딥러닝 방법론의 이해

- CNN, RNN, Transformer를 중심으로

성신여자대학교
이성건 교수

목차

- Deep Learning
- Convolutional Neural Network
- Recurrent Neural Network
- Transformer

Reinforcement Learning(강화학습)

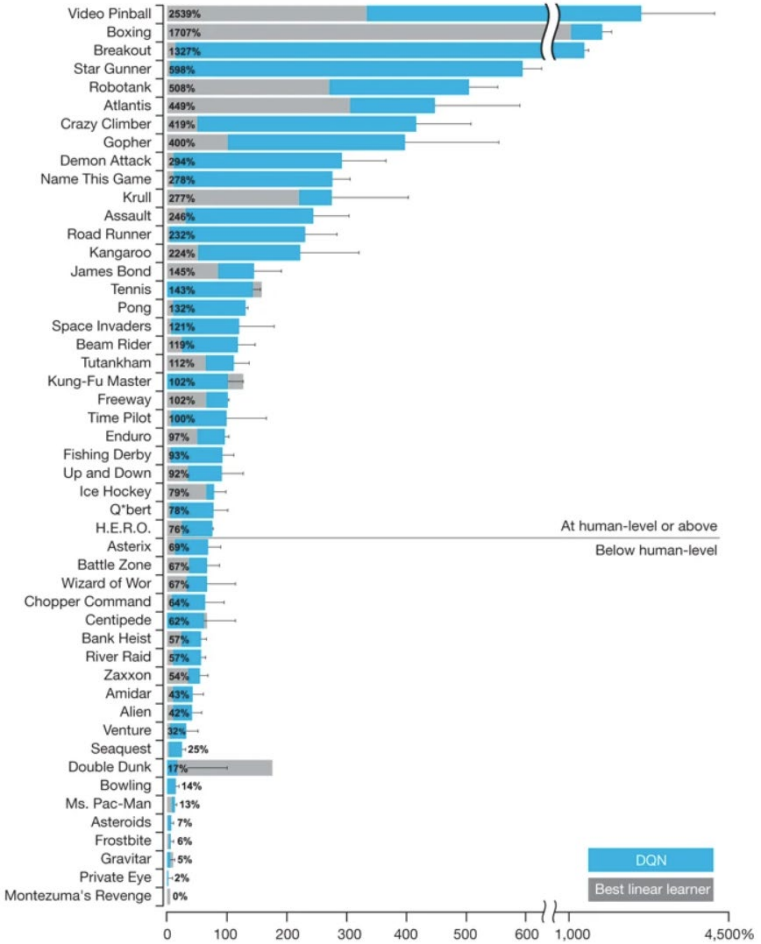
- Reinforcement : 강화(强化), 보강(補强), 보상(補償)
- the process of encouraging or establishing a belief or pattern of behavior, especially by encouragement or reward
- 인공지능 학습의 프레임 워크
- 주어진 환경(데이터)와 상호작용하며 보상을 최대화함으로써 학습을 수행함

Reinforcement Learning in Games

- Human-level control through deep reinforcement learning(2015, Nature)
- https://www.nature.com/articles/nature14236
- Deep Reinforcement Learning (Deep Q-Networks : DQN)
- Atari Games 2600

Galaxian:

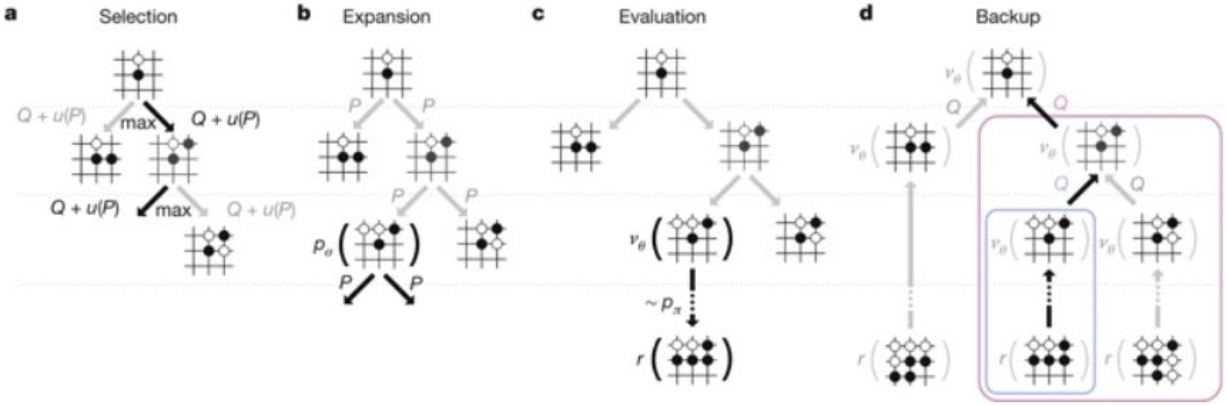
Figure 3: Comparison of the DQN agent with the best reinforcement learning methods[15] in the literature.



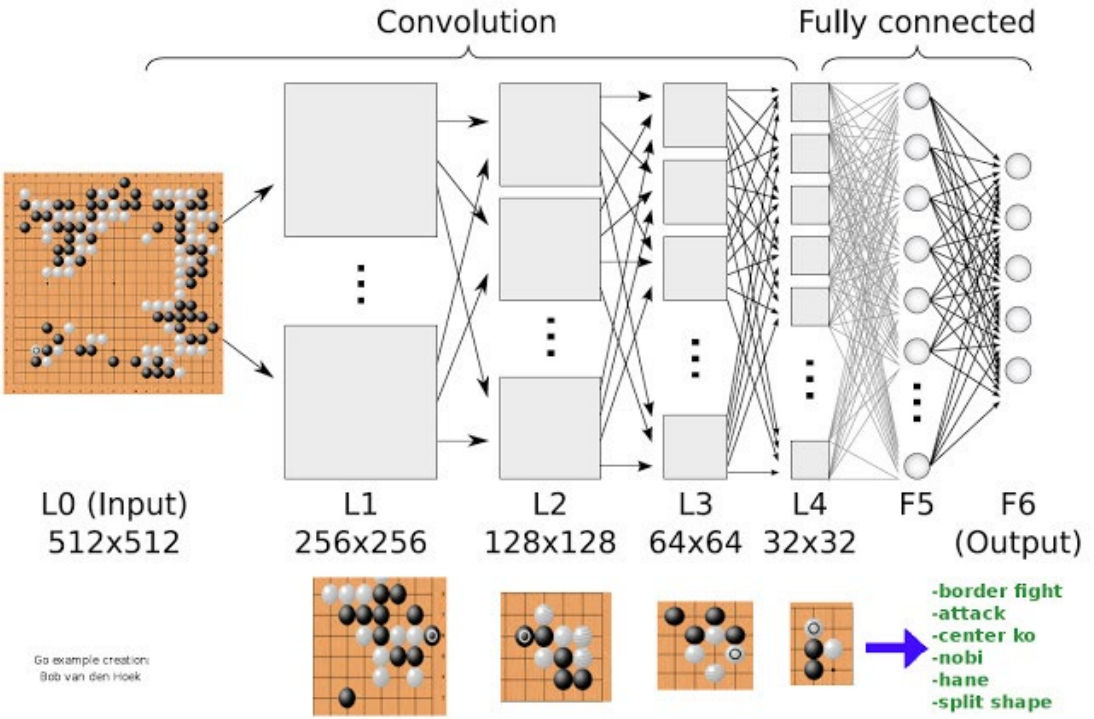
AlphaGo

- Mastering the game of Go with deep neural networks and tree search(Nature, 2016)
- https://www.nature.com/articles/nature16961

Figure 3: Monte Carlo tree search in AlphaGo.



AlphaGo



Neural network architecture

The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used *k* = 192 filters; Fig. 2b and

Image Segmentation

Hyundai motors Project – Detecting Obstacles in Road

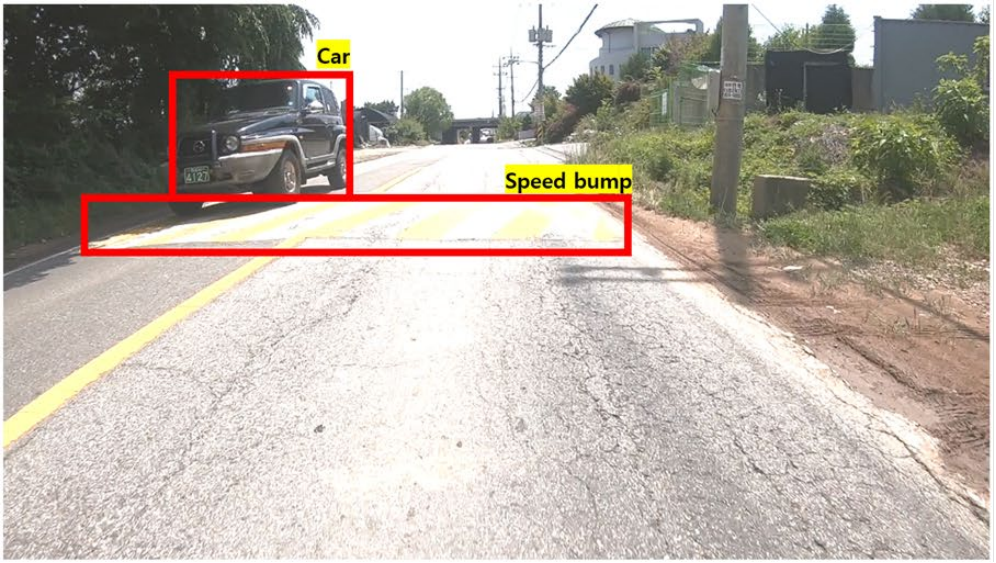
과속방지턱 (Speed bump)

도로품질 저하 (Road deterioration)

구덩이 (Port hole)

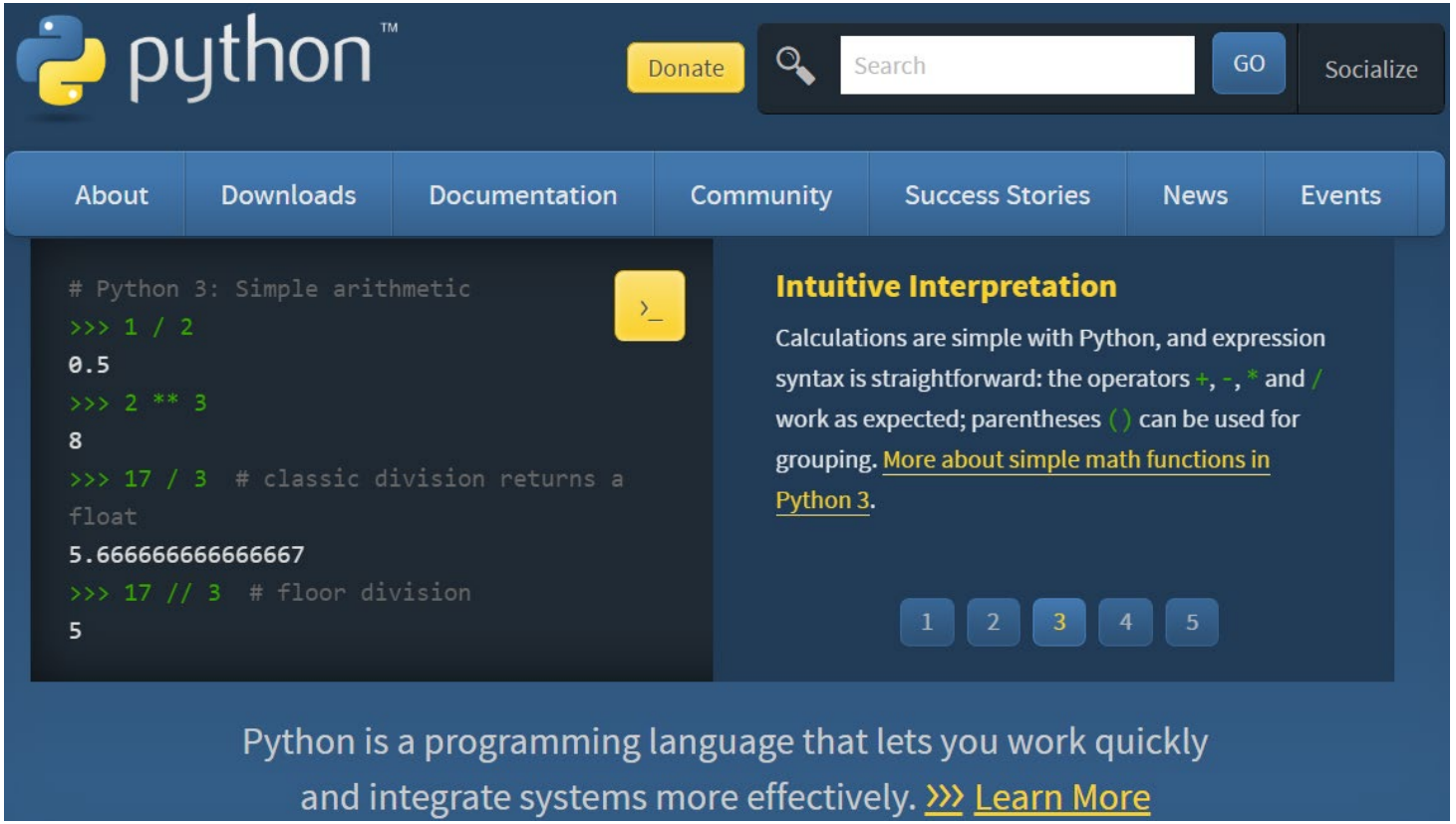
도로주행 장애물 (Litter including fallen tree and others)

Hyundai motors Project – Detecting Obstacles in Road



Tool : Python, COLAB, Fast AI

- C, C++, JAVA 보다 배우기 쉽고 문법이 간단
- 1/3 정도의 코드만으로 동일한 기능 수행
- Free
- https://www.python.org



Tool : Python, COLAB, Fast AI

- 파이썬 패키지

Anaconda : https://www.anaconda.com/
Tensorflow : https://www.tensorflow.org/?hl=ko
PyTorch : https://pytorch.org/
FastAI : https://www.fast.ai/

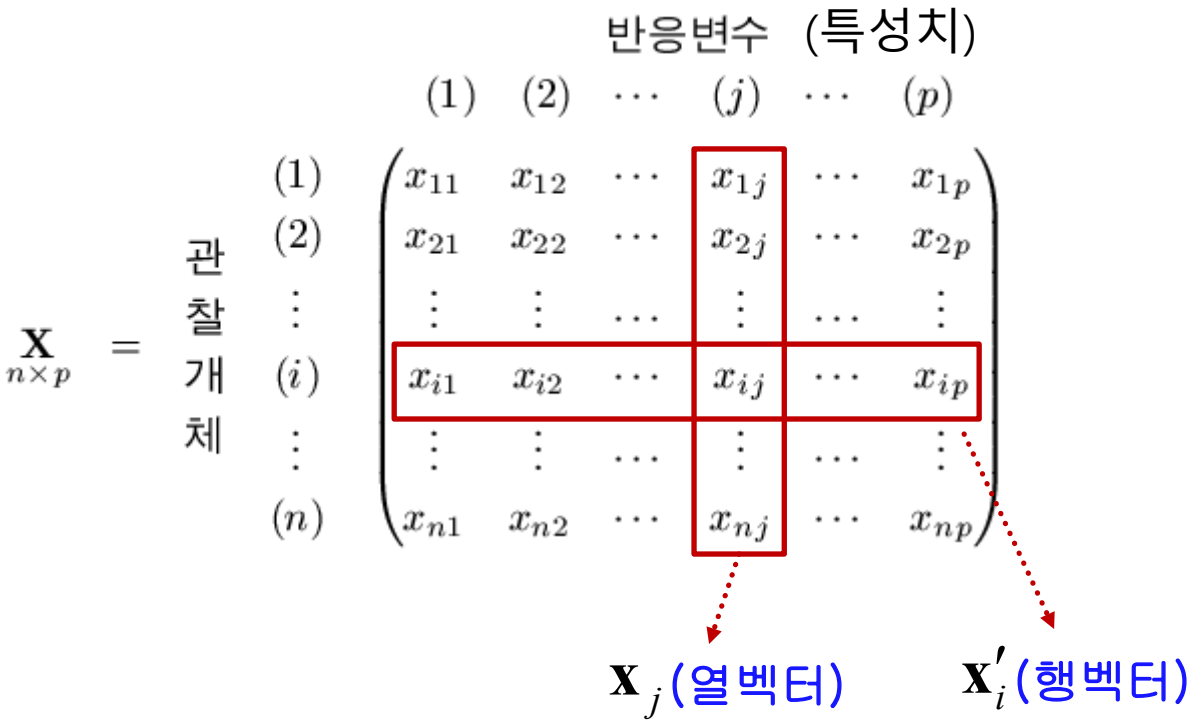
- 프로그래밍 에디터

Jupyter Notebook

- COLAB : http://colab.research.google.com/

Linear Algebra

- 행렬(matrix)
- 텐서(tensor)



평균

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_N}{N}$$

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

```
def list_mean(p):
    total = 0.0
    for t in p:
        total += t
    mean = total / len(p)
    return mean
```

$$(1,1,\ldots,1,1) \cdot \begin{pmatrix} x1 \\ x2 \\ \cdot \\ \cdot \\ \cdot \\ xN \end{pmatrix} \cdot 1/N$$

np.matmul(a, b)

- 반복문을 피함
- 데이터를 텐서에 저장하여, 텐서끼리의 간단한 사칙연산으로 해결
- 병렬처리

Regression

- 회귀분석: 반응변수가 설명변수들에 의해 어떻게 설명(예측)되는지를 알아보기 위해 그 관계를 적절한 함수식으로 표현하여 분석하는 통계적 자료분석 방법

$$Y = f(X_1, X_2, \cdots, X_p) + \varepsilon$$

- 반응변수(Response Variable), 종속변수(Dependent Variable), 목표변수(Target variable), 아웃풋/출력(output)
- 설명변수(Explanatory Variable), 예측변수(Predictor), 독립변수(Independent Variable), 공변량(Covariate), 회귀변수(Regressor), 요인(Factor), 매개변수(Carrier) , 인풋/입력(input)

- 근사(approximation)의 불일치를 나타내는 확률오차(random error)

회귀 (Regression)

Table 8.1. Galton's 1885 cross-tabulation of 928 adult children born of 205 midparents, by their height and their midparent's height.

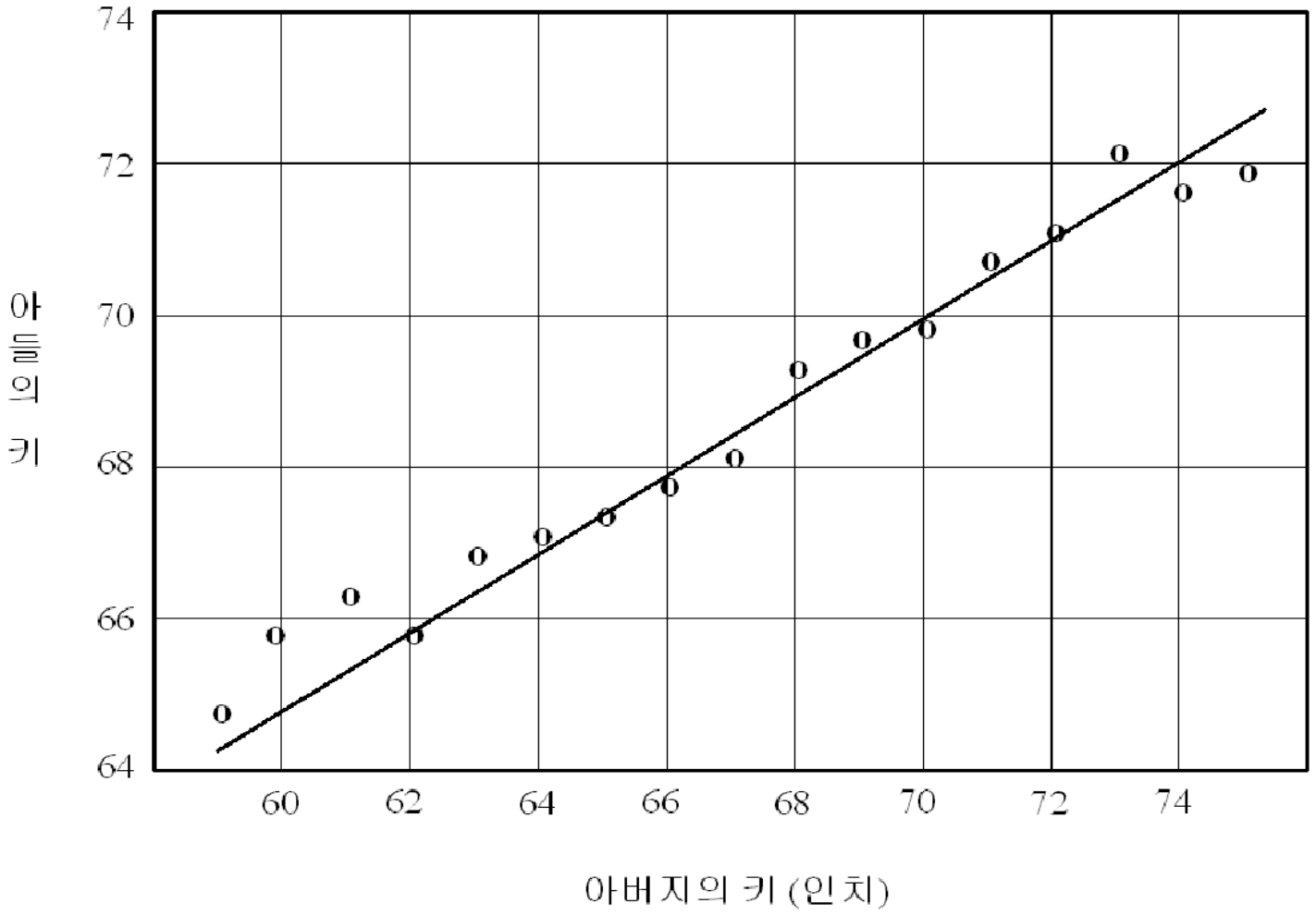
Height of the mid-parent in inches	Height of the adult child														Total no. of adult children	Total no. of mid-parents	Medians
	<61.7	62.2	63.2	64.2	65.2	66.2	67.2	68.2	69.2	70.2	71.2	72.2	73.2	>73.7			
>73.0	—	—	—	—	—	—	—	—	—	—	—	1	3	—	4	5	—
72.5	—	—	—	—	—	—	—	1	2	1	2	7	2	4	19	6	72.2
71.5	—	—	—	—	1	3	4	3	5	10	4	9	2	2	43	11	69.9
70.5	1	—	1	—	1	1	3	12	18	14	7	4	3	3	68	22	69.5
69.5	—	—	1	16	4	17	27	20	33	25	20	11	4	5	183	41	68.9
68.5	1	—	7	11	16	25	31	34	48	21	18	4	3	—	219	49	68.2
67.5	—	3	5	14	15	36	38	28	38	19	11	4	—	—	211	33	67.6
66.5	—	3	3	5	2	17	17	14	13	4	—	—	—	—	78	20	67.2
65.5	1	—	9	5	7	11	11	7	7	5	2	1	—	—	66	12	66.7
64.5	1	1	4	4	1	5	5	—	2	—	—	—	—	—	23	5	65.8
<64.0	1	—	2	4	1	2	2	1	1	—	—	—	—	—	14	1	—
Totals	5	7	32	59	48	117	138	120	167	99	64	41	17	14	928	205	—
Medians	—	—	66.3	67.8	67.9	67.7	67.9	68.3	68.5	69.0	69.0	70.0	—	—	—	—	—

Source: Galton (1886a).

Note: All female heights were multiplied by 1.08 before tabulation. Galton added an explanatory footnote to the table: "In calculating the Medians, the entries have been taken as referring to the middle of the squares in which they stand. The reason why the headings run 62.2, 63.2, &c., instead of 62.5, 63.5, &c., is that the observations are unequally distributed between 62 and 63, 63 and 64, &c., there being a strong bias in favour of integral inches. After careful consideration, I concluded that the headings, as adopted, best satisfied the conditions. This inequality was not apparent in the case of the Mid-parents." Galton republished these data in 1889, where they are referred to as the R.F.F. Data (Record of Family Faculties); he then noted that the first row must be in error (four children cannot have five sets of parents), but he claimed that "the bottom line, which looks suspicious, is correct" (p. 208).

Francis Galton(1889) : 아버지의 키와 아들의 키의 관계를 연구

회귀 (Regression)



Karl Pearson(1903) : 아들의 키 ≈ 33.73 + 0.516 x 아버지의 키

회귀 (Regression)

추정 (Estimation)

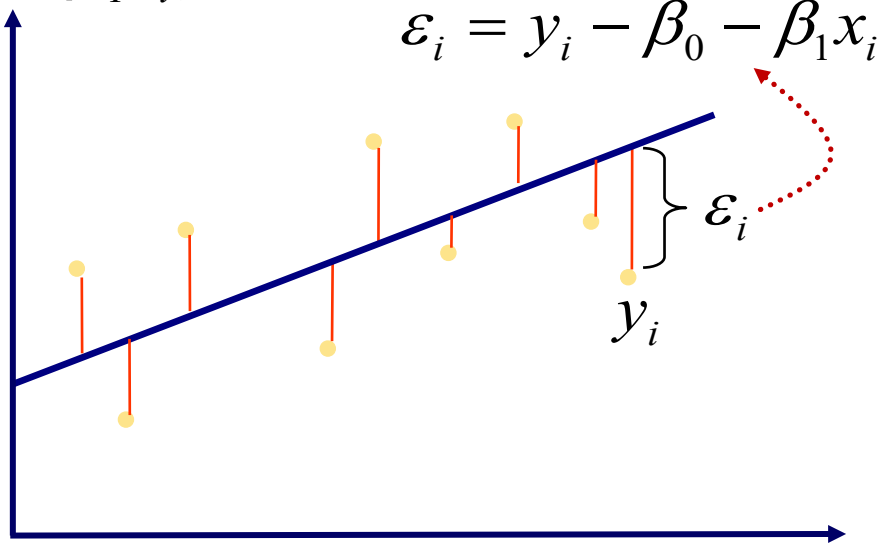
✓반응변수 대 예측변수의 산점도에 있는 점들을 가장 잘 적합(best fit) 혹은 표현하는 직선을 찾는 것

- 최소제곱추정 (Least Square Estimation)

$$Min\ S(\beta_0, \beta_1) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\Rightarrow \quad \hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$



Deep Learning

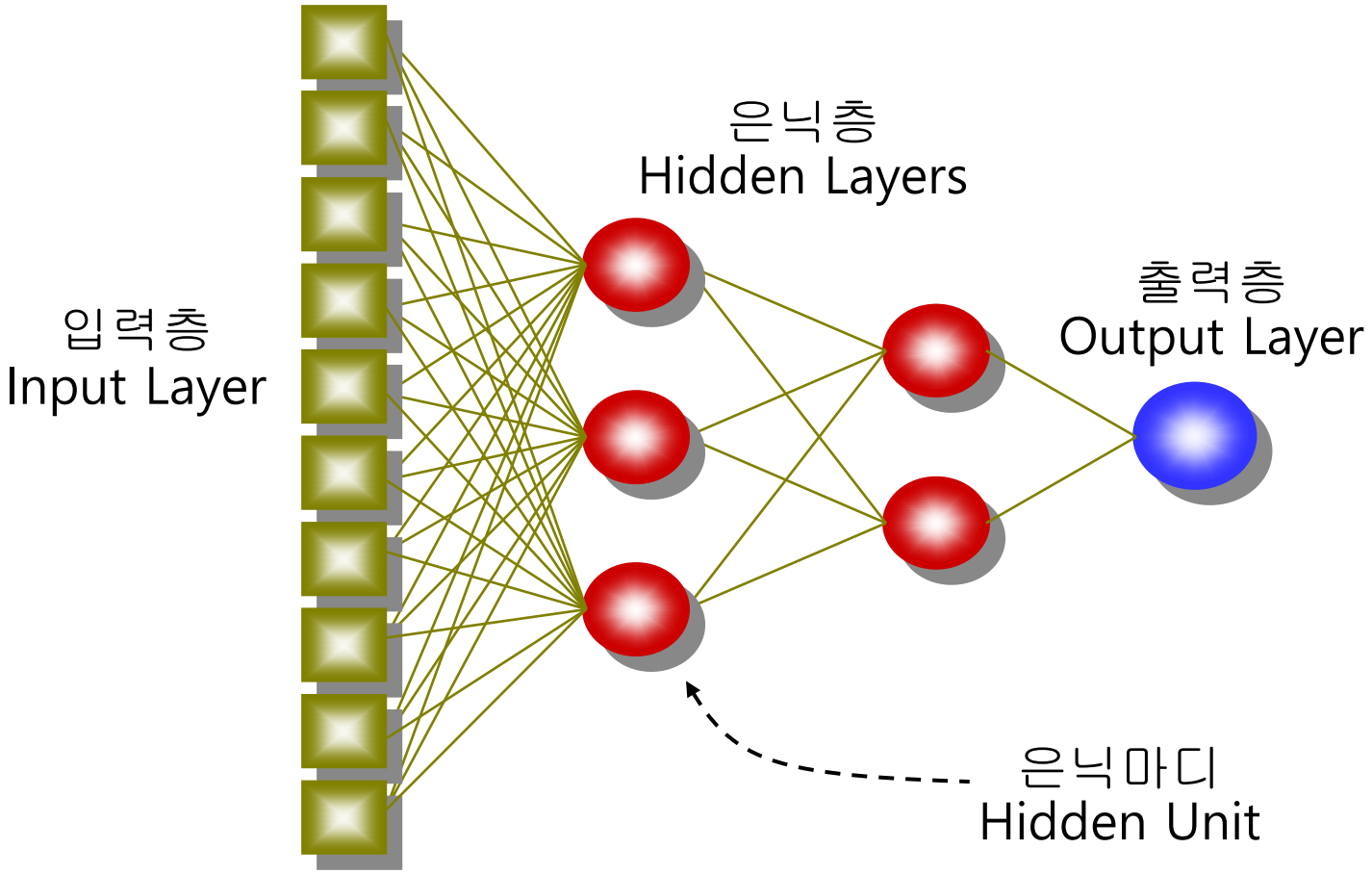
Deep Learning

- 인공신경망의 연속된 은닉층들은 우리의 뇌가 하는 것 처럼 일련의 단계를 거쳐 사물을 인지하고 문제를 처리

- 'Deep Learning'은 입력(input)과 출력(output)사이에 여러 개의 은닉층을 가진 인공신경망

- Deep Neural Network

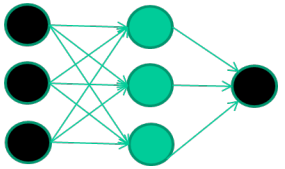
Feed Forward Network



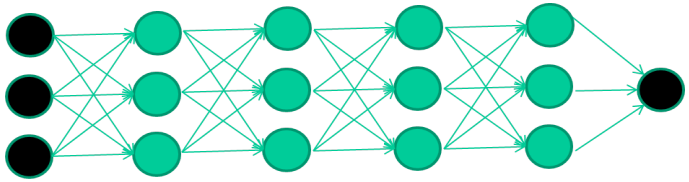
무엇이 새로운가?

- 인공신경망은 50년 이상 된 것인데 딥러닝이 새로운 것인가?

- 하나의 은닉층을 갖는 신경망을 학습시키는 것은 결과가 좋다

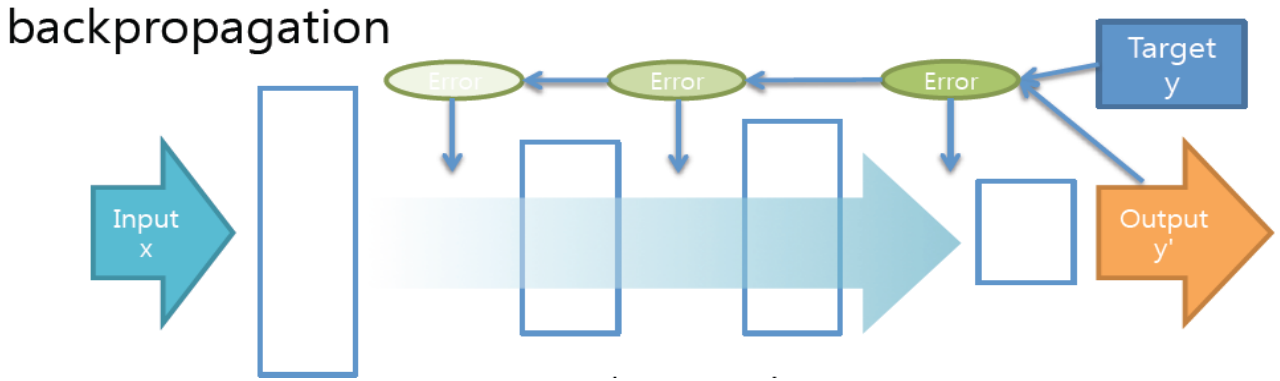


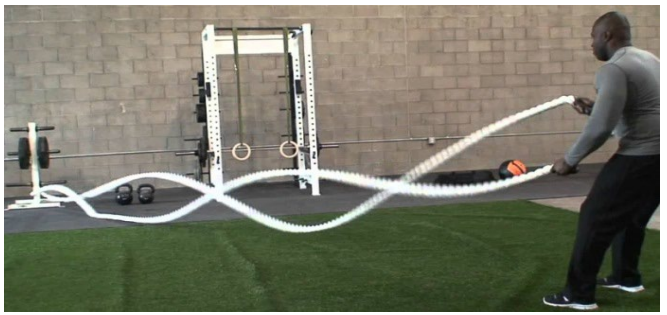
- 그러나 많은 은닉층을 갖는 신경망은 학습이 잘 되지 않음



역전파알고리즘(backpropagation)의 문제점

- 연결 가중치(weight)의 추정방법
- 출력층(Output)에 가까운 층(layer)으로 부터 가중치 조정이 역전파됨
- 멀어질 수록 조정이 잘 되지 않음
- 복잡한 신경망에 대해 학습이 잘 안됨



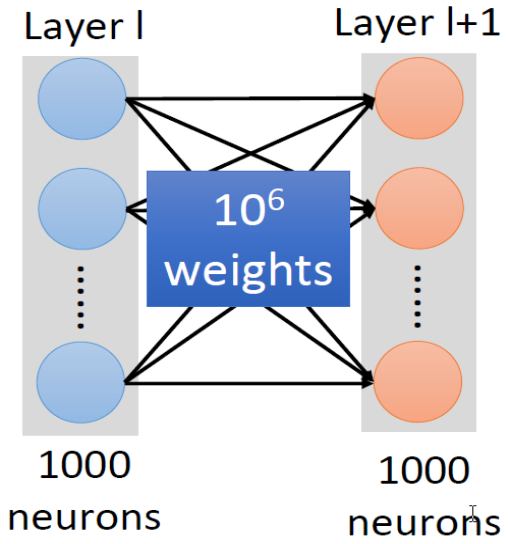


딥러닝 - 최적의 함수

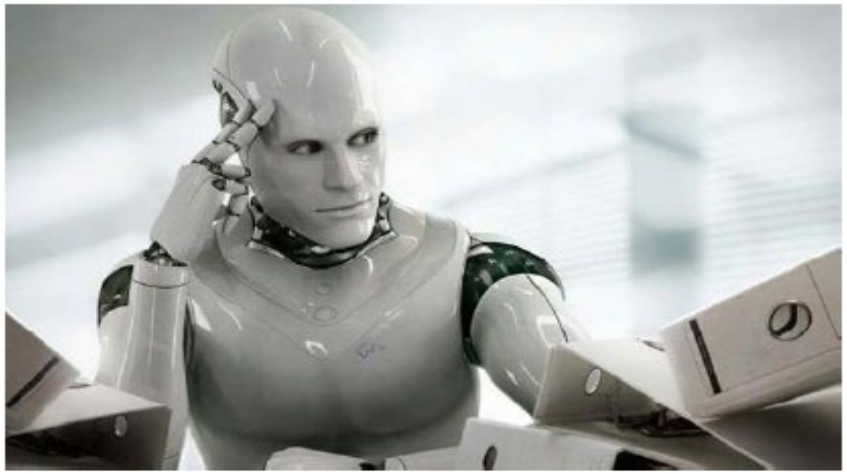
총손실(total loss)를 최소화 하는 네트워트 가중치 parameter를 찾아야 함

$$\text{Network parameters } \theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$$

- **찾아야 하는 가중치의 값이 매우 많음**
- **음성인식 : 8 layer, 1000neurons/layer**
- $10^3 \times 10^3 \times 10^3 \times 10^3 \times 10^3 \times 10^3 \times 10^3 \times 10^3$



인공지능이 최적 가중치를 찾는 방법



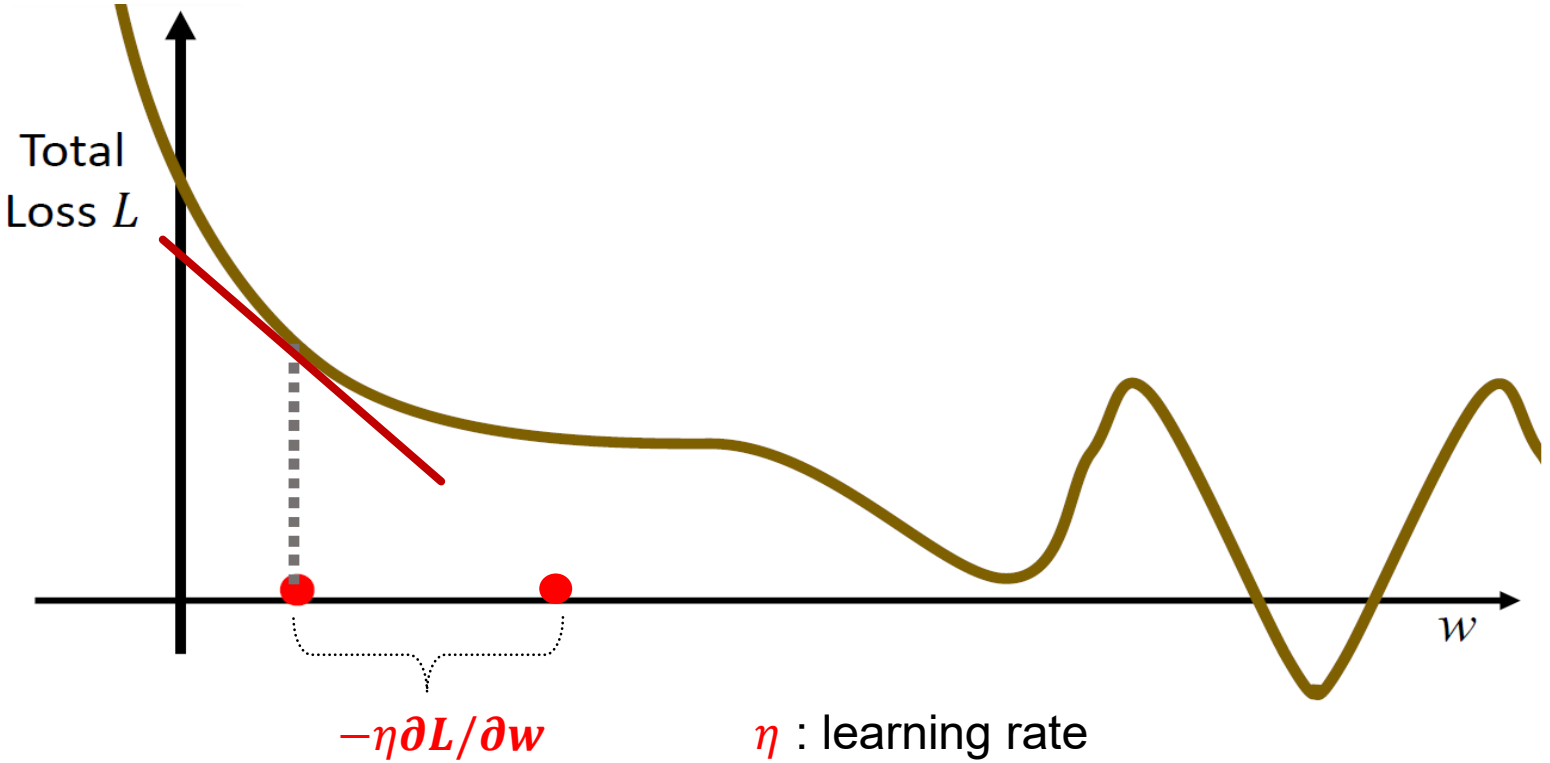
ALPHAGO?

Mr. Randy Pierce, **blind mountain climber**

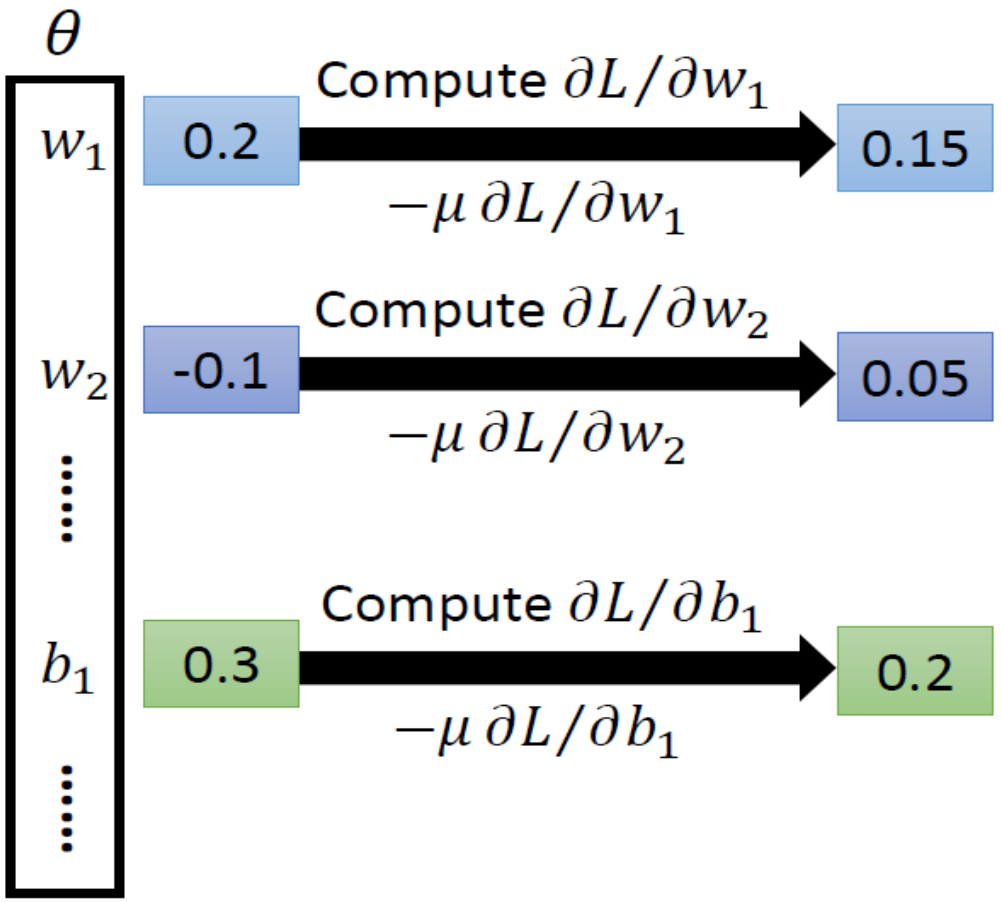
- 간단하게 고민해서 매우 쉽게 답을 찾는 멋진 모습? No!!
- 눈을 감고, 주변의 높이를 발로 반복적으로 가늠하여, 북한산 올라가기(최대화 문제), 해변 찾아가기(최소화 문제)

Gradient Descent

1. 초기값 w 선택
2. $\boldsymbol{\partial L/\partial w}$를 계산.
3. $w = w - \eta \frac{\partial L}{\partial w}$
4. 2단계와 3단계를 w값의 변화가 거의 없을 때 까지 반복.



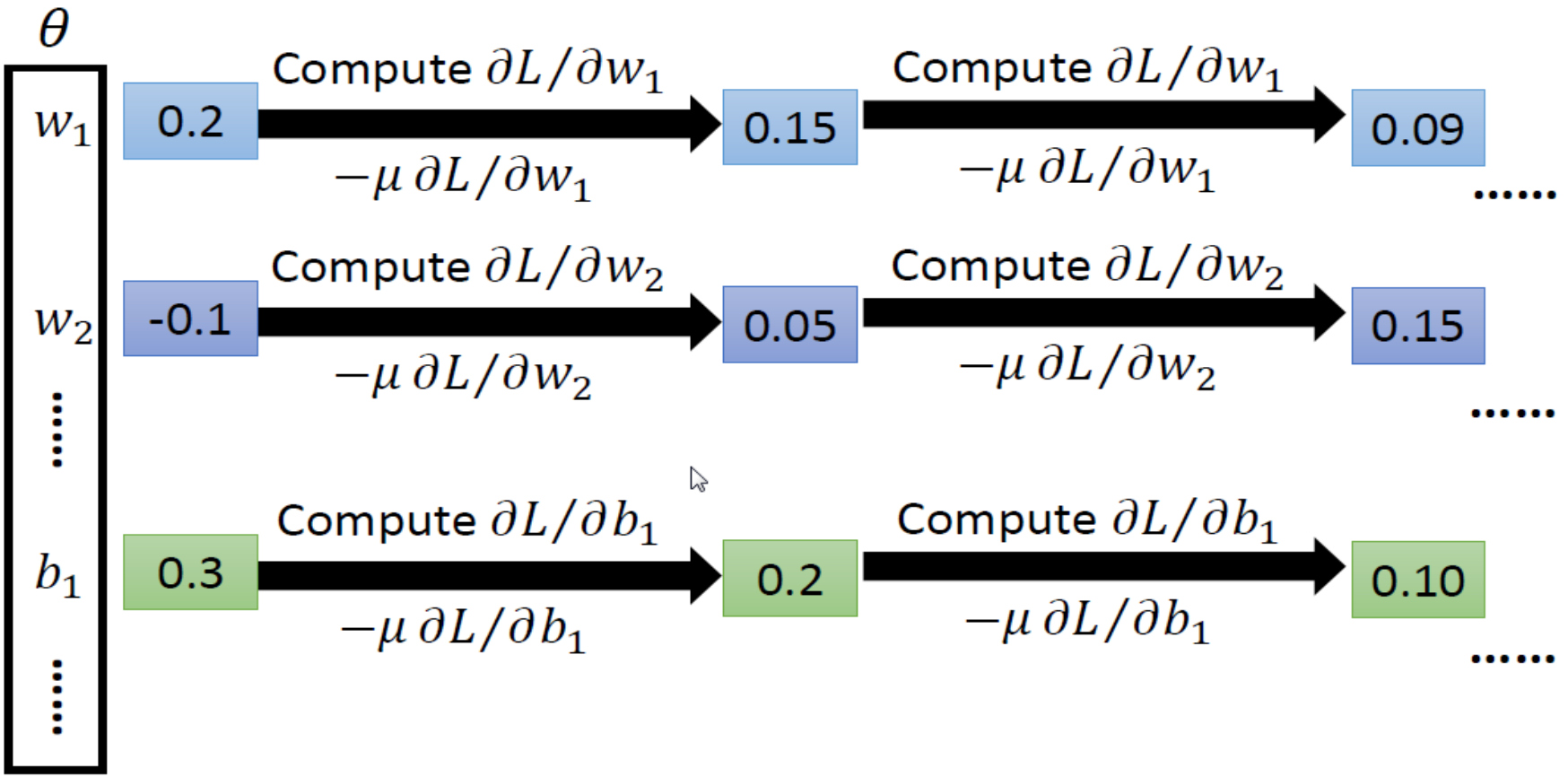
Gradient Descent



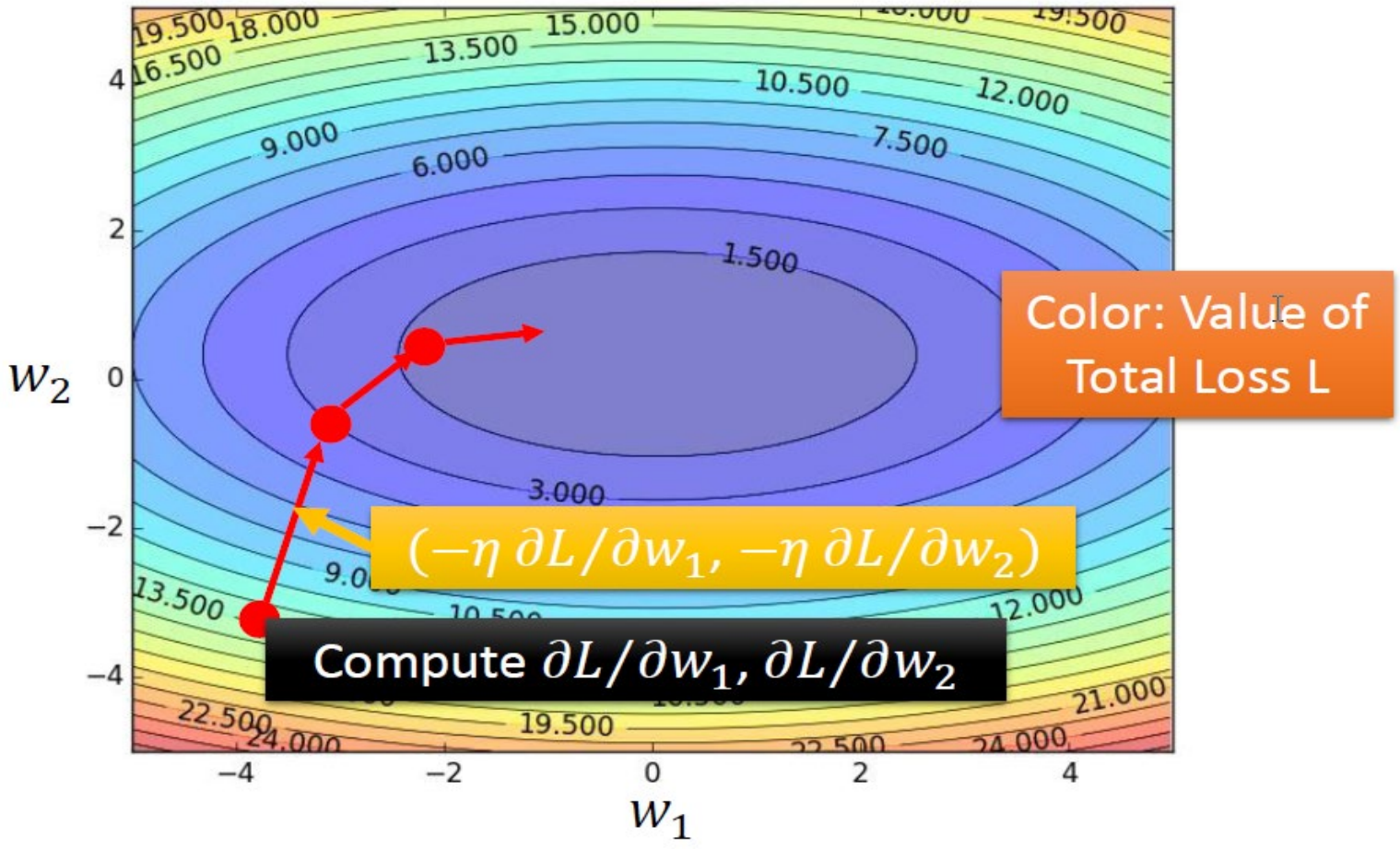
$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

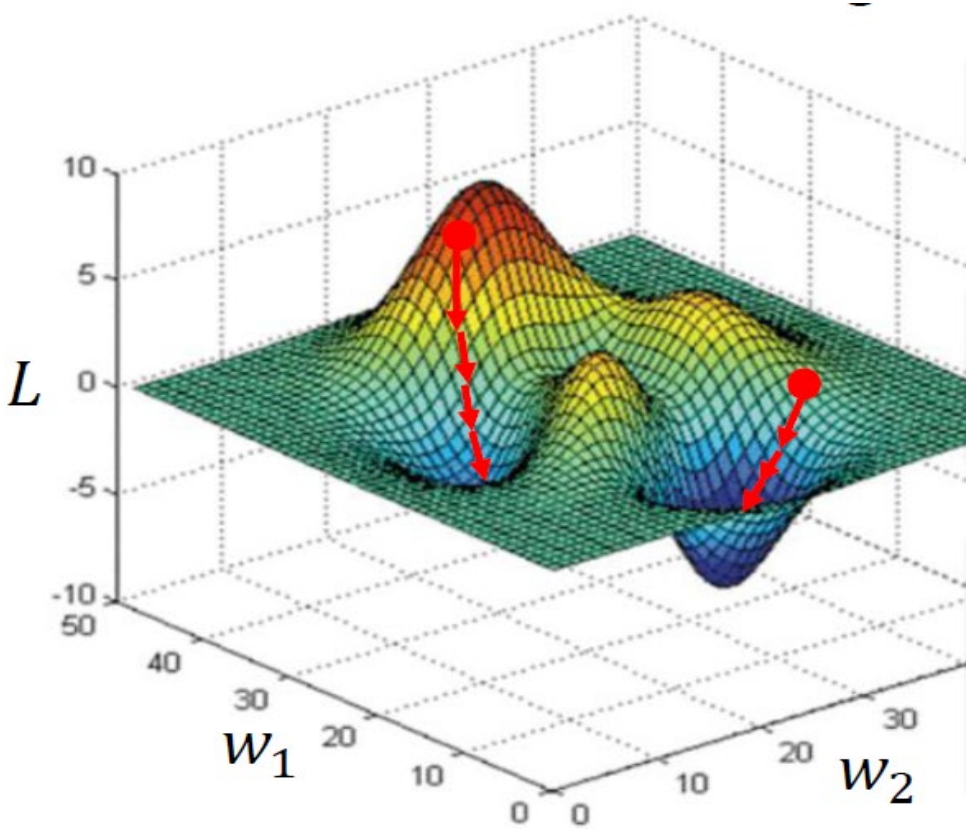
Gradient Descent



Gradient Descent



Gradient Descent –Difficulty – 로컬 최소값

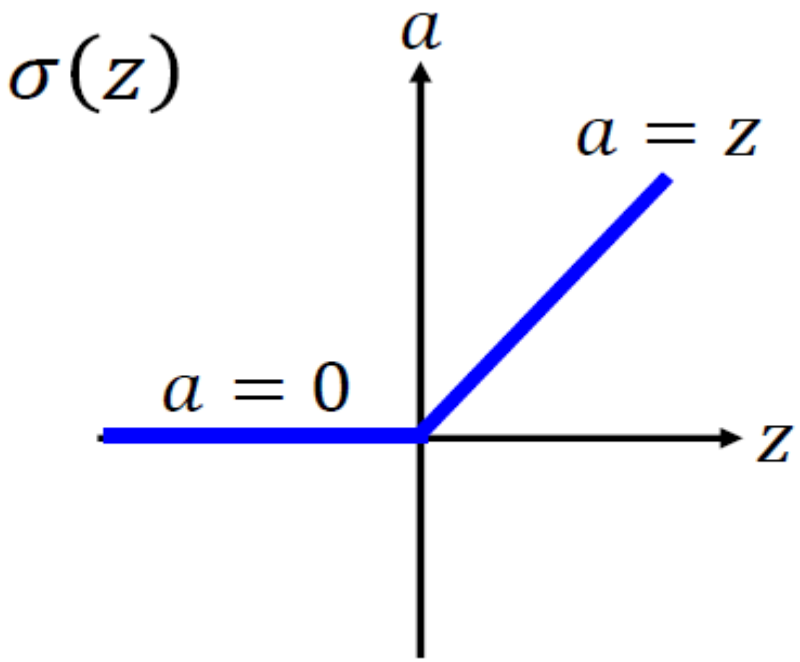


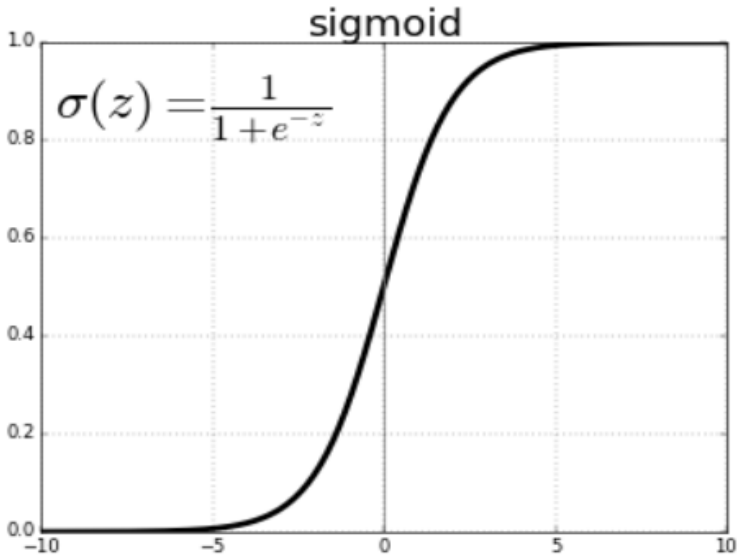
모형 성능 향상을 위한 방법

- 적절한 손실함수(loss function)의 선택
- 미니배치(Mini-batch)
- 새로운 활성함수(activation function)-ReLu
- 학습률 (learning rate) 조정
- 모멘텀(Momentum)
- 가중치 감퇴(Weight decay)
- 드롭아웃(Dropout)

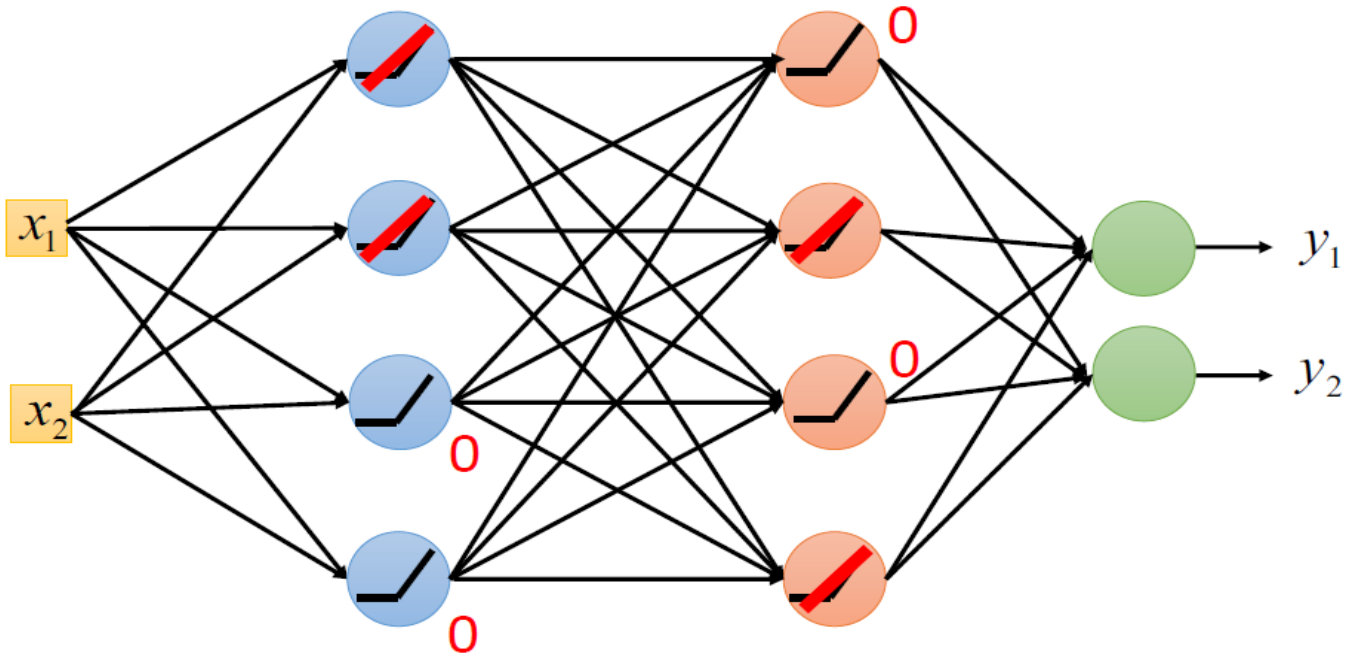
활성함수 : Rectified Linear Unit (ReLU)

$$\sigma(Z) = max(0, Z)$$

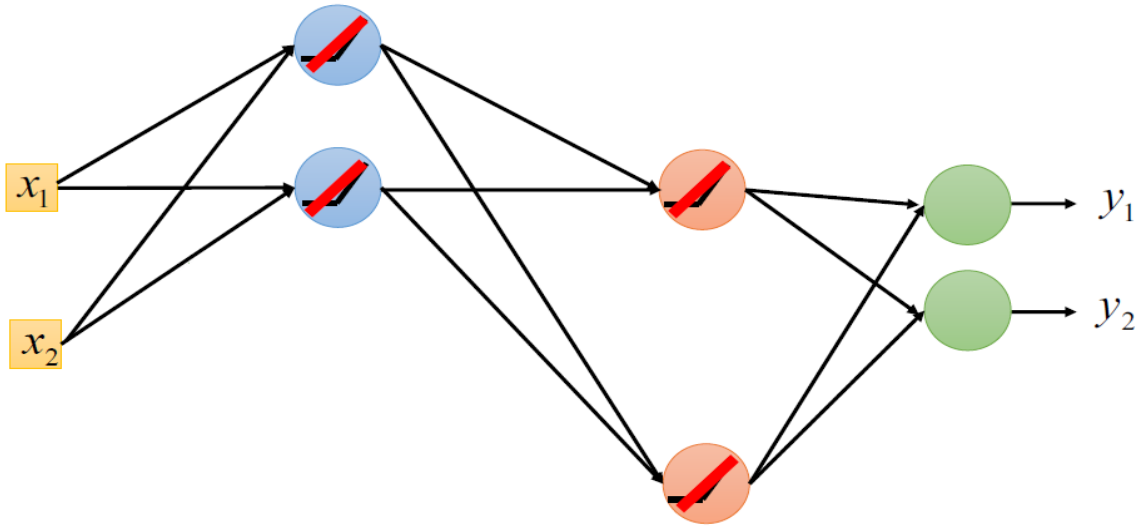




활성함수 : ReLU



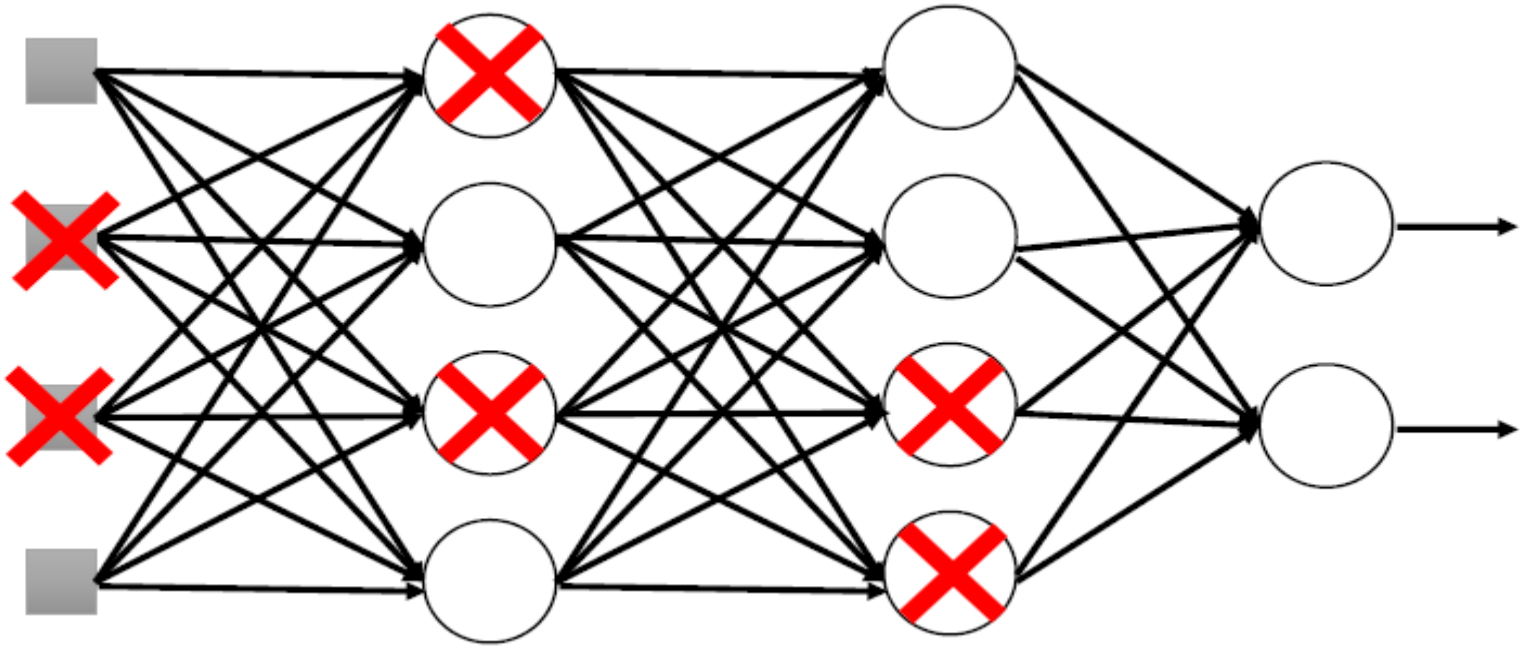
활성함수 : ReLU



단순한(thinner) 네트워크 구조 생성

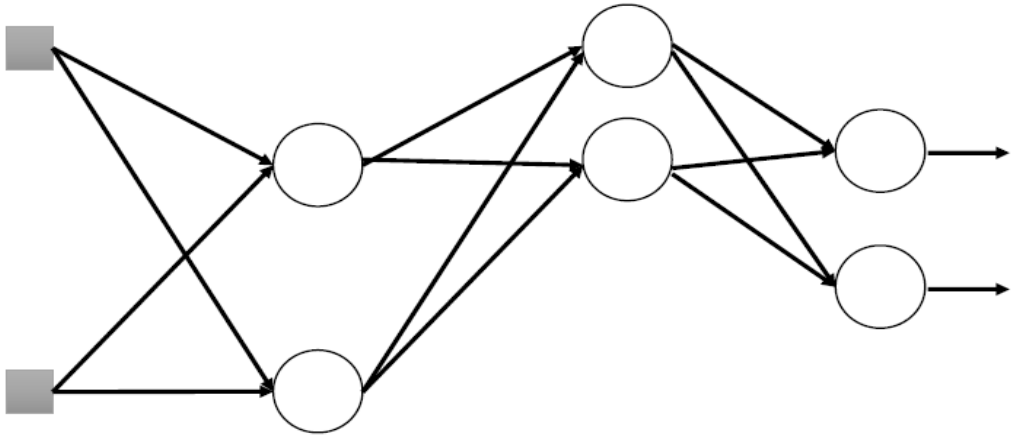
입력 근처에서 작은 기울기 값을 가지지 않게 됨

Dropout. 뉴런을 랜덤으로 제거



- 가중치를 업데이트하기 전에 뉴런의 p%를 제거

Dropout. 뉴런을 랜덤으로 제거



- 네트워크의 구조에 변화
- 간단해 짐(thinner)
- 각각의 mini-batch에서 제거할 뉴런을 추출(resample)

Dropout – 팀플?

- 팀프로젝트에서 각자가 다른 팀원들이 열심히 할 것으로 생각하면 본인은 열심히하지 않음 → 결과적으로 아무도 열심히 하지 않음

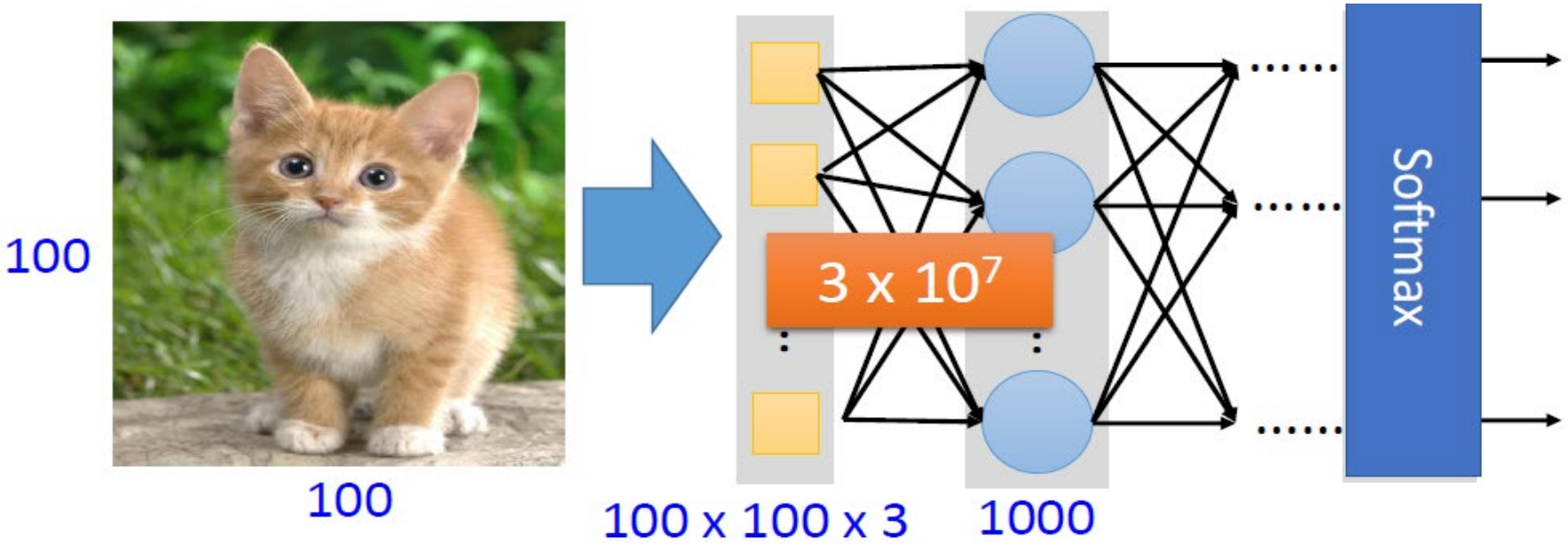
- 만일 다른 팀원이 수강철회(dropout)를 하면, 본인은 열심히 하게 됨.

- 다른 팀원들이 수강철회를 했다고 가정하고, 본인 모두가 열심히 함

- 최종 팀프로젝트 발표(test)에서 팀원 모두가 참여하면 좋은 결과를 도출하게 됨

Convolutional Neural Network

Image 데이터의 CNN



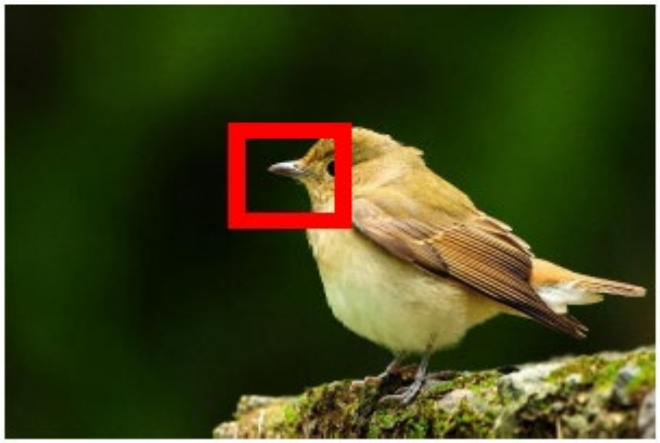
- 이미지 인식의 특성을 고려하여 네트워크를 간단하게 만들 수 있을까에 대한 고민

CNN

- 특정 패턴(patterns)은 전체 이미지에서 아주 작은 부분임
- 각 뉴런들은 패턴을 찾기 위해 전체 이미지 모두를 탐색할 필요는 없음
- 예 : 새의 부리 패턴만을 찾자

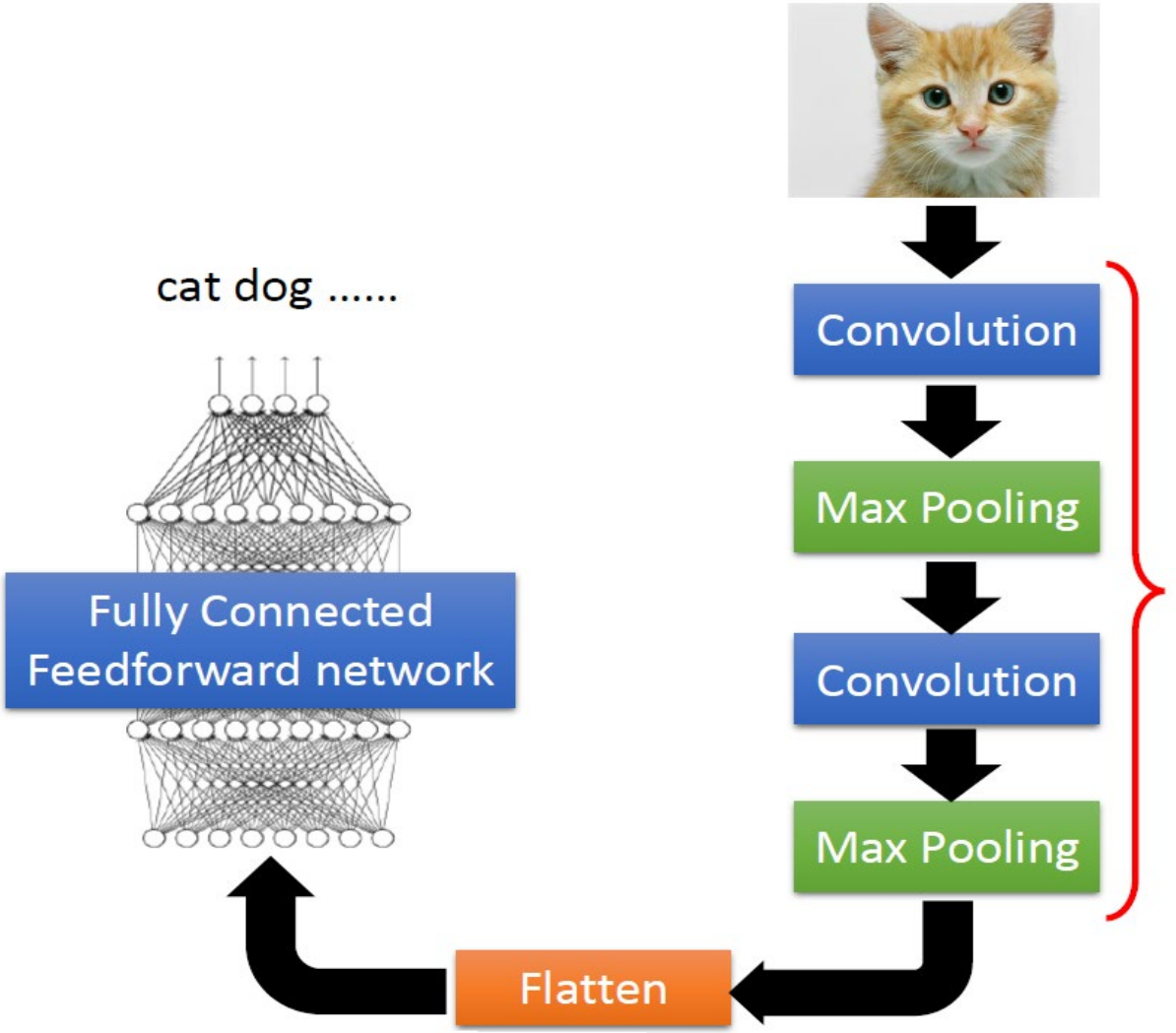
- 작은 영역에 대해, 적은 수의 가중치를 이용

CNN



- 동일한 패턴이 다른 영역에서 나타남
- 동일한 패턴이라고 인식가능해야 함

전체 CNN의 과정


cat dog .....
Fully Connected Feedforward network
Convolution
Max Pooling
Convolution
Max Pooling
Flatten

CNN –Convolution

Filter를 이용해 학습해야 할 네트워크 parameter를 설정

예 > (3 x 3)의 작은 패턴들.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1 Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2 Matrix

Stride. 보폭

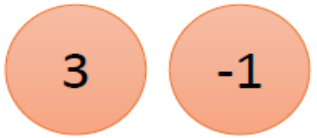
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

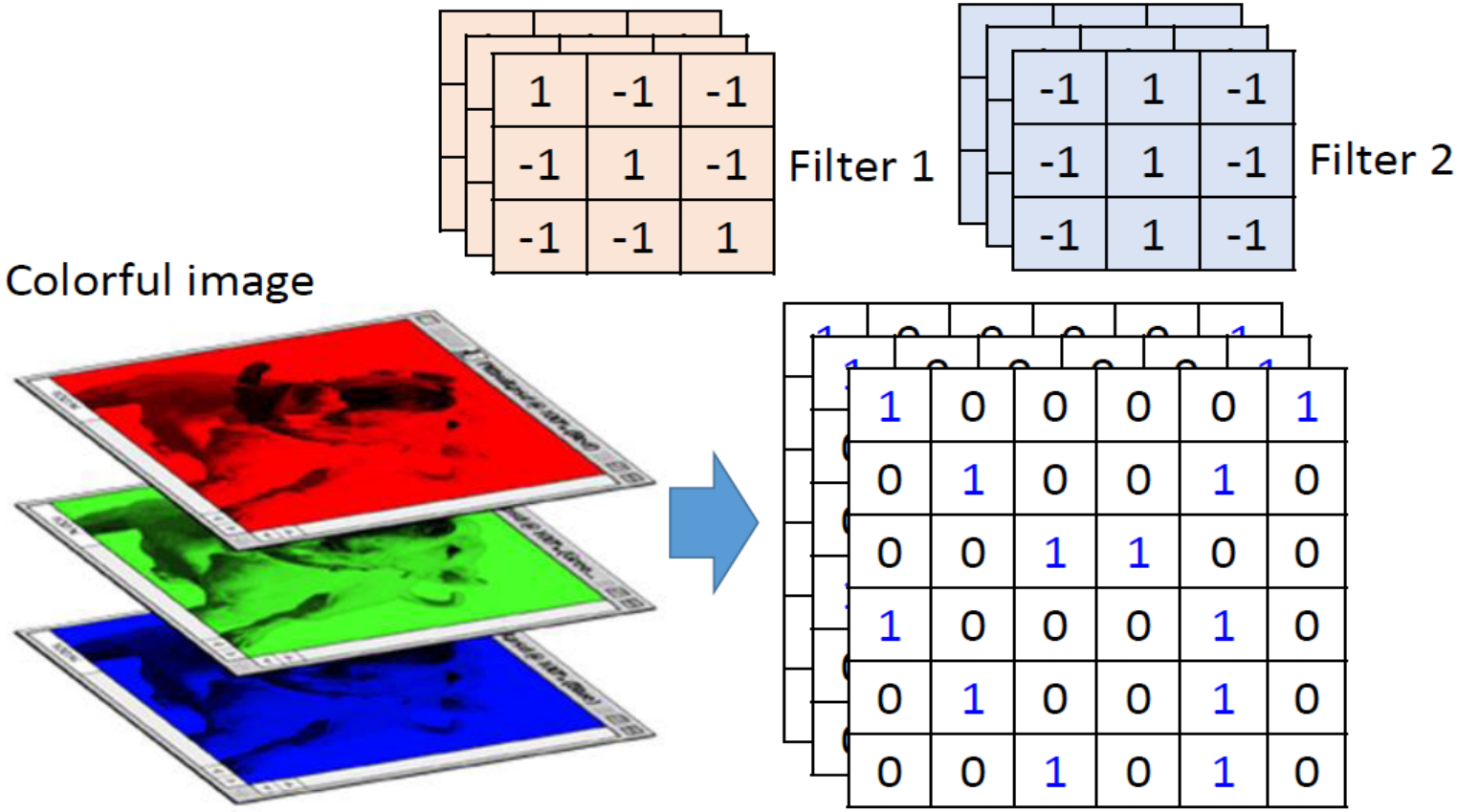
stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

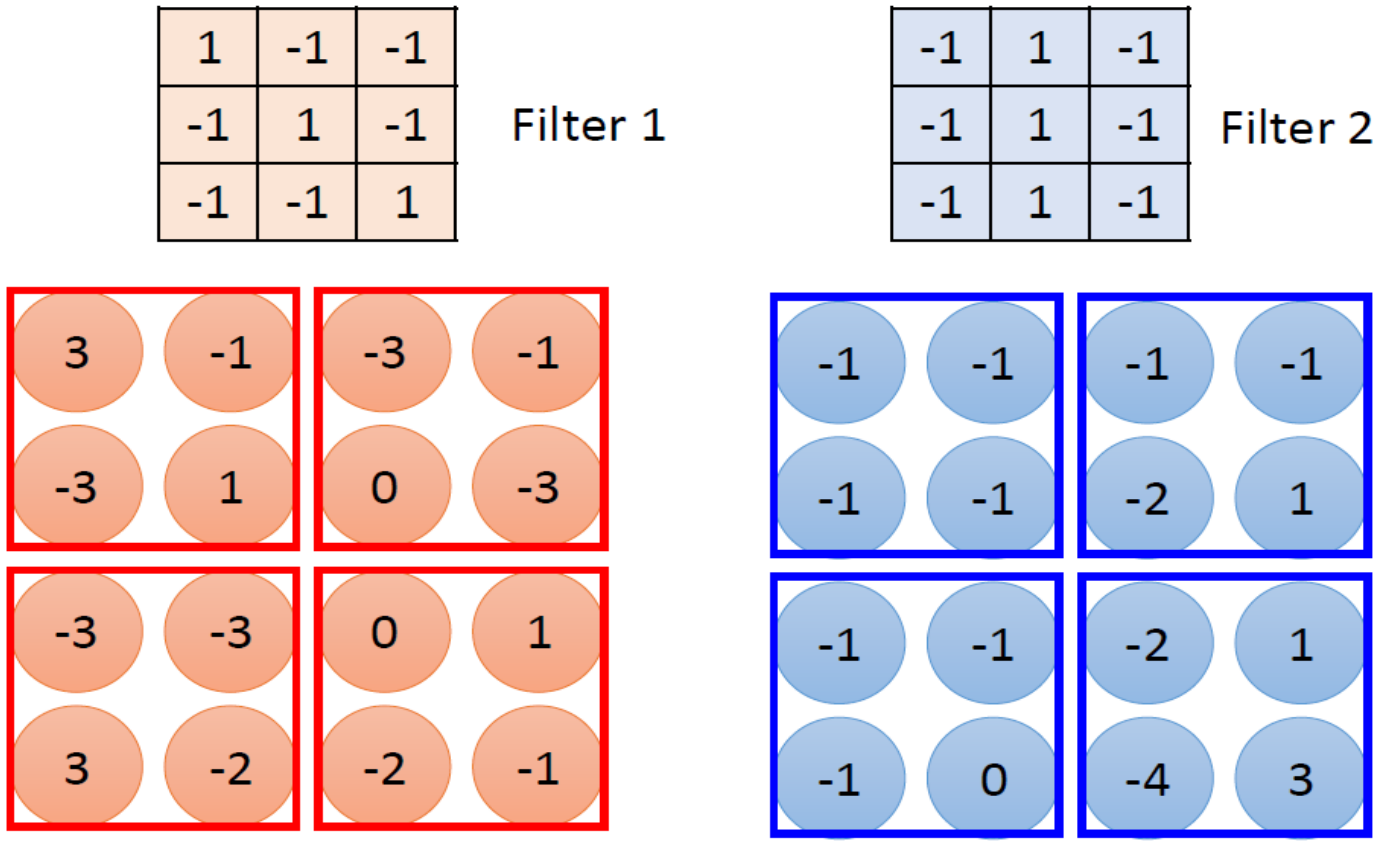
6 x 6 image



Colorful image

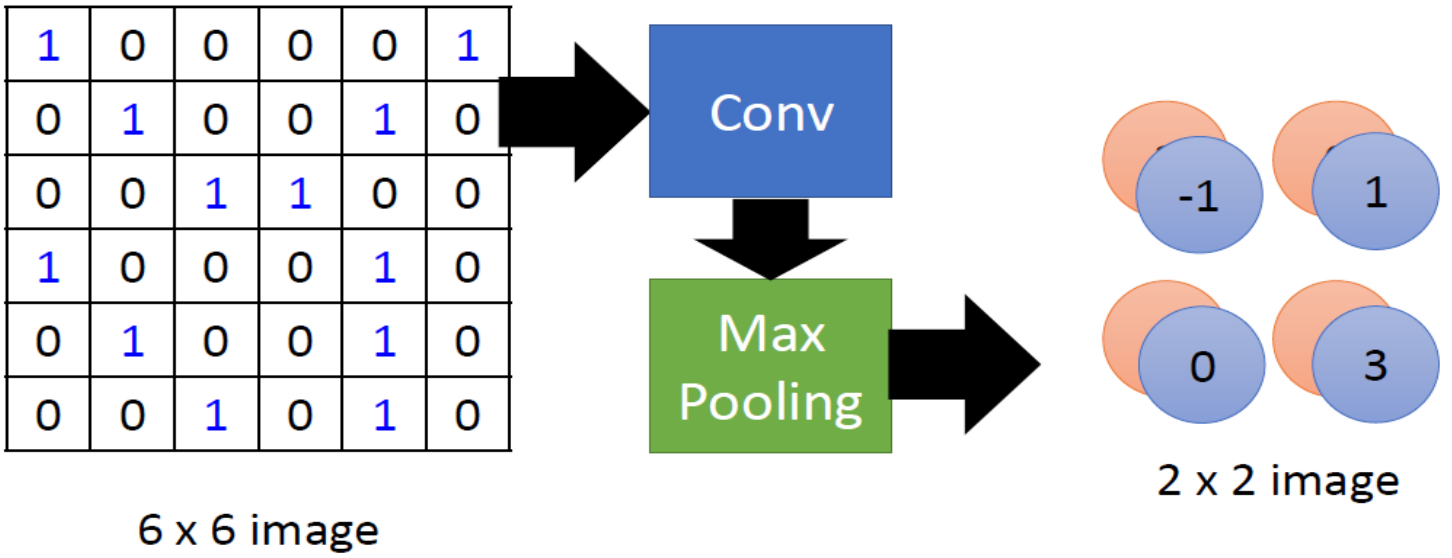


Max Pooling

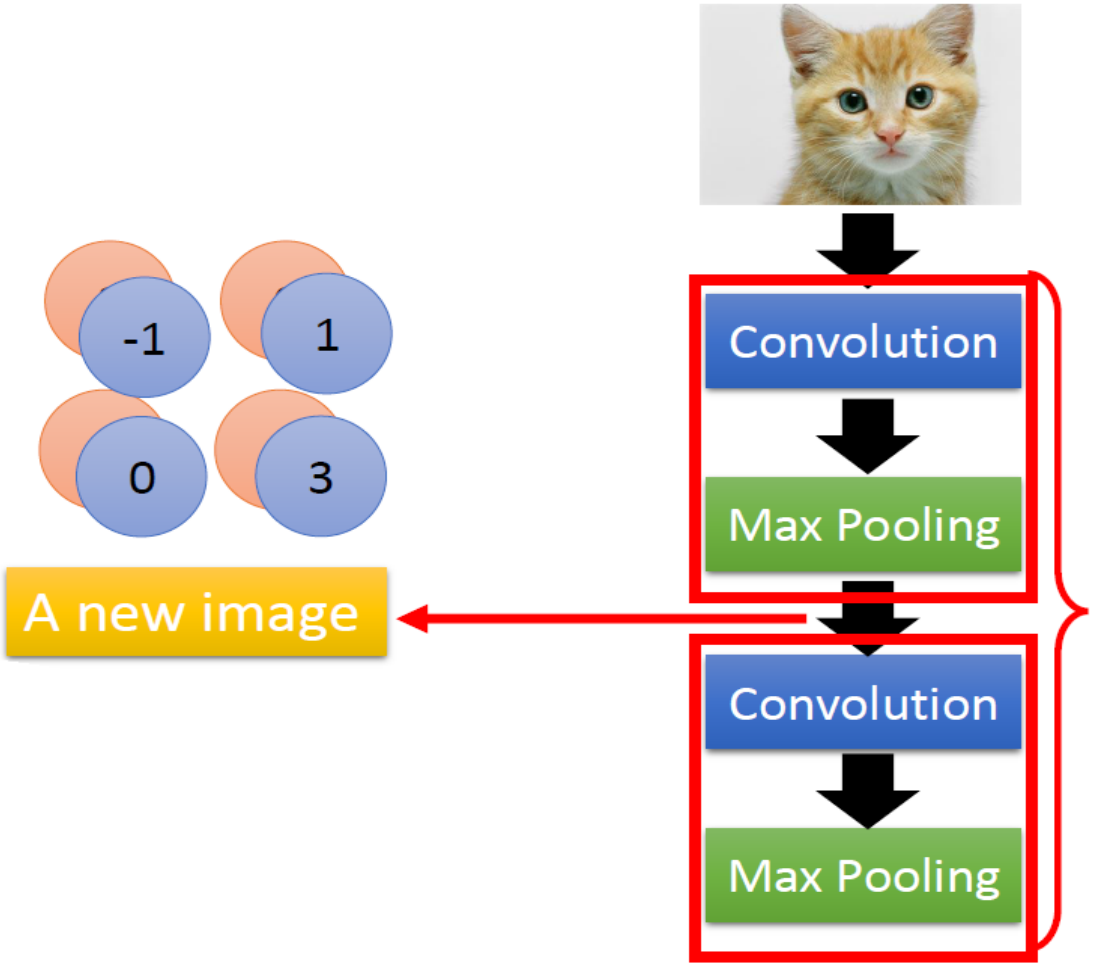

1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
-1 1 -1
-1 1 -1
-1 1 -1
Filter 2
3 -1
-3 1
-3 -1
0 -3
-3 -3
3 -2
0 1
-2 -1
-1 -1
-1 -1
-1 -1
-2 1
-1 -1
-1 0
-2 1
-4 3

CNN

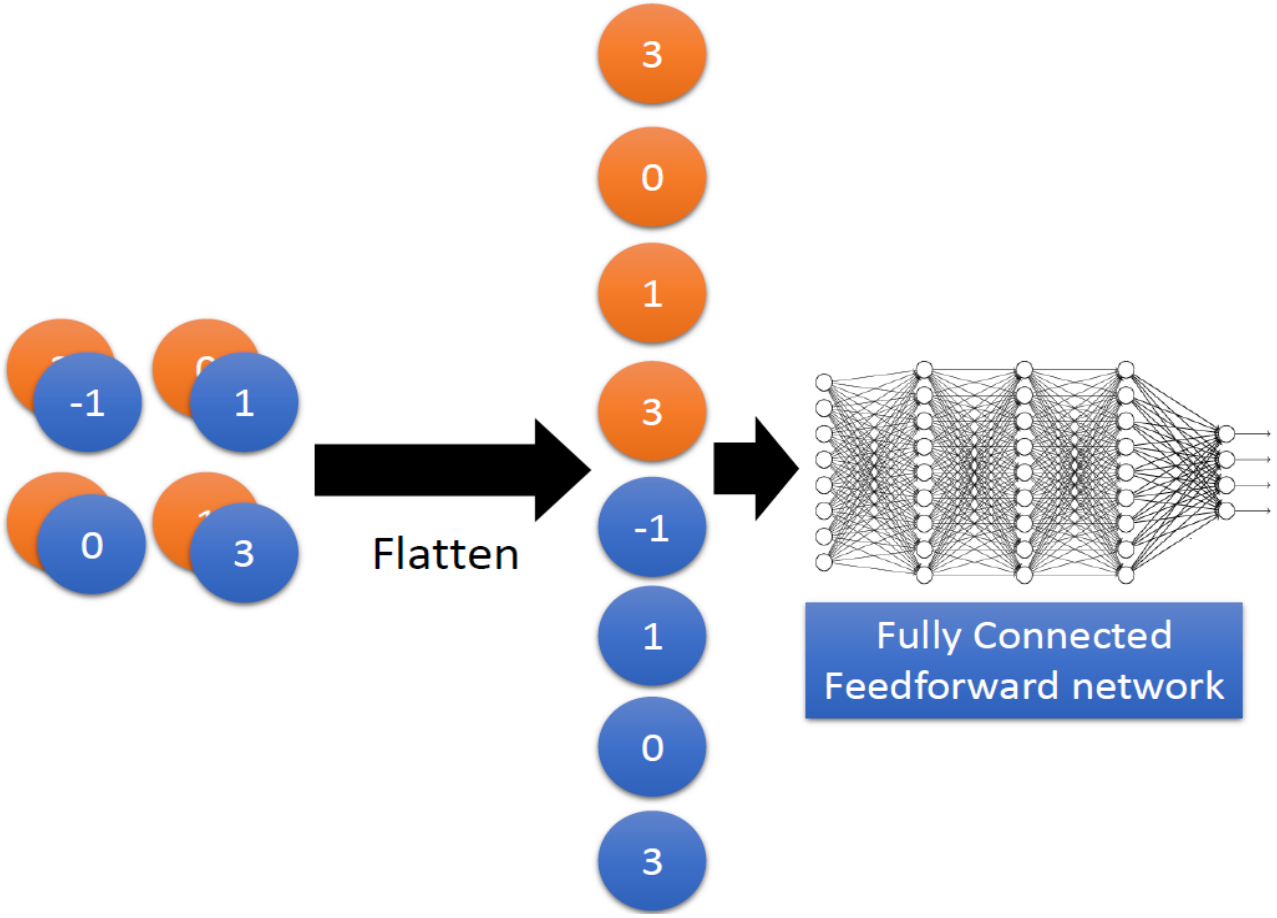
Convolution과 Maxpooling을 통해 더 작은 새로운 이미지 데이터 생성



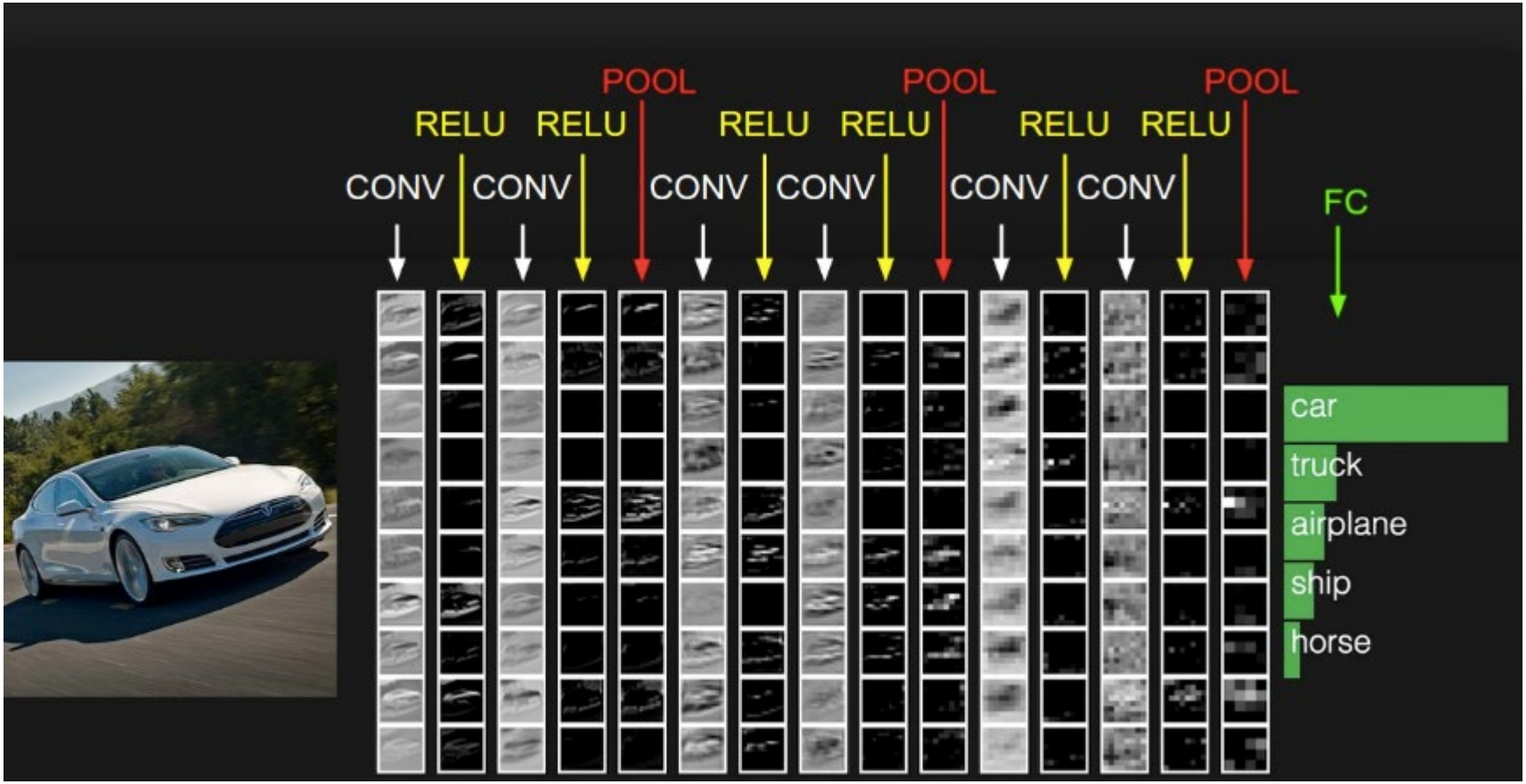
CNN


-1
1
0
3
A new image
Convolution
Max Pooling
Convolution
Max Pooling

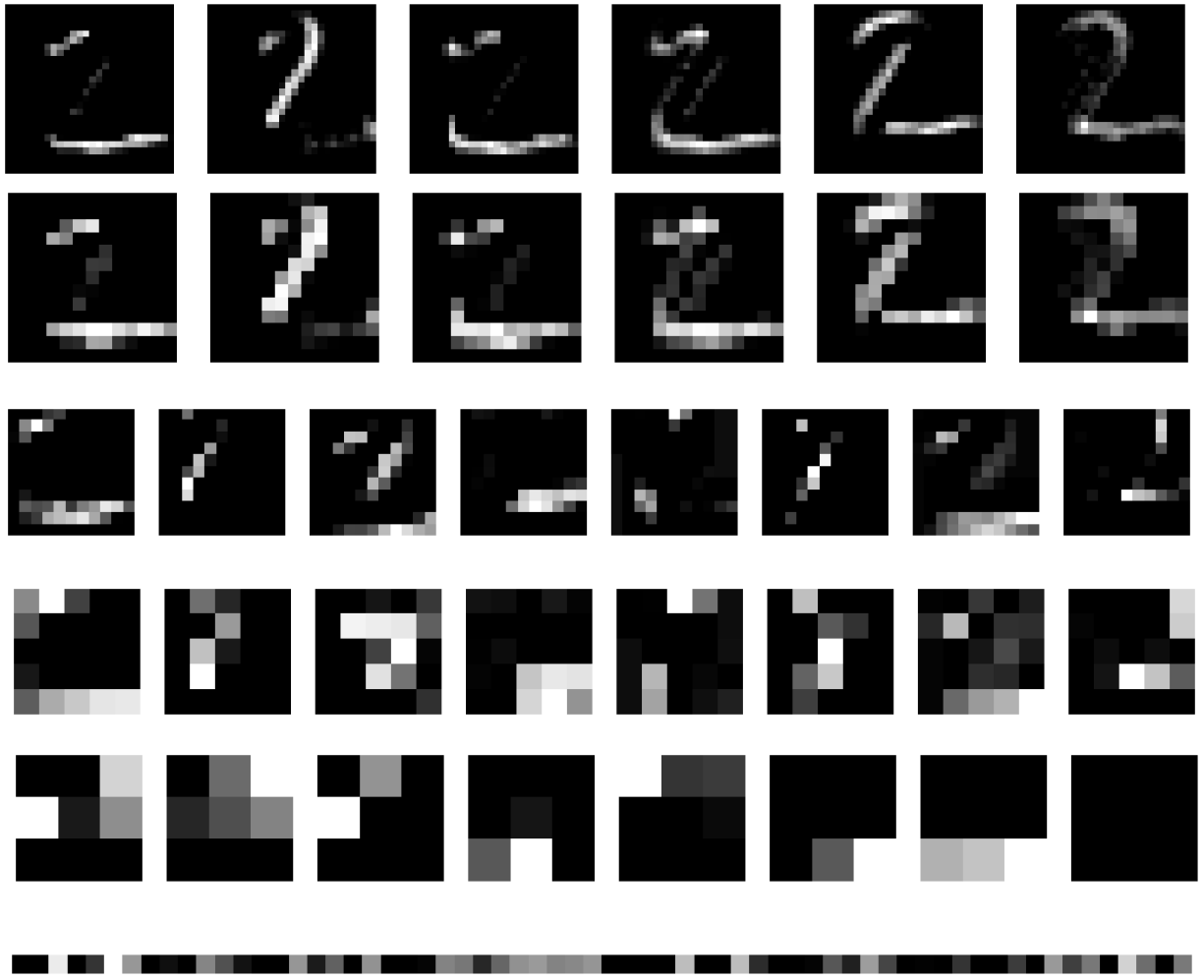
Flatten. 펼치기


-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

컨벌루션 이미지 예시

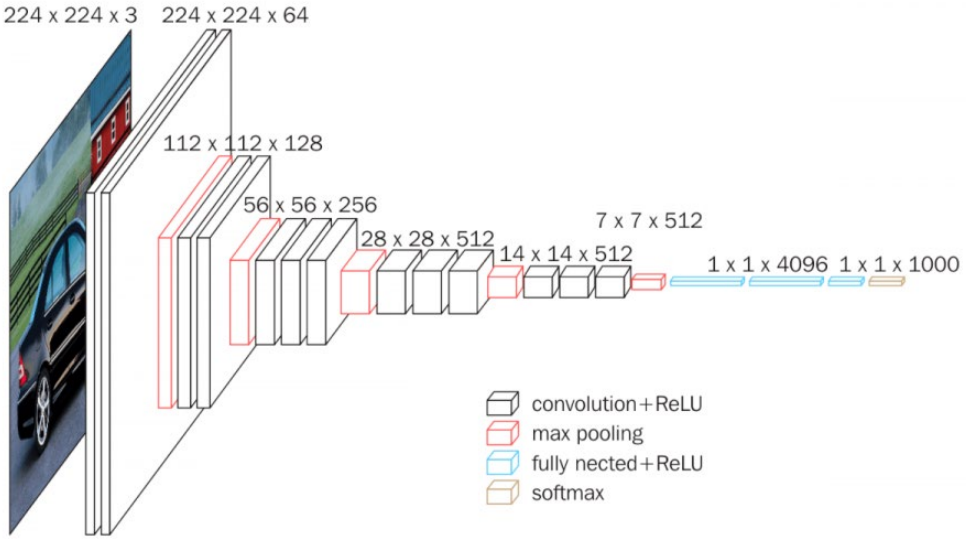


http://cs231n.github.io/convolutional-networks/

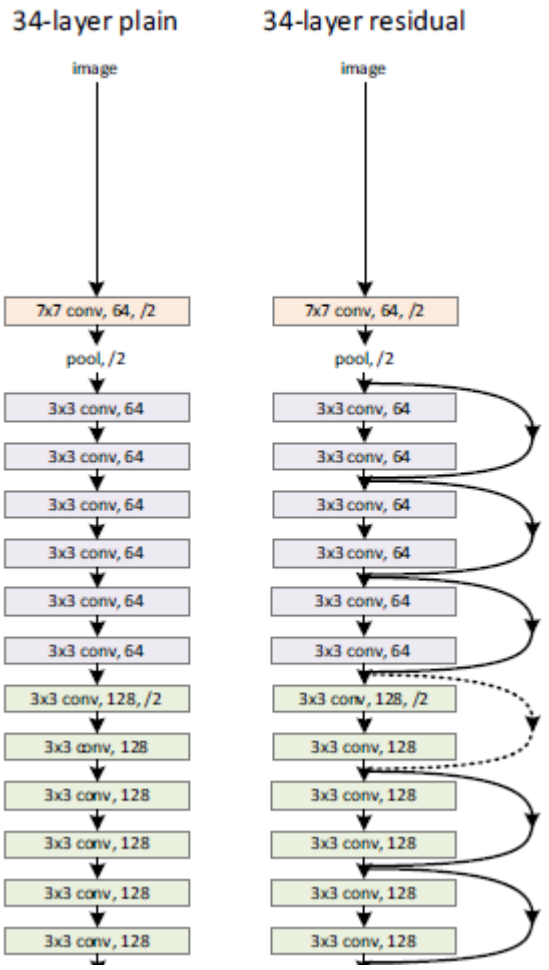


VGG net

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

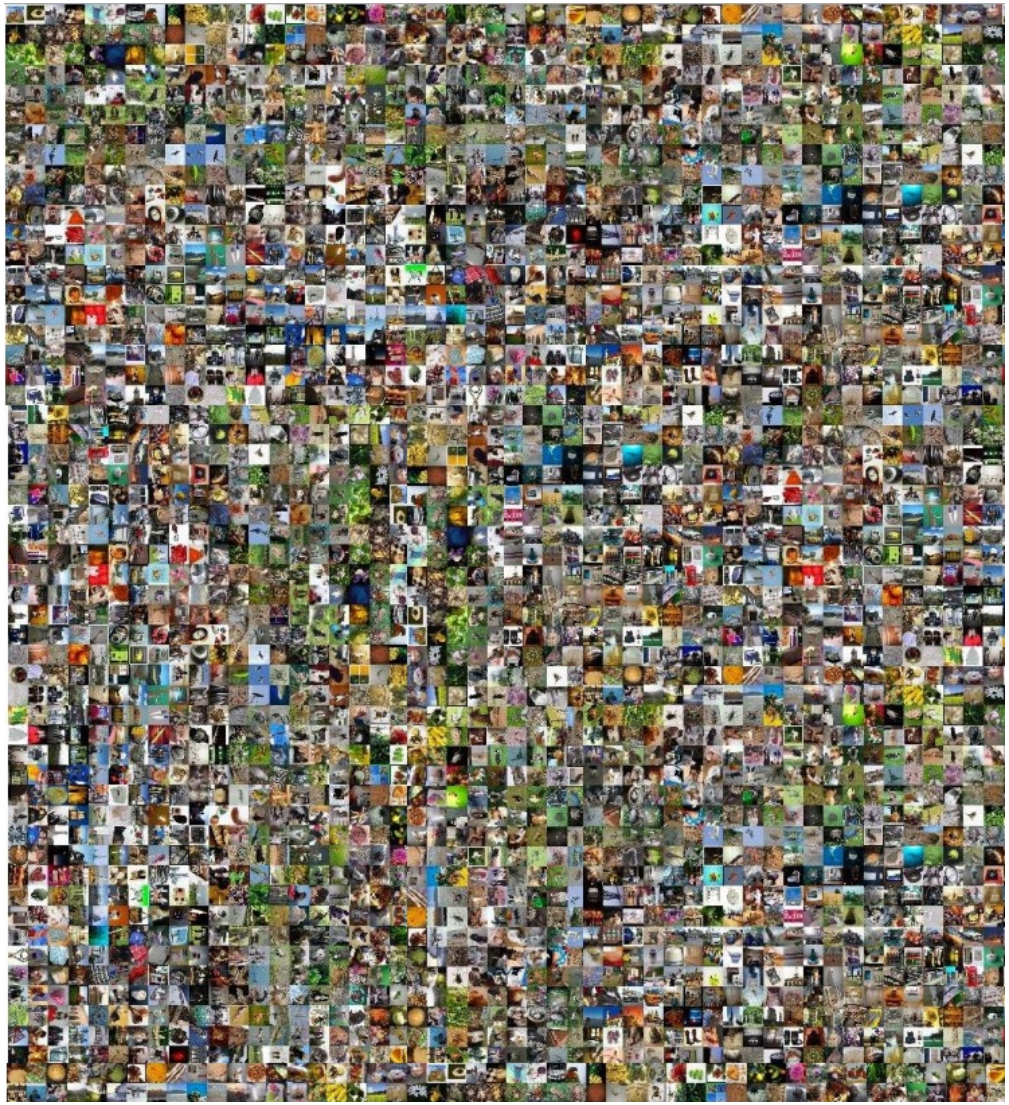


Resnet – 건너 뛰면서 깊게



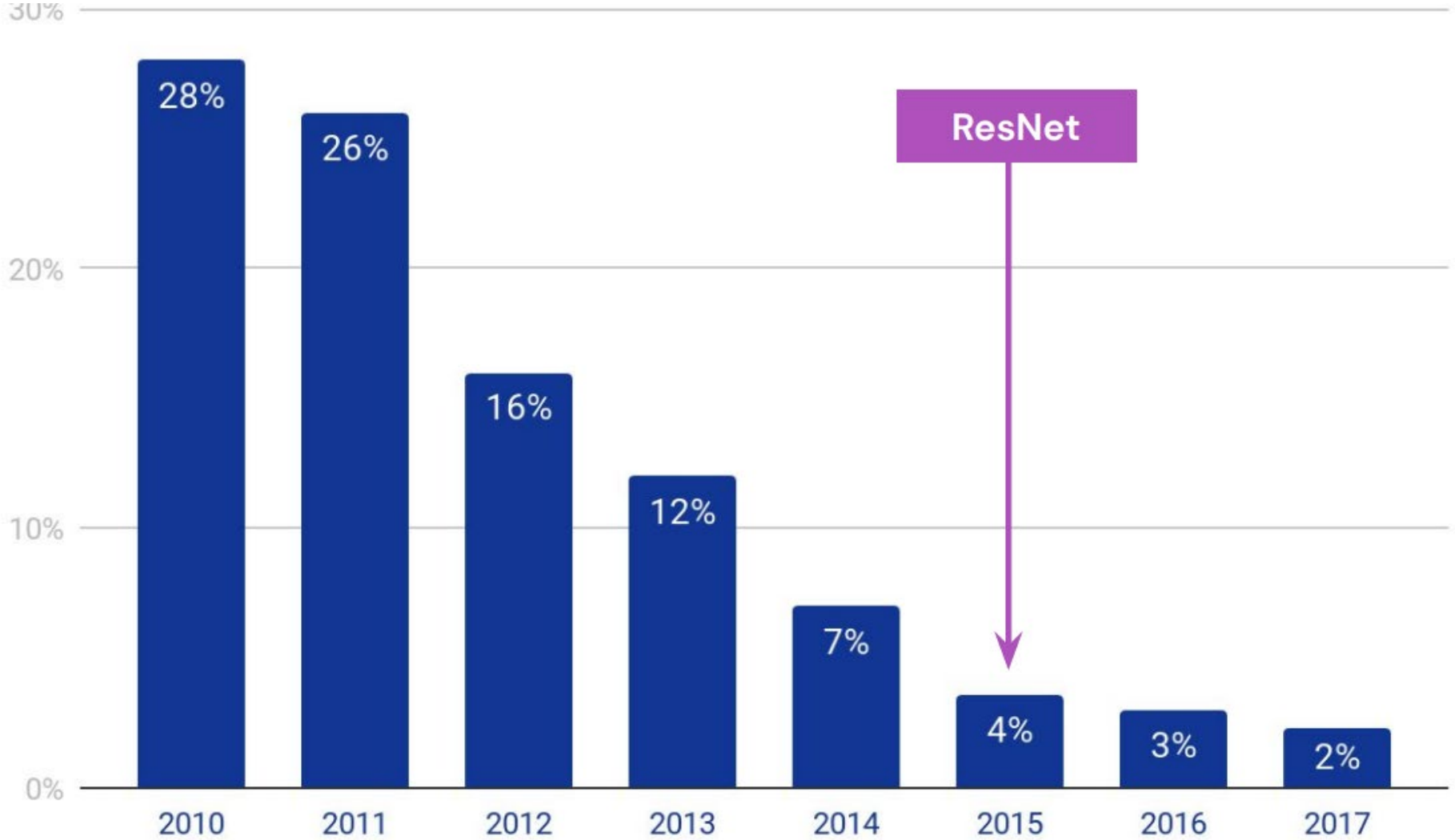
The ImageNet challenge

- Major computer vision benchmark
- Ran from 2010 to 2017
- 1.4M images, 1000 classes
- Image classification



Reference: Deep Mind Deep Learning Lecture

The ImageNet challenge



Reference: Deep Mind Deep Learning Lecture

Mask R-CNN

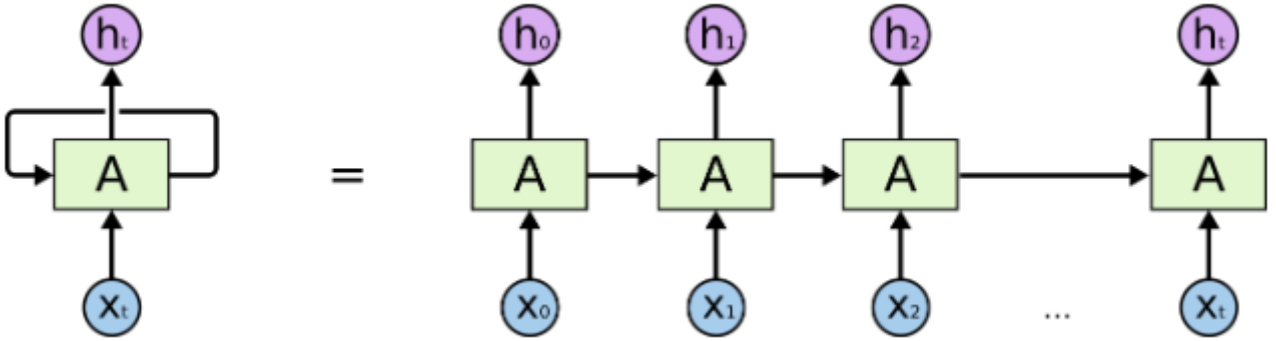
Reference: Stanford CS 231n

Recurrent Neural Networks

Sequence data

예시>문장 읽기

- 하나의 단어만으로는 전체 문장을 이해할 수 없음
- 이전의 단어와 현재의 단어를 조합하면서 이해해 나감(시계열 : time series)
- CNN 구조에서는 처리 불가능



Recurrent Neural Network

- X의 시계열자료를 각 시점에서 재귀식에 적용함

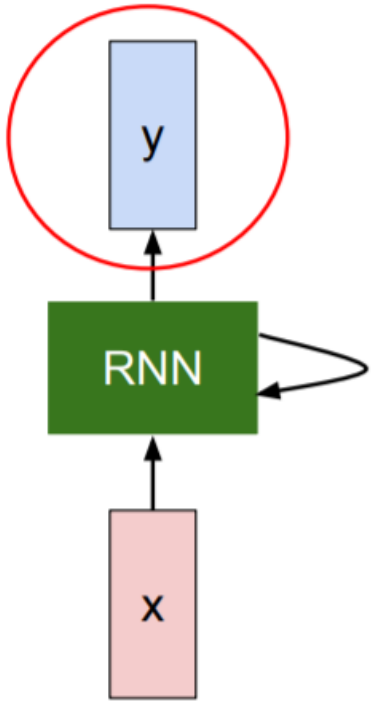
$$h_t = f_W(h_{t-1}, x_t)$$

new state

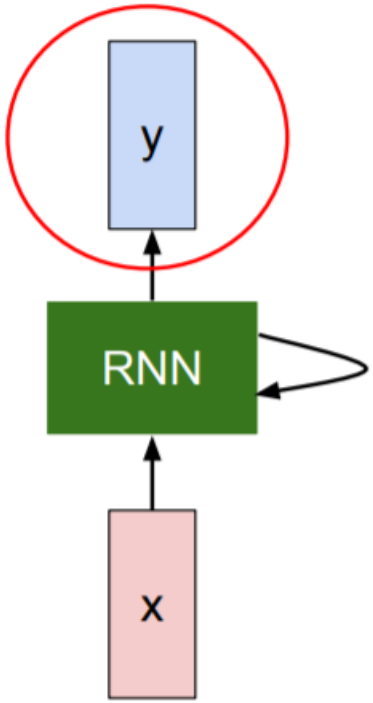
old state

input vector at some time step

some function with parameters W



(각 시점마다 동일한 함수 fw 적용)



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

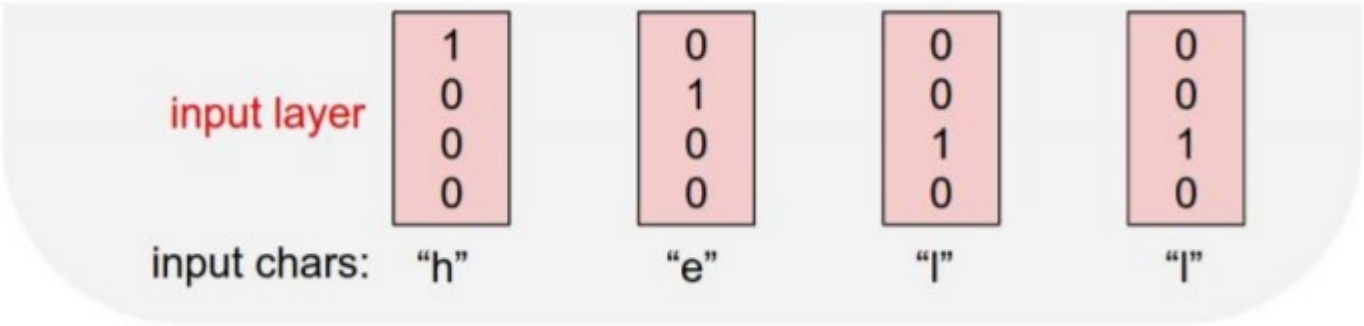
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

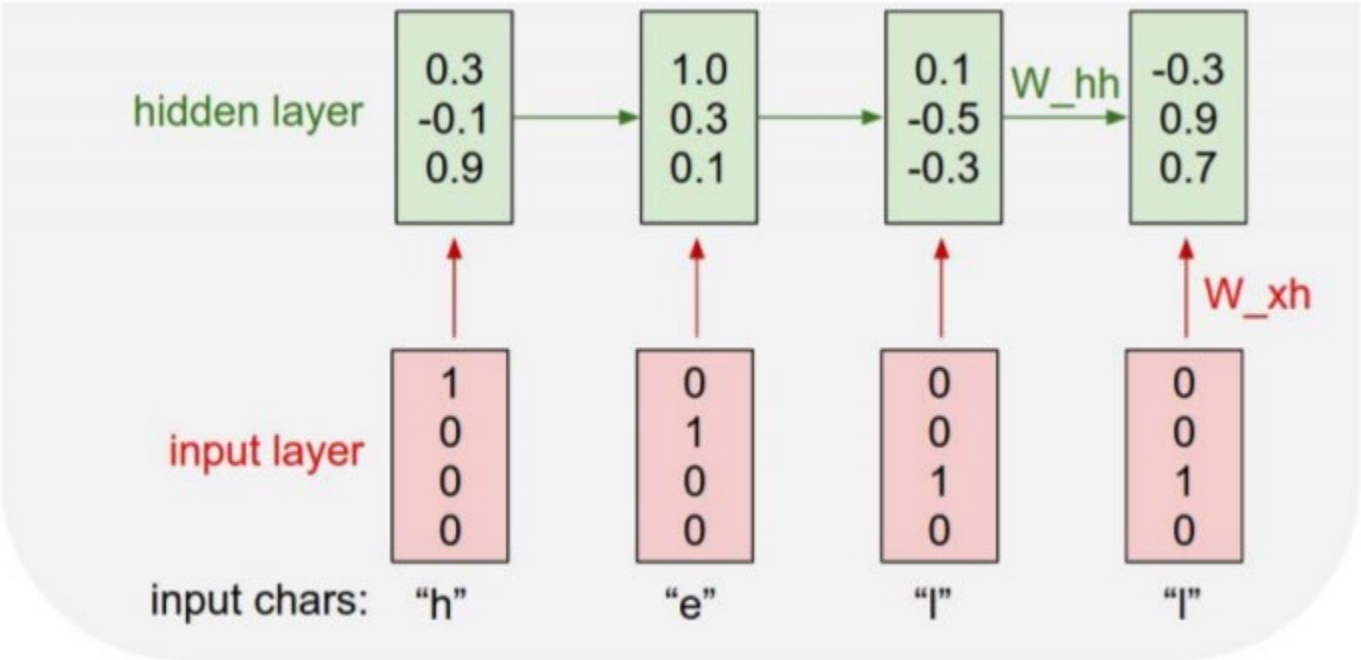
Example: Character-level Language Model

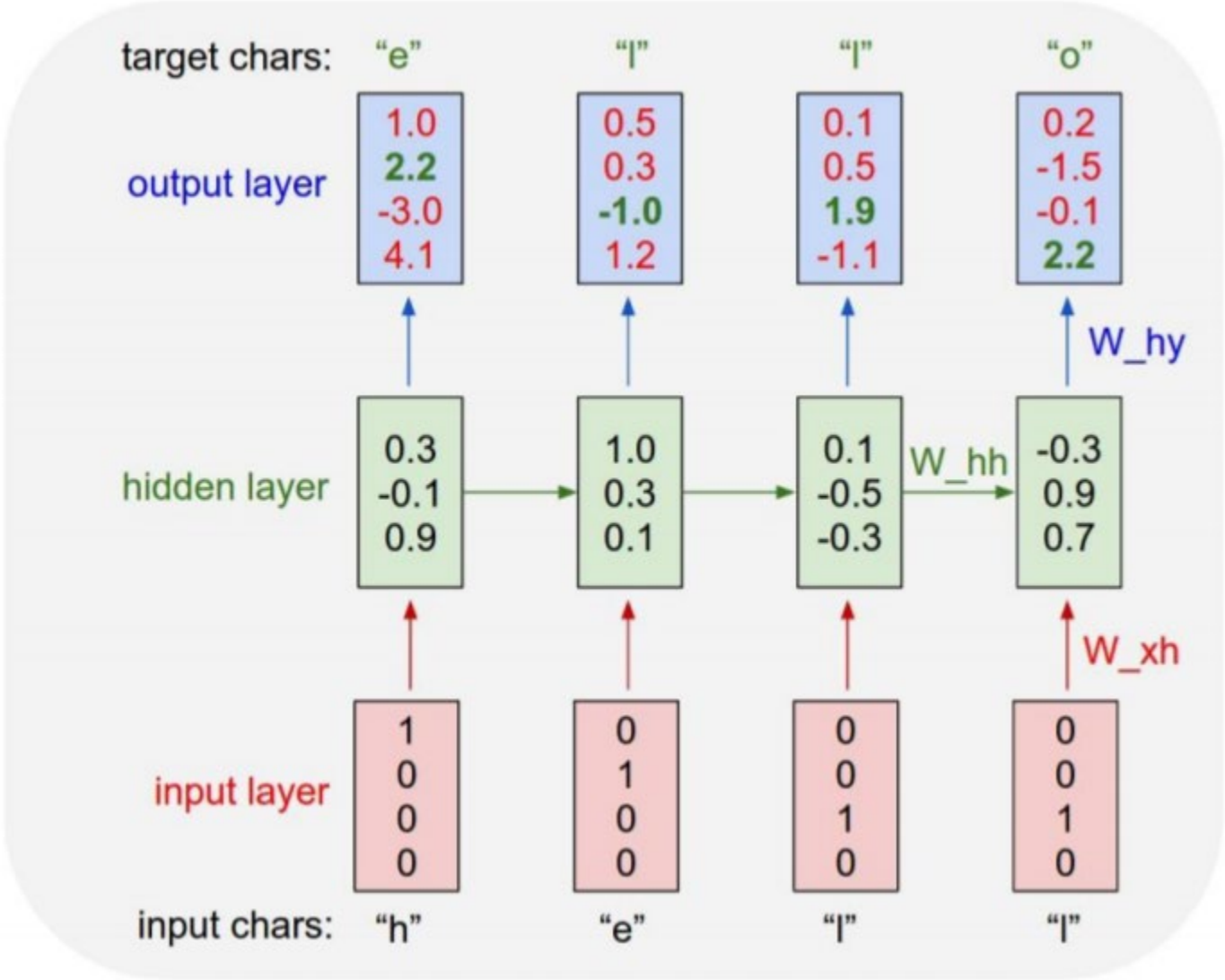
Vocabulary: [h,e,l,o]

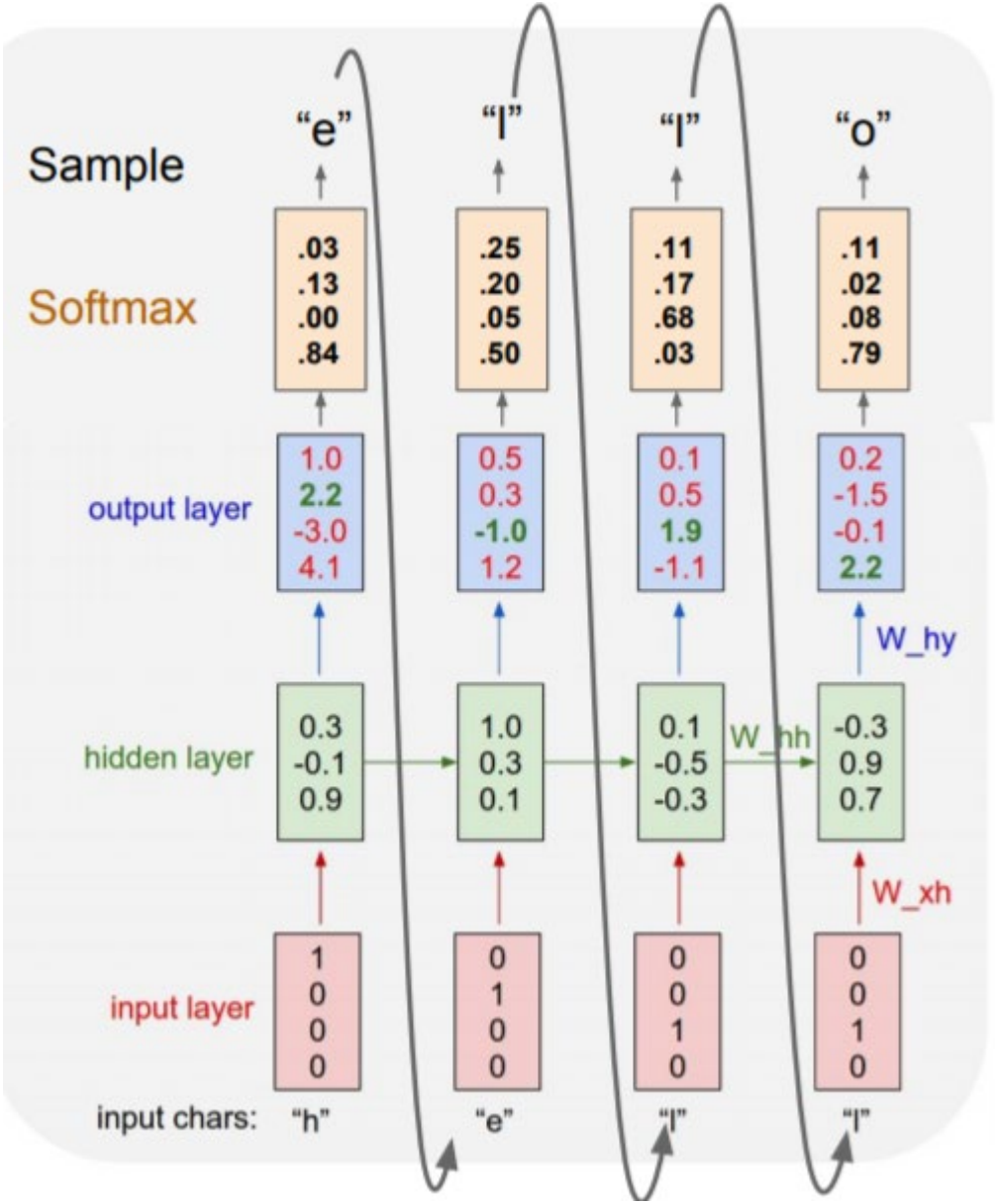
Example training sequence: "hello"



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$




target chars:
"e"
"l"
"l"
"o"
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"


Sample
Softmax
output layer
hidden layer
input layer
input chars:
"h"
"e"
"l"
"o"
"e"
"l"
"l"
.03
.13
.00
.84
.25
.20
.05
.50
.11
.17
.68
.03
.11
.02
.08
.79
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
-0.3
0.9
0.7
W_hy
W_hh
W_xh
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0

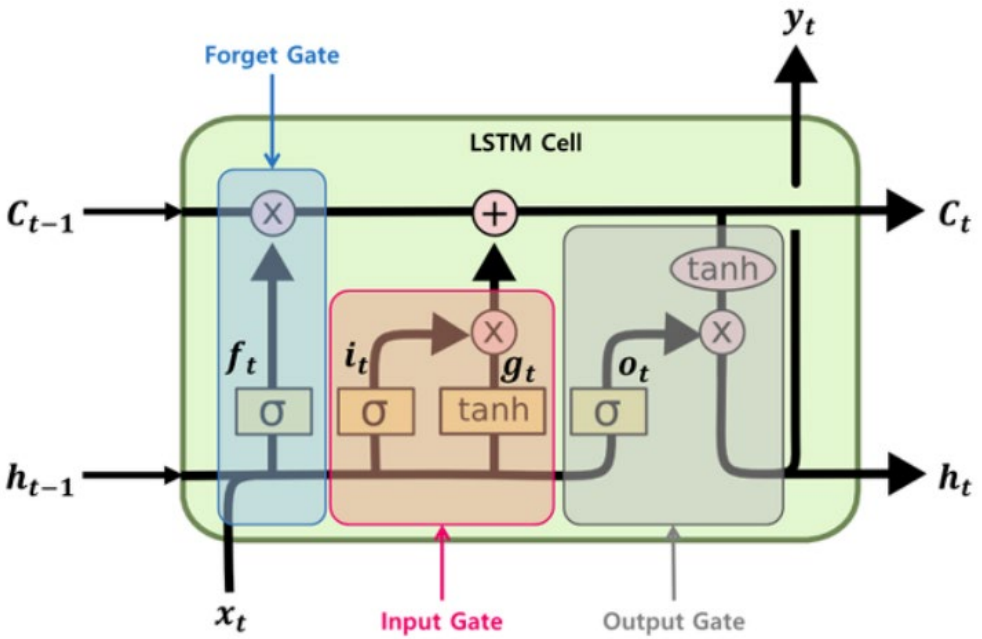
LSTM(1997)

- Gradient 소멸(vanishing)의 문제
- Gradient 폭주(exploding)의 문제

- 단순한 RNN에서는 시계열/시퀀스 데이터의 길이가 길어질 수록 곱셈의 횟수가 증가하면서 작동이 잘 안됨(동일 위치의 가중치)

- 시계열 데이터에 있는 이전 상태의 "기억"에 대한 효과를 놓치기 쉬움
- 먼 시간대 사이의 패턴 포착에 어려움

- Long Short Term Memory



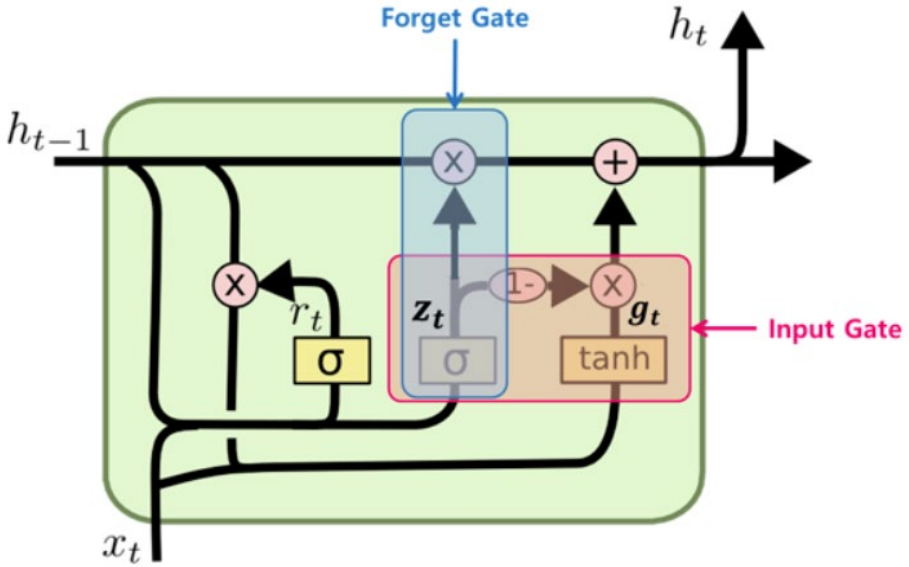
- 기억셀(memory cell)의 추가 (C)

망각게이트(forget gate) : 과거의 정보를 몇% 기억할 것인가? (완전차단/기억)

입력게이트(input gate) : 현재의 정보를 몇% 기억할 것인가? 과거의 정보와 결합

출력게이트(output gate)

GRU(2014)

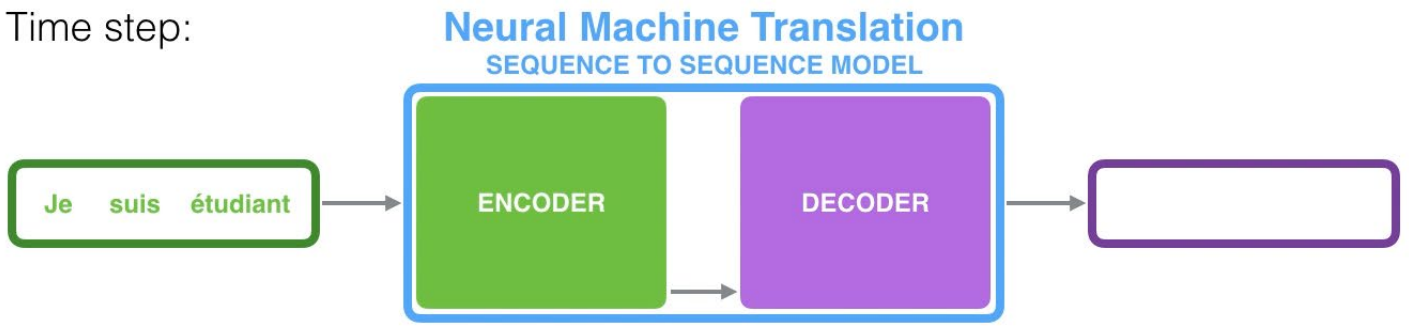


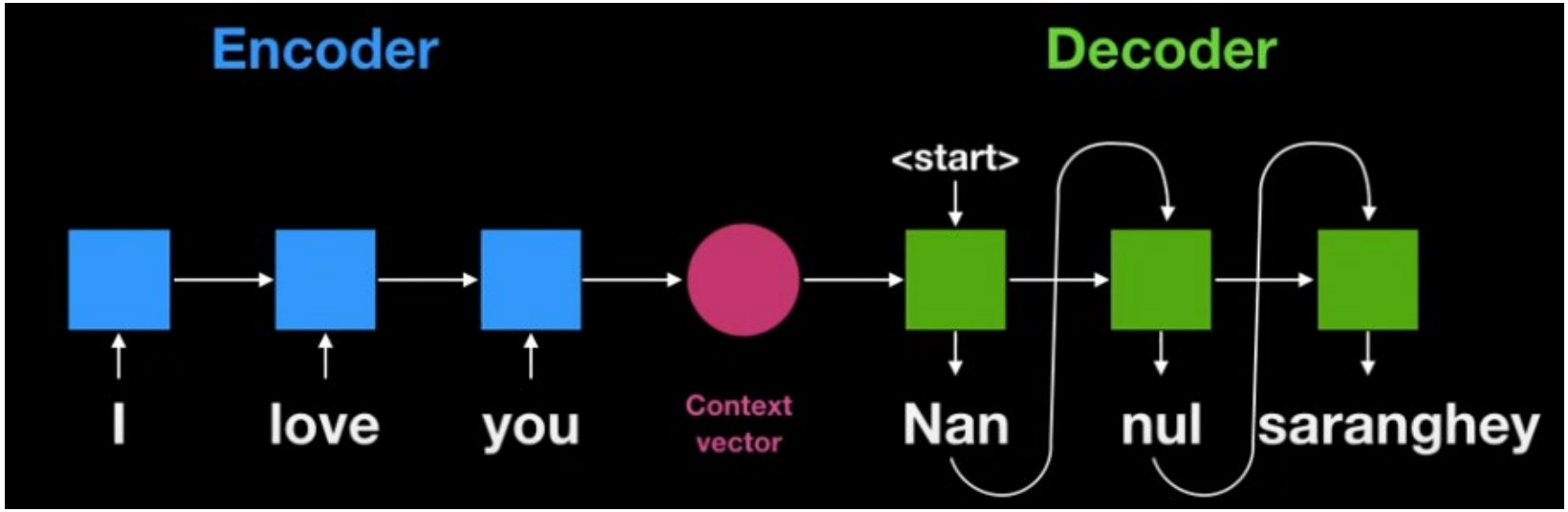
- Gated Recurrent Unit
- LSTM과 유사하며 연산이 더 간단함
- Cell state 없음

Transformer

기계번역 알고리즘 : Seq2seq

LSTM이나 GRU를 기반으로하여 encoder part와 decoder part로 구성된 모형





Seq2seq+Attention architecture의 한계

- Seq2seq 기반 방법의 단점
- 순차적인 계산에 의한 문제점 => 시간!
- 예측에 필요한 계산 시간 (순차적으로 계산하므로)
- 학습에 필요한 시간 (BPTT)

- RNN에서 순차적인 연결고리를 끊어버린다면?
- 병렬적인 계산이 가능해질 것
- Transformer (Vaswani et al., 2017)

Transformer

- 기계 번역을 위해 만들어진 모형.

- 순차적인 연결고리를 모두 끊어버림.
- 계산의 병렬화가 가능해짐.

- 순서의 표현을 하기 위해 positional encoded vector를 활용.

- Self-attention 기법을 사용하여 문장 내의 시멘틱 정보를 추출

- Architecture: Encoder + Decoder

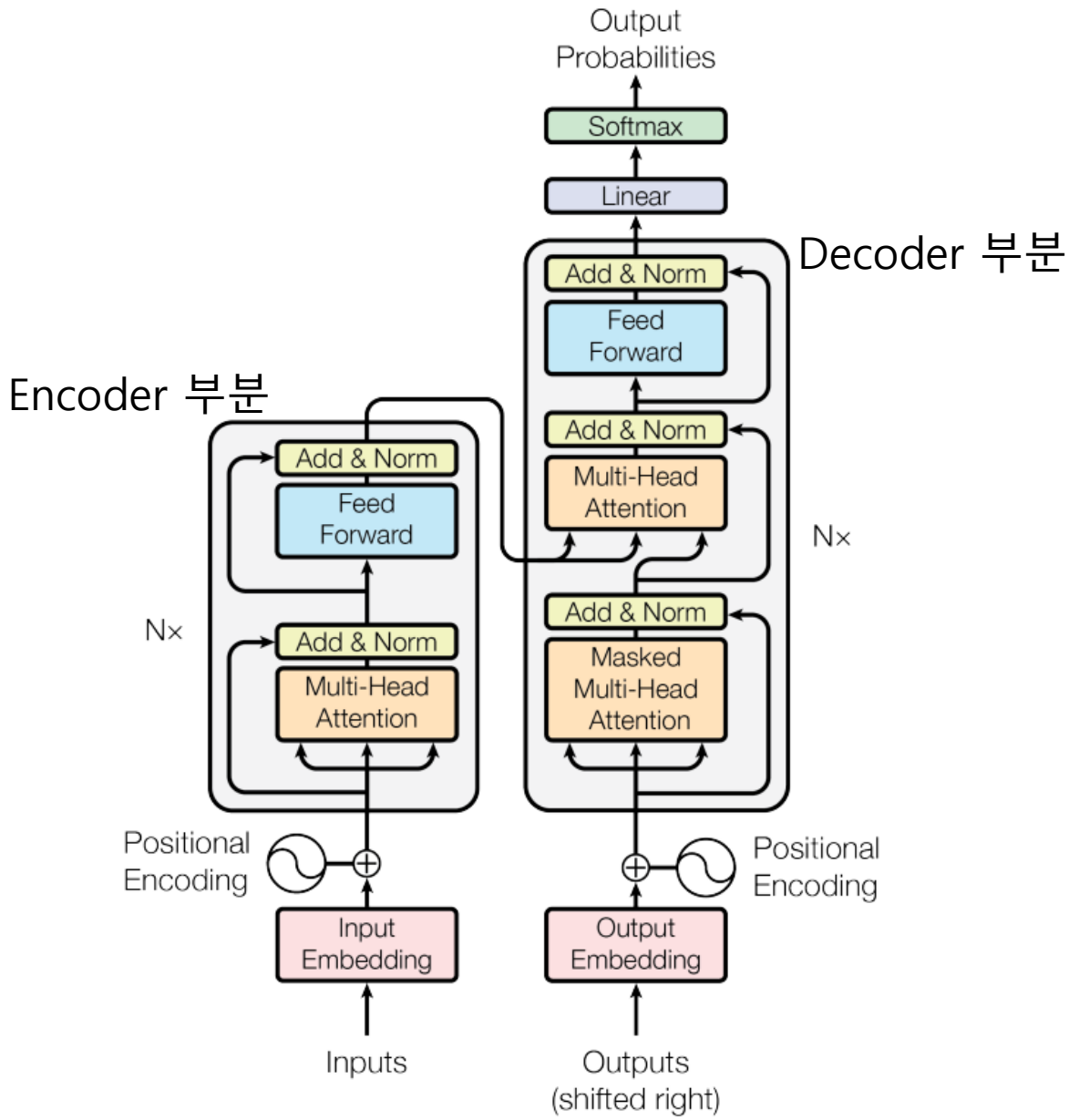


Figure 1: The Transformer - model architecture.

Post-Transformer

- Transformer의 Encoder 만을 이용
- BERT, RoBERTa, etc.

- Transformer의 Decoder 만을 이용
- GPT-1, GPT-2, etc.

- Encoder + Decoder
- XLNet, BART, etc.

Attention!!!

- “Attention is all you need”, Vaswani et al., 2017
- Transformer는 Self-attention 메커니즘 기반의 자연어 처리 분야 언어 모형

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research

Aidan N. Gomez*† University of Toronto

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Self-attention mechanism

- Self-attention 이란?

- 각 문장에서 집중해야 하는 단어 (가중치가 높은 단어)는 무엇인지,
  각 단어가 집중하고 있는 문장 내에 단어가 무엇인지 표현할 수 있음.

- 단어마다 어떤 단어들과 관련성이 높은지를 문장 내에서 찾아 반영
  => self-attention!

- 기존 attention은 번역될 문장과의 관련성을 반영했음.

- 단순한 행렬 곱(matrix multiplication)을 통하여 연산(내적).

Self-attention mechanism

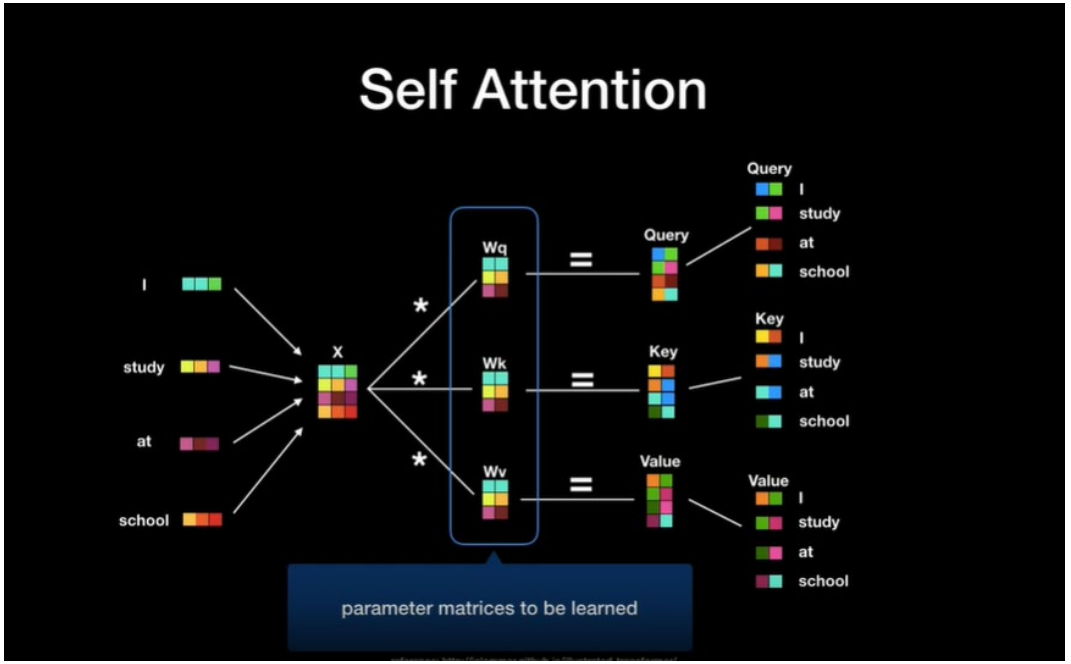
- ex: “The animal didn’t cross the street because it was too wide.”
- it 이라는 단어가 문장 내에 어느 단어와 가까워야 하는가?
- animal? street? 혹은 두 단어 모두 가능성이 있는가?

Self-attention mechanism
(source: http://jalammar.github.io/illustrated transformer/)

Self-attention mechanism

1. Query, Key, Value 벡터 생성

- 모든 단어의 input embedding을 선형 변환
- Query, Key, Value 벡터 생성 (64차원)
- 선형 변환을 위한 모수: (Wq, Wk, Wv)



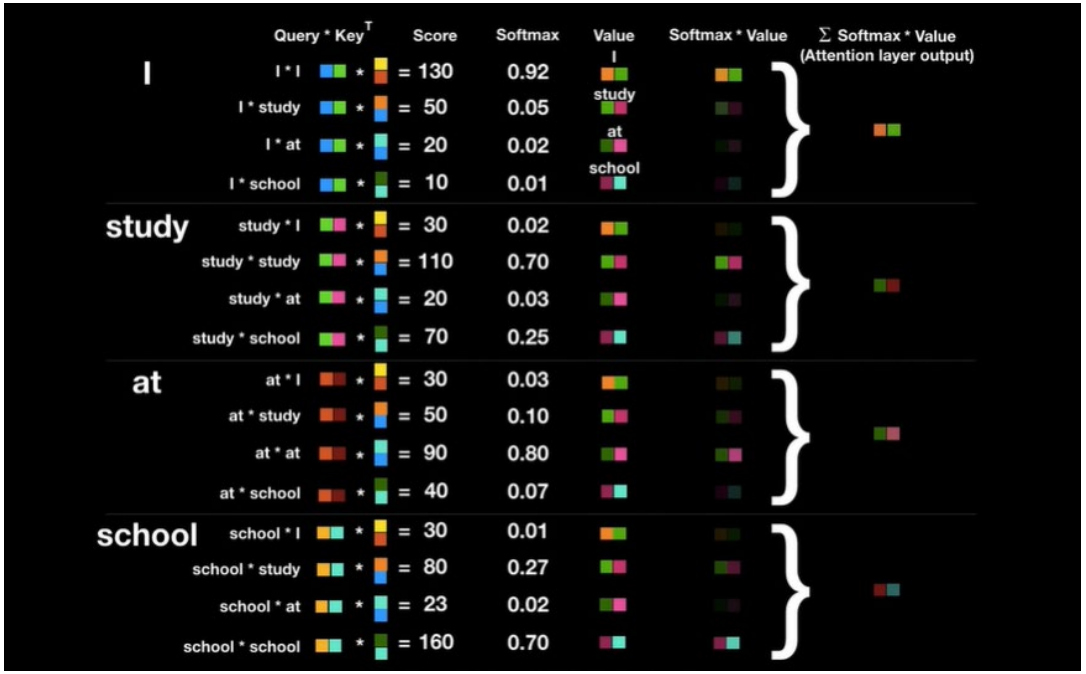
(source: https://www.youtube.com/watch?v=mxGCEWOxfe8)

Self-attention mechanism

2. Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Query와 Key를 내적하여 score 계산 – 연관있는 단어의 score가 큼
- Softmax 함수 통과 후 Value들과 내적하여 각 단어마다 attention을 반영한 벡터를 생성. (Attention layer output)

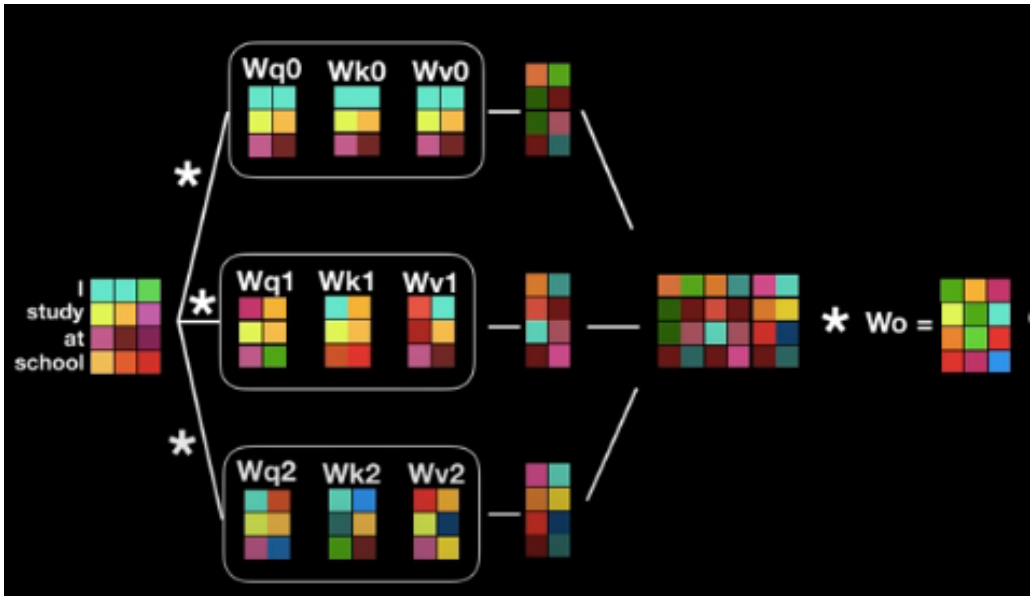


(source: https://www.youtube.com/watch?v=mxGCEWOxfe8)

Self-attention mechanism

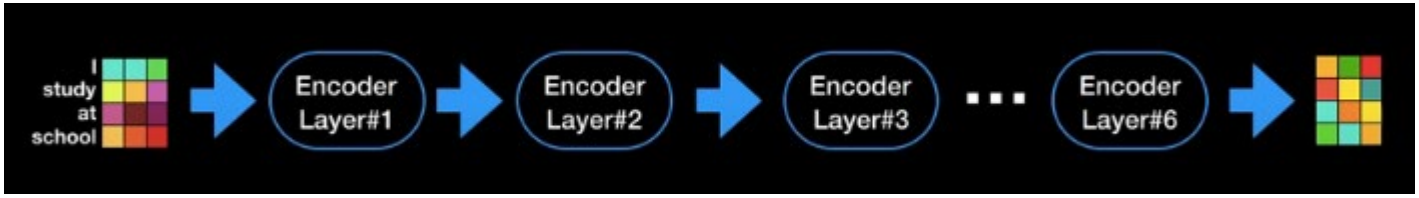
3. Multi-Head Attention

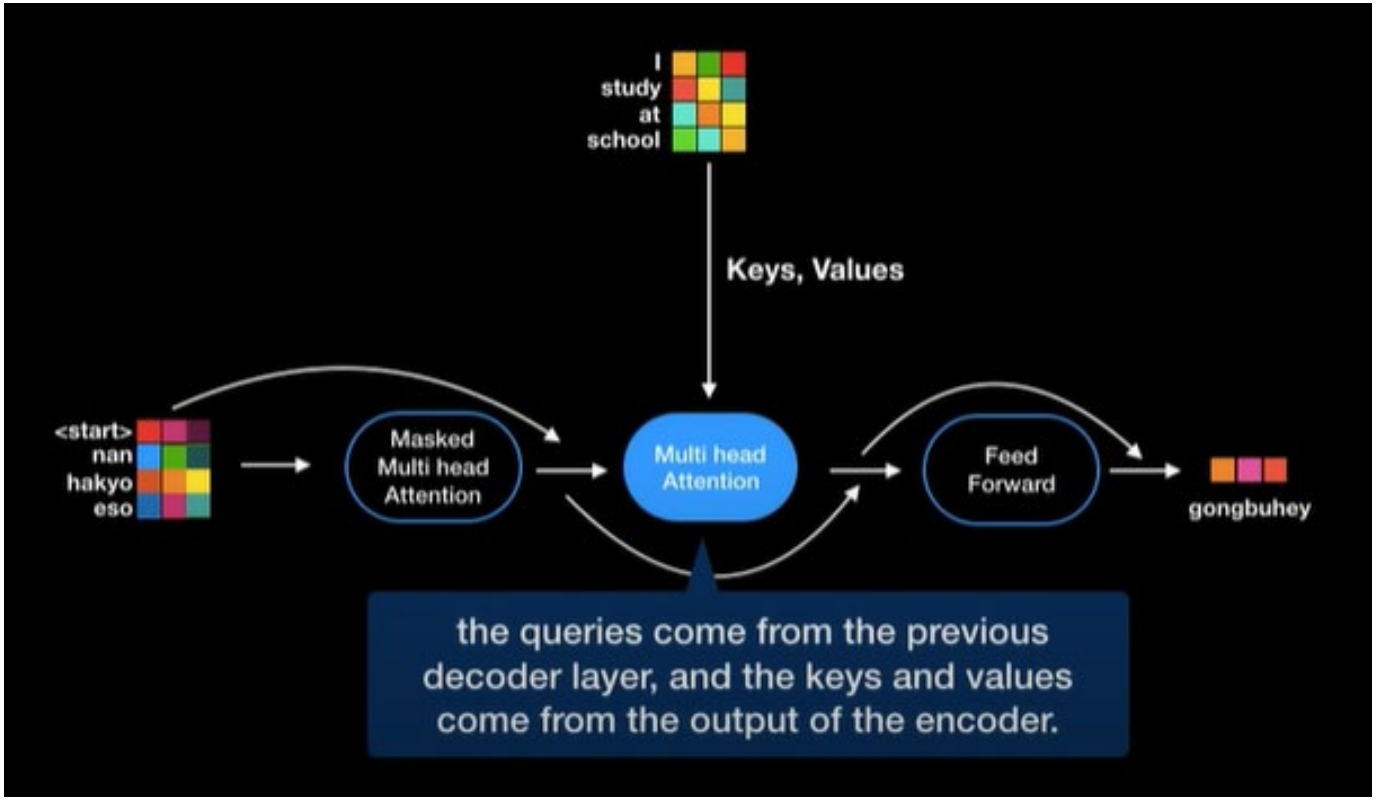
- 여러 개의 attention 연산을 동시에 수행. (그림의 경우 3개의 attention이 수행되고 있음, 실제로는 8개를 사용)
- 각 attention으로부터 나온 결과값들을 concatenate.
- 적절한 선형 변환을 통해 원 문장 embedding과 같은 차원으로 만듦.



(source: https://www.youtube.com/watch?v=mxGCEWOxfe8)

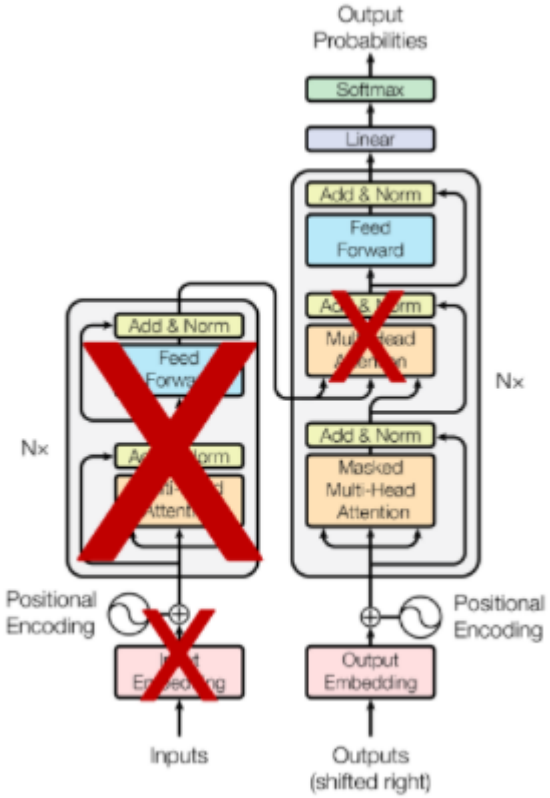
Encoder - Decoder

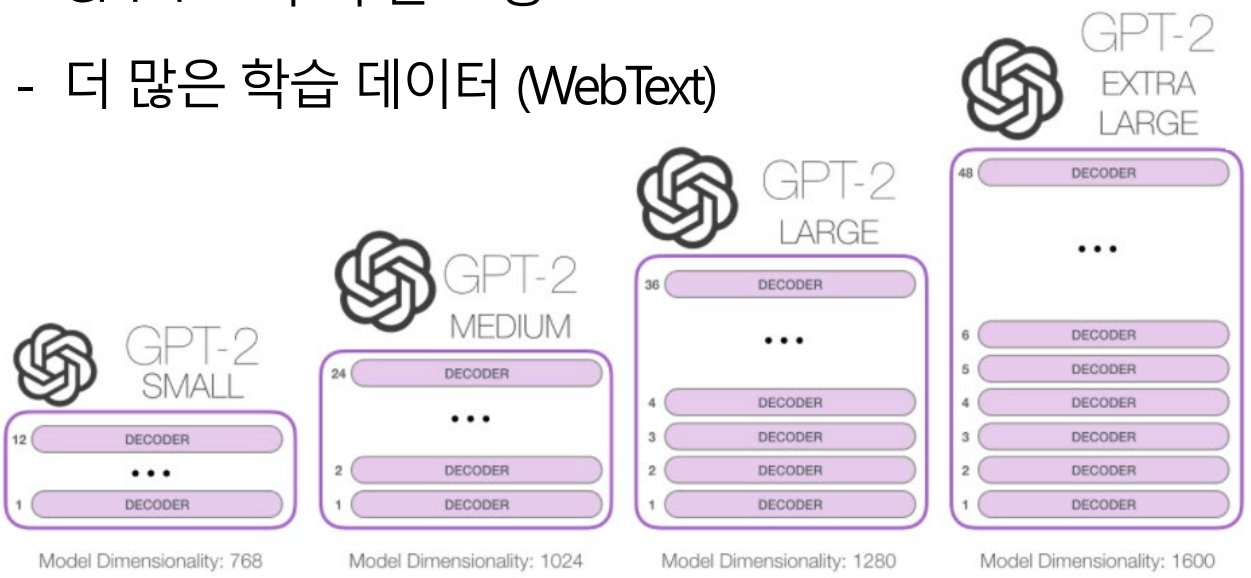




GPT (Generative Pre-Training)

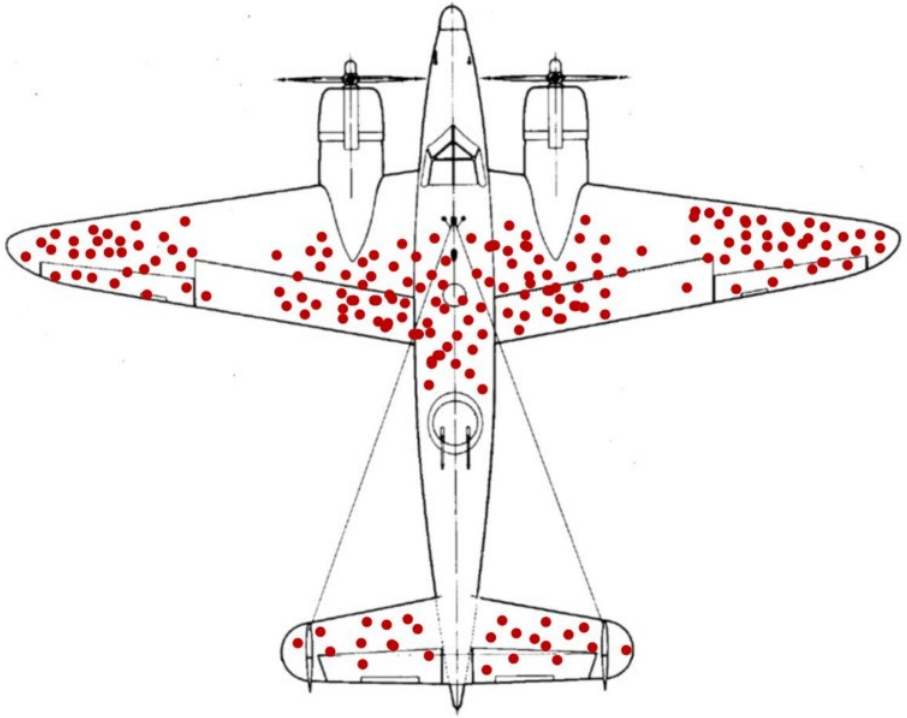
- GPT-1 (Radford et al., 2018)
- Transformer의 Decoder 기반 언어 모형
- 12개의 decoder 사용.
- GPT-2 (Radford et al., 2019)
- GPT-1 보다 더 큰 모형
- 더 많은 학습 데이터 (WebText)





인공지능의 시대의 데이터

- 새로운 알고리즘의 개발 보다는 GPU에 집중하는 현상 발생
- GPU의 성능향상 및 수요 증가
- 양질의 데이터 확보를 위한 노력이 필요 (선택편향)




Thank
you!