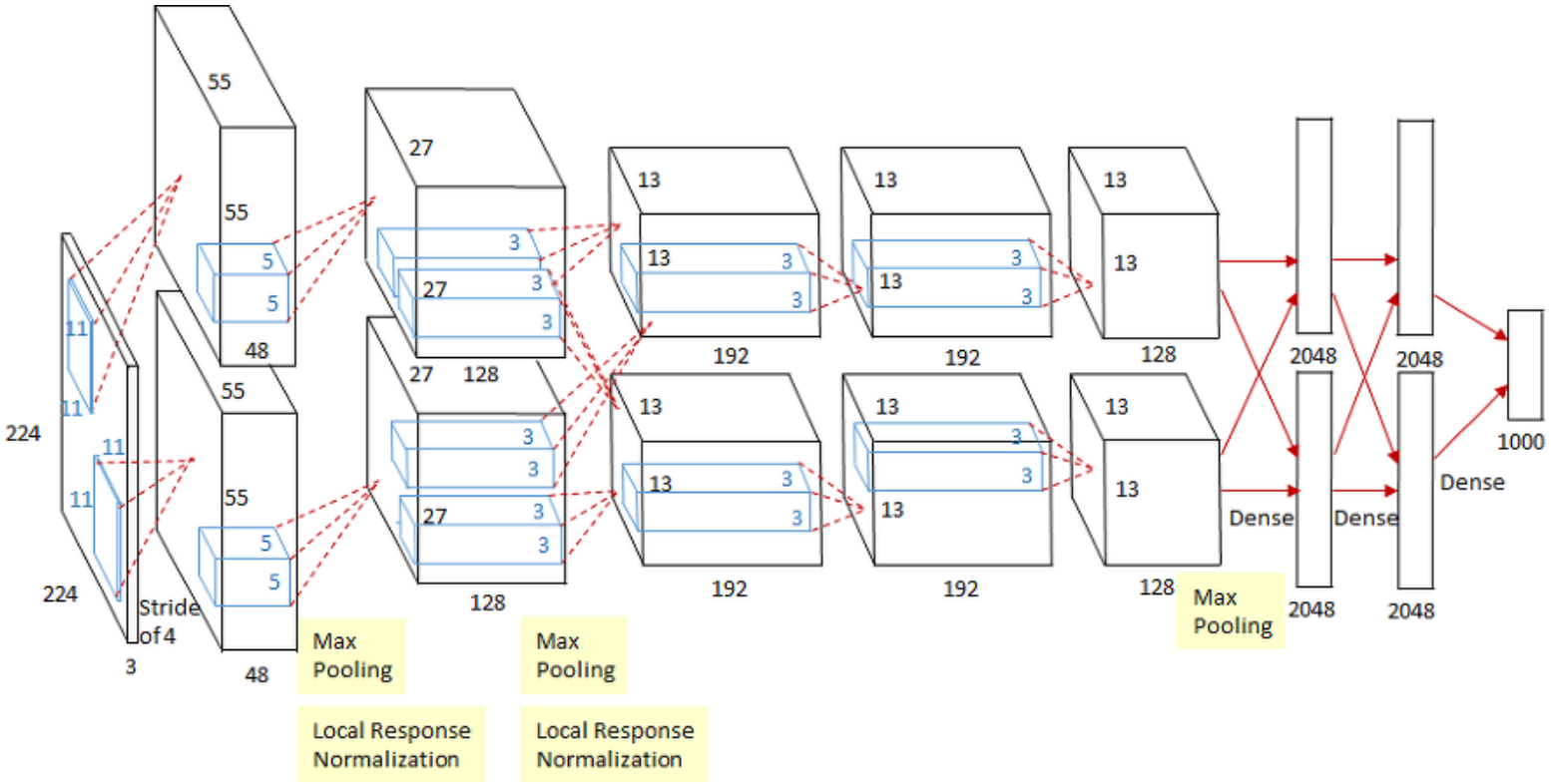
Dropout

뉴런을 랜덤하게 생략하고 학습하는 것, 많은 모델을 사용, 앙상블하지 않고도 그와 같은 효과를 낼 수 있음
(특정 뉴런에 대한 과대적합 방지)
생략된 구조는 서로 Weight를 공유하고 있으며, test-set에 적용시에는 weight 전체를 사용
(대신 dropout_rate을 곱해준다)





Alexnet



Type	Filter	stride	padding	Input	output	remarks
Input layer	-	-	-	-	-	-
Conv1	11 x 11 x 3 x 96	4	0	224 x 224 x 3	55 x 55 x 96	-
MaxPool1	3 x 3	2	-	55 x 55 x 96	27 x 27 x 96	-
Norm1	-	-	-	27 x 27 x 96	27 x 27 x 96	LRN을 이용한 normalization layer
Conv2	5 x 5 x 96 x 256	1	2	27 x 27 x 96	27 x 27 x 256	-
MaxPool2	3 x 3	2		27 x 27 x 256	13 x 13 x 256	-
Norm2	-	-	-	13 x 13 x 256	13 x 13 x 256	LRN을 이용한 normalization layer
Conv3	3 x 3 x 256 x 384	1	1	13 x 13 x 256	13 x 13 x 384	-
Conv4	3 x 3 x 384 x 384	1	1	13 x 13 x 384	13 x 13 x 384	-
Conv5	3 x 3 x 384 x 256	1	1	13 x 13 x 384	13 x 13 x 256	-
MaxPool3	3 x 3	2	-	13 x 13 x 256	6 x 6 x 256	-
FC1	-	-	-	6 x 6 x 256	4096	-
FC2	-	-	-	4096	1000	-
Output layer	-	-	-	-	-	-