6. Techniques for Training Neural Networks

Seongchan Kim

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

자기소개

- 이름: 김성찬
- 소속: 고려대학교 정보대학 컴퓨터학과
- 짬
  - 이제 곧 4학년 😭
  - AIKU 전 학회장 🤩
  - 고려대학교 컴퓨터비전 연구실 CVLAB 학부 인턴 😍
- 관심 분야
  - NeRF 🔫
  - Video 🎬
  - 과연 인공지능의 정의란 무엇인가?

Overview of Today's Topic

- 머신러닝 : 데이터를 통해 모델을 학습시켜 시스템의 성능을 향상시키는 기술

Overview of Today's Topic

- 머신러닝 : 데이터를 통해 모델을 학습시켜 시스템의 성능을 향상시키는 기술

- MLP, CNN, RNN, Transformer ➔ 모델 구성 요소

Overview of Today's Topic

- 머신러닝 : 데이터를 통해 모델을 학습시켜 시스템의 성능을 향상시키는 기술
- MLP, CNN, RNN, Transformer ➔ 모델 구성 요소
- 모델이 중요한 구성요소이지만, 딥러닝은 다양한 요소가 함께 고려되어야 함.

Overview of Today's Topic

- 머신러닝 : 데이터를 통해 모델을 학습시켜 시스템의 성능을 향상시키는 기술
- MLP, CNN, RNN, Transformer ➔ 모델 구성 요소
- 모델이 중요한 구성요소이지만, 딥러닝은 다양한 요소가 함께 고려되어야 함.
- 어떻게 학습할 것인가?

Contents

- How to Train Well?
  - Criteria
  - Robust and Efficient System
- How to Robust?
  - Overfitting and Underfitting
  - Methods for Robustness
- How to Efficient?
  - Time and Space
  - Methods for Efficiency

How to Train Well?

How to Train Well? : Criteria

Criteria

모델이 잘 학습되었는가를 알기 위해서는 그에 대한 기준이 필요하다.

Criteria

모델이 잘 학습되었는가를 알기 위해서는 그에 대한 기준이 필요하다.

Robust and Efficient

Criteria

Robust and Efficient

Criteria

- 학습 데이터와 검증 데이터가 다르다면, 어떤 결과가 나타날까?

- 예를 들어, 학습 데이터는 실제 강아지 사진로,
  검증 데이터는 강아지 그림으로 구성되어 있다면?

Criteria

Robust and **Efficient**

Criteria

- Training Time
- Real-Time

- Data Efficient
- Model Size

How to Train Well? : Robust and Efficient System

Robust and Efficient System

- Robust System
  모델을 적용하는 과정에서 만나는 데이터와 학습 데이터 간의
  예상치 못한 데이터 분포의 차이가 있음에도 불구하고
  모델이 우리가 예상하는 결과를 보인다면 해당 모델은 **Robust**하다.

- Efficient System
  모델이 학습하는 과정 또는 적용되는 과정에서
  시공간적으로 효율적이고 효과적으로 작동한다면
  해당 모델 또는 시스템은 **Efficient**하다.

How to Robust? : Overfitting and Underfitting

Overfitting and Underfitting : Evaluation

- 학습과 평가는 함께 이루어져야 한다.

1. Validation
   학습 데이터 : 검증 데이터 = 8 : 2
2. Cross Validation
   1. LOOCV (Leave-One-Out Cross Validation)
   2. K-Fold Cross Validation

Overfitting and Underfitting : Evaluation

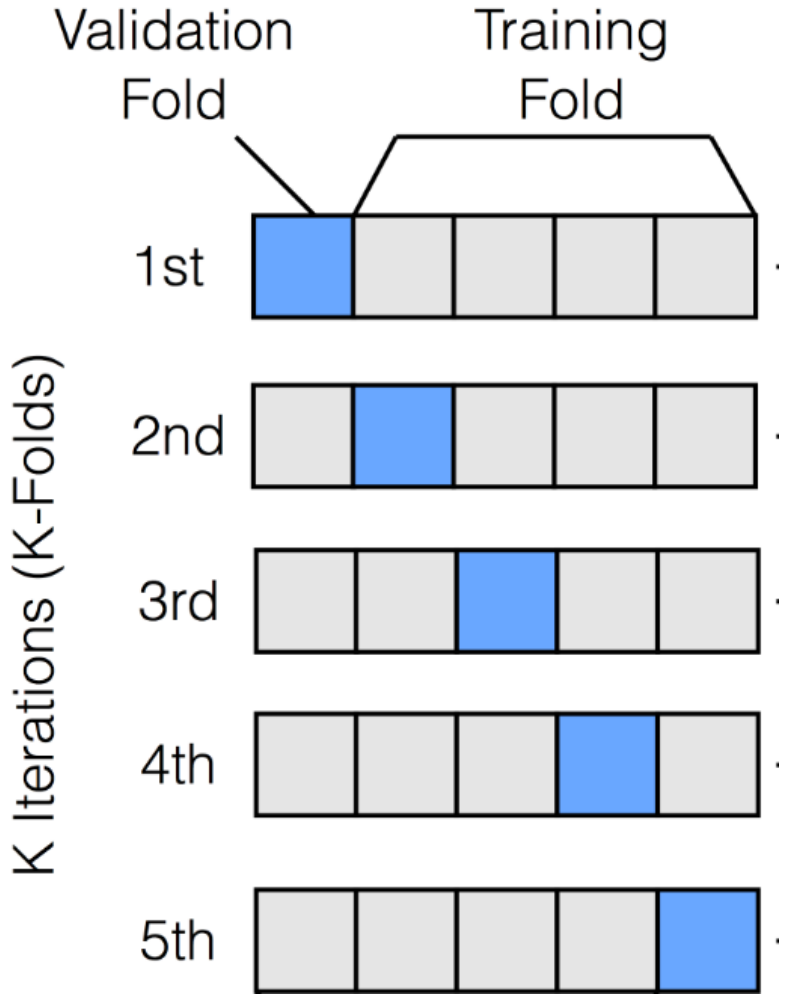
- 학습과 평가는 함께 이루어져야 한다.

1. Validation
2. Cross Validation
    1. LOOCV (Leave-One-Out Cross Validation)
       전체 데이터 크기 = N, 다음 과정을 N번 반복
        1. 검증 데이터 1개 샘플링
        2. N – 1개의 학습 데이터 학습 후 검증
    2. K-Fold Cross Validation

Overfitting and Underfitting : Evaluation

- 학습과 평가는 함께 이루어져야 한다.

1. Validation
2. Cross Validation
    1. LOOCV (Leave-One-Out Cross Validation)
    2. K-Fold Cross Validation
       전체 데이터 크기 = N x K (N개씩 K 묶음)
       다음 과정을 K번 반복
        1. 검증 데이터 묶음 1개 샘플링
        2. 학습 데이터 묶음 K – 1개로 학습 후 검증



https://androidkt.com/pytorch-k-fold-cross-validation-using-dataloader-and-sklearn/

Overfitting and Underfitting : Error

- 데이터를 나타내는 기저함수가 존재한다.

$$y = f(x) + \epsilon$$

- 실제 데이터는 노이즈가 포함되어 있다.

$$\hat{y} = \hat{f}(x)$$

- 기저함수를 잘 근사하는 함수를 모델링하는 것이 목표다.

$$Error(x) = E[(y - \hat{f})^2]$$

Overfitting and Underfitting : Error

에러의 구성

- 데이터 내 노이즈
- 모델의 분산
- 모델의 편향

$$Error(x) = E[(y - \hat{f})^2]$$

$$= E[y^2] + E[\hat{f}^2] - 2E[y\hat{f}]$$

$$= Var[y] + \{E[y]\}^2 + Var[\hat{f}] + \{E[\hat{f}]\}^2 - 2E[(f + \epsilon)\hat{f}]$$

$$= Var[y] + Var[\hat{f}] + f^2 + \{E[\hat{f}]\}^2 - 2E[f\hat{f}]$$

$$= \sigma^2 + Var[\hat{f}] + (f - E[\hat{f}])^2 = \sigma^2 + Var[\hat{f}] + Bias[\hat{f}]^2$$

Overfitting and Underfitting : Bias and Variance

- Model Complexity ↑ Representation ↑ : Powerful

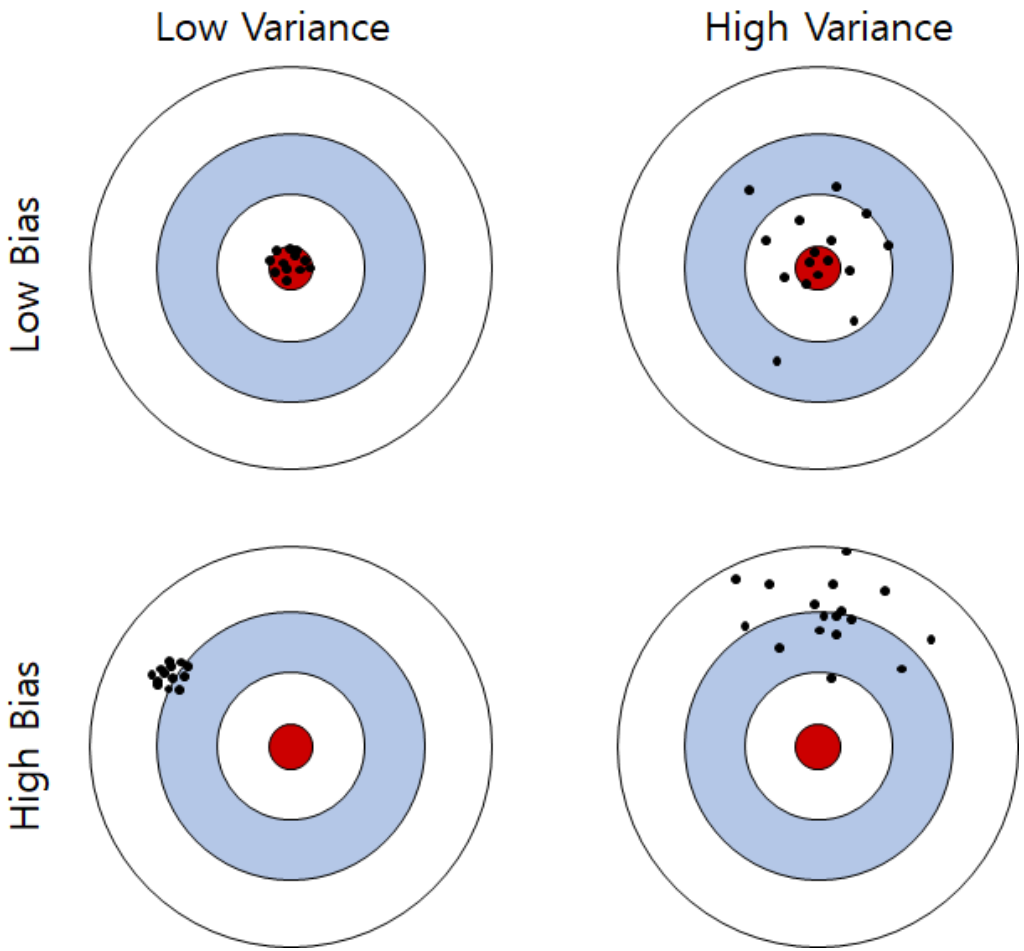
- Bias : 모델이 복잡할수록 편향은 작다.

  모델이 단순할수록 모델이 가지는 성질에 의한 한계가 크다.

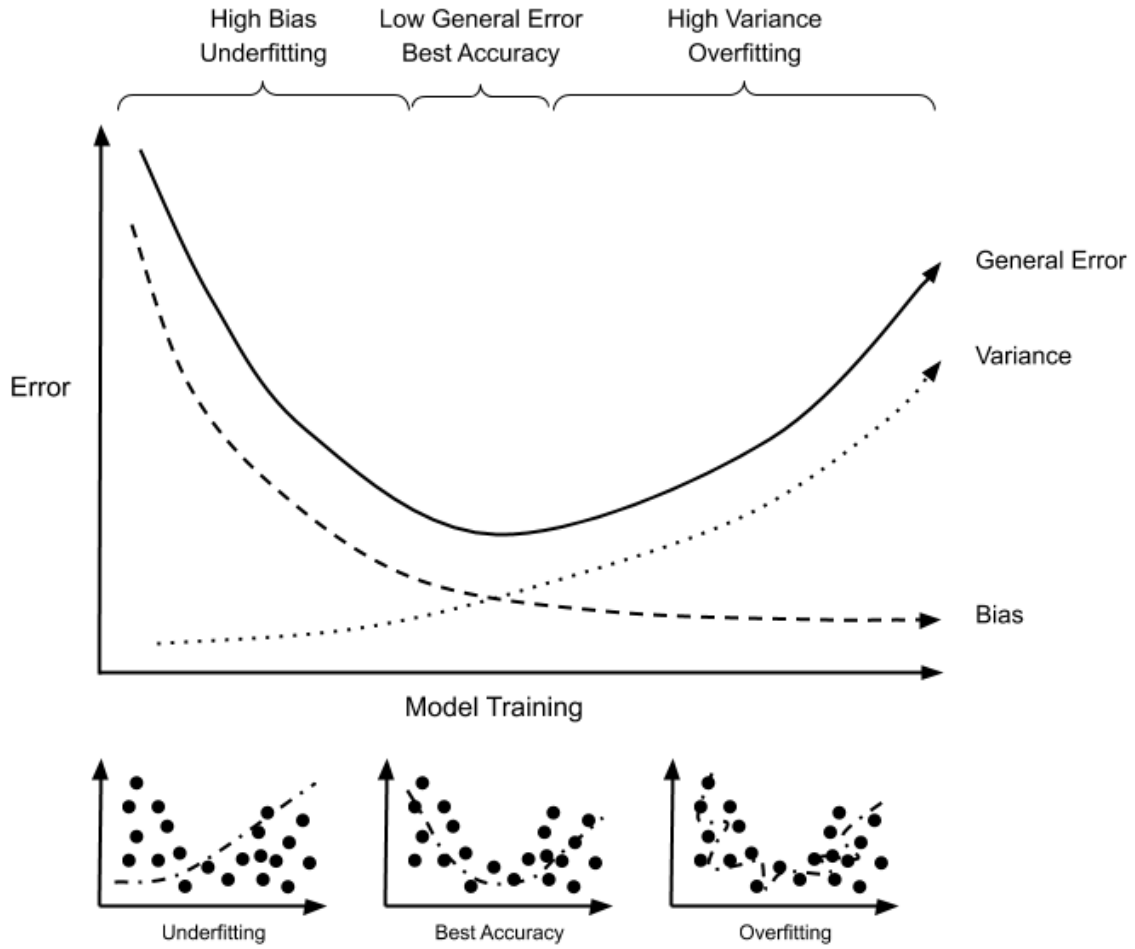
- Variance : 모델이 단순할수록 분산은 작다.

  모델이 복잡할수록 데이터 내 노이즈까지 학습하기 쉽다.

Overfitting and Underfitting : Bias and Variance



https://m.blog.naver.com/ckdgus1433/221594203319



https://www.ml-science.com/bias-variance-tradeoff

Overfitting and Underfitting : What is Goal?

- 기본적으로 딥러닝 모델은 모델 복잡도가 크다.
  따라서 표현력이 크고, 데이터 내 노이즈까지 학습하기 쉽다.

- 이는 모델의 분산이 커지는, 오버피팅이 발생하는 요인이다.

- Robust한 모델을 만들기 위해서는 데이터 내 노이즈가 아닌,
  데이터를 잘 설명하는 패턴을 인식하는 것이 굉장히 중요하다.

How to Robust? : Methods for Robustness

Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization

Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

- Weight Decay : make simple model
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization

$$\mathcal{L}_{L1} = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

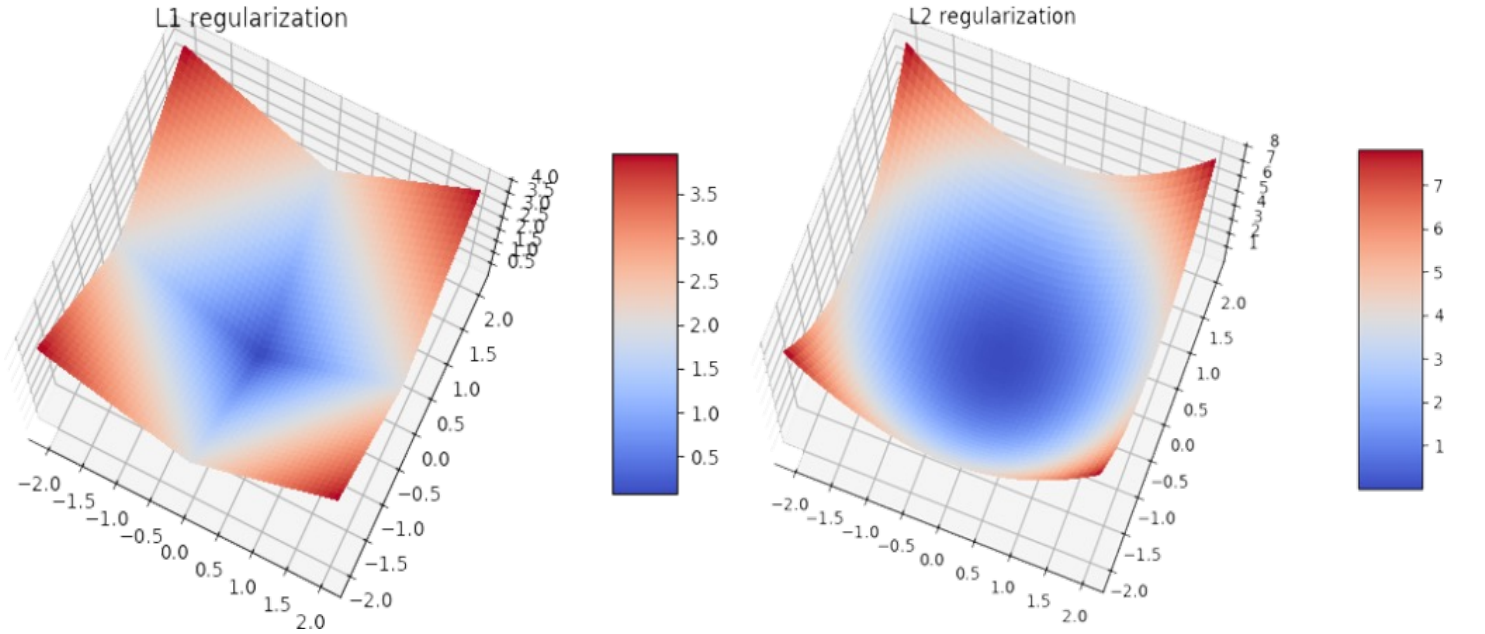
- Weight Decay : make simple model
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization

$$\mathcal{L}_{L2} = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization



https://laid.delanover.com/difference-between-l1-and-l2-regularization-implementation-and-visualization-in-tensorflow/

Methods for Robustness : Regularization

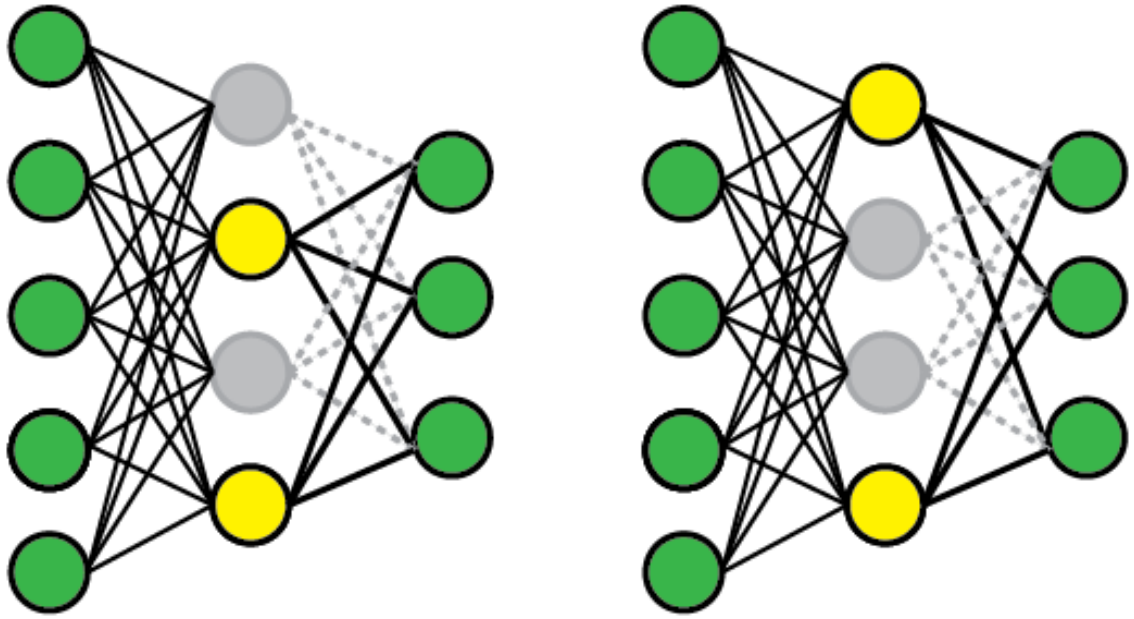
Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect : Randomness to Model
- Data Augmentation
- Entropy Regularization

Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

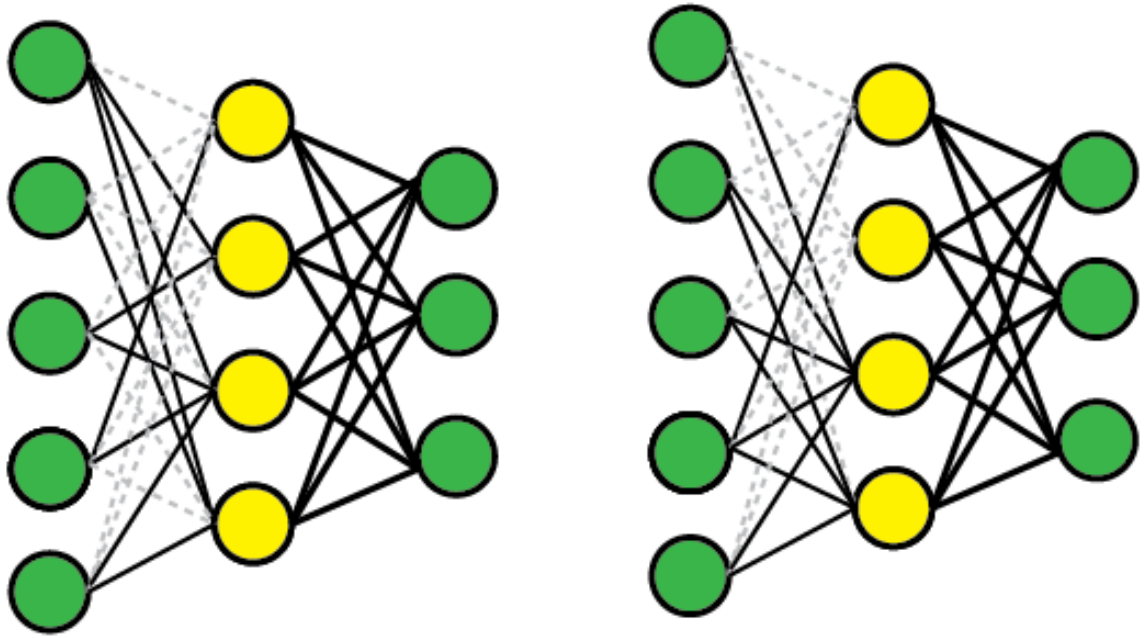
- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization



Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization



Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

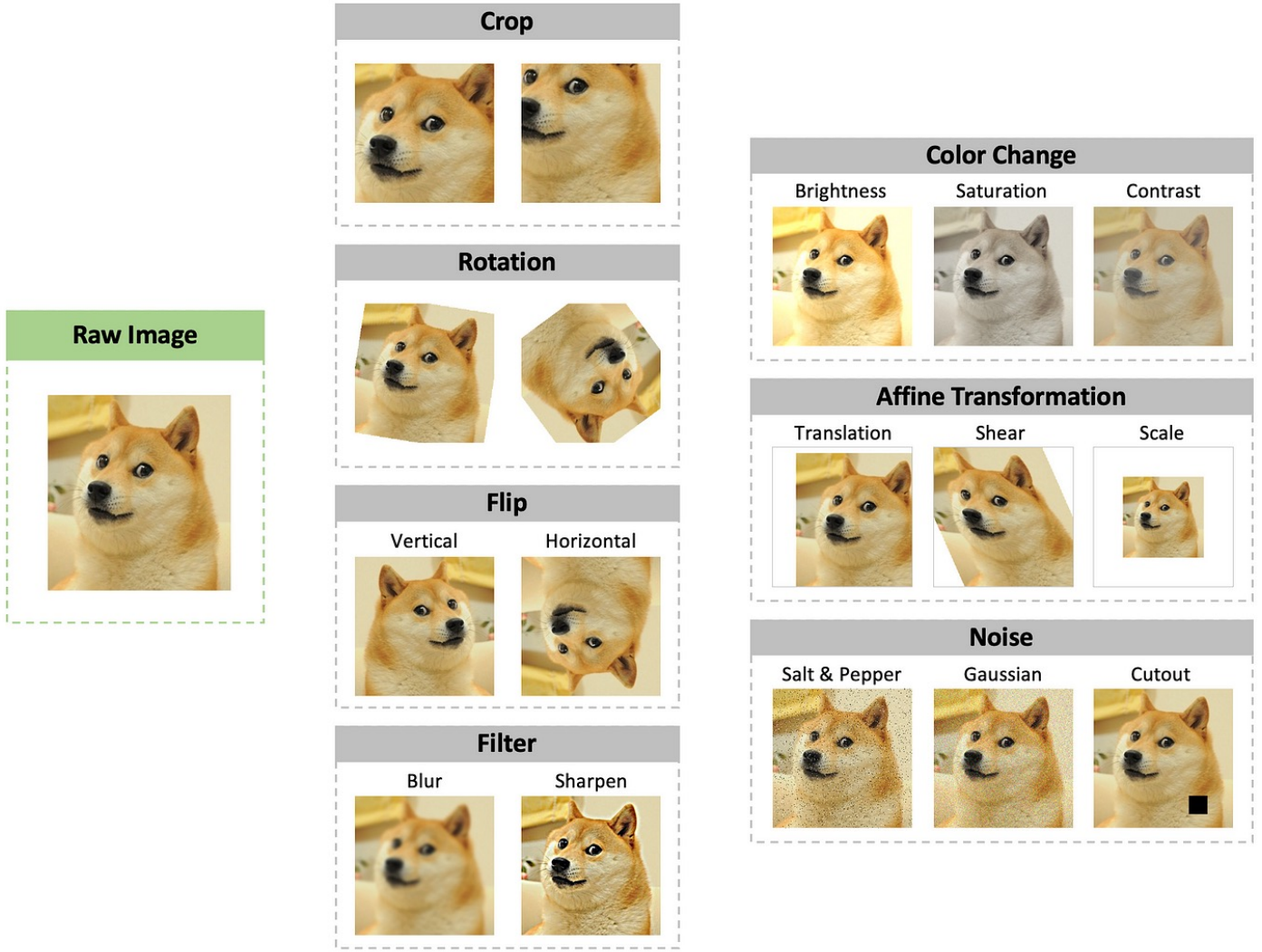
- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation : Randomness to Data
- Entropy Regularization

Methods for Robustness : Regularization

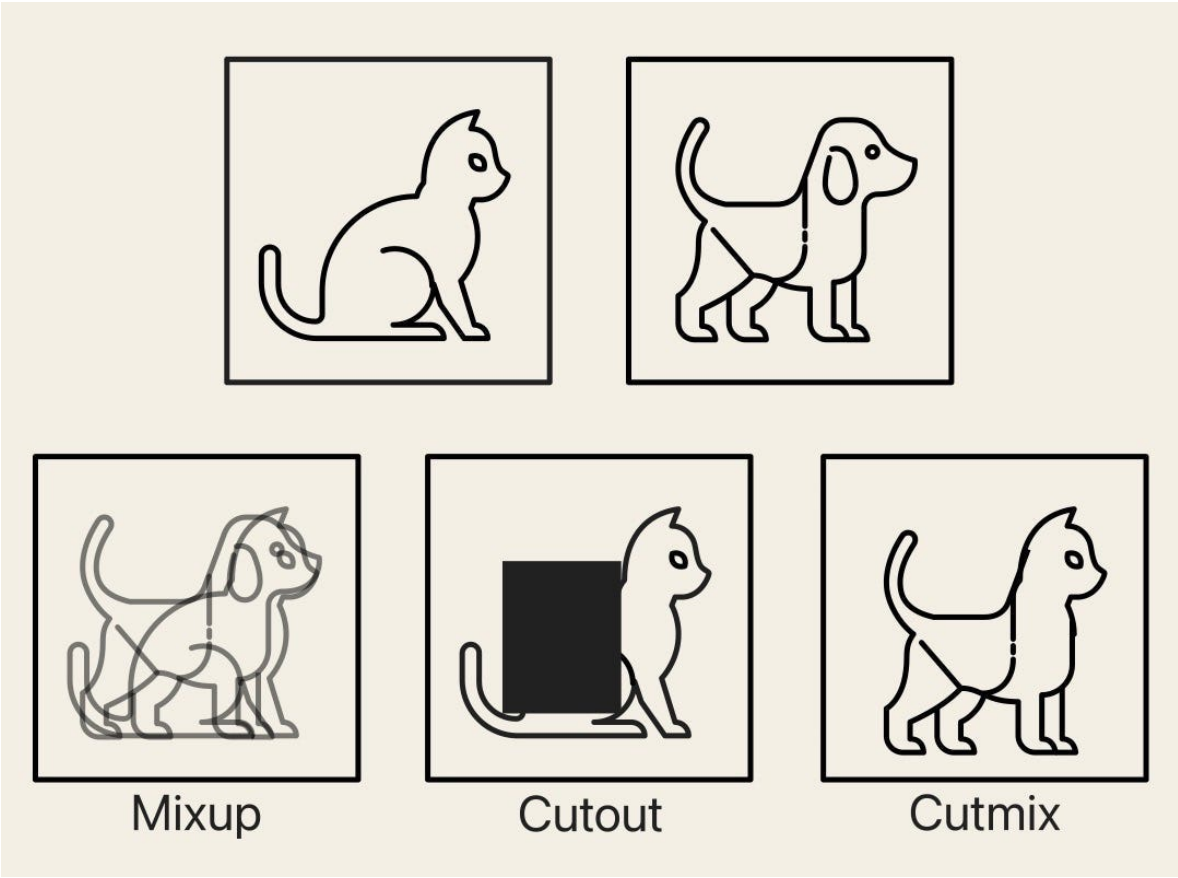
Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation : Computer Vision
- Entropy Regularization

Methods for Robustness : Regularization



Methods for Robustness : Regularization



Methods for Robustness : Regularization

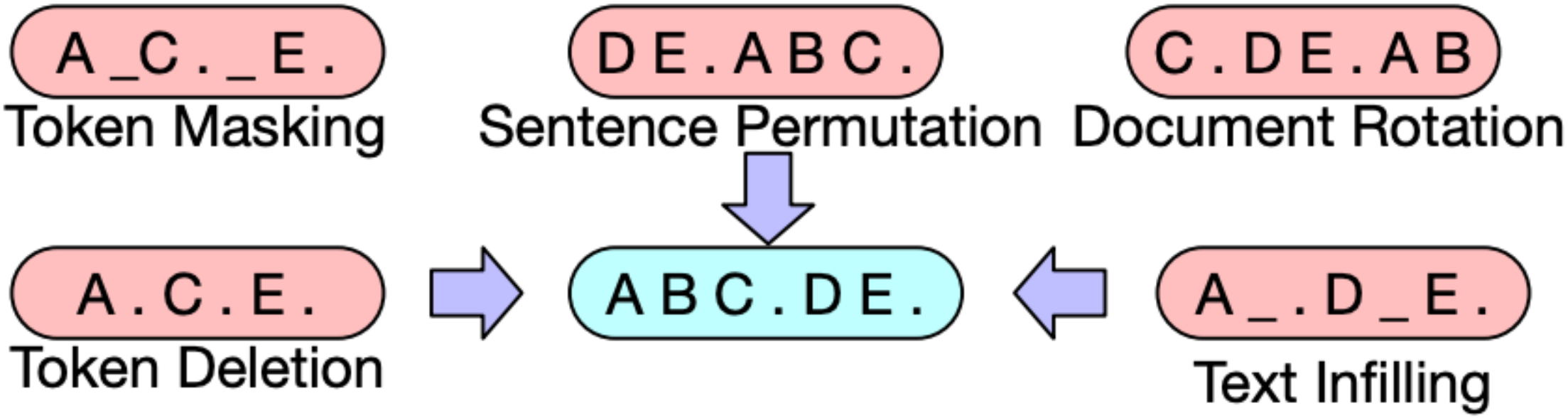
Regularization은 Randomness와 관련 있다!!!

- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation : Natural Language Process
- Entropy Regularization

Methods for Robustness : Regularization

- Back Translation
  - Ex) I have no time. → je n'ai pas le temps → I do not have time.
- Synonym Replacement
  - This article will focus on summarizing data augmentation techniques in NLP.
    → This write-up will focus on summarizing data augmentation methods in NLP.
- Random Insertion
  - This article will focus on summarizing data augmentation techniques in NLP.
    → This write-up will focus on summarizing data augmentation methods in NLP.
- Random Swap

Methods for Robustness : Regularization



+ Using Generative Models

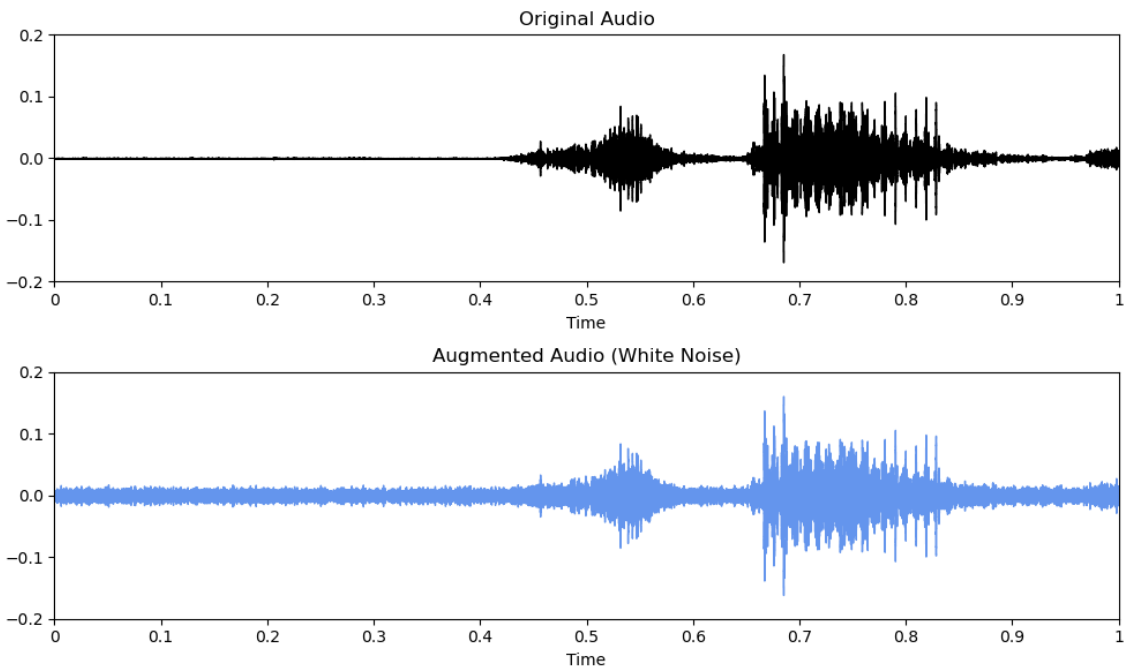
Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

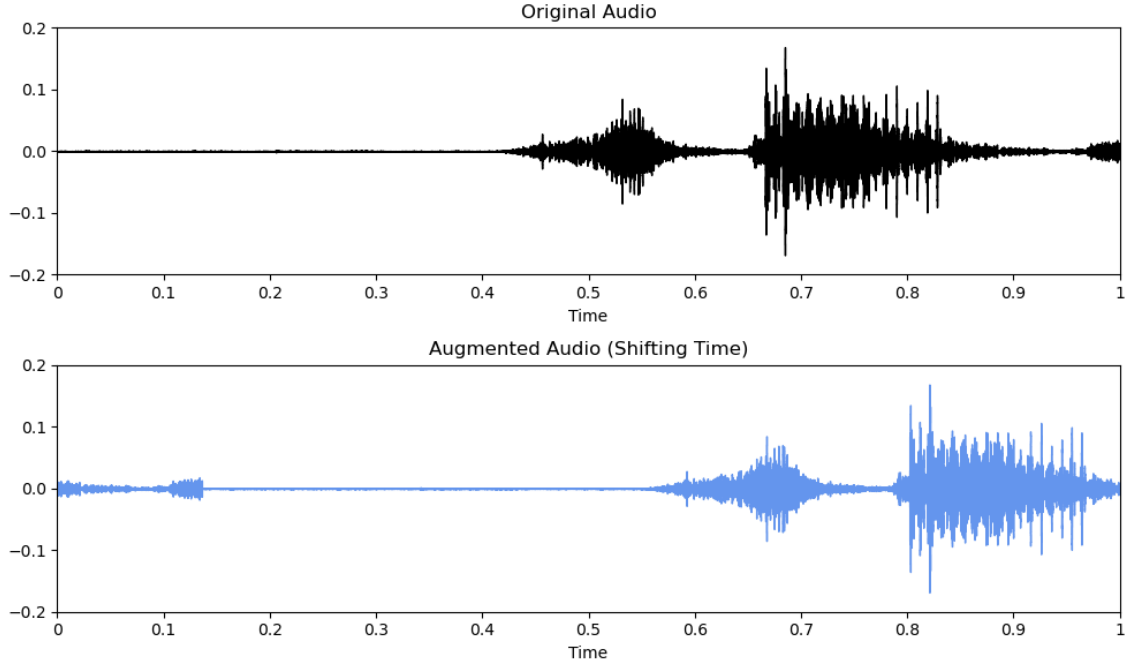
- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation : Audio
- Entropy Regularization

Methods for Robustness : Regularization

- Noise injection

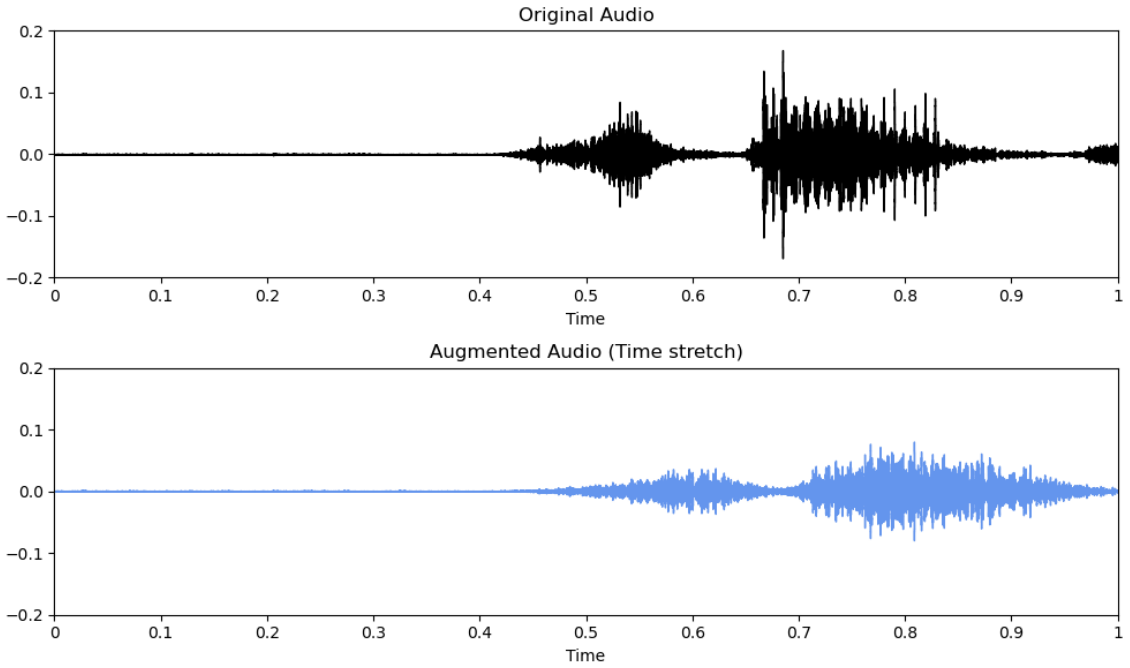


- Shifting time

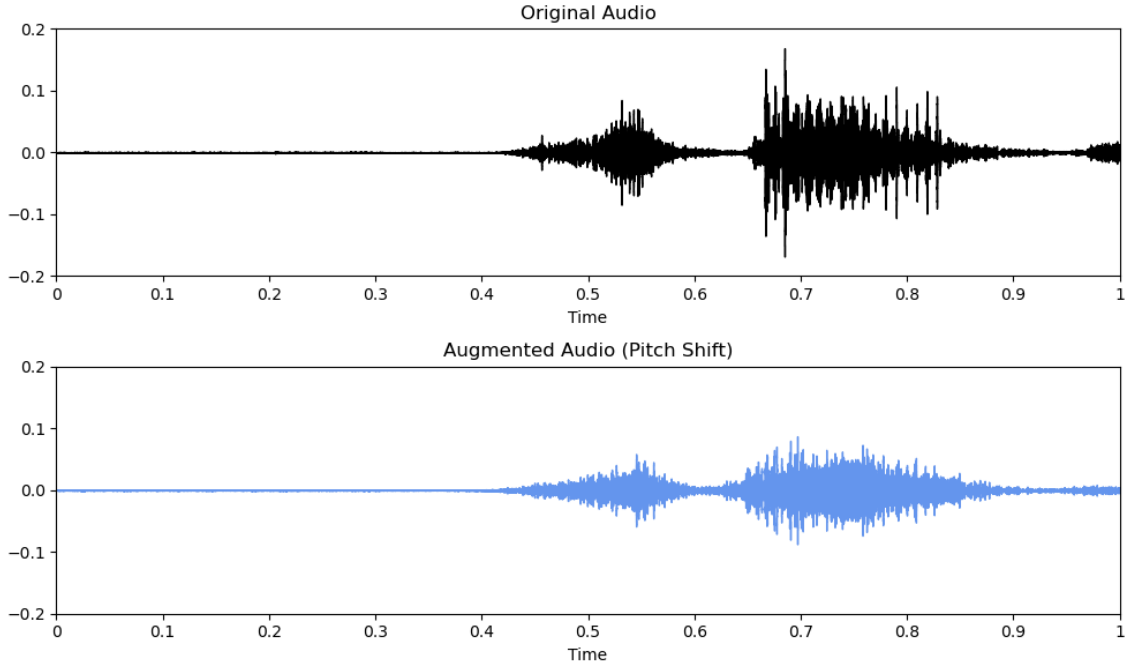


Methods for Robustness : Regularization

- Changing speed



- Changing pitch



Methods for Robustness : Regularization

Regularization은 Randomness와 관련 있다!!!

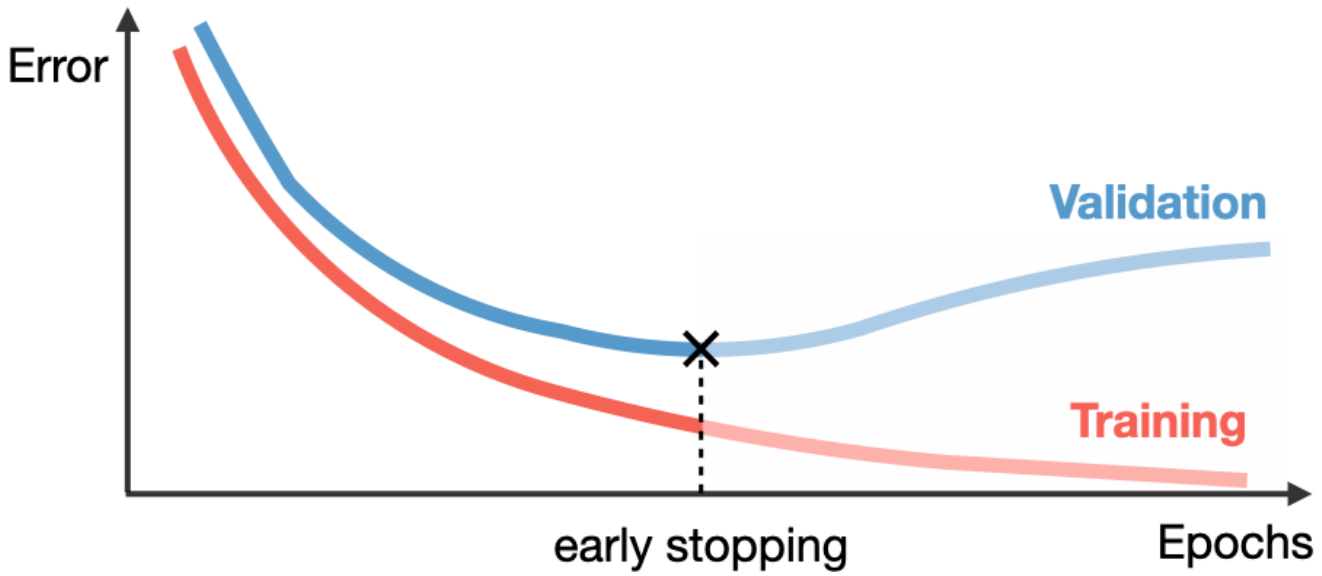
- Weight Decay
  - L1 Regularization
  - L2 Regularization
- Dropout & DropConnect
- Data Augmentation
- Entropy Regularization : minimize randomness

$$H(p) = -\sum_{k=1}^{K} p(y_k) \log p(y_k)$$

Methods for Robustness : Early Stopping

- 학습이 제대로 이루어진다면
  학습 데이터에 대한 Loss와
  검증 데이터에 대한 Loss가
  동시에 감소해야 한다.

- 검증 데이터에 대한 Loss가
  증가하기 시작하면
  학습을 종료한다.



Methods for Robustness : Contrastive Learning

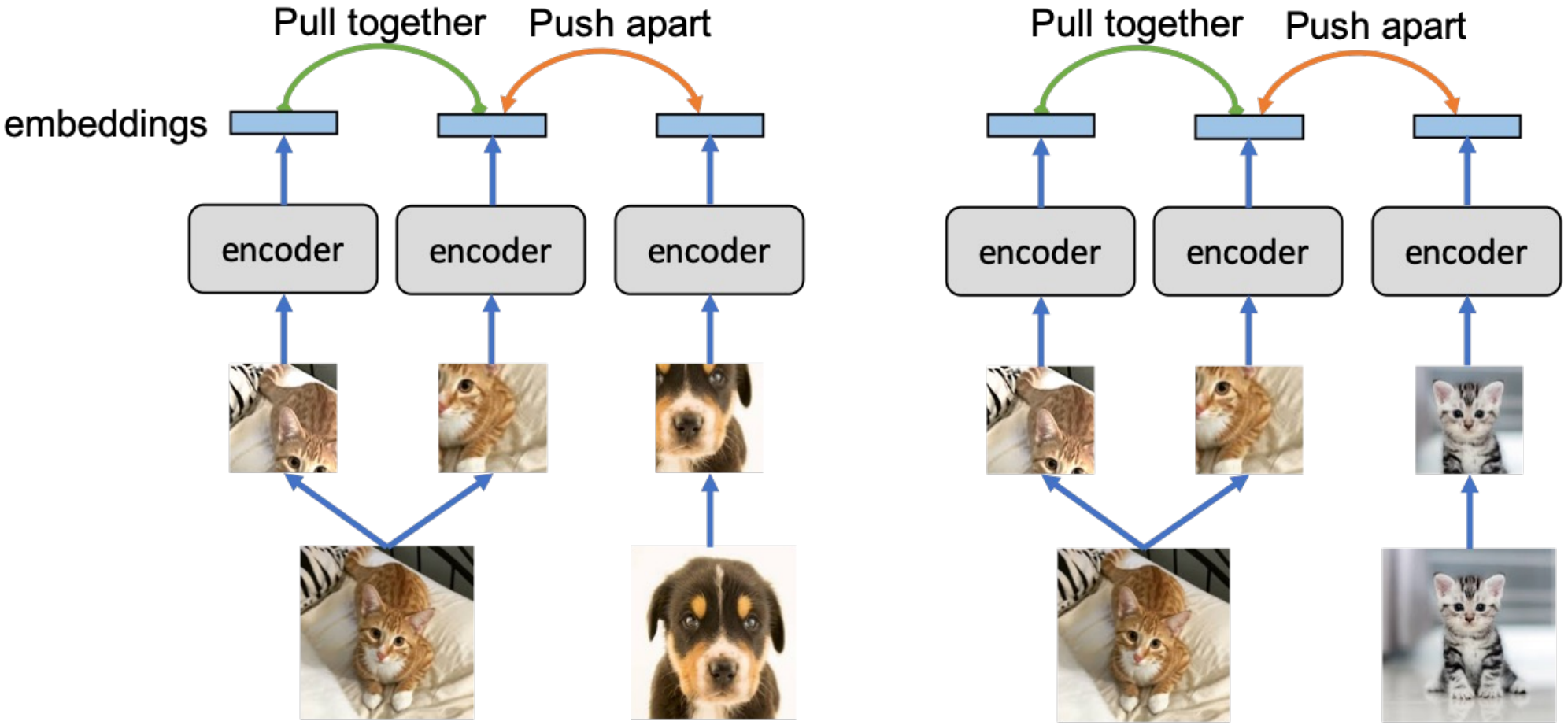


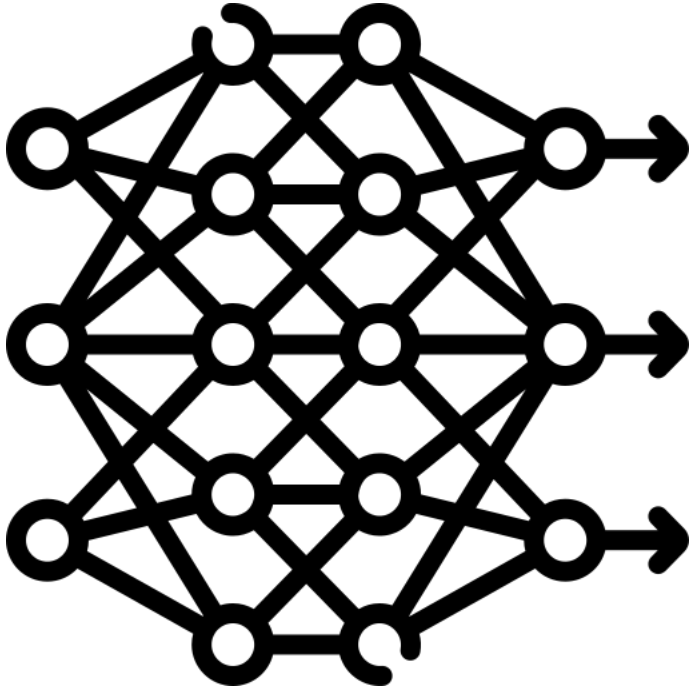
Figure 1 (a) (b)

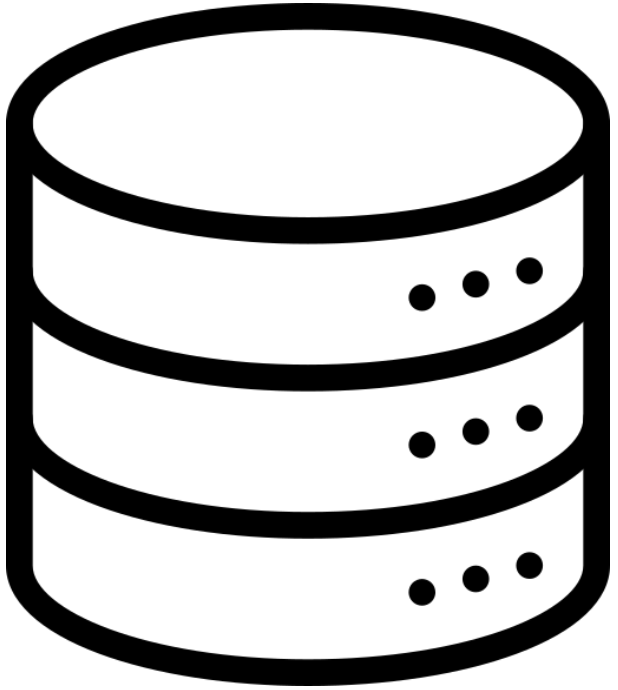
https://blog.salesforceairesearch.com/prototypical-contrastive-learning-pushing-the-frontiers-of-unsupervised-learning/

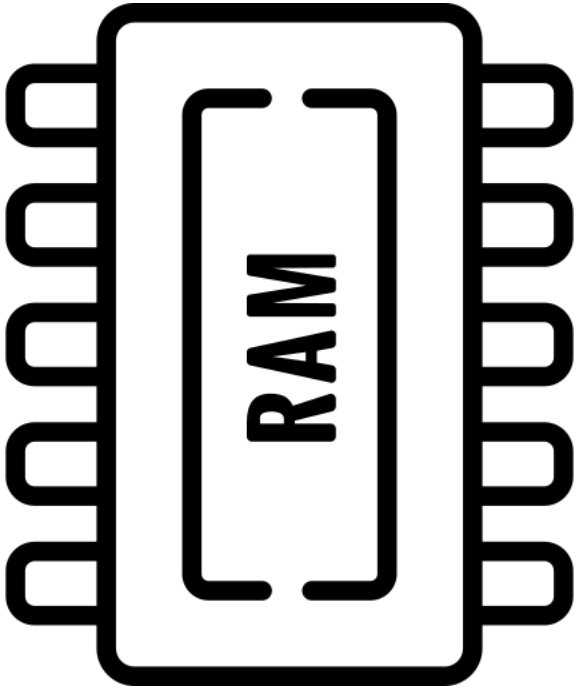
How to Efficient?

How to Efficient? : Space and Time

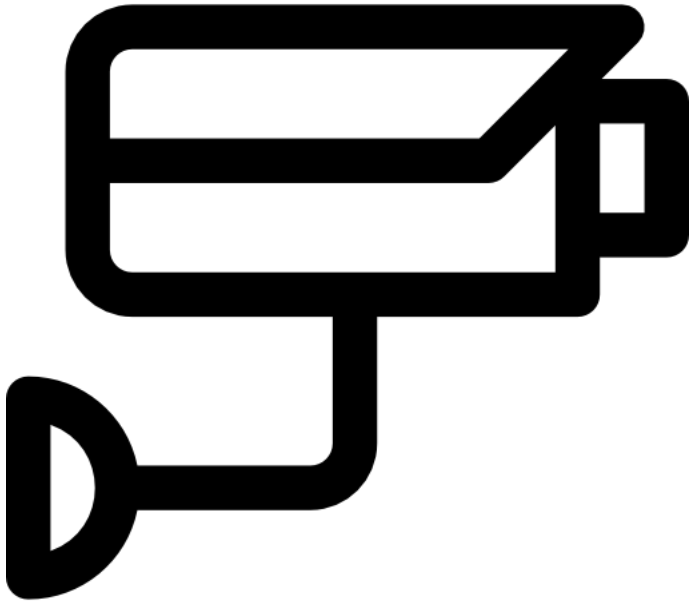
Space and Time : Space

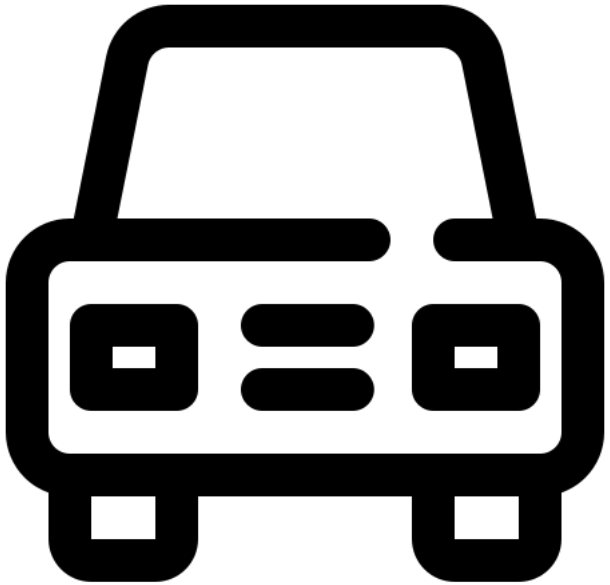






Space and Time : Time





How to Efficient? : Methods for Efficiency

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

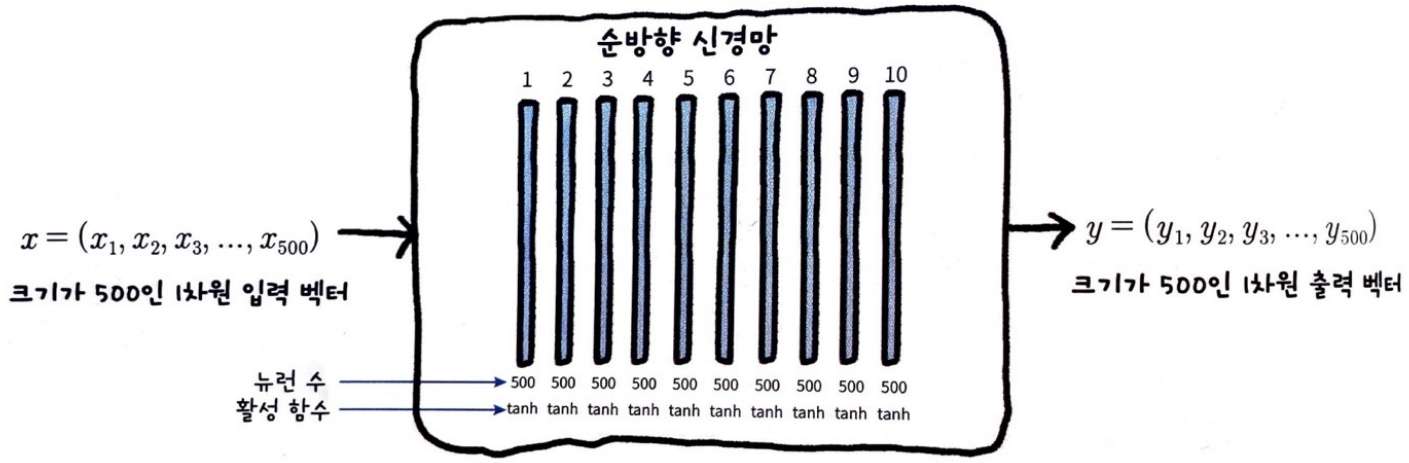


그림 5-4 10계층 순방향 신경망 모델

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

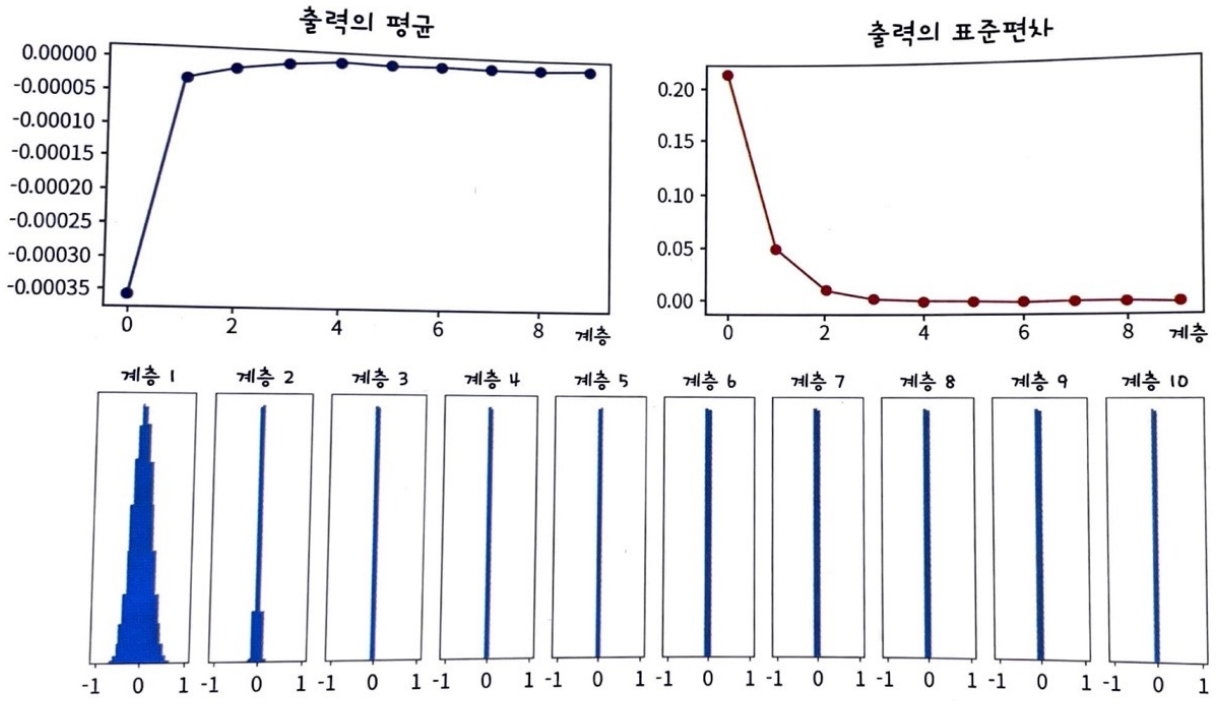


그림 5-5 가중치를 아주 작은 난수로 초기화한 경우

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

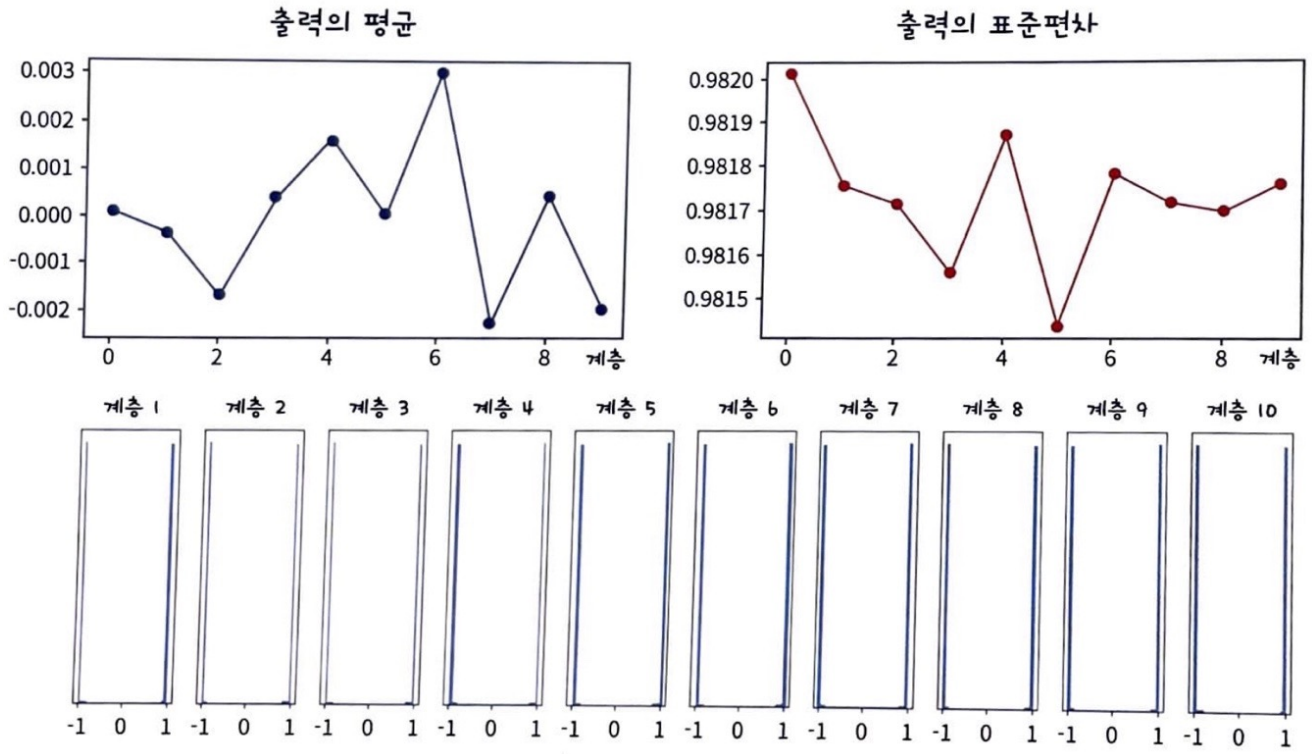


그림 5-6 가중치를 큰 난수로 초기화한 경우

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

$$y = z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

- 활성화 함수를 선형 함수로 가정한다.
  즉, 활성화 함수의 영향을 배제한다.
- 입력과 가중치는
  독립인 정규분포를 따른다.
- 목표
  입력과 출력의 분포가 같아지도록 하는
  가중치의 분포를 찾아라!!!

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

$$
\begin{aligned}
Var(y) &= \sum_{i=1}^{n} E\left((W_i x_i)^2\right) - E(W_i x_i)^2 \\
&= \sum_{i=1}^{n} E(W_i^2) E(x_i^2) - E(W_i)^2 E(x_i)^2 \\
&= \sum_{i=1}^{n} \left(Var(W_i) + E(W_i)^2\right)\left(Var(x_i) + E(x_i)^2\right) - E(W_i)^2 E(x_i)^2 \\
&= \sum_{i=1}^{n} E(W_i)^2 Var(x_i) + E(x_i)^2 Var(W_i) + Var(x_i) Var(W_i) \\
&= \sum_{i=1}^{n} Var(x_i)\, Var(W_i)
\end{aligned}
$$

$$\left(Var(W_i) = E(W_i^2) - E(W_i)^2 \rightarrow E(W_i^2) = Var(W_i) + E(W_i)^2\right)$$

$$Var(y) = nVar(x_i)Var(w_i)$$

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

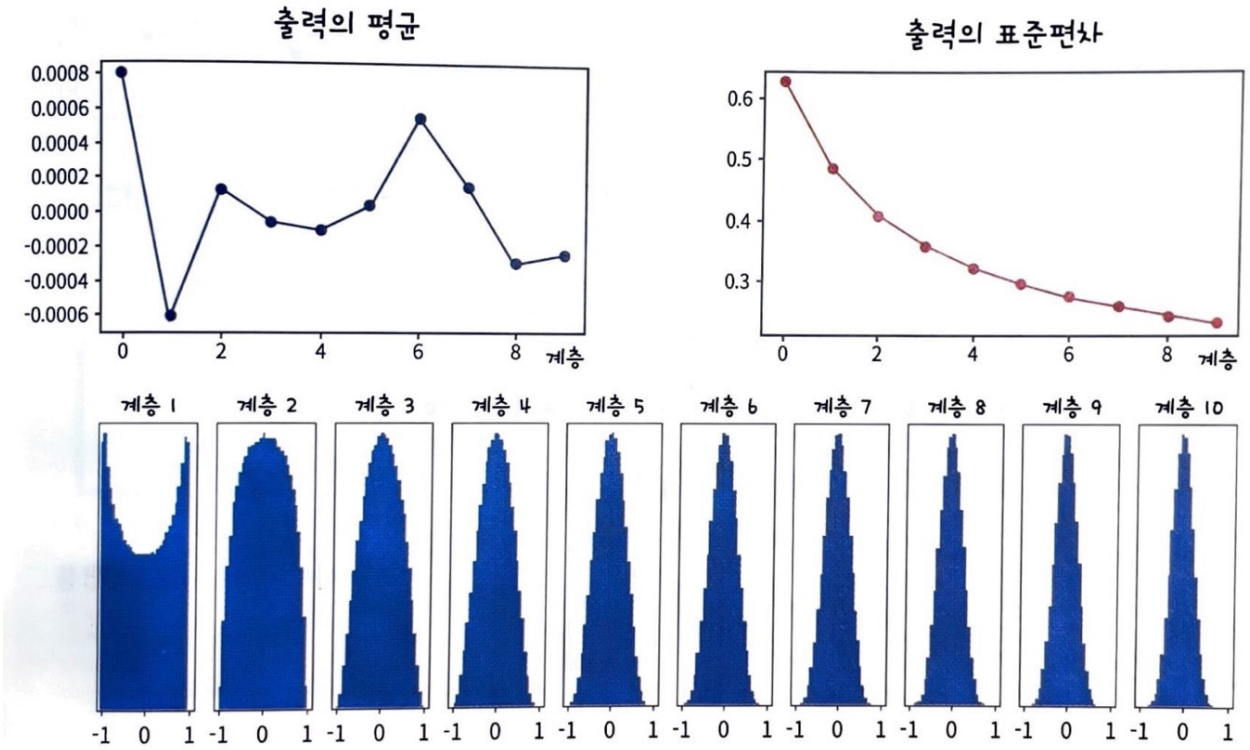


그림 5-9 하이퍼볼릭 탄젠트 사용 시 Xavier 초기화

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

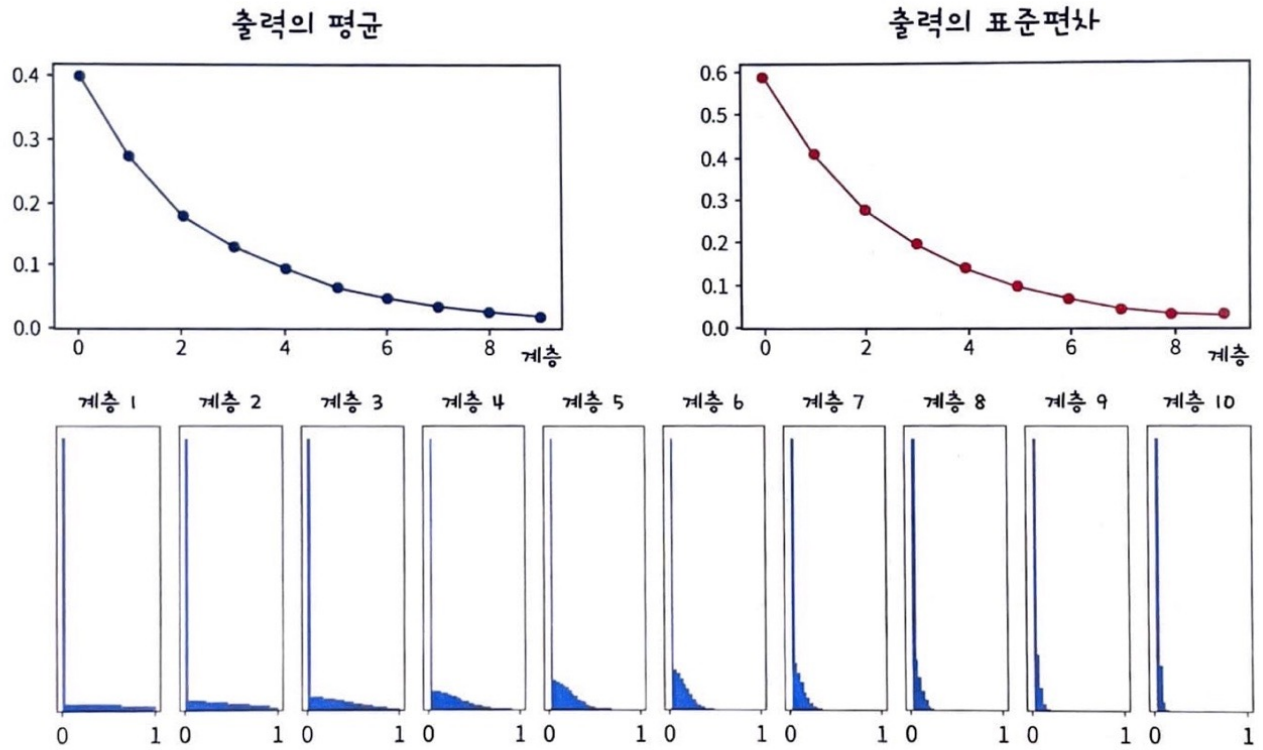


그림 5-11 ReLU 사용 시 Xavier 초기화

Methods for Efficiency : Time Efficient

- Weight Initialization
  - Random Initialization
  - Xavier Initialization
  - He Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

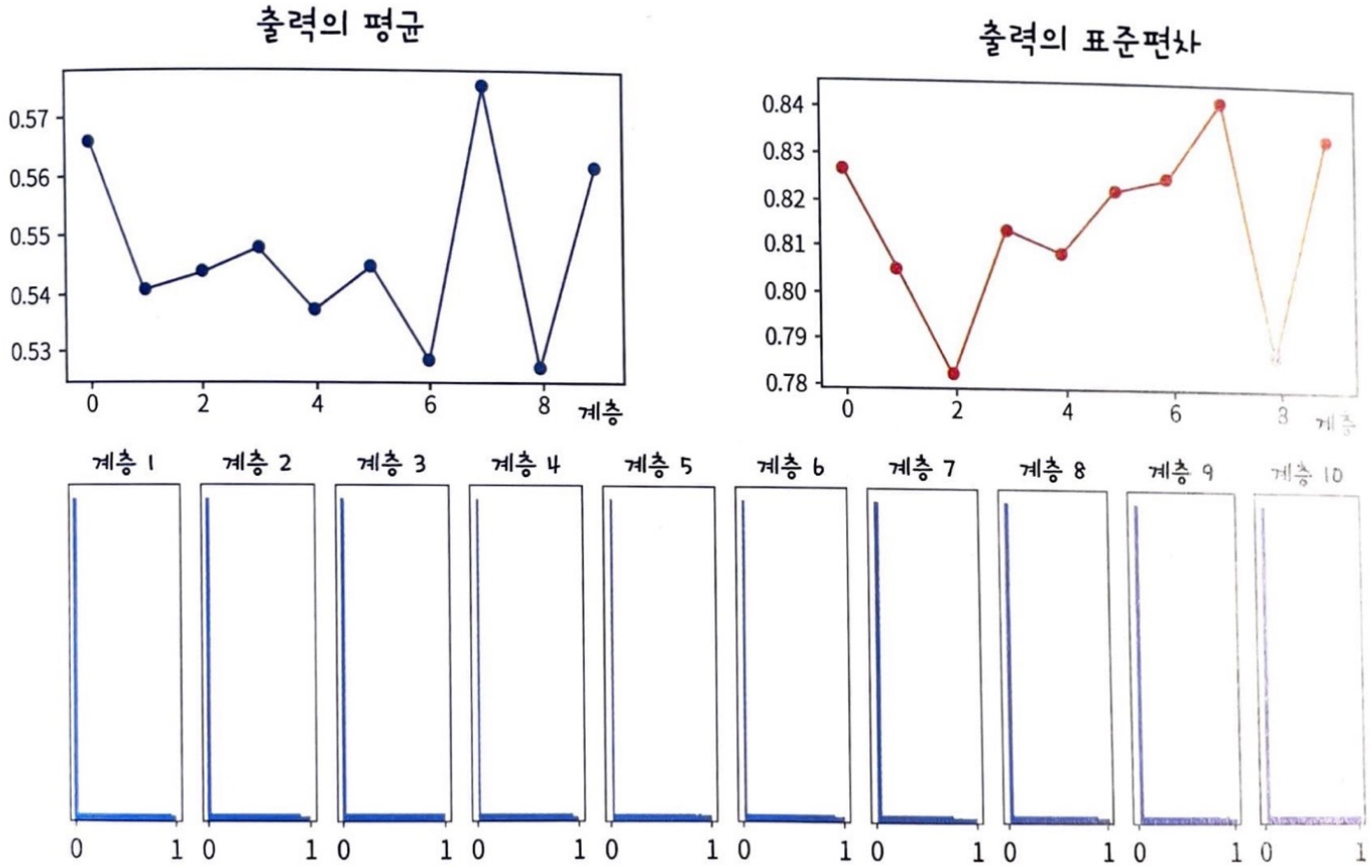
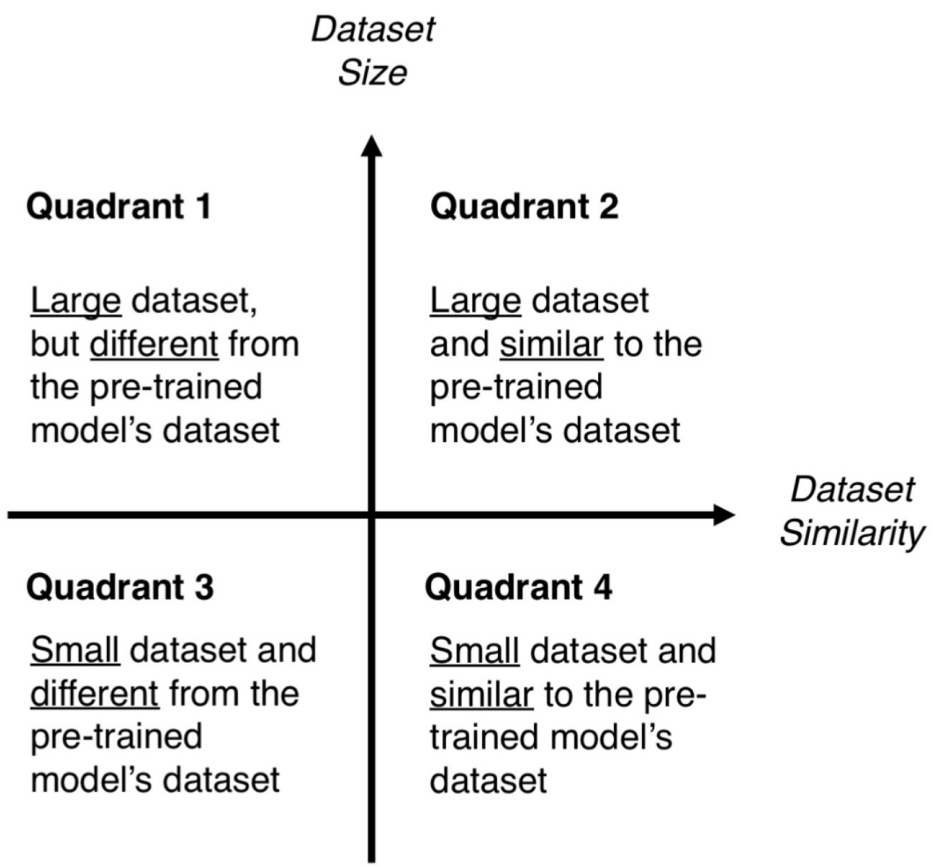


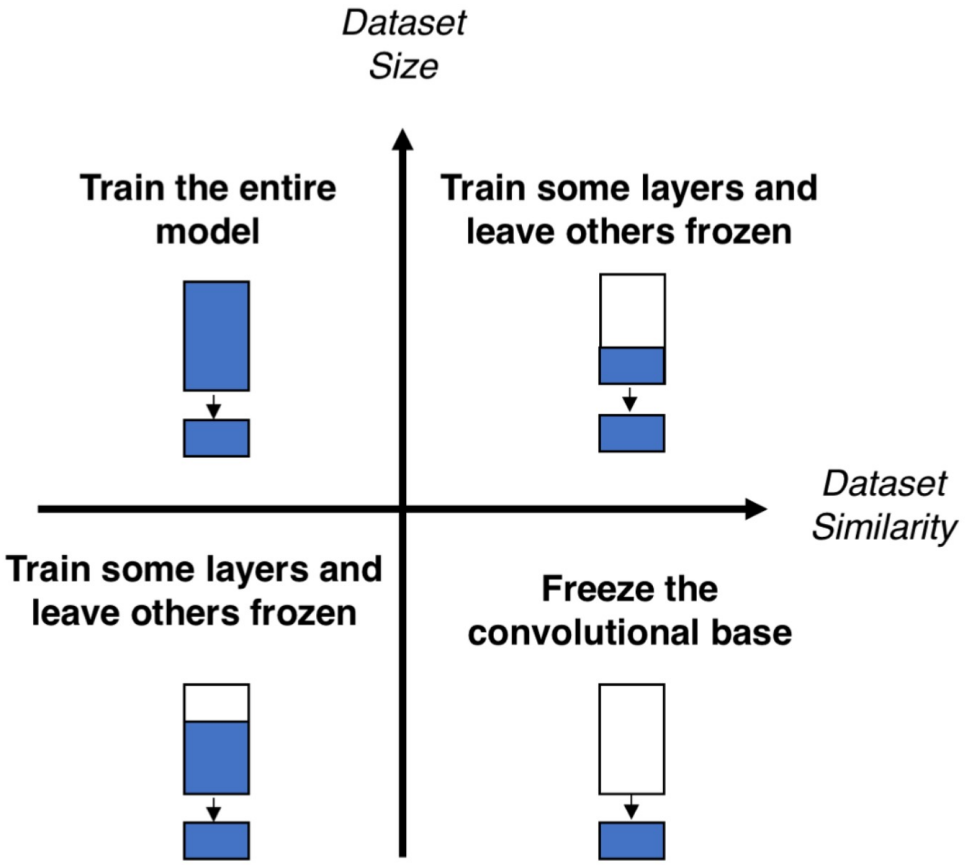
그림 5-13 ReLU 사용 시 He 초기화

Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

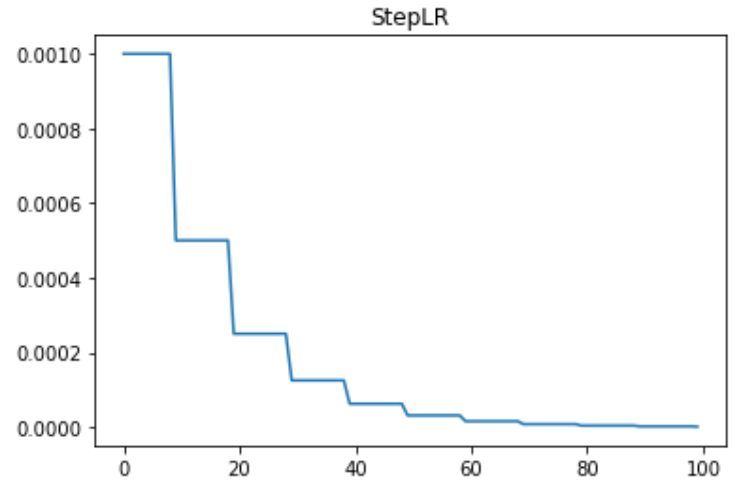
Methods for Efficiency : Time Efficient

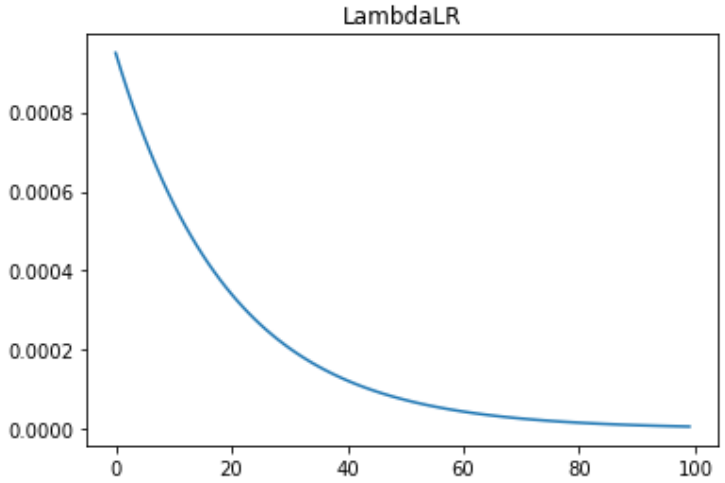


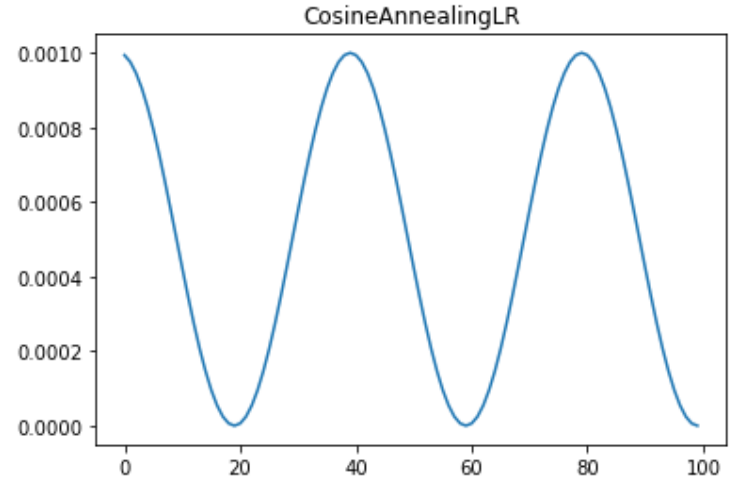


Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- **Learning Rate Schedules**
- Optimization Algorithms



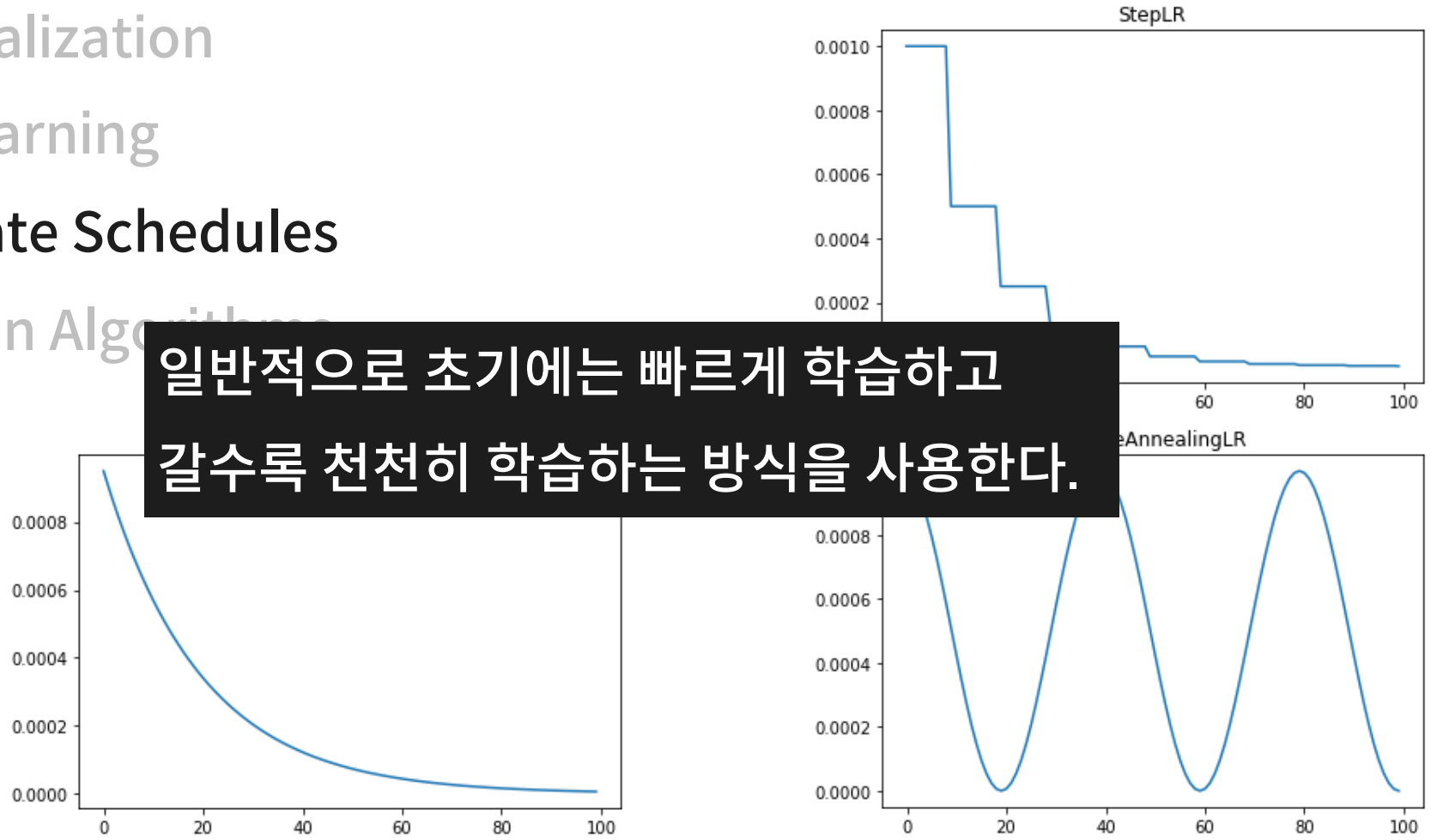




Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms

일반적으로 초기에는 빠르게 학습하고
갈수록 천천히 학습하는 방식을 사용한다.



Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms: SGD

$$W = W - \eta \frac{\partial L}{\partial W}$$

Pros

- Gradient가 0인 곳에서 탈출 가능
- 빠른 학습

Cons

- Local Minima
- Noisy Gradient
- Slow but Consistent Gradient

Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms: Momentum

$$v = \alpha v - \eta \frac{\partial L}{\partial W}$$

$$W = W + v$$

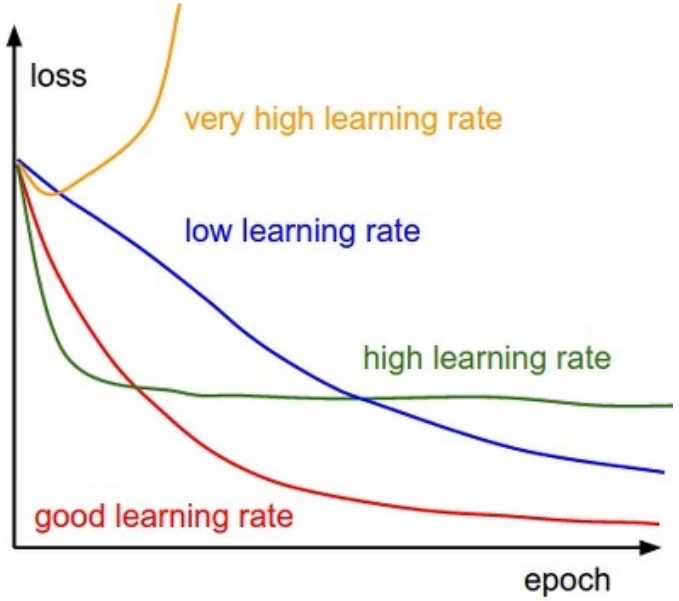
- 가속도 개념 활용
- SGD의 문제점 해결
  - Local Minima
  - Noisy Gradient
  - Slow but Consistent Gradient

Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms: RMSProp

AdaGrad	RMSprop
$\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$	$\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\sqrt{\delta + r}} \odot \boldsymbol{g}$	$\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\sqrt{\delta + r}} \odot \boldsymbol{g}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

- How to get Good Learning Rate?



- 처음에는 빠르게, 나중에는 느리게
- AdaGrad (Adaptive Gradient)

Methods for Efficiency : Time Efficient

- Weight Initialization
- Transfer Learning
- Learning Rate Schedules
- Optimization Algorithms:
  Adam = Momentum + RMSProp

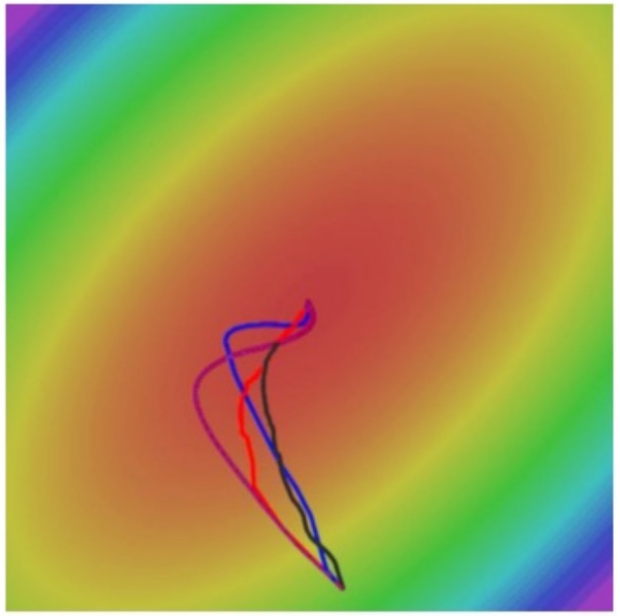
$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$$

$$\hat{m_t} = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v_t} = \frac{v_t}{1-\beta_2^t}$$

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v_t} + \epsilon}}\hat{m_t}$$



– sgd

– sgd + momentum

– RMSProp

– Adam

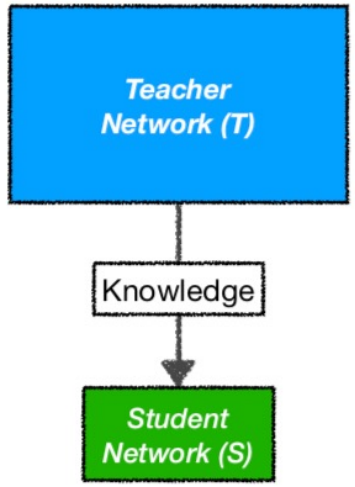
(source: cs231n)

Methods for Efficiency : Space Efficient

- Knowledge Distillation
- Self-supervised Learning
- Pseudo-Label
- Domain Adaptation
- Domain Generalization

Introduction

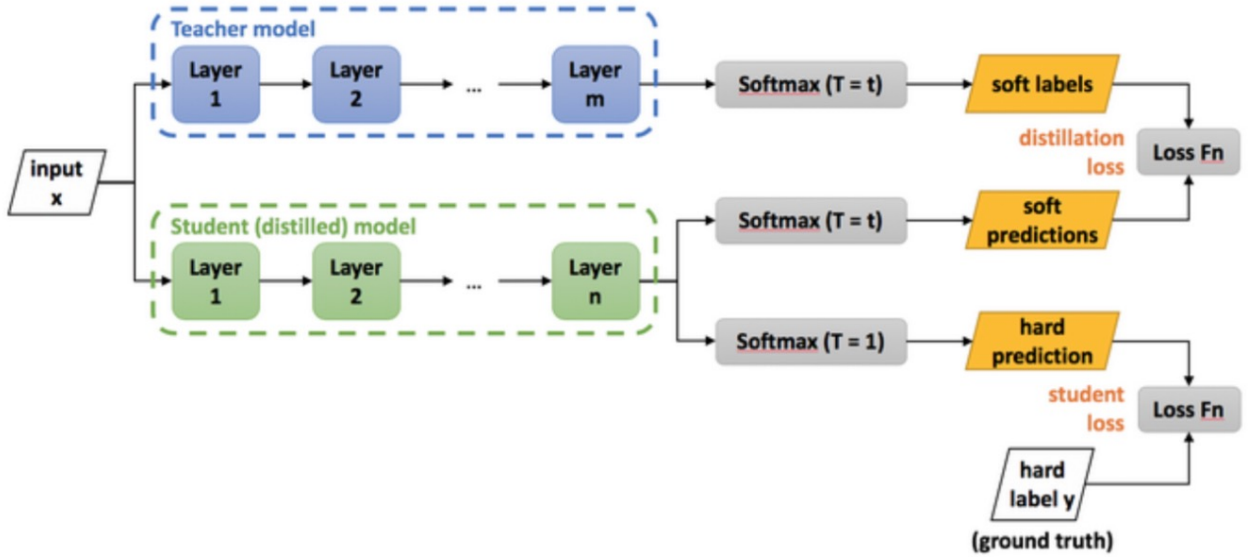
Knowledge Distillation



1. **Teacher Network (T)**
   - cumbersome model
   ex) ensemble / a large generalized model
   - (pros) excellent performance
   - (cons) computationally expansive
   - can not be deployed when limited environments

2. **Student Network (S)**
   - small model
   - suitable for deployment
   - (pros) fast inference
   - (cons) lower performance than T

Methods for Efficiency : Space Efficient



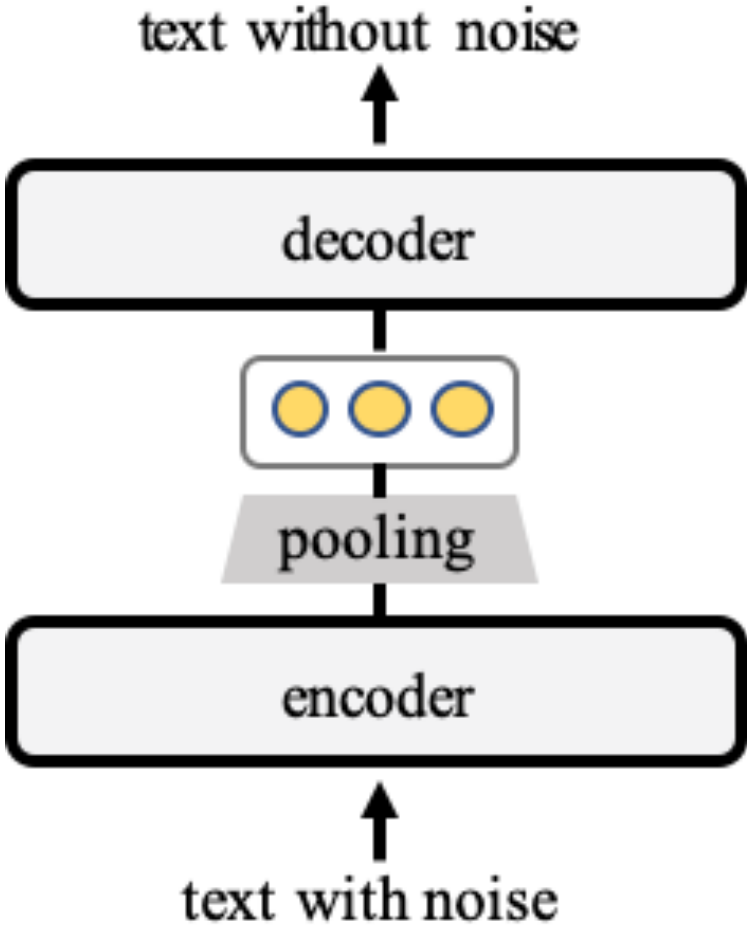
$$L = \sum_{(x,y)\in\mathbb{D}} L_{KD}(S(x,\theta_S,\tau), T(x,\theta_T,\tau)) + \lambda L_{CE}(\hat{y}_S, y)$$

- 확률은
  0과 1로 이루어져 있지 않다.

Methods for Efficiency : Space Efficient

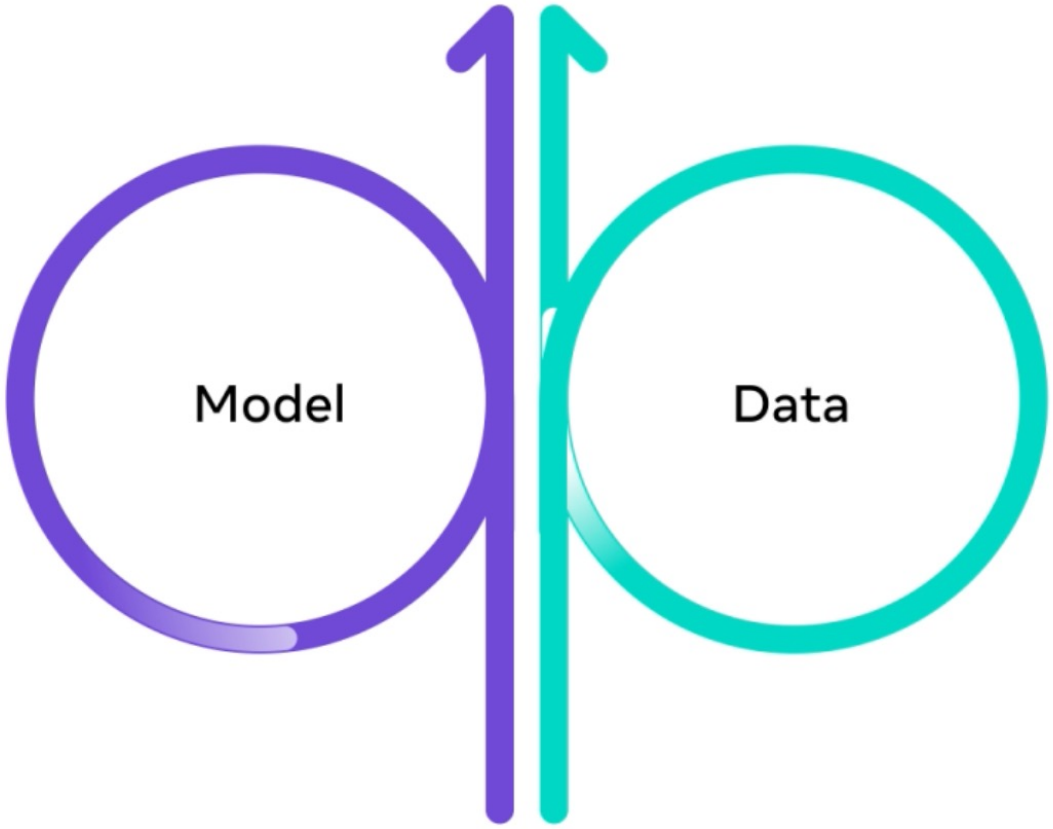
- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label
- Domain Adaptation
- Domain Generalization

- NeRF
- ChatGPT
- BERT
- TSDAE



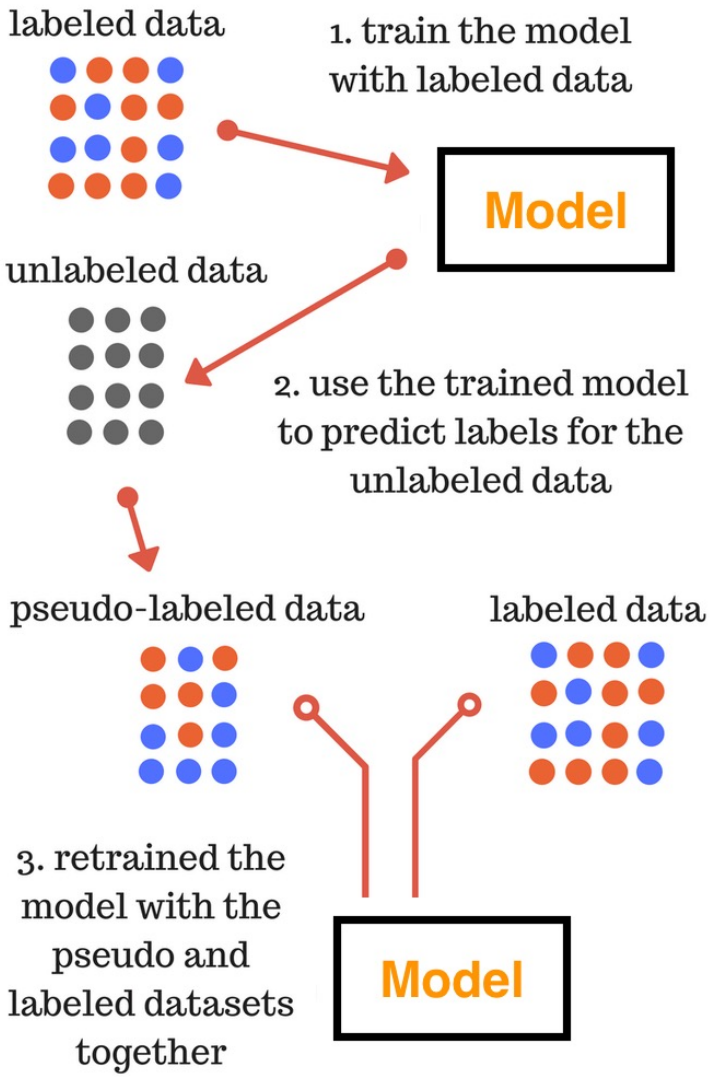
Methods for Efficiency : Space Efficient

- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label: SAM
- Domain Adaptation
- Domain Generalization



Methods for Efficiency : Space Efficient

- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label
- Domain Adaptation
- Domain Generalization



Methods for Efficiency : Space Efficient

- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label: Noisy Student
- Domain Adaptation
- Domain Generalization

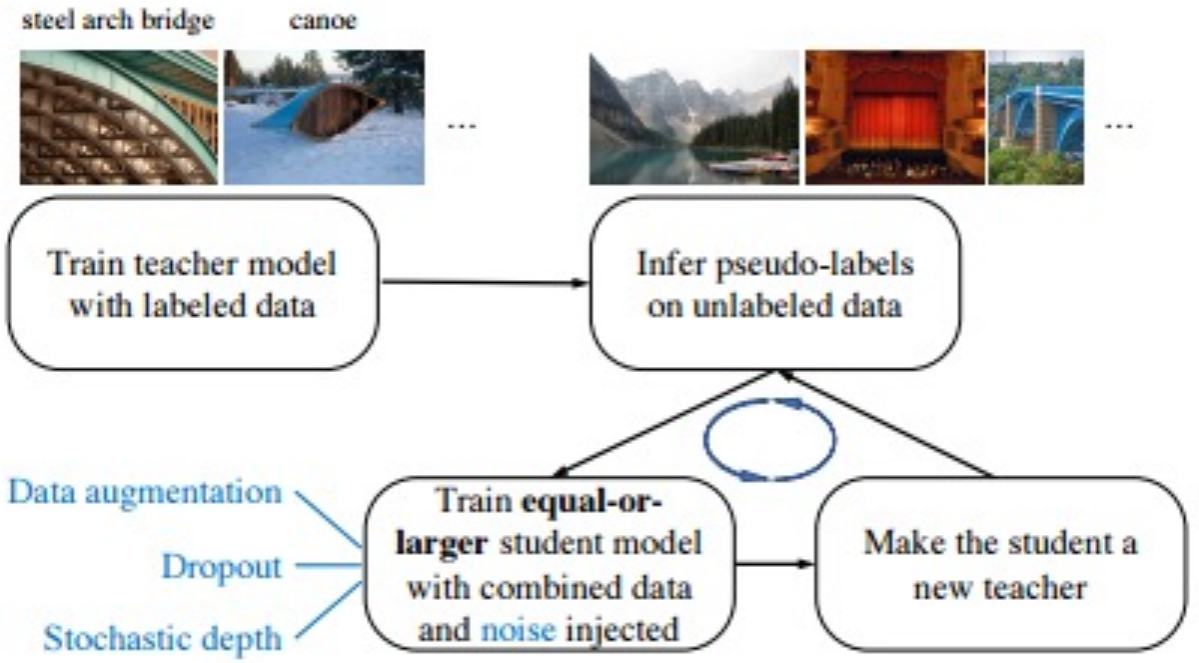


Figure 1: Illustration of the Noisy Student Training. (All shown images are from ImageNet.)

Methods for Efficiency : Space Efficient

- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label
- Domain Adaptation
- Domain Generalization

- Setting
  - 학습 데이터와 테스트 데이터가 서로 다르다.
  - 학습 데이터는 레이블이 있지만
  - 테스트 데이터는 레이블이 없다.

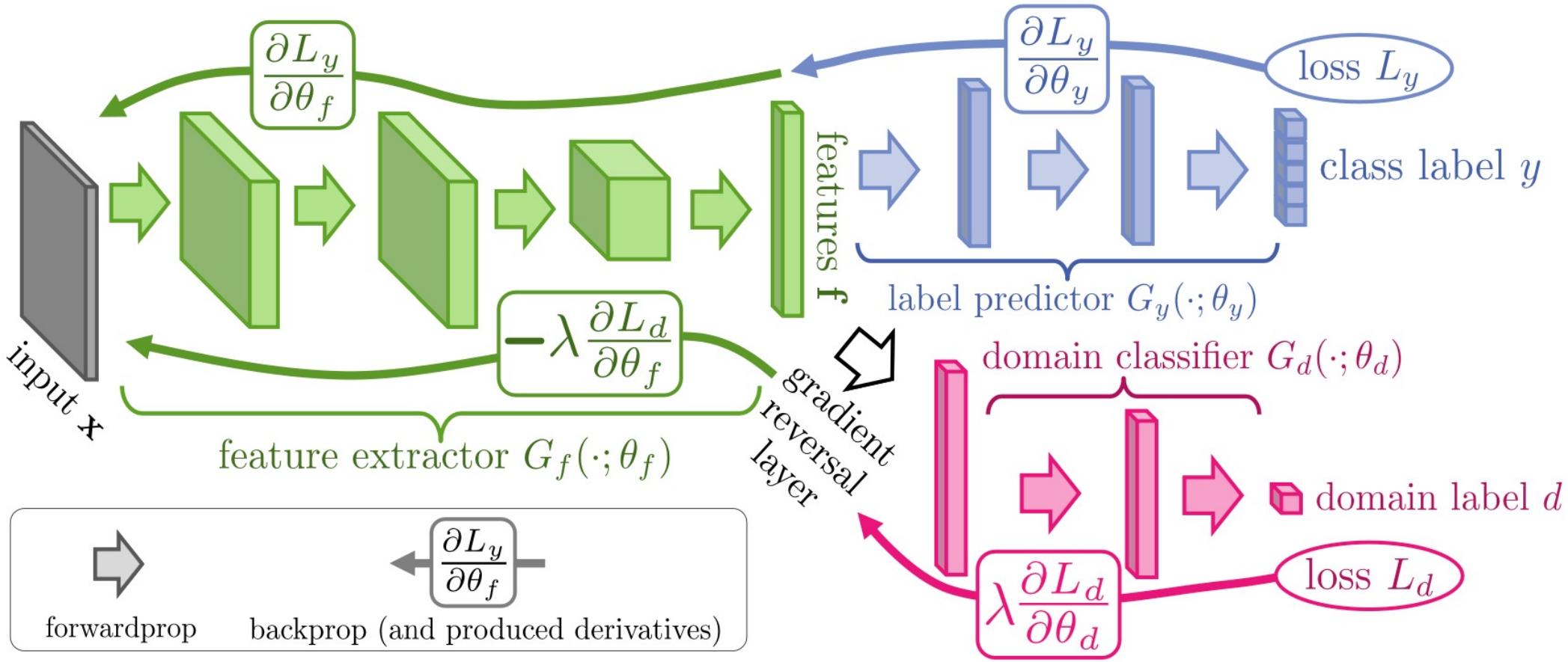
- Robust to distribution change

Methods for Efficiency : Space Efficient

Feature Extractor가 생성하는 Feature는 다음과 같아야 한다.

- Discriminativeness: Good for Classification
- Domain-invariance: Bad for Domain Classification

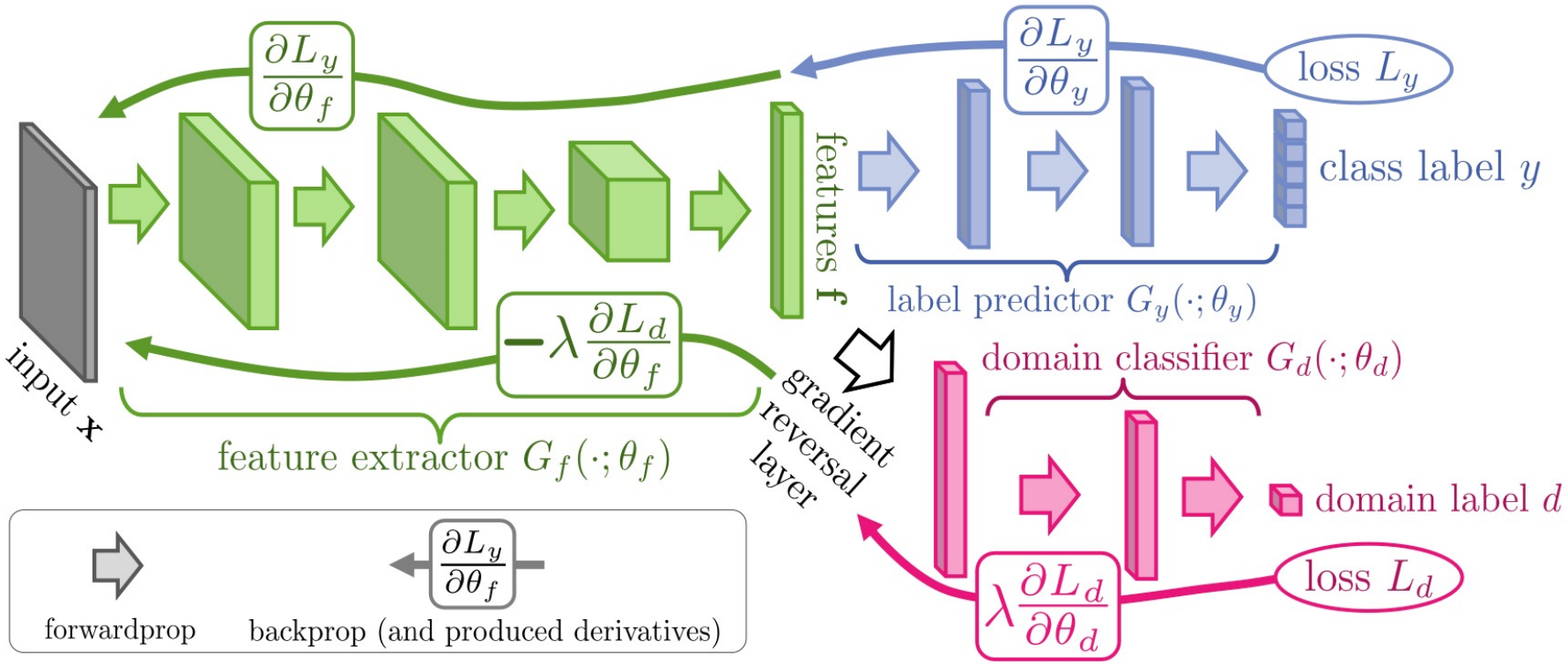
Methods for Efficiency : Space Efficient



Classification – Blue + Green

- Gradient Descent – Classfication Loss가 줄어들도록

Methods for Efficiency : Space Efficient



Domain Classification – Pink + Green

- Pink – Gradient Descent, Domain Classification Loss가 줄어들도록
- Green – Gradient Ascent, Domain Classification Loss가 증가하도록

Methods for Efficiency : Space Efficient

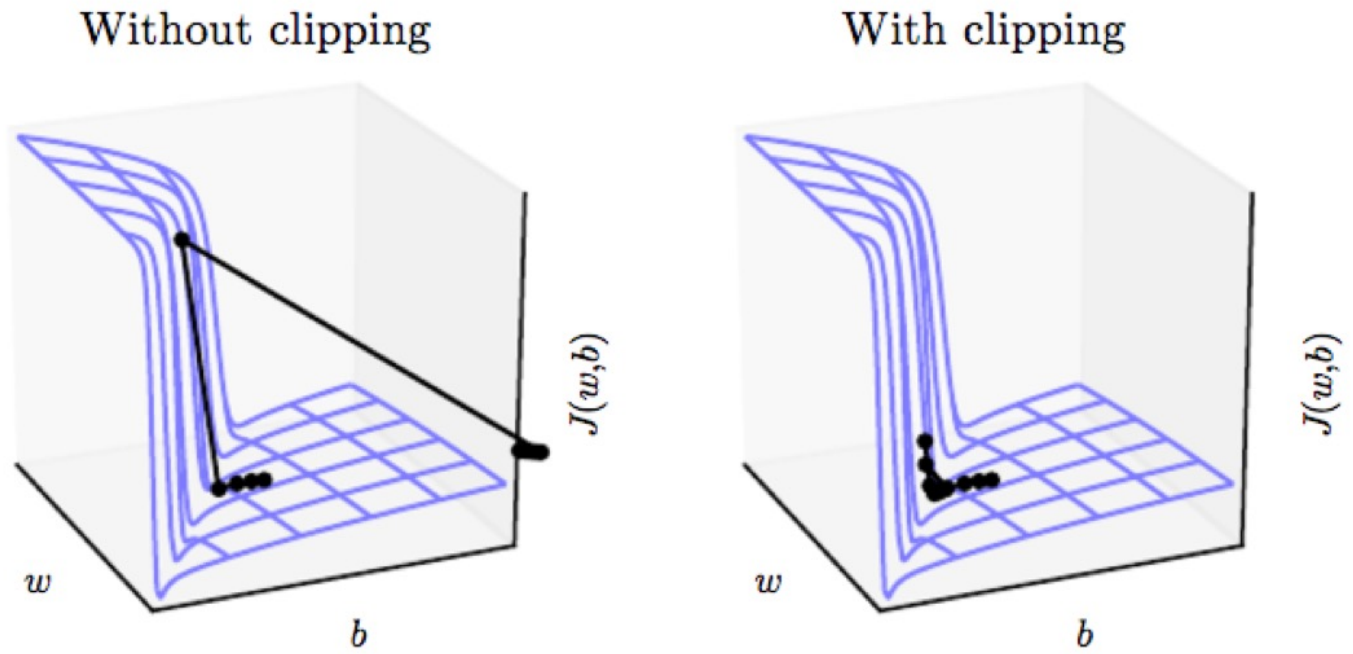
- Knowledge Distillation
- Self-supervised Learning
- Pseudo Label
- Domain Adaptation
- Domain Generalization

- Setting
  - Domain Adaptation과 달리
    테스트 데이터가 없다.
  - 도메인 A과 B 중
    A에서만 데이터를 얻을 수 있는 상황에서
    A에서 얻은 데이터를 통해
    B에서도 좋은 성능을 얻을 수 있어야 한다.
  - Domain-agnostic representation

Methods for Efficiency : Stable Learning

- Gradient Clipping
- Normalization
- Residual Connection

- Gradient Vanishing
- Gradient Exploding



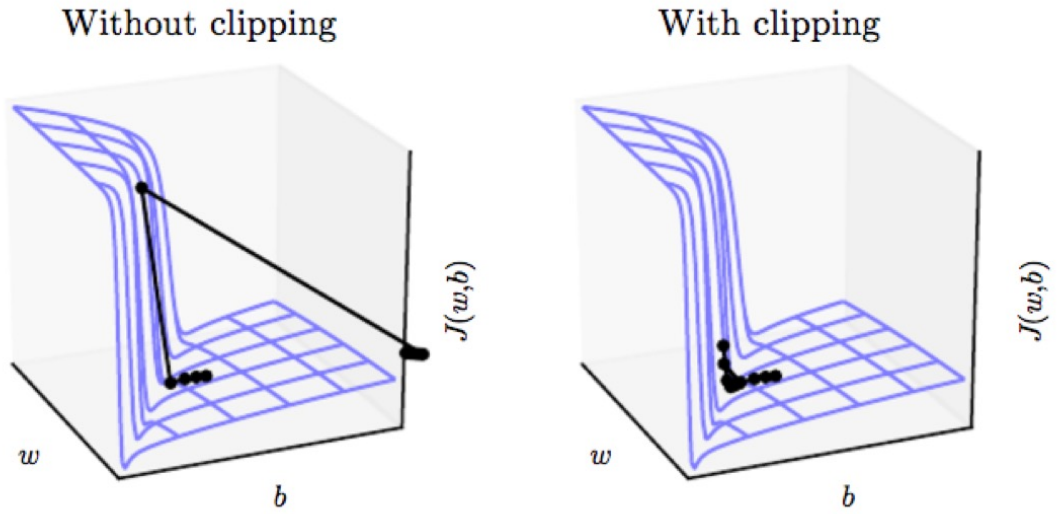
Methods for Efficiency : Stable Learning

- Gradient Clipping
- Normalization
- Residual Connection

- Gradient Vanishing
- Gradient Exploding

$$\frac{\partial \epsilon}{\partial \theta} \leftarrow \begin{cases} \frac{\text{threshold}}{\|\hat{g}\|}\hat{g} & \text{if } \|\hat{g}\| \geq \text{threshold} \\ \hat{g} & \text{otherwise} \end{cases}$$

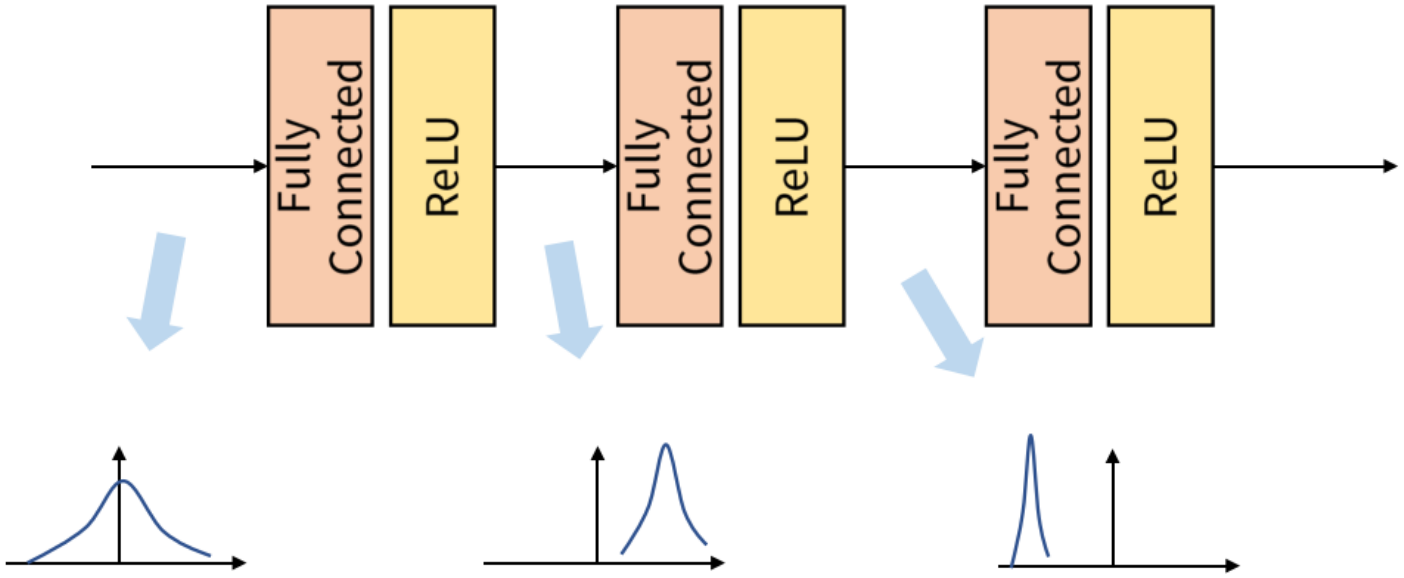
$$\text{where } \hat{g} = \frac{\partial \epsilon}{\partial \theta}.$$



Methods for Efficiency : Stable Learning

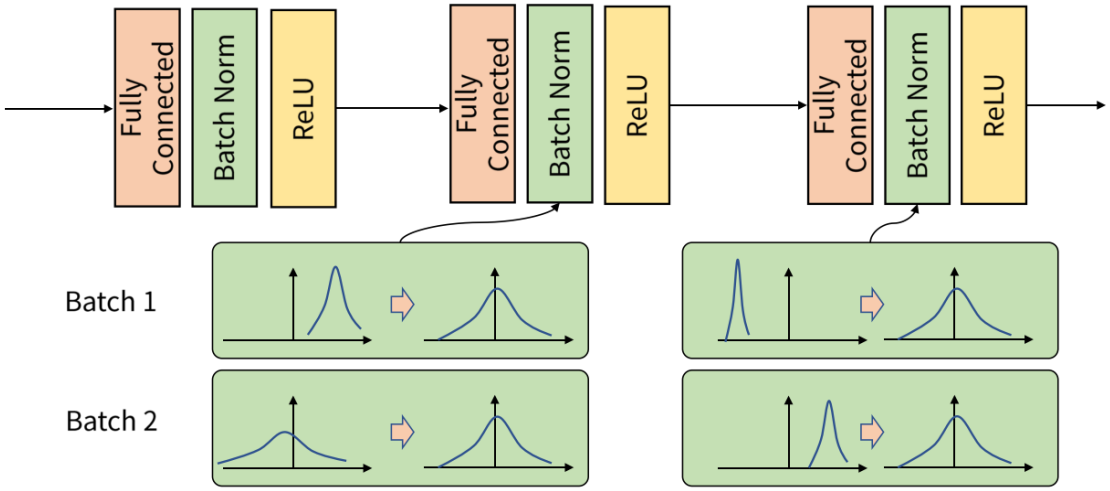
- Gradient Clipping
- Normalization
- Residual Connection

- 학습이 불안정한 이유는
  Internal Covariance Shift



Methods for Efficiency : Stable Learning

- Gradient Clipping
- Normalization
- Residual Connection

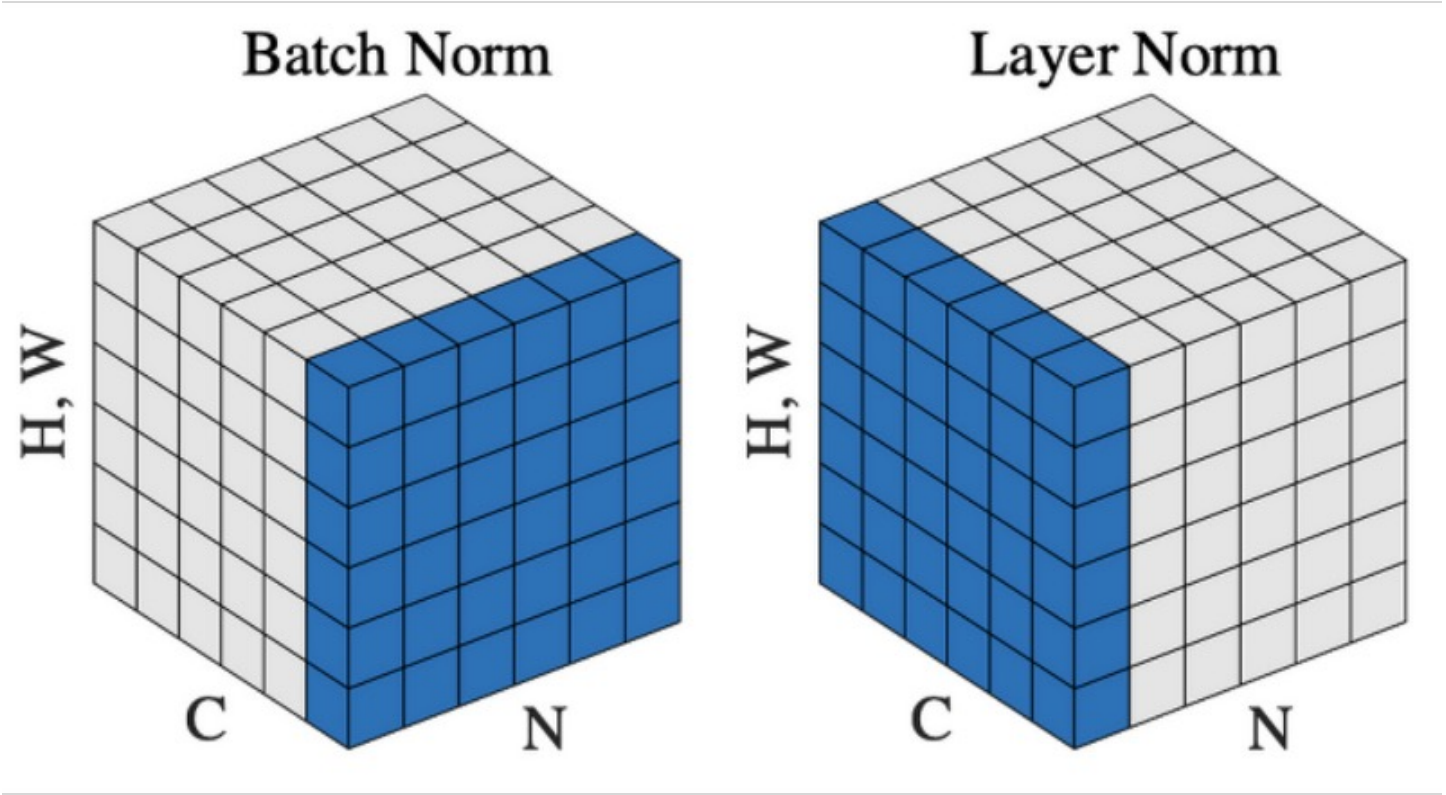


- Whitening과 달리
  평균과 분산 또한 변화한다.

- 테스트 시, 학습 과정에서 구한
  평균과 분산을 활용한다.

$$Z = \frac{X - m}{\sigma}$$

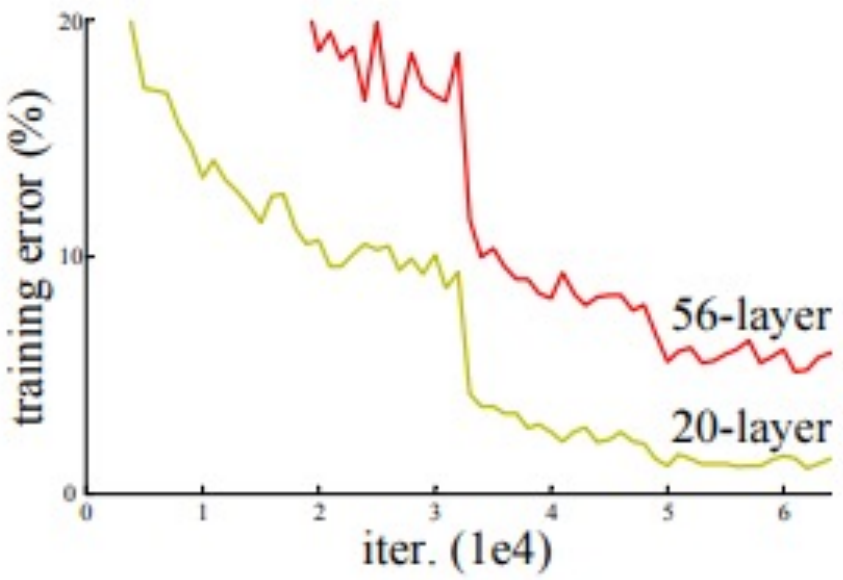
Methods for Efficiency : Stable Learning

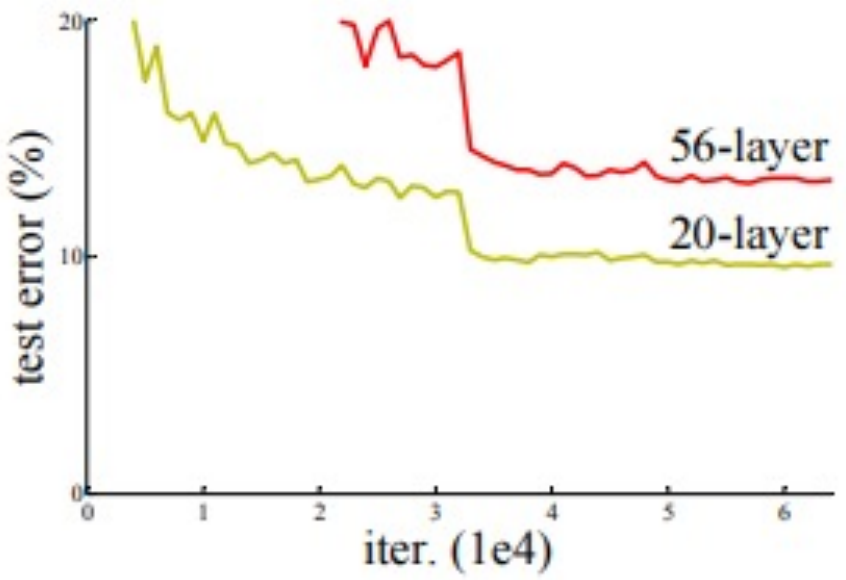


Methods for Efficiency : Stable Learning

- Gradient Clipping
- Normalization
- Residual Connection

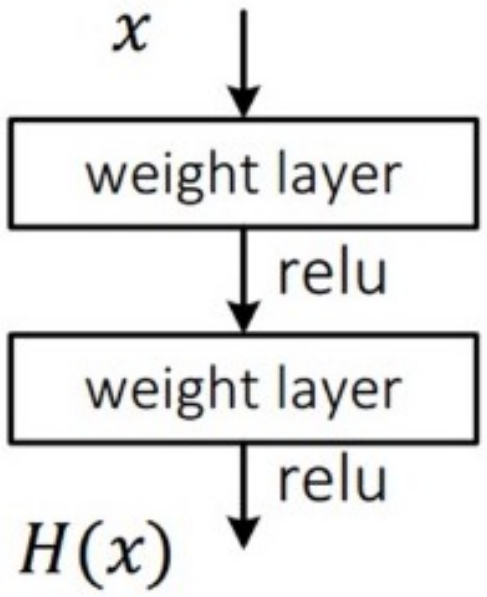
- Gradient Vanishing

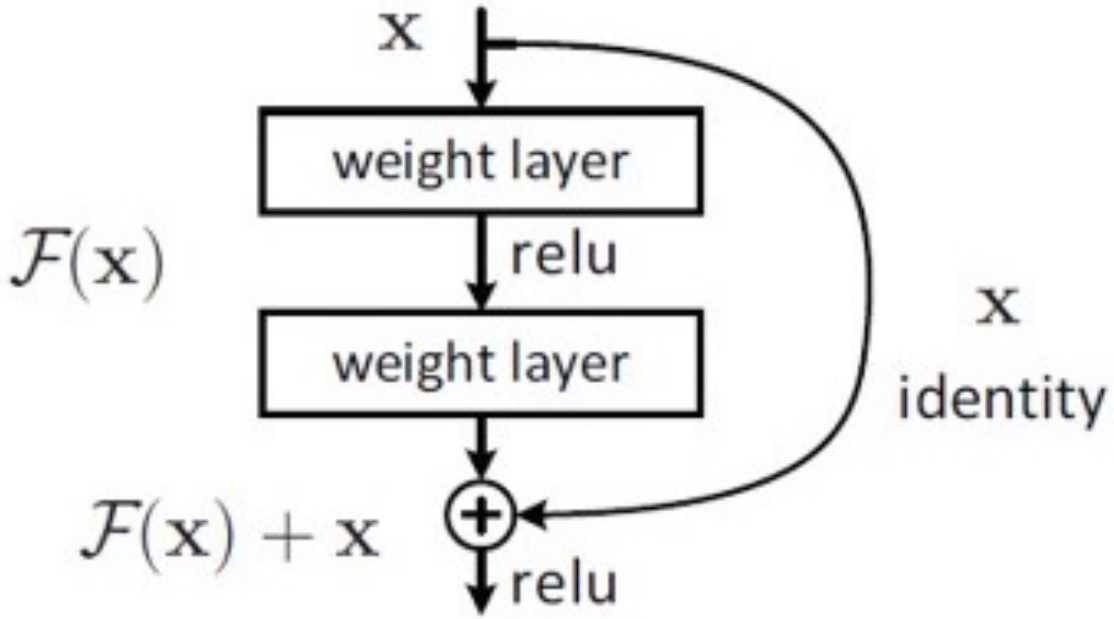




Methods for Efficiency : Stable Learning

- Gradient Clipping
- Normalization
- Residual Connection





Methods for Efficiency : Hyperparameter Tuning

- What is Tuning?
  - https://github.com/google-research/tuning_playbook
  - https://scikit-learn.org/stable/modules/grid_search.html#
  - https://medium.com/curg/%EB%94%A5%EB%9F%AC%EB%8B%9D-%EC%84%B1%EB%8A%A5%EC%9D%84-%EB%86%92%EC%9D%B4%EA%B8%B0-%EC%9C%84%ED%95%9C-%EB%8B%A4%EC%96%91%ED%95%9C-%EA%BF%80%ED%8C%81%EB%93%A4-1910c6c7094a
- AutoML
  - CASH (Combined Algorithm Selection and Hyperparameter optimization)
  - NAS (Neural Architecture Search)

Let’s Discuss!!! 🤔

- Model Complexity가 큰 모델의 Variance와
  2가지 이상의 방법 간의 관계에 대해 설명해보세요. (아래 예시는 제외)
- Ex) Dropout 또는 DropConnect를 사용하면, Ensemble과 유사한 효과가 나타나 모델의 Variance를 줄일 수 있다. 또한 Free riding이 줄어드는 효과 때문에 더 좋은 representation을 얻을 수 있다. Randomness가 증가하는 것이 모델의 한계를 주어 모델의 Variance를 줄일 수 있다.

🤗 감사합니다 🤗