ONECLICK AI

정규화와 UNet

연준모

목차

1. 정규화의 종류와 차이 정규화?

정규화란?

딥러닝은 선형 변환 될 때 다음과 같은 방식으로 변환된다 $y = wx + b$

딥러닝에서, 은닉층이 2 층이라고 가정했을 때, 1층에서의 출력값이 2층의 입력으로 들어간다는 사실을 이미 알고 있다.

학습의 어려움 : 내부 공변량 변화 (Internal Covariate Shift, ICS)

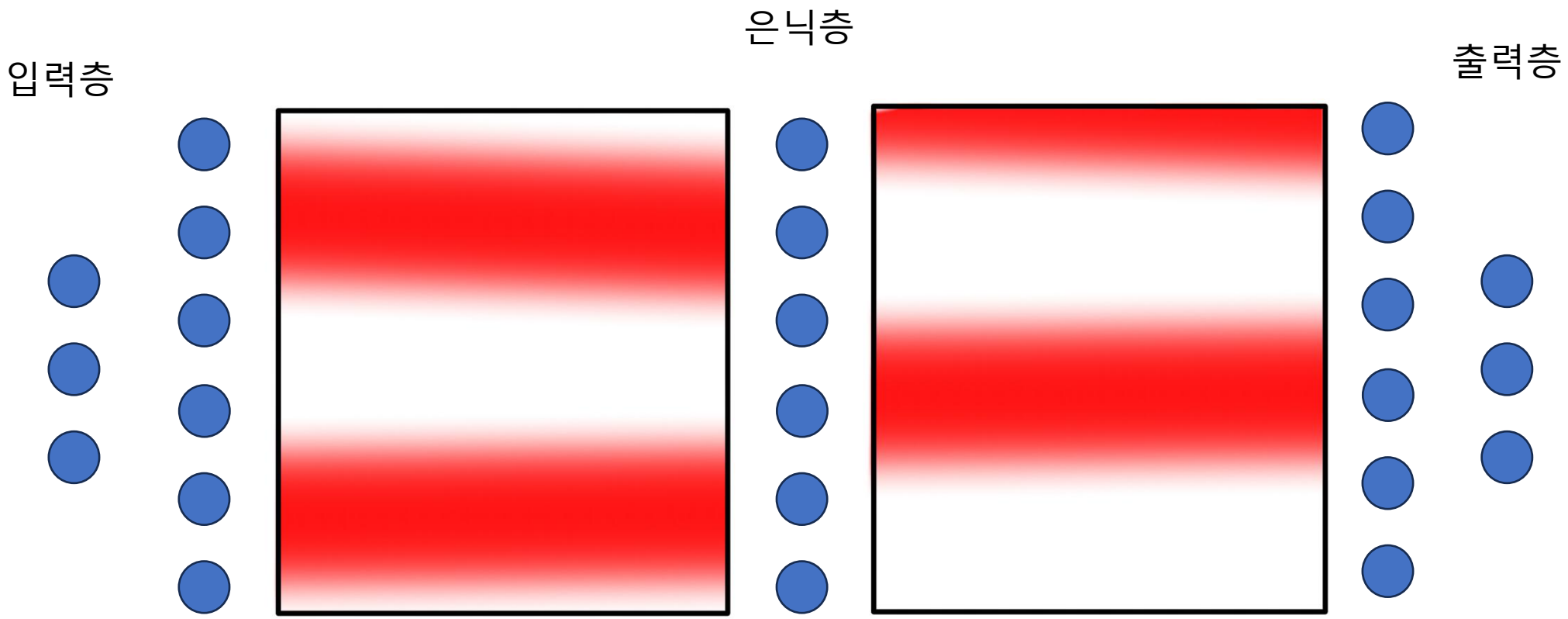
딥러닝은 층이 깊습니다. 1번 레이어의 가중치 W_1이 업데이트되면, 그 출력 y_1의 분포(평균, 분산)가 바뀐다(w가 곱해지고, b가 더해지기 때문)

2번 레이어는 이 y_1을 입력으로 받는데, 학습 스텝마다 입력 데이터의 분포가 계속 변하는 문제가 발생한다.

이는 2번 레이어가 안정적으로 학습하는 것을 방해한다. 비유하자면, 계속해서 움직이는 과녁을 맞추려는 것과 같다.

1. 정규화의 종류와 차이 정규화?

정규화란?



이렇게 분포가 바뀌면 진동이 생겨 딥러닝이 망한다

1. 정규화의 종류와 차이 정규화?

정규화란?

정규화의 핵심 목적

이 ICS 문제를 해결하는 것이 정규화의 가장 중요한 목적이다.

각 레이어에 들어오는 입력 벡터들의 분포를 강제로 평균 0, 분산 1로 표준화 시킨다.

정규화의 주 목적

1. 학습 안정화: 모든 레이어가 일관된 분포의 입력을 받아 학습에만 집중할 수 있다.
2. 학습 가속화: 학습이 안정되므로, 학습률(Learning Rate)을 훨씬 높게 설정할 수 있어 모델이 빠르게 수렴한다.

데이터 분포 안정화

1. 정규화의 종류와 차이 정규화?

정규화란?

대표적인 정규화에는 다음 4 가지가 있는데,

- Batch Normalization 2
- Layer Normalization 1
- Instanse Nomalization 3 특수 목적으로 잘 안쓰임
- Group Nomalization 4 특수 목적으로 잘 안쓰임

중요도 순서

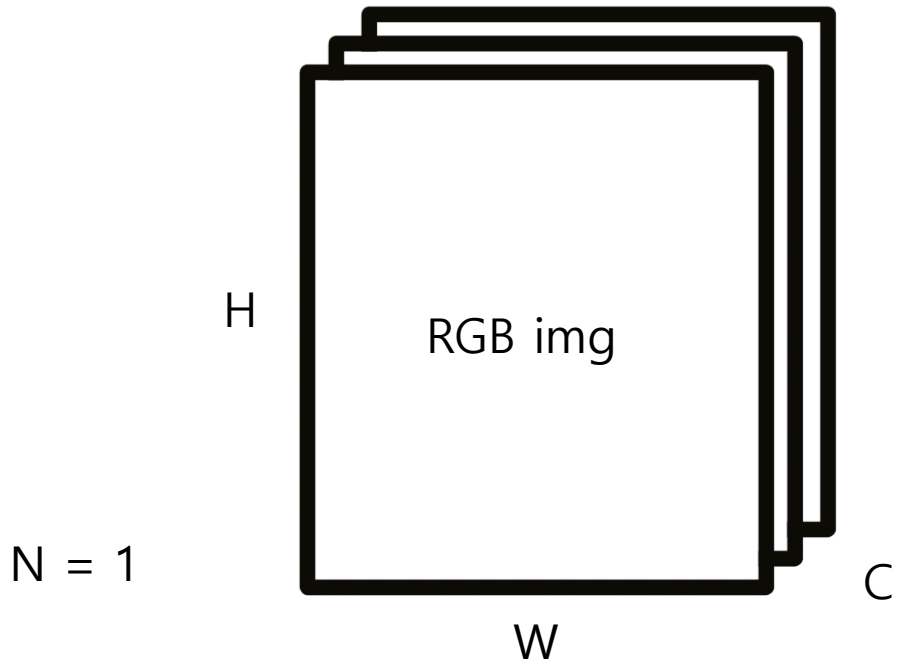
Batch Nomalization?

통계량(평균, 분산)을 미니배치단위로 계산하여, 입력 분포의 변화(Internal Covariate Shift)를 줄인다.

$x \in R^{N \times C \times H \times W}$

- (N): 배치 크기 (batch size)
- (C): 채널 수
- (H, W): 공간적 크기

주로 CNN에서 사용



1. 정규화의 종류와 차이 1. Batch Normalization?

Batch Nomalization?

Batch Size?

한 번의 가중치 업데이트(학습)에 사용되는 데이터 샘플의 개수

총 데이터가 1000개라면,

한번에 100개씩 10번 학습할 것인지, 한 번에 1000개를 다 학습할 것인지

특징	큰 배치 크기 (예: 256, 1024)	작은 배치 크기 (예: 8, 32, 64)
학습속도	빠름 (GPU 병렬 처리에 유리)	느림 (데이터를 자주 나눠서 처리)
메모리 사용량	매우 높음 (큰 GPU VRAM 필요)	낮음 (작은 GPU에서도 가능)
하습 안정성	안정적 (많은 데이터의 평균을 따름)	불안정 (노이즈) (적은 데이터에 민감)
일반화 성능	나빠질 수 있음 (Sharp Minima)	좋아질 수 있음 (Flat Minima)

1. 정규화의 종류와 차이 1. Batch Normalization?

Batch Nomalization?

각 채널 c 에 대해, 배치 전체의 ($N \times H \times W$) 원소에 대해 평균과 분산 계산

평균

$$\mu_c = \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{n,c,h,w}$$

N(Batch Size) : 배치에 포함된 이미지의 개수 (예: 32장)
C (Channel) : 채널 개수 (예: 흑백은 1, RGB 컬러는 3)
H (Height) : 이미지의 세로 픽셀 수
W (Width) : 이미지의 가로 픽셀 수

$\boldsymbol{x_{n,c,h,w}}$: n번째 이미지의 c번째 채널, (h, w) 위치에 있는 픽셀(특성) 값 하나를 의미

$\sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W}$: c번째 채널에 해당하는 모든 값($x_{n,c,h,w}$)을 전부 더하라

Batch Nomalization?

각 채널 c 에 대해, 배치 전체의 ($N \times H \times W$) 원소에 대해 평균과 분산 계산

분산

$$\sigma_c^2 = \frac{1}{NHW} \sum_{n,h,w} \left(x_{n,c,h,w} - \mu_c\right)^2$$

편차의 제곱을 다 더한 후

데이터의 총개수로 나누어 분산 구한다

Batch Nomalization?

정규화 및 복원 단계

$$\hat{x} * n, c, h, w = \frac{x * n, c, h, w - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}$$ 정규화

$x_{n,c,h,w} - \mu_c$: (값 - 평균). 이렇게 하면 평균이 0

$\sqrt{\sigma_c^2 + \epsilon}$: 표준편차

ϵ : 아주 작은 값. 분모가 0이 되는 것을 방지하기 위함

→ 모든 데이터가 평균 0, 표준편차 1인 분포로 정규화

1. 정규화의 종류와 차이 1. Batch Normalization?

Batch Nomalization?

데이터를 너무 강하게 (평균 0, 분산 1로) 제약했다.
하지만 어쩌면 네트워크는 평균이 0.5, 분산이 1.2인 상태를 더 선호할 수도 있다
네트워크가 스스로 최적의 평균과 분산을 학습할 수 있도록 유연성을 더해준다

$$y_{n,c,h,w} = \gamma_c \widehat{x_{n,c,h,w}} + \beta_c$$

정규화된 데이터($\hat{x}$)를 학습 가능한 파라미터(γ_c, β_c)를 이용해 스케일을 조절하고 이동시킨다.

γ_c : 새로운 스케일(분산)을 조절.
β_c : 새로운 이동(평균)을 조절.
$\widehat{x_{n,c,h,w}}$: 방금 만든 '평균 0, 분산 1'짜리 데이터 이다.
$y_{n,c,h,w}$: 최종적으로 다음 레이어로 전달되는 결과값.

- 배치 크기에 의존적
- 학습 안정화, 수렴 가속, 규제 효과 (regularization)
- 작은 배치에서는 통계량이 불안정 → 성능 저하

정규화를 안 하는 게 낫다고 판단하면 $\gamma_c = \sigma_c$, $\beta_c = \mu_c$에 가깝게 학습하여 정규화 효과를 되돌릴 수도 있다
이 γ_c와 β_c는 모델이 학습 과정에서 스스로 최적의 값을 찾아간다.

1. 정규화의 종류와 차이 2. Layer Normalization?

Layer Nomalization?

배치 크기 의존성을 없애기 위해, 각 샘플 단위로 전체 피처의 통계량을 계산한다.

$$x \in R^{N \times C \times H \times W}$$

Batch와 마찬가지로, 평균 분산 계산 하지만, batch에서는 동일한 채널에서만, 여기서는 한 데이터에 대해

$$\mu_n = \frac{1}{CHW} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{n,c,h,w}$$

$$\sigma_n^2 = \frac{1}{CHW} \sum_{c,h,w} \left(x_{n,c,h,w} - \mu_n\right)^2$$

$$\hat{x} * n, c, h, w = \frac{x * n, c, h, w - \mu_n}{\sqrt{\sigma_n^2 + \epsilon}}$$

정규화

$$y_{n,c,h,w} = \gamma_c \widehat{x_{n,c,h,w}} + \beta_c$$

복원

- 배치 크기와 무관
- RNN, Transformer 등 시퀀스 기반 모델에서 주로 사용
- 미니배치 크기가 1이어도 안정적

1. 정규화의 종류와 차이 3. 비교

비교

특징	Batch Normalization	Layer Normalization
계산 범위	배치N 내 동일 채널C	단일 데이터n 내 모든 채널C, H, W
배치 크기 의존성	매우 높음	없음 (독립적)
주요 사용처	CNN (컴퓨터 비전)	RNN, 트랜스포머 (자연어 처리)

1. 정규화의 종류와 차이 norm?

Norm?

벡터 공간에서 벡터 v의 크기 또는 길이를 측정하는 함수 $||v||$ 이다.

L_p-Norm: $||x||_p = (\sum_{i=1}^{n}|x_i|^p)^{1/p}$

p 값에 따라 벡터 공간에서 두 점 사이의 거리를 다르게 정의하며,
머신러닝에서 규제(Regularization)의 핵심 개념으로 사용된다

벡터 공간에서 원점(0, 0, ...)에서 특정 지점(벡터 $\mathbf{x}$)까지 얼마나 떨어져 있는가를 측정하는 방법,
또는 한 지점(벡터 $\mathbf{x}$)에서 다른 지점(벡터 $\mathbf{y}$)까지 얼마나 떨어져 있는가 를 재는 척도($||x-y||_p$)

L_p-Norm은 이 거리를 재는 다양한 방식을 공식화한 것. p 값에 따라 거리를 재는 방식이 달라진다

정규화 기법(L1 정규화, L2 정규화)에서 가중치 벡터의 크기에 페널티를 부여하여 과적합을 방지하는 데 사용
또한, 손실 함수(Loss Function)에서 예측 오차의 크기를 측정할 때도 사용

1. 정규화의 종류와 차이 norm?

Norm?

p=1 (L1 Norm): $\left|\left|x\right|\right|_1 = \sum_{i=1}^{n}|x_i|$

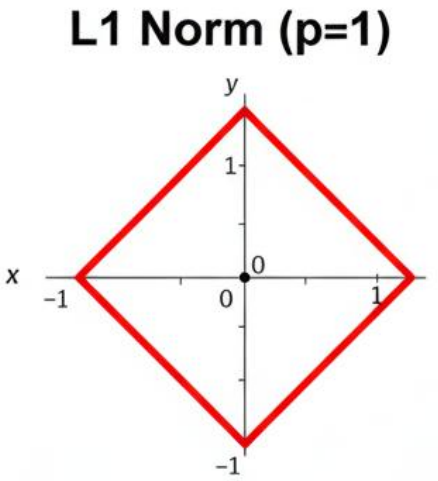
각 원소의 절댓값을 모두 더한 값. 맨해튼 거리(Manhattan Distance)라고도 부른다.

$||x||\_1 = \sum|x\_i| = |x\_1| + |x\_2| + \ldots + |x\_n|$

도시의 격자무늬 길(블록)을 따라 이동하는 것

원점 (0,0)에서 (3,4)까지 L1 거리는?
가로로 3블록|3|, 세로로 4블록|4| 이동해야 한다.
총 거리는 3 + 4 = 7 이다.

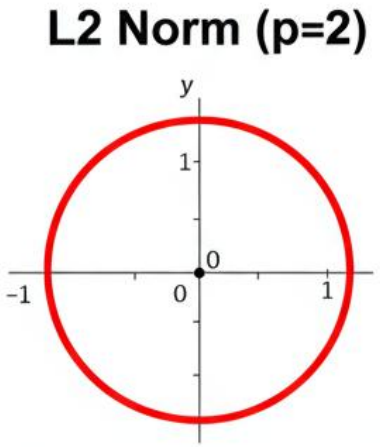
원점으로부터 L1 거리가 1인 점들의 집합은 마름모 형태를 띈다



LASSO (라쏘) 회귀에 사용된다.
특징(Feature): 중요하지 않은 변수(특징)의 가중치를 0으로 만드는 경향이 있어,
특징 선택(Feature Selection)에 유용하다.
이상치(outlier)에 덜 민감하다.

1. 정규화의 종류와 차이 norm?

Norm?



p=2 (L2 Norm): $\left|\left|x\right|\right|_2 = \left(\sum_{i=1}^{n}|x_i|^2\right)^{1/2} = \sqrt{\sum_{i=1}^{n} x_i^2}$

우리가 흔히 아는 유클리드 거리와 같다.

$$\left|\left|x\right|\right|_2 = \sqrt{\sum x_i^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

우리가 흔히 아는 두 점 사이의 직선 거리이다. 각 원소를 제곱해서 더한 뒤, 전체에 제곱근을 씌운다.

원점 (0,0)에서 (3,4)까지 L2 거리는?
피타고라스의 정리를 생각하면 된다.
총 거리는 $\sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$

원점으로부터 L2 거리가 1인 점들의 집합은 우리가 아는 원 형태를 띈다
Ridge (릿지) 회귀에 사용된다.
특징(Feature): 가중치 값을 0에 가깝게 줄이기는 하지만 0으로 만들지는 않는 경향이 있다.
모델을 전반적으로 부드럽게(Overfitting 방지) 만든다.
제곱을 하기 때문에 이상치(outlier)에 L1보다 민감하게 반응한다.

1. 정규화의 종류와 차이 norm?

Norm?

$p = \infty$ (L-infinity Norm): $||x||_\infty = \max_i(|x_i|)$

벡터의 원소 중 절댓값이 가장 큰 값이다. Maximum Norm 이라고도 한다.

여러 차원(방향)으로 이동할 때, 그중 가장 많이 이동한 단일 차원의 거리만 본다

벡터 (3, -4, 1)의 L-infinity Norm은?

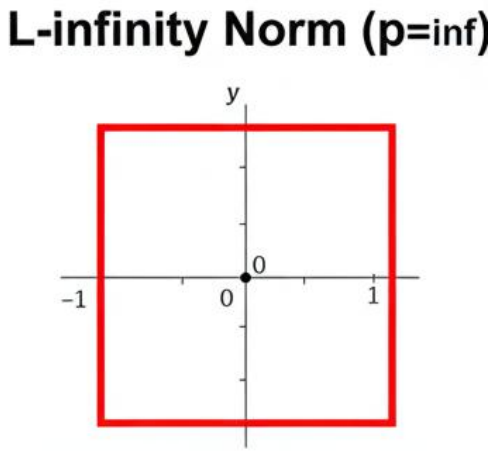
각 원소의 절댓값은 $|3|, |-4|, |1|$이다.

이 중 가장 큰 값은 4이다.

원점으로부터 L-infinity 거리가 1인 점들의 집합은 정사각형 형태를 띈다

모델의 최악의 오류(worst-case error)를 최소화하고자 할 때 사용된다.

여러 값 중 가장 크게 튀는 값을 제어할 때 유용하다.



1. 정규화의 종류와 차이 norm?

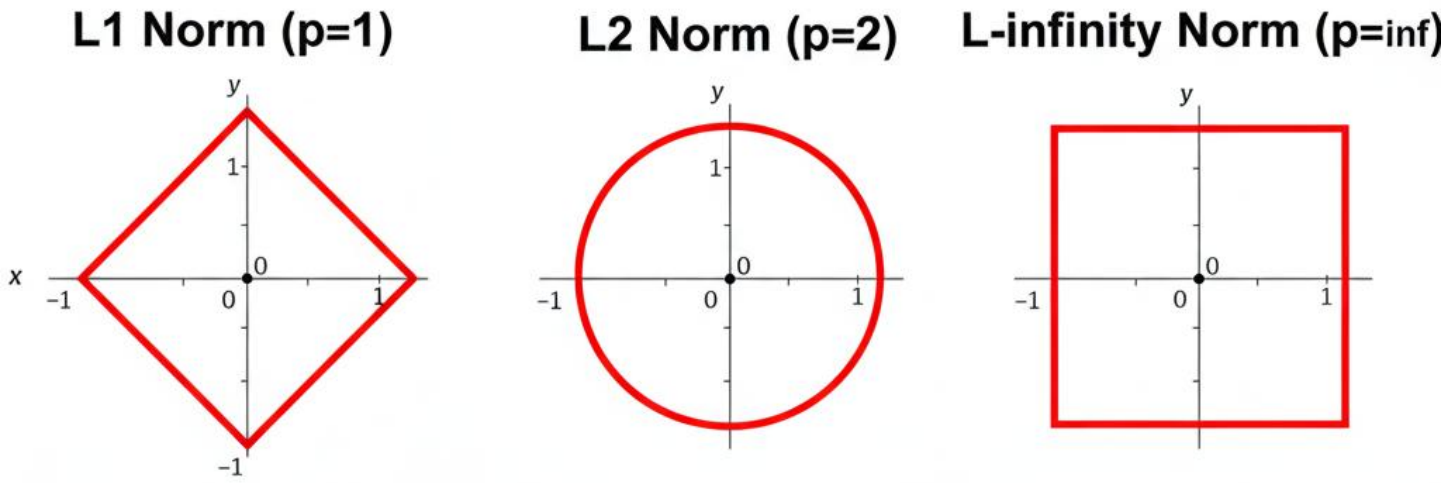
Norm?

원점으로부터 L1? 이게 뭔 소리고 왜 중요한지

$p = 1$ (L1 Norm / 맨해튼 거리)
$||(x, y)||_1 = |x| + |y| = 1$

$p = 2$ (L2 Norm / 유클리드 거리)
$||(x, y)||_2 = \sqrt{x^2 + y^2} = 1$

$p = \infty$ (L-infinity Norm)



모델이 얼마나 구린가 또는 모델이 얼마나 복잡한가를 측정하는 척도 $||(x, y)||_\infty = \max(|x|, |y|) = 1$

규제를 통한 정규화가 궁금하다면? : https://gihak111.github.io/ai/2025/11/05/L1_L1_Norm_upload.html

1. 정규화의 종류와 차이 비교

ONECLICK AI

구분	L1 Norm	L2 Norm	Batch Normalization	Layer Normalization
범주	수학적 Norm	수학적 Norm	Normalization	Normalization
계산 방식	벡터 요소 절댓값의 합 $\sum$	x_i		
적용 대상	주로 가중치	주로 가중치	활성화 함수 입력	활성화 함수 입력
주요 목적	Lasso 정규화에 사용. 중요하지 않은 가중치를 0으로 만들어 특징 선택 효과.	Ridge 정규화에 사용. 모든 가중치를 0에 가깝게 줄여 과적합 방지	훈련 속도를 높이고, 내부 공변량 변화를 줄여 훈련 안정화.	RNN/Transformer 등 시퀀스 모델 및 작은 배치 사이즈에서 훈련 안정화.
특징	결과 벡터가 희소해짐.	모든 가중치에 균등하게 페널티 부여.	배치 크기에 의존적. 배치 크기가 작으면 성능 저하.	배치 크기에 독립적.

https://arxiv.org/abs/1502.03167

https://arxiv.org/abs/1607.06450

2. U-Net 알아보기 어떻게 생겼을까?

U-Net?

이미지 분할을 위한 모델 FCN(Fully Convolutional Network)
대칭적인 U자형 구조와 스킵 연결(Skip Connections)을 핵심

인코더 (수축 경로)

디코더 (확장 경로)

스킵 연결 (인코더, 디코더를 연결)

https://arxiv.org/abs/1505.04597

2. U-Net 알아보기 어떻게 생겼을까?

U-Net?

이미지 분할을 위한 모델 FCN(Fully Convolutional Network)
이미지의 모든 픽셀이 어디에 속하는지(종양, 배경, 혈관) 그런거 판단 할 수 있다.
대칭적인 U자형 구조와 스킵 연결(Skip Connections)을 핵심

인코더 (수축 경로) 이미지가 무엇인지 파악

디코더 (확장 경로) 이미지가 어디에 있는지 파악, 스킵 연결

최종 출력

어디서 본 구조죠 Yolo neck 이랑 비슷한 느낌 나오죠

2. U-Net 알아보기 Encoder?

Encoder

입력 이미지에서 특징(feature)을 추출하고, 이미지의 전반적인 문맥(Context)을 파악

1. 반복 블록: (3x3 Convolution + ReLU) * 2회
2. 다운샘플링: (2x2 Max Pooling)

그냥 CNN이죠?

2. U-Net 알아보기 Encoder?

Encoder

Conv Layer

입력텐서, 필터를 다음과 같이 하고, S = 1, P = 0 이라 하자

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \qquad K = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

1. 첫 번째 위치 연산

$$\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = (1 \cdot 1) + (2 \cdot 0) + (3 \cdot 1) + (5 \cdot 0) + (6 \cdot 1) \quad \dots = 30$$

스트라이크가 1 이므로, 한 칸 이동해서 연산

$$\begin{pmatrix} 2 & 3 & 4 \\ 6 & 7 & 8 \\ 10 & 11 & 12 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = (2 \cdot 1) + (3 \cdot 0) + (4 \cdot 1) + (6 \cdot 0) \quad \dots = 35$$

반복해서 얻은 최종 결과물

$$B = \begin{matrix} [[30, 35], \\ [50, 55]] \end{matrix}$$

2. U-Net 알아보기 Encoder?

Encoder

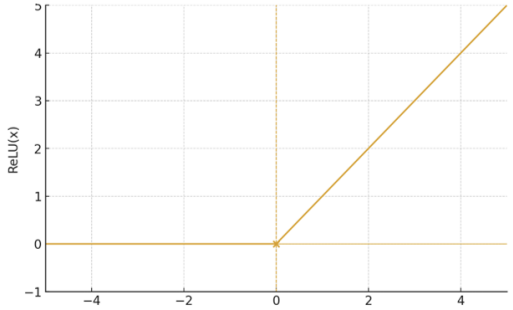
RelULayer

✓ **RelULayer** **독립적인 계층으로 모델에 추가될 수 있기 때문에 활성화 함수임에도 불구하고 레이어 이다.**

활성화 함수 중 하나

음수값은 0으로, 양수값은 그대로 유지

역전파가 존재



두 개의 선형이 합쳐져 비선형성을 가진다

비 선형성

여러 개의 ReLU 레이어를 쌓으면

복잡한 패턴을 학습 가능

계산 효율성

복잡한 뎃셈, 뺄셈 없고 단순히 0과 입력값을 비교하는게 전부라 매우 빠르다

기울기 소실 문제 완화

시그모이드는 입력값이 너무 크거나 작을 때 미분값이 0에 가까워 지는 기울기 소실이 발생. 하지만, ReLU는 양수 영역에서 항상 미분값이 1 이므로, 역전파에서 매우 안정

2. U-Net 알아보기 Encoder?

Encoder

Maxpooling Layer

예시를 통해 알아보자

4 * 4 인 입력텐서 A $A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$ 최종출력텐서 => $B = \begin{pmatrix} 6 & 8 \\ 14 & 16 \end{pmatrix}$

F = 2

S = 2

이면, 다음과 같은 4 범위가 된다

1 => $A_{0:2,0:2} = \begin{pmatrix} 1 & 2 \\ 5 & 6 \end{pmatrix}$ 2 => $A_{0:2,2:4} = \begin{pmatrix} 3 & 4 \\ 7 & 8 \end{pmatrix}$

3 => $A_{2:4,0:2} = \begin{pmatrix} 9 & 10 \\ 13 & 14 \end{pmatrix}$ 4 => $A_{2:4,2:4} = \begin{pmatrix} 11 & 12 \\ 15 & 16 \end{pmatrix}$

2. U-Net 알아보기 Encoder?

Encoder

1. 3x3 Convolution
2. ReLU
3. 3x3 Convolution
4. ReLU
5. 2x2 Max Pooling

2. U-Net 알아보기 Decoder?

Decoder

압축된 특징 맵을 바탕으로 픽셀 단위의 정확한 위치 정보를 복원

1. 업샘플링: (2x2 Up-Convolution 또는 Transposed Convolution)
2. Skip Connection 결합: (아래 3번 항목 참조)
3. 반복 블록: (3x3 Convolution + ReLU) * 2회

2. U-Net 알아보기 Decoder?

Decoder

2x2 Up-Convolution 전치 컨볼루션 사용(Transposed Convolution)

특징 맵의 크기(해상도)를 2배로 늘리고, 채널 수는 반으로 줄이는 역할

입력 특징 맵 (I): 2×2 크기 (채널 1개 가정)

커널 (K): 2×2

스트라이드 S: 1

2. U-Net 알아보기 Decoder?

Decoder

2x2 Up-Convolution

$I = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ $K = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

출력 크기는

$O = (I - 1) \times S + K = (2 - 1) \times 1 + 2 = 3$

따라서, 3 * 3 이 된다

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ $\begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$

$\begin{pmatrix} 3 & 6 \\ 9 & 12 \end{pmatrix}$ $\begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}$

2. U-Net 알아보기 Decoder?

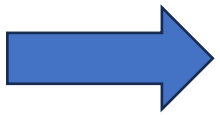
Decoder

2x2 Up-Convolution

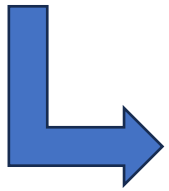
그런데, 스트라이드가 1 이므로,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

$$\begin{pmatrix} 3 & 6 \\ 9 & 12 \end{pmatrix} \qquad \begin{pmatrix} 4 & 8 \\ 12 & 16 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 2+2 & 4 \\ 3+3 & 4+6+6+4 & 8+8 \\ 9 & 12+12 & 16 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 4 & 4 \\ 6 & 20 & 16 \\ 9 & 24 & 16 \end{pmatrix}$$

따라서, 3 * 3 의 크기를 가지게 된다

2. U-Net 알아보기 Decoder?

Decoder

Skip Connection

인코더의 특징 맵을 디코더의 대응하는 레이어로 직접 복사하여 연결하는 구조

인코더는 컨볼루션과 풀링을 거치며 이미지의 무엇인지 정보는 잘 학습하지만, 어디인지 정보는 손실

디코더가 업샘플링을 통해 해상도를 복원할 때, 인코더에서 손실되었던 정확한 픽셀 레벨의 위치 정보를 Skip Connection을 통해 다시 공급받는다

디코더는 업샘플링된 흐릿한 문맥 정보와 인코더의 선명한 위치 정보를 결합하여, 정교한 Segmentation Mask를 생성할 수 있다

2. U-Net 알아보기 Decoder?

Decoder

Skip Connection

1. 인코더 특징 맵
   폴링 되기 전의 고해상도 특징 맵
   예시 : 64 x 64 x 128
   (높이x너비x채널)

2. 디코더 특징 맵
   Up-Convolution을 거친 후의 특징 맵
   예시 : 64 x 64 x 128

두 특징 맵을 채널 방향으로 결합
64 x 64 x (128 + 128) = 64 x 64 x 256

+ U-Net 원본 논문에서는 Padding을 사용하지 않아 인코더/디코더 맵의 크기가 약간 다를 수 있어,
  중앙을 기준으로 Crop 하여 크기를 맞춘 후 연결

2. U-Net 알아보기 Decoder?

Decoder

반복 블록 (3x3 Convolution + ReLU) * 2회

결합된(Concatenated) 특징 맵을 입력받아 특징을 정제하고 학습

Up-Convolution을 통해 전달된 문맥정보와
Skip Connection으로 복사된 위치정보를
이 3x3 Conv 연산을 통해 본격적으로 융합하여 의미 있는 새로운 특징을 추출

3x3 커널을 2회 연속 적용하여, 해당 해상도(Resolution) 레벨에서 더 복잡하고 정교한 패턴을 학습할 수 있다

2. U-Net 알아보기 Decoder?

Decoder

1. 2x2 Up-Convolution

2. Skip Connection

3. 3x3 Convolution

4. ReLU

5. 3x3 Convolution

6. ReLU

디코더는 원본 이미지와 동일한 해상도가 될 때 까지 반복한다

2. U-Net 알아보기 최종출력?

최종출력

이전 단계(마지막 디코더 블록)까지의 출력은 [높이 x 너비 x 64]처럼 복잡한 특징정보를 담고 있다

1x1 컨볼루션을 통해 이 64개의 특징을 조합하여, 각 픽셀이 어떤 클래스에 속하는지를
나타내는 최종 클래스 맵(Class Map)으로 변환한다

1. 1x1 Convolution
2. Softmax

2. U-Net 알아보기 최종출력?

최종출력

1x1 Convolution

이미지의 높이(H)와 너비(W)는 변경하지 않고, 채널(Channel) 수만 조절

셀 하나하나를 보면서, 64개의 채널 값을 입력받아 우리가 원하는 K개의 채널 값으로 압축(요약)

이진 분할

입력: [512 x 512 x 64]

출력: [512 x 512 x 1]

다중 클래스 분할 ex) 3

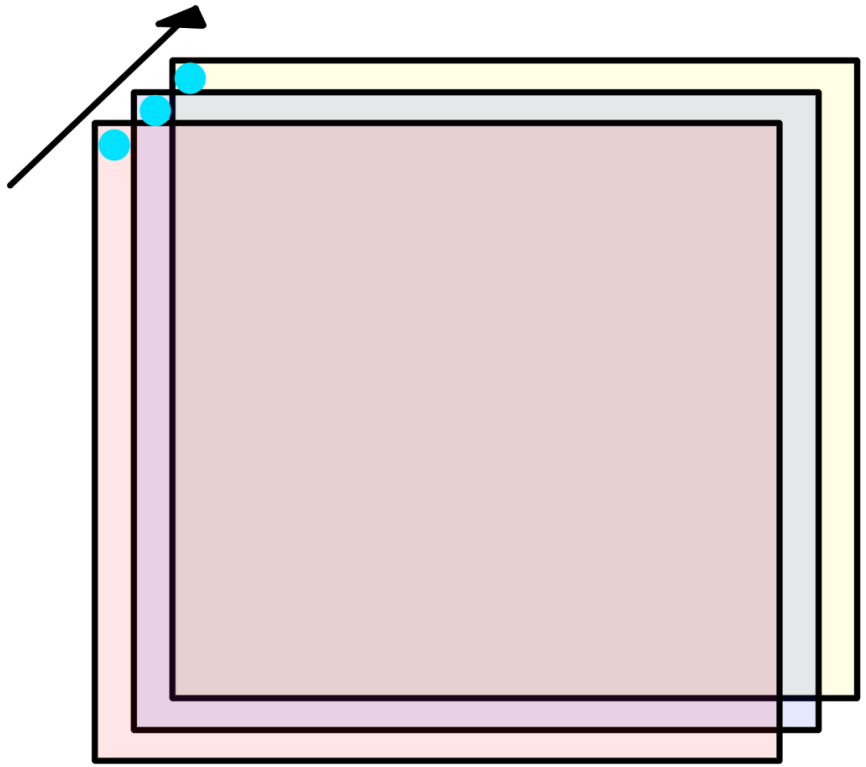
입력: [512 x 512 x 64]

출력: [512 x 512 x 3]

Softmax(다중)이나 Sigmoid(이진)을 사용해서 최종 출력을 계산

2. U-Net 알아보기 최종출력?

최종출력

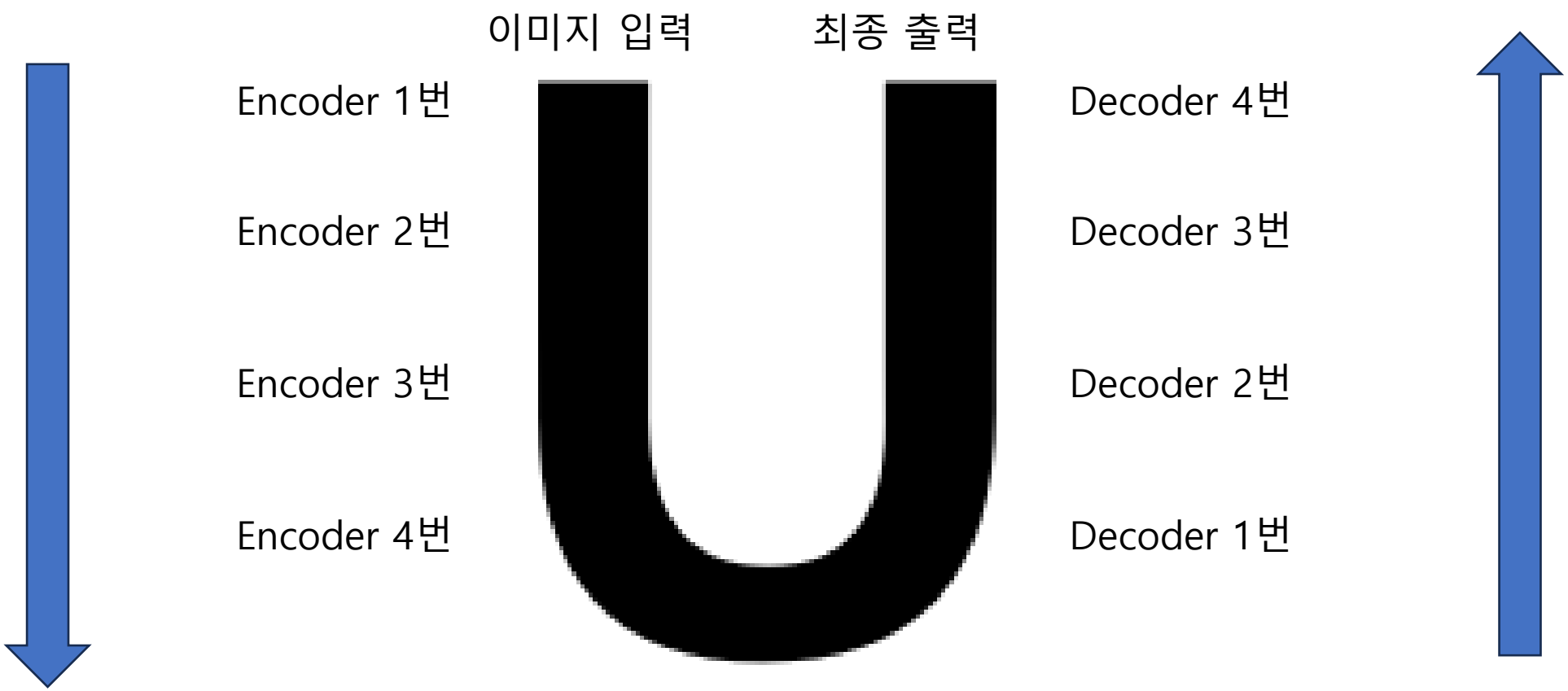


저 파란 점 찍힌 부분 을 컨볼루션을 통해서 하나로 합친다(압축)

그 과정에서 각 채널에 가중치를 곱한 후 합치게 된다

결국 모든 특징이 하나로 합쳐진 하나의 채널만 남는다

2. U-Net 알아보기 종합



바닥 Bottleneck (conv * 2)

더 복잡하고 의미 있는 고차원 특징을 학습하기 위함

2. U-Net 알아보기 U-Net? Yolo?

U-Net과 Yolo는 어떻게 다른가

특징을 추출(Encoding)하고 다시 특징을 조합(Decoding/Fusing)한다는 점에서 굉장히 유사

U-Net (Segmentation):
입력과 똑같은 크기의 픽셀 단위 지도(Mask)를 만들기 위해 U자 구조로 잃어버린 위치 정보를 완벽하게 복원한다.

YOLO (Detection):
객체를 잘 찾기 위해 U자 구조(Neck)로 다양한 크기(Multi-scale)의 특징을 융합(Fusion)하고, 헤드가 이 융합된 정보에서 좌표(Box)를 예측한다.

2. U-Net 알아보기 U-Net? Yolo?

	YOLO (Object Detection)		U-Net (Segmentation)
백본 (Backbone)	이미지에서 특징 추출 (압축)	인코더 (Encoder)	이미지에서 특징 (Context)을 추출 (압축)
넥(Neck)	특징 융합 다양한 크기의 객체를 잘 감지하기 위해, 고해상도 특징(위치)과 저해상도 특징(문맥)을 섞어 여러 스케일의 맵을만든다.	디코더(Decoder)	공간 복원 픽셀 단위의 정확한 위치를 복원하기 위해, 업샘플링된특징에 인코더의 특징을 그대로 가져와 붙인다
헤드 (Head)	예측기 융합된 특징 맵을 보고 객체의 존재 여부, 좌표, 클래스를 예측	최종출력	픽셀 분류기 복원된 최종 특징맵의모든 픽셀을분류

2. U-Net 알아보기 Encoder, Decoder

Encoder, Decoder

Encoder : 전체의 내용이나 특정 부분의 의미를 이해

Decoder : 어진 조건(시작 픽셀, 텍스트 프롬프트)으로부터 새로운 이미지를 생성

Encoder + Decoder : 입력 내용을 조건에 맞게 변형

구분	Encoder - Decoder	Encoder Only	Decoder Only
핵심 역할	시각적 변환 / 설명	시각적 이해 / 분류	시각적 생성
주요 작업	이미지 캡셔닝, 스타일 전이	이미지 분류, 객체 감지, 분할	텍스트-투-이미지 생성
입력	이미지	이미지	텍스트 프롬프트 + 시작 이미지/픽셀
출력	텍스트 또는 다른 이미지	레이블, 바운딩 박스, 마스크	새로운 이미지
대표 모델	U-Net, Autoencoder	CNN, YOLO	Diffusion

2. U-Net 알아보기 Encoder, Decoder

Encoder, Decoder

구분	Encoder - Decoder	Encoder Only	Decoder Only
핵심 역할	변형 (Seq2Seq)	이해 (NLU)	생성 (NLG)
주요 작업	번역, 요약	분류, 개체명 인식	챗봇, 글쓰기
입력	양방향 + 단방향	양방향	단방향 (순차적)
출력	텍스트	텍스트	텍스트
대표 모델	T5, BART	BERT, RoBERTa	GPT, LLaMA

모델 코드

https://huggingface.co/gihakkk/U-Net

감사합니다