

# 자연어처리팀

2팀

한준호

윤세인

김나현

윤여원

권능주

# INDEX

---

1. 가중치 초기화
2. Dropout
3. 정규화
4. 데이터 증강
5. 파인 튜닝
6. 임베딩

## 가중치 초기화의 중요성



## 좋은 학습의 조건



각 뉴런의 활성화 함수 출력 값이 고르게 분포되어 있어야 함



레이어 간에 다양한 데이터가 흐름



많은 특징을 학습함으로써  
효율적인 신경망 학습이 가능함

## 1

## 가중치 초기화

## 가중치 초기화의 중요성

초기 가중치 설정에 따라

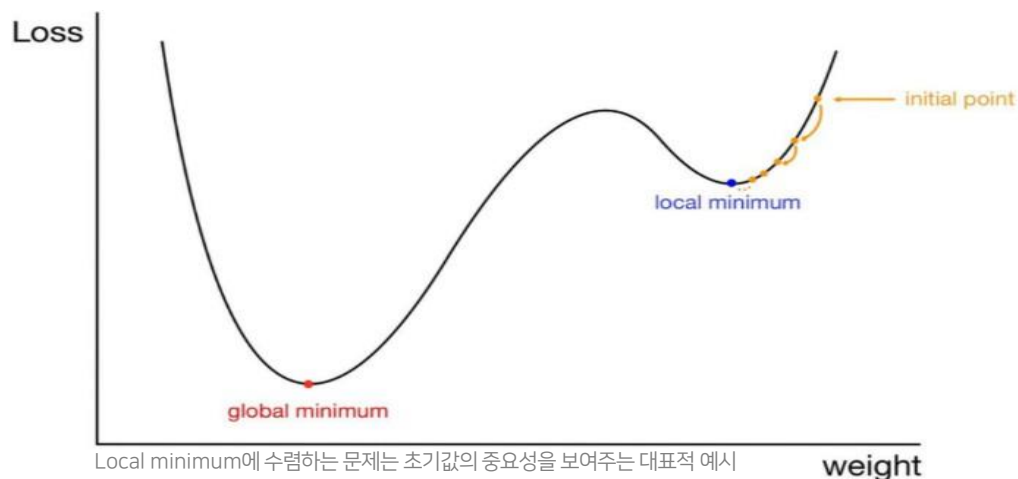
기울기 소실 문제 (Gradient 폭발 및 소멸 문제)

혹은 표현력에 한계를 갖는 여러 가지 문제 발생



즉, 같은 모델을 훈련시키더라도

가중치가 어떤 초기 값을 갖느냐에 따라 훈련 결과가 달라짐



## 가중치 초기화의 종류

### Zero Initialization

모든 가중치를 0으로 초기화시키는 것

### Random Initialization

가중치를 정규분포 혹은 균일분포를 따르는 랜덤으로 초기화

### Xavier Initialization

가중치를 정규분포 혹은 균일분포로부터 랜덤으로 초기화  
+ 각 레이어의 입력과 출력에 따른 유동적인 분산을 사용함

### He Initialization (by Kaiming He)

Xaiver Initialization에서  $\sqrt{2}$ 배를 한 방법

# 1

## 가중치 초기화

### Zero Initialization

#### Zero Initialization

모든 가중치를 0으로 초기화시키는 것



모든 파라미터의 값이 같으므로 역전파 시 동일하게 업데이트됨



즉, 딥러닝에서 좋은 가중치 초기화 전략이 아님

## 1

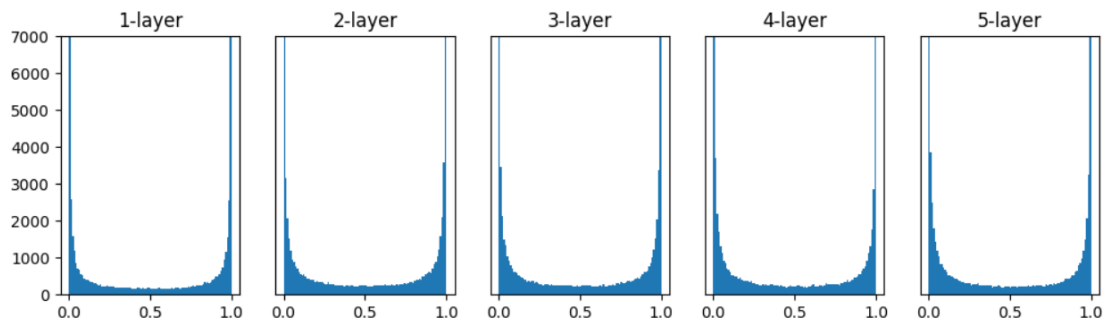
## 가중치 초기화

## Random Initialization

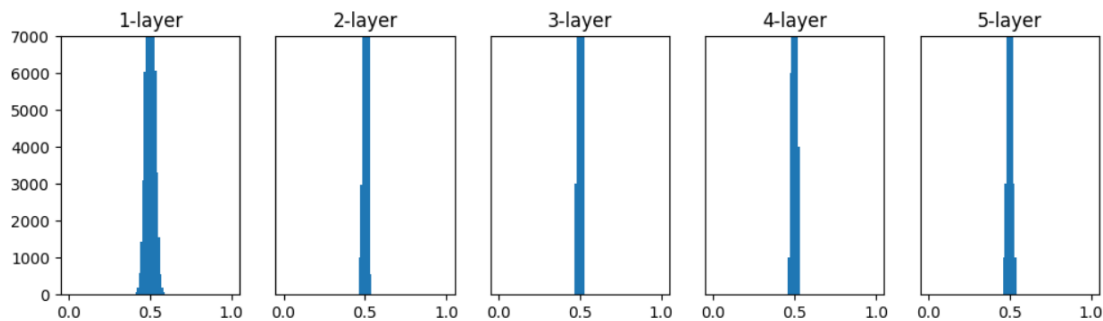
## Random Initialization

가중치를 정규분포 혹은 균일분포를 따르는 **랜덤으로** 초기화

$$X \sim N(0, 1)$$



$$X \sim N(0, 0.01^2)$$



## 1

## 가중치 초기화

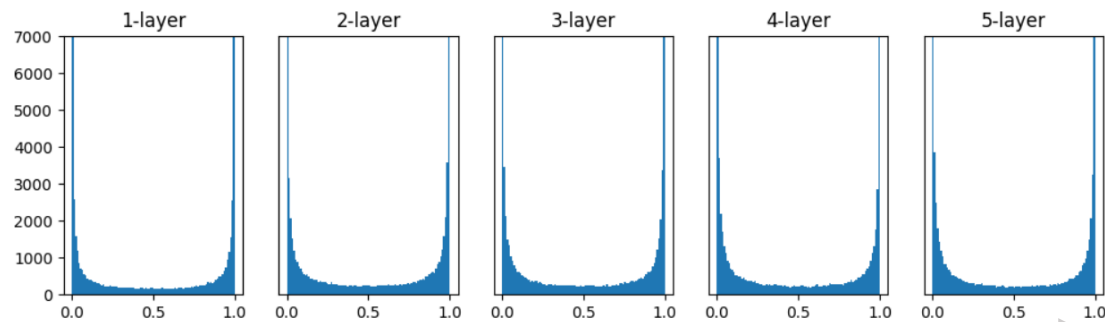
## Random Initialization

$$X \sim N(0, 1)$$

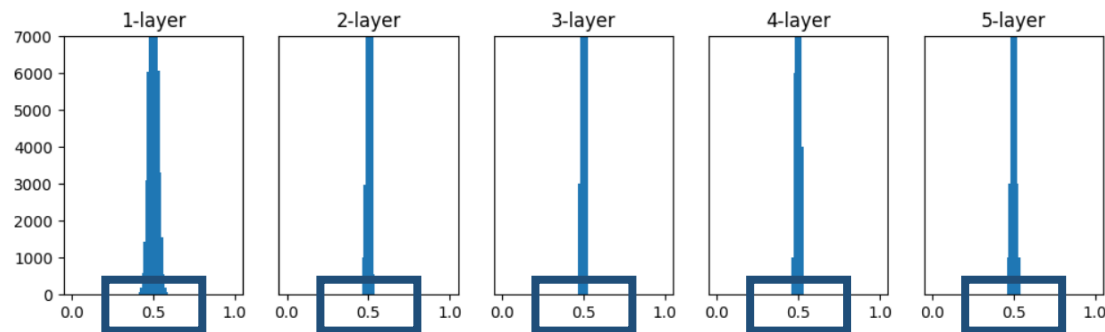


표준편차 감소

$$X \sim N(0, 0.01^2)$$



레이어를 통과할수록 그 출력 값이 0 또는 1에 치우침



출력 값이 0.5에 몰림

⋮

학습이 아예 안 되는 것은 아니지만, **심층 신경망의 깊이가 깊어질수록 학습이 어려움**



## Xavier Initialization

### Xavier Initialization

가중치를 정규분포 혹은 균일분포로부터 **랜덤으로 초기화**  
 + **각 레이어의 입력과 출력에 따른 유동적인 분산**을 사용함

#### Xavier Normal Initialization

$$X \sim N(0, \text{Var}(W)), \text{Var}(W) = \sqrt{\frac{2}{n+m}}$$

#### Xavier Uniform Initialization

$$X \sim U(-\sqrt{\frac{6}{n+m}}, \sqrt{\frac{6}{n+m}}), \text{Var}(W) = \sqrt{\frac{2}{n+m}}$$

레이어의 입력 노드의 개수

레이어의 출력 노드의 개수

# 1

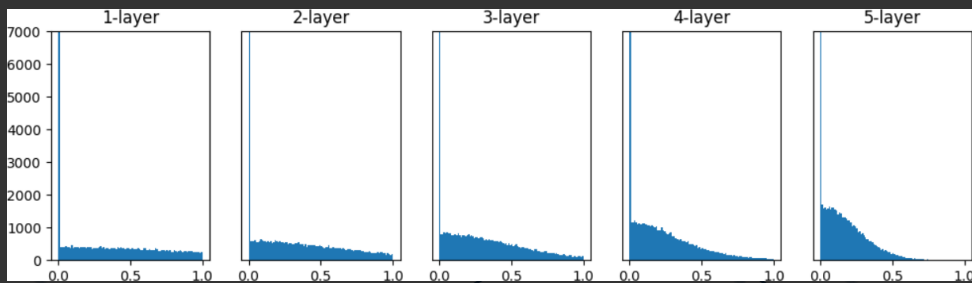
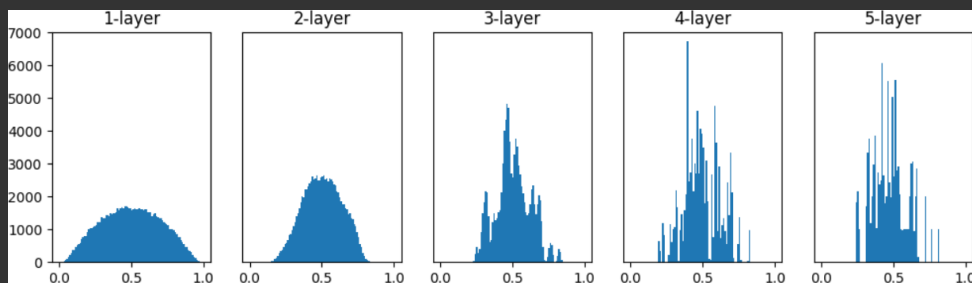
## 가중치 초기화

### Xavier Initialization



1. Xavier Initialization의 **취약점**

2. 레이어 간에 다양한 데이터가 흐름 → 많은 특징 학습 → 효율적인 신경망 학습



레이어의 **입력** 노드의 개수

레이어의 **출력** 노드의 개수

Sigmoid 함수에서는  
가중치가 고르게 분포

랜덤 초기화를 사용하는 분산을 사용함

ReLU 함수에서는  
레이어가 깊어질수록  
출력값이 0으로 치우침

Xavier Uniform Initialization

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

## He Initialization

He Initialization (by Kaiming He)

Xavier Initialization에서  $\sqrt{2}$ 배를 한 방법

WHY?

ReLU는 입력이 음수일 때 출력이 전부 0

→ 더 넓게 분포시키기 위해서

He Normal Initialization

$$X \sim N(0, \text{Var}(W)), \text{Var}(W) = \sqrt{2} \cdot \sqrt{\frac{2}{n+m}}$$

He Uniform Initialization

$$X \sim U\left(-\sqrt{2} \cdot \sqrt{\frac{6}{n+m}}, +\sqrt{2} \cdot \sqrt{\frac{6}{n+m}}\right)$$

# 2

Dropout

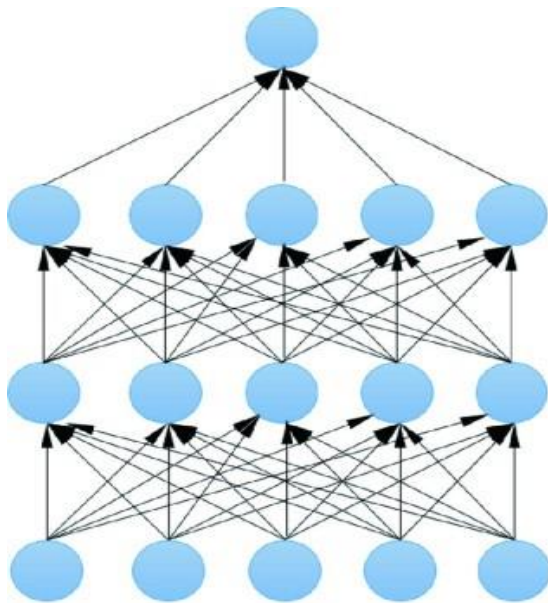
# 2

## Dropout

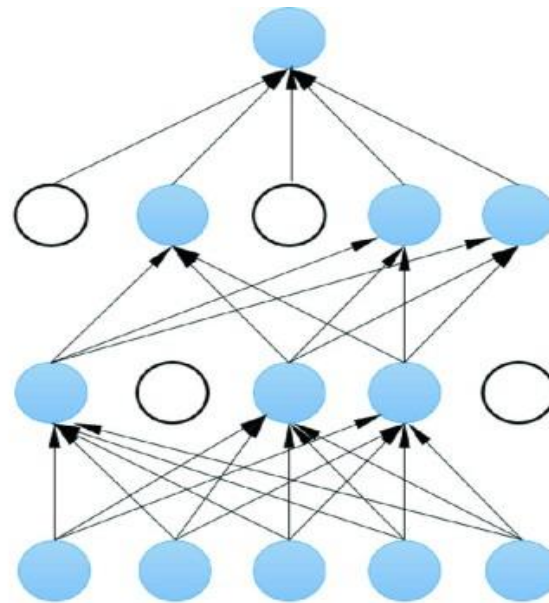
### Dropout

#### Dropout

서로 연결된 레이어에서 0부터 1사이의 확률로 뉴런을 제거하는 기법



(a) A standard deep neural network.

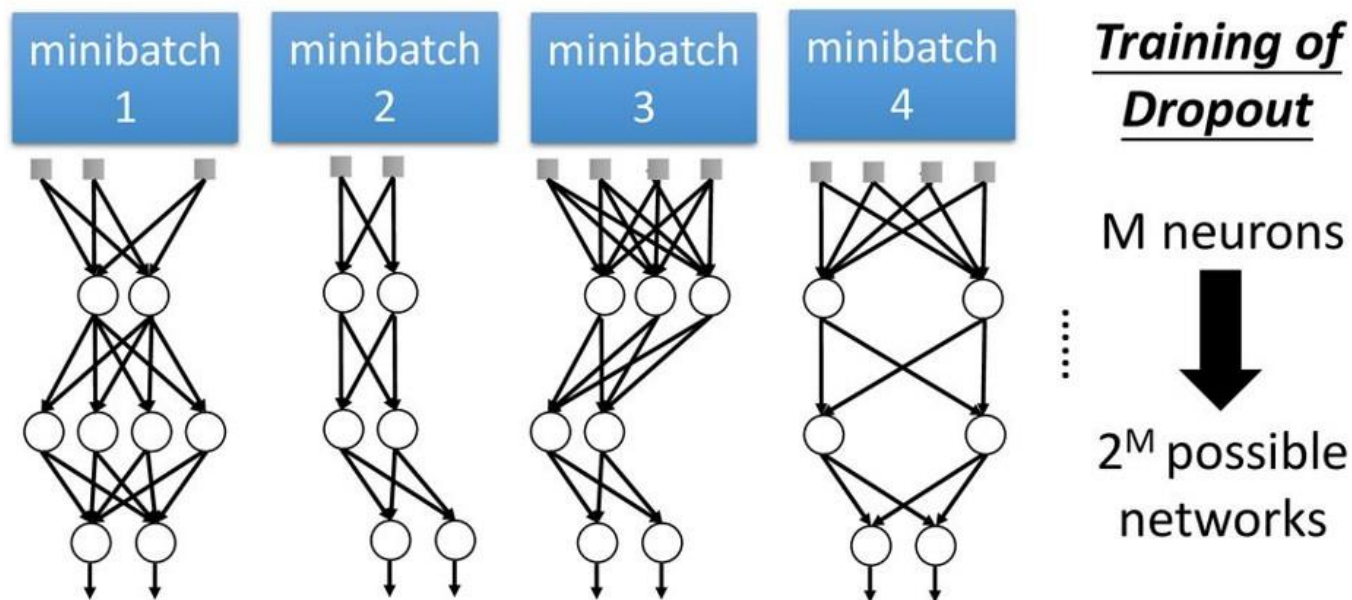


(b) After applying dropout.

# 2

## Dropout

### Dropout



랜덤으로 일정 비율 노드 제외하고 학습해 모델 앙상블 효과

⋮

여러 개의 네트워크 형성 효과로 특정 노드에 대한 지나친 집중 방지

과적합 방지에 도움이 됨

# 3

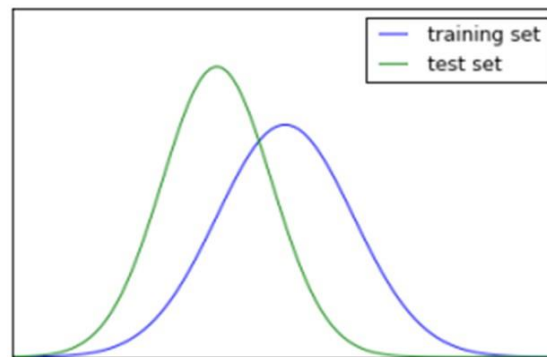
정규화

### 3 정규화 (Normalization)

#### 공변량 변화 (Covariance Shift)

##### 공변량 변화

입력 데이터의 분포가 학습할 때와 테스트할 때 다르게 나타나는 상황



(a) Covariate shift



학습 데이터를 이용해 만든 모델이 테스트 데이터를 가지고 추론할 때

성능에 영향을 미칠 수 있음

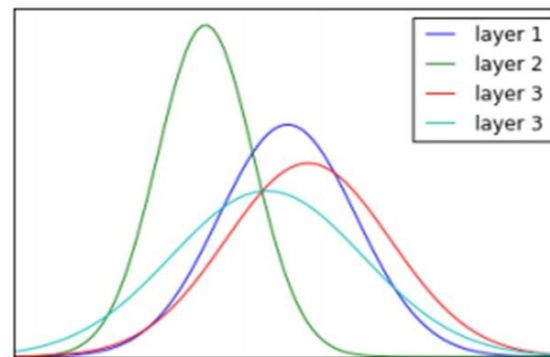


### 3 정규화 (Normalization)

#### 내부 공변량 변화 (Internal Covariate Shift, ICS)

##### 내부 공변량 변화

공변량 변화가 **신경망 내부에서** 일어나는 상황



(b) Internal covariate shift



신경망에서 층이 깊어질수록 심화되는  
**내부 공변량 변화 문제를 해결**하는 것은 중요한 문제!

## 정규화

정규화 (Normalization)

모든 데이터들의 스케일을 동일하게 만드는 작업



각 feature들이 동등한 중요도를 가질 수 있도록 함

## Normalization

Batch Normalization

Layer Normalization



정규화 방법으로는 배치 정규화와 레이어 정규화가 있음

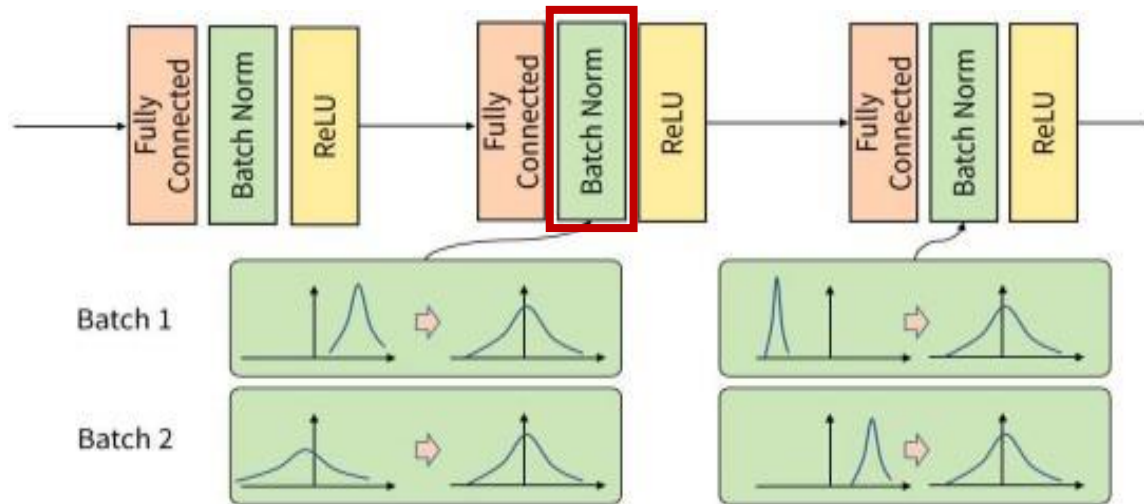
# 3 정규화 (Normalization)

## Batch Normalization

### 배치 정규화 (Batch Normalization)

인공 신경망의 각 층에 들어가는 입력(Batch) 단위로 **평균과 분산으로 정규화**

각 층에서 **활성화 함수를 통과하기 전에** 정규화



### 3 정규화 (Normalization)

#### Batch Normalization

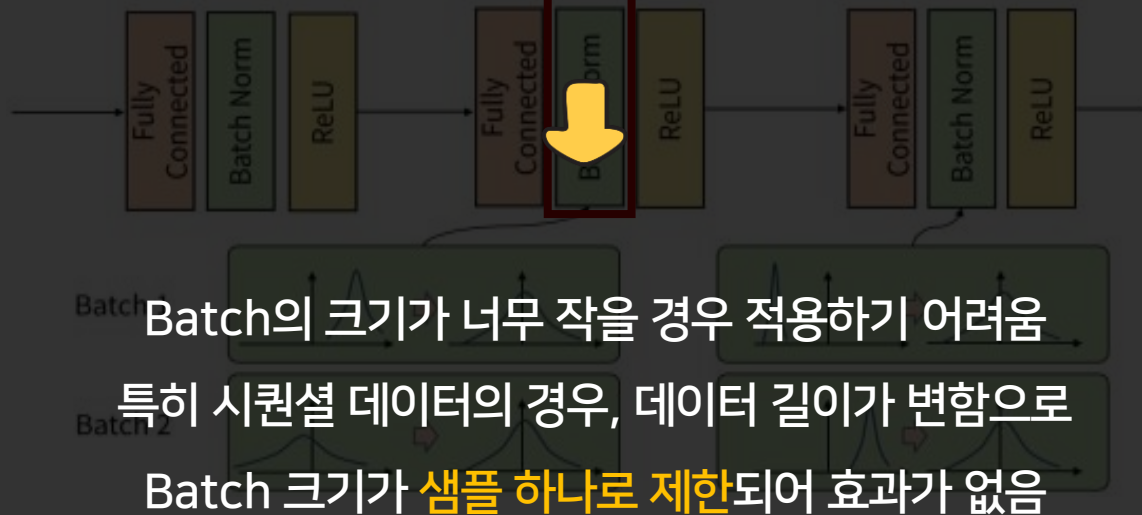


배치 정규화 (Batch Normalization)

## Batch Normalization의 **취약점**

인공 신경망의 각 층에 들어가는 입력(Batch)을 평균과 분산으로 정규화

학습 과정에서는 입력 데이터에 대한 평균과 분산을 구하기 쉽지만  
각 층에서 활성화 함수를 통과하기 전에 정규화  
**예측 과정에서는 어려움**

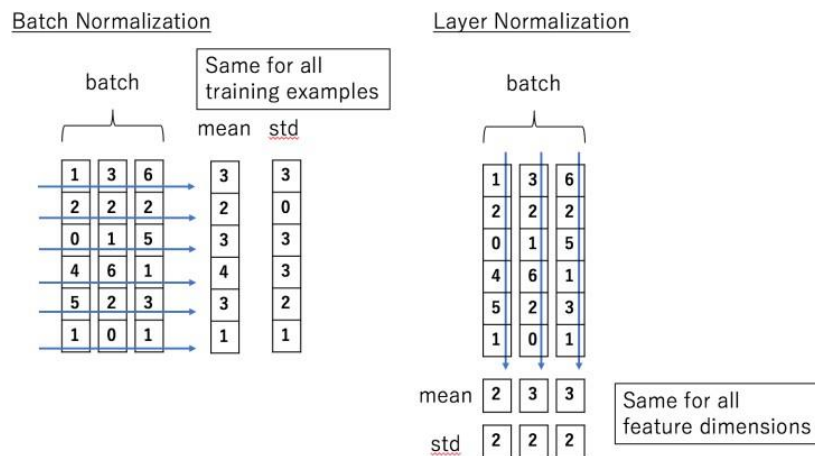


### 3 정규화 (Normalization)

## Layer Normalization

### 레이어 정규화 (Layer Normalization)

데이터 샘플 단위로 정규화를 실시하는 것



시퀀셜 데이터는 길이가 다양하므로 Batch의 크기가 1이 될 수밖에 없음

⋮

Attention 구조나 RNN에서 일반적으로 더 많이 사용됨

# 4

데이터 증강

## 데이터 증강

## 데이터 증강

원본 데이터로부터 새로운 데이터를 만드는 기법  
학습 데이터 부족으로 과적합 위험이 있거나,  
클래스 불균형을 해소할 때 사용

## 이미지 증강

- 기하학적 변환
- 색 변환
- 이미지 섞기
- 랜덤하게 자르기

## 텍스트 증강

- 역번역
- EDA



## 데이터 증강

## 데이터 증강의 장점

데이터 증강

원본 데이터  모델 예측력 향상 만드는 기법

학습 데이터 부족으로 과적합 위험이 있거나,  
데이터 수집 비용 감소



희귀한 사건에 대한 예측력 강화



이미지 증강

텍스트 증강

자연어 데이터의 경우 의미 변형에 유의해서 사용

- 가이더를 통한

- 색 변화

ex) NLP팀은 뒤풀이를 안 갔다 / NLP팀은 뒤풀이를 못 갔다

- 이미지 섞기

- 역번역

- EDA

- 랜덤하게 자르기



## 데이터 증강 in 컴퓨터 비전

### 기하학적 변환

기존 이미지를 **crop, rotate, flip** 등의 변환을 주어 새로운 이미지 생성

### 색 변환

기존 RGB **색상에 다양한 변화**를 주어 새로운 이미지 생성

### 이미지 섞기 & 랜덤하게 자르기

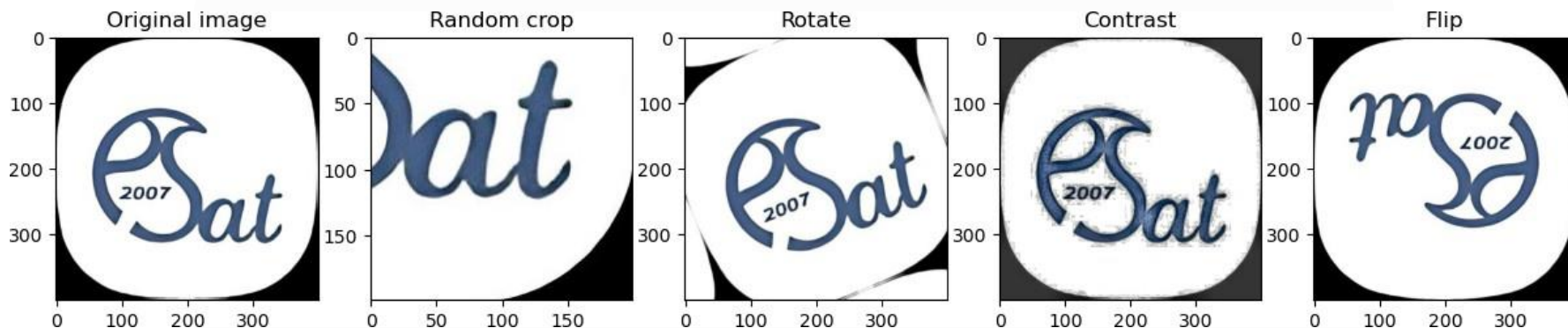
여러 개의 **이미지를 섞거나**  
랜덤하게 **픽셀을 지워** 새로운 이미지 생성

## 데이터 증강 in 컴퓨터 비전

## 기하학적 변환

기존 이미지를 **crop, rotate, flip** 등의  
변환을 주어 새로운 이미지 생성

## 색 변환



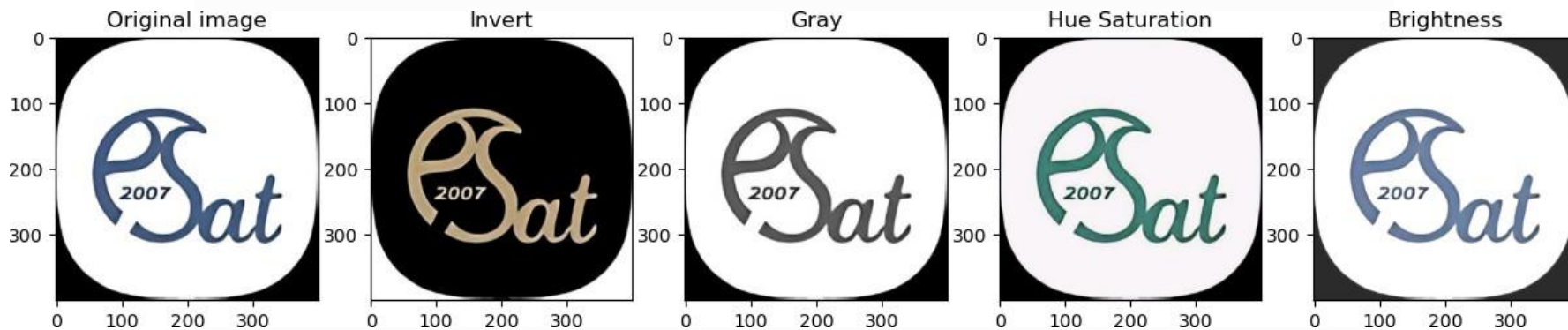
랜덤하게 픽셀을 지워 새로운 이미지 생성

## 데이터 증강 in 컴퓨터 비전

## 색 변환

기존 RGB 색상에 다양한 변화를 주어  
새로운 이미지 생성

## 기하학적 변환



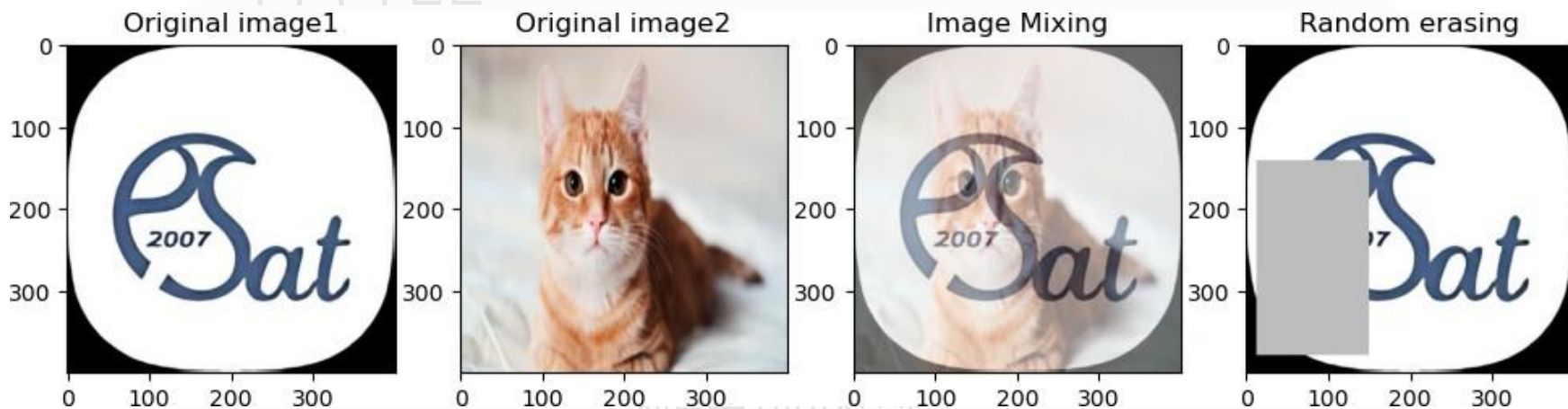
랜덤하게 픽셀을 지워 새로운 이미지 생성

## 데이터 증강 in 컴퓨터 비전

## 이미지 섞기 &amp; 랜덤하게 자르기

여러 개의 이미지를 섞거나  
랜덤하게 픽셀을 지워 새로운 이미지 생성

기하학적 변환



## 데이터 증강 in 자연어 처리

### 역번역

원본 문장을 다른 언어로 번역 후  
다시 원본 언어로 번역해 새로운 데이터 생성

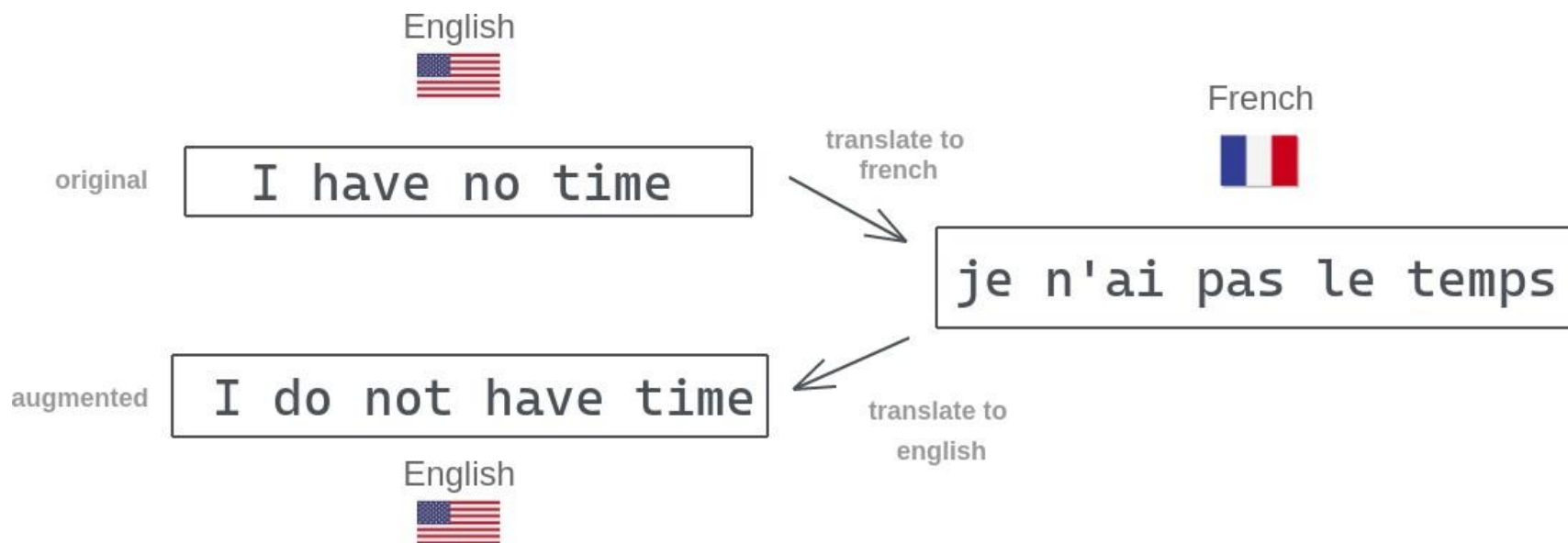
### EDA

문장을 변형시키는 SR, RI, RS, RD 방법을 통해  
새로운 데이터 생성

## 데이터 증강 in 자연어 처리

## 역번역

원본 문장을 다른 언어로 번역 후  
다시 원본 언어로 번역해 새로운 데이터 생성



## 데이터 증강 in 자연어 처리

EDA(Easy Data Augmentation)

문장을 변형시키는 SR, RI, RS, RD 방법을 통해  
새로운 데이터 생성

SR(Synonym Replacement)

특정 단어를 유의어로 교체하는 방법

RS(Random Swap)

임의의 두 단어 위치를 교체하는 방법

RI(Random Insertion)

임의의 다른 동의어를 삽입하는 방법

RD(Random Deletion)

임의의 단어를 삭제하는 방법

## 데이터 증강 in 자연어 처리

## SR(Synonym Replacement)

**Original sentence:** This *article* will focus on summarizing data augmentation *techniques* in NLP.

**Augmented sentence:** This *write-up* will focus on summarizing data augmentation *methods* in NLP.

## RI(Random Insertion)

**Original sentence:** This *article* will focus on summarizing data augmentation *techniques* in NLP.

**Augmented sentence:** This article will focus on *write-up* summarizing data augmentation techniques in NLP *methods*.

랜덤한 단어 삽입 시 문장의 의미를 크게 바꾸기 때문에 동의어 삽입



## 데이터 증강 in 자연어 처리

## RS(Random Swap)

**Original sentence:** This *article* will focus on summarizing data augmentation *techniques* in NLP.

**Augmented sentence:** This *techniques* will focus on summarizing data augmentation *article* in NLP.

## RD(Random Deletion)

**Original sentence:** This article *will* focus on summarizing data augmentation *techniques* in NLP.

**Augmented sentence:** This article  focus on summarizing data augmentation  in NLP.

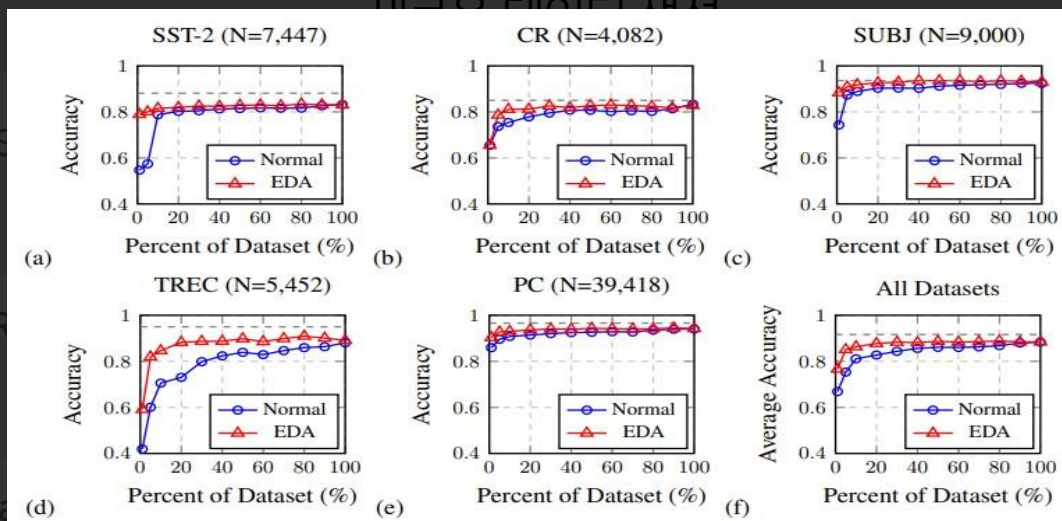


## 데이터 증강 in 자연어 처리 EDA에 대한 의문점

EDA(Easy Data Augmentation)

EDA는 단어를 랜덤하게 적용하기 때문에

SR, RI, RS, RD 방법을 통해  
직관적으로 볼 때 효과가 없어보임



하지만 해당 방법을 제안한 논문에 따르면

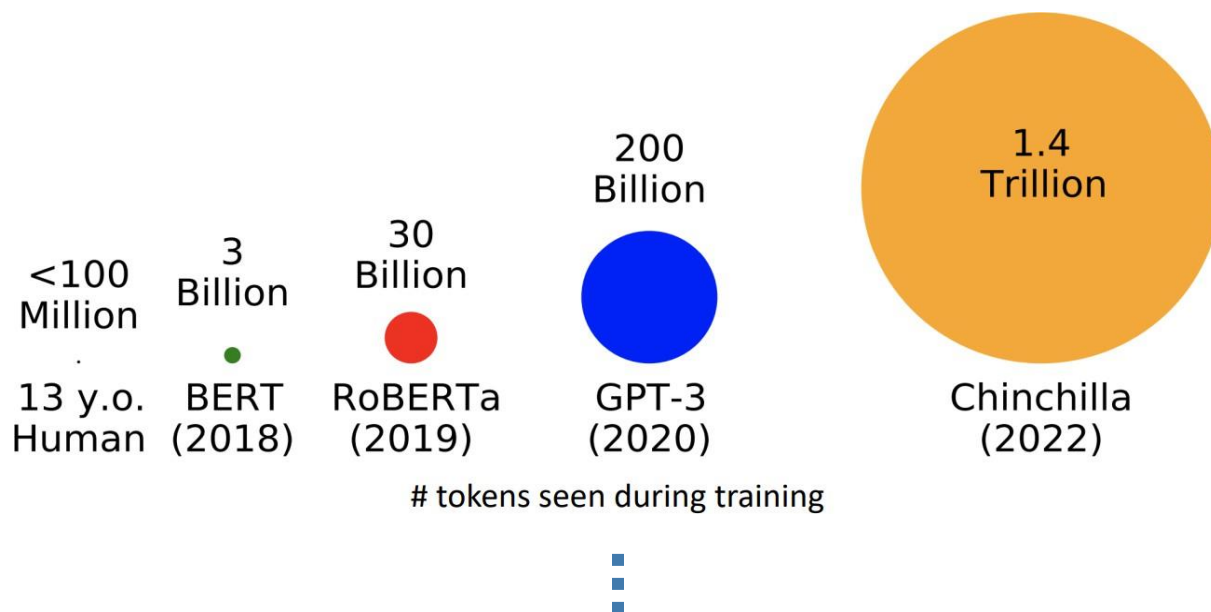
EDA는 학습 데이터가 적은 상황에서

일부 노이즈를 추가함으로 과적합 방지의 효과가 있음

# 5

파인 튜닝

## 파인 튜닝



딥러닝 모델들은 점차 커져 감

복잡한 모델을 수많은 데이터셋으로 학습하는 것은 불가능

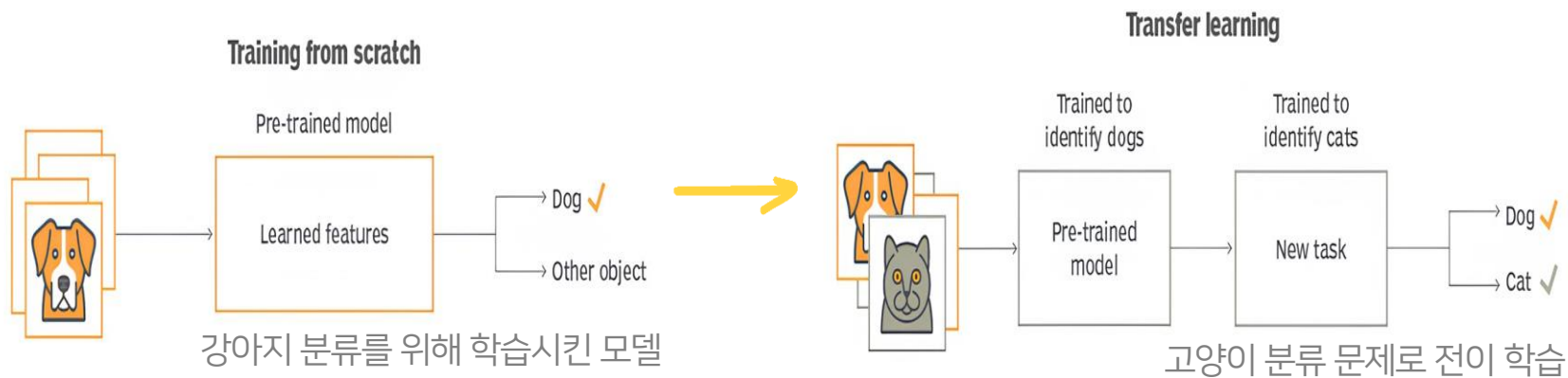


이미 훈련된 사전 학습 모델 필요

## 전이 학습 (Transfer Learning)

### 전이 학습

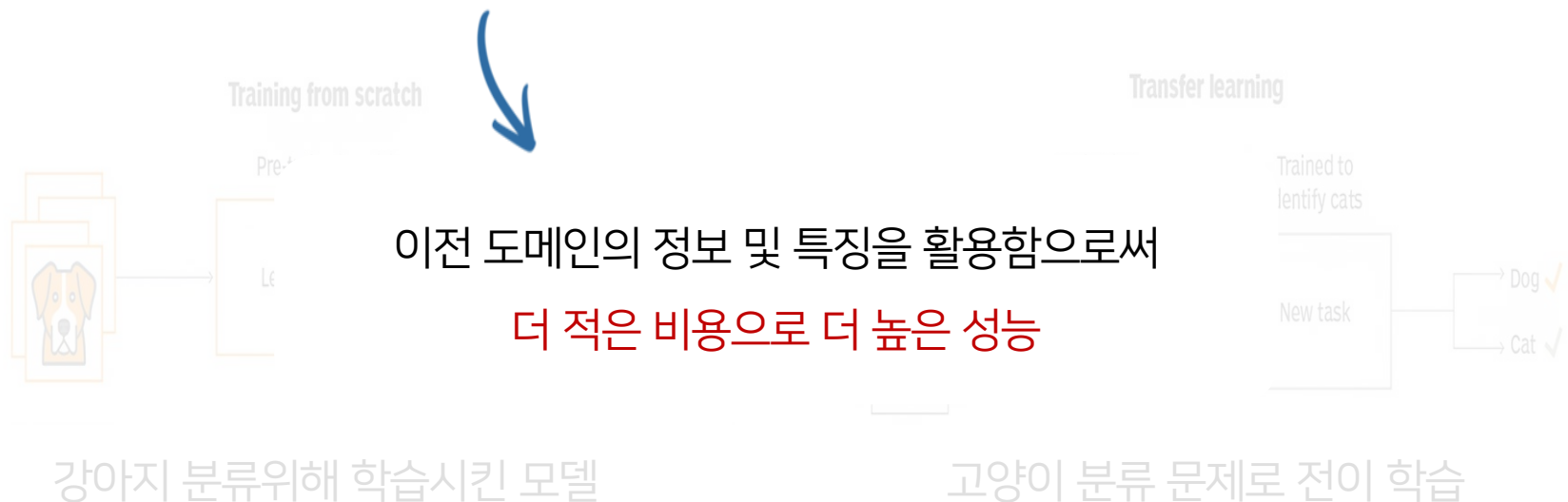
한 분야의 문제를 해결하기 위해 얻은 지식과 정보를  
다른 문제를 푸는데 사용하는 방식



## 전이 학습 (Transfer Learning)

### 전이 학습

한 분야의 문제를 해결하기 위해 얻은 지식과 정보를  
다른 문제를 푸는데 사용하는 방식



## 파인 튜닝 (Fine Tuning)

### 파인 튜닝 (Fine Tuning)

사전 학습 모델의 가중치가 새로운 데이터에 대해 훈련되는 전이 학습에 대한 접근 방식

사전 학습 모델을 사용하는  
효과적인 방법인 전이 학습



새로운 문제

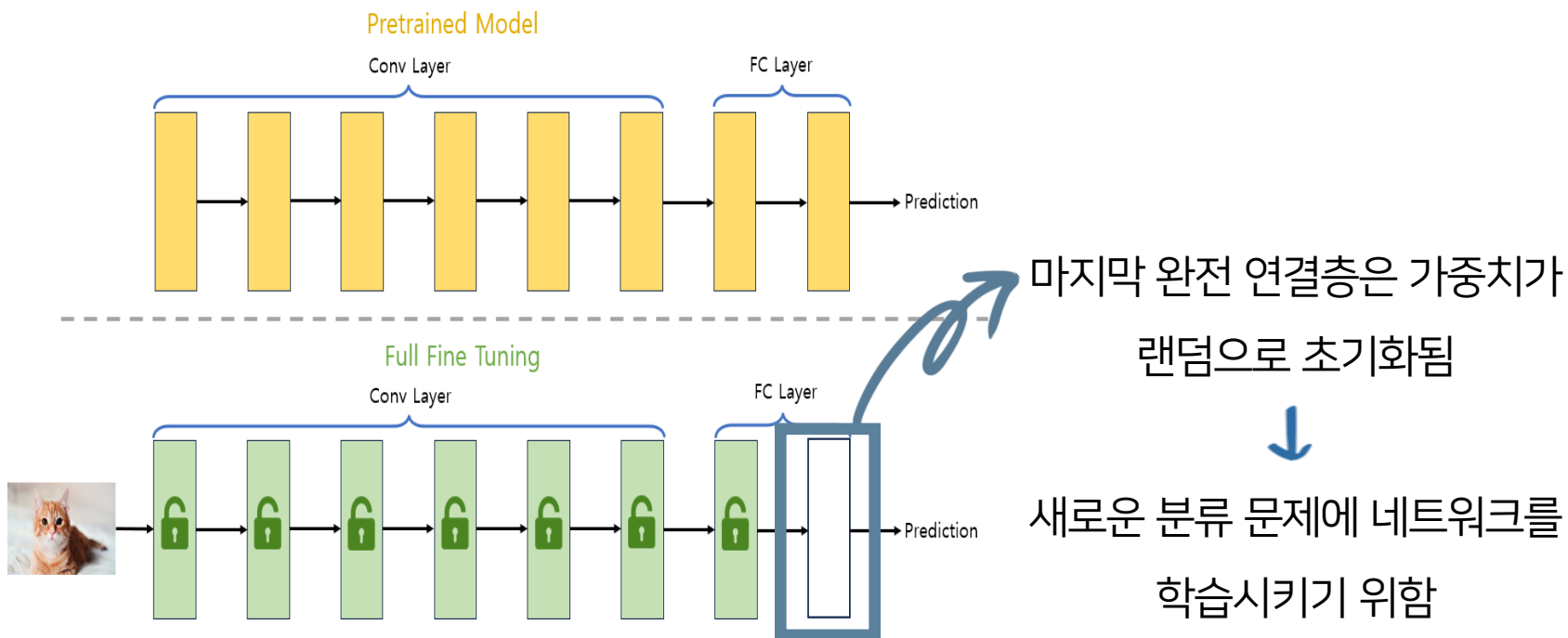


사전 학습 모델을 문제에 따라 변형시켜 더 좋은 성능을 만들 수 있는 방법

## 파인 튜닝 예시

## 신경망 전체에 적용하기 (Full Fine Tuning)

사전 학습 모델의 가중치를 초기값으로 설정 후  
모델의 학습에 따라 모든 가중치를 업데이트하는 방식





## 파인 튜닝 예시

### 신경망 전체에 적용하기 (Full Fine Tuning)

사전 학습 모델의 가중치를 초기값으로 설정 후  
모델의 학습에 따라 모든 가중치를 업데이트하는 방식

#### 도메인 특화

특정 도메인의 데이터로 모델을 적응시켜  
일반적인 모델보다 더 좋은 성능을 보임

사전 학습 모델과 새로운 문제의  
도메인 사이에 차이가 크더라도 사용 가능

#### 비싼 학습 비용

매우 큰 모델의 모든 가중치 업데이트는  
계산량이 매우 많아 학습 비용이 비쌈

높은 수준의 하드웨어 요구  
큰 모델 학습 위해 높은 수준의  
GPU나 TPU 필요

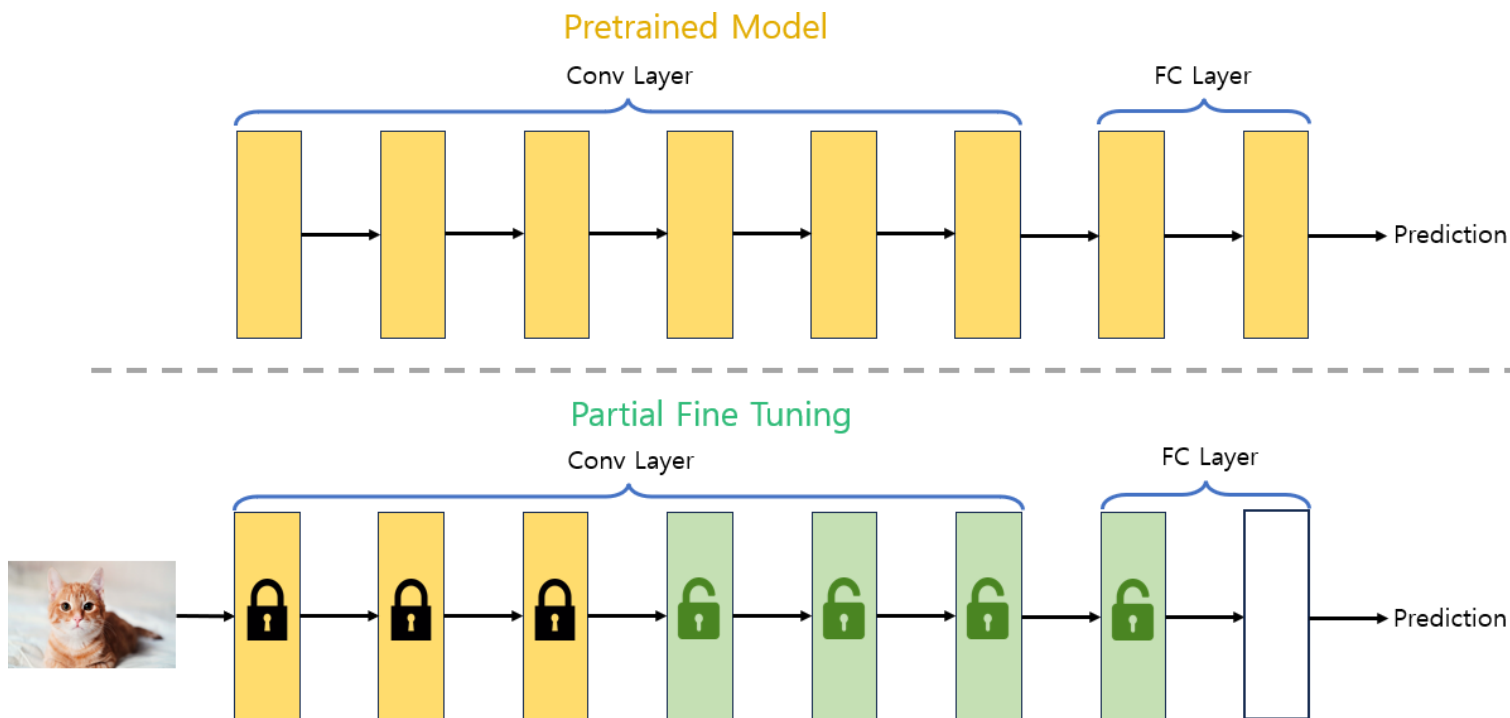
# 5

## 파인 튜닝

### 파인 튜닝 예시

#### 일부 레이어에 적용하기 (Partial Fine Tuning)

앞의 일부는 고정하고(Freezing) 나머지 파라미터 일부에 대해 파인 튜닝을 적용하는 방법



# 5

## 파인 튜닝

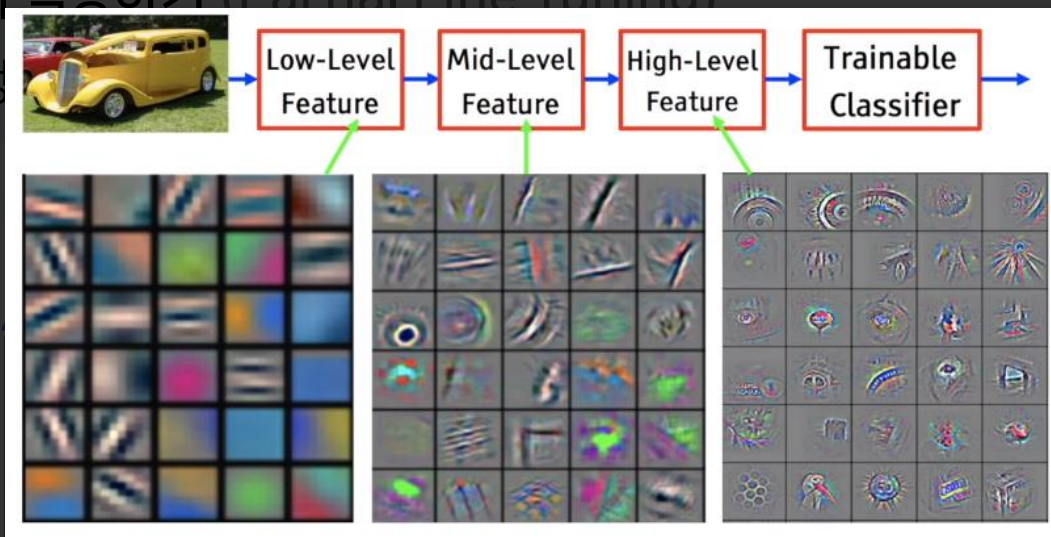


파인 튜닝 예시

### 심층 레이어를 Unfreezing하는 이유

일부 레이어에 적용하기 (Partial Fine Tuning)

앞의 일부는 고정



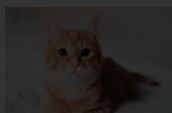
을 적용하는 방법

저층 레이어에서는 낮은 수준의 특징을 포착하지만,

Conv Layer

FC Layer

심층 레이어에서는 문제와 관련될 가능성이 높은 복잡한 특징을 포착함



심층 레이어의 Unfreezing으로 효과적인 학습 가능

Prediction

## 파인 튜닝 예시

### 일부 레이어에 적용하기 (Partial Fine Tuning)

앞의 일부는 고정하고(Freezing) 나머지 파라미터 일부에 대해 파인 튜닝을 적용하는 방법

#### 도메인 적응

Full Fine Tuning에 비해 덜 유연하지만,  
파라미터 업데이트를 통해  
새로운 도메인에 잘 적응

#### 더 저렴한 학습 비용

일부 파라미터만 업데이트 진행

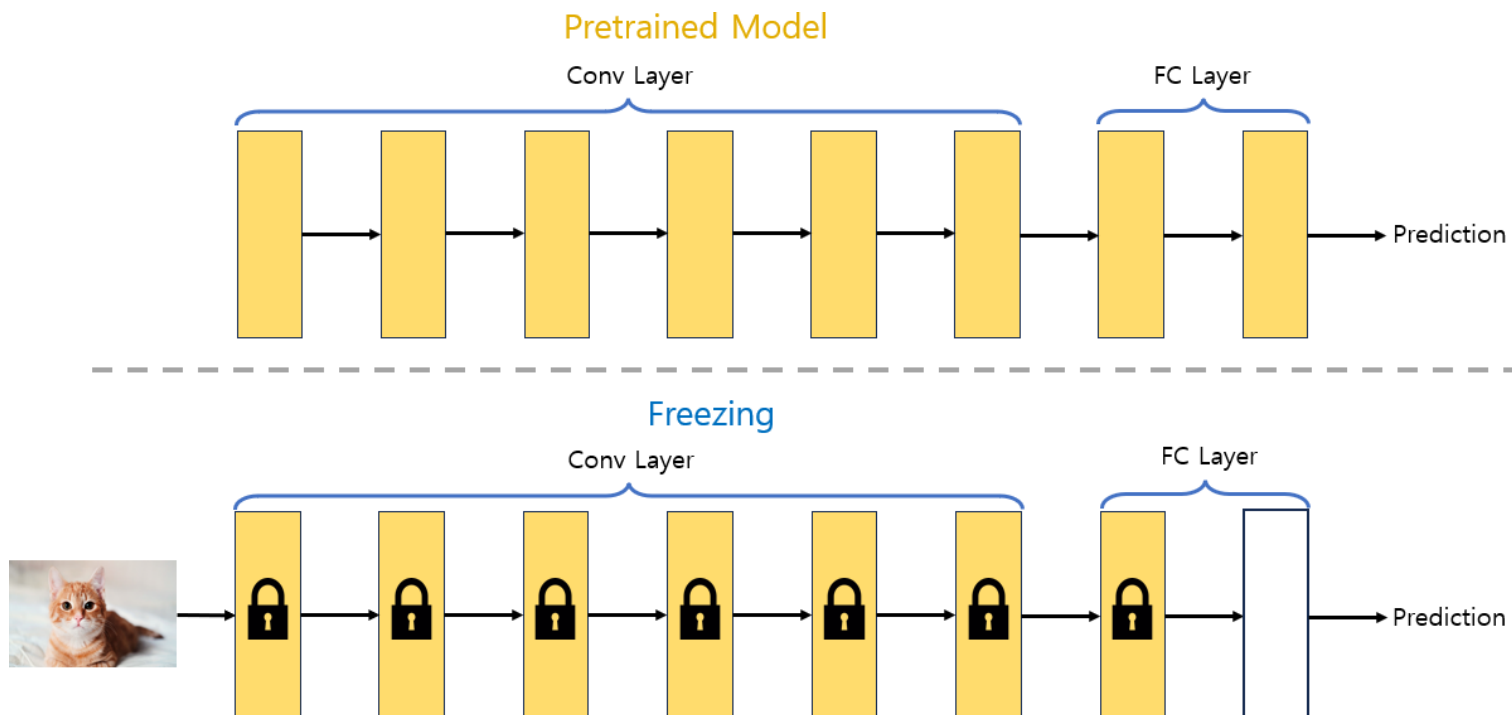
#### 높은 수준의 하드웨어 요구

여전히 큰 모델 학습 시키는 일이기에  
높은 수준의 하드웨어 요구

## 파인 튜닝 예시

## 모델 얼리기 (Freezing)

마지막 완전 연결층을 제외한 모든 가중치들을 고정하는 방법



## 파인 튜닝 예시

## 모델 얼리기 (Freezing)

마지막 완전 연결층을 제외한 모든 가중치들을 고정하는 방법

## 유사 도메인 적응력

비슷한 도메인의 문제에 대해

좋은 성능을 보임

## 적은 하드웨어 부담

마지막 완전 연결층만 업데이트 해

학습 비용과 하드웨어 부담이 적음

## 부족한 도메인 적응력

앞의 두 모델에 비해 유연성이 떨어져

도메인 적응력 부족

# 6

임베딩

## 임베딩

텍스트, 이미지, 오디오 등의 객체를  
고정적인 크기의 실수 벡터 형태로 표현한 결과물



숫자를 나열한 리스트  
{1, 5, 13, ...}



"BANKING" = {0.286, 0.792, -0.177, -0.107, ..., 0.271}

이처럼 하나의 단어를 일련의 숫자들로 구성된 벡터로 표현



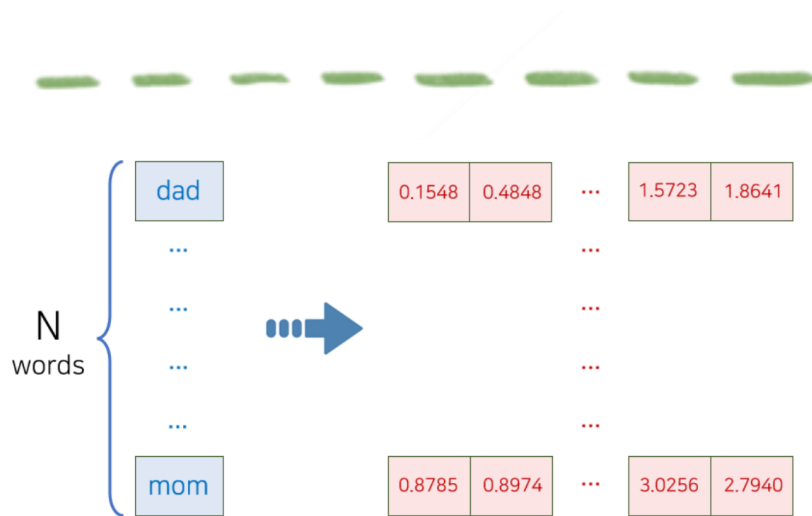
# 6

## 임베딩(embedding)

### 임베딩 예시

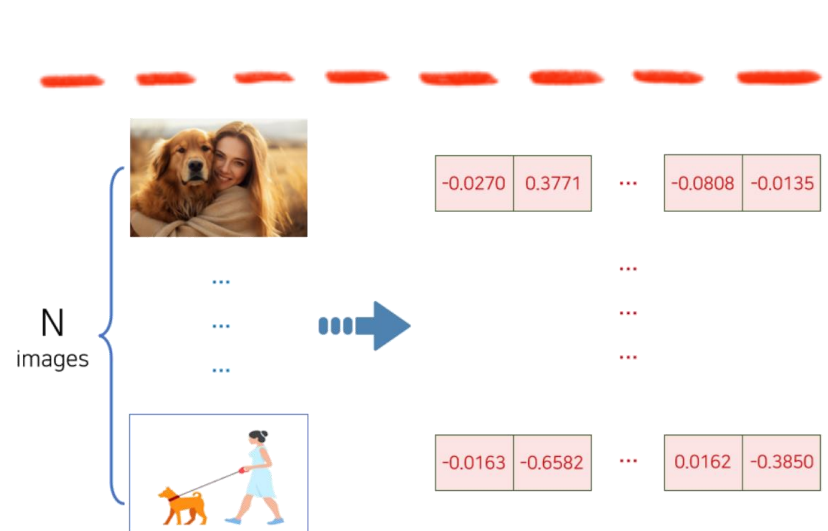
#### 단어 임베딩

dad와 mom을 포함한  
N개의 단어를 임베딩을 통해 표현



#### 이미지 임베딩

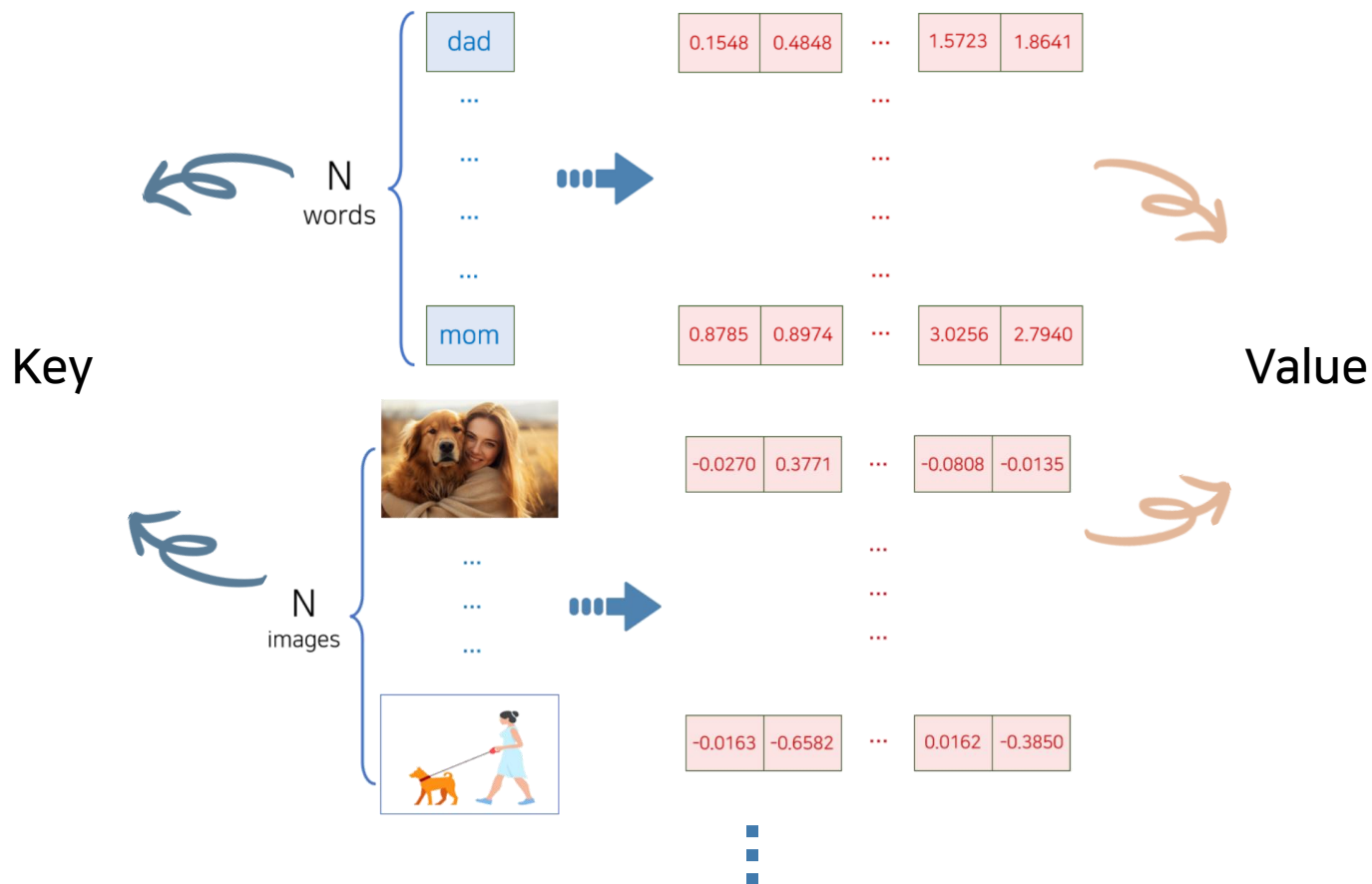
N개의 이미지를  
임베딩을 통해 표현



## 6

## 임베딩(embedding)

## 임베딩 예시



임베딩은 Key-Value 형태로 대응되는 하나의 사전임

## 6

## 임베딩(embedding)

## 임베딩 예시

Key

각 요소에 해당하며  
해당 인덱스를 통해  
임베딩 행렬에 접근

N  
words

dad

...

...

...

...

mom

0.1548

0.4848

...

1.5723

1.8641

0.8785

0.8974

...

3.0256

2.7940



-0.0270

0.3771

...

-0.0808

-0.0135

N  
images

...

...

...



-0.0163

-0.6582

...

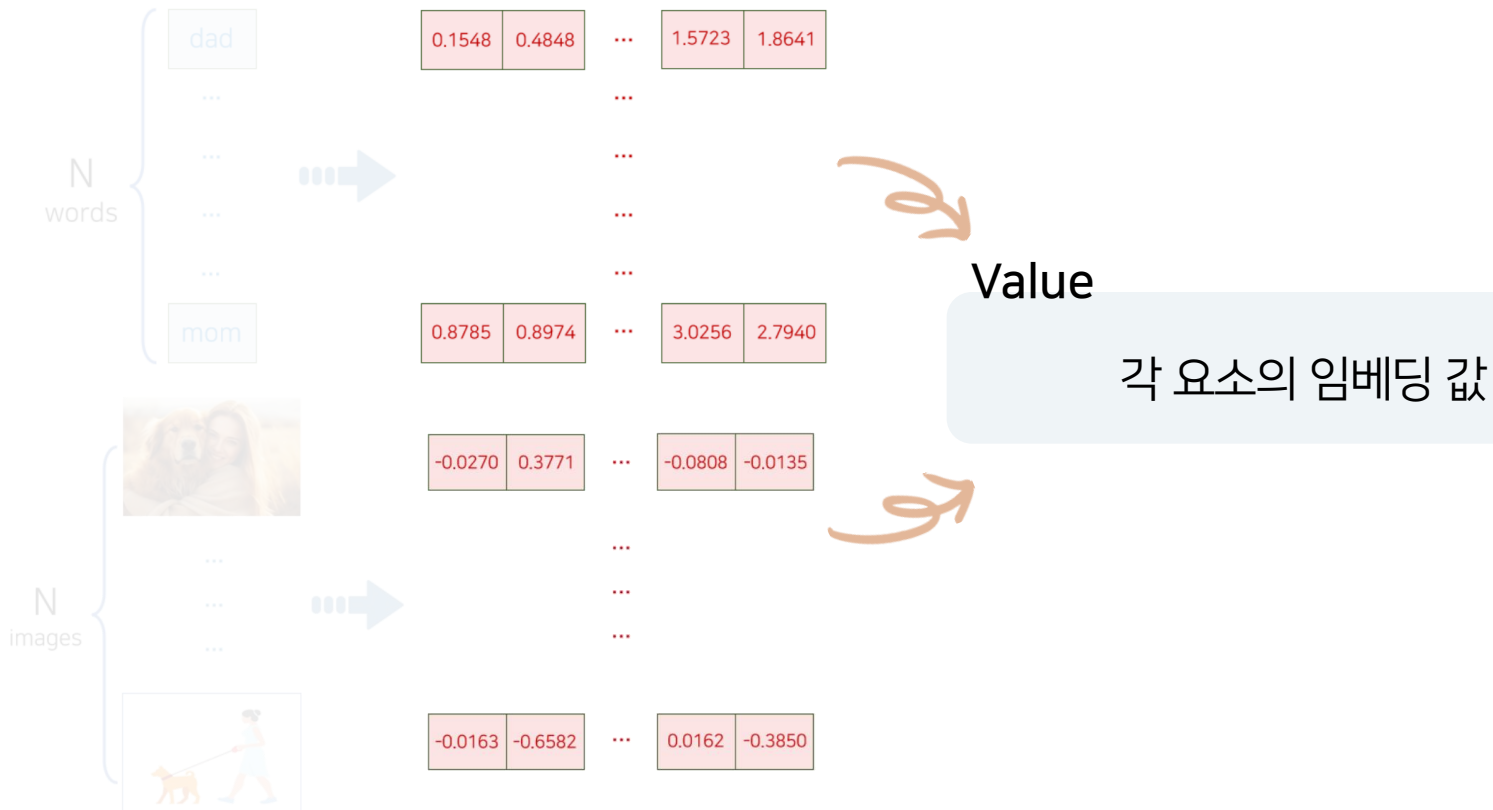
0.0162

-0.3850

## 6

## 임베딩(embedding)

## 임베딩 예시



## 6

## 임베딩(embedding)

## 임베딩 접근방법

Sentence

This is my car

→ 텍스트 데이터가 주어짐

Vocab ( a aaron ... car ... is ... my ... this ... zombie )



Indices ( 0 1 ... 3412 ... 5281 ... 6899 ... 8678 ... 9999 )

→ 각각의 단어는 그에 상응하는  
인덱스 값에 매칭Vocab indices  
= Input $x_0$  $x_1$  $x_2$  $x_3$ 

8678

5381

6899

3412

→ 이 인덱스 값으로 Input  
embedding에 접근

e.g. 'car'을 표현하고 싶다면 car에 대응되는 임베딩 행렬의 3412번째 요소를 가져오면 됨

## 6

## 임베딩(embedding)

## 임베딩 접근방법



Sentence { This is my car } → 그러면 .. 이 벡터들은 어떻게 만든 거지? 텍스트 데이터가 주어짐

Vocab ( a aaron ... car ... is ... my ... this ... zombie )  
 Indices ( 0 1 ... 3412 ... 5281 ... 6899 ... 8678 ... 9999 )

임베딩은 사전 학습된 모델을 사용하여 만들  
 ↓  
 각각의 단어는 그에 상응하는  
 인덱스 값에 매칭

텍스트 데이터 : Word2Vec, GloVe, Bert

이미지 데이터 : CNN, ResNet, ViT

Vocab indices = Input { 8678 5381 6899 3412 } → 이 인덱스 값이 Input embedding에 전달

구체적인 방법은 NLP 2주차 클린업 참고!

e.g. 'car'을 표현하고 싶다면 car에 대응되는 임베딩 행렬의 3412번째 요소를 가져오면 됨

## 임베딩의 목적

임베딩은 텍스트, 이미지 등의 **고차원의 데이터**를  
**저차원의 벡터**로 축소하는 방식



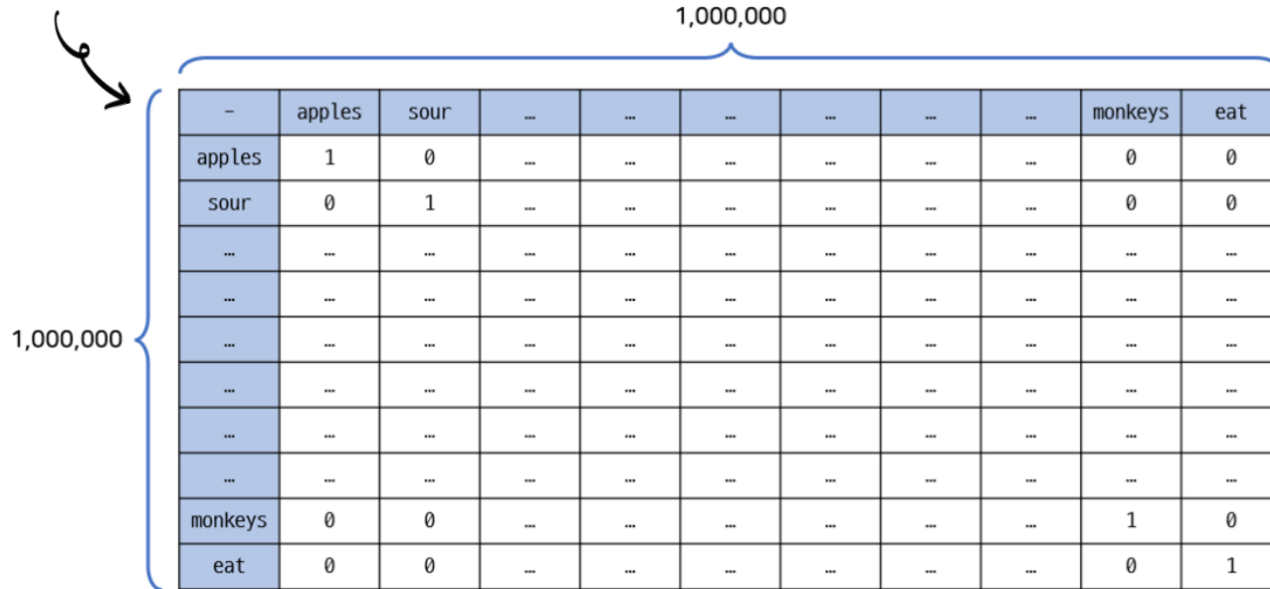
데이터 존재 여부에 따라 0 또는 1로 표현하는 기존의 벡터화 방식인  
One-Hot Encoding의 비효율성을 해결

## 6

## 임베딩(embedding)

## 임베딩의 목적

One-Hot Encoding



The diagram illustrates a One-Hot Encoding matrix. A large blue bracket on the left indicates the matrix has 1,000,000 rows. A smaller blue bracket at the top indicates it has 1,000,000 columns. An arrow points from the text 'One-Hot Encoding' to the matrix. The matrix is a grid where each row represents a word and each column represents a position in the vocabulary. The words 'apples', 'sour', 'monkeys', and 'eat' are shown as examples. 'apples' has a 1 in the second column and 0s elsewhere. 'sour' has a 1 in the third column and 0s elsewhere. 'monkeys' has a 1 in the 1,000,000th column and 0s elsewhere. 'eat' has a 1 in the 1,000,001st column and 0s elsewhere. All other words (represented by '...' in the first three rows) have 0s in all columns.

	-	apples	sour	...	...	...	...	...	...	monkeys	eat
apples	1	0	...	...	...	...	...	...	...	0	0
sour	0	1	...	...	...	...	...	...	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
monkeys	0	0	...	...	...	...	...	...	...	1	0
eat	0	0	...	...	...	...	...	...	...	0	1

위의 예시처럼 100만 개의 단어를 가지고 있다면, 단어 벡터 하나의 차원은 100만이 됨  
 → 심각한 공간의 낭비 + 단어의 의미 표현 X ( $\because$  0, 1로만 인코딩)

이러한 한계점을 보완할 수 있는 것이 임베딩!



# 6

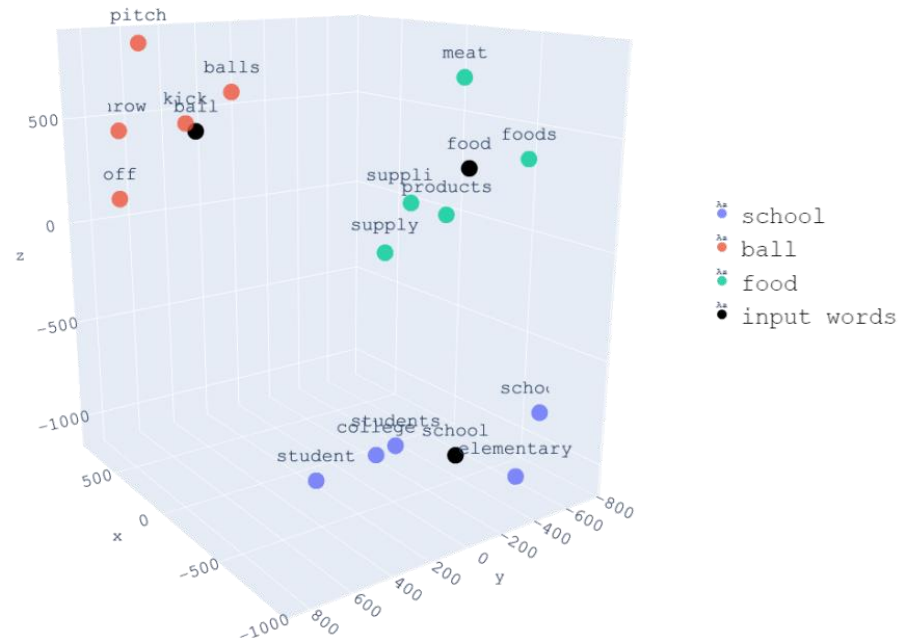
## 임베딩(embedding)

### 임베딩의 이점

#### 의미 표현

사전 학습된 모델들은 각 단어/이미지  
사이의 관계, 유사도, 특징들을 학습하여  
하나의 벡터로 만들어 냄

→ 데이터의 의미를 표현할 수 있음



INPUT WORDS: school, ball, food

위 예시를 통해

단어들의 의미와 관계가 보존됨을 파악 가능

# 6

## 임베딩(embedding)

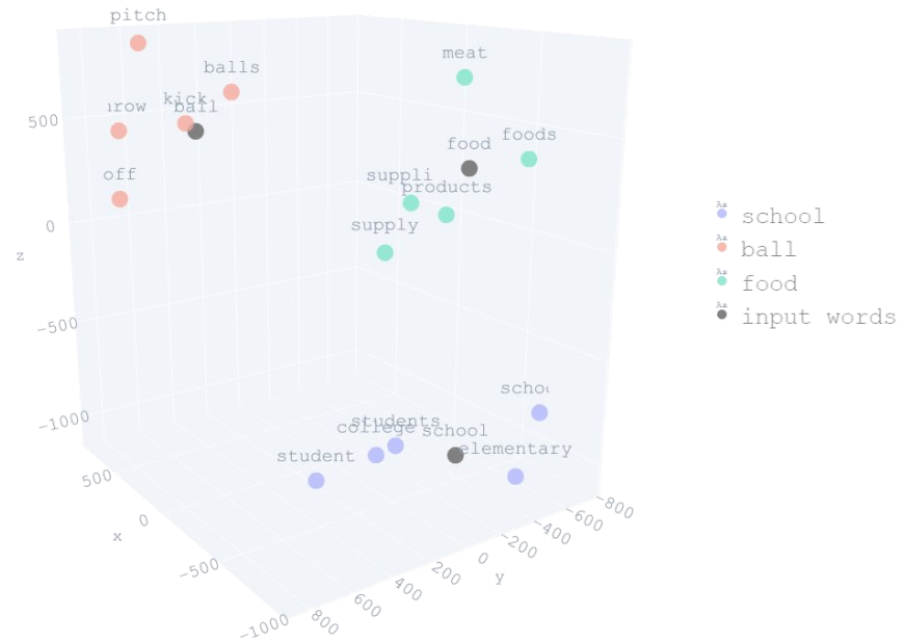
### 임베딩의 이점

#### 유사도 계산

- ✓ 비슷한 데이터는 가까운 거리에 위치
- ✓ 임베딩은 데이터의 의미를 보존



코사인 유사도를 통해  
데이터 사이의 유사도를 계산



#### 코사인 유사도

$$similarity = \frac{a \cdot b}{|a| \cdot |b|}$$

## 임베딩의 이점

### 차원축소

임베딩은 고차원의 데이터를 저차원의 밀집 벡터(Dense Vector)로 표현

→ 효율적인 계산 가능, 메모리 절약

GPT도 256 ~ 3,096차원 <-> One-Hot Encoding은 수백 만 차원

### 일반화 성능

모델이 이전에 본 적이 없는 예시에 대해서도 좋은 성능을 낼 수 있음  
여전히 데이터 자체의 의미와 데이터 간의 관계를 보존하고 있기 때문

# 다음 주 예고

---

1. 자연어

2. 단어 표현 방법

3. 언어 모델링(LM)

4. 기계 번역(MT)

---

**감사합니다**

---