딥러닝 강좌 기초

Written by 박 철(e2g1234@naver.com)

목차

- ✓ 딥러닝의 개요 및 역사
- ✓ Perceptron
- ✓ Neural Network
- ✓ CNN - Convolutional Neural Network
- ✓ LeNet 5
- ✓ ALEXNET
- ✓ GoogLeNet
- ✓ ResNet

https://hunkim.github.io/ml/

딥러닝의 개요 및 역사

Intelligence

경험에 맞게 조정 된 불완전한 지식에 기초한 합리적인 판단 (추론)을위한 역량.

The capacity for rational judgment (inference) based on imperfect knowledge, adapted with experience.

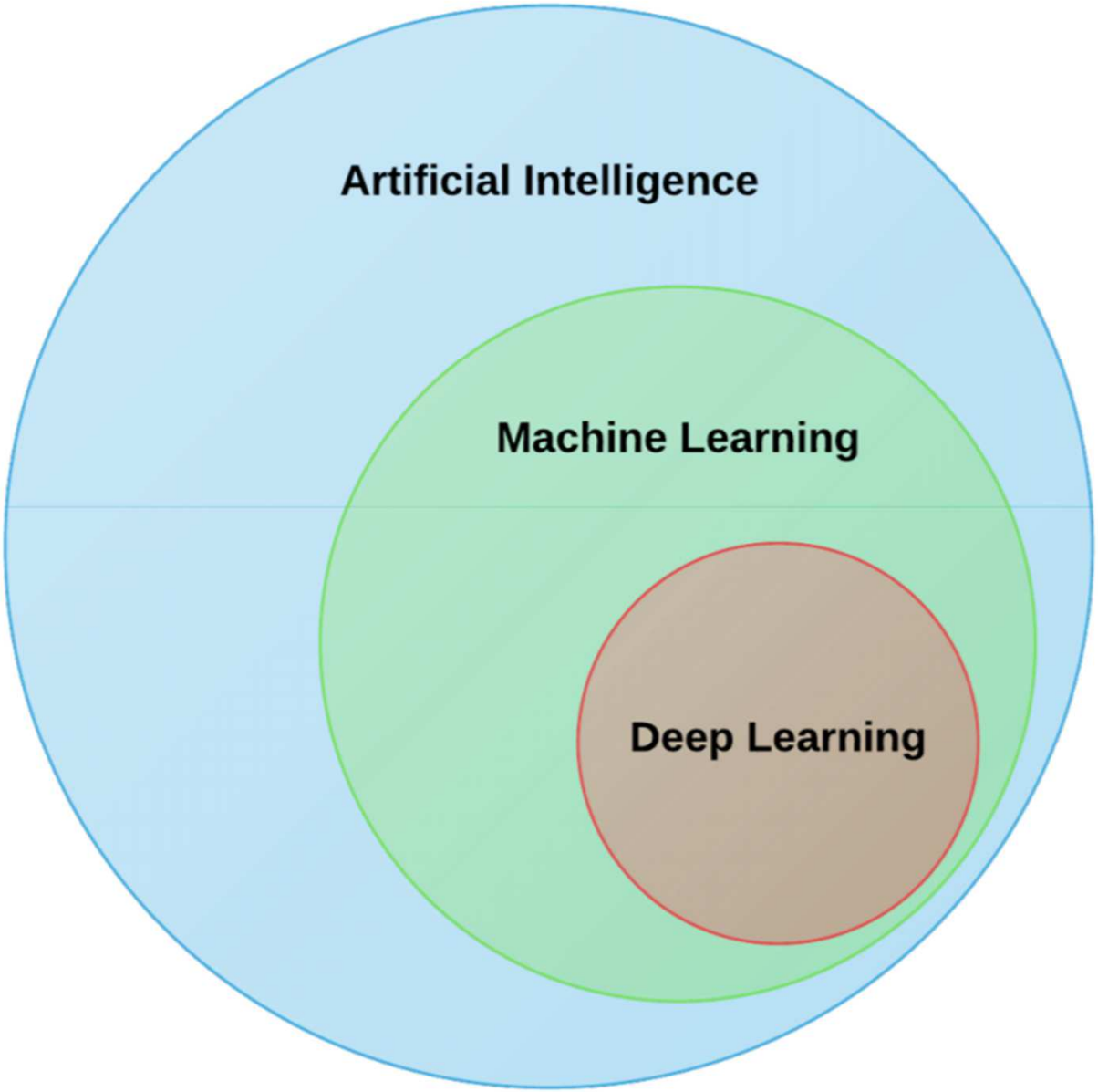
학습은 경험했던 데이터의 응축 (통계 모델)을 형성한다.

Learning is forming a condensate (statistical model) of Experienced data.

불확실성이 없는 지능은 없습니다.

There is no intelligence without uncertainty.

인공지능의 분류



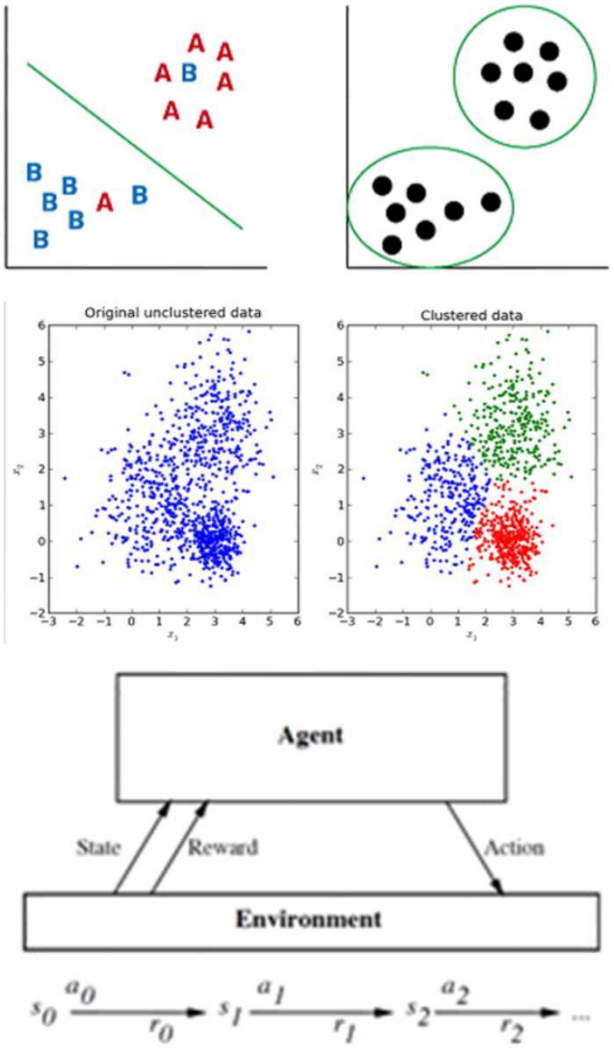
학습방법

- 지도학습
- 비지도학습
- 강화학습

기계학습의 종류

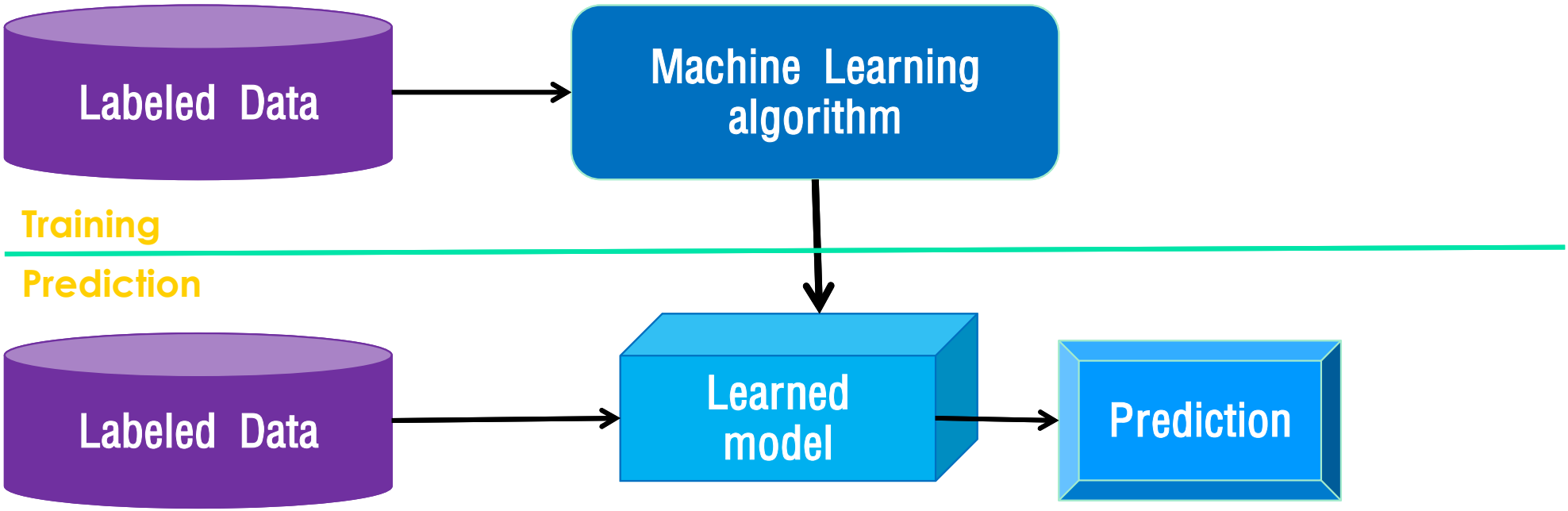
- ✓ 기계학습 기반의 인공지능은 목표 달성 과정을 전문가가 일일이 모델링하고 구현
- ✓ 강화학습 기반의 인공지능은 스스로 현재의 환경을 인식하고 행동하면 목표를 달성해 나감.

- ✓ **Supervised Learning 지도학습**
  - Input 과 labels을 이용한 학습
  - 분류(classification), 회귀(regression)
  - Y = f(x)
- ✓ **Unsupervised Learning 비지도 학습**
  - Input만을 이용한 학습
  - 군집화(clustering), 압축(compression)
  - X ~ p(x) or x = f(x)
- ✓ **Reinforcement Learning 강화학습**
  - Label 대신 reward가 주어짐
  - Action selection, policy learning
  - Find a Policy, p(als)
    - which maximizes the sum of reward



Machine Learning Basics

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed



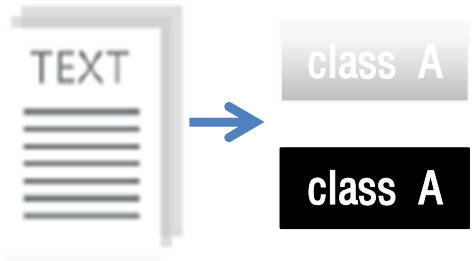
Methods that can learn from and make predictions on data

Types of Learning

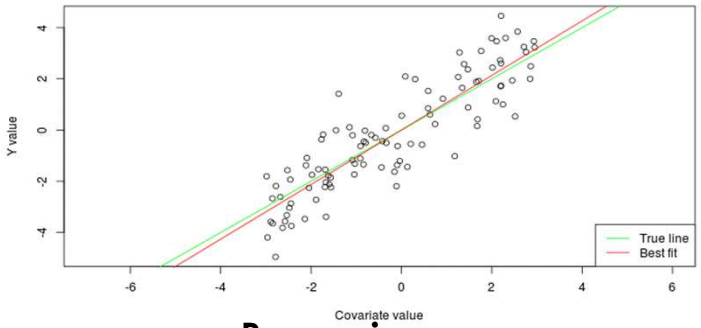
Supervised: Learning with a labeled training set
Example: email *classification* with already labeled emails

Unsupervised: Discover patterns in unlabeled data
Example: *cluster* similar documents based on text

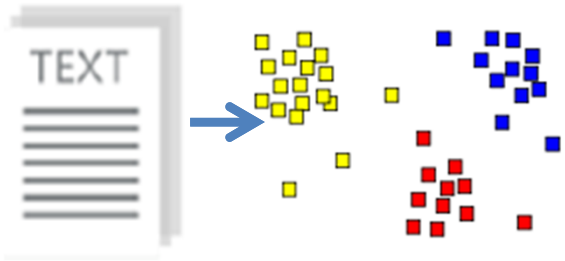
Reinforcement learning: learn to act based on feedback/reward
Example: learn to play Go, reward: *win or lose*



Classification



Regression

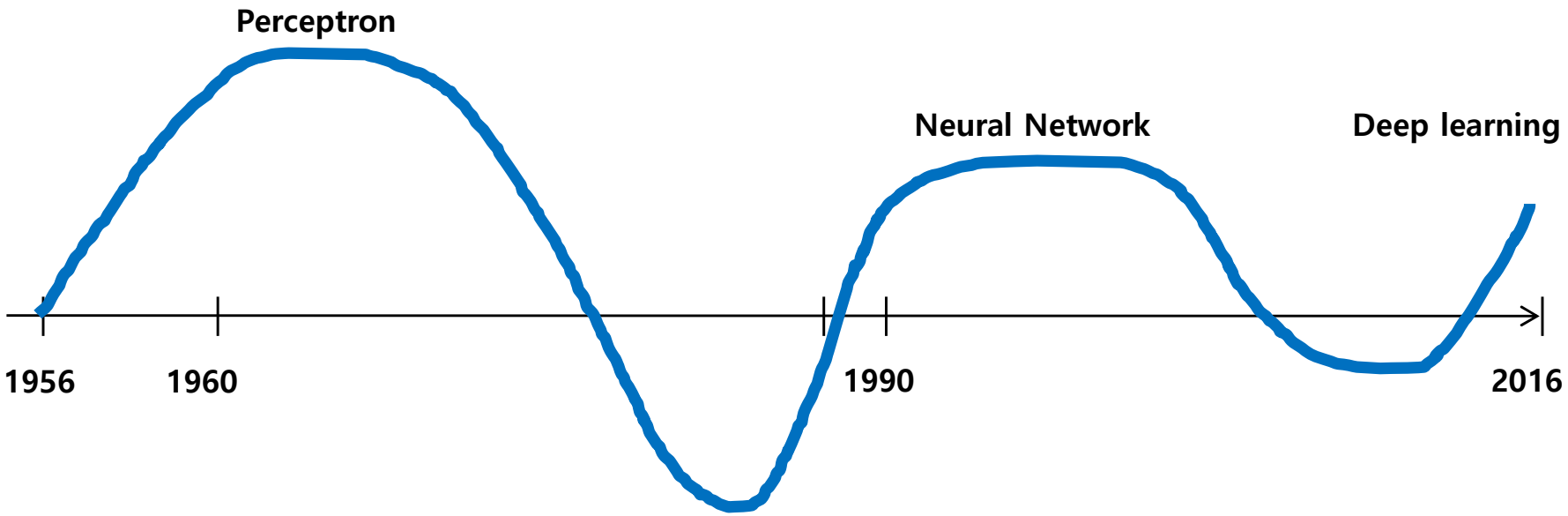


Clustering

Anomaly Detection
Sequence labeling
…

http://mbjoseph.github.io/2013/11/27/measure.html

인공지능의 역사와 특이점



1957년 - 퍼셉트론 Frank Rosenblatt

Hardware implementations
XOR **문제를 해결하지 못함**

1969 minsky and papert

퍼셉트론이 XOR **연산을 할 수 없음을 증명함**
인공지능 연구의 1**차 빙하기의 서막**
다층 퍼셉트론으로 XOR **연산 가능**
: back propagation**에 대한 해답이 없었음**

1986 MLP(Neural Network)

1974, 1982 paul werbos
1986 geoffrey hinton
Back propagation
Vanishing gradient **문제 발생**
– 20**년간 해결하지 못함** – **제**2**의 빙하기**

1998년

Lecun**에 의해 초기** CNN**인** LeNet-5

2006 2007

2006**년 초기화** / 2007**년** Relu
GPU – big data

2012년

ALEXnet 84%

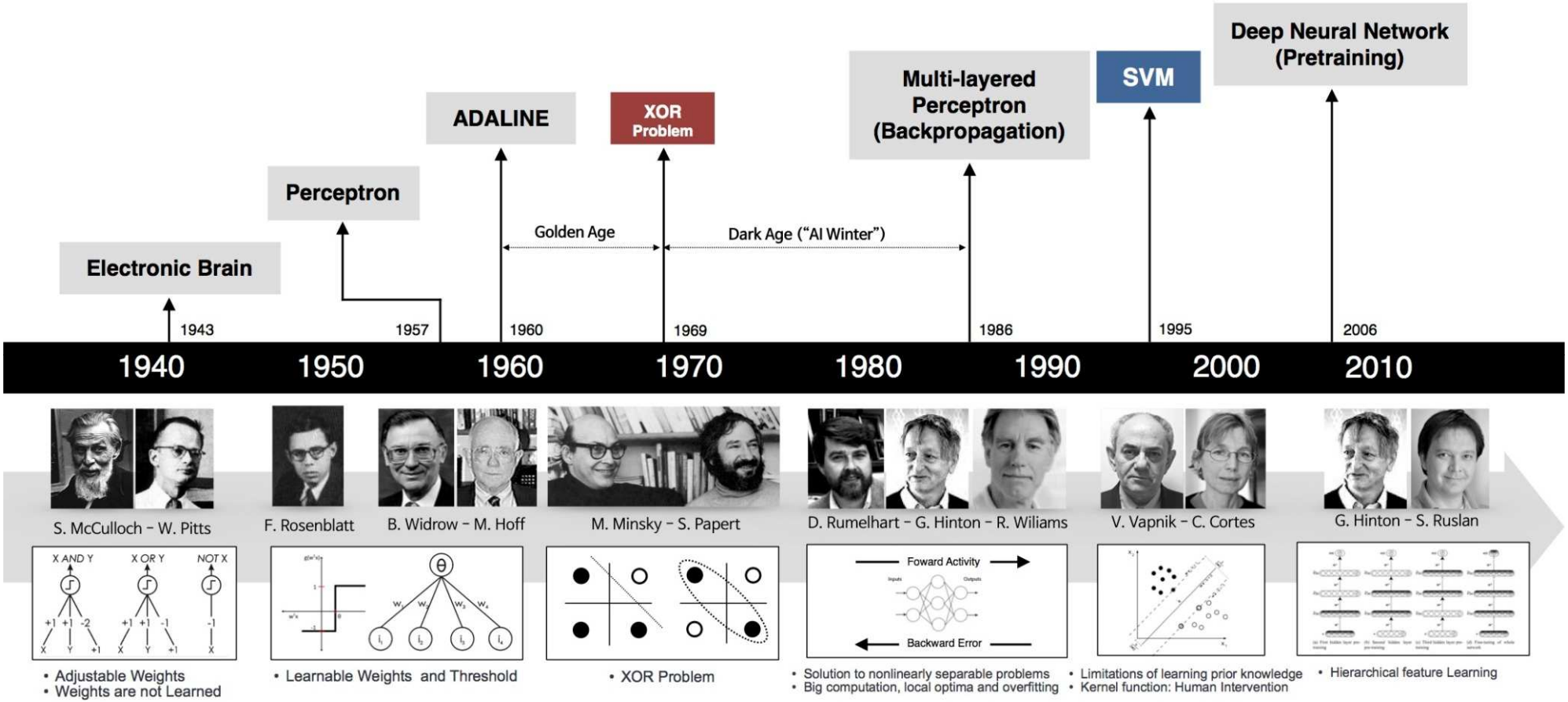
2014년

GoogLe Net VGG

2015년

RESNET

인공지능의 역사


Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

1943년 Artificial Neural Networks

인공신경망
(Artificial Neural Networks : ANNs)이란 개념

1943년 발표 논문 에서 최초로 제안됨
McCulloch, Warren S., and Walter Pitts.
"A logical calculus of the ideas immanent in nervous activity"

McCulloch과 Pitts
인간의 신경 구조를 복잡한 스위치들이 연결된 네트워크로 표현할 수 있다고 제안

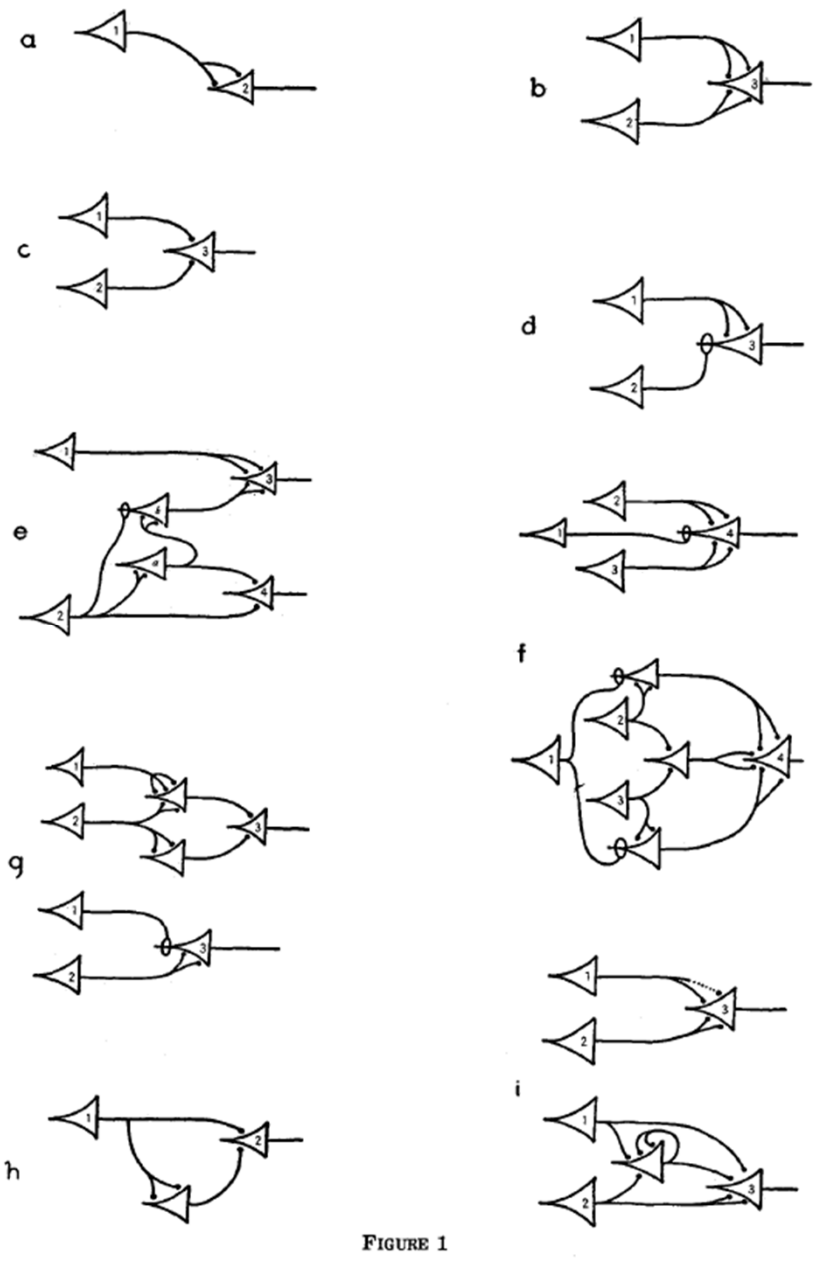


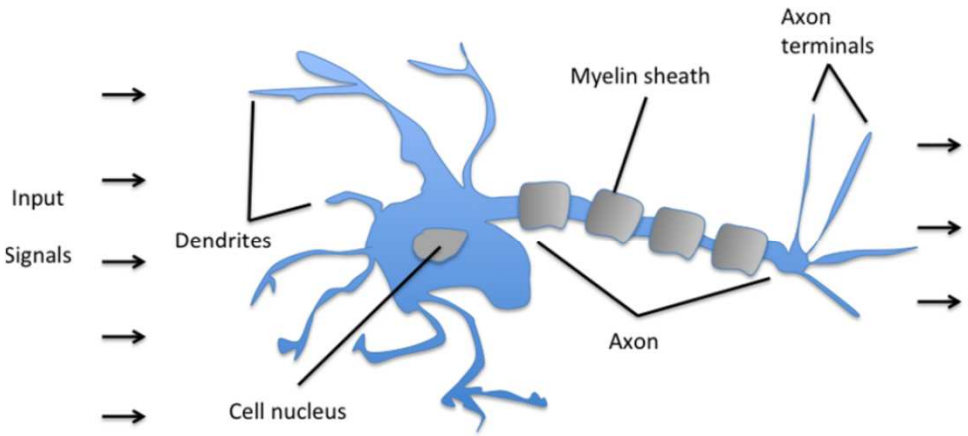
FIGURE 1

1958년 Perceptron

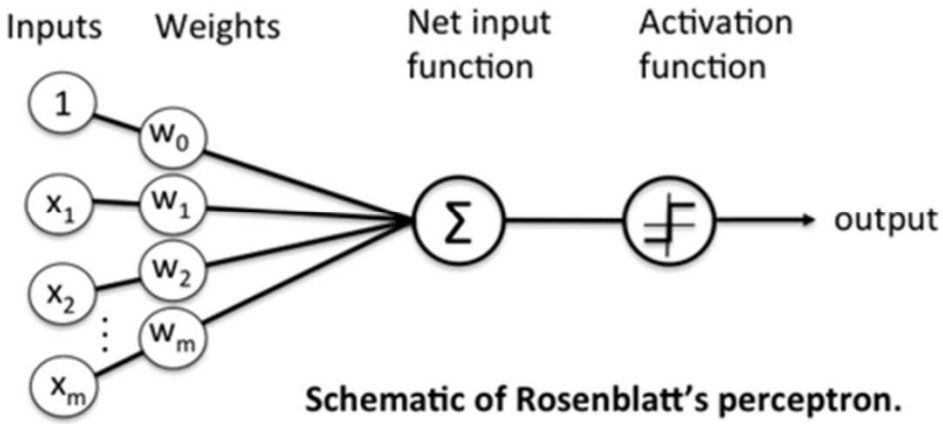
1958년 Frank Rosenblatt이 발표한 논문
"The perceptron: A probabilistic model for information storage and organization in the brain."

퍼셉트론(Perceptron)이라는 선형분류를 수행할 수 있는 피드포워드 뉴럴네트워크를 제안

선형 분류기의 구조
입력과 Weight들의 곱을 모두 더한 뒤 활성함수(Activation Function) –에) 시그모이드(Sigmoid) 함수- 를 적용해서 그 값이 0보다 크면 1, 0보다 작으면 -1을 출력



Schematic of a biological neuron.



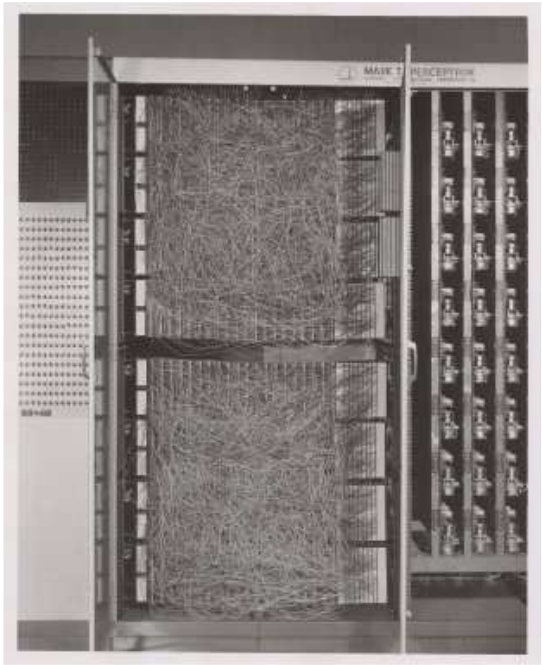
Schematic of Rosenblatt's perceptron.

1958년 Perceptron

현재 사용하는 딥러닝도 이 구조를 여러개의 노드와 여러개의 레이어로 확장했다는 것이 다를 뿐 퍼셉트론과 근본적인 구조는 같다.

퍼셉트론 학계와 매스컴으로부터 엄청난 기대와 주목을 받음

많은 사람들이 퍼셉트론을 통해 진짜 인간과 같은 인공지능을 만들 수 있을 것이라고 기대하였음

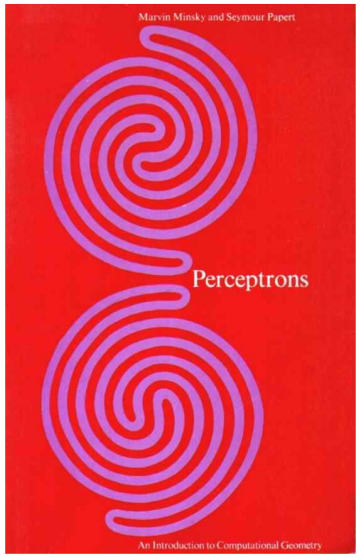


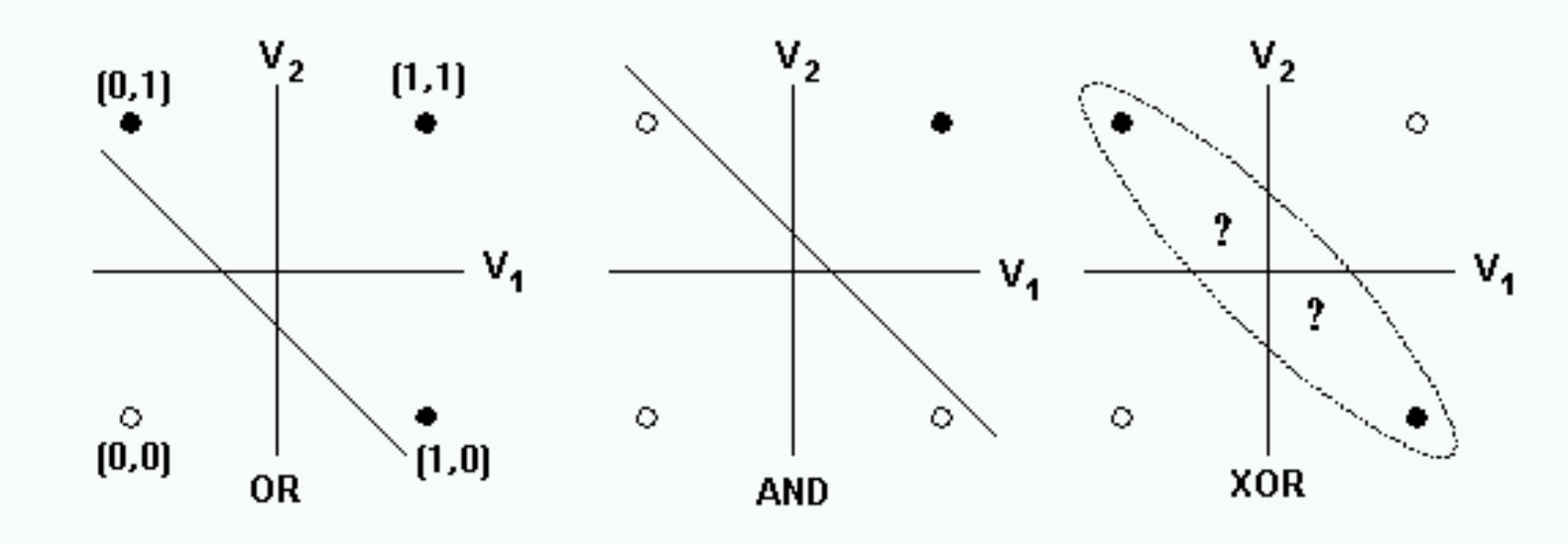
1969년 XOR

1969년 Marvin Minsky, Seymour Papert
"Perceptrons: an introduction to computational geometry"**서적 발간**
- **퍼셉트론의 한계를 수학적으로 증명**
기대와 열기는 급속히 사그라 들었음

퍼셉트론은 단순한 선형 분류기에 불과
- **간단한 XOR 분류조차 수행할 수 없다고 지적**

이 책으로 인해 인공신경망 연구는 인공지능 연구 그룹의 관심에서 멀어지게 되고, 연구자들은 다른 방법들을 탐구하기 시작함



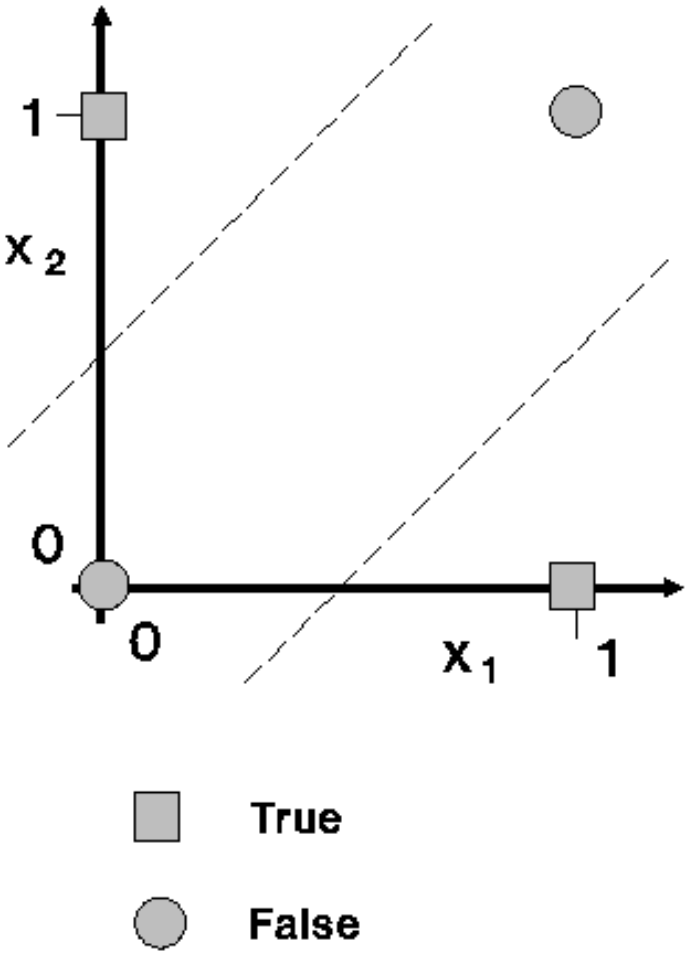


1986년 Back Propagation

인공신경망의 꾸준히 연구

1986**년**
McClelland, James L., David E. Rumelhart, and Geoffrey E. Hinton
"Parallel Distributed Processing" **서적**
히든 레이어를 가진 Multi-Layer Perceptrons(MLP)**과**
Backpropagation Algorithm **제시**

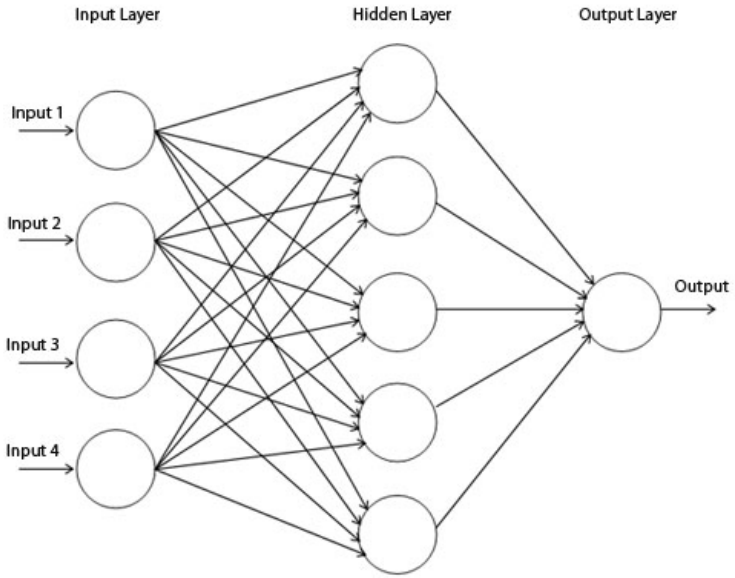
기존의 퍼셉트론 :
선형 분류기라는 한계에 의해 XOR **문제를 해결할 수 없음**
Multi-Layer Perceptrons(MLP)
히든 레이어(Hidden Layer) - **중간 레이어 추가**
선형 분류 판별선 여러개 그리는 효과를 얻음
-> XOR **문제를 해결**

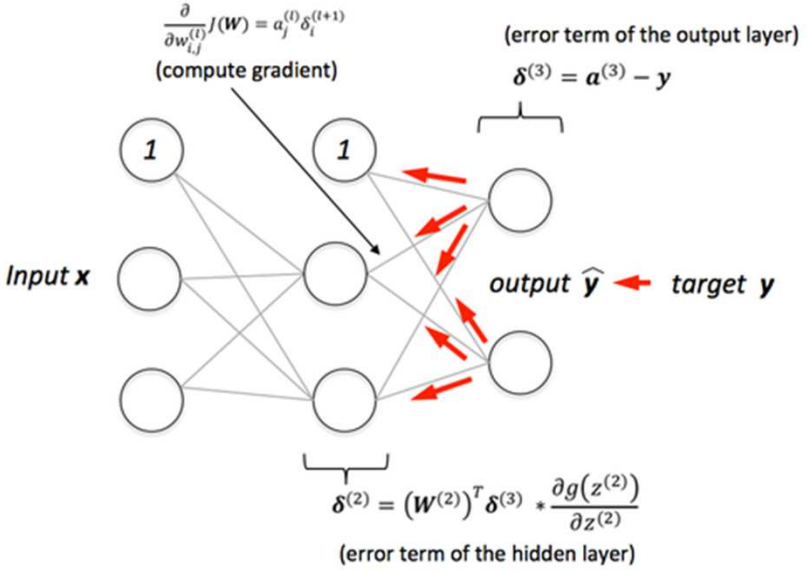


1986년 Back propagation

MLP : **파라미터가 개수가 많아지면서 적절한** Weight**와** Bias **학습 어려워짐**
Backpropagtion Algorithm **제안 문제 해결**
McClelland, James L., David E. Rumelhart, and Geoffrey E. Hinton

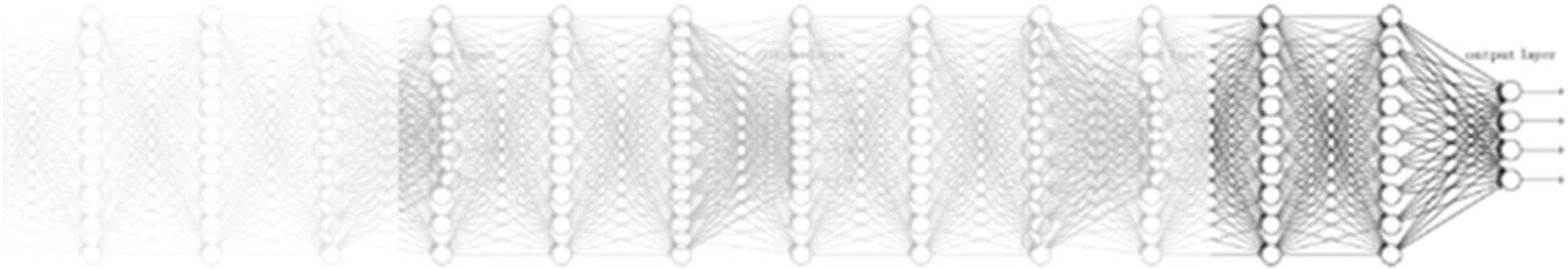
Backpropagtion Algorithm - **오류 역전파 알고리즘** -
정방(Feedforward) **연산 이후, 에러**(Error)-**예측 값과 트루 값과의 오차**- **후방**(Backward)**으로 다시 보내 줌**
많은 노드를 가진 MLP**라도 최적의** Weight**와** Bias**를 학습할 수 있도록 함**





1986년 Vanishing Gradient Problem

하지만, MNIST 문자 분류와 같은 응용 사례에도 불구하고, MLP는 다른 현실세계의 문제에 잘 작동하지 않고,
레이어를 깊게 쌓을수록 Backpropagation 과정에서 Gradient가 사라지는 Vanishing Gradient Problem 때문에 한계에 직면하게 되고,
MLP보다 강력한 성능을 보여주는 SVM과 같은 Kernel Method가 등장함으로써, 인공신경망 연구는 다시 인공지능 학계의 주류에서 밀려나나게 된다.

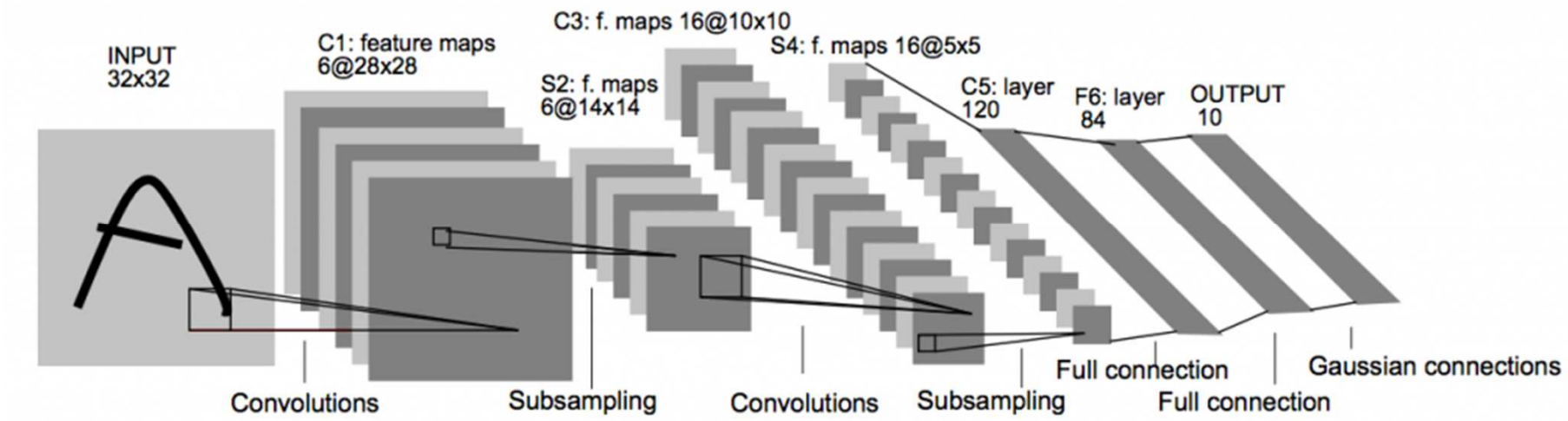


1989년 CNN paper / 1998년 LeNet-5

Yann Lecun 1989**년 논문**
"Handwritten digit recognition with a back-propagation network"
MLP**와** Backpropagation Algorithm**을** Convolution**과 결합**
필기숫자 인식(MNIST) **문제에 적용**

https://www.youtube.com/watch?v=FwFduRA_L6Q

1998**년 논문**
"Gradient-Based Learning Applied to Document Recognition"
LeNet-5 : Convolutional Neural Networks(CNNs)**의 시초가 되는** Neural Networks **구조 제안**
우체국의 수표 인식 시스템에 실용적으로 적용



2006년 초기화 - RBM

Not all 0's

• Challenging issue

Hinton et al. (2006) "A Fast Learning Algorithm for Deep Belief Nets"

- Restricted Boltzmann Machine (RBM)

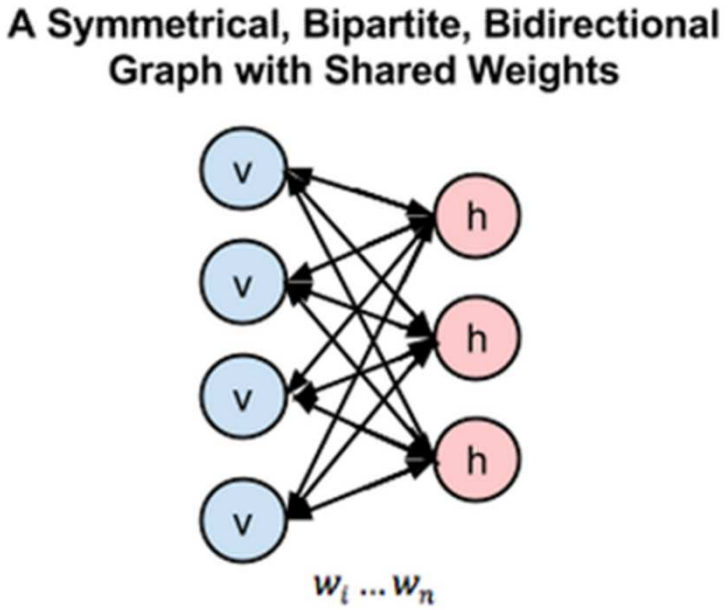
Apply the RBM idea on adjacent two layers as a pre-training step

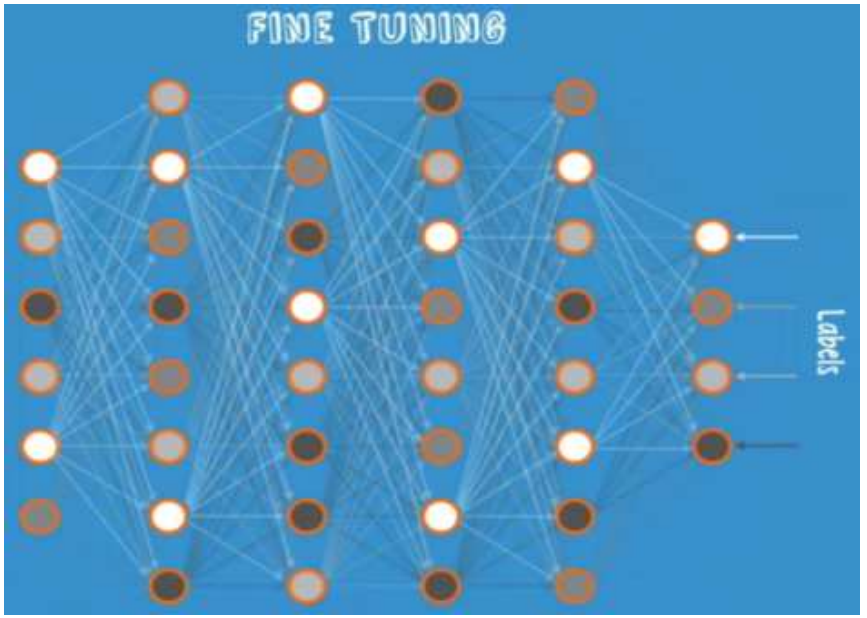
Continue the first process to all layers

This will set weights

Example: Deep Belief Network

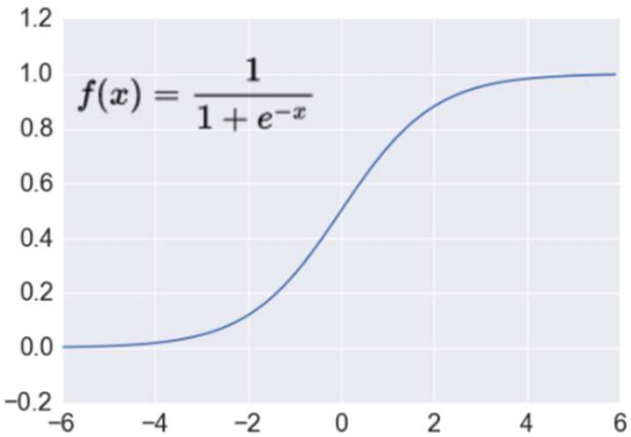
- Weight initialized by RBM

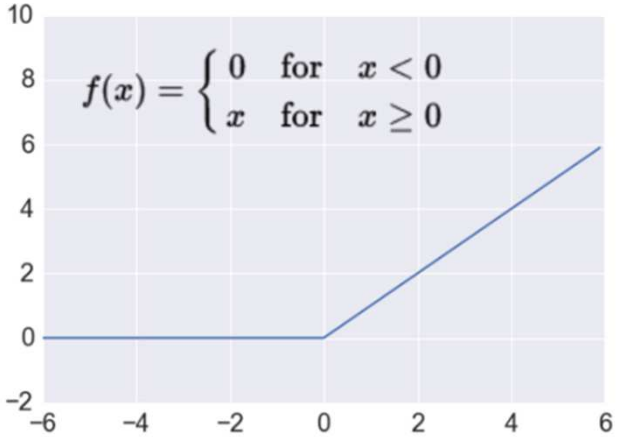


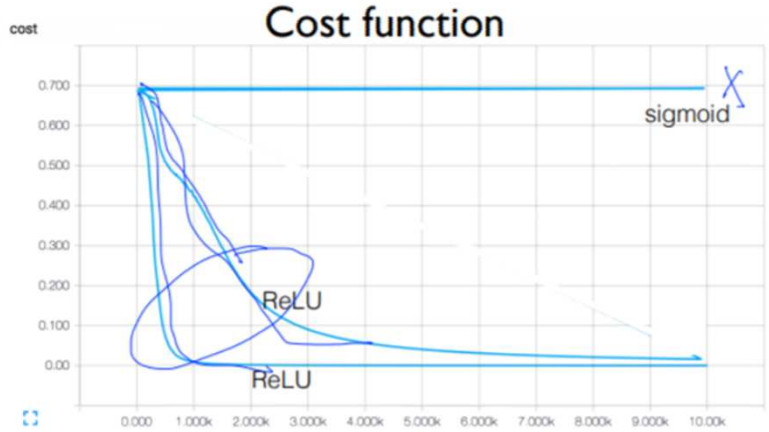


2007년 Relu

이전 Sigmoid Function **활성화 함수로 많이 사용**,
층(layer)수가 깊어질수록 Gradient **값이 감소하는** Gradient Vanishing **문제를 발생**
Gradient Vanishing**를 해결하기 위한 방법으로**
Rectified Linear Function **사용**

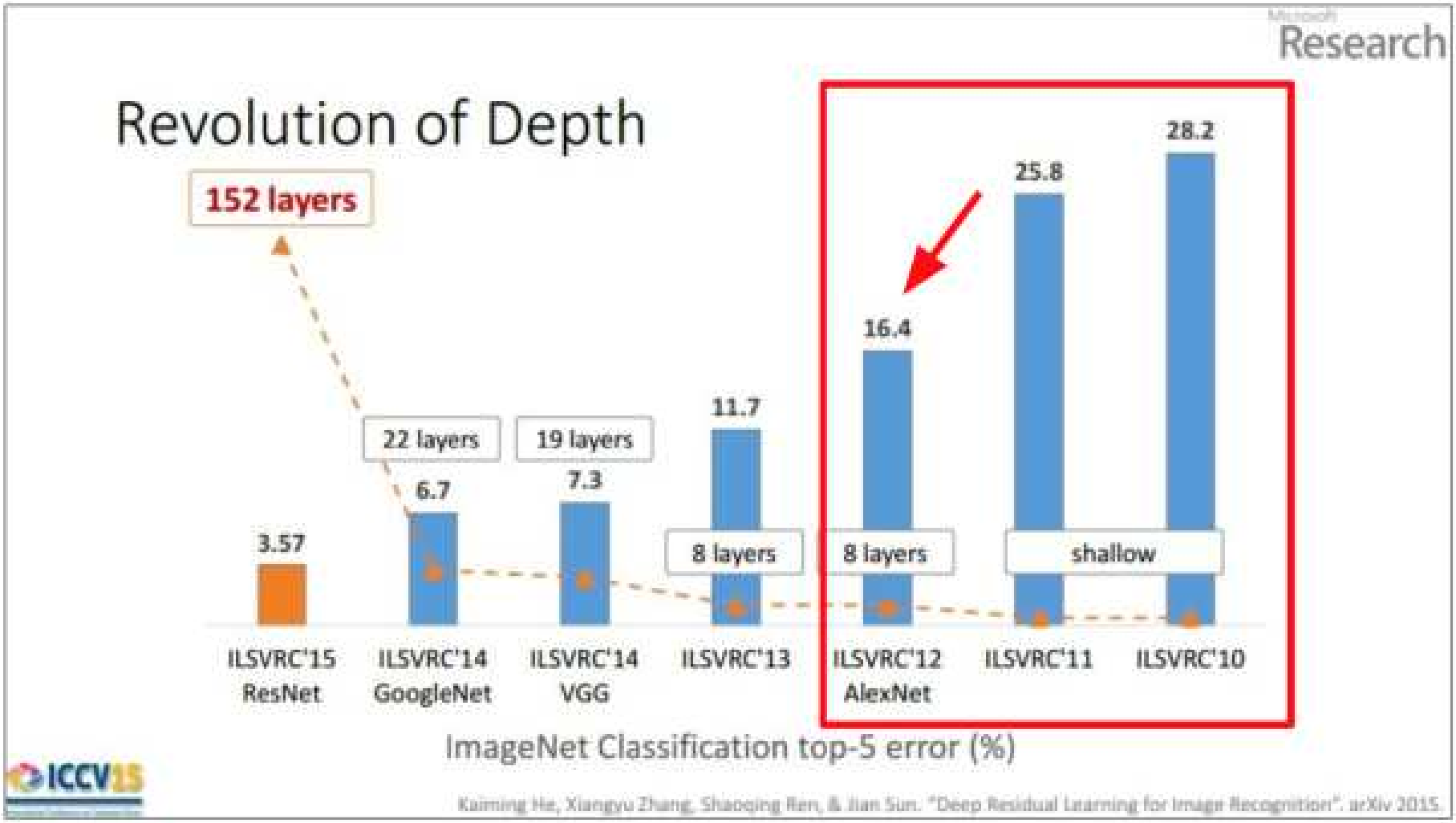






2012년 – 2015년 이미지넷

✓ 2016년 2.99%



(slide from Kaiming He's recent presentation)

DEEP Reinforcemation Learning

DQN

- DeepMind Technologies 2013**년** 12**월 논문**

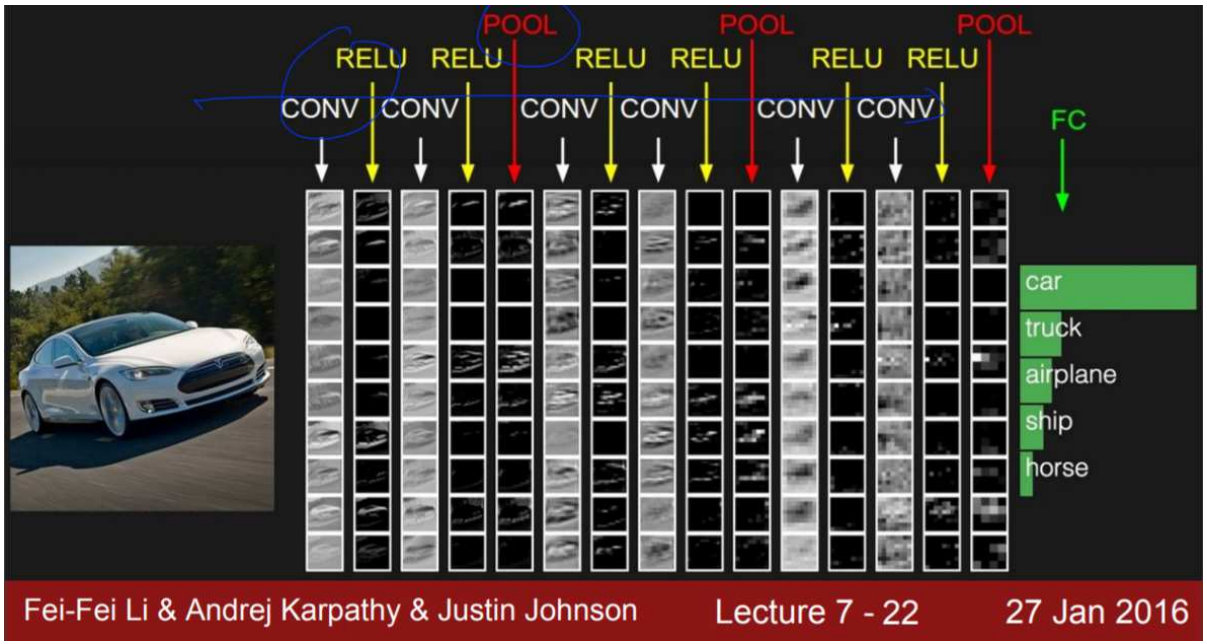
“Playing Atari with Deep Reinforcement Learning”

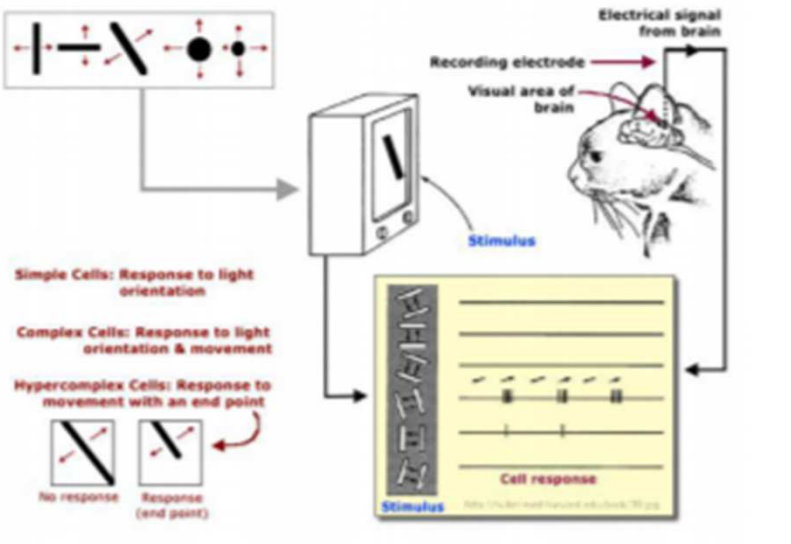
강화 학습 문제에 deep learning**을 적용한 첫 번째 사례**

- 2015**년** Nature **논문**

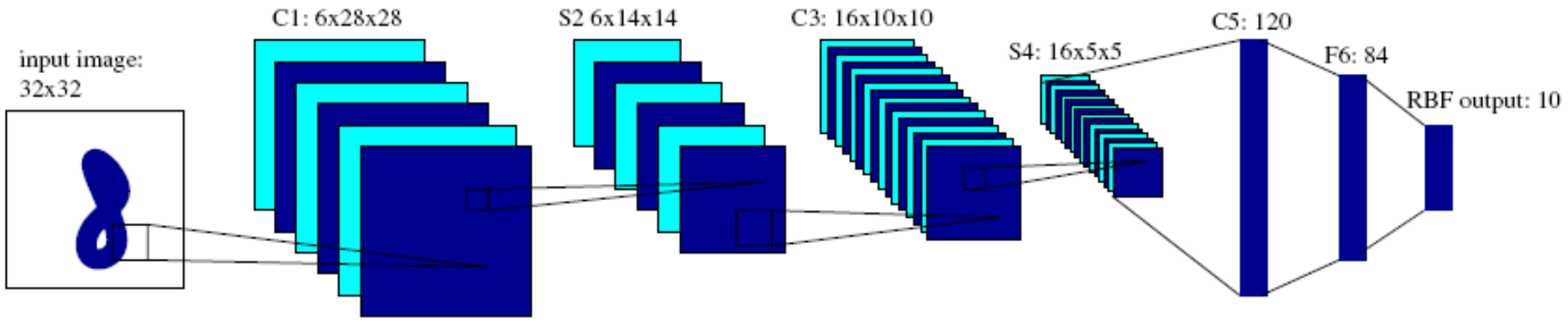
- “Human-Level Control Through Deep Reinforcement Learning”

CNN
Convolutional Neural Network





Hubel & Wiesel, 1959



Convolution layer

			0	0	0	0	0	0
			3	4	5	6	7	0
			12	11	10	9	8	0
0	15	16	17	18	19	20	21	0
0	28	27	26	25	24	23	22	0
0	29	30	31	32	33	34	35	0
0	42	41	40	39	38	37	36	0
0	43	44	45	46	47	48	49	0
0	0	0	0	0	0	0	0	0

1	2	3	4	5	6	7
14	13	12	11	10	9	8
15	16	17	18	19	20	21
28	27	26	25	24	23	22
29	30	31	32	33	34	35
42	41	40	39	38	37	36
43	44	45	46	47	48	49

channel

Input(N) : 7x7
Pad : 1
Filter(F) : 3 x 3
Stride :1

Filter와 input image가 겹치는 부분을 합성곱을 통하여 하나의 값이 생성됨

Scan : Filter가 Stride 크기 만큼 이동함

http://setosa.io/ev/image-kernels/

I_{11}	I_{12}	I_{13}
I_{21}	I_{22}	I_{23}
I_{31}	I_{32}	I_{33}

F_{11}	F_{12}	F_{13}
F_{21}	F_{22}	F_{23}
F_{31}	F_{32}	F_{33}

O_{22}

$$I_{11} \times F_{11} + I_{12} \times F_{12} + I_{13} \times F_{13} +$$
$$I_{21} \times F_{21} + I_{22} \times F_{22} + I_{23} \times F_{23} +$$
$$I_{31} \times F_{31} + I_{32} \times F_{32} + I_{33} \times F_{33} = O_{22}$$

Output size = (N-F+2PAD)/stride + 1
= (7 - 3 + 2)/1 + 1 = 7

Pooling layer

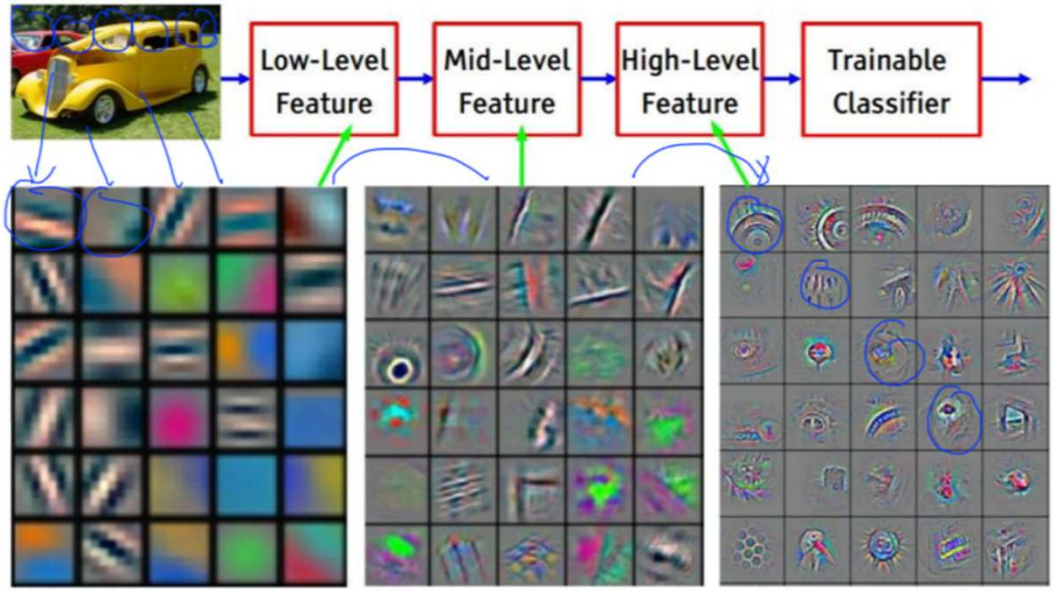
원본 이미지를 축소하여 필터를 키우는 효과로 사용.

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Max pool with 2x2 filters and stride 2

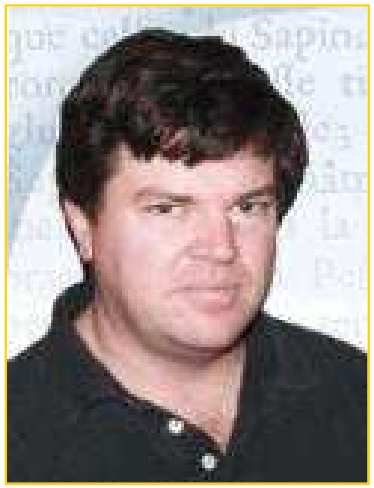
6	8
3	4

[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

History



Yann LeCun, Professor of Computer Science
The Courant Institute of Mathematical Sciences
New York University
Room 1220, 715 Broadway, New York, NY 10003, USA.
(212)998-3283 yann@cs.nyu.edu

- In 1995, Yann LeCun and Yoshua Bengio introduced the concept of convolutional neural networks.

LeNet 5

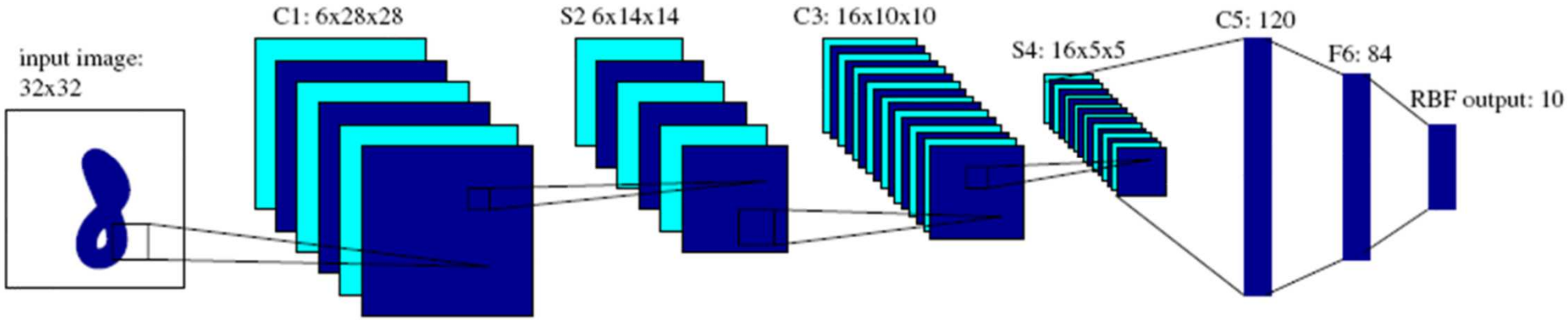
Conv filters were 5x5, applied at stride 1 Subsampling (Pooling) layers were 2x2 applied at stride 2 i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

LeNet5

- ✓ Introduced by LeCun.
- ✓ raw image of 32 × 32 pixels as input.

C1,C3,C5 : Convolutional layer.
5 × 5 Convolution matrix.
S2 , S4 : Subsampling layer.
Subsampling by factor 2.
F6 : Fully connected layer.

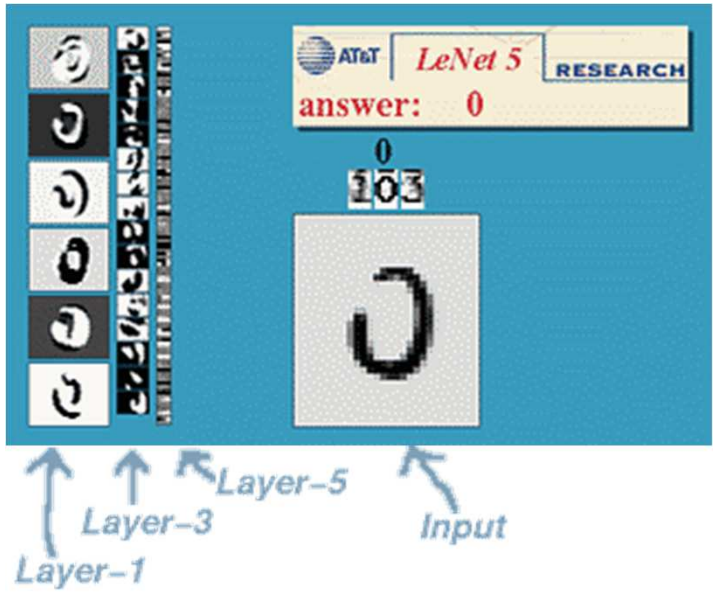
About 187,000 connection.
About 14,000 trainable weight.

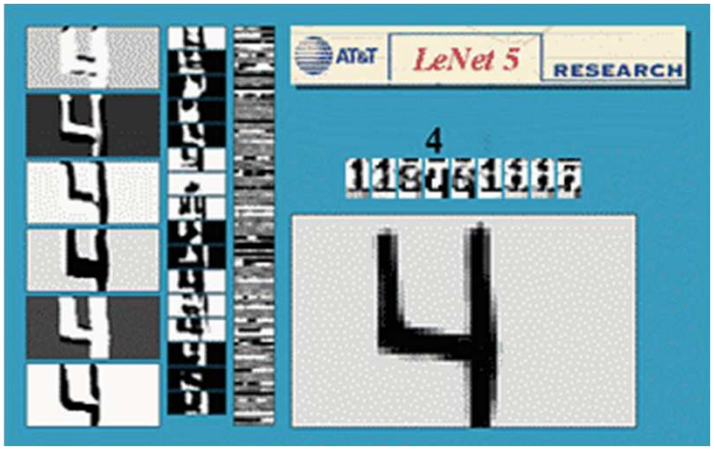


About LeNet 5

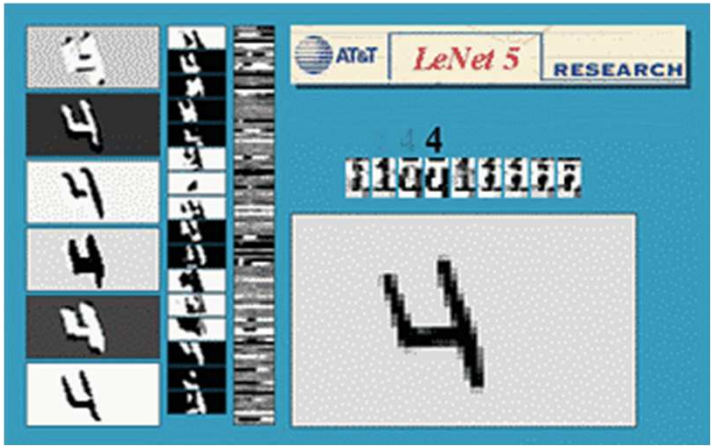
- ✓ Convolution과 Subsampling을 반복적으로 거치면서, 마지막에 Fully-connected Muti-layered Neural Network로 Classification을 수행하고 있다.
- ✓ C1에서는 5x5 Convolution 연산하여 28 x 28 사이즈의 6개의 feature map을 생성한다.
- ✓ 5 x 5 Convolution kernel에 bias가 더해져, 하나의 feature map을 만드느데 (5 x 5 + 1 = 26)개의 자유 파라미터가 있고, 6개의 feature map을 생성하므로 156개의 자유 파라미터가 있습니다.
- ✓ S2에서는 Subsampling 하여 feature map의 크기를 14 x 14로 줄입니다.
- ✓ Average pooling을 수행하므로 각각의 feature map 마다 weight 1 + bias 1의 자유 파라미터가 있어 총 12개의 자유 파라미터가 있습니다.
- ✓ C3에서는 5 x 5 Convolution 연산하여 10 x 10 사이즈의 16개의 feature map을 생성합니다.

LeNet5

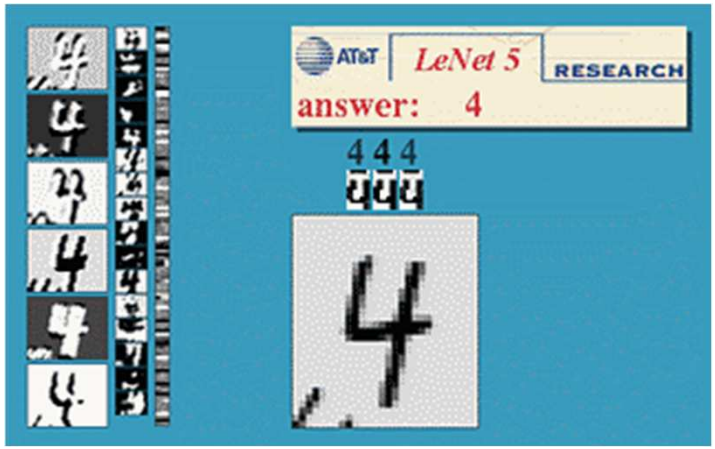

AT&T
LeNet 5
RESEARCH
answer: 0
0
Layer–1
Layer–3
Layer–5
Input

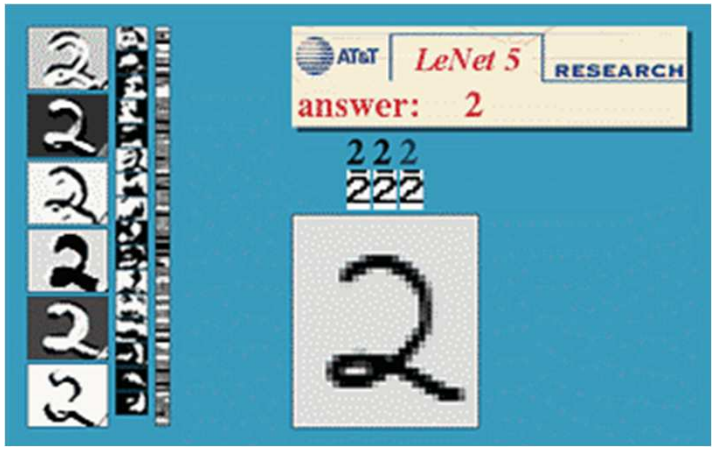

AT&T
LeNet 5
RESEARCH
4

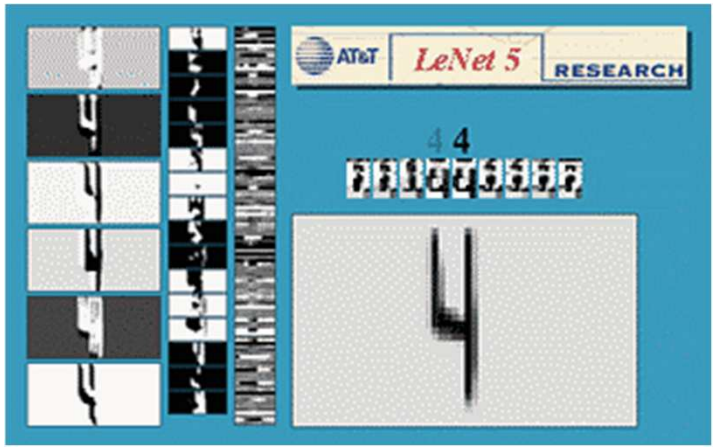

AT&T
LeNet 5
RESEARCH


AT&T
LeNet 5
RESEARCH
4

LeNet5


AT&T
LeNet 5
RESEARCH
answer: 4


AT&T
LeNet 5
RESEARCH
answer: 2


AT&T
LeNet 5
RESEARCH


AT&T
LeNet 5
RESEARCH
answer: 6

ALEXNET

[Krizhevsky et al. 2012]

Alexnet

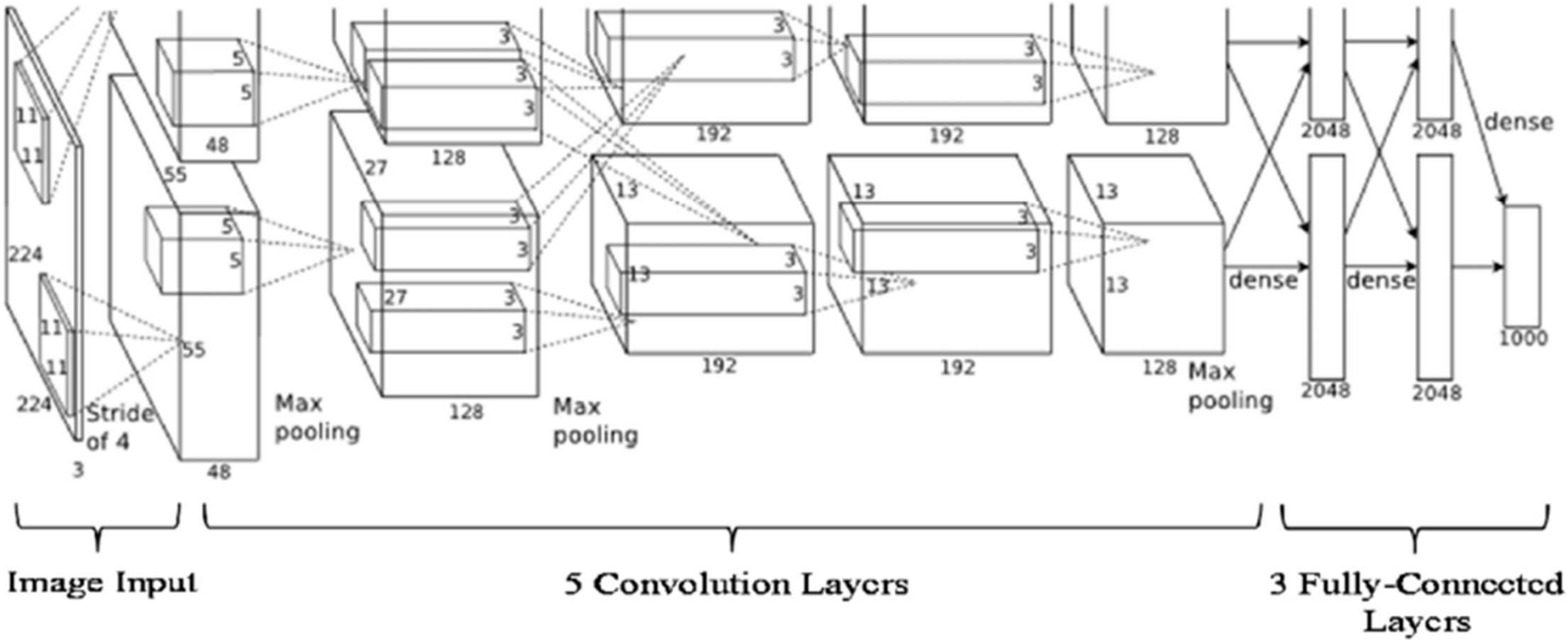
- ✓ Alex Krizhevsky의 이름을 따서 작명
- ✓ 2012년 ILSVRC에서 압도적인 winner가 된 네트워크
- ✓ 네트워크 이후 ZFNet, NIN, VGGNet, GoogLeNet, ResNet등 다양한 뉴럴넷 기반의 모델들이 ILSVRC 혹은 다른 데이터셋에서 outperform한 결과를 보임
- ✓ AlexNet은 이 돌풍을 열게 한 선두주자

- ✓ Datasets
  - 평균 이미지를 입력 이미지에 빼어 zero-centered된 이미지를 입력 이미지로 사용

- ✓ ReLU Nonlinearlity
  - Vanishing gradient 문제를 해결하기 위해 sigmoid 혹은 tanh 대신 ReLU activation 함수 사용

- ✓ Multiple GPU 사용
  - GPU(GTX 580, 3GB VRAM)의 메모리 부족 문제로 두개의 GPU를 병렬로 사용

Alexnet



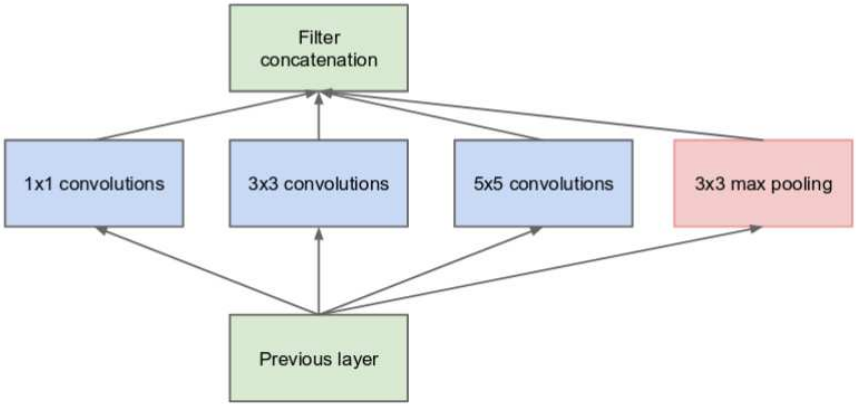
입력 이미지가 [227x227x3]**이라 가정하고** conv1**의 필터는** [11x11]**이** 96**개이고** stride**는** 4
conv1**을 거치고 나온 출력** volume**은** [55x55x96] ➔ (227-11)/4/1 + 1**에 의해** 55**가 나옴** 96**은 필터의 수와 동일**
parameter**의 수는** 11*13*96**으로** 35K **정도**
conv1**을 나온 출력값을** [3x3], stride**가** 2**인** pooling **레이어에 적용하면 출력값은** [27x27x96]**으로 나옴**
(55-3)/2 + 1**에 의해** 27**이 나옴** parameter **수** 0

GoogLeNet

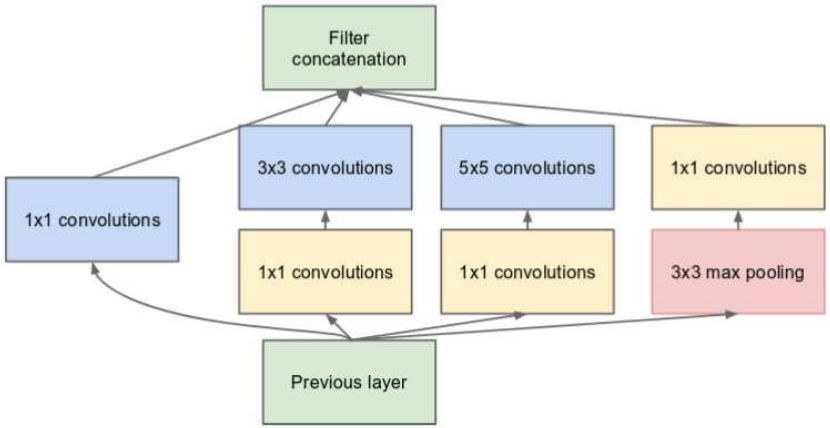
[Szegedy et al., 2014]

Inception

✓

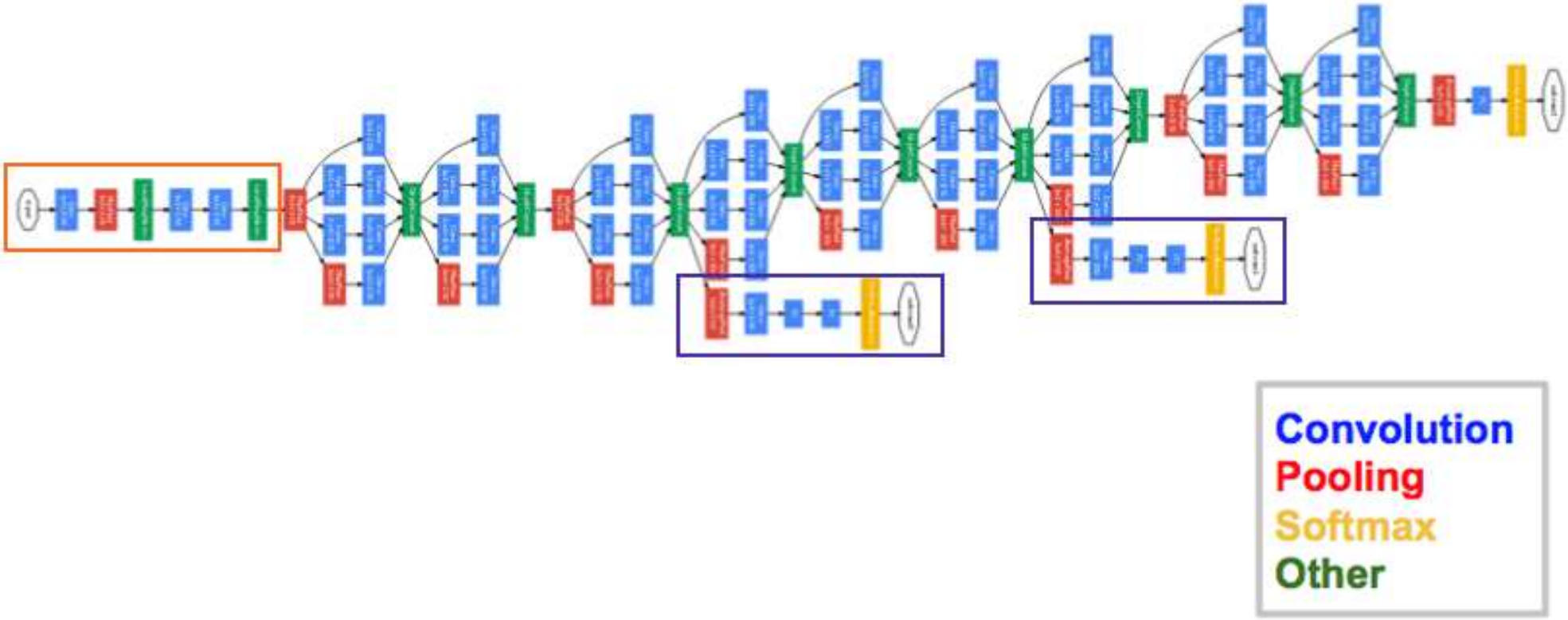


(a) Inception module, naïve version

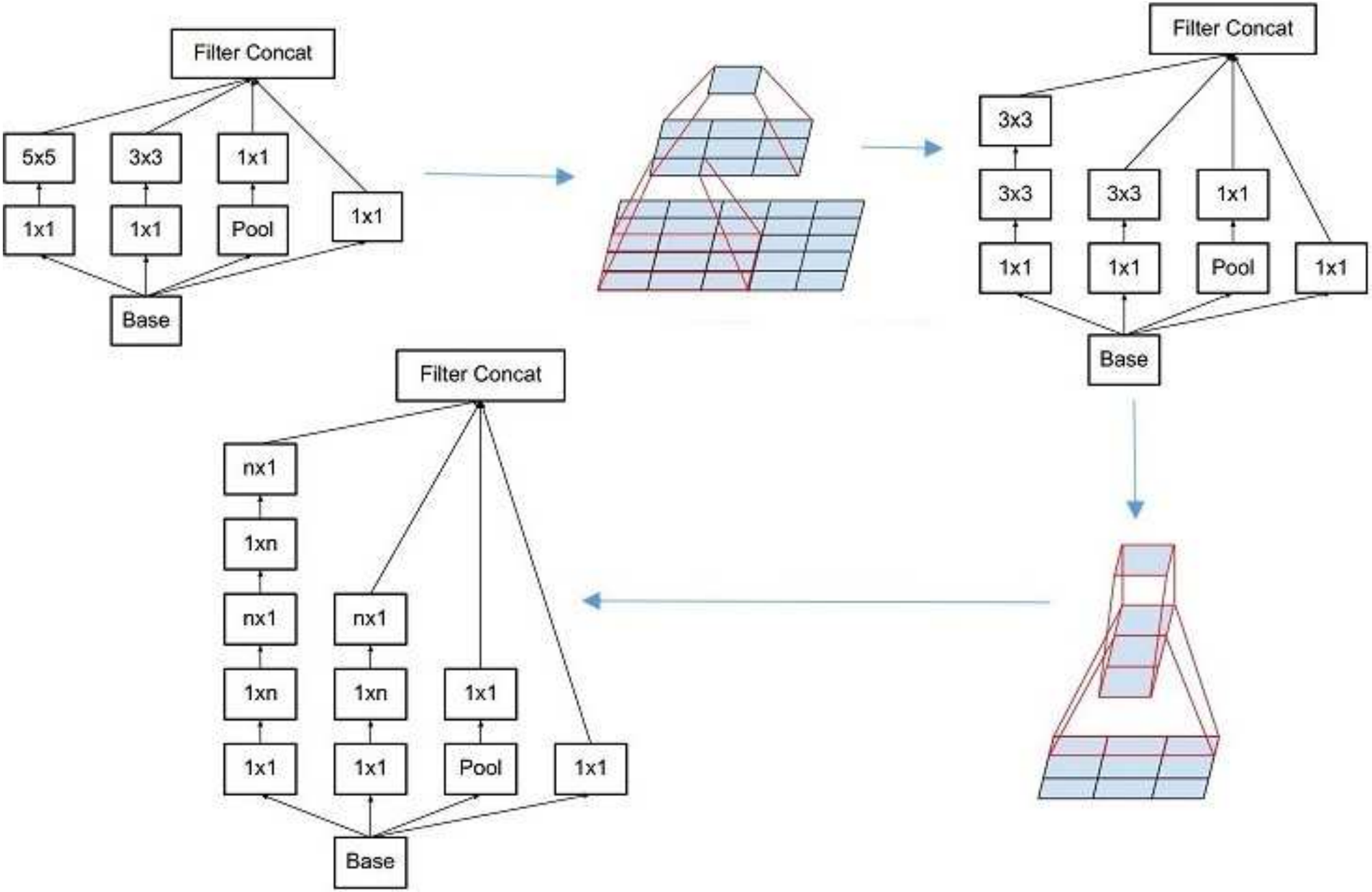


(b) Inception module with dimension reductions

GoogLeNet의 구조

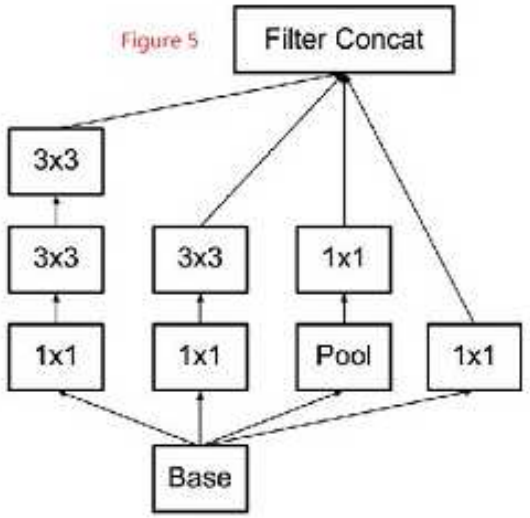


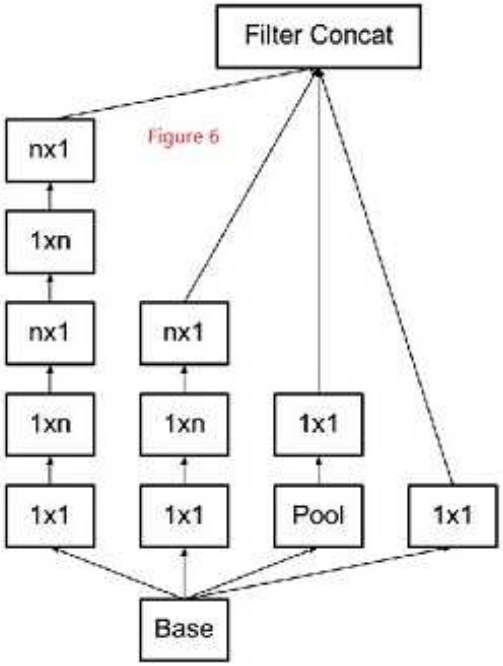
Factorizing Convolution

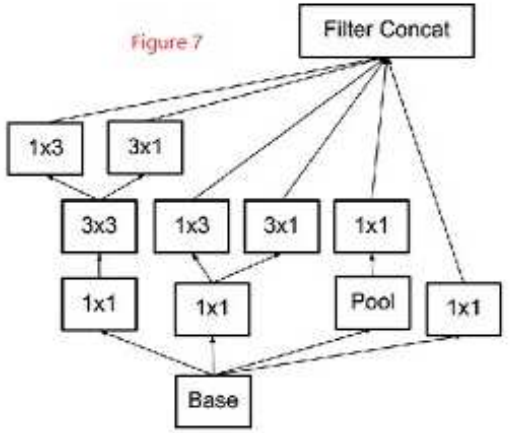


Factorizing Convolution

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000







VGGNet

[et al., 2014]

VGG 개요

- ✓ VGGNet is invented by VGG (Visual Geometry Group) from University of Oxford
- ✓ VGGNet은 2014년 ILSVRC에서 GoogLeNet과 함께 높은 성능을 보인 네트워크
- ✓ 간단한 구조, 단일 네트워크에서 좋은 성능 등을 이유로 여러 응용 분야에서 기본 네트워크로 많이 사용

Architecture

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					

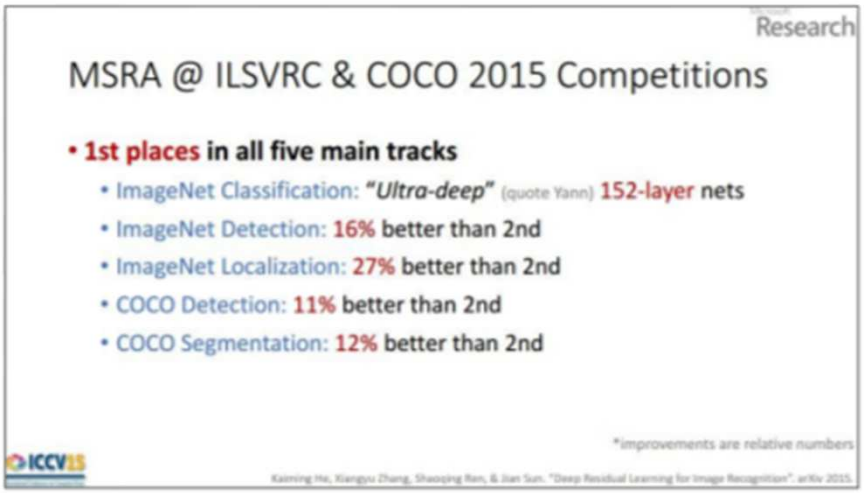
VGGNet 특징

- ✓ 3x3 필터 여러 개의 장점 두가지
- ✓ 첫째
  - 여러 개의 ReLU non-linear를 사용
  - 큰 필터로 구성된 하나의 레이어를 작은 필터의 여러 레이어로 나누었기 때문에 ReLU non-linearlity가 들어갈 곳이 더 많아짐
  - 이는 decision function이 더 discriminative하다는 의미
- ✓ 둘째
  - 학습해야할 weight의 수가 많이 줄어듬
  - 3x3 conv 레이어 3 개와 7x7 conv 레이어 1 개 비교
  - 두 레이어 모두 C개의 채널을 가진다.
  - 7x7 conv 레이어의 parameter의 개수 7^2 x C^2=49 x C^2
  - 3개의 3x3 conv 레이어 3*(3^2 x C^2) = 27 x C^2
  - 학습해야할 parameter의 개수가 적다면 regularization 측면에서 큰 이점을 보임

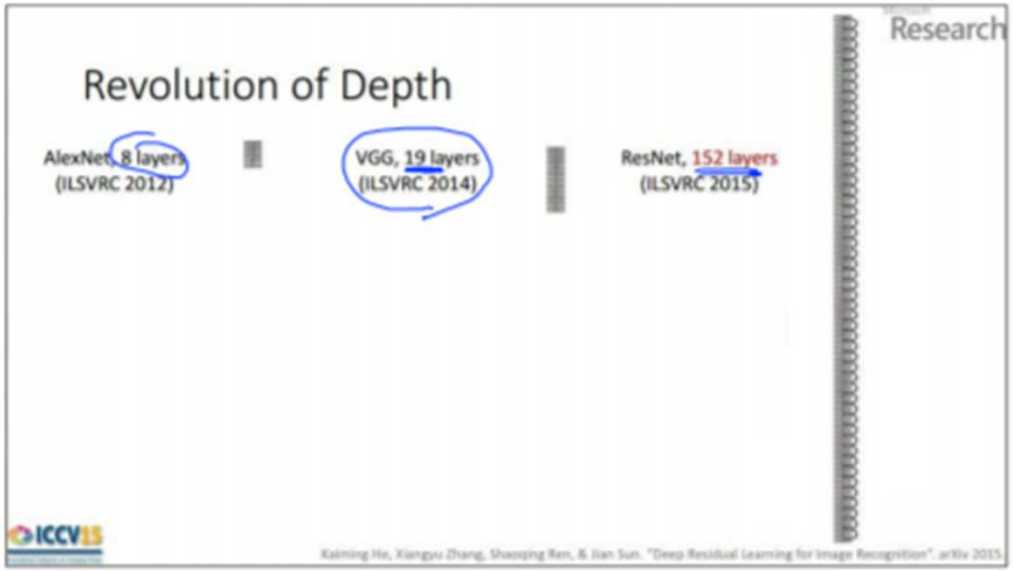
ResNet

[He et al., 2015]

ResNet



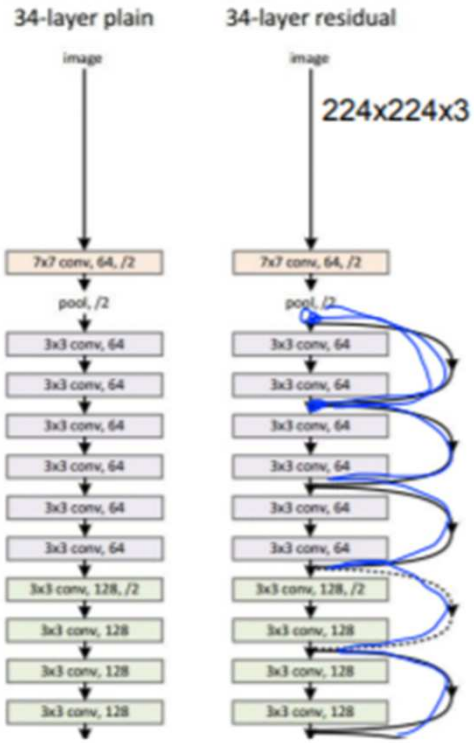
Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w



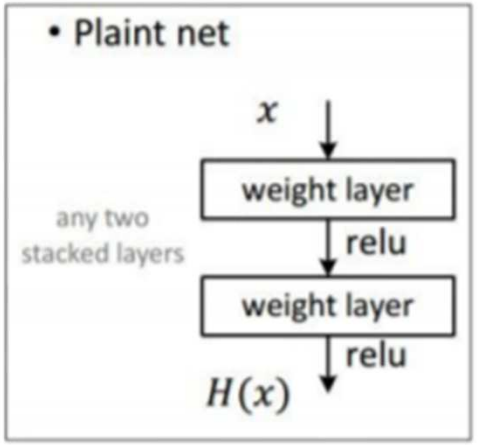
(slide from Kaiming He's recent presentation)

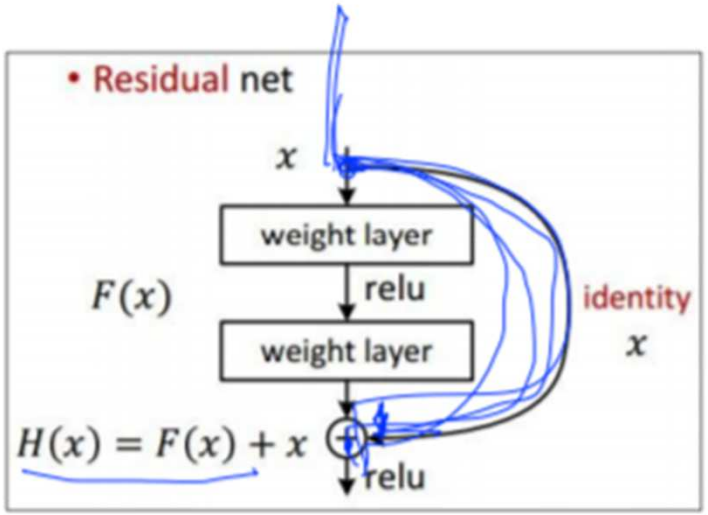
2-3 weeks of training on 8 GPU machine

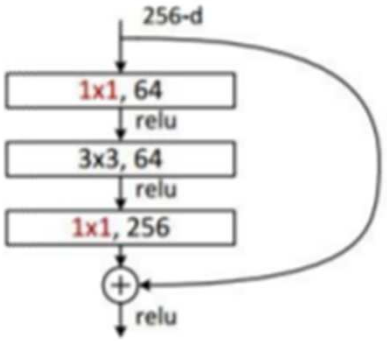
at runtime: faster than a VGGNet! (even though it has 8x more layers)

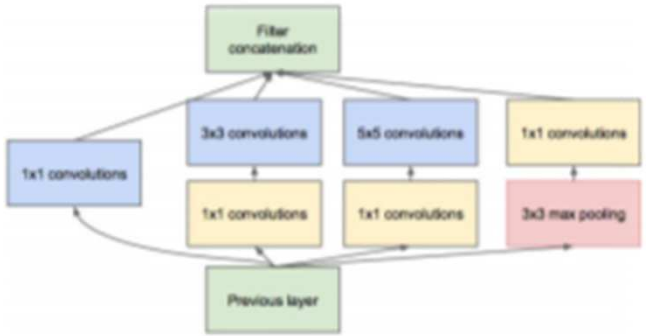


ResNet


• Plaint net
x
weight layer
relu
weight layer
relu
$H(x)$
any two
stacked layers


• Residual net
x
weight layer
relu
weight layer
$F(x)$
identity
x
$H(x) = F(x) + x$
relu


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu


Filter concatenation
1x1 convolutions
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

감사합니다.

e2g1234@naver.com