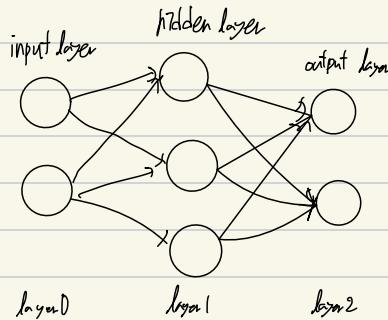


Neural network



Matrix - Inner product $\Rightarrow \text{np. dot}(\text{matrixA}, \text{matrixB})$

$$i \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7, 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7, 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$A \times B = C$$

$$aR \times aC = bR \times bC$$

Implement 3-layer neural network

from prob perceptron

$$\begin{array}{c} (x) \\ \swarrow w_1 \\ (y) \\ \searrow w_2 \end{array} \Rightarrow \begin{cases} y = 0 & (b + w_1x_1 + w_2x_2 \leq 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases} \quad (b = \text{bias})$$

Perceptron specifying bias

$$\begin{array}{c} (1) \\ \swarrow b \\ (x_1) \xrightarrow{w_1} (y) \\ \searrow w_2 \\ (x_2) \end{array}$$

$$y = h(b + w_1x_1 + w_2x_2)$$

$$h(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases} = u(x)$$

activation function : 활성화 함수

$$\begin{array}{c} (1) \\ \swarrow b \\ (x_1) \xrightarrow{w_1} (a) \xrightarrow{h} (y) \\ \searrow w_2 \\ (x_2) \end{array}$$

$$a = b + w_1x_1 + w_2x_2$$

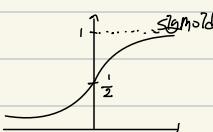
$$y = h(a)$$

+ activation function = non-linear

If linear: meaningless in multi-layer

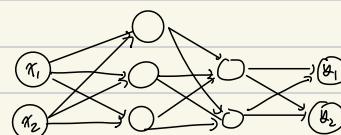
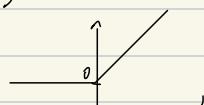
Sigmoid function

$$h(x) = \frac{1}{1 + e^{-x}}$$



ReLU (Rectified Linear Unit)

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$



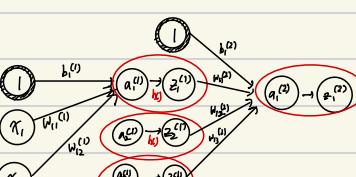
$$\begin{array}{c} (x_1) \\ \swarrow w_{11}^{(0)} \\ (x_2) \end{array} \xrightarrow{w_{12}^{(0)}} a_1^{(1)}$$

$$\xrightarrow{w_{13}^{(0)}} a_2^{(1)}$$

$$\begin{array}{c} (x_1) \\ \swarrow w_{11}^{(1)} \\ (x_2) \end{array} \xrightarrow{w_{12}^{(1)}} a_1^{(2)}$$

$$a_i^{(l)} = w_i^{(l)}x_i + w_{i+1}^{(l)}x_{i+1} + \dots + w_n^{(l)}x_n + b_i^{(l)}$$

equal with vector inner-product



output functions $g(x)$

$$a_i^{(l)} \rightarrow y_i$$

identity function (恒等함수)

$$a_i^{(l)} \rightarrow y_i$$

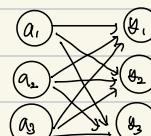
softmax function ($\frac{e^{a_k}}{\sum_j e^{a_j}}$)

$$y_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

a_i : k-th input

y_i : output layer node number

y_k : k-th output



Softmax 끝에 구현시 주의점 \Rightarrow overflow 문제

거듭제곱 사용 \Rightarrow 매우 큰 값 \Rightarrow 큰 값에서 나눗셈에서 끊임

$$\begin{aligned} \text{softmax } y_3 &= \frac{\exp(a_3)}{\sum_{i=1}^3 \exp(a_i)} = \frac{\exp(a_3)}{\sum_{i=1}^3 \exp(a_i + b_i)} \\ &= \frac{\exp(a_3 + b_3)}{\sum_{i=1}^3 \exp(a_i + b_i)} \\ &= \frac{\exp(a_3 + c)}{\sum_{i=1}^3 \exp(a_i + c)} \end{aligned}$$

$\Rightarrow C'$ 의 차원 수 대입하여도 결과 동일

\Rightarrow 일괄적으로 모델로 넣기 위해 정규화를 통한 확장된 이론

Softmax 확장의 특징

$$\text{softmax 확장 } y_i \Rightarrow 0 \leq y_i \leq 1.0$$

$$\frac{1}{\sum_i} y_i = 1$$

\Rightarrow Softmax의 출력 = 확률

Machine learning : 디지털 속 규칙을 쓰고 찾아내는 과정

Flatten

$$\left[\begin{array}{c} a_{11} a_{12} a_{13} \\ a_{21} a_{22} a_{23} \\ a_{31} a_{32} a_{33} \end{array} \right] \in M_{3,3} \rightarrow \left[\begin{array}{c} a_{11} a_{12} a_{13} \\ a_{21} a_{22} a_{23} \\ a_{31} a_{32} a_{33} \end{array} \right] \in \mathbb{R}^9$$

MNIST 이미지 차원 28x28 Matrix Flatten 784 pixels

$$\begin{matrix} 784 \text{ node} & \longrightarrow 9 \text{ node softmax} \\ 0 & ① \\ 0 & ② \\ \vdots & \vdots \\ 0 & ⑨ \end{matrix}$$

9개 노드가 결과 분포

one-hot encoding $\Rightarrow [0, 0, 0, 1, 0, 0, 0]$

정답에 있는 원소 = 1 하나의 원소에만 1이 정답인 원소의 (hot)

+ machine learning's 2 step : 학습, 추론 중

학습 단계에서는 훈련용 데이터 사용

추론 단계에서는 예측,

\Rightarrow 학습 적용 시 학습으로 대체되는 훈련X 예측

Normalizaton 정규화 일정 범위 (0~255)를 (0, 1.0) 범위로 변환

pre-processing 단계 신경망의 입력데이터에 특별 변화를 주는 것

훈련데이터 개수

능력 \Rightarrow 훈련하고 싶은 데이터 수

Batch : 일괄적으로 처리되는 집단 = 학습문

= 데이터를 처리하는 기본 단위

two steps solving problem with neural network

Batch size $\uparrow \Rightarrow$ memory cost $\uparrow \Rightarrow$ speed \uparrow

Training 단계 \Rightarrow weight parameter update based on data

Batch processing 단계 처리

Inference 단계 \Rightarrow classify input data from trained model.

전체 데이터 세트를 작은 배치들로 나누고 각 배치를 독립적으로 처리

forward propagation 순방향 전파

단계 처리 시 \rightarrow 개인적 처리로 사용

- 입력층에서 출력층까지 순차적으로

\rightarrow 병렬 처리 가능

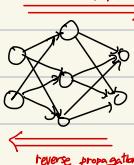
forward propagation

\rightarrow 반복적인 업데이트 (각 배치마다 그레이드를 사용)

reverse propagation 역방향 전파

\Rightarrow 개별 미분(gradient) 계산

- loss (loss)를 기준으로 기울기 (gradient) 계산



Batch Normalization 레이어의 초기 or 끝

- Min-Max Normalization 레이어의 최솟값 0, 최댓값 1
- Z-score Normalization (Standardization) 평균 0 표준편차 1

N개의 데이터 모두에 대한 고차원트리의 예

$$E = -\frac{1}{N} \sum_{k=1}^N t_{nk} \log t_{nk}, t_{nk} = n^{th} \text{ Data's } k^{th} \text{ value}$$

Mini-Batch Training

1. 각 배치의 평균 및 분산을 계산

레이어의 값을 풀어 현재 단계의 근사치로 사용

2. 미세분을 정규화 (각 배치 데이터에서 평균과 분산으로 나눈 Z-score)

3. 정규화된 데이터를 스케일링 및 시프트 ($y_{(batch)} = f(x_{(batch)})$ 로 하고 이를)

($y_{(batch)}$)과는 평균과 분산, 학습 단계에서 평균)

온라인식의 사용 이유

- 손실함수 \Rightarrow 학습률을 끌어내는 미세한

- why? 정규화가 아닌 온라인식은 학습률을 조절하는가

\Rightarrow 학습률이 '기본'

선행 학습에서 차별의 미세한 수 디렉 \Rightarrow 온라인식 학습을 통해 학습하는 미세한 힘의

\Rightarrow 미세한의 평균(기울기) 계산 \Rightarrow 미세한 학습(평균 공부 \Rightarrow 학습 방법 변화)

기존 미세한식을 써 줄 때마다 시스템이 어떻게 변화하는가

Train 학습 : 학습 데이터로부터 기울기 미세한수의 최적값을 기울기로 찾는다

Machine Learning = Feature는 사실이 크게

Deep learning = Feature의 규모 작아

정도 차이가 대체로 미세한 \Rightarrow 미세한 정도 불가

Machine Learning \Rightarrow Data \rightarrow Training Data = 초기 미세한수 찾는
 \rightarrow Test Data = 모델 평가

Coefficient : 한 데이터에 따른 차별의 최적화된 SEM

경사하강법 Gradient Descent Method

$$x_0 = x_0 - \eta \frac{\partial f}{\partial x_0} \quad (\eta = \text{학습률} = \text{learning rate}, \text{학습률})$$

$$x_i = x_i - \eta \frac{\partial f}{\partial x_i}$$

Loss Function : 선형방정식의 대푯값을 나타내는 기준

= 예측과 실제값 사이에 흐름하는 힘

Hyper Parameter 초미세한수

사람의 직업 결정에 따른 미세한수 or learning rate

Sum of Squares for Error · SSE (교차 제곱합)

$$E = \frac{1}{2} \sum_k (y_k - \hat{y}_k)^2, \text{ 즉: 선형방정식 흐름(제곱합)}$$

\hat{y}_k : 정답 대푯값

선형방정식의 기울기

k : 레이어 차원수

기울기 W , 흐름 $L \Rightarrow$ 기울기 $= -\frac{\partial L}{\partial W}$

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad \frac{\partial L}{\partial W} = \frac{\partial L}{\partial w_{11}} \quad \frac{\partial L}{\partial w_{12}} \quad \frac{\partial L}{\partial w_{13}}$$

Cross Entropy Error: CEE (교차 엔트로피 예)

$$E = -\frac{1}{n} \sum_k y_k \log \hat{y}_k \quad (k = 1, 2, \dots, n)$$

학습 알고리즘 구현

1. 미니 배치 \rightarrow 미니 배치 선형 \rightarrow 최적화 경제법 SGD

2. 경계 선형

3. 미지수 경계

4. 1~3 단계 반복

경계화 학습 계층 구현

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \Rightarrow \frac{\partial f}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$



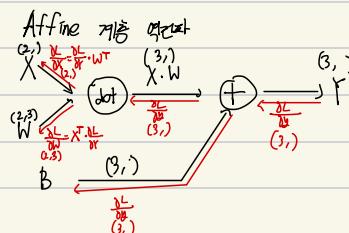
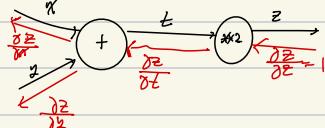
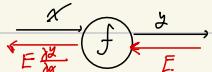
Epoch : 훈련 데이터를 봄는 횟수?

Sigmoid 계층 구현

오차역전파법 back propagation

계산 그래프를 통해 계산 \Rightarrow 훈련 데이터를 통해 맨 첫 번째 계층 계산 가능

$$y = \frac{1}{1 + e^{-x}} \Rightarrow \frac{\partial y}{\partial x} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial x} = \frac{\partial L}{\partial y} \cdot y(1-y)$$



곱셈 노드의 역전파

$$z = x_1 y_1$$

$$\Rightarrow \frac{\partial z}{\partial x_1} = y_1, \frac{\partial z}{\partial y_1} = x_1$$

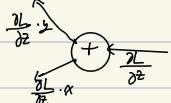


\Rightarrow 훈련 때 그대로 전달

곱셈 노드의 역전파

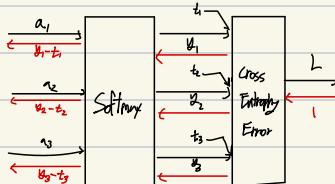
$$z = \alpha y$$

$$\Rightarrow \frac{\partial z}{\partial x} = y, \frac{\partial z}{\partial y} = \alpha$$



\Rightarrow 서로 다른 학습률에 맞게

Sigmoid-with-Loss 계층



Softmax와 연관화

1. 미니 배치

2. 경계 선형

3. 미지수 경계

4. 1~3 단계 반복

Optimization 최적화

매개변수의 최적값을 찾는 행위

$$RMSPROP \quad h_i = \rho h_{i-1} + (1-\rho) \frac{\partial L_i}{\partial W} \odot \frac{\partial L_i}{\partial W}$$

$$W = W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

SGD 속도가 정지하는 법

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

직접 W 에 대한 흡수를 계산해

SGD의 단점

별도 매개변수에 영향을 주기 위해 반복적 계산을 해야 한다.

\Rightarrow anisotropy (방향성, 방향에 따른 성과(경기)가 달라지는) gradient에서 영향을

Adain = Momentum + RMSPROP = 진행하는 속도에 관계 + 즉응적 최적화

hyperparameter : η , 학습률 계수 ρ , 초기 모멘텀 계수 ρ_0

$$\theta_t = \theta_{t-1} - \eta \frac{m_t}{\sqrt{v_t + \epsilon}}$$

momentum 방법
rmprop 방법

$$t \leftarrow t+1$$

Momentum - 속도, 솔루션 관계

$$V \leftarrow \alpha V - \eta \frac{\partial L}{\partial W}$$

$$W \leftarrow W + V, \quad V = \text{velocity}$$

αV = 블록이 이동한 거리를 먼저 산출할 때 사용되는 계산법 (마찰, 저항)

Weight Decay Technic

가장자 매개변수의 값이 커지도록 차단 \Rightarrow Overfitting ↓

• 초기값을 모두 같은 값으로 설정하면 안되는 이유

\rightarrow 초기화된 값에서 값이 뚱뚱이 걸친

AdaGrad - learning rate decay technic

↳ 차임을 진행하면서 학습률은 점차 줄어듭니다

↳ 매개변수 전체의 학습률 값을 일정적으로 산출

↓ 발견

AdaGrad - 개별 매개변수에 적응적으로 학습률 조정

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$

↳ 개별 매개변수 학습률 조정

$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

\hookrightarrow 매개변수 학습률 많이 조정된 경우 \Rightarrow 학습률 ↓ 조정

단점 - 과거의 기울기를 계산하는데 계산량

\Rightarrow 계산량 ↑ \Rightarrow 경신 정도 ↑

\Rightarrow 계산량 ↑에 신경

개선 \Rightarrow RMSPROP

↳ 과거의 모든 경과 계산 보관

\Rightarrow 과거의 기울기들을 평균내, 최근 변화가

\Rightarrow Exponential Moving Average, EMA, 지수평활화

Xavier 초기화

• 초기값은 표준편차 $\frac{1}{\sqrt{n}}$ 의 표준 편차, 즉 표준

• but 출판과 학습 속도를 고려 (sigmoid or tanh)

(~~표준~~)에서는 다른 초기화 필요

He 초기화

• 초기 \neq 노드 수 1일 때 표준偏差 $\frac{2}{\sqrt{n}}$ 확장 적용

ReLU는 초기 편차 0 \Rightarrow 더 높게 초기화해야?

Batch Normalization

각 층이 활성화를 적당히 제한하도록 강제화

방법 - 활성화 개선, 오차율 하락, 모델학습 빠름 (드롭아웃 적용방법)

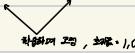
학습시 미니배치 단위로 계산 (평균, 표준)

$$\mu_b \leftarrow \frac{1}{m} \sum_{j=1}^m \mathbf{x}_j$$

$$\sigma_b^2 \leftarrow \frac{1}{m} \sum_{j=1}^m (\mathbf{x}_j - \mu_b)^2$$

$$\hat{\mathbf{x}}_j \leftarrow \frac{\mathbf{x}_j - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}}, \quad \epsilon = 10^{-7}, \text{ prevent zero division}$$

$$y_j \leftarrow \gamma \hat{\mathbf{x}}_j + \beta \quad (\gamma = \text{scaling}, \beta = \text{shifting})$$



기아리 파라미터의 학습률 담수

> 훈련 데이터에서 계산된 파라미터 평균을 데이터 삽입 (Validation Data)

데이터에서 흔적 - 훈련 데이터 : 미개방식 학습

- 검증 데이터 : 키아리 파라미터 성능 평가

- 시험 데이터 : 신경망의 법론 성능 평가

기아리 파라미터 초기화

- 초기값이 존재하는 범위 조건식 활용

⇒ 0. 기아리 파라미터 값 범위 설정

1. 설정된 범위에서 기아리 파라미터 값 무작위 추출
2. 학습 후 검증 데이터로 정확도 평가 (epoch 각계)
3. 1, 2 단계 동일 확률 분포 후 범위 조정

Overfitting

사용 - 1. 미개방식 알고리즘에서 훈련에 활용

- 2. 훈련 데이터 학습

⇒ weight decay로 많이 사용

다른 방법 = Bayesian Optimization

(Practical Bayesian Optimization of Machine Learning)

논문 참고

Dropout

능력을 얻으려면 오버피팅 학습

훈련 중은 모듈의 노드를 무작위로 절연 안전한 파라미터 학습

시험 때는 모든 노드에 모두 연결, 각 노드의 출력에 훈련에 속해 있는 노드 훈련

Ensemble Learning

개별적으로 학습한 각 모델의 출력을 결합하여 구현

Dropout과 유사함. (예전 유제와 같이 = 예전 퀘션처럼)

(즉슨 예전 퀘션 ≠ 예전 유제처럼)

Convolutional Neural Network, CNN

Convolution Layer, Pooling Layer 사용

기본 계층 : Affine - ReLU

CNN 계층 : Conv - ReLU - (Pooling), Affin-ReLU (결과에 추가로 계층)

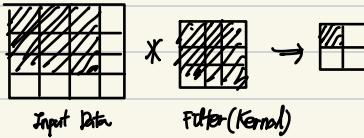
기존 학습모델 계층의 문제

→ 데이터의 차원 유지 ($(1, 28, 28)$ 입력 $\rightarrow (704,)$)

3D Data \rightarrow 2D Data 사용 \rightarrow 3D Data

Feature Map : 학습 계층의 결과물 예시

Convolution



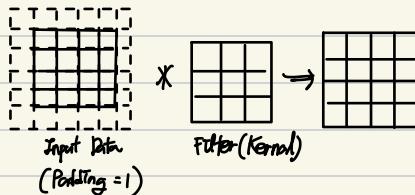
Filter의 Window를 유행 대로 이용하기

연결 합성-추가 (Fixed Multiply-Add, FMA)

$$\begin{array}{c} \text{---} \\ \text{Input Data} \\ \text{---} \end{array} + \begin{array}{c} \text{---} \\ \text{bias} \\ \text{---} \end{array} \Rightarrow \begin{array}{c} \text{---} \\ \text{Output Data} \\ \text{---} \end{array}$$

Padding

Convolution \Rightarrow Input Data 차원을 통일 $\Rightarrow (0) \text{으로 처리}$



Stride

필터를 적용하는 위치의 간격

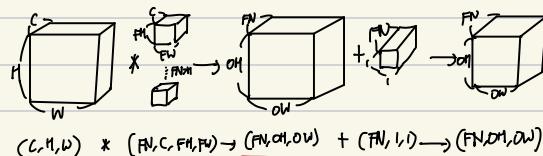
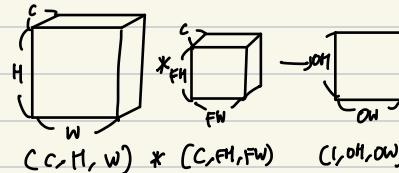
Input Size (M, W) Filter Size (FH, FW) Output Size (OH, OW)

Padding (P) Stride(S)

$$OH = \frac{M+2P-FH}{S} + 1, OW = \frac{W+2P-FW}{S} + 1$$

3D Conv

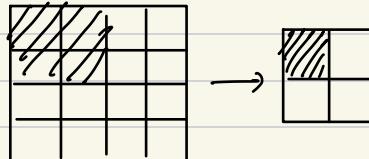
입력 데이터의 차원수와 필터의 차원수 같아야 함



NumPy Broadcast 기법

Pooling

MB, 카테고리 별 차이의 평균을 계산



pooling \Leftrightarrow max pooling = 최대값이 대표
average pooling = 평균값이 대표

$n \times n$ pooling \rightarrow stride=1 혹은 pooling size

feature - no training parameter

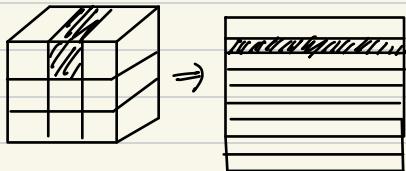
- no change of dimension (channel)

- 결과물 (batch) 크기를 줄일 수 있음

im2ool

spread input data for filtering

4D → 2D



The diagram illustrates the input data and filter application process. On the left, a 3x3 grid labeled "input Data" is shown above a 3x1 vector labeled "Jm^2Col". To the right, a 3x3 grid labeled "filter" is shown above a 1x3 vector. An arrow points from the input data to the filter, indicating the convolution operation.

NumPy's transpose

$$\begin{array}{l} \text{Shape } (N, H, W, C) \longrightarrow (N, C, H, W) \\ \text{Index } 0, 1, 2, 3 \quad \text{transpose} \quad 0, 3, 1, 2 \end{array}$$

LeNet - 활성화 함수 = Sigmoid

Poofy : SubSampling

AlexNet - 활성화함수 = ReLU

- LAN (국내 정부 및 기관) 활용 이용

- Dropout 10%

Deep CNN

1 layer = Conv → ReLU → Conv → ReLU → Pool

filter size = 3×3 (small)

총 1 깊어있수록 새널 수 1

P_{avg}/k_B 계층 초기 \Rightarrow 충돌 데미터 공간 초기

가장 좋은 optimizer = adam

Data Augmentation, 데이터 확장

운전용 데이터셋을 안전적으로 확장(crop, flip)

회의 갈이의 중요성

$$5 \times 5 \text{ conv} = (3 \times 3 \text{ conv}) \times 2$$

parameter 25 > 16

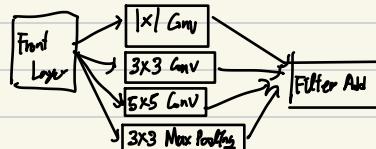
작은 필터를 경험 \Rightarrow 디지털 신호 \downarrow \Rightarrow 수용역 (receptive field) \downarrow

+ Gau 사이에 환경화 풍수 짜묘 \Rightarrow 비선형 표현 ॥

VGG 기본적인 CNN + 특별 있는 Conv. 단위별로 가중치 적용 16층

3x3의 작은 페터 쌍 Gnv 가중의 연속

2019년 2학기 코기 강의

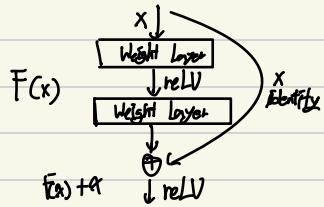


ResNet 층 개수가 많으면 계산량 ↓

↓ 해결법

Skip-Connection

입력 이미지를 학습한 계층과 원래 이미지 출력이 같음



Skip Connection \Rightarrow Backpropagation 더 쉬워

더 쉬워

사용 결과 블록의 CNN - R-CNN

영역 = 이미지 \rightarrow 추출 영역 \rightarrow CNN 훈련 계산 \rightarrow 영역 예측

Segmentation, 영역, 이미지 재생성 예측 - FCN