DATA
SCIENCE LAB

CNN 3

23.02.21 / 8기 유채원

0. INTRO

CNN1

Computer Vision이란? CNN은 왜 등장했을까? CNN 기본 개념

Convolutional Layer / Padding, Stride / Batch Normalization, …

CNN2

지금까지는 **어떤 CNN 모델**들이 있었지? CNN 모델의 발전과정

CNN Architecture / Comparing Model Complexity

CNN3

그래서 **성능 좋은 (CNN) 모델**을 만들려면 어떻게 해야 되는데?

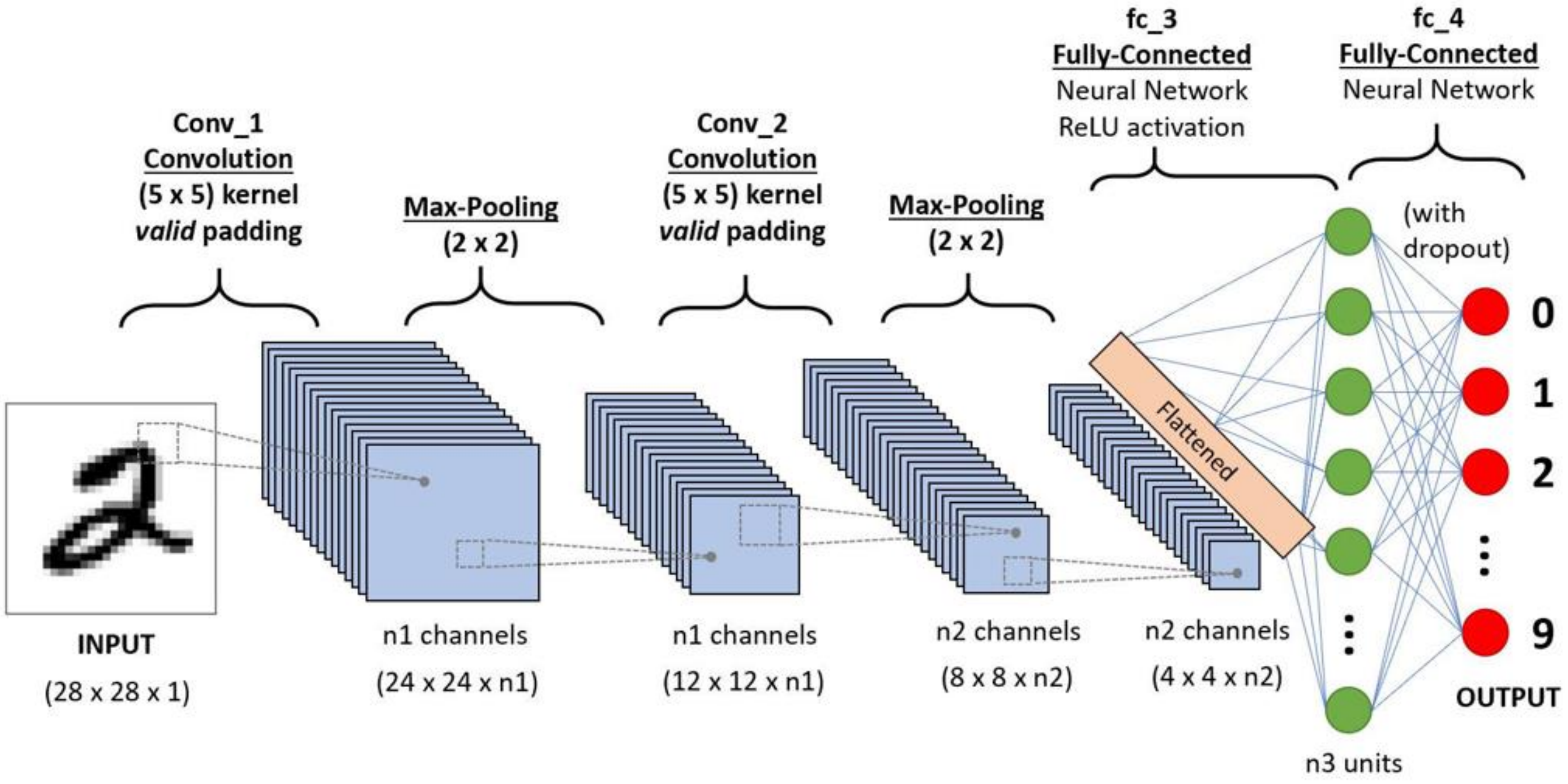
CNN을 이용하는 이유 / **전이학습(Transfer Learning)**

CONTENTS

CONTENTS

성능 좋은 모델을 만들기 위해서는?

0. CNN 복습



A CNN sequence to classify handwritten digits

0. CNN 복습

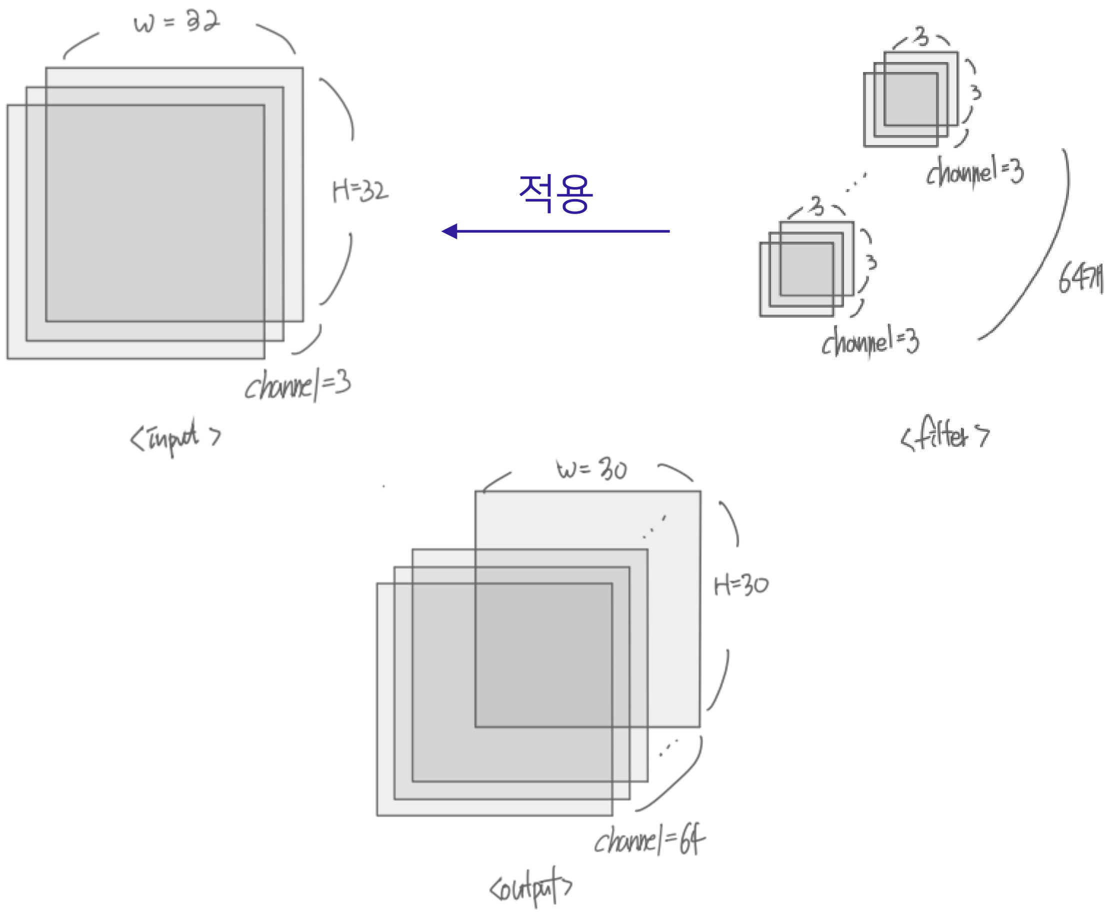
A. Feature Extractor

Feature Extractor는 아래 과정의 반복

Conv Layer → Batch Normalization → Activation func (ReLU, Sigmoid, …) → (Dropout) → (Pooling)

0. CNN 복습

A-1. Convolutional Layer



Input : 32 * 32 * 3

3 * 3 Filter 64개 적용

Output : 32 * 32 * 64 (Stride=1, Padding

X

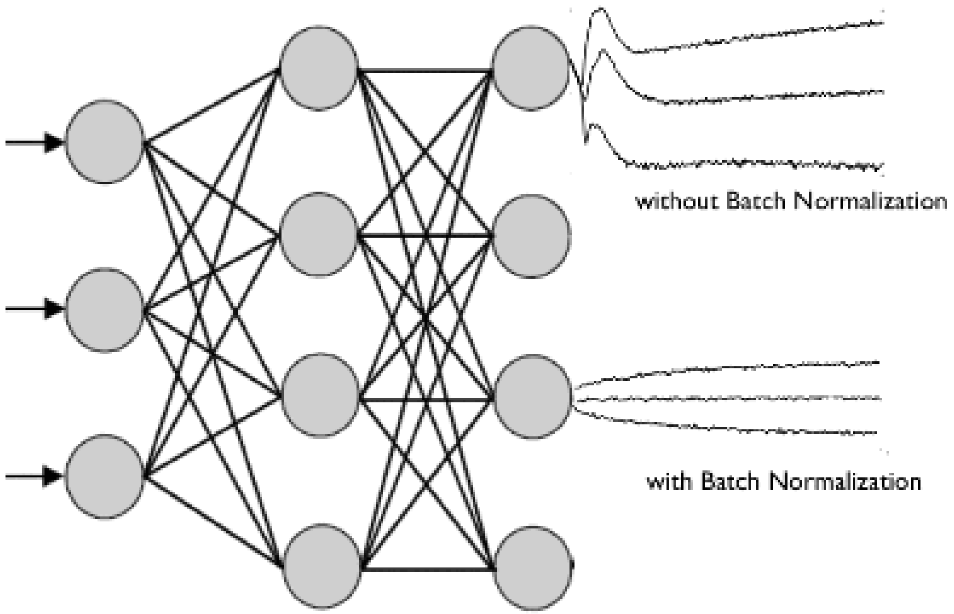
$$dim_{out} = \frac{N - F + 2P}{S} + 1$$

0. CNN 복습

A-2. Batch Normalization

미니배치의 평균과 분산을 이용하여 **정규화**한 후

scale(γ) 및 shift(β)를 한 후 activation function에 적용



BN is done neuron-by-neuron

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

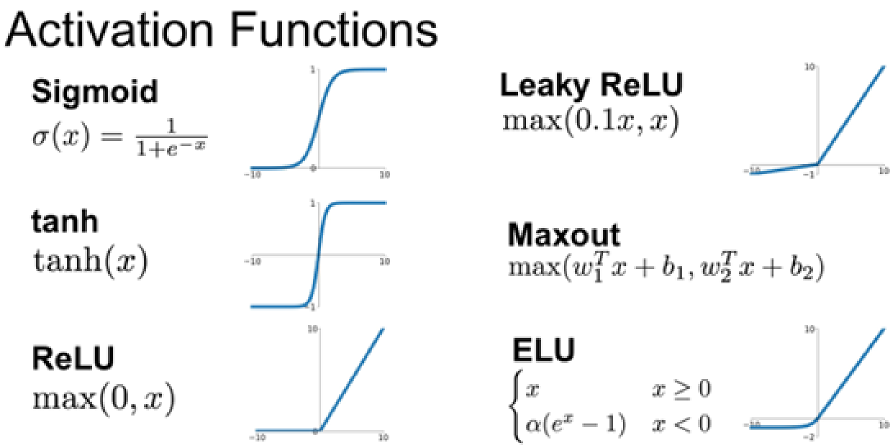
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

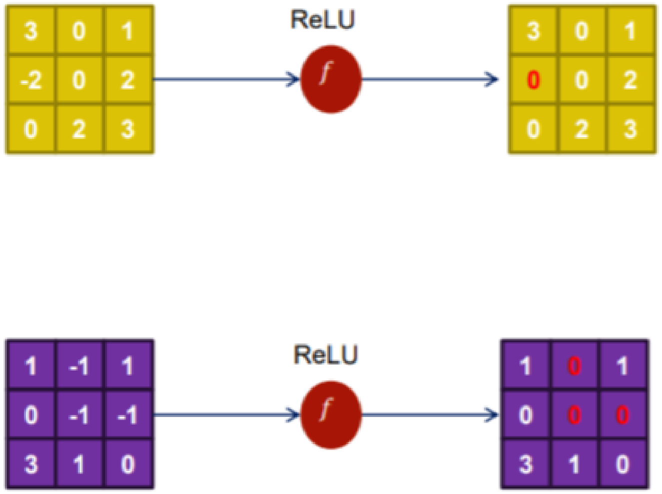
0. CNN 복습

A-3. Activation Functions

- Per-element on activation map (CNN)
- 원래 NN 필수 요소
- For non-linearity

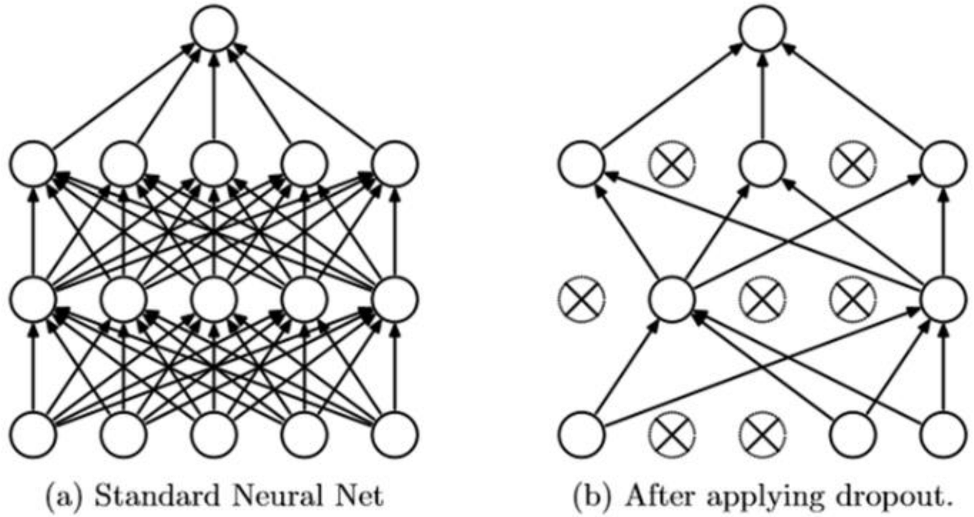


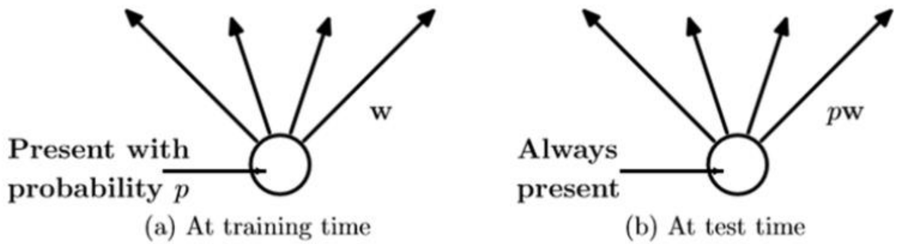
- ReLU



0. CNN 복습

A-4. Dropout





학습할 때마다 매번 랜덤하게 뽑은 node를 제외하고 학습

Train 이후 test(inference)할 때에는 모든 node 사용

0. CNN 복습

A-4. Dropout

Ex) dropout 적용 예시

$$softmax(W_3 \cdot \tanh(W_2 \cdot \mathrm{mask}(D, \tanh(W_1 \cdot input\_vector)))$$

Where $D = (d)_{ij}$ and $d_{ij} \sim B(1, p = 0.5)$

* Mask(D,M) : 행렬 D와 M의 element-wise 곱

CNN에서는?

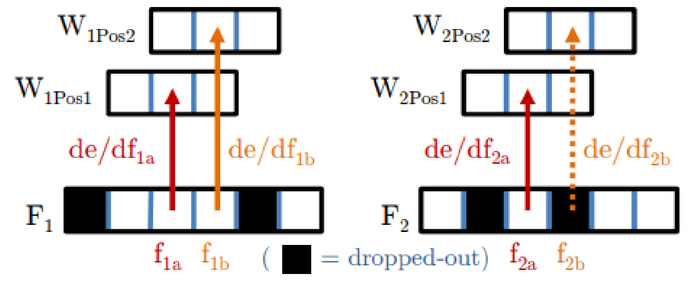


Figure 3: Standard Dropout after a 1D convolution layer

Standard Dropout : activation을 드롭아웃

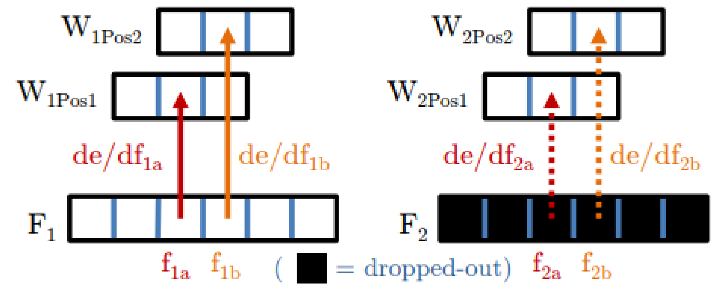


Figure 5: *SpatialDropout* after a 1D convolution layer

Spatial Dropout : 채널 자체를 드롭아웃

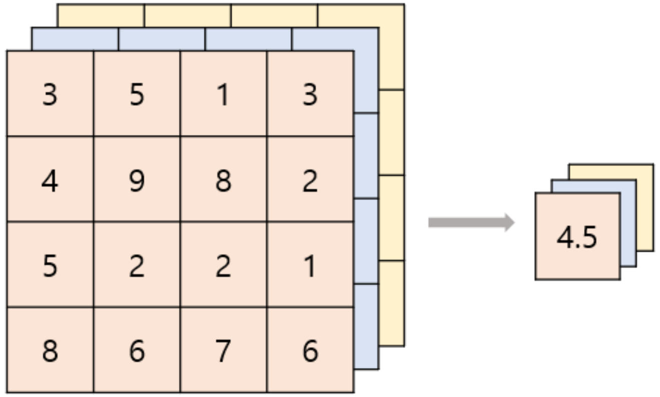
Ref) Efficient Object Localization Using Convolutional Networks
Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, Christoph Bregler New York University(2015)

0. CNN 복습

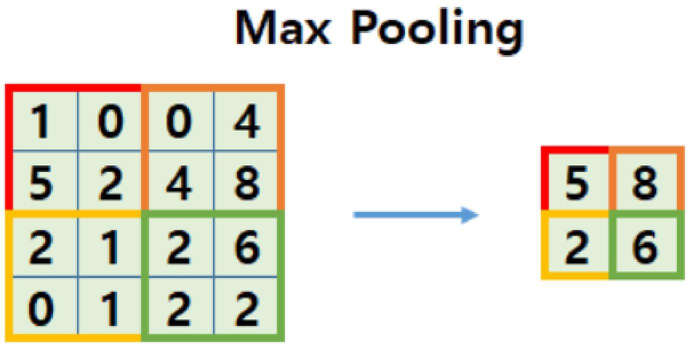
A-5. Pooling

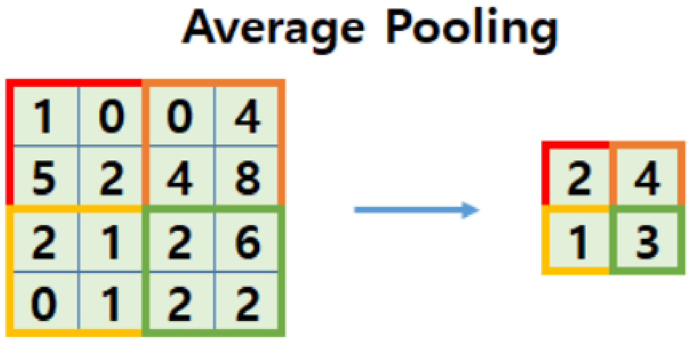
- 따로 학습할 필요 X
- 채널 별로 독립적으로 진행(input의 채널 수는 변화시키지 않는다.)
- 입력 데이터의 변화에 영향을 적게 받음
- 보통 filter size와 stride 크기를 동일하게 함(겹치지 않게)

-> 차원을 감소시켜 계산 효율을 높인다!



Global Average Pooling





Classifier

Classifier는 Feature Extractor 과정의 반복이 끝나면

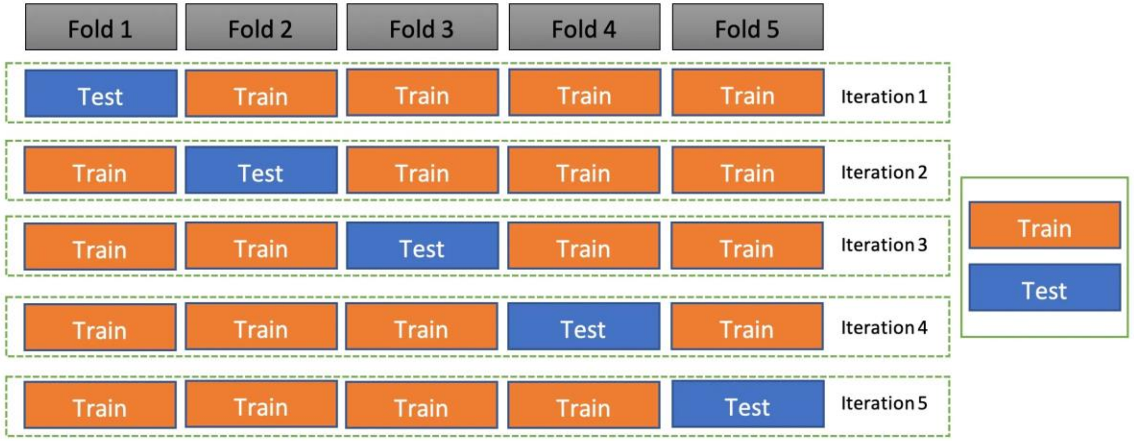
Global Average Pooling → FC Layer → Softmax → Classification

1. Overfitting in NN

데이터 자체와 관련

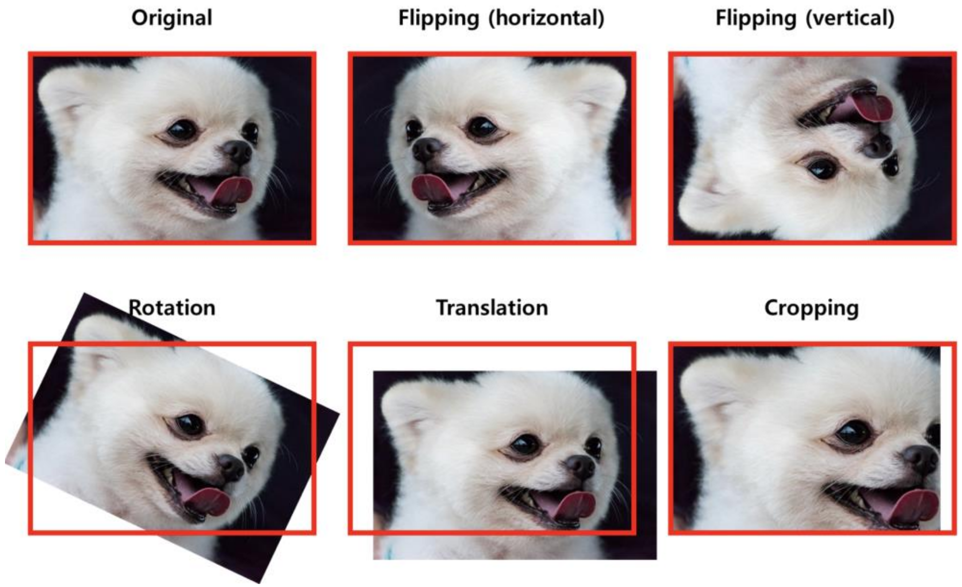
A. Cross Validation

K-fold Validation , Stratified Cross Validation



B. Data Augmentation

Data 수 자체를 늘린다!



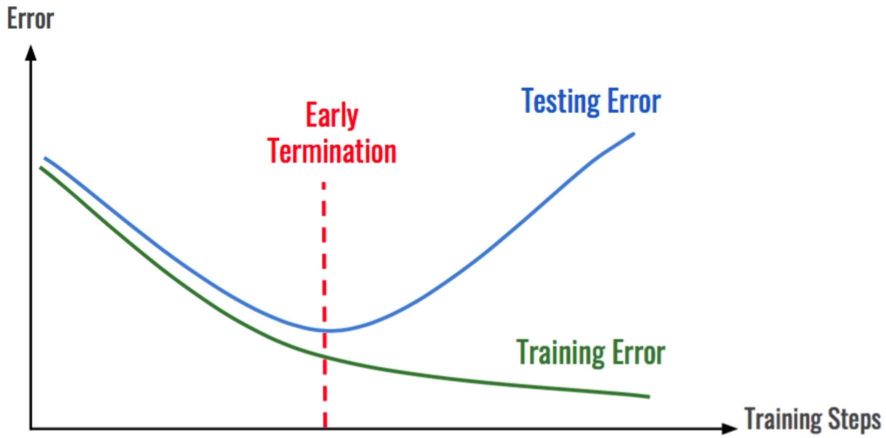
1. Overfitting in NN

학습과 관련

C. Early-stopping

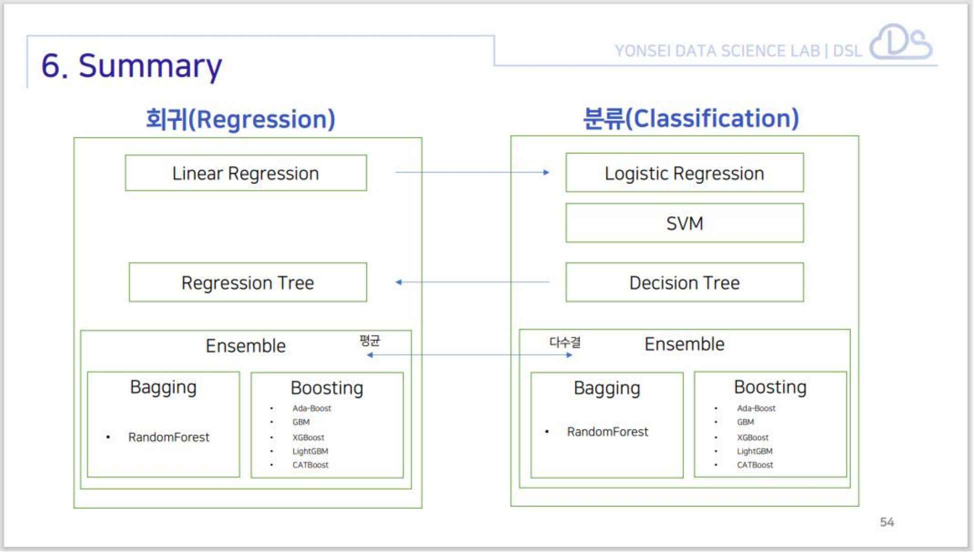
과적합되기 전에 멈춘다!

Loss가 일정 횟수(patience변수) 안에 개선이 되지 않으면 조기 종료를 해준다.



D. Ensemble

Bagging, Boosting



Ref) [0202] Decision Tree & Ensemble 세션 자료

1. Overfitting in NN

학습과 관련

E. Regularization

Ridge(L2) & Lasso(L1)

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

F. Dropout

아까 했으니 PASS…

A~F 외에도 하이퍼파라미터(lr, 노드 수, layer 수…)를

조절해주는 방법이 있다!

2. Speed of Convergence

Strategies to handle Speed of Convergence

(얼마나 빠르게 global optimization에 수렴하는가?)

A. Momentum term

B. Activation Function

C. Weight Initialization

D. Learning Rate

E. Batch Normalization (BN)

2. Speed of Convergence

Strategies to handle Speed of Convergence

(얼마나 빠르게 global optimization에 수렴하는가?)

A. Momentum term

B. Activation Function

C. Weight Initialization

D. Learning Rate

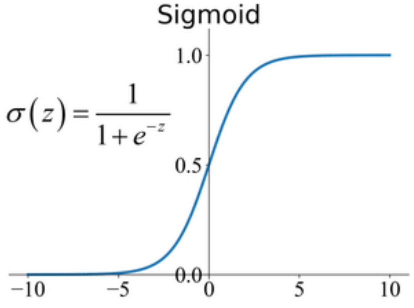
E. Batch Normalization (BN)

2. Speed of Convergence

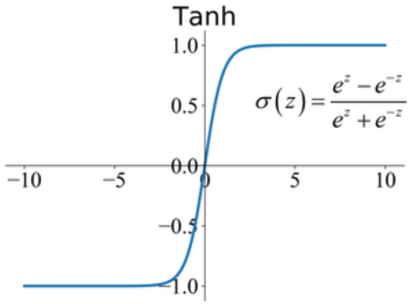
Activation Function & Activation Values

활성화 함수를 거쳐서 나온 값들

Sigmoid & Tanh Function

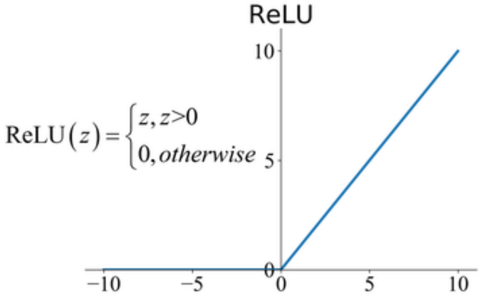


(a)

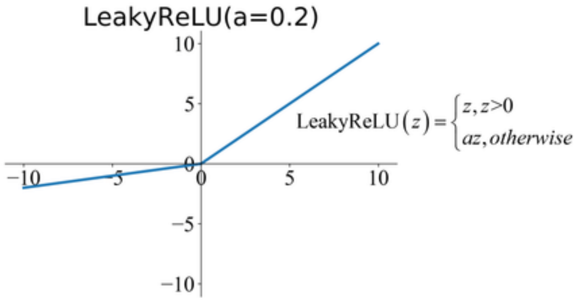


(b)

ReLU & LeakyReLU Function



(c)



(d)

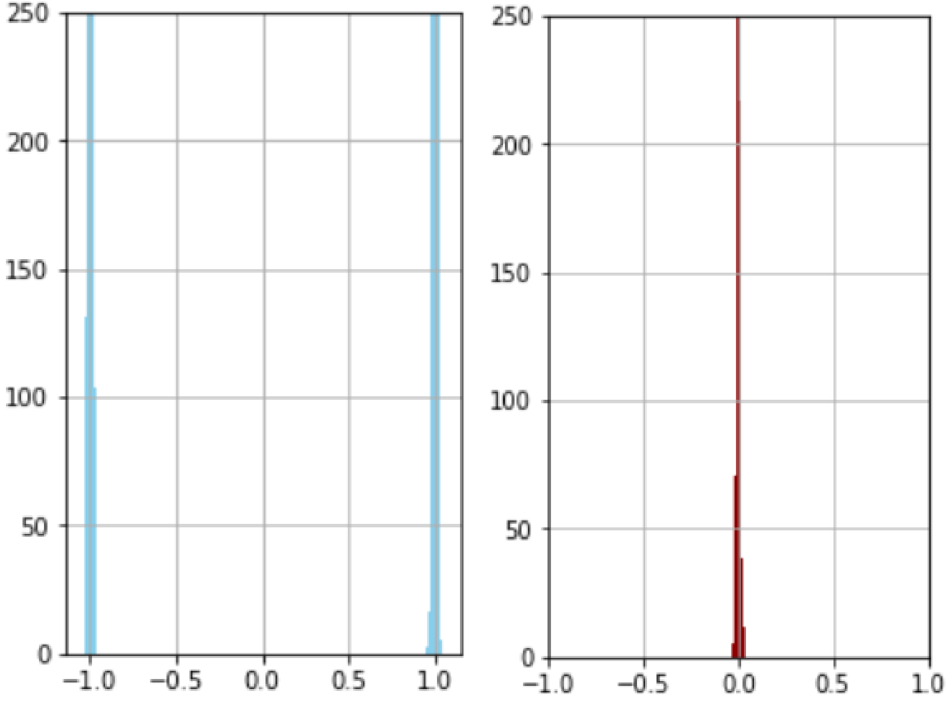
Gradient Vanishing의 문제…

2. Speed of Convergence

Activation Value 분포

Activation Value의 분포가 극단으로 치우쳐있거나 한쪽으로 몰려있다면?

또 Gradient Vanishing의 문제…

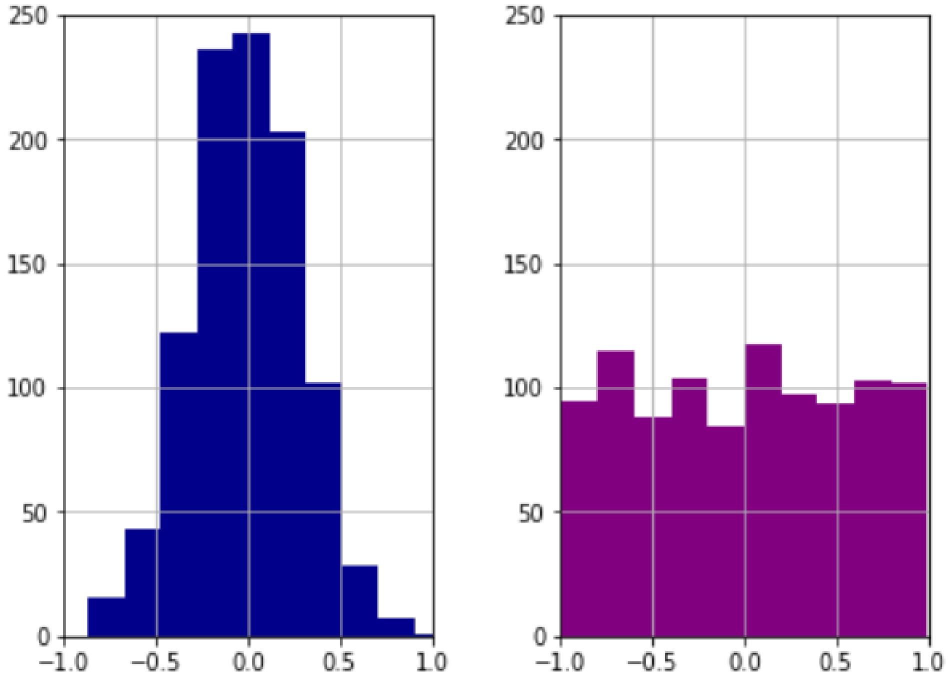


Y값들이 한 값으로 수렴하는 문제

2. Speed of Convergence

Activation Value 분포

다음과 같은 분포가 나오기를 원한다.



Activation Value에 영향을 주는 요소:

Input(X), weight, activation function

Input initialization은 CNN1에서 배웠으니

이번에는 weight initialization을 진행해보자!

2. Speed of Convergence

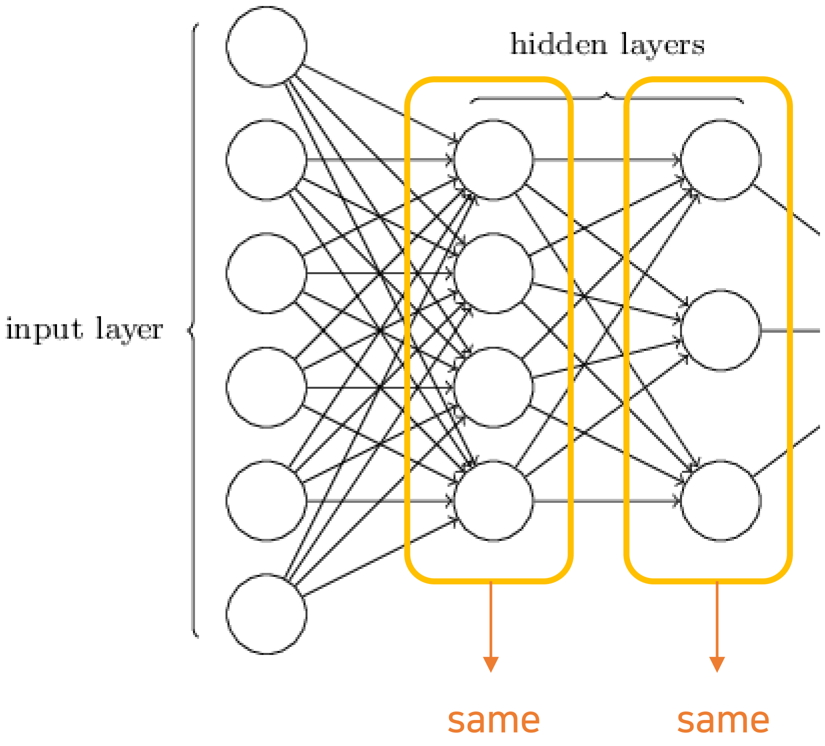
Weight Initialization (weight을 어떻게 설정해야 좋은 Activation Value 분포가 나올까?)

Poor Initialization Problem

A. Weight들을 다 0으로 만들어보자. (zero-init)

-> entire neurons do the same task

$$\text{neuron} = \sigma_1^{(1)}(\text{weight}_1^{(1)} * \text{input} + \text{bias}_1^{(1)})$$
$$\text{neuron} = \sigma_2^{(1)}(\text{weight}_2^{(1)} * \text{input} + \text{bias}_2^{(1)})$$
$$\text{neuron} = \sigma_3^{(1)}(\text{weight}_3^{(1)} * \text{input} + \text{bias}_3^{(1)})$$

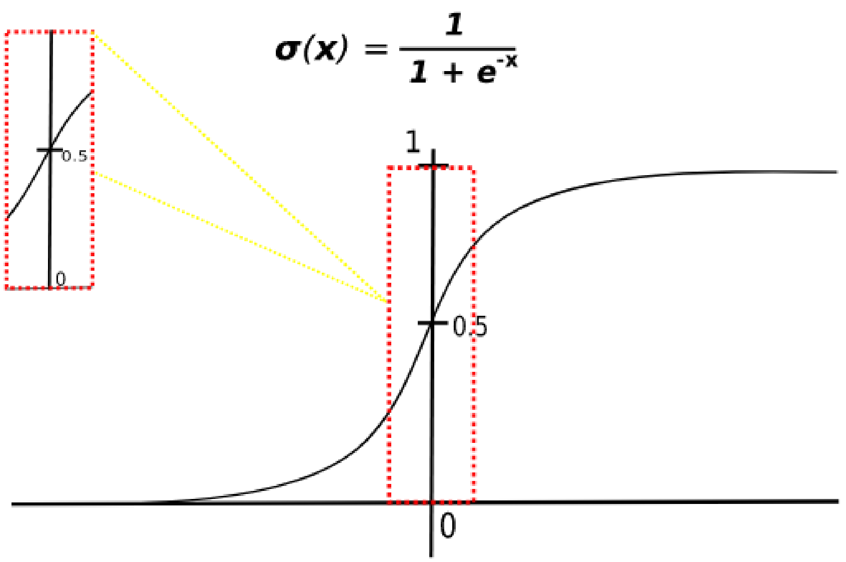


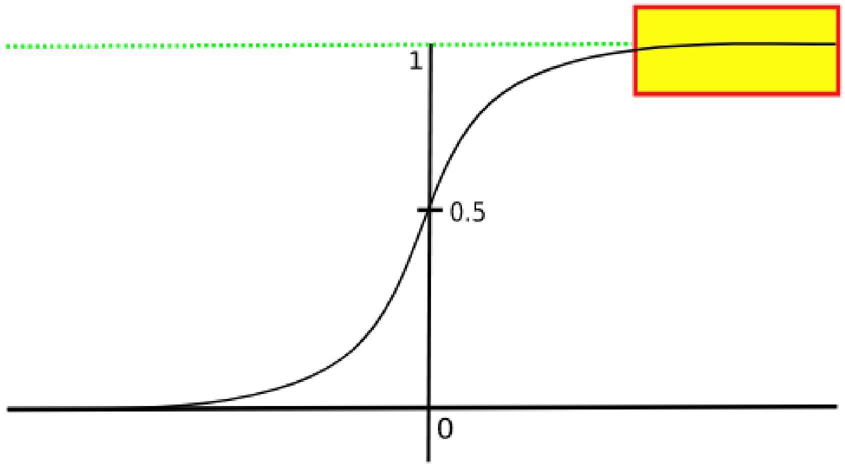
2. Speed of Convergence

Weight Initialization (weight을 어떻게 설정해야 좋은 Activation Value 분포가 나올까?)

Poor Initialization Problem

B. Too small or large weights





2. Speed of Convergence

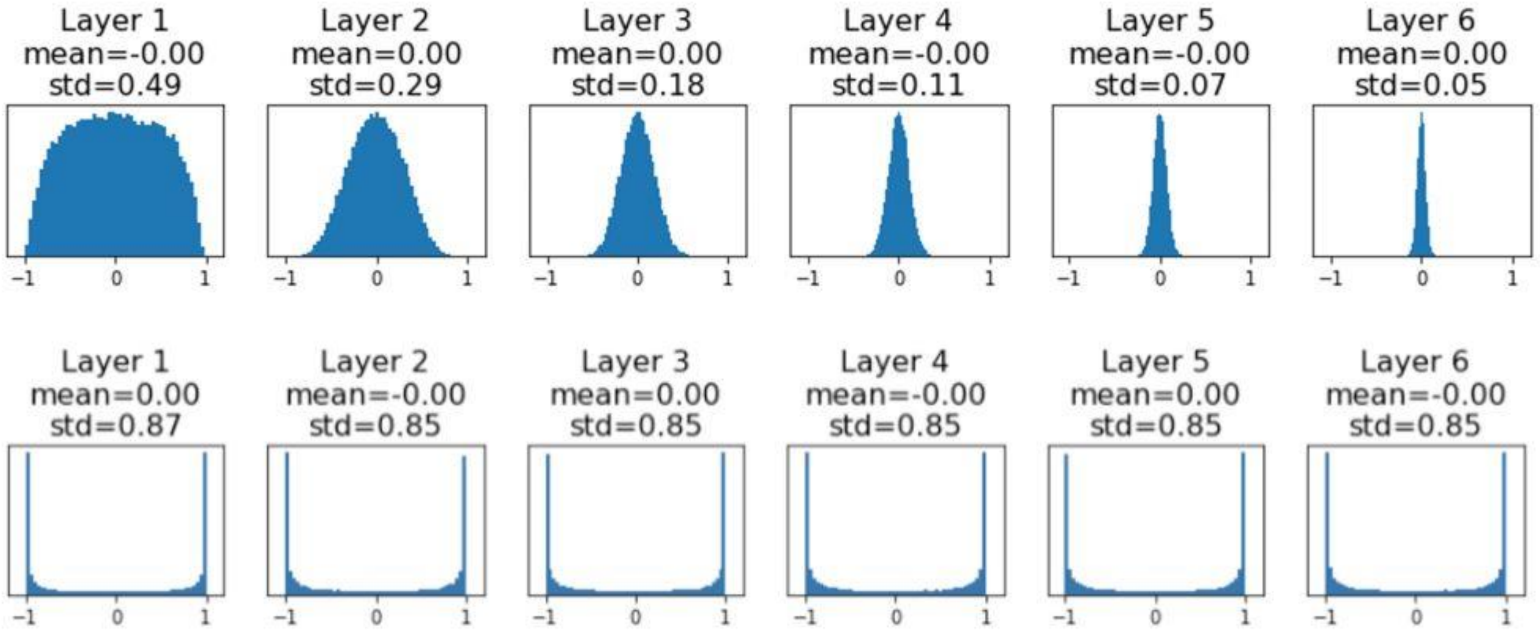
Weight Initialization (weight을 어떻게 설정해야 좋은 Activation Value 분포가 나올까?)

Poor Initialization Problem

C. Weights with too small or large variance (example with tanh)

W ~ G(0,0.01^2)

W ~ G(0,0.05^2)



2. Speed of Convergence

Weight Initialization

다음이 성립하도록 weight들을 초기화시켜야겠다.

1. Activation value들의 평균이 0이 되도록
2. Layer들을 통과할 때마다 Activation Value들의 분산이 크게 변하지 않도록

↓

결론

- Sigmoid, tanh는 Xavier Initialization 이용
- ReLU는 Kaiming He Initialization 이용
- 최근의 대부분의 모델에서는 He Initialization 주로 선택

2. Speed of Convergence

Weight Initialization

Solutions for Poor Initialization Problem

A. Xavier Initalization

B. Kaiming He Initalization

2. Speed of Convergence

Xavier Initialization

Xavier Normal Initialization

$$W \sim N(0, Var(W))$$

① $Var(W) = \dfrac{2}{n^{[l-1]} + n^{[l]}}$

② Var(W) = 1/n[l-1]

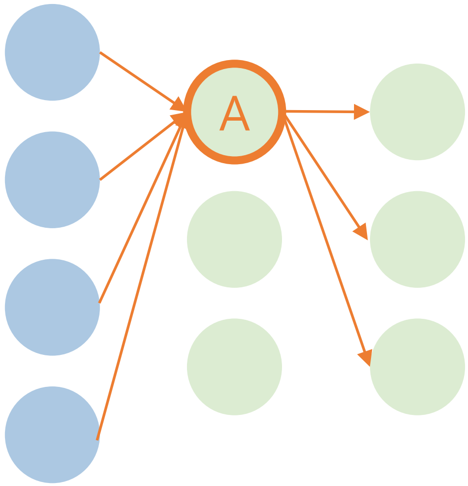
Xavier Uniform Initialization

$$W \sim U\left(-\sqrt{\frac{6}{n_{in} + n_{out}}},\ +\sqrt{\frac{6}{n_{in} + n_{out}}}\right)$$

<참고>

$$
\begin{aligned}
Var(a_i^{[\ell-1]}) &= Var(a_i^{[\ell]}) \\
&= Var(z_i^{[\ell]}) && \longleftarrow \text{linearity of } \textbf{tanh} \text{ around zero} \quad tanh(z) \approx z \\
&= Var\Big(\sum_{j=1}^{n^{[\ell-1]}} w_{ij}^{[\ell]} a_j^{[\ell-1]}\Big) \\
&= \sum_{j=1}^{n^{[\ell-1]}} Var(w_{ij}^{[\ell]} a_j^{[\ell-1]}) && \longleftarrow \text{variance of independent sum} \quad Var(X+Y) = Var(X) + Var(Y) \\
&= \sum_{j=1}^{n^{[\ell-1]}} E[w_{ij}^{[\ell]}]^2 Var(a_j^{[\ell-1]}) + \\
&\quad E[a_j^{[\ell-1]}]^2 Var(w_{ij}^{[\ell]}) + && \longleftarrow \text{variance of independent product} \quad Var(XY) = E[X]^2Var(Y) + E[Y]^2Var(X) + Var(X)Var(Y) \\
&\quad Var(w_{ij}^{[\ell]}) Var(a_j^{[\ell-1]}) \\
&= n^{[\ell-1]} Var(w_{ij}^{[\ell]}) Var(a_j^{[\ell-1]}) \implies Var(W) = \frac{1}{n^{[\ell-1]}}
\end{aligned}
$$

Forward로 계산했을 때임.
Backprop도 같이 고려한다면
Var(W) = 2 / (in+out)



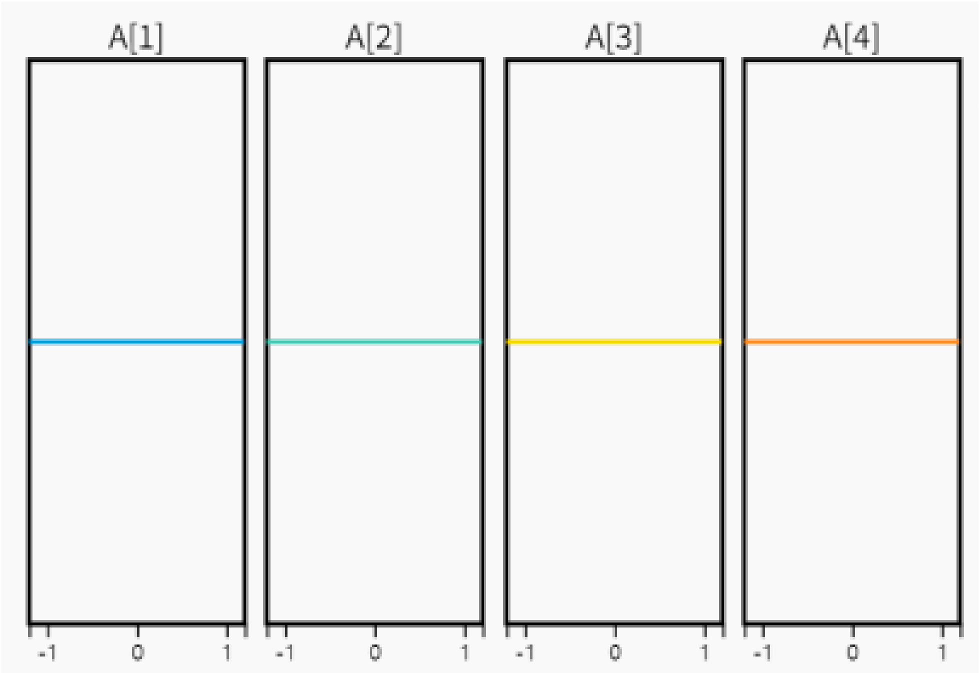
Weight connected with A node

① W(i) ~ G(0, 2/7)

② W(i) ~ G(0,1/4)

2. Speed of Convergence

Xavier Initialization



Activation Values

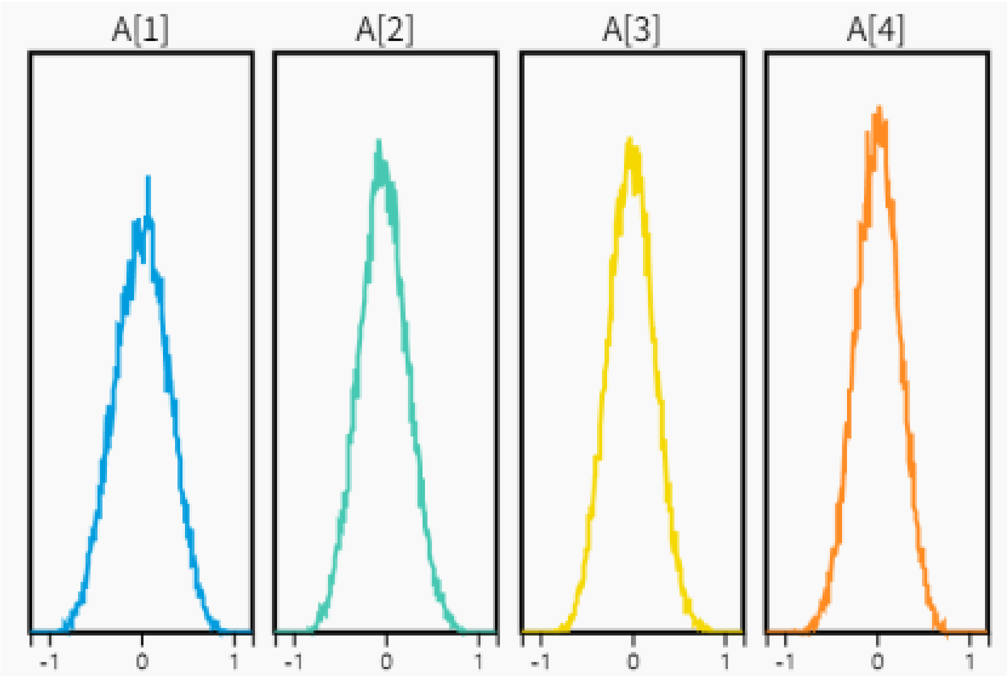
Just after

100 batch

(100 iteration)

Using Tanh

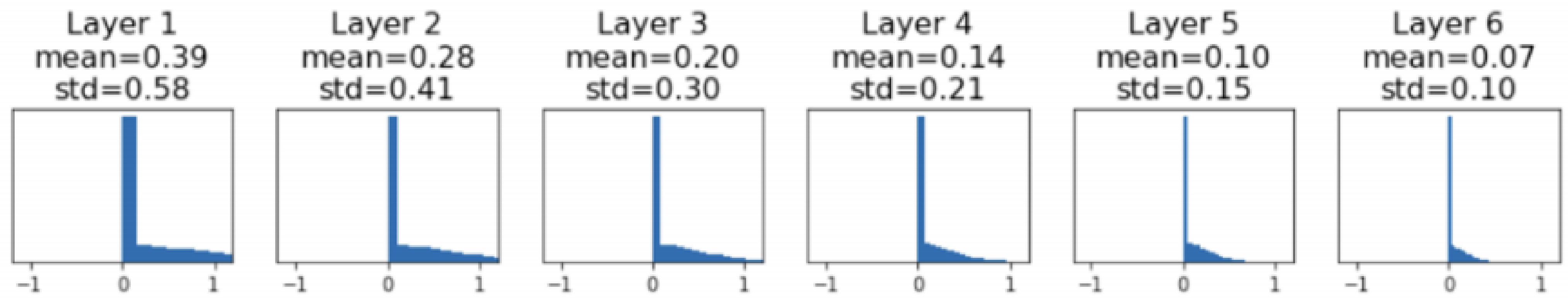
Activation Func.



2. Speed of Convergence

Xavier Initialization

근데 ReLU Function에는 잘 안 된다.



Change tanh -> ReLU

Activation Values converge to 0

그래서 나온 게 Kaiming He Initialization

2. Speed of Convergence

Kaiming He Initialization

He Normal Initialization

$$W \sim N(0, Var(W))$$

$$Var(W) = \frac{2}{n_{in}}$$

너무 모이는 것을 방지해준다....

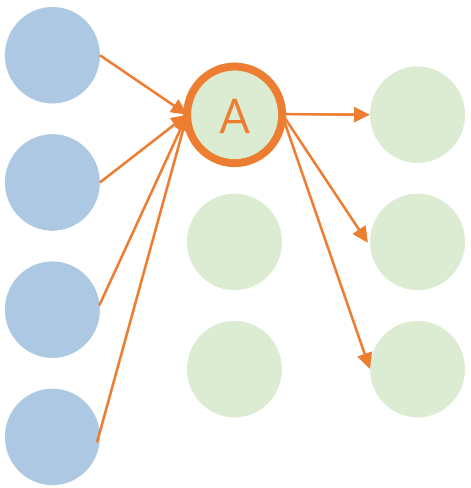
He Uniform Initialization

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}},\ +\sqrt{\frac{6}{n_{in}+n_{out}}}\right)$$

<참고>

$$\begin{aligned}
E[x^2] &= \int_{-\infty}^{\infty} x^2 P(x)dx \\
&= \int_{-\infty}^{\infty} max(0,y)^2 P(y)dy \\
&= \int_{0}^{\infty} y^2 P(y)dy \\
&= 0.5 * \int_{-\infty}^{\infty} y^2 P(y)dy \\
&= 0.5 * Var(y)
\end{aligned}$$

Xavier 식 증명에
위와 같은 성질을 대입한다.

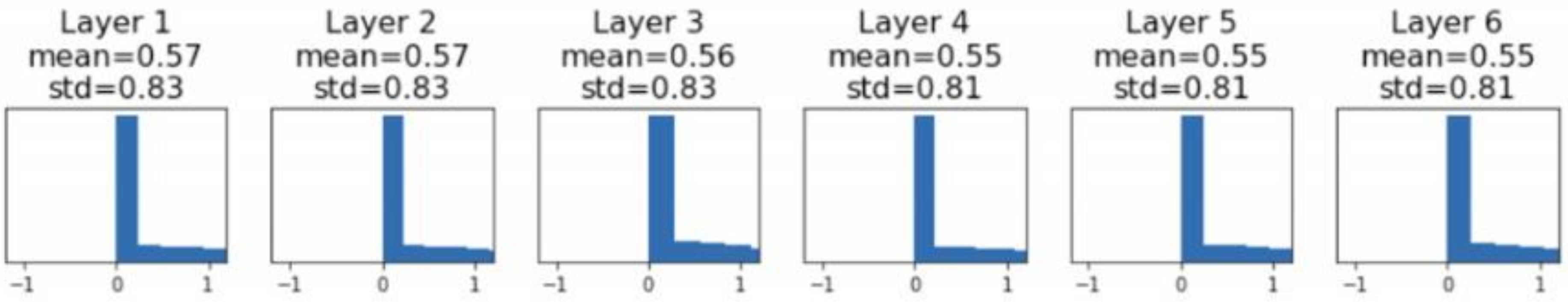


Weight connected with A node

① W(i) ~ G(0, 2/4)

2. Speed of Convergence

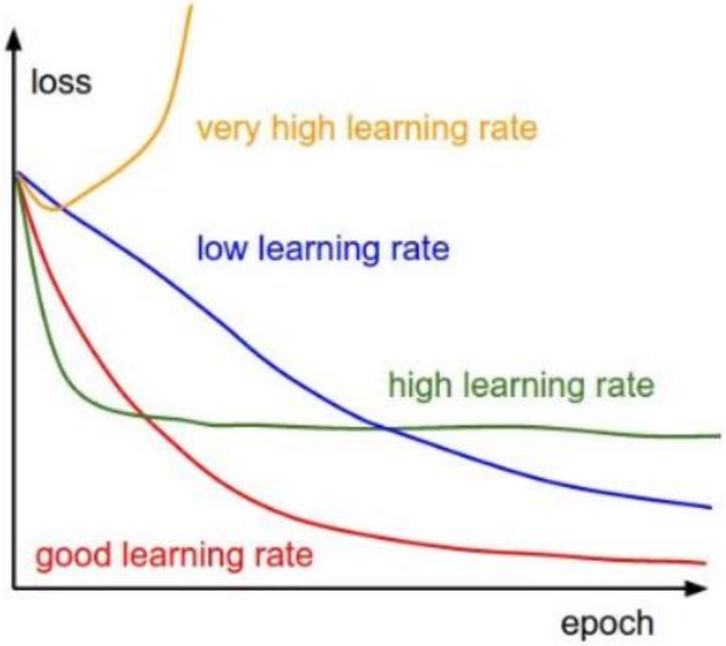
Kaiming He Initialization

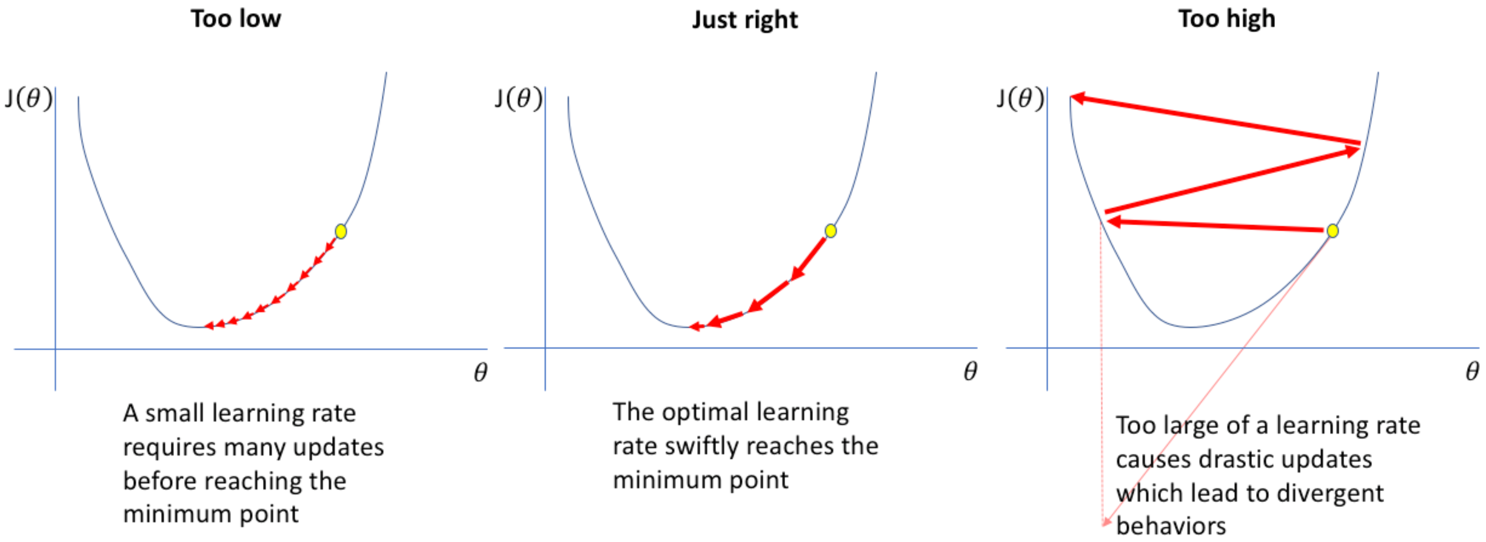


Works well with ReLU Function!

2. Speed of Convergence

Learning Rate





최적값에 가까워질수록 learning rate 줄이는 방법

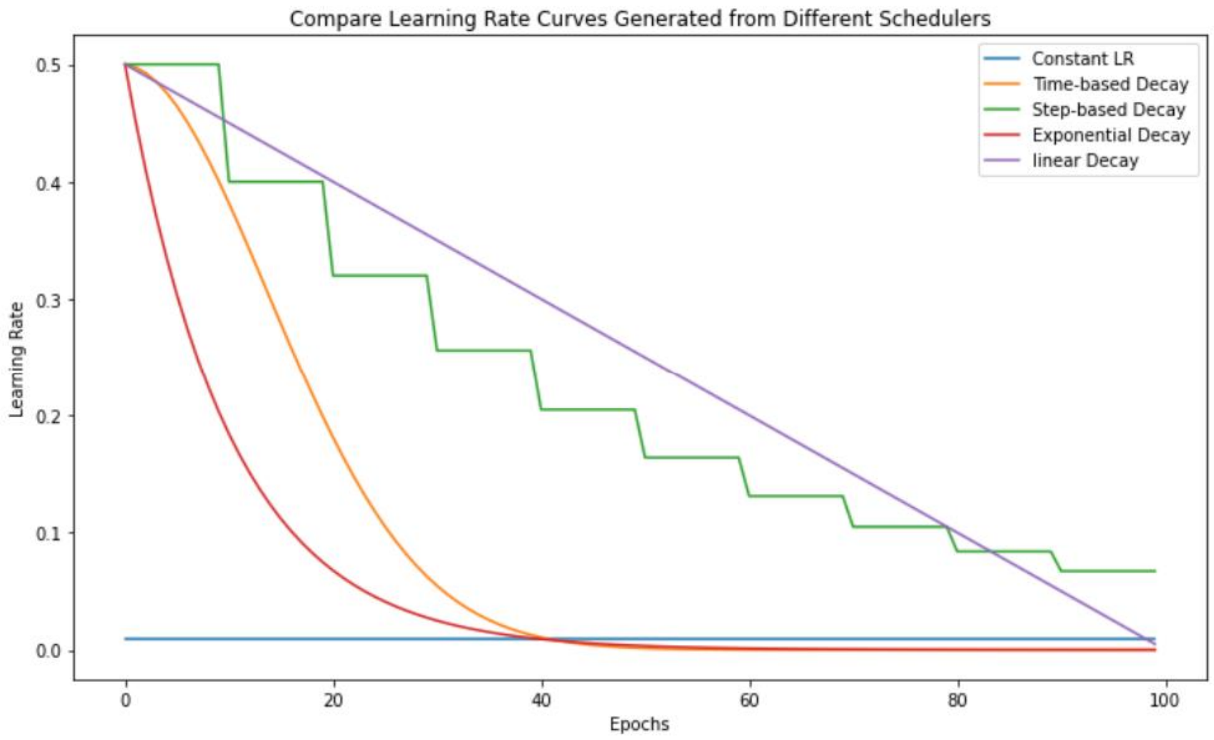
: learning rate scheduling

2. Speed of Convergence

Learning Rate Scheduling

학습이 진행됨에 따라 epoch 또는 iteration 간에 학습률을 조정하는 사전 정의된 프레임워크

A. Step decay (초록색 선)



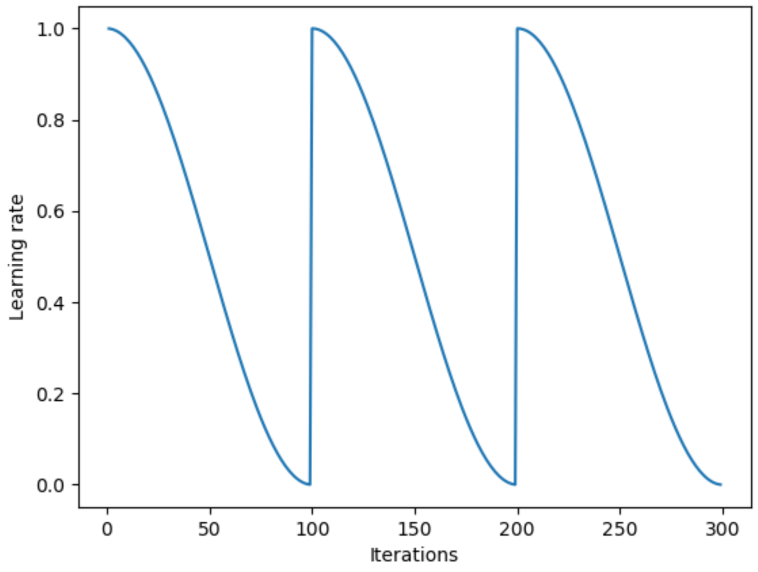
Reduce a% of learning rate after b epoch

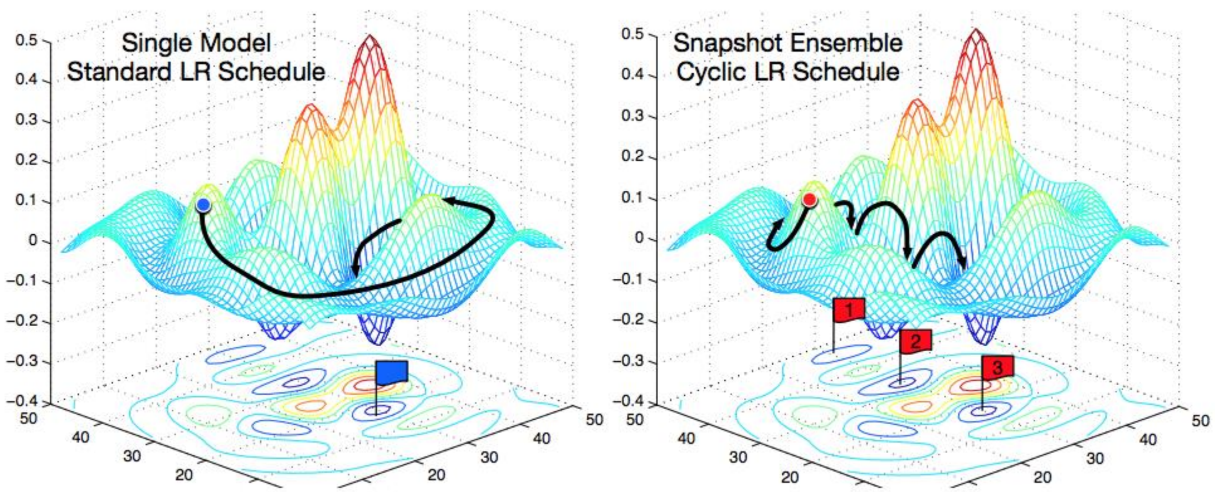
hyperparameter

2. Speed of Convergence

Learning Rate Scheduling

B. Cosine Annealing decay : local minimum, 안장점에서 빠르게 벗어나게 해준다.





$$\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)\right)$$

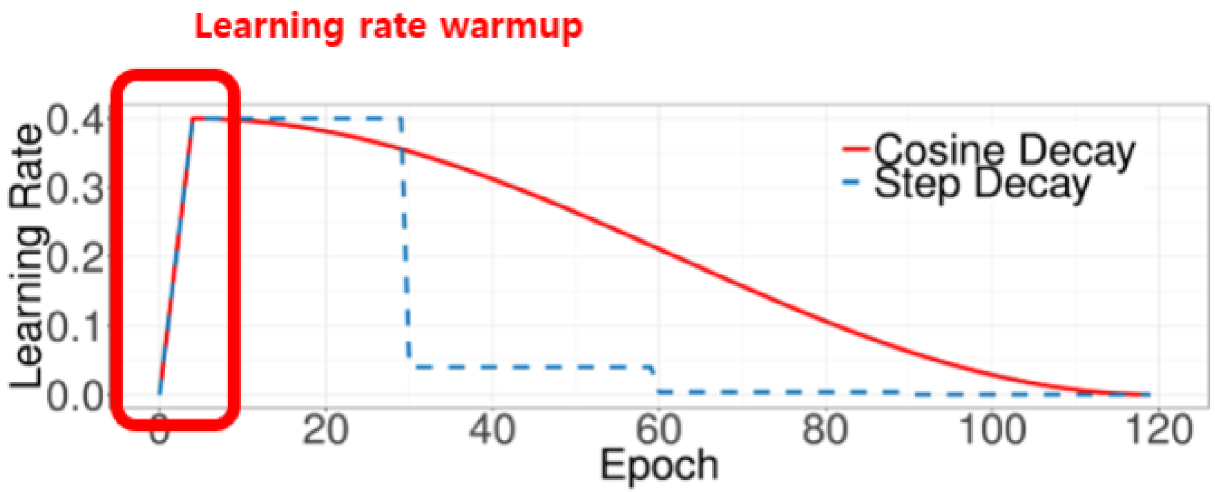
수렴 -> 탈출 -> 수렴 -> 탈출… 반복

2. Speed of Convergence

Learning Rate Scheduling

Warmup

- 모든 scheduler에 적용 가능
- Prevent divergence



(a) Learning Rate Schedule

2. Speed of Convergence

Batch Normalization

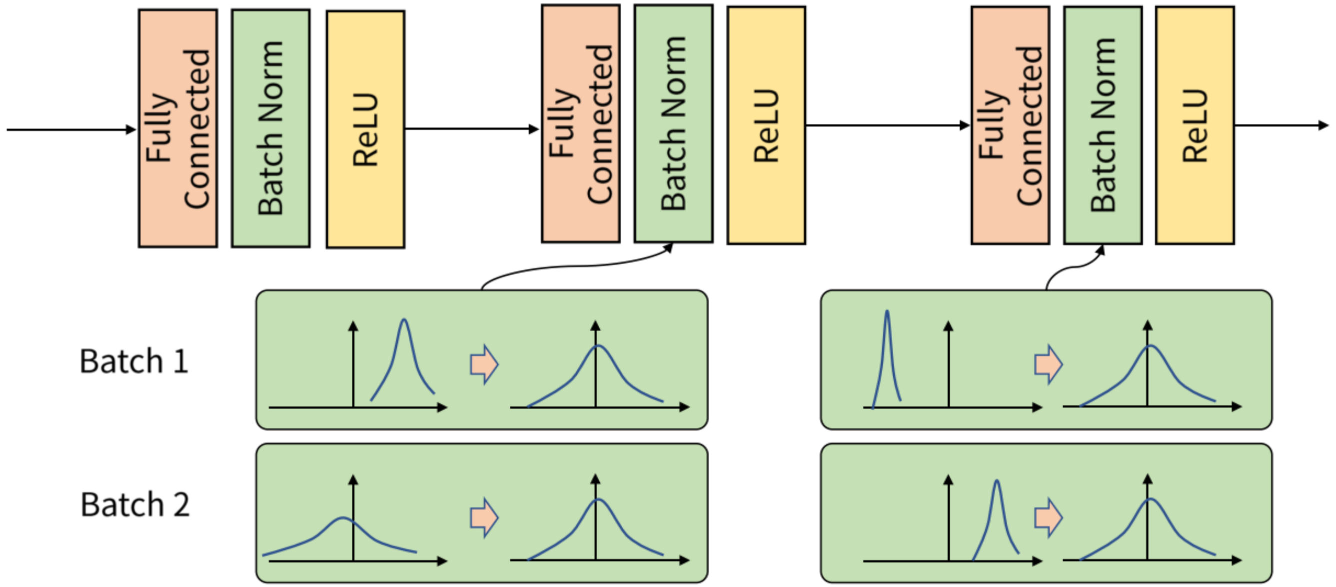
: Internal Covariate Shift 문제 해결 위해 고안

장점

- Weight initialization의 영향을 줄임
- High learning rate 가능하게
- Sigmoid, tanh 사용 가능하게
- Increase accuracy

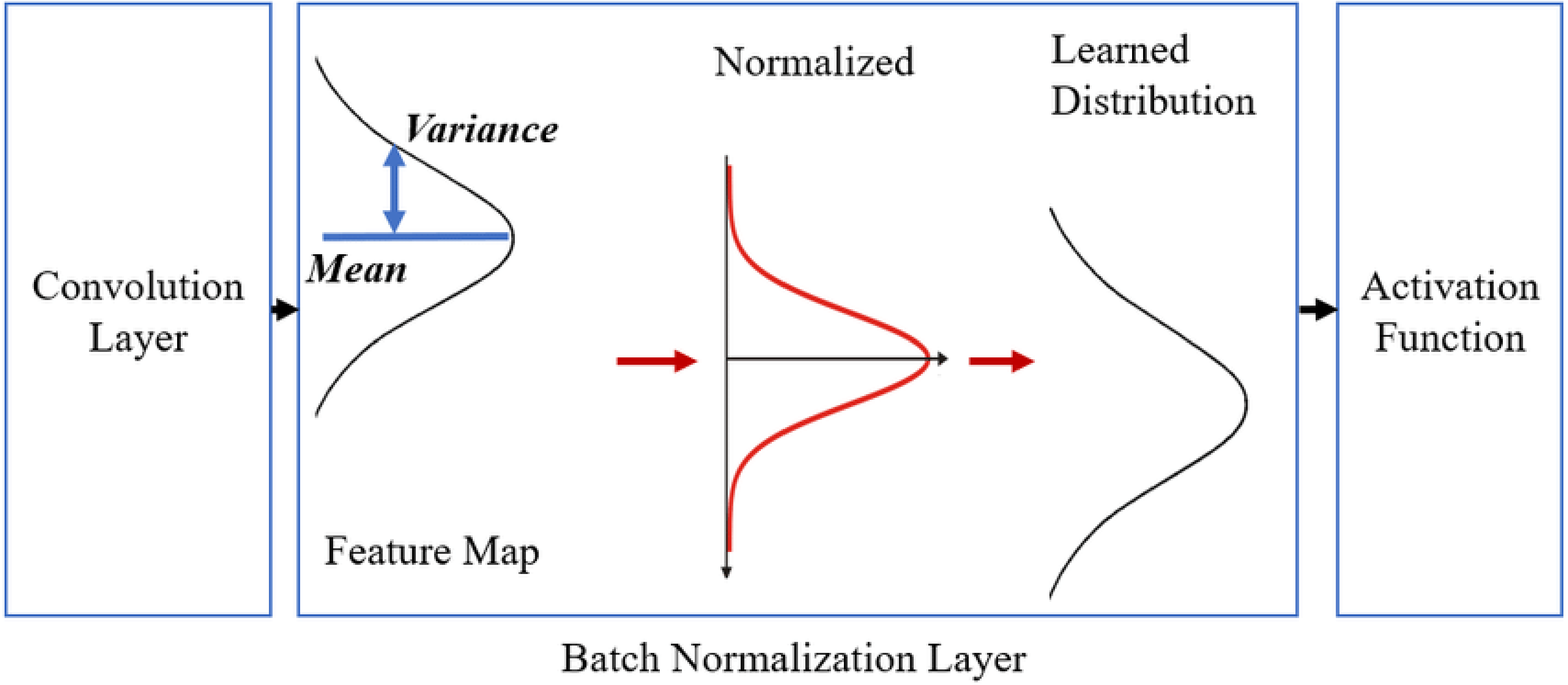
단점

- Batch number should be large
- Dynamic network structure & RNN에 적합 X



2. Speed of Convergence

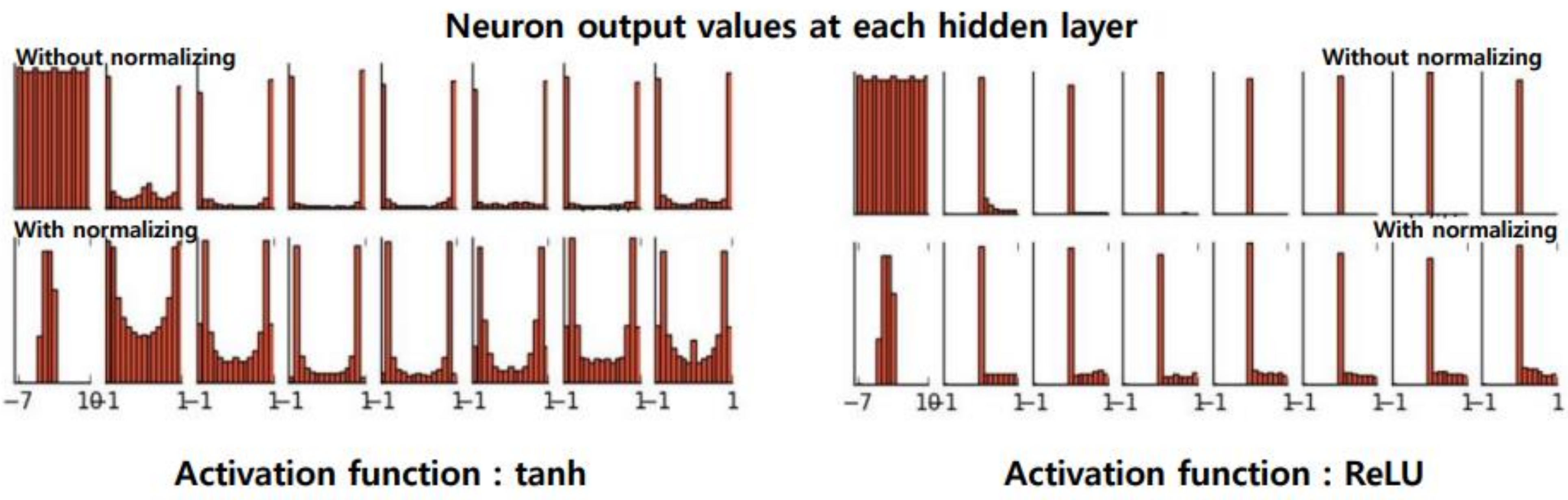
Batch Normalization



2. Speed of Convergence

Batch Normalization

Without using any initialization, BN makes very nice activation value distribution



2. Speed of Convergence

Strategies to handle Speed of Convergence

(얼마나 빠르게 global optimization에 수렴하는가?)

A. Momentum term

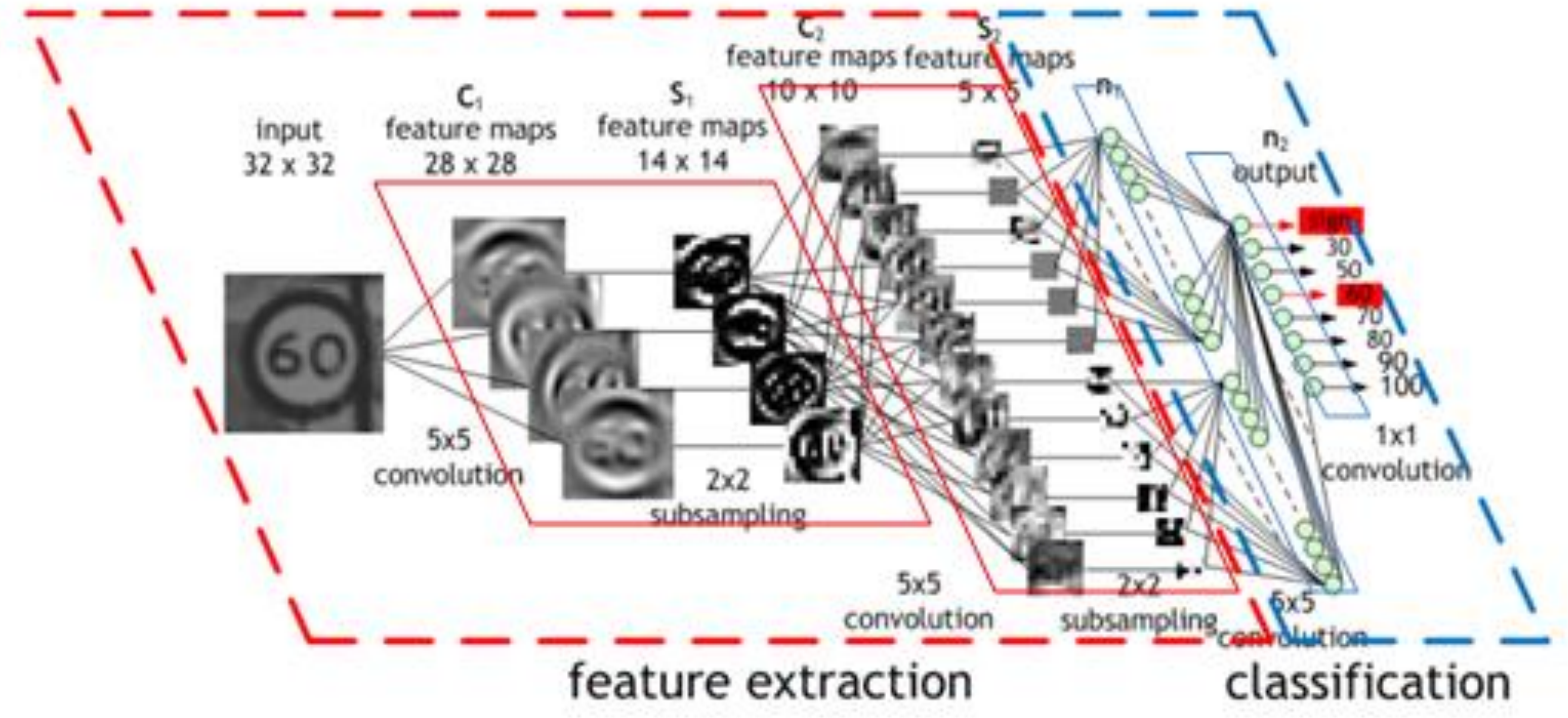
B. Activation Function

C. Weight Initialization

D. Learning Rate

E. Batch Normalization (BN)

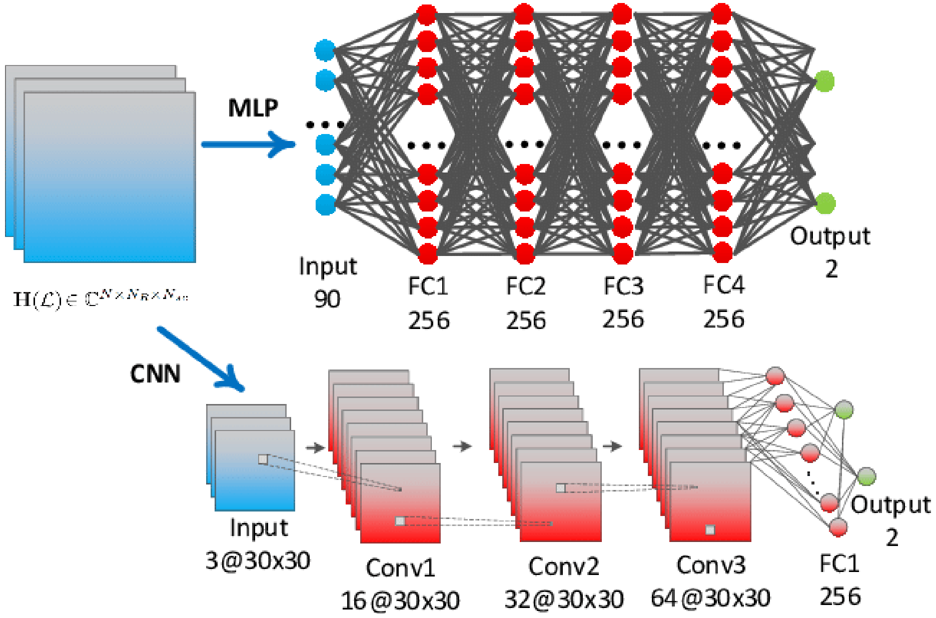
3. Why CNN?


input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
30
50
70
80
90
100
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
5x5
convolution
1x1
convolution
feature extraction
classification

3. Why CNN?

CNN vs MLP

CNN is a special case of MLP



MLP = fully connected

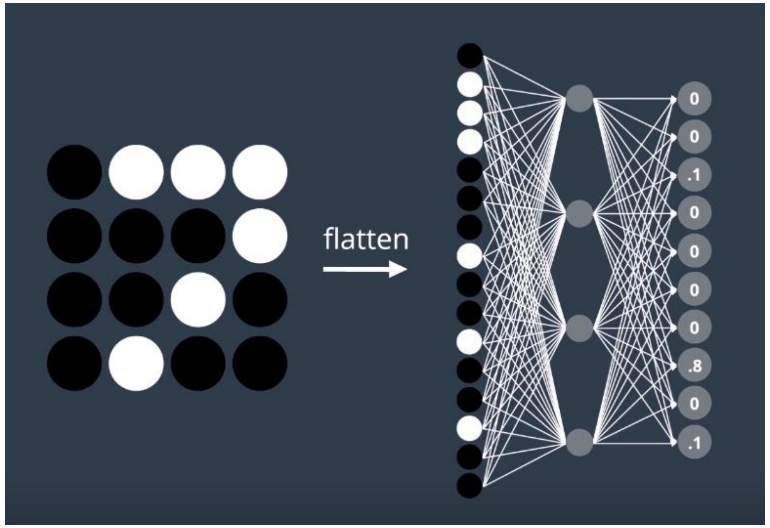
모든 input feature들을 다 고려한다.

다음 뉴런도 모든 input들을 다 반영하고 있는 상태

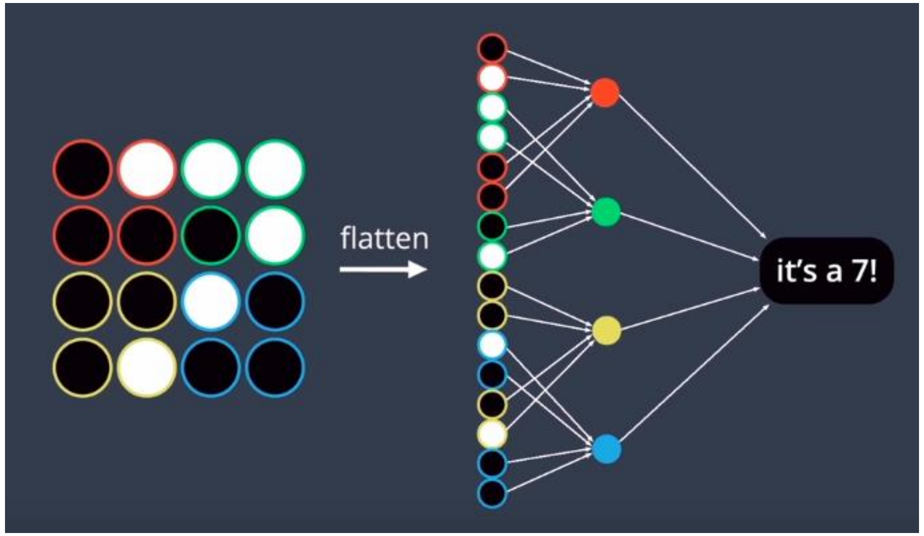
3. Why CNN?

CNN의 특징

A. Local Connectivity



MLP



CNN

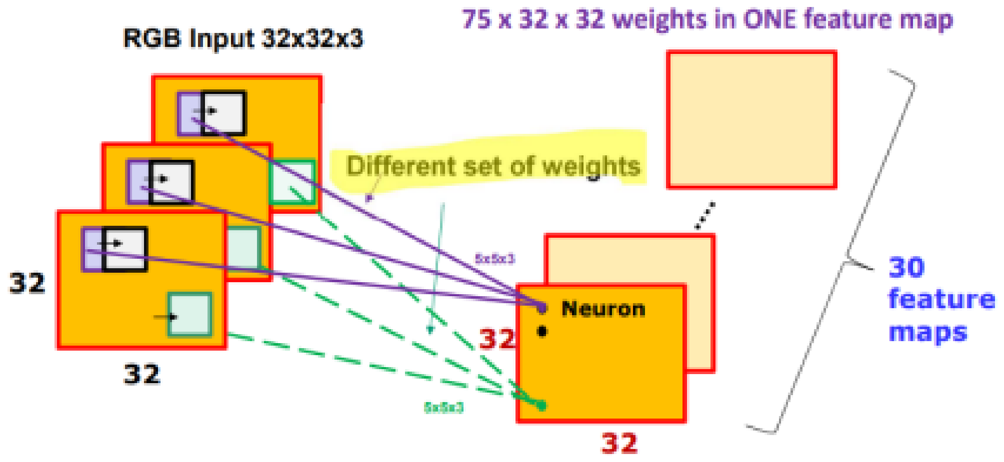
+ bias는 모델이 local information을 잘 학습하도록 도와준다.

3. Why CNN?

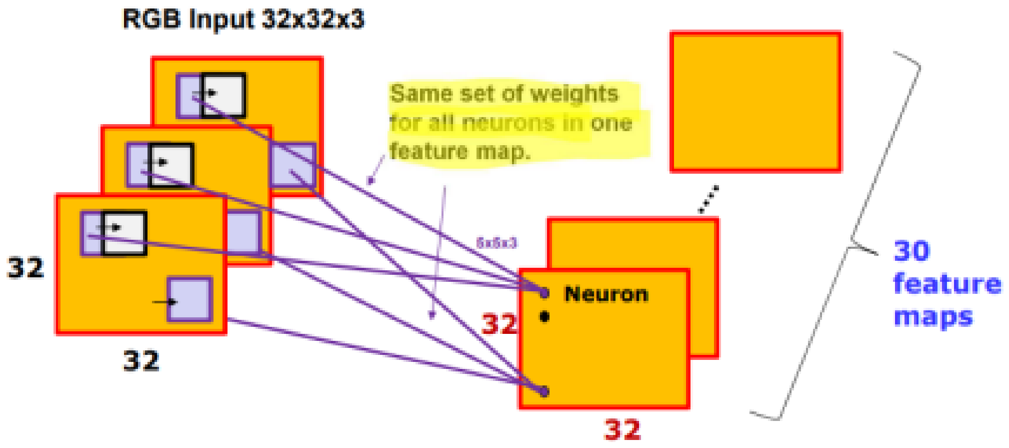
CNN의 특징

B. Weight Sharing

Filter 한 개로 feature map 하나 생성
= 하나의 feature map에 대한 가중치는 동일



MLP



CNN

Total Weight 개수:

(5*5*3) * (32*32*30)

Total Weight 개수:

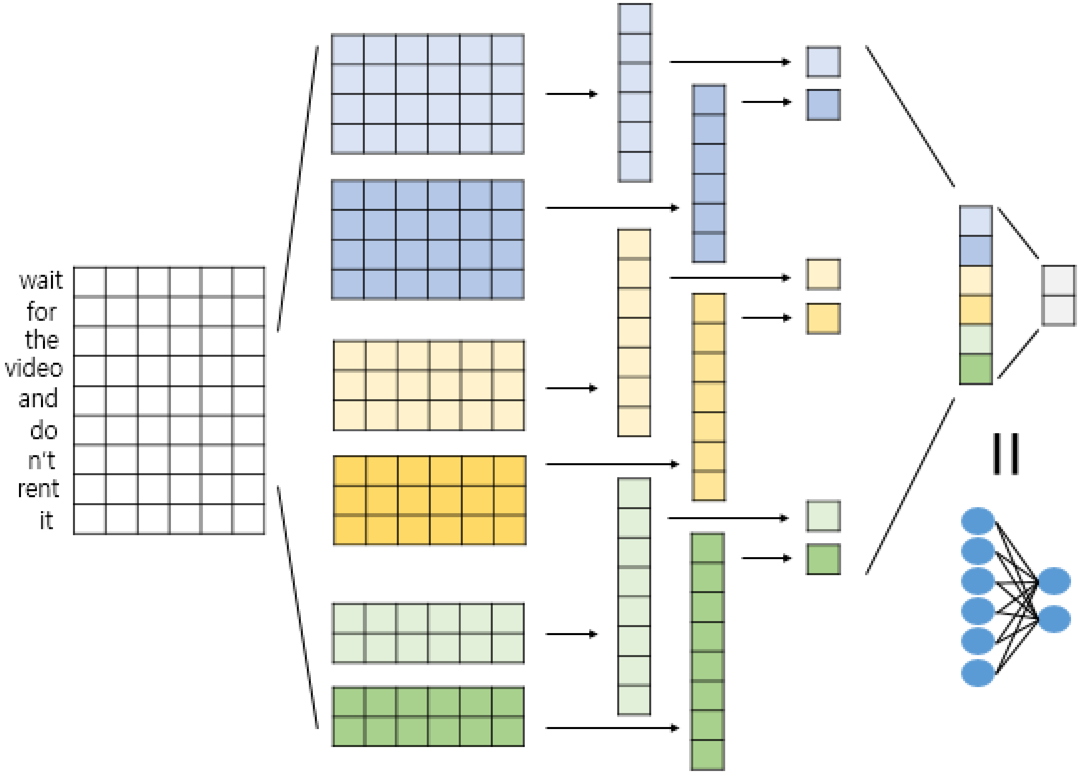
(5*5*3) * 30

3. Why CNN?

1D CNN

이미지 처리 외에도 CNN을 사용할 수 있다! (1)

→ 시계열, NLP



필터가 움직이는 방향이 한 방향 -> 1D CNN

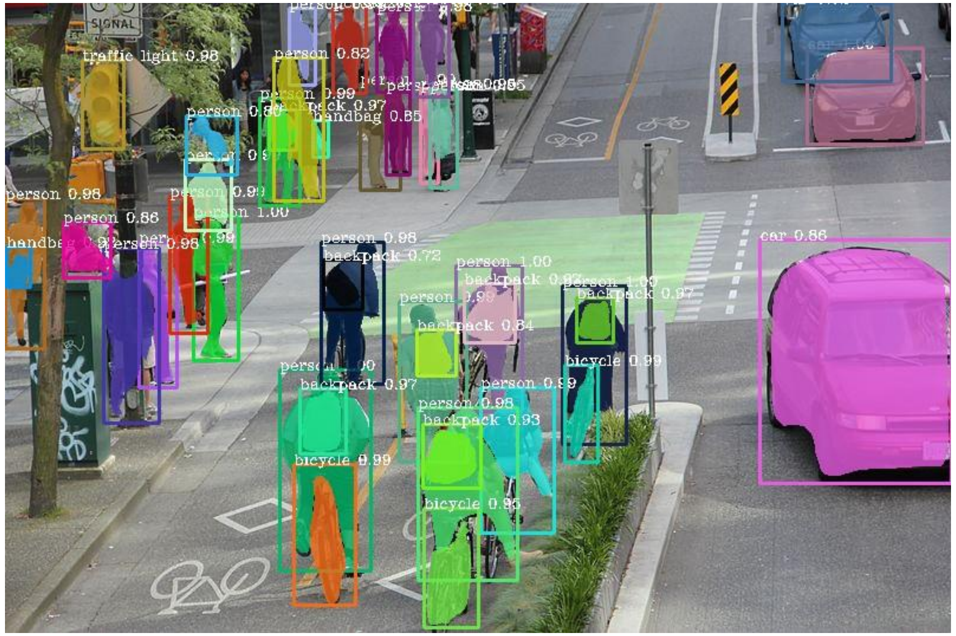
Window size = kernal size

3. Why CNN?

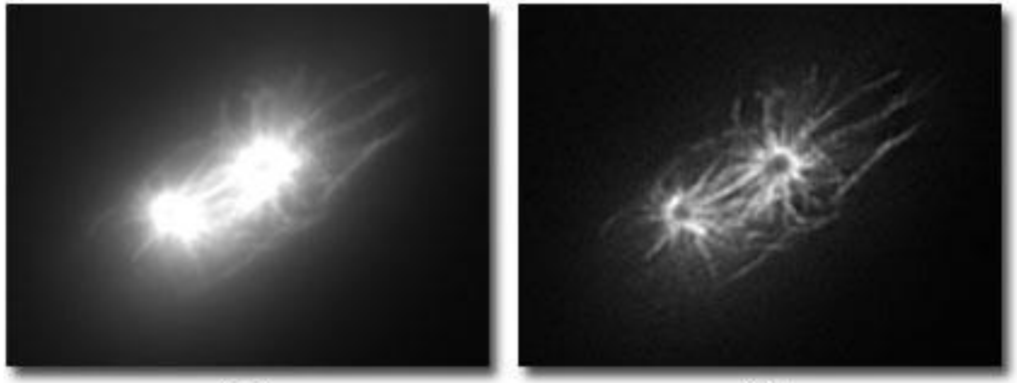
Deconvolution

이미지 처리 외에도 CNN을 사용할 수 있다! (2)

→ Image Segmentation, Super resolution



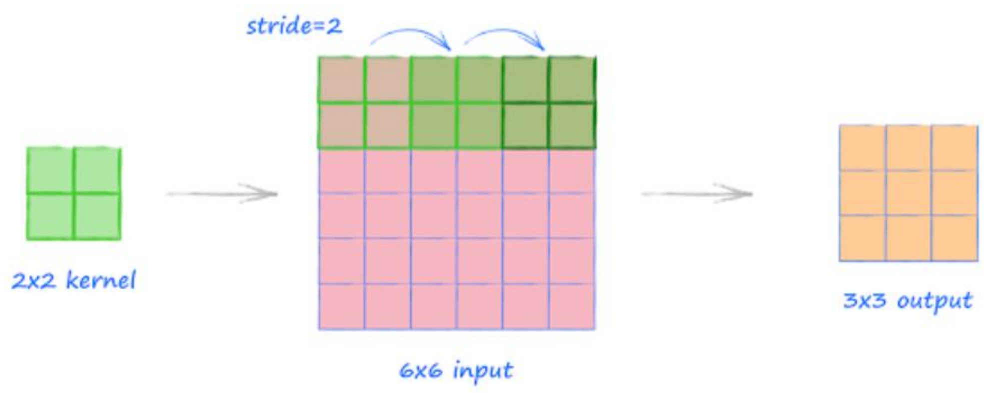
Before and After Nearest Neighbor Deconvolution Analysis



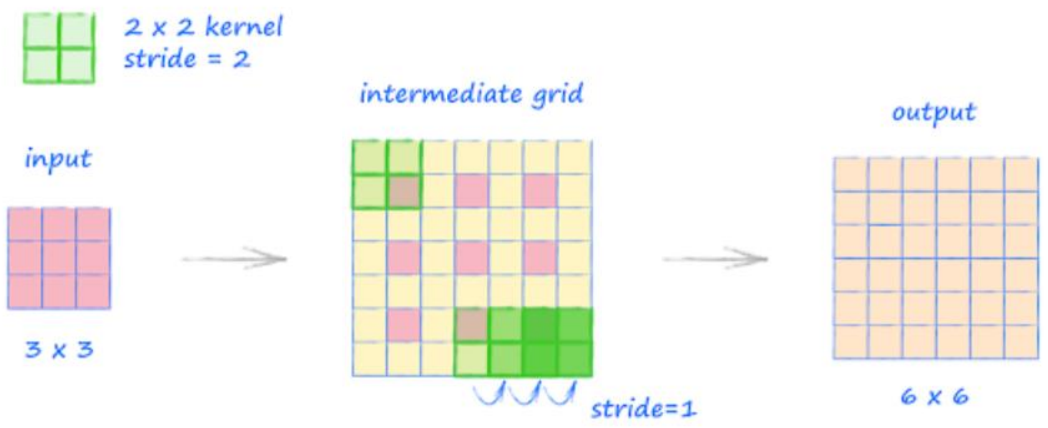
(a) Figure 1 (b)

Deconvolution

Ex) stride = 2, no padding



Convolution



Deconvolution

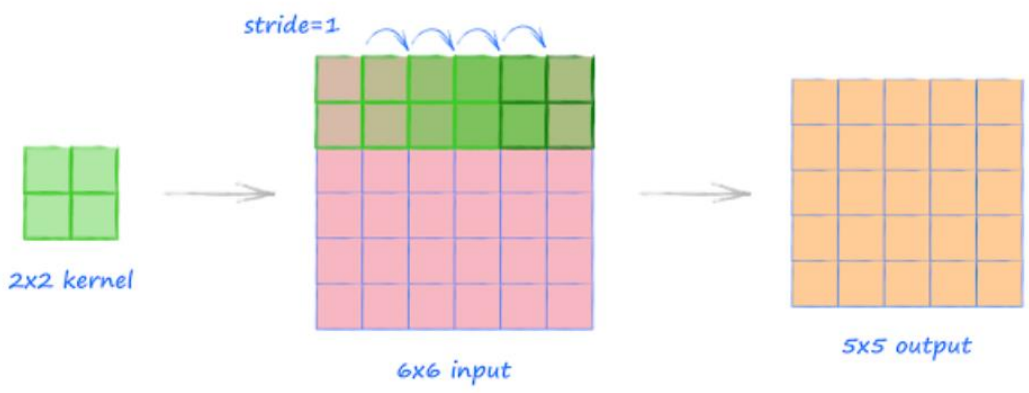
Deconvolution에서 stride는 kernel stride를 의미하는 것이 아님.

Stride = 각각의 input pixel이 얼마나 떨어져있는지 (intermediate grid)

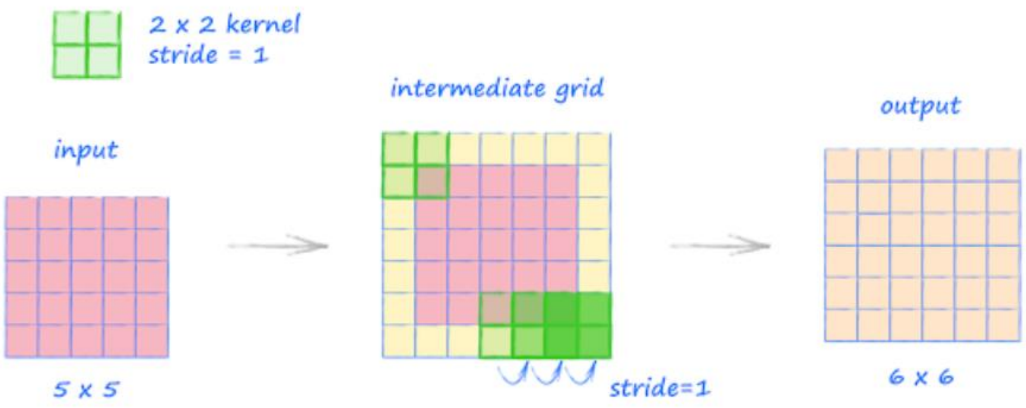
3. Why CNN?

Deconvolution

Ex) stride = 1, no padding



Convolution



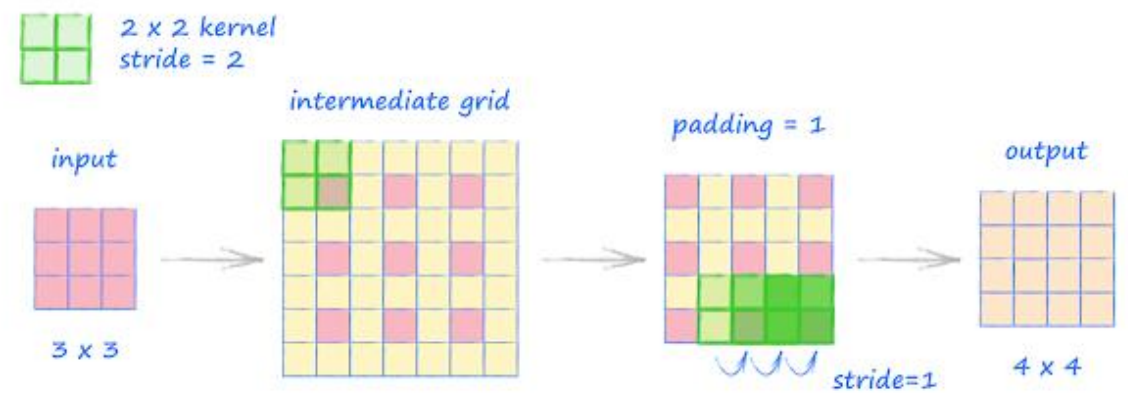
Deconvolution

Stride = 1

3. Why CNN?

Deconvolution

Ex) stride = 2, with padding



Deconvolution

$$\boxed{dim_{out}} = \frac{N - F + 2P}{S} + 1$$

역연산

N = (input-1)*S + F – 2P

Padding = 1

: 테두리를 없애준다

Ref) http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

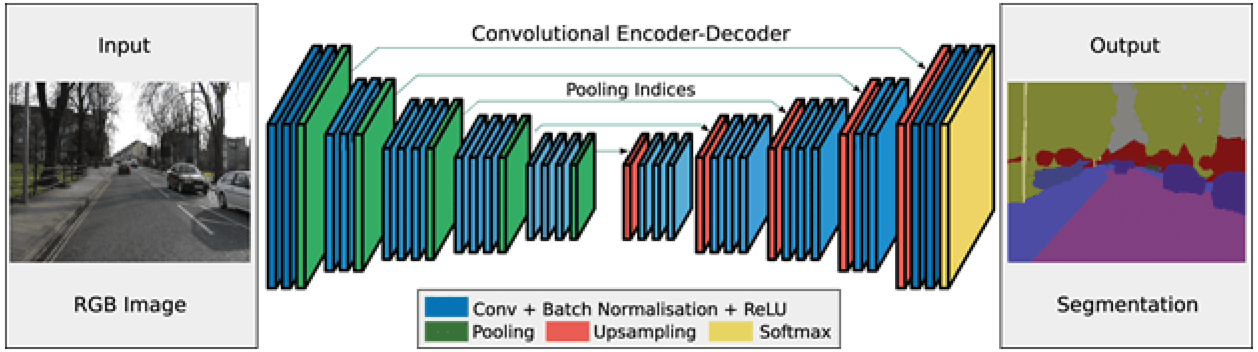
3. Why CNN?

Deconvolution

Application – Image Segmentation

어떻게 보면 새로운 이미지를 생성한 것

: Deconvolution 이용해야 함

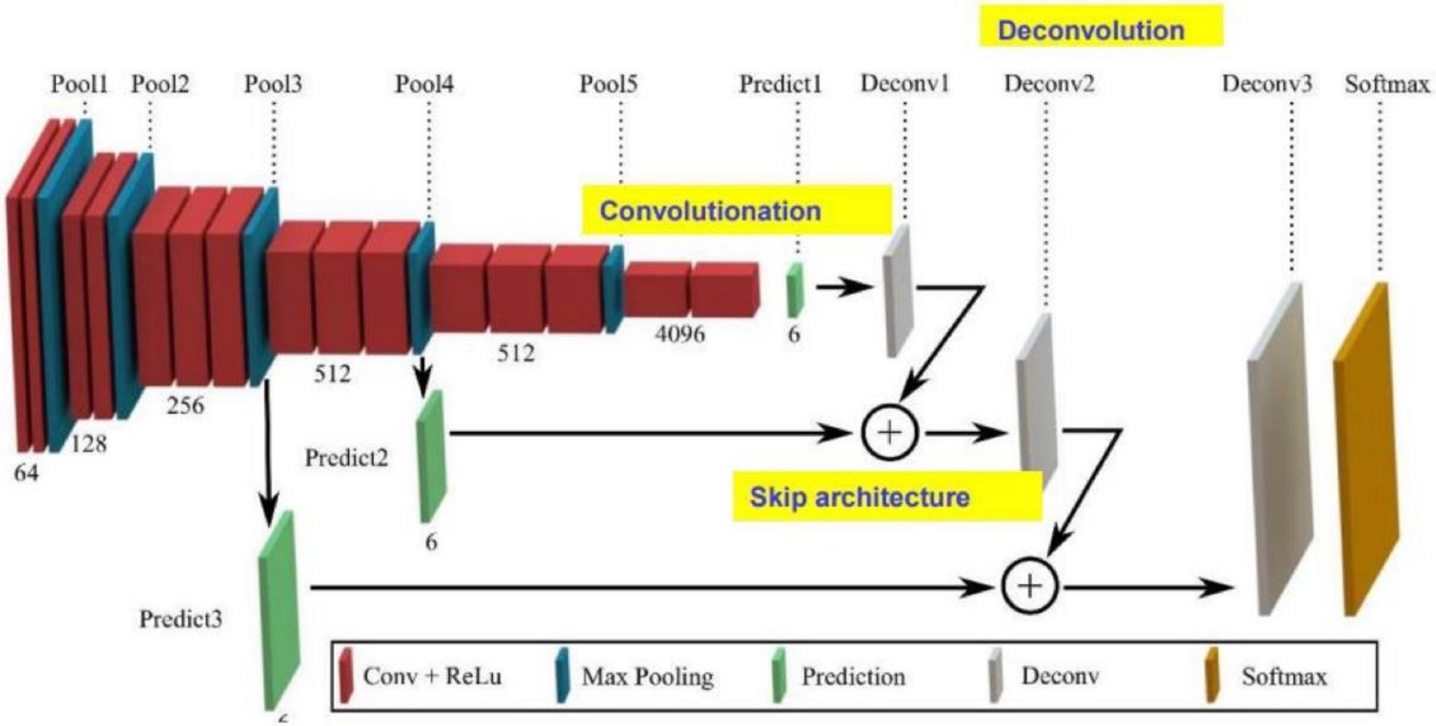


3. Why CNN?

FCN

(Fully Convolution Network)

1. Convolutionation
2. Deconvolution
3. Skip Architecture



3. Why CNN?

FCN

Convolutionation

기존에는 다른 사이즈의 Input image를 넣으면 차원 문제가 발생

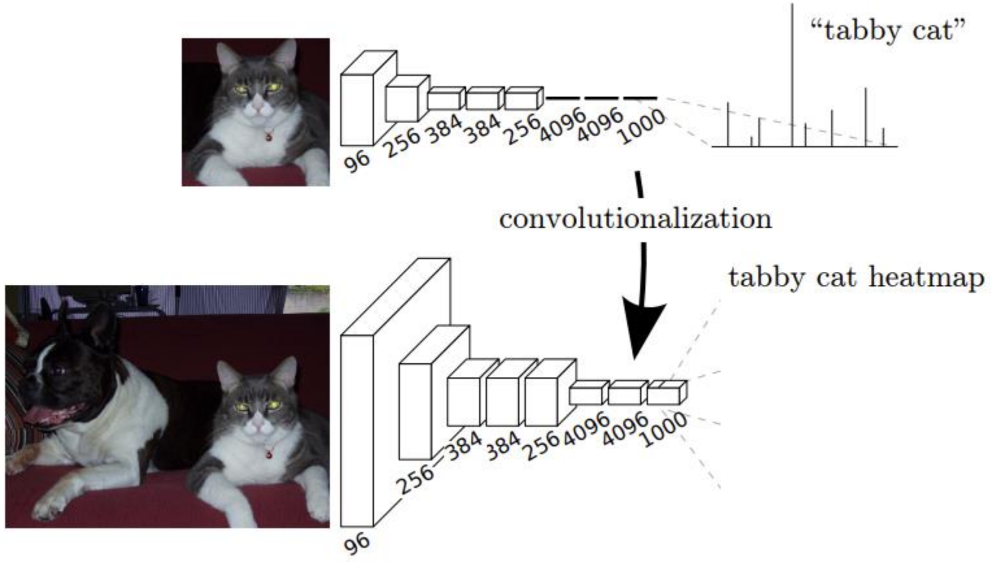
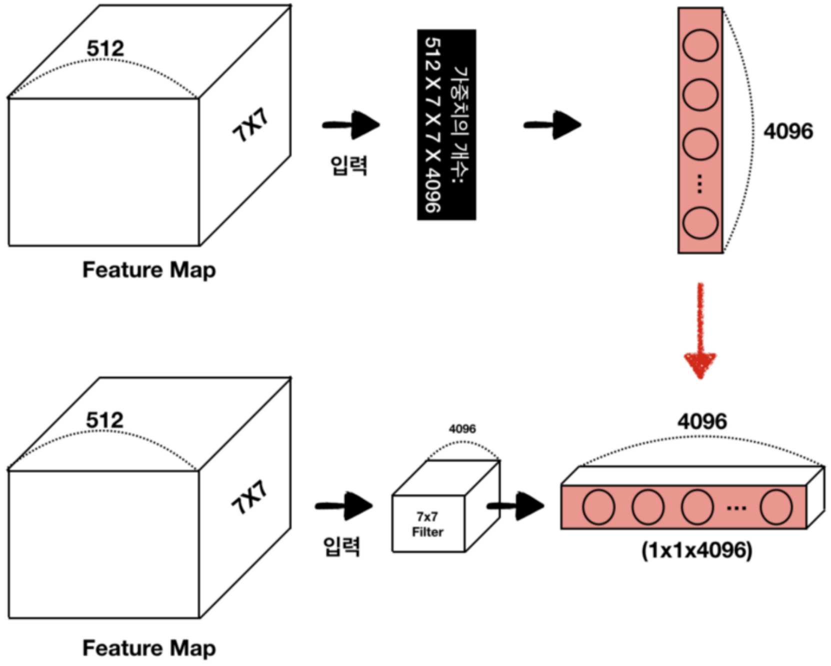


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.



FC layer 쓰면 : # weights vary with input vector size

Convolution layer : # weights <- kernel size로 결정

3. Why CNN?

FCN

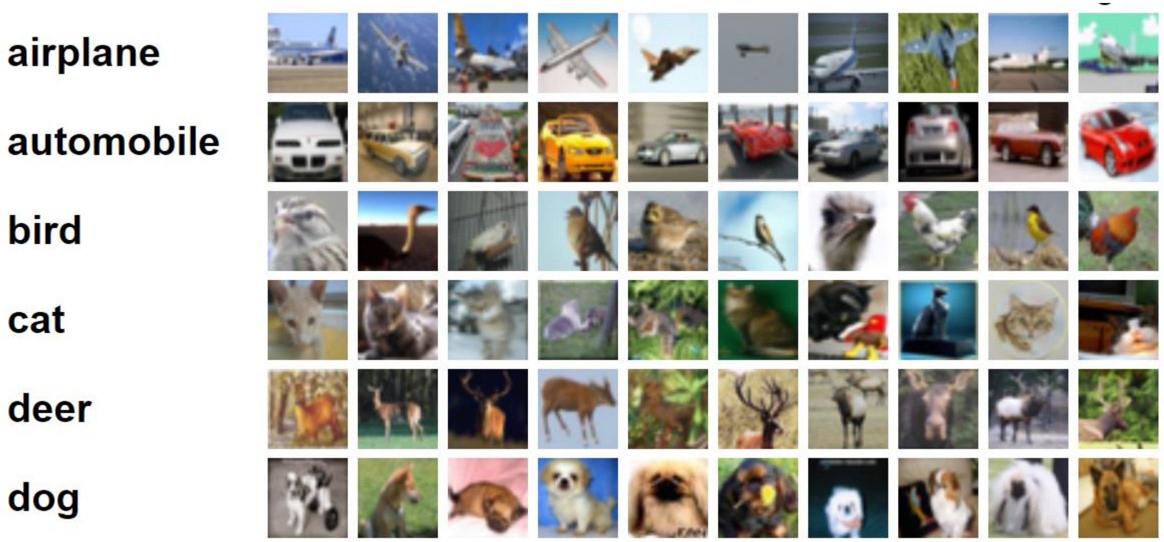
Skip-Connection

Feature map size가 달라 elementwise calculation이 안됐었는데…

1*1 convolution 이용하자!

4. Transfer Learning

전이학습



고해상도 컬러 이미지를 인식하기 위해서는?

최소 5개 이상의 convolution layer + 2개 이상의 fc layer 로 train을 진행하고

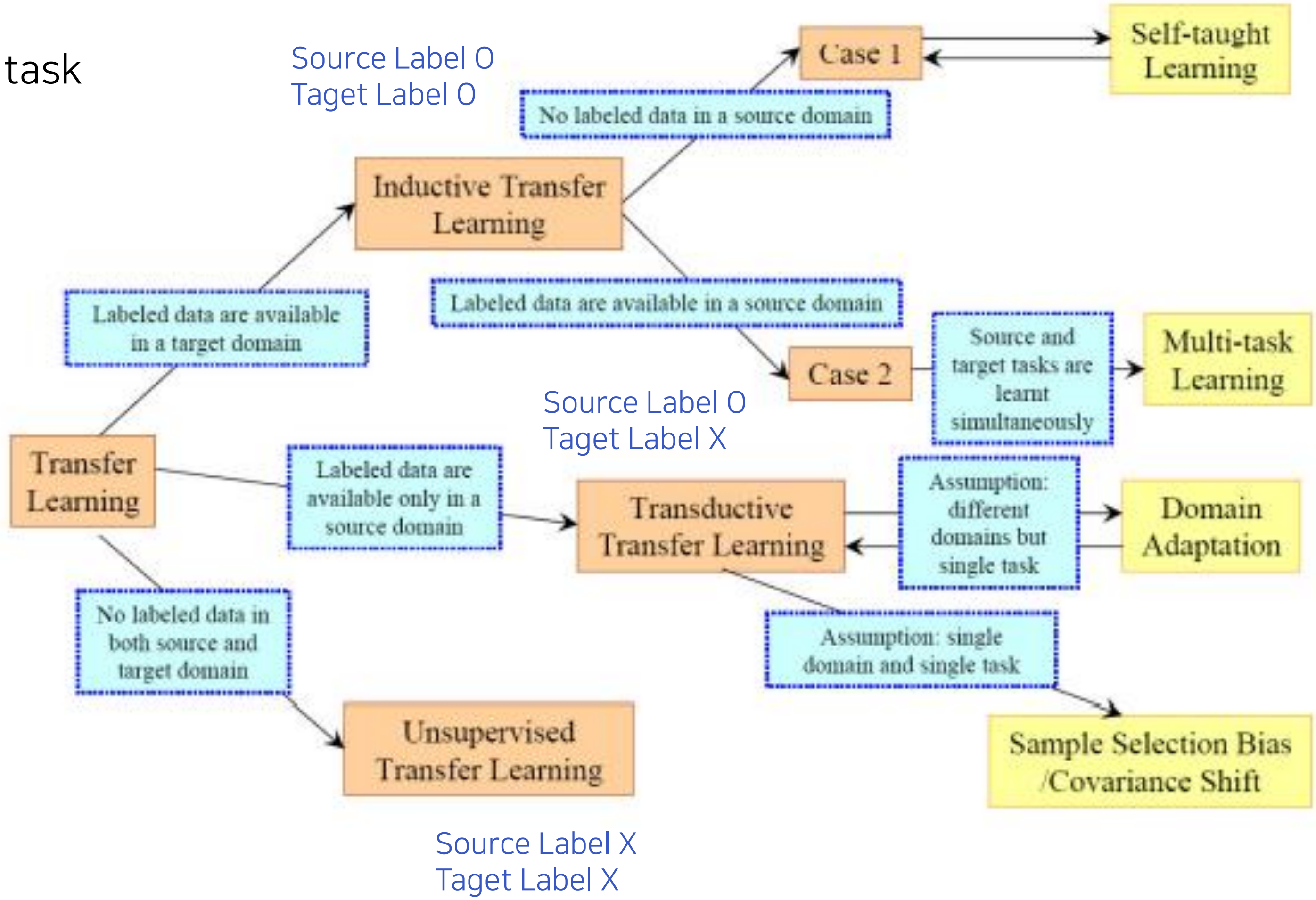
1개의 CPU 환경이라면 수백~수천시간이 소요됨

돈도 없고…시간도 없는데… -> pretrained model을 이용해보자!

4. Transfer Learning

전이학습 종류

Source task vs target task

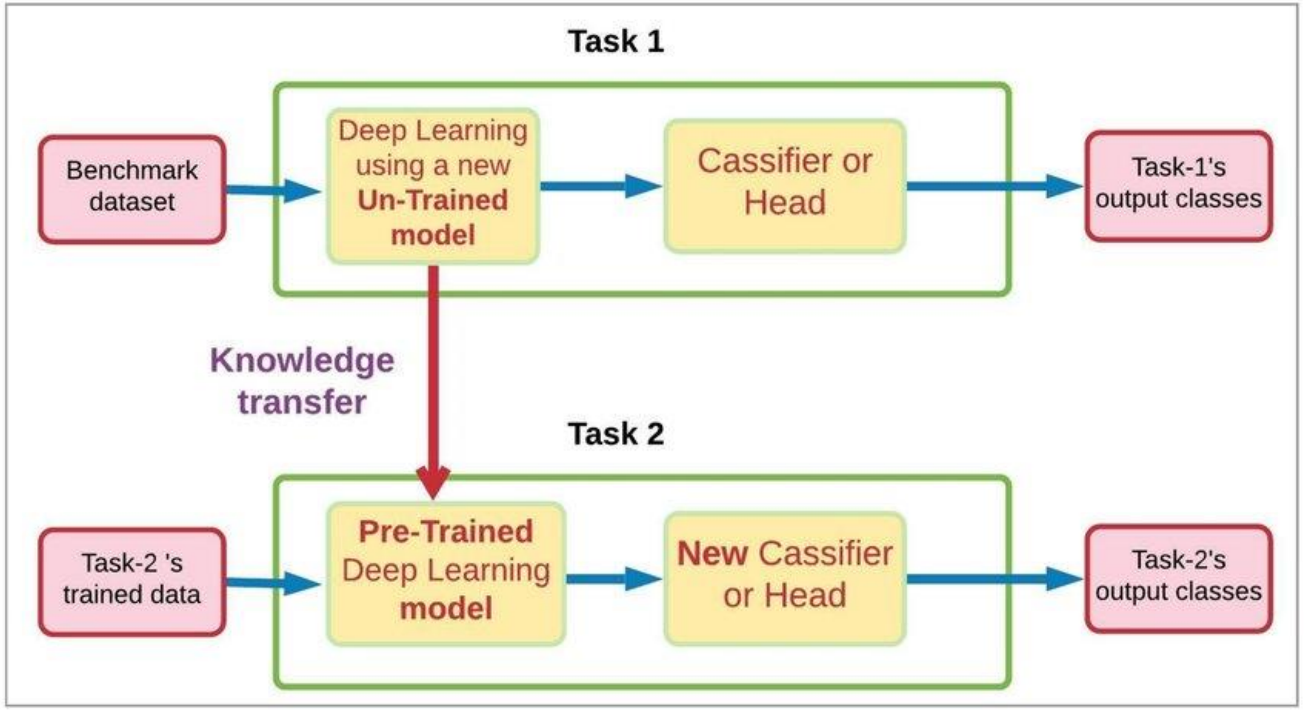


4. Transfer Learning

전이학습

Freeze : weight 값을 그대로 가져오기

Fine-tuning : 원래 learning rate를 줄여 weight를 미세 조정



1. 기존 classifier 부분 제거
2. Pre-traine된 feature extractor 부분 가져오기
3. New Classifier
4. fine-tuning 진행

4. Transfer Learning

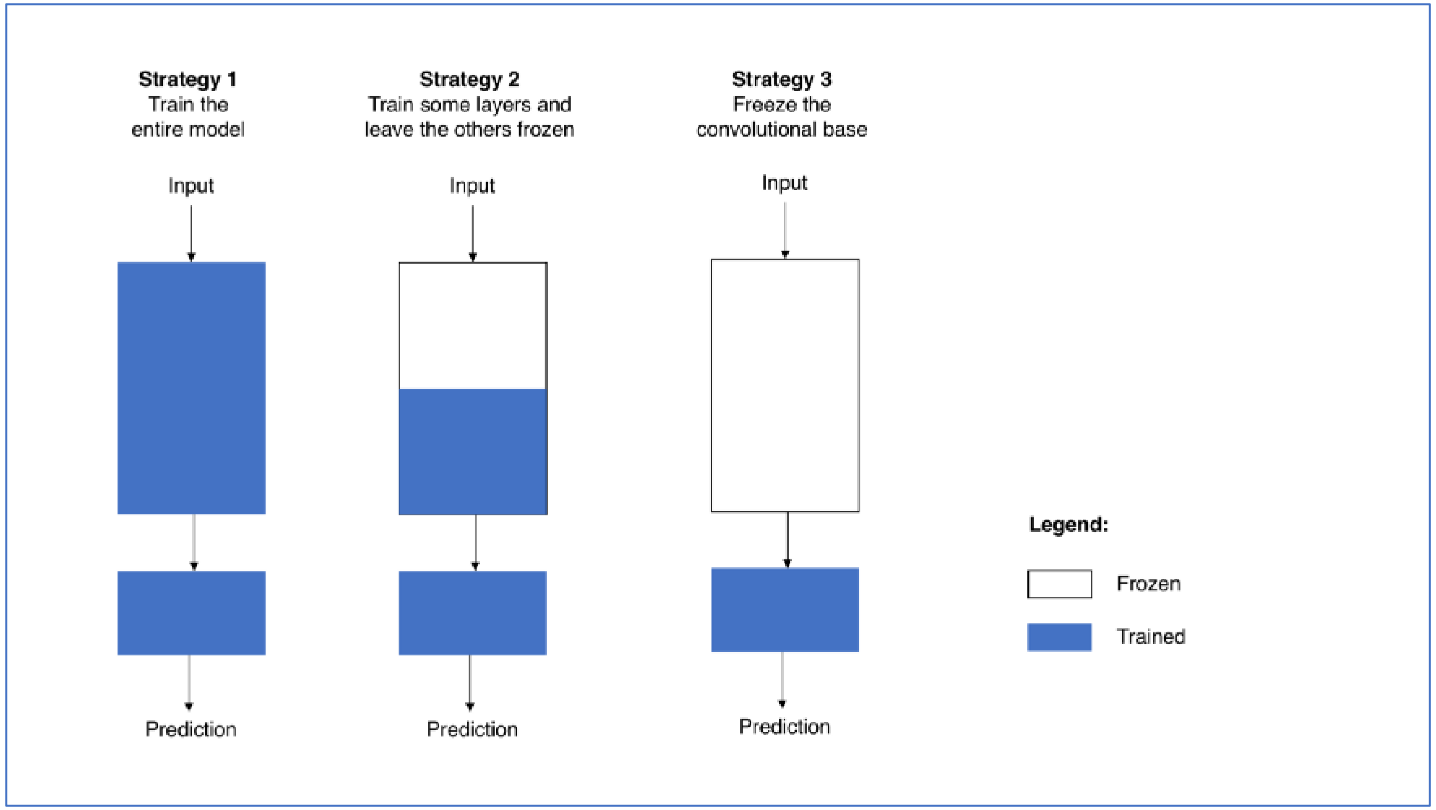
전이학습

Fine-tuning

Strategy 1

Train the entire model

: dataset이 클 때 가능



4. Transfer Learning

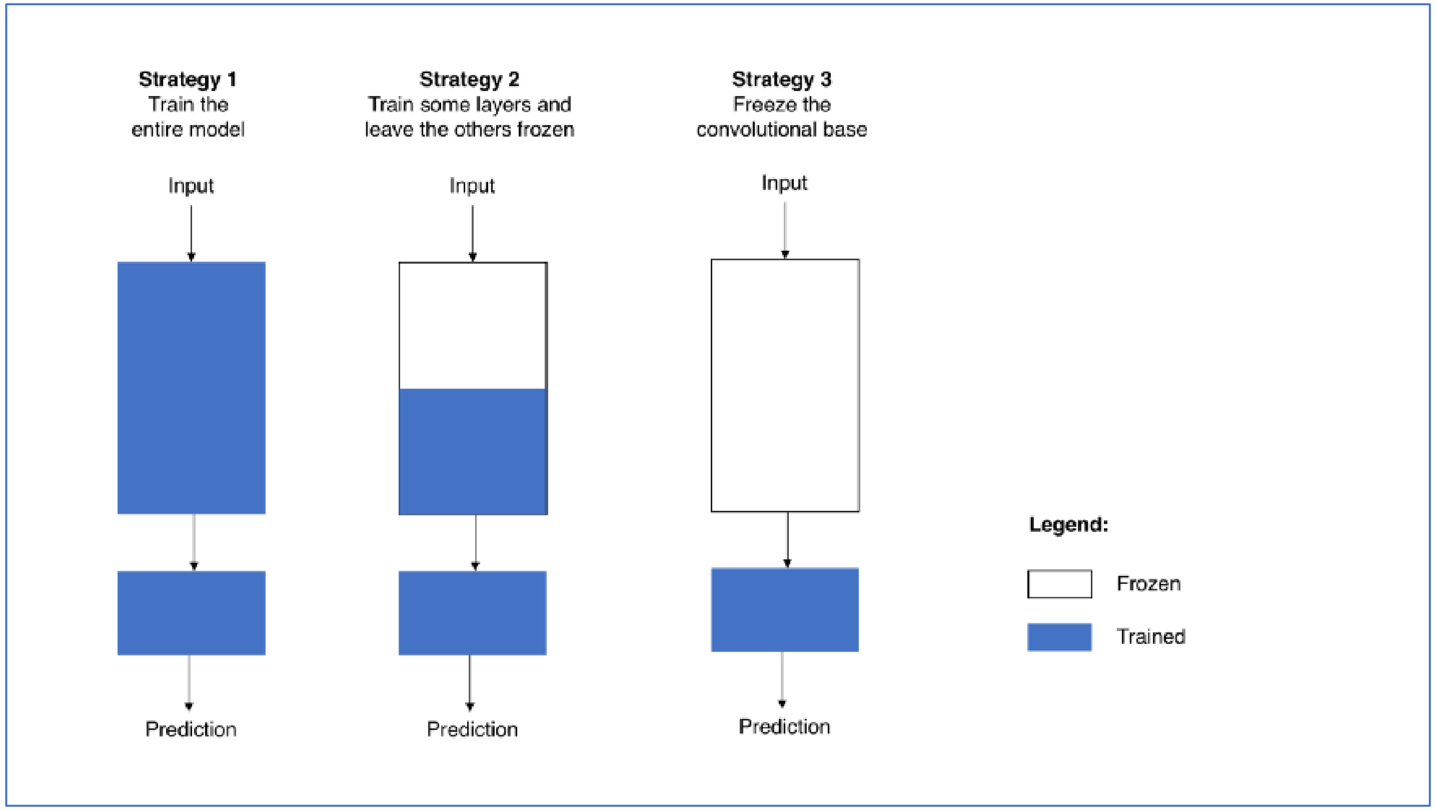
전이학습

Fine-tuning

Strategy 2

Train some layers

- feature extraction 시에 low-level Features부터 추출
- Task가 비슷하다면 굳이 모든 layer를 다 train할 필요는 없음



4. Transfer Learning

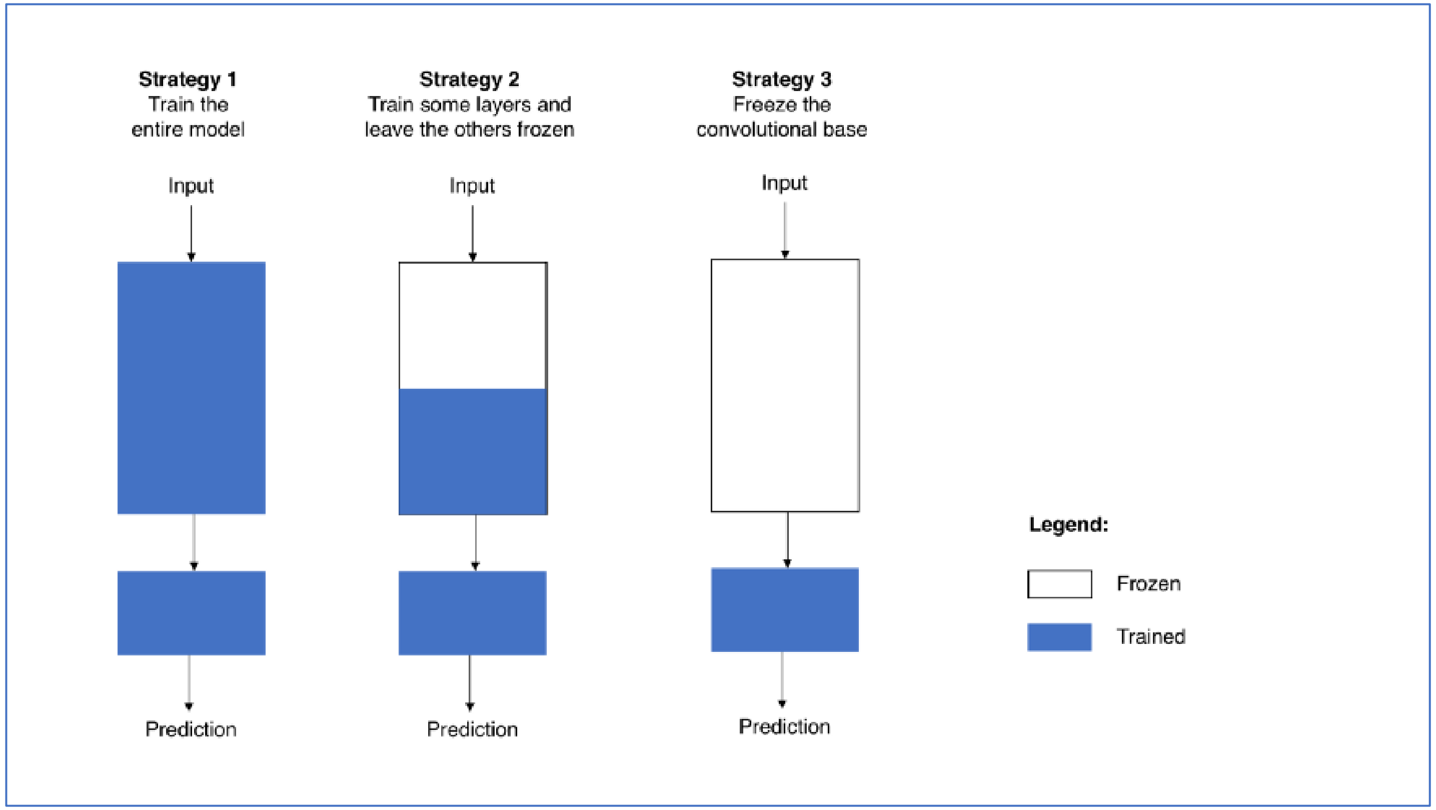
전이학습

Fine-tuning

Strategy 3

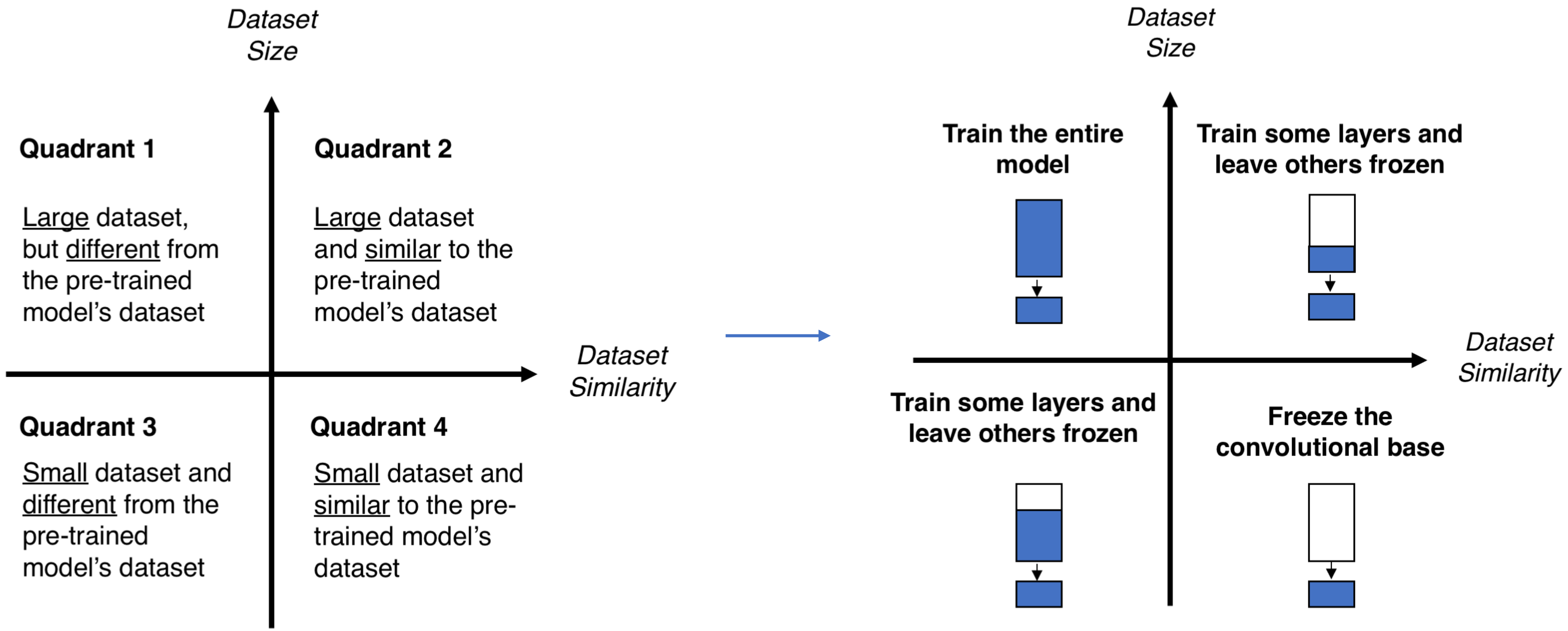
Freeze convolutional base

- Dataset 작고 task가 유사하다면 유리



4. Transfer Learning

Size Similarity Matrix



6. Summary

Speed of Convergence

- Weight Initialization – Xavier, He
- Learning rate – cosine annealing with warmup
- Batch Normalization

Why CNN?

- 이미지 처리에 특화되어 있지만… 다른 분야(RNN, 시계열)에서도 충분히 사용 가능하다

Transfer Learning

- Power tool when we want to train with a small amount of data

6. Summary

Reference

- 22-2 정규세션 : 7기 전재현님 <CNN>
- 22-2 정규세션 : 7기 이승연님 <CNN Implementation>
- https://www.youtube.com/watch?v=ScWTYHQra5E&t=370s&ab_channel=SebastianRaschka **L11.6 Xavier Glorot and Kaiming He Initialization**
- http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html Deconvolution 관련
- https://gaussian37.github.io/dl-concept-batchnorm/ Batch normalization
- Efficient Object Localization Using Convolutional Networks, Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, Christoph Bregler New York University(2015)

DATA
SCIENCE LAB

발표자 유채원 010 – 8736 - 1915
E-mail: elinayoo71@yonsei.ac.kr