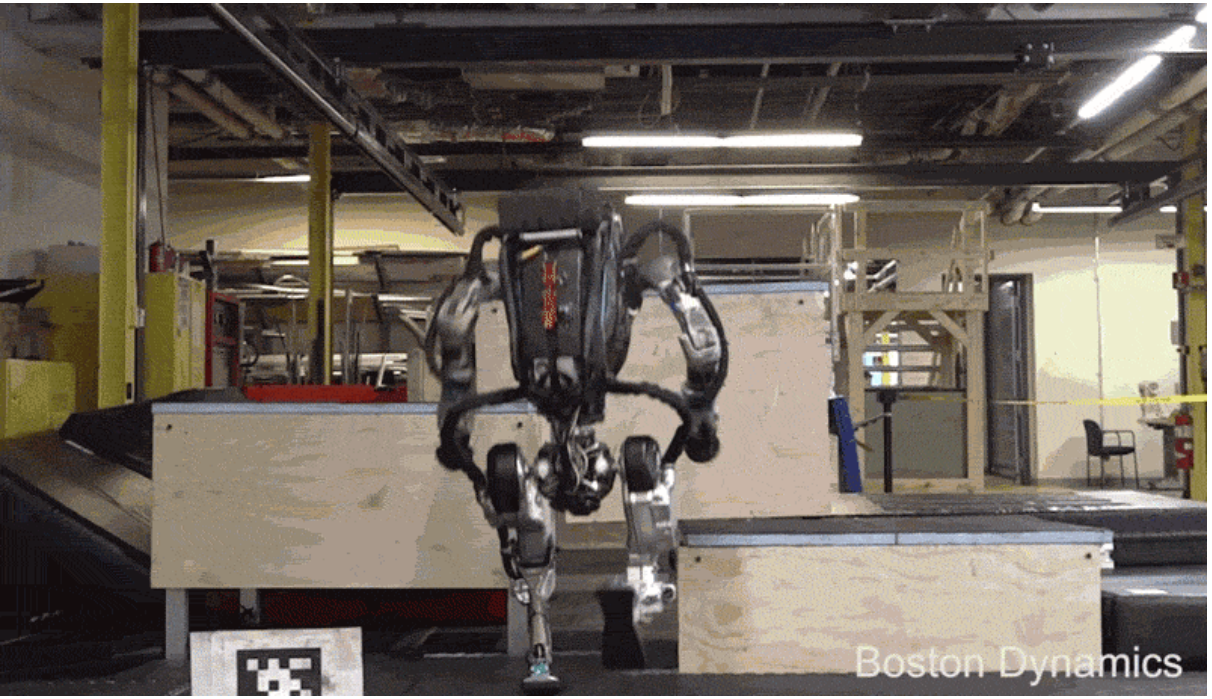
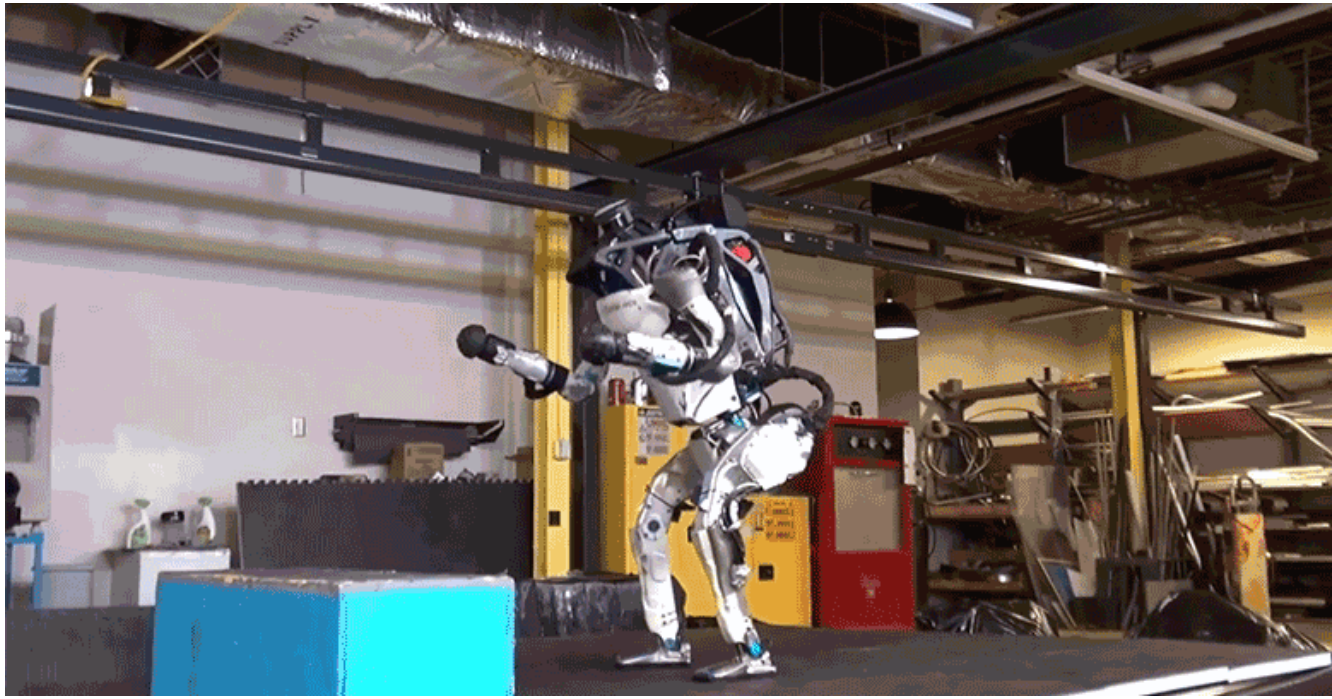
Introduction to Deep Learning & Neural Networks

2001: A Space Odyssey

Boston Dynamics' Robots





2018 Google Assistant

AI Speaker

AI vs Human

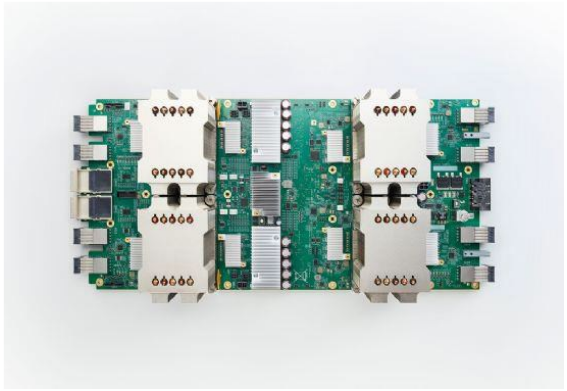
Deep Blue – 2승 3무 1패

Watson

AlphaGo

TPU Server used against Lee Sedol

AlphaGo Lee – 4승 1패



TPU Board used against Ke Jie

AlphaGo Master – 3승 0패

Dota 2 & StarCraft 2

Elon Musk's AI beats the world's best Dota 2 players

Staff Writer 14 August 2017 15 Comments

DeepMind's AlphaStar AI wins 10-1 against professional StarCraft II players

Abner Li - Jan. 24th 2019 12:50 pm PT @technacity

The Singularity is Near

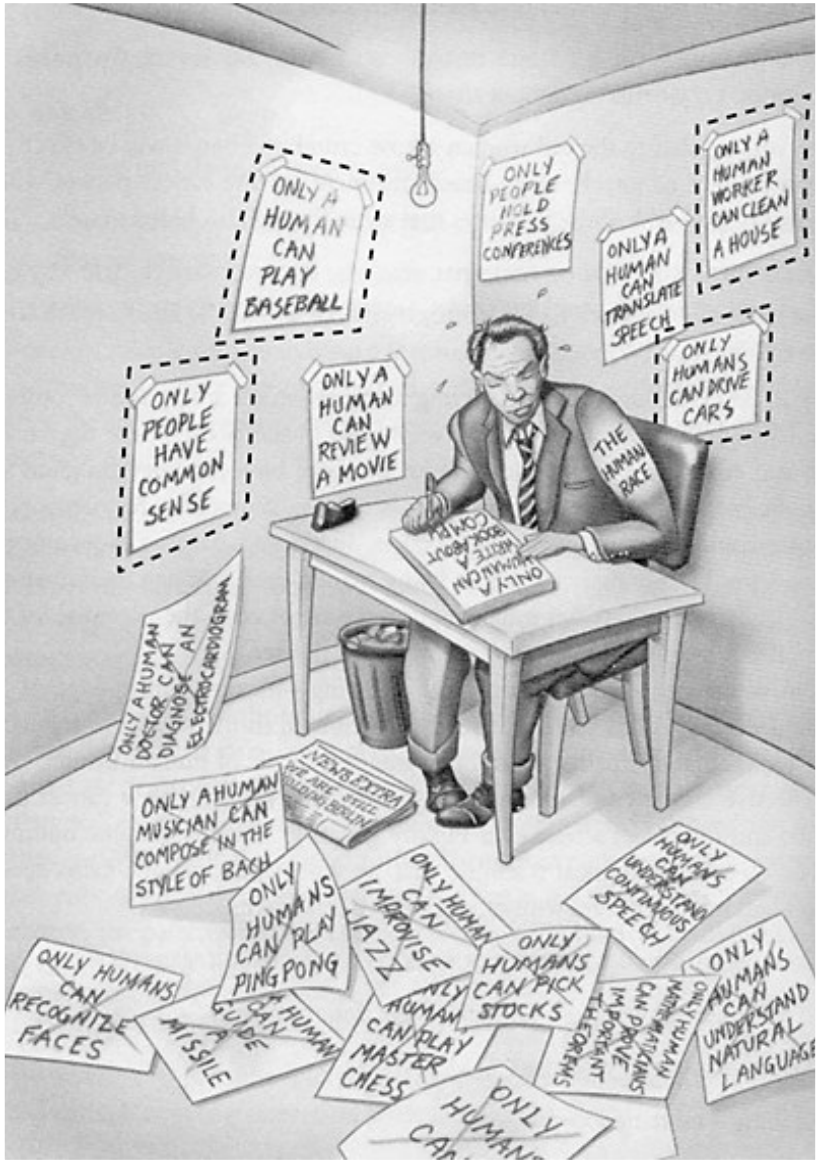
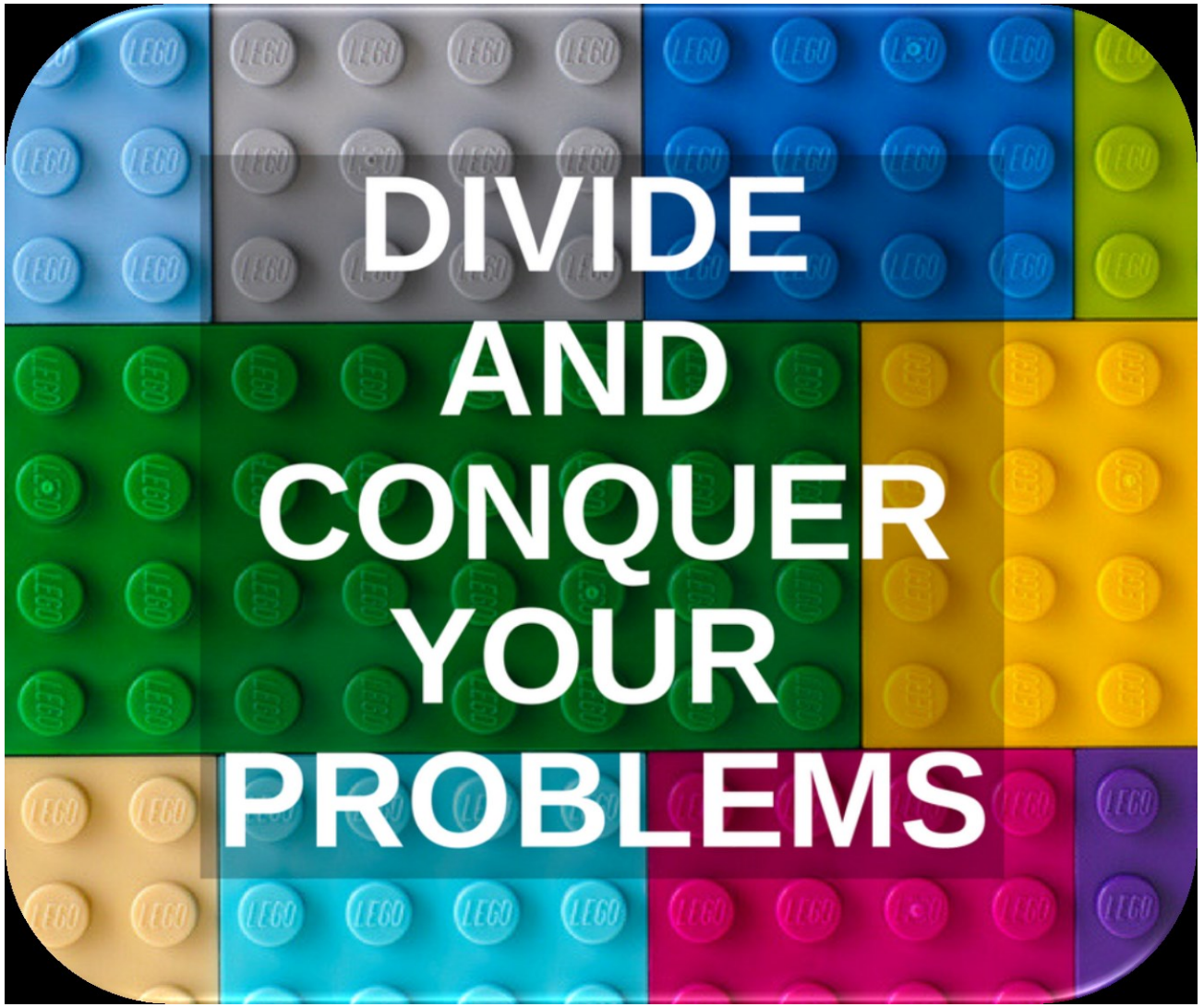


Image from Kurzweil's The Singularity Is Near, published in 2005.

Divide and Conquer



AI, ML, DL

Artificial Intelligence

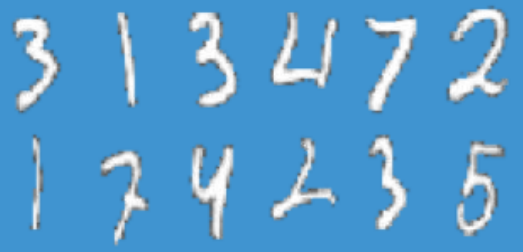
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

Extract patterns from data using neural networks



Why Learning?

What do you see?

How do we do that?!

https://github.com/glouppe/info8010-deep-learning

Why Learning?

Sheepdog or mop?

Chihuahua or muffin?

https://github.com/glouppe/info8010-deep-learning

Why Learning?

- The automatic extraction of semantic information from raw signal is at the core many applications, such as
  - Image recognition
  - Speech processing
  - Natural language processing
  - Robotic control
  - … and many others

- How can we write a computer program that implements that?

What is This?

- The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:

This is a mushroom.

https://github.com/glouppe/info8010-deep-learning

What is This?

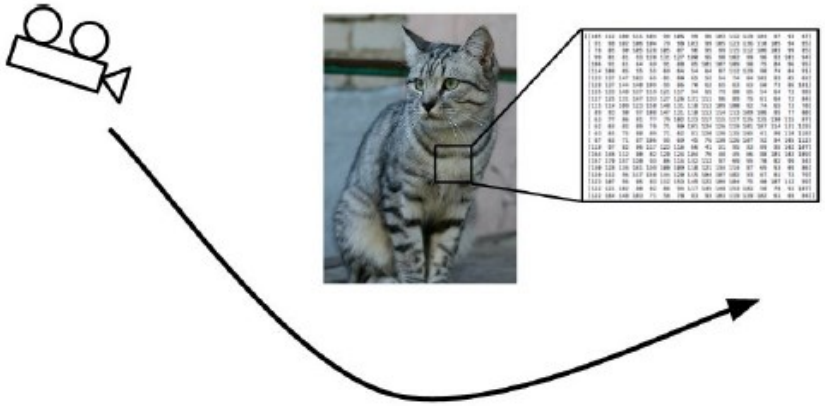
```
array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
        [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
        [0.4117647 , 0.34117648, 0.37254903, 1.        ],
        ...,
        [0.20392157, 0.23529412, 0.17254902, 1.        ],
        [0.16470589, 0.18039216, 0.12156863, 1.        ],
        [0.18039216, 0.18039216, 0.14117648, 1.        ]],

       [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
        [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
        [0.21176471, 0.2       , 0.20392157, 1.        ],
        ...,
        [0.1764706 , 0.24705882, 0.12156863, 1.        ],
        [0.10980392, 0.15686275, 0.07843138, 1.        ],
        [0.16470589, 0.20784314, 0.11764706, 1.        ]],

       [[0.14117648, 0.12941177, 0.10980392, 1.        ],
        [0.21176471, 0.1882353 , 0.16862746, 1.        ],
        [0.14117648, 0.13725491, 0.12941177, 1.        ],
        ...,
        [0.10980392, 0.15686275, 0.08627451, 1.        ],
        [0.0627451 , 0.08235294, 0.05098039, 1.        ],
        [0.14117648, 0.2       , 0.09803922, 1.        ]],

       ...,
```

This is a mushroom.

https://github.com/glouppe/info8010-deep-learning

What is a CAT?

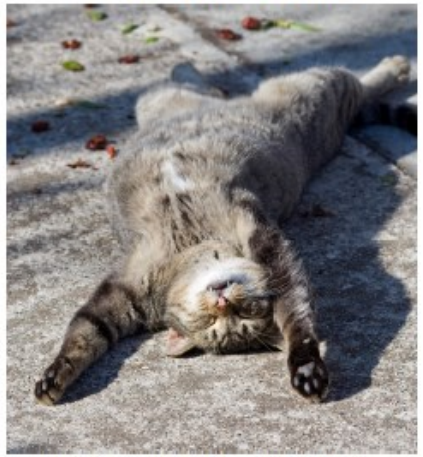
Viewpoint



Illumination

This image is CC0 1.0 public domain

Deformation



This image by Umberto Salvagnin is licensed under CC-BY 2.0

Occlusion

This image by jonsson is licensed under CC-BY 2.0

Clutter

This image is CC0 1.0 public domain

Intraclass Variation

This image is CC0 1.0 public domain

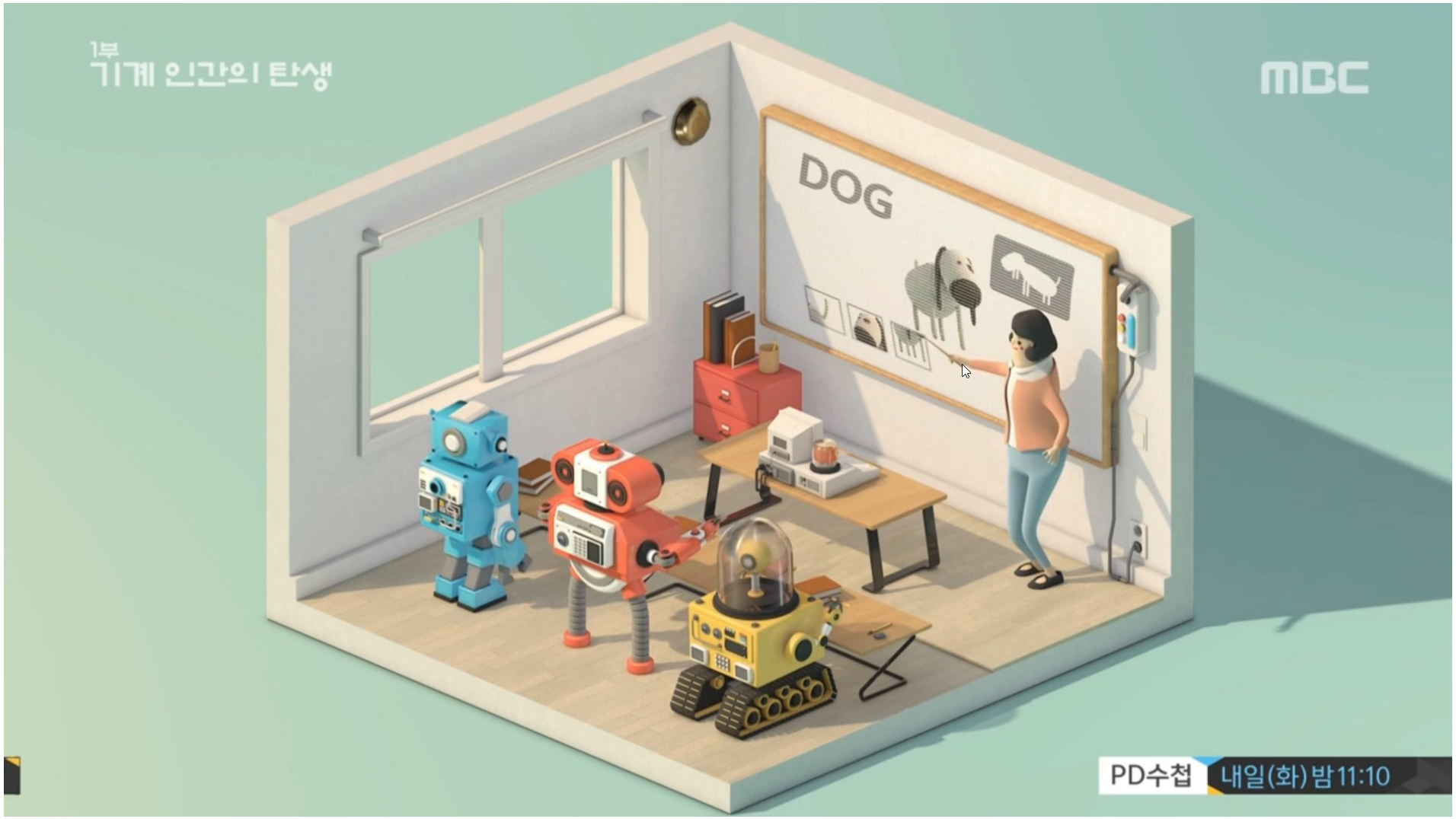
Machine Learning

- Extracting semantic information requires models of high complexity, which cannot be designed by hand.

- However, one can write a program that learns the task of extracting semantic information.

- Techniques used in practice consist of:
  - defining a parametric model with high capacity,
  - optimizing its parameters, by "making it work" on the training data.

어떻게 학습할 것인가?



https://youtu.be/f_uwKZIAeM0

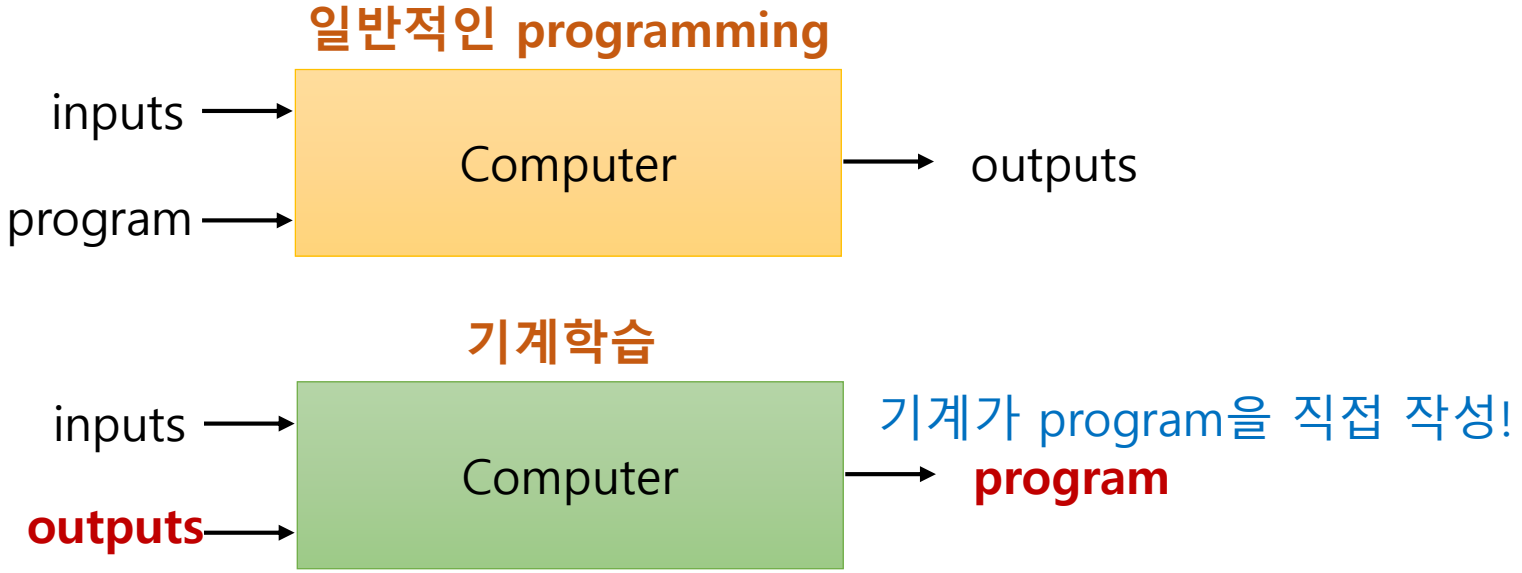
Credit : MBC 기계 인간의 탄생

기계학습(Machine Learning) – Data Driven Approach

- 기계학습(Machine Learning)

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

기계 학습은 "컴퓨터에 명시적으로 프로그래밍하지 않고 학습 할 수 있는 능력을 부여하는" 컴퓨터 과학의 하위 분야입니다.



Machine learning ⊆ artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.
Subfields: vision, robotics, machine learning, natural language processing, planning, …

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING

Classification, regression

UNSUPERVISED LEARNING

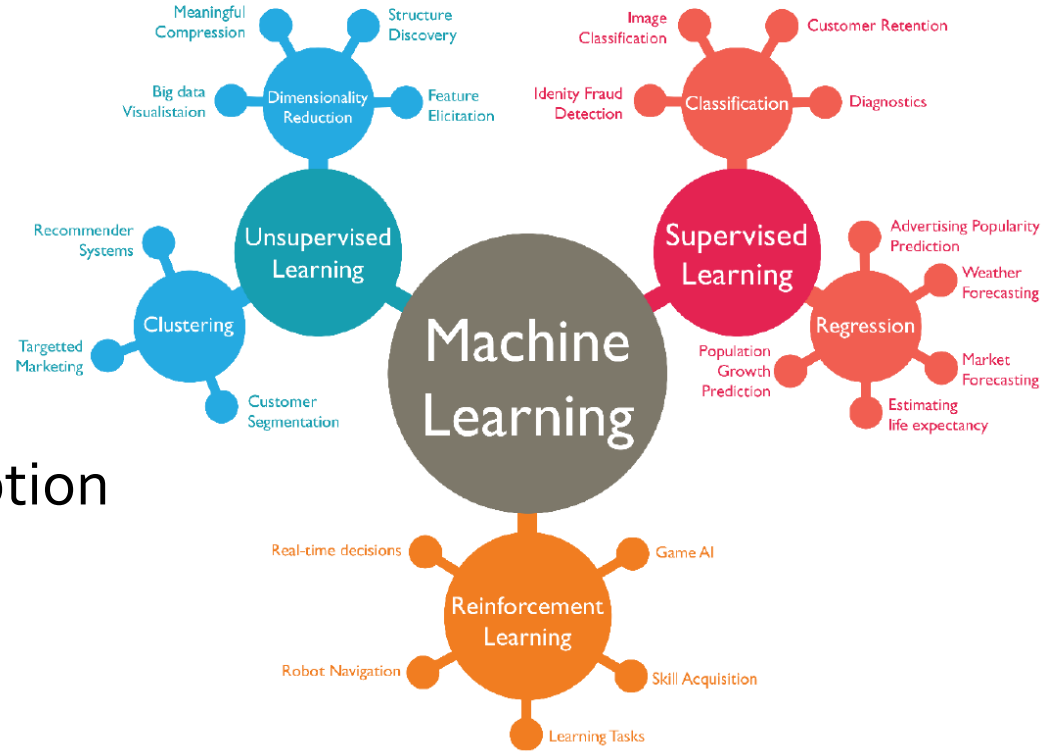
Clustering, dimensionality reduction, recommendation

REINFORCEMENT LEARNING

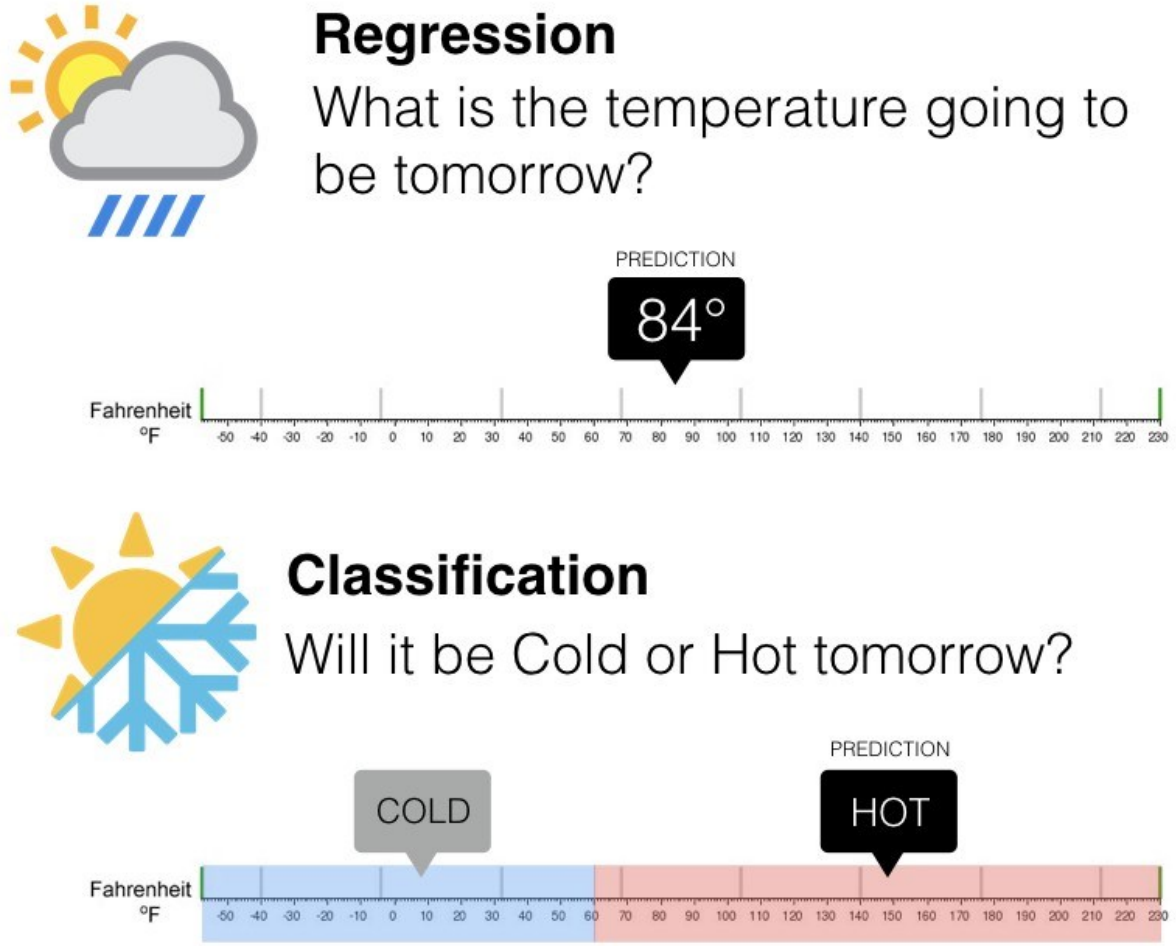
Reward maximization

Machine Learning의 종류

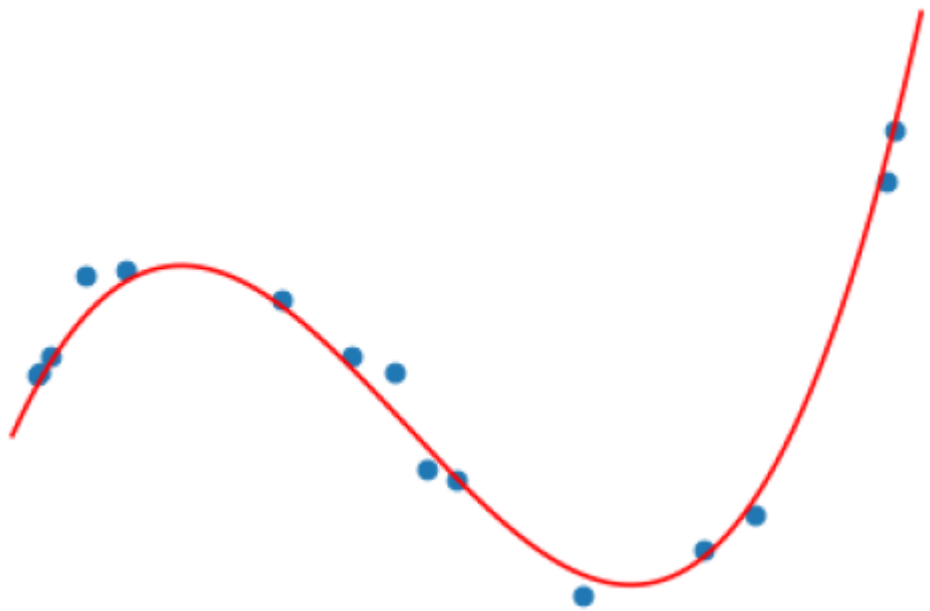
- Supervised Learning(지도학습)
  - Input 과 labels을 이용한 학습 → function approximator
  - 분류(classification), 회귀(regression)
- Unsupervised Learning(비지도학습)
  - Input만을 이용한 학습 → (shorter) description
  - 군집화(clustering), 압축(compression)
- Reinforcement Learning(강화학습)
  - Trial and error를 통한 학습 → sequential decision making
  - Action selection, policy learning



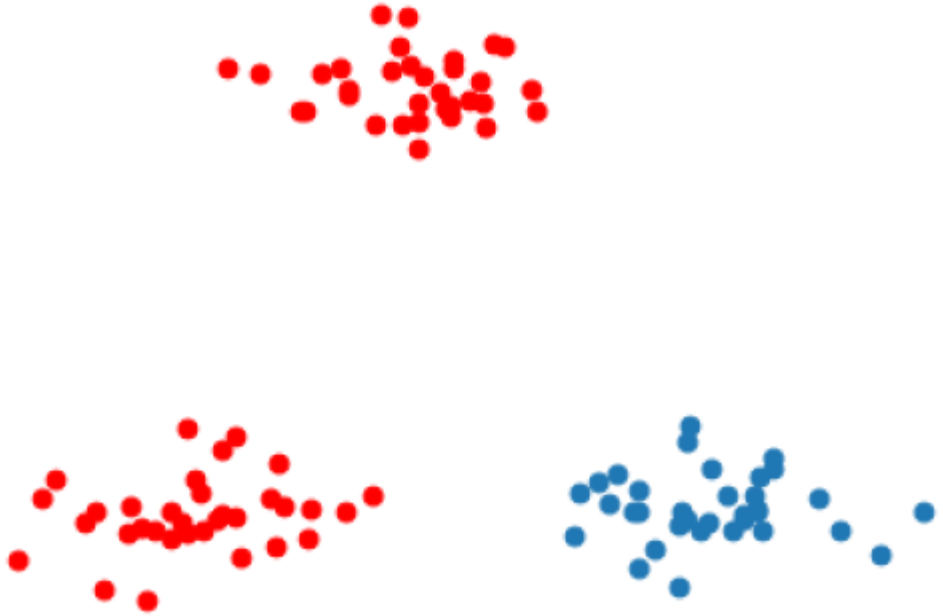
Supervised Learning – Regression vs Classification



Supervised Learning – Regression vs Classification



Regression aims at estimating relationships among (using continuous) variables



Classification consists in identifying a decision boundary between objects of distinct classes

https://github.com/glouppe/info8010-deep-learning

Supervised Learning – Regression vs Classification

- Classification: Given $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \{1, \ldots, C\}$, for $i = 1, \ldots, N$, we want to estimate for any new $\mathbf{x}$,

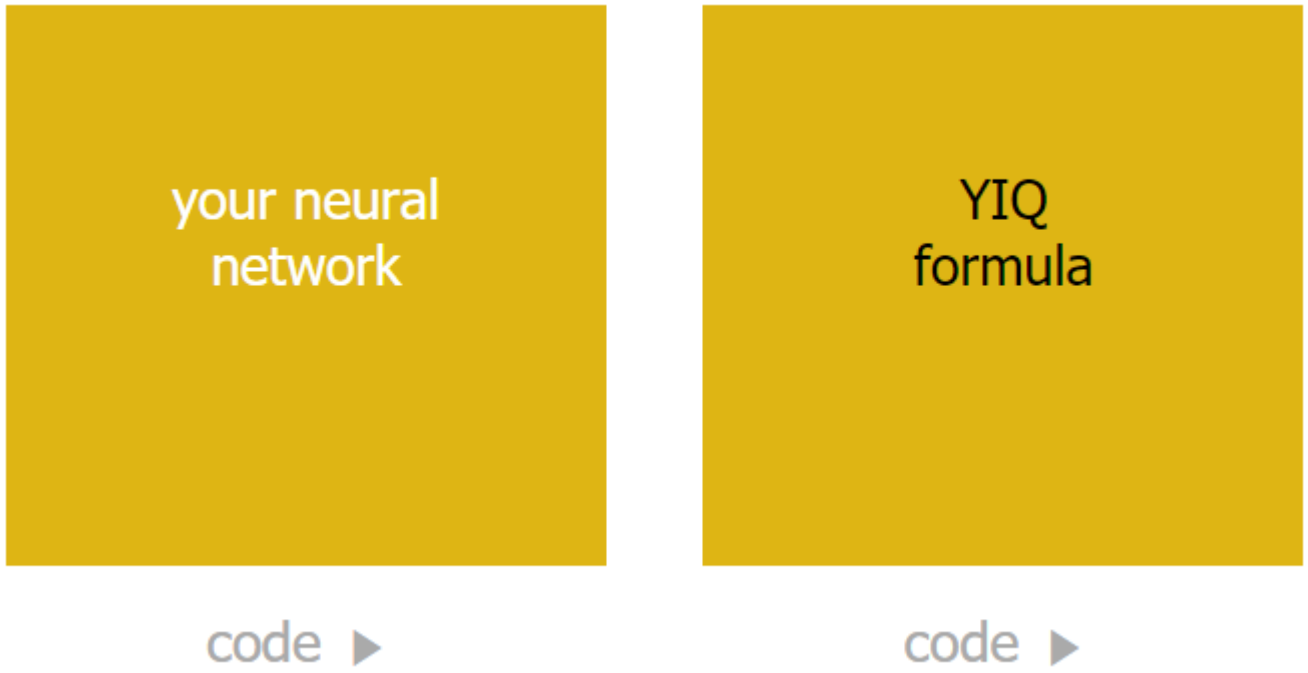
$$\arg\max_y P(Y = y | X = \mathbf{x}).$$

- Regression: Given $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} = \mathbb{R}^p \times \mathbb{R}$, for $i = 1, \ldots, N$, we want to estimate for any new $\mathbf{x}$,

$$\mathbb{E}\left[Y | X = \mathbf{x}\right].$$

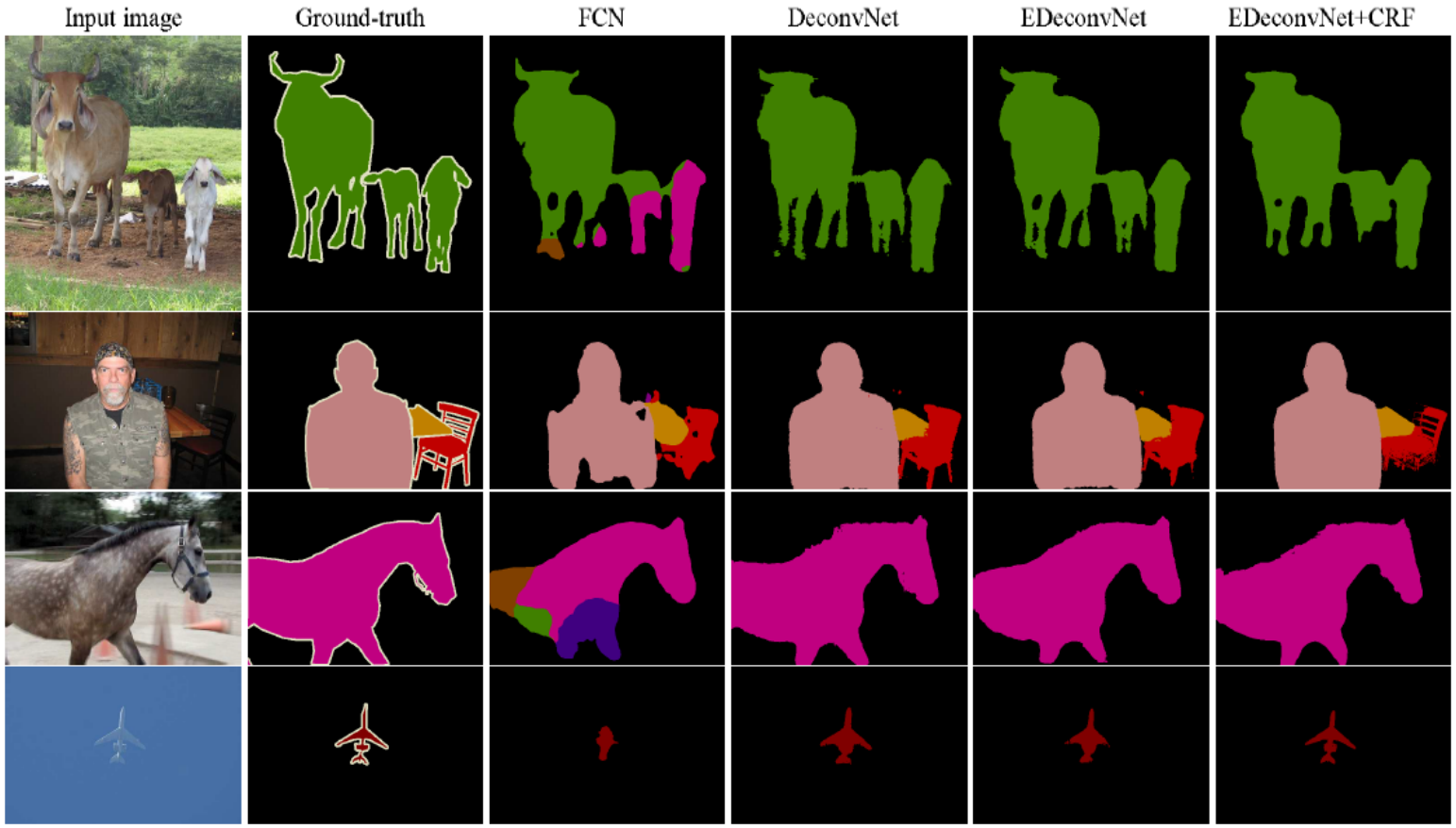
https://github.com/glouppe/info8010-deep-learning

Supervised Learning



Object Detection

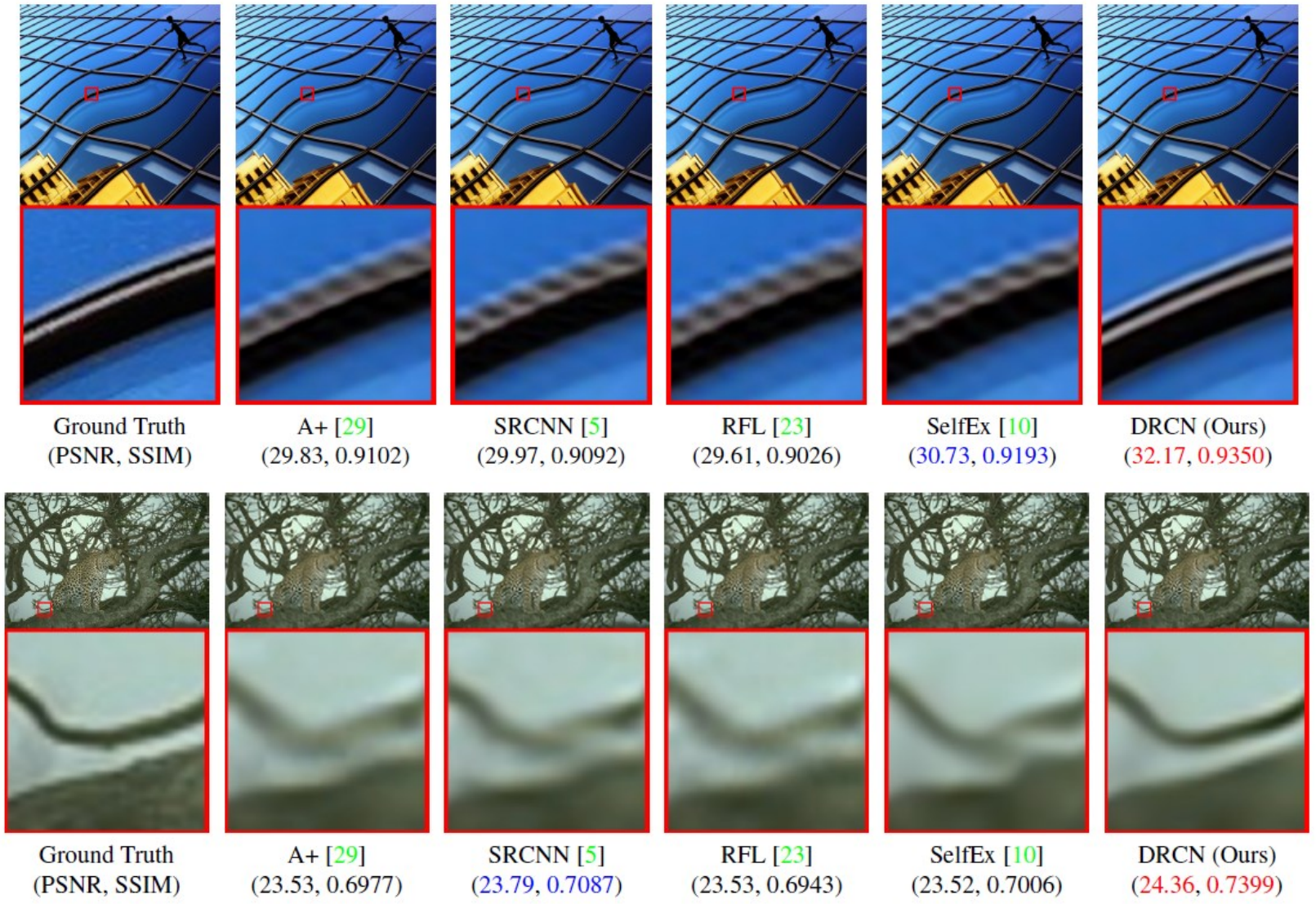
Segmentation



(a) Examples that our method produces better results than FCN [19].

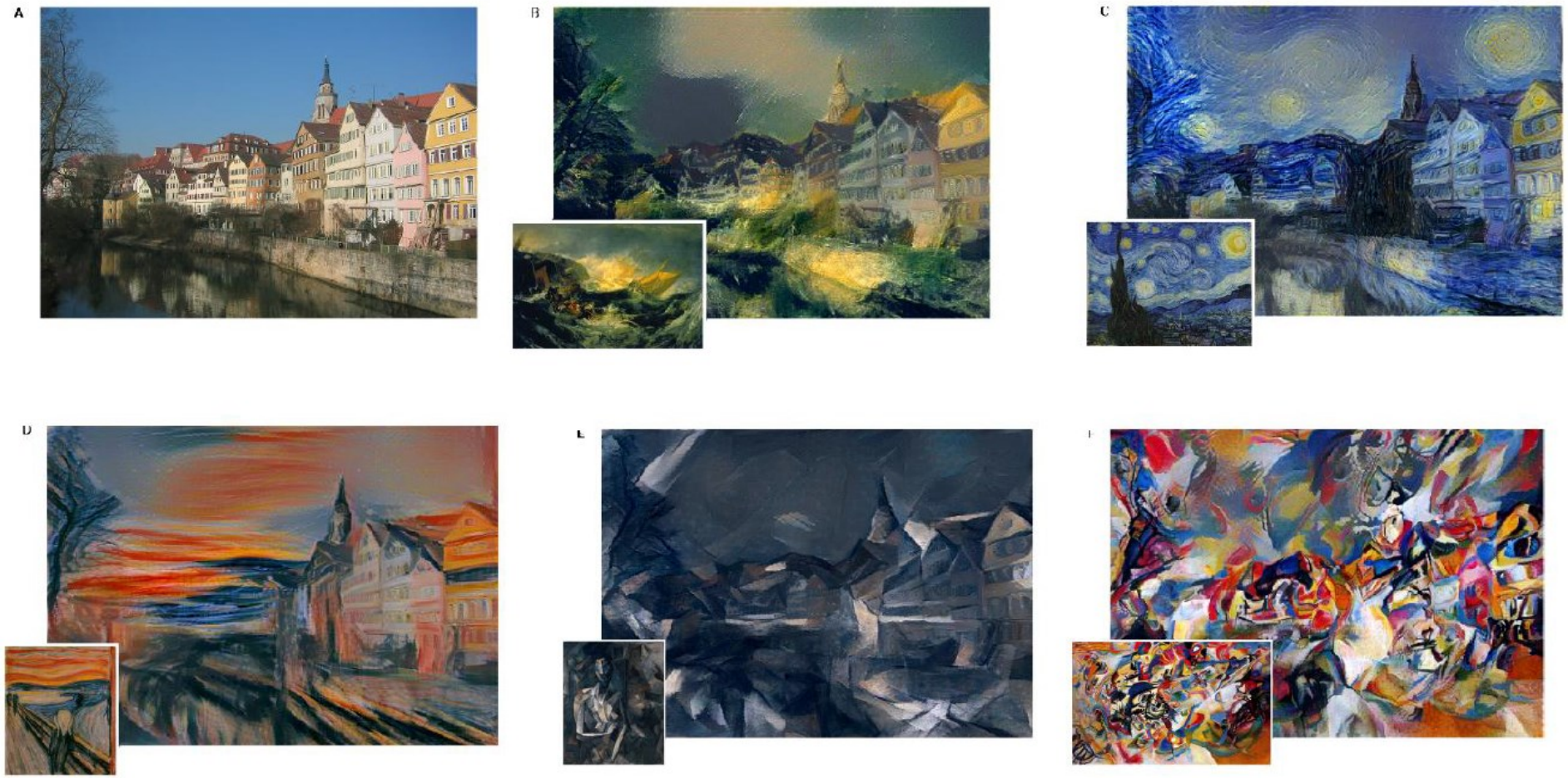
"Learning Deconvolution Network for Semantic Segmentation

Super Resolution



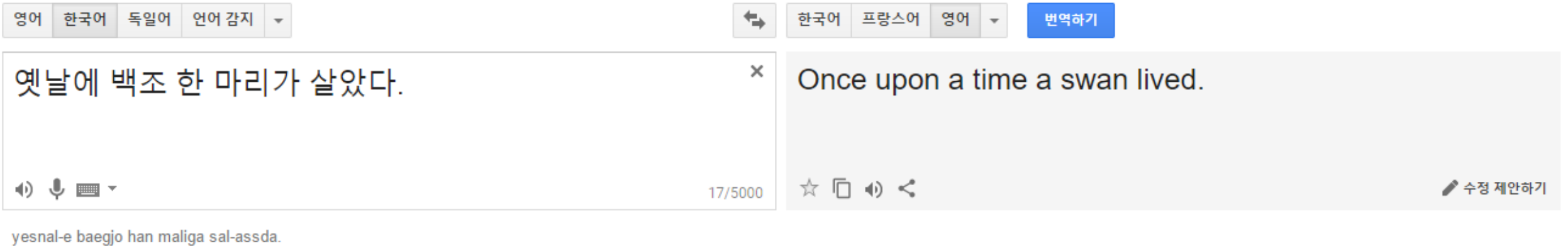
"Deeply-Recursive Convolutional Network for Image Super-Resolution"

Artistic Style Transfer



"Image Style Transfer Using Convolutional Neural Networks"

Machine Translation



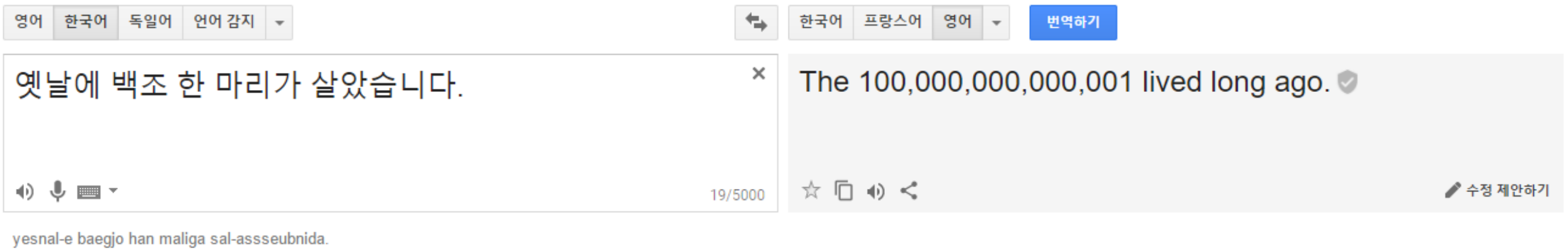
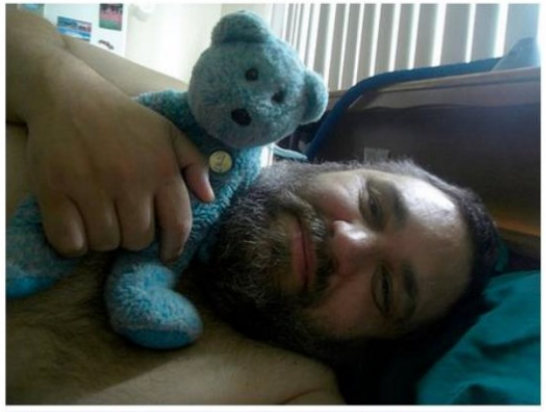


Image Captioning

a group of people standing around a room with remotes
logprob: -9.17

a young boy is holding a baseball bat
logprob: -7.61

a cow is standing in the middle of a street
logprob: -8.84



a baby laying on a bed with a stuffed bear
logprob: -8.66

a young boy is holding a baseball bat
logprob: -7.65

a woman holding a teddy bear in front of a mirror
logprob: -9.65

Visual QnA

Q: What is the boy holding?

DPPnet: surfboard

DPPnet: bat

Q: What is the animal doing?

DPPnet: resting (relaxing)

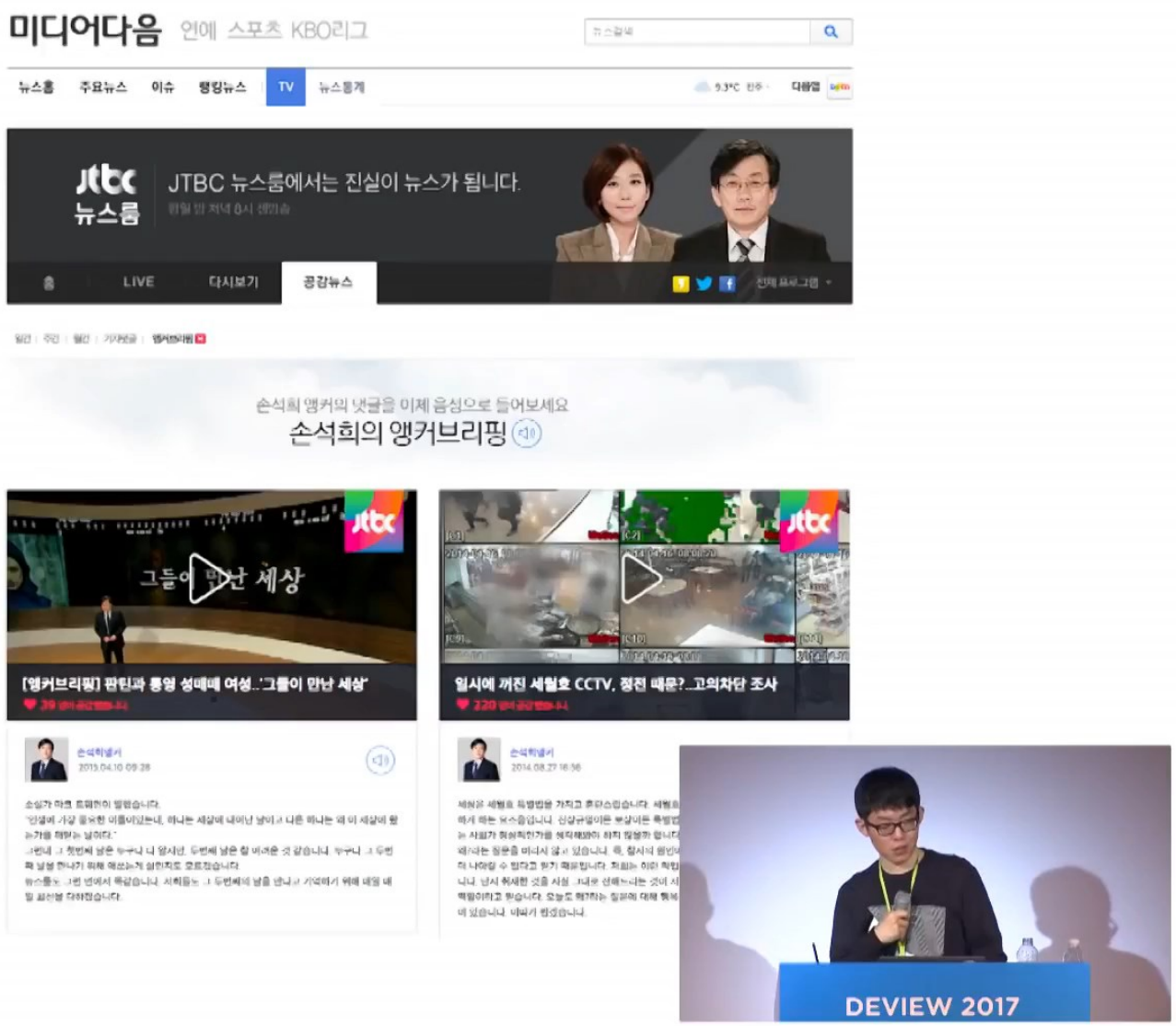
DPPnet: swimming (fishing)

"Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction"

Auto Speech Recognition

Text to Speech

kakao



Unsupervised Learning

- 아래 음식을 둘로 분류하면?


(1)


(2)


(3)


(4)


(5)

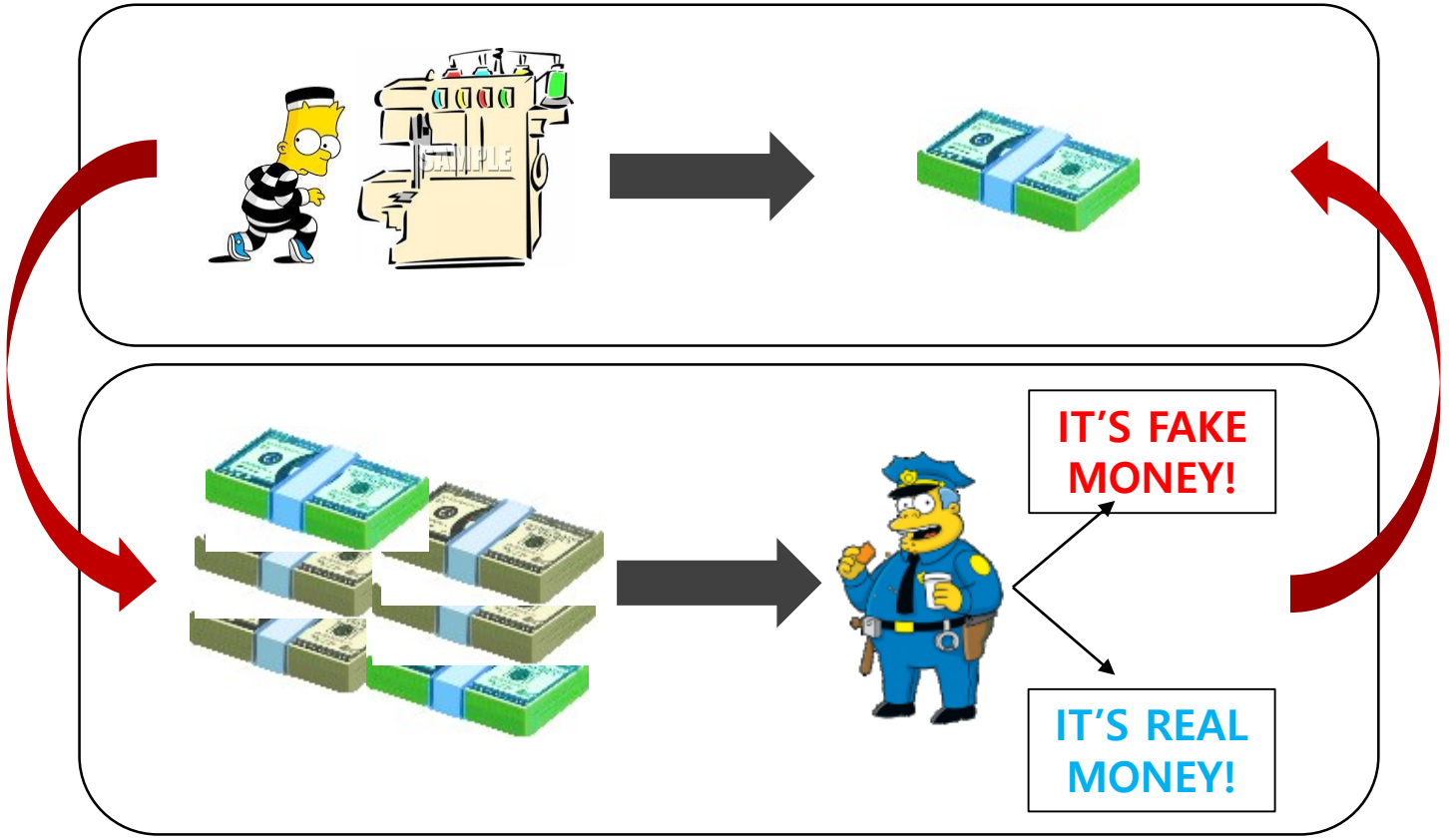

(6)


(7)

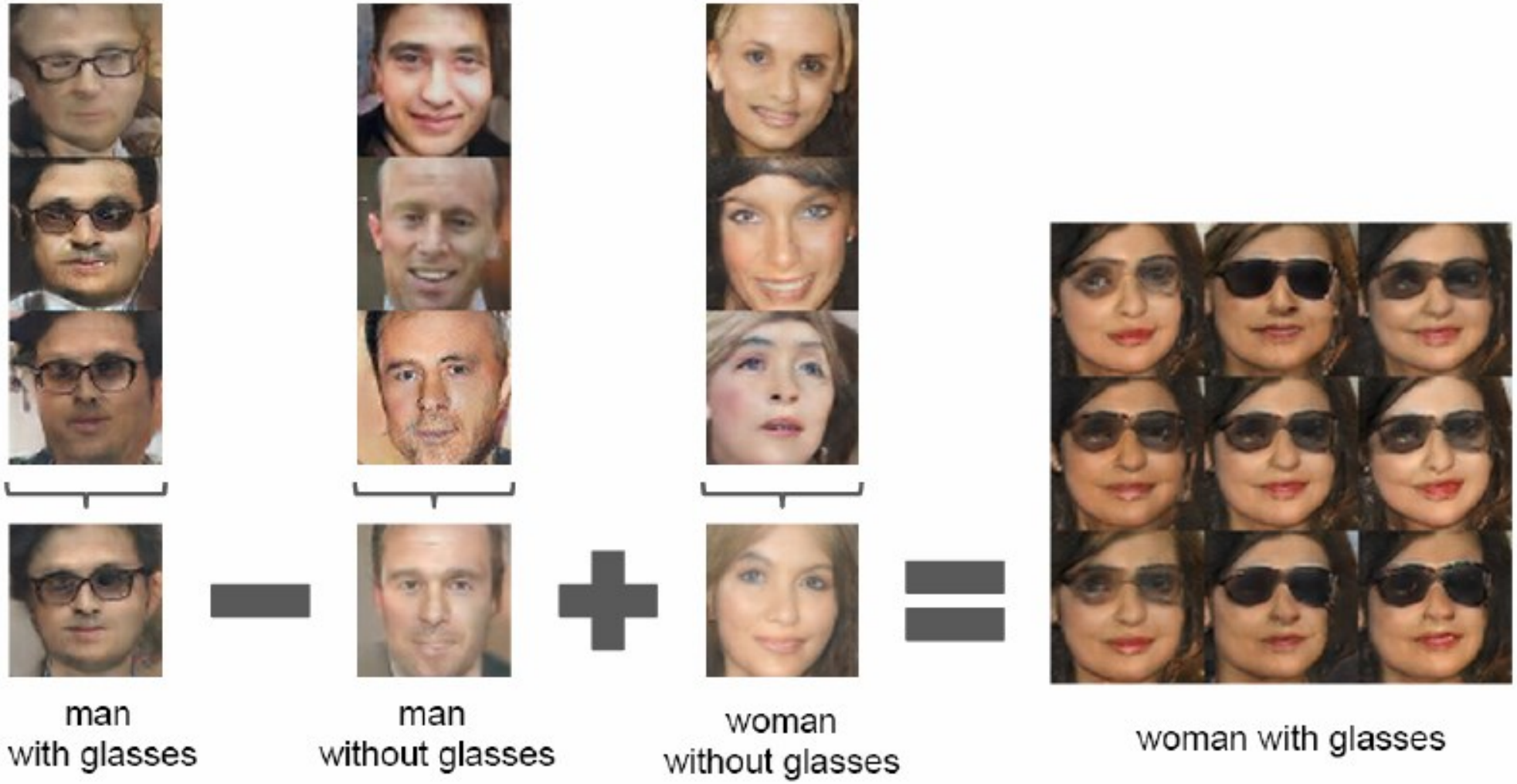

(8)

Generative Adversarial Network

- Counterfeiters vs Police Game



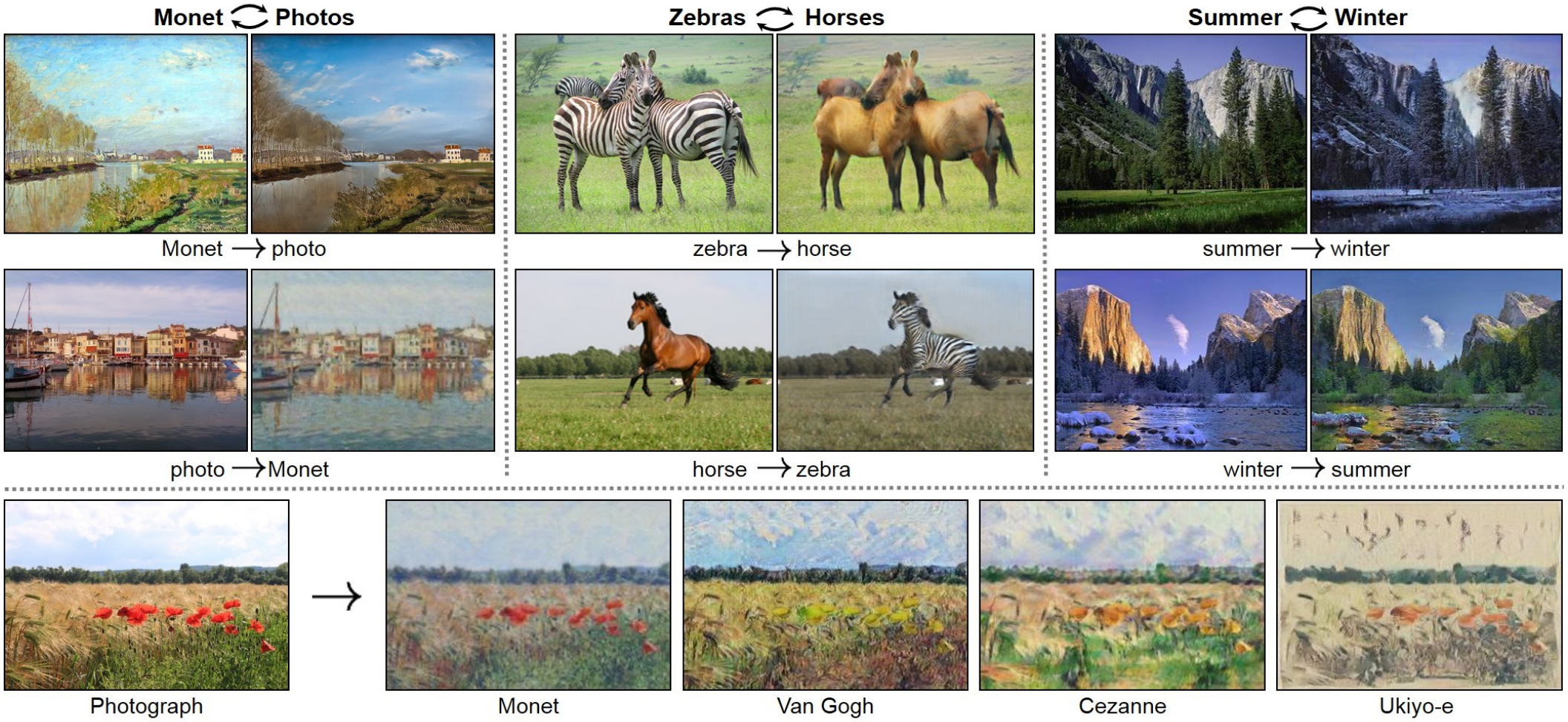
DCGAN


man
with glasses
man
without glasses
woman
without glasses
woman with glasses

Text to Image Generation

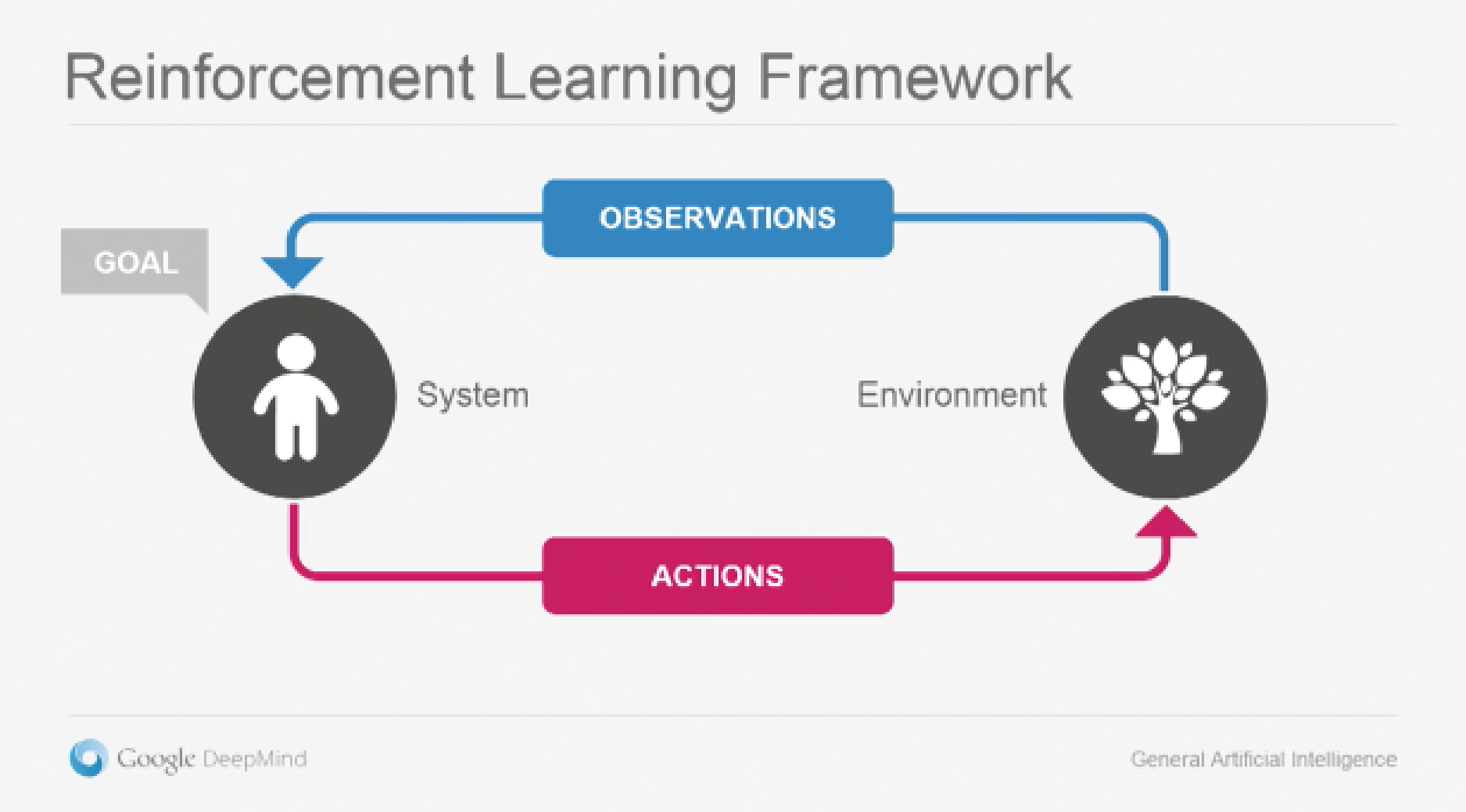
"StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks "

CycleGAN

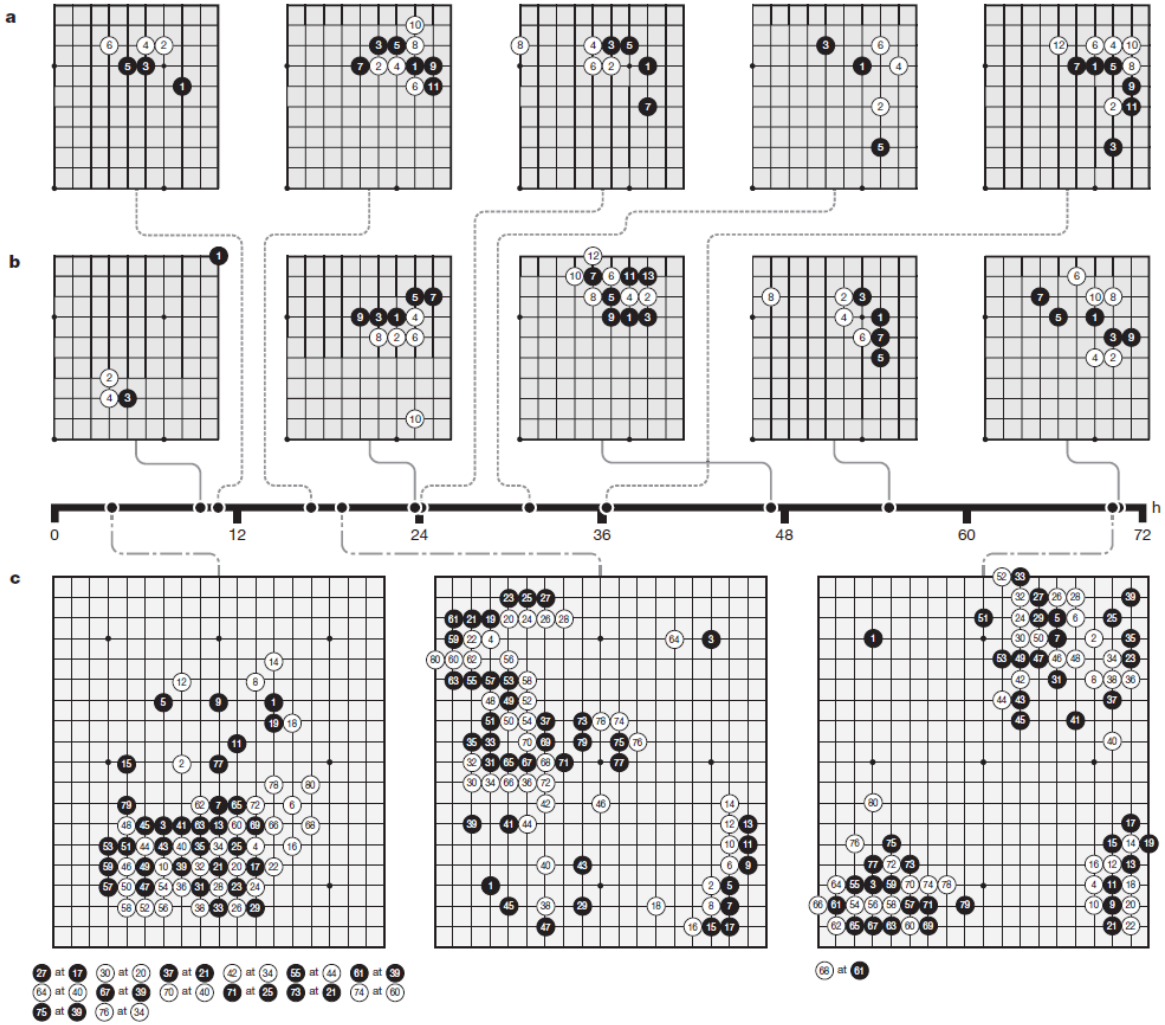

Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Video Generation

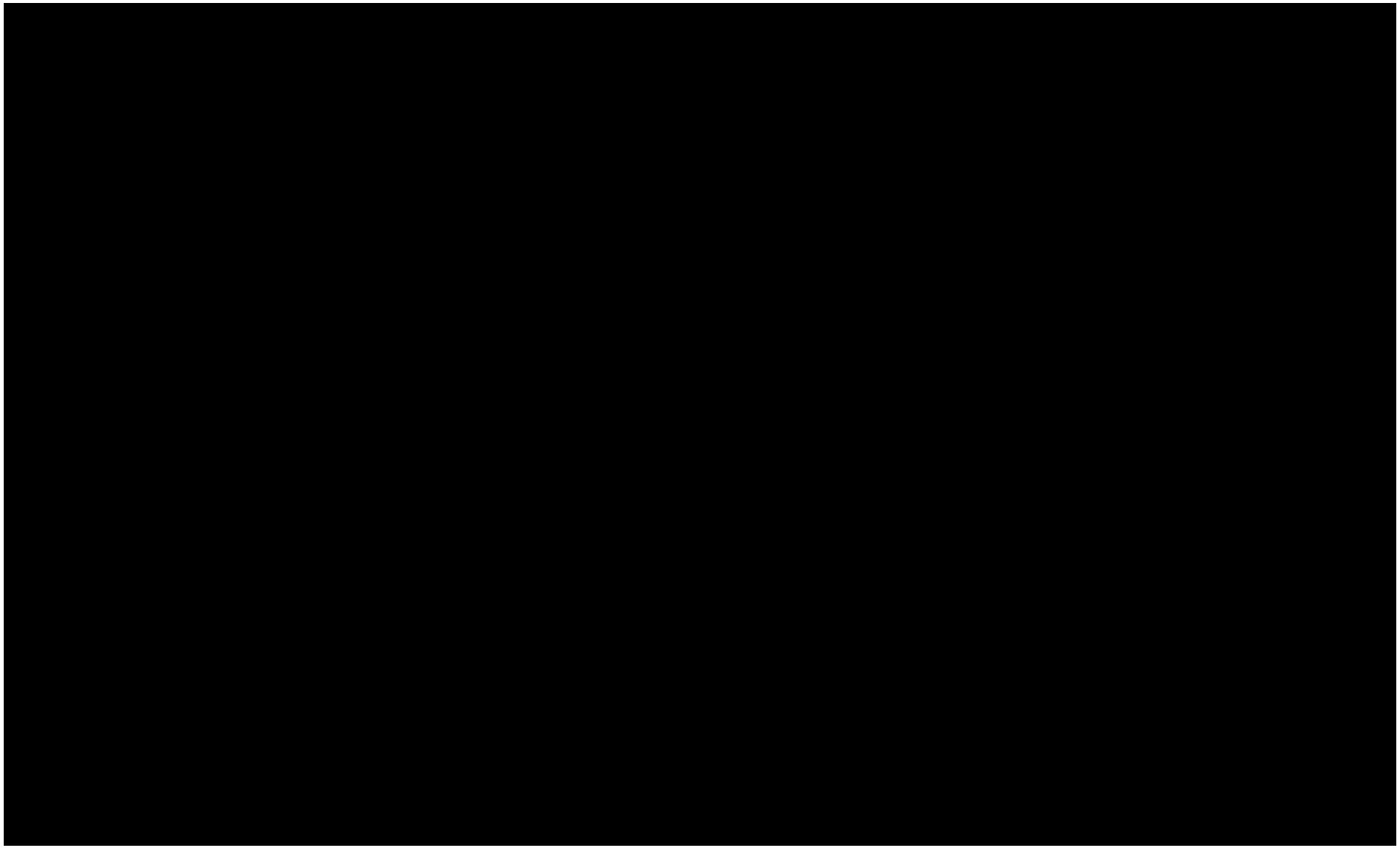
Reinforcement Learning



AlphaGo Zero



Atari Games



https://youtu.be/SuooulLpjDo

Atari Games

Space Invaders

https://youtu.be/Q70ulPJW3Gk

Other Games

- Super Mario World
  - https://youtu.be/L4KBBAwF_bE
  - https://youtu.be/qv6UVOQ0F44
- Cookie Run
  - https://youtu.be/exXD6wJLJ6s
- Starcraft I
  - https://youtu.be/GgkmJDjeJtw
- GTA
  - https://youtu.be/X4u2DCOLoIg

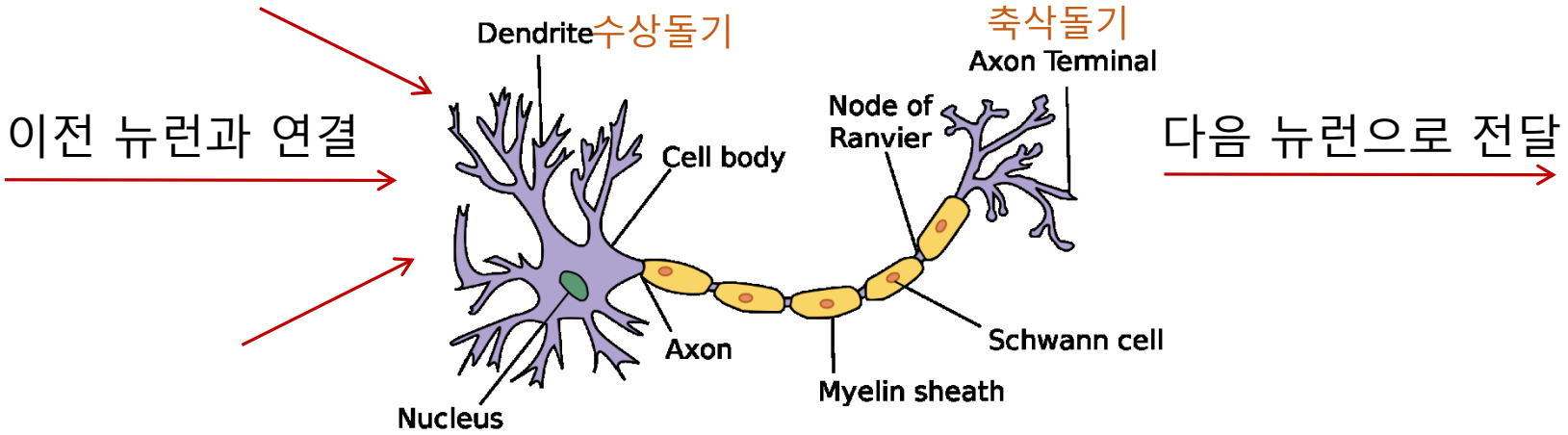
Autonomous Driving

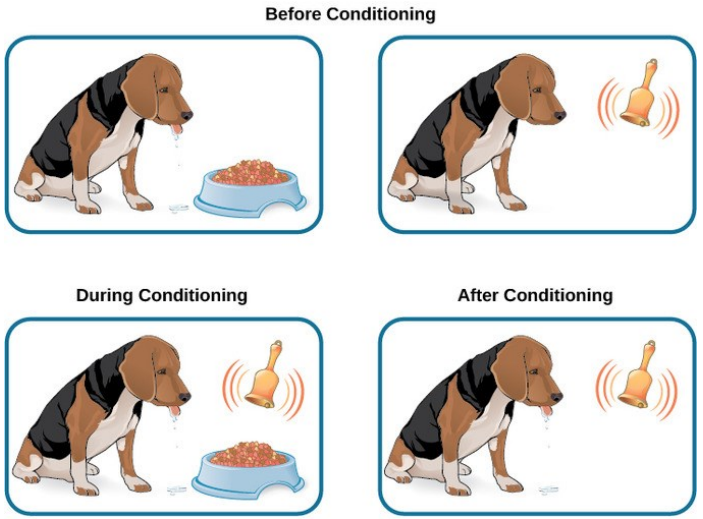
Quiz

- □와 △에 들어갈 정수는?
  - 3 x □ + 2 x △ = 1
  - 1 x □ + 4 x △ = -3
  - 5 x □ + 5 x △ = 0
  - 8 x □ + 3 x △ = 5
- **□ = 1, △ = -1**

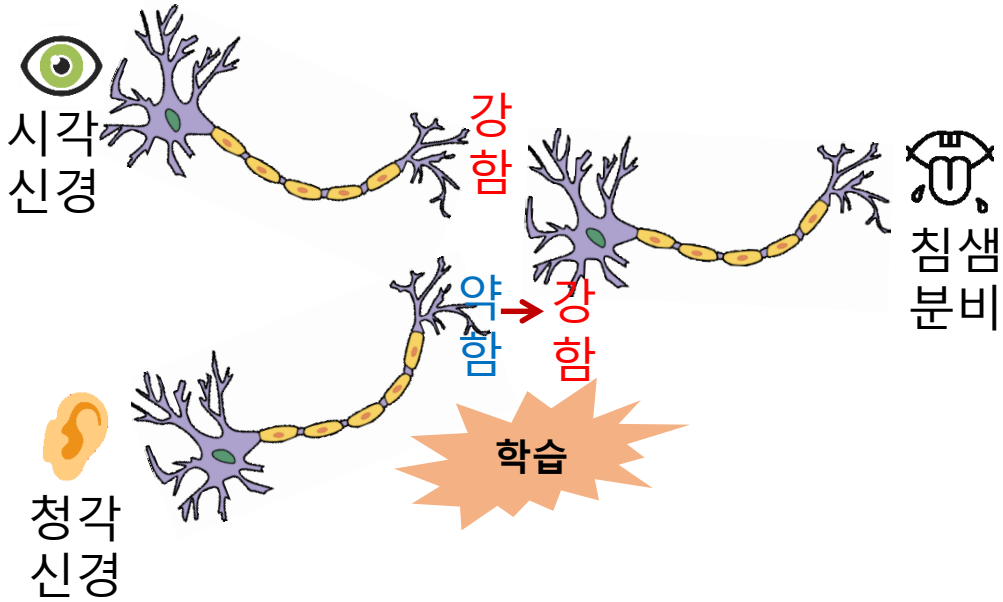
- (3, 2), (1, 4), (5, 5), (8, 3) 은 input data, 1, -3, 0, 5 는 label이다
- □와 △를 weight라고 하며 이 weight 값을 기계가 스스로 학습을 통해 찾아내도록 하는 것이 neural network를 이용한 기계학습이 하는 일

뉴런과 사람의 학습

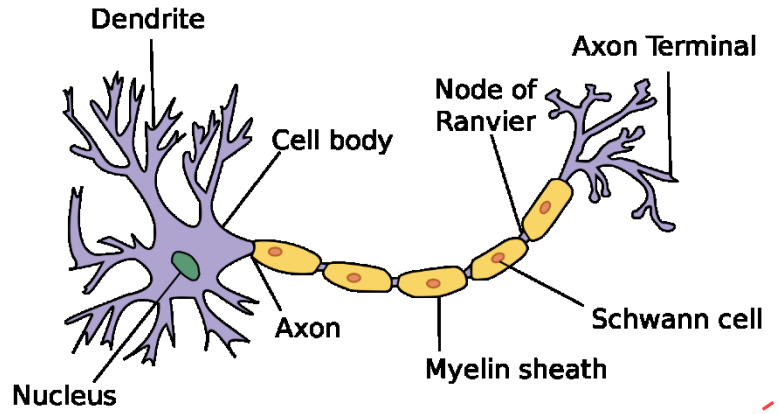


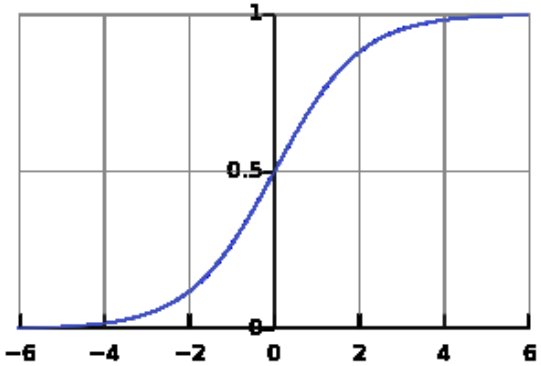


파블로프의 개 실험



Perceptron(Artificial Neural Network)





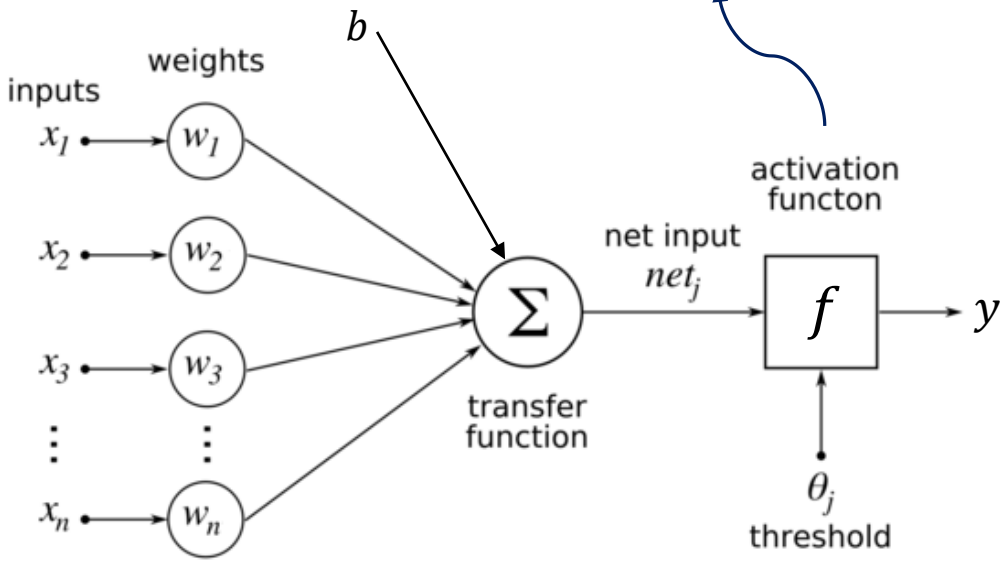
sigmoid activation function

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

$$y = f(\mathbf{wx} + \mathrm{b})$$

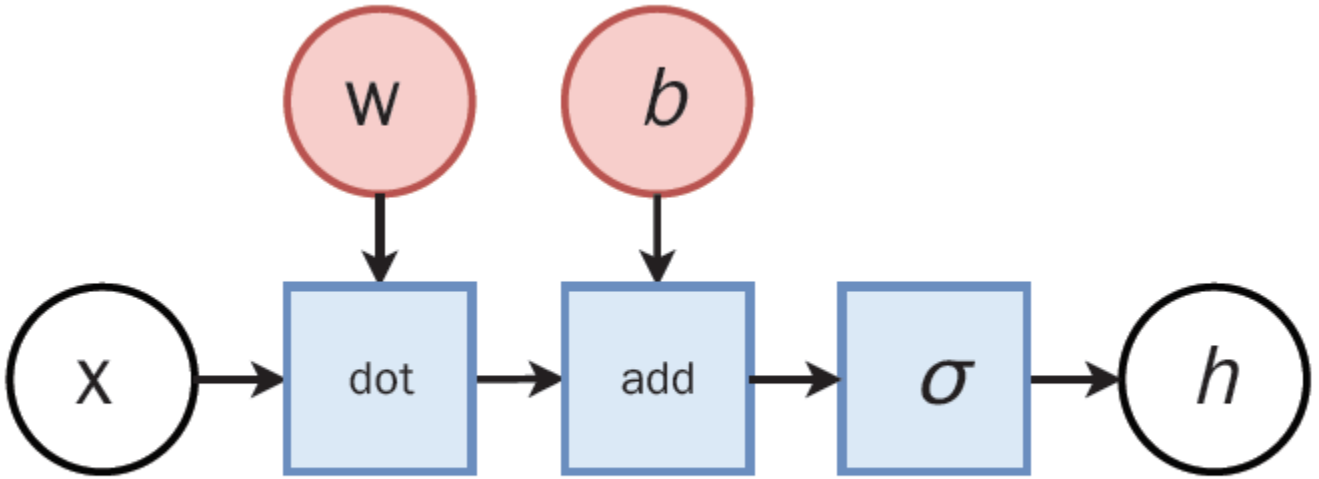
$$\boldsymbol{w} = [w_1\ w_2\ w_3\ \ldots\ w_n]$$

$$\boldsymbol{x} = [x_1\ x_2\ x_3\ \ldots\ x_n]^T$$



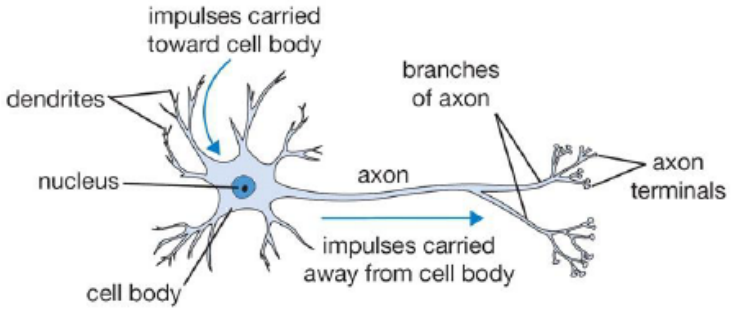
<Perceptron>

Computation Graph

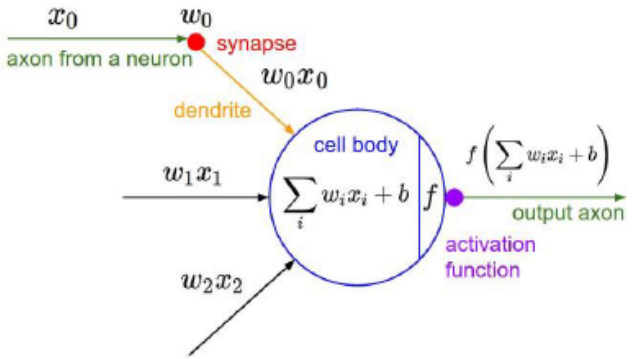


This unit is the lego brick of all neural networks!

Neuron vs Perceptron



- **Neuron:** computational building block for the brain

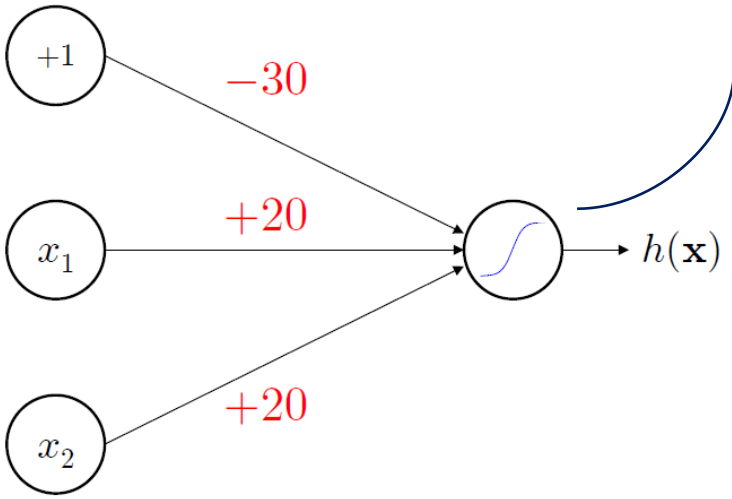


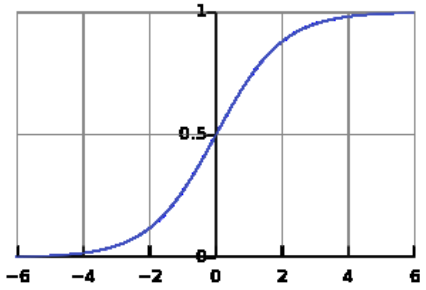
- **(Artificial) Neuron:** computational building block for the "neural network"

Key Difference:

- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no "layers". **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm**: ANNs use gradient descent for learning. We don't know what human brains use
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually never stop learning. ANNs first train then test.

Example of ANN(logical AND)

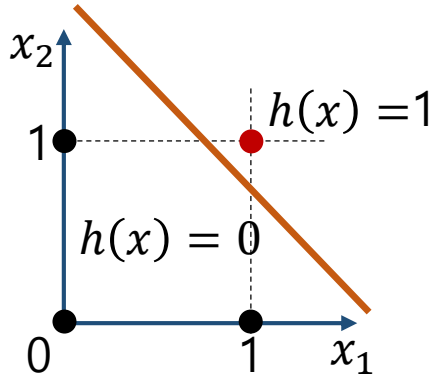




x_1	x_2	$h(\mathbf{x})$
0	0	$\sigma(-30) \approx 0$
0	1	$\sigma(-10) \approx 0$
1	0	$\sigma(-10) \approx 0$
1	1	$\sigma(10) \approx 1$

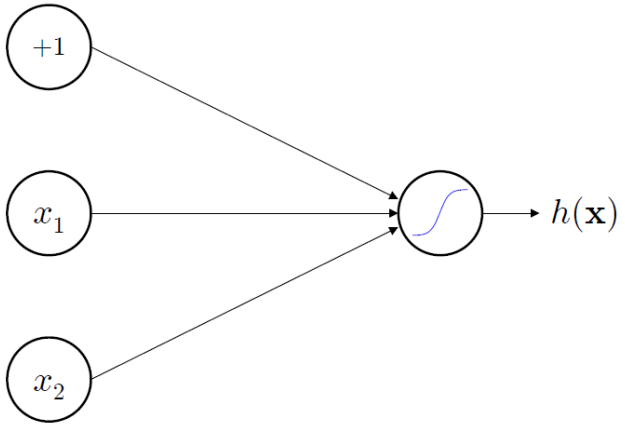
$$h(\mathbf{x}) = \sigma(-30 + 20x_1 + 20x_2)$$

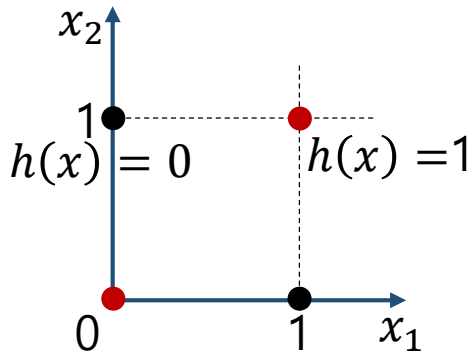
학습이란 이러한 weight 값(-30, 20, 20)을
기계 스스로 찾을 수 있도록 해주는 과정!



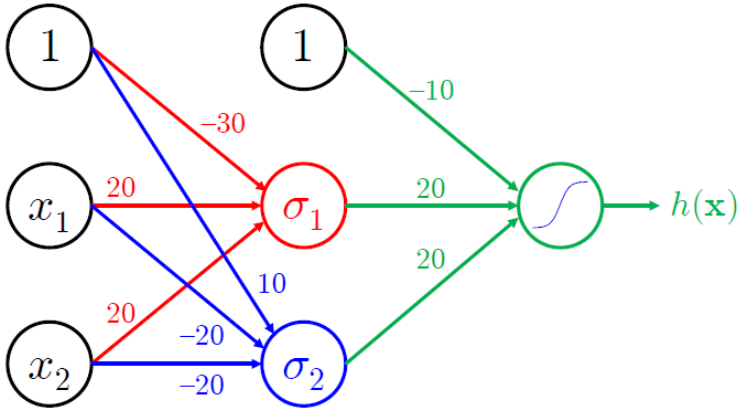
Logical XNOR with Perceptron

- Is it possible(Linearly separable)?





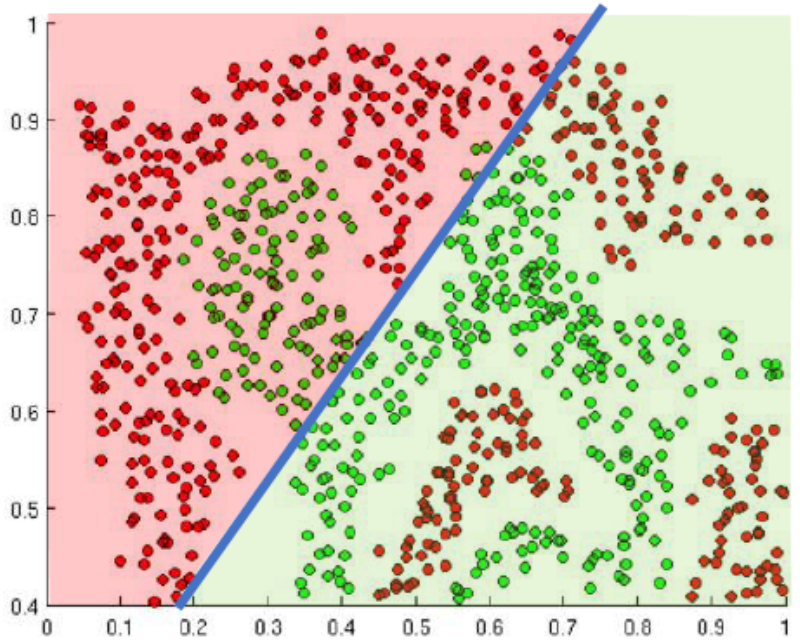
- We need more perceptrons and more layers



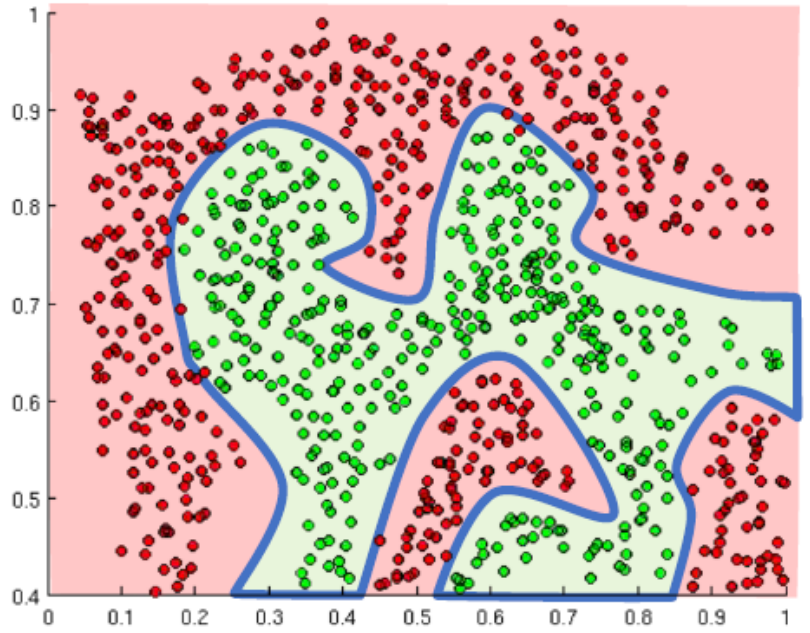
x_1	x_2	σ_1	σ_2	$h(\mathbf{x})$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*



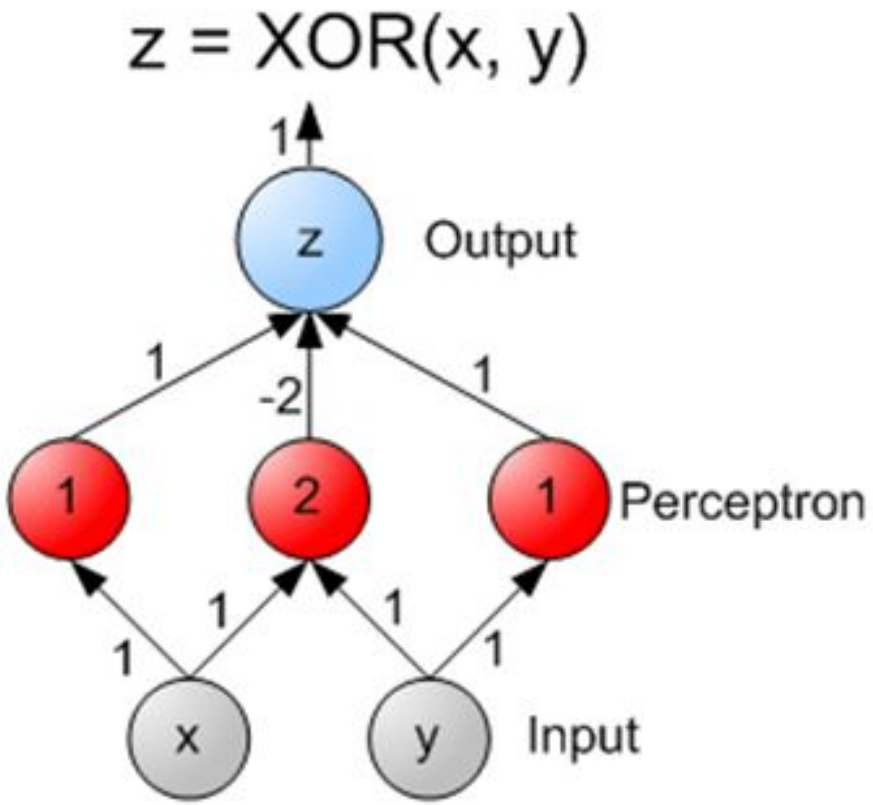
Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

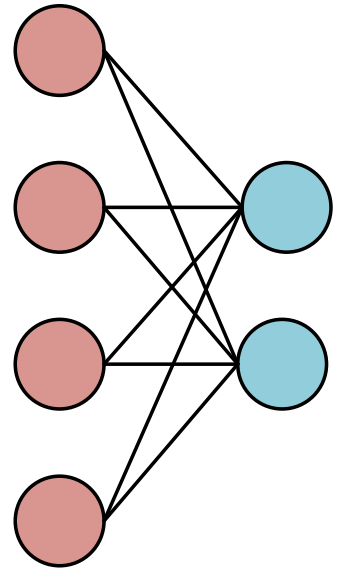
Universal Function Approximation

The *universal approximation theorem* for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, e.g. for the sigmoidal functions. (Wikipedia.org)



Single-Layer Perceptron & Multi-Layer Perceptron(MLP)

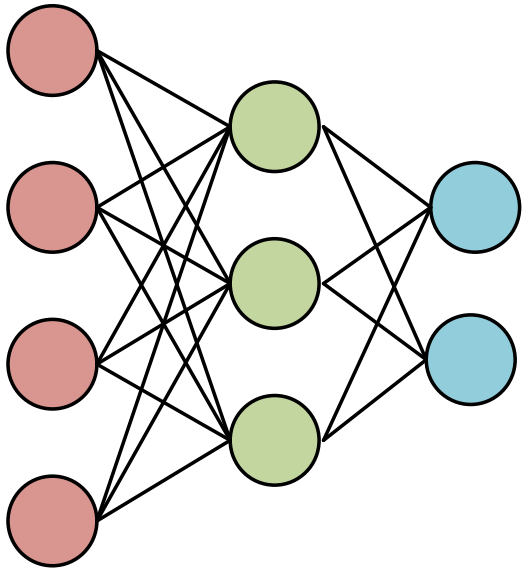
Single-Layer Perceptron



input layer

output layer

Multi-Layer Perceptron



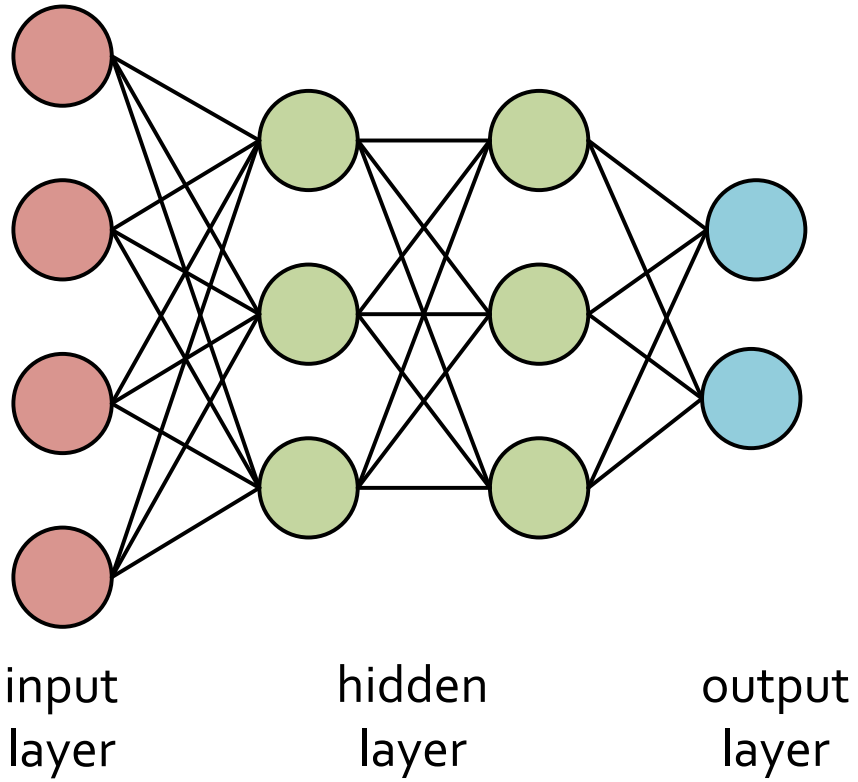
input layer

hidden layer

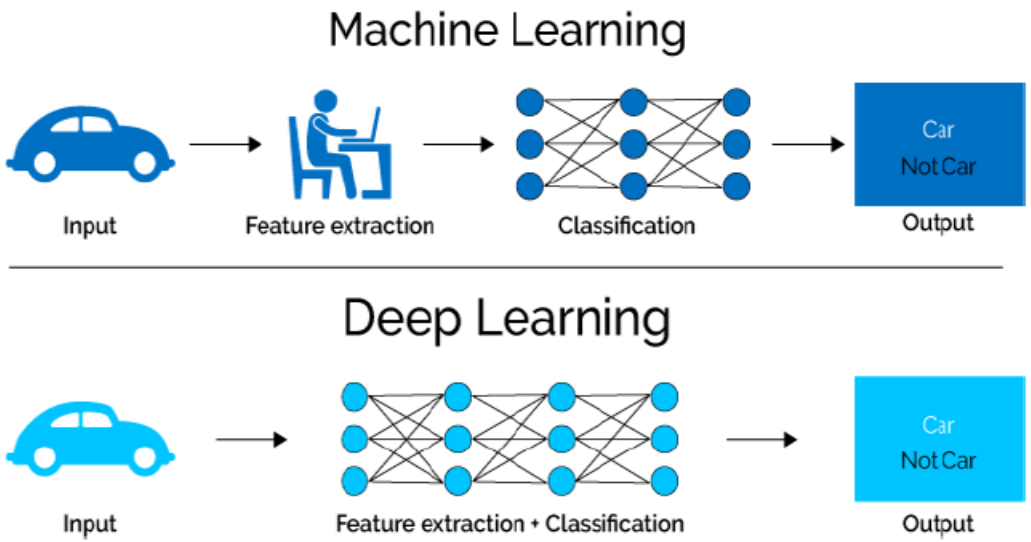
output layer

Deep Learning

- Deep Neural Network을 이용한 Machine Learning 방법
  - Deep Neural Network : Hidden layer 수가 2개 이상인 network



Machine Learning vs Deep Learning



ML : Audio → Acoustic Model → Phonetic Model → Language Model → Text

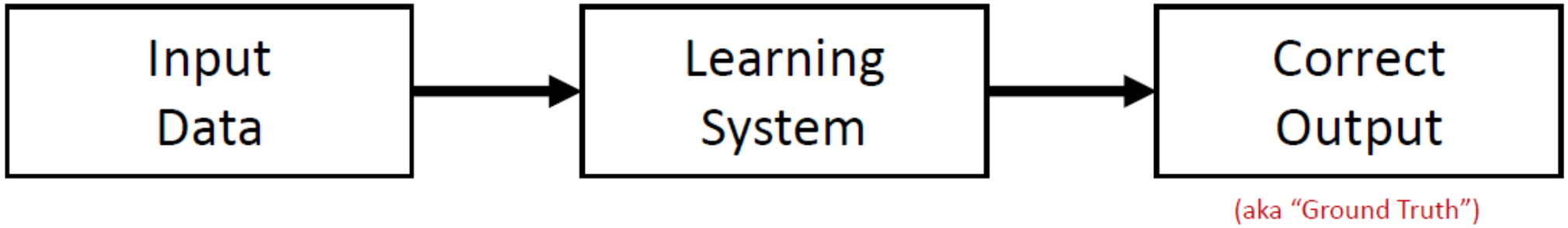
DL : Audio → Deep Networks → Text (End to End Learning)

ML : Pixels → Key Points → SIFT Features → Deformable Part Model → Label

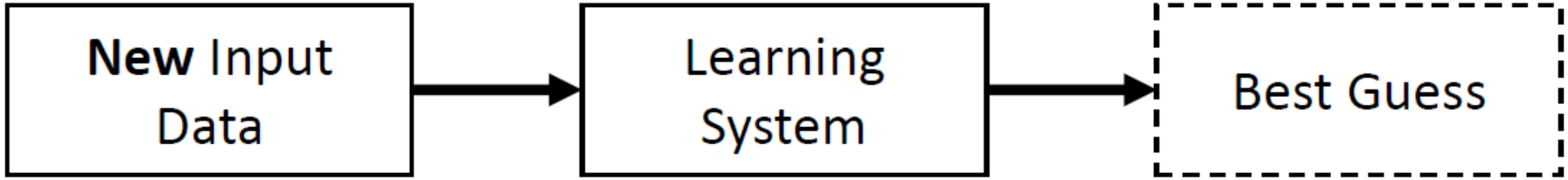
DL : Pixels → Deep Networks → Label (End to End Learning)

Training and Testing

Training Stage:



Testing Stage:

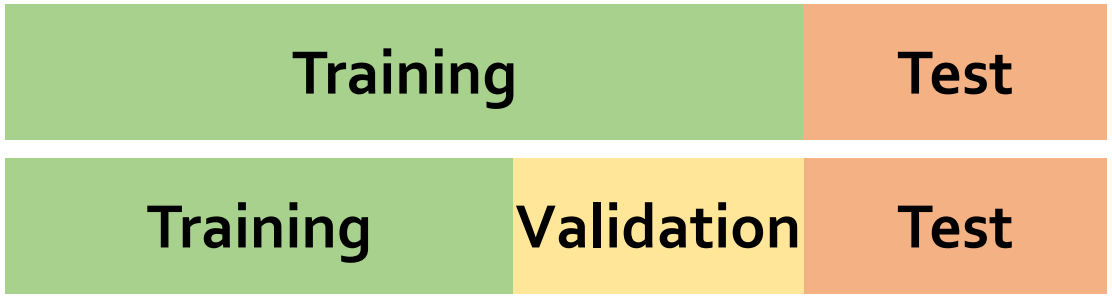


Dataset

- Training set, Test set

  or

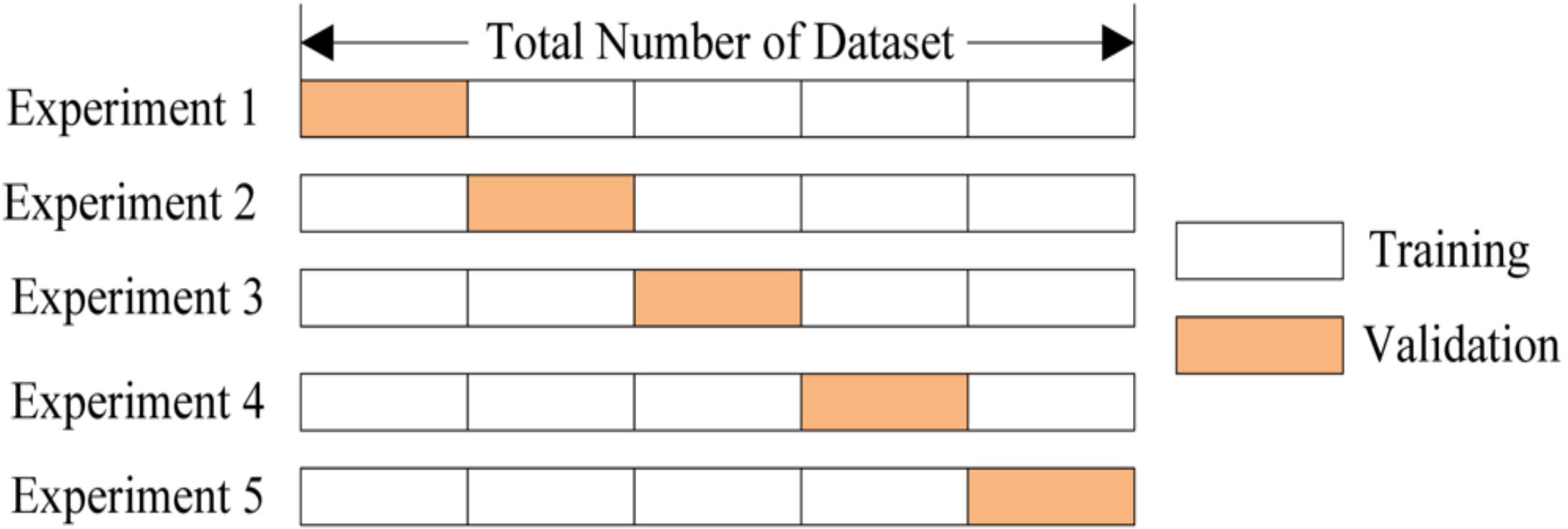
- Training set, Validation set, Test set



- Training → Tuning(validation) → Test

- In any case, DO NOT USE the test set before the training is over!

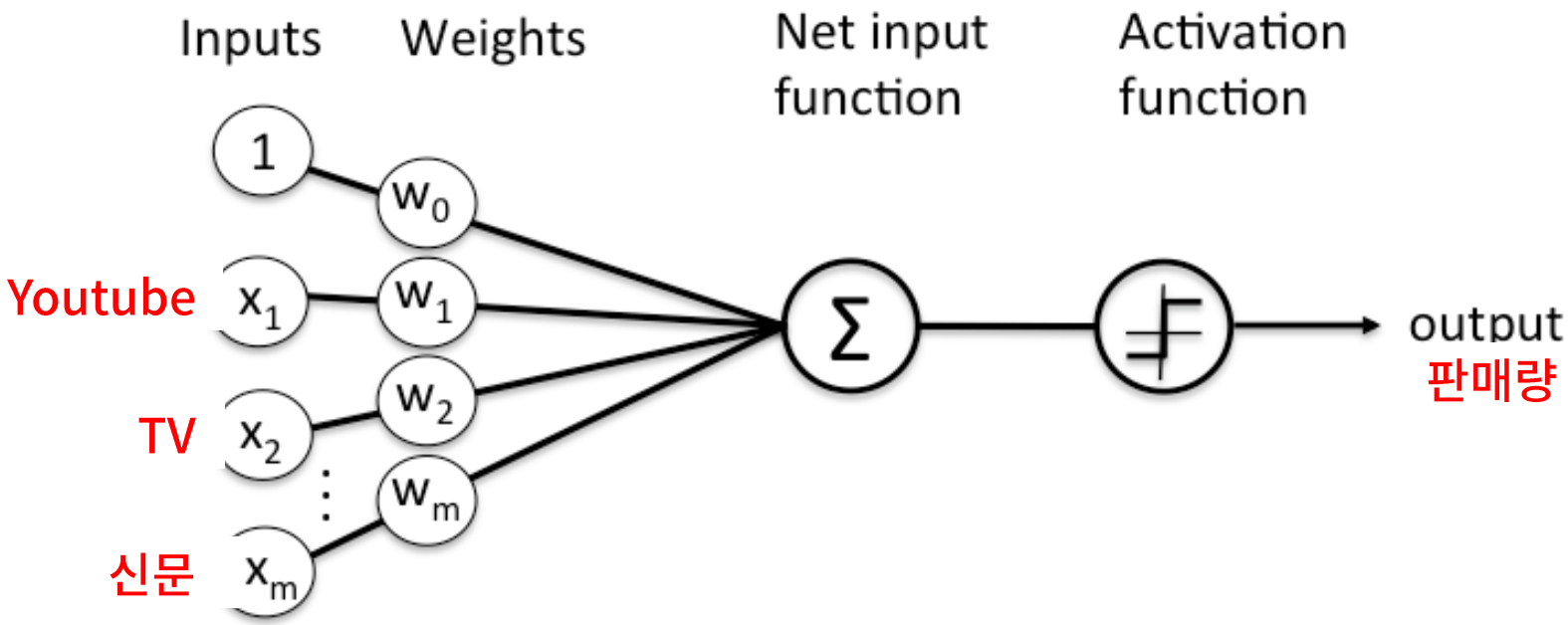
Cross-Validation



An Example of Perceptron

어떤 상품을 판매하는 회사에서

Youtube, TV, **신문 광고료에 따른 판매량을 예측**



Youtube	TV	신문	판매량
130	70	55	221
44	40	45	109
17	46	70	73
81	40	57	181
52	12	11	90
9	49	61	52
57	33	23	122
30	30	25	100
…	…	…	…

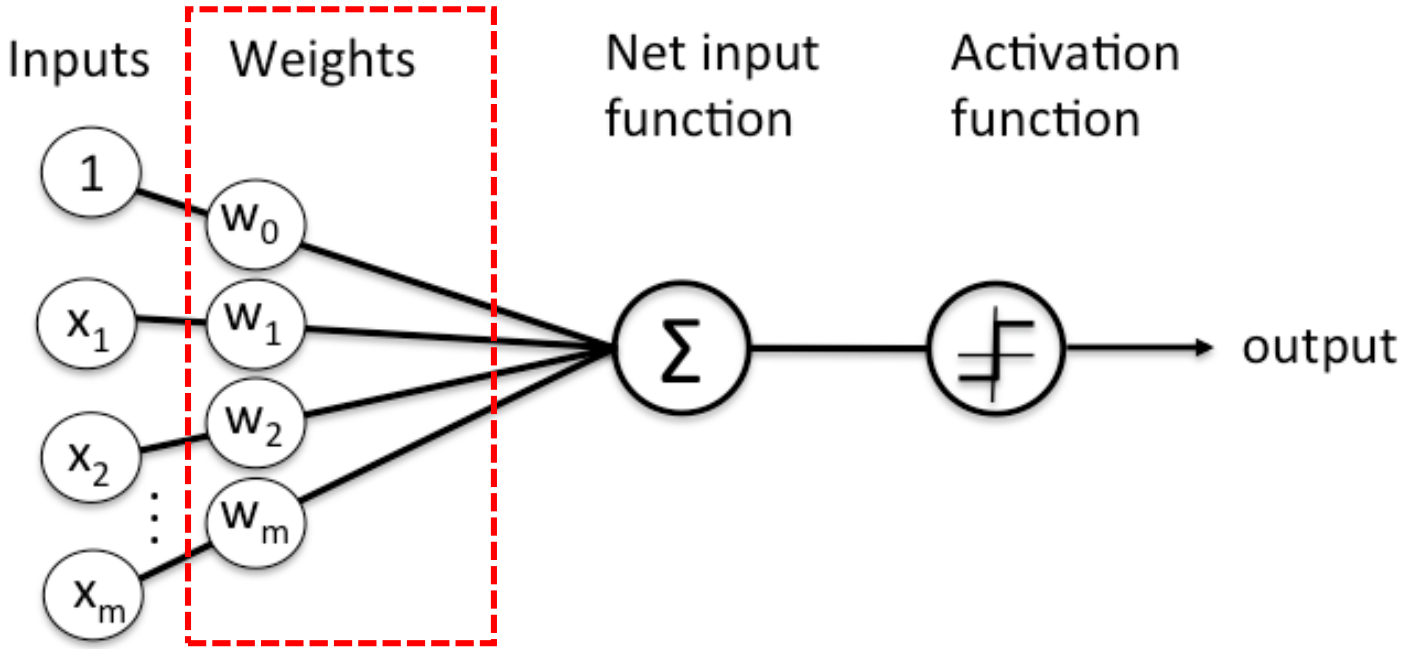
Weight Initialization

원하는 Weight를 어떻게 찾을까?

그 전에 아무것도 모르는 상태에서 Weight의 초기값은 어떻게?

잘 모르겠으니 일단 Random한 값으로 시작하자!

– Random Initialization!

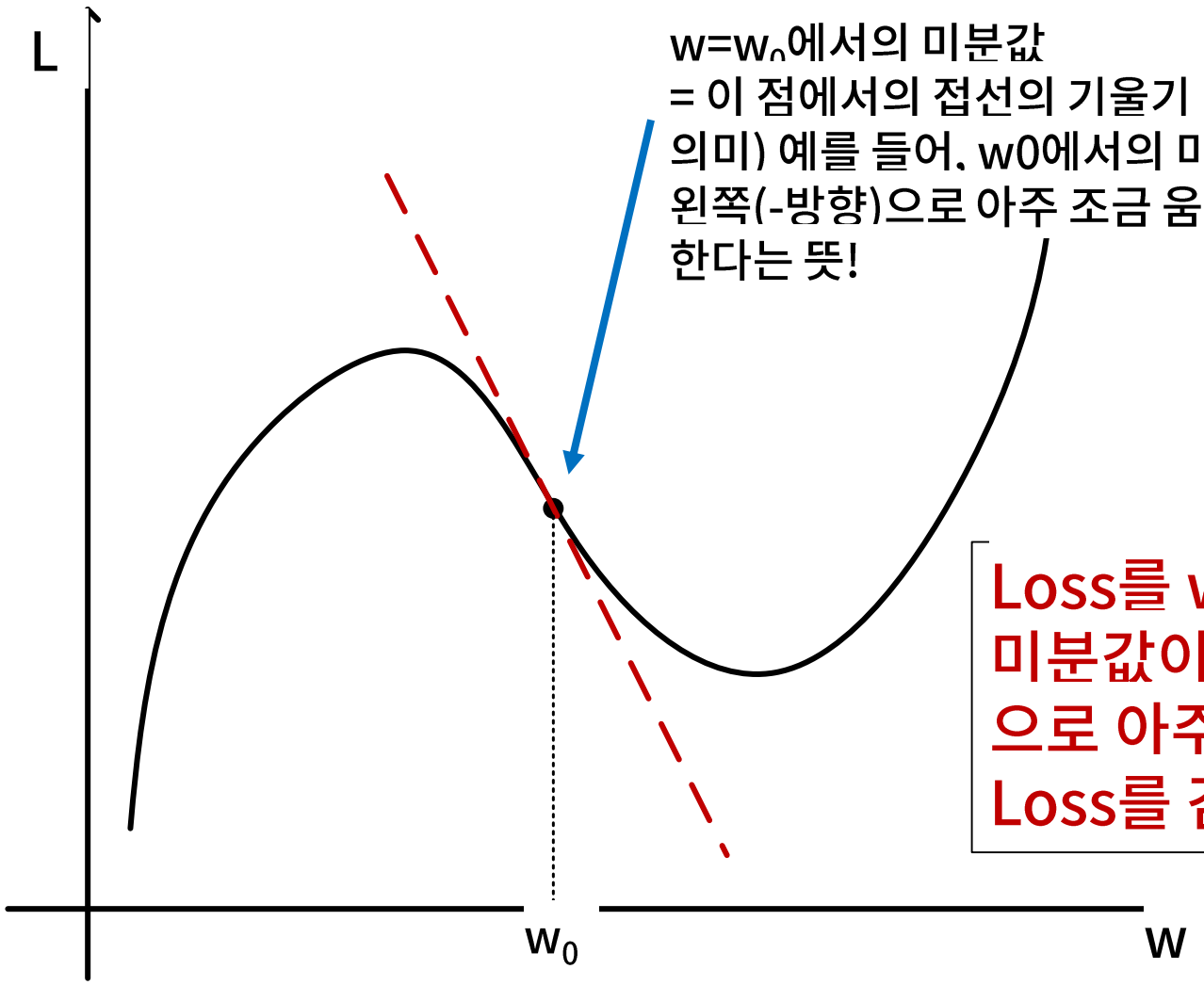


Loss Function(Cost Function)

- Neural Network이 얼마나 잘하는 지 or 얼마나 못하는 지에 대한 척도가 필요함
  - Loss Function / Cost Function
- 많이 쓰는 방법 중 한가지
  - (Neural Network의 출력과 실제 정답의 차이)^2
  - 예) 2가지 광고료 조합에 대하여 Neural Network이 예측한 판매량 : (100, 80), 실제 판매량 : (105, 78)일 때,

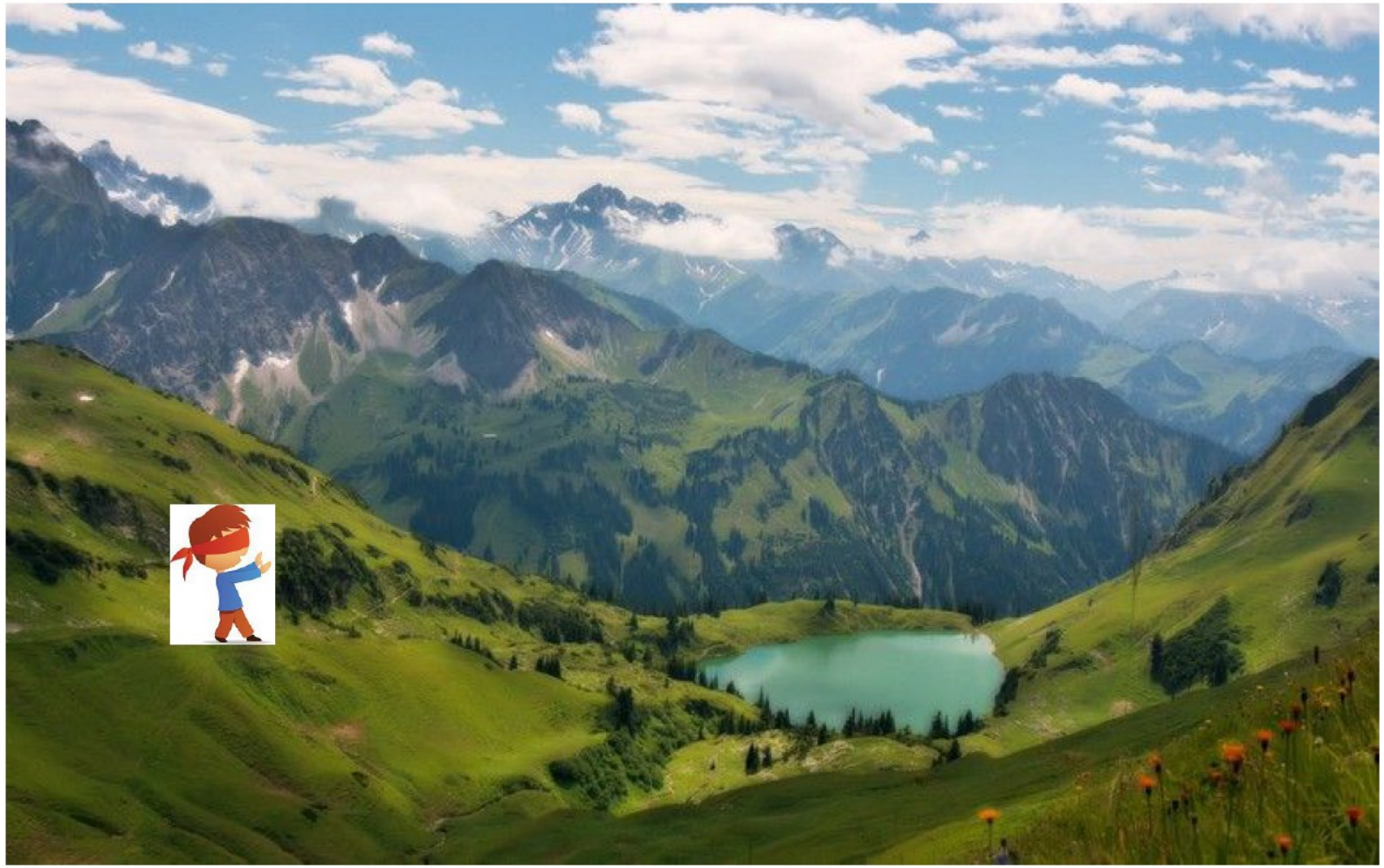
    $L = (105 - 100)2 + (78 - 80)2 = 29$
- 이제 필요한 것은 Loss Function의 값이 줄어들도록 weight 값들을 조금씩 바꾸는 것
  - Weight를 어떻게 바꿔야 Loss Function 값이 줄어들까? → 미분!

미분??



Loss를 w로 미분하고,
미분값이 가리키는 방향의 반대방향으로 아주 조금씩 w를 바꿔나가면 Loss를 감소시킬 수 있다!!

Gradient Descent



Gradient Descent

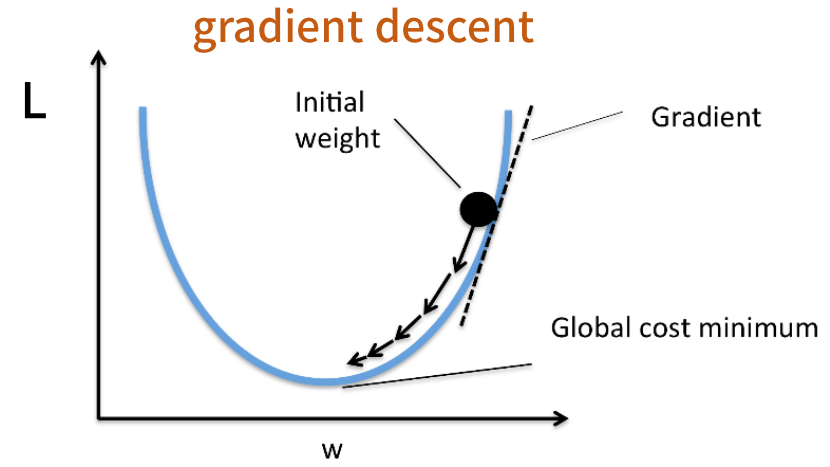
Loss Function의 미분(Gradient)를 이용하여 weight를 update하는 방법

$$w_{new} = w_{old} - \eta \nabla_w L$$

Weight update $= w_{new} - w_{old}$

$$= -\eta \nabla_w L$$

- $-$: Loss를 감소시키는 방향 (Descent)
- η : 아주 조금씩 이동 (Learning Rate)
- $\nabla_w L$: 미분값 (Gradient)



Gradient Descent

- 내가 가진 모든 training data에 대하여 Neural Network의 출력과 실제 정답(label)을 비교하여 각각의 Loss를 계산하고 이를 모두 더해서 전체 Loss를 계산한다
- 그리고, 이 loss를 weight로 미분한 다음 그 미분값(gradient)가 가리키는 방향의 반대방향으로 weight 값을 아주 조금씩 바꿔나간다

- 그런데 내가 가진 data가 아주아주 많은데 weight 값을 아주 조금 바꾸기 위해서 모든 data에 대해서 loss를 다 계산해야 한다…
- Data가 더 많아질수록 학습이 더 느려진다…

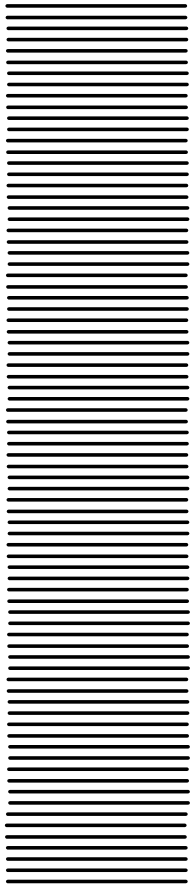
Batch / Stochastic / Mini-batch GD

모든 training data에 대하여 loss를 다 계산하는 건 너무 오래 걸리니까 전체 data 중에 일부 data만 뽑고(sampling) 이 data들이 전체 training data를 잘 대표한다고 가정해서 이 data들로만 loss를 계산하여 weight를 update해도 되지 않을까?

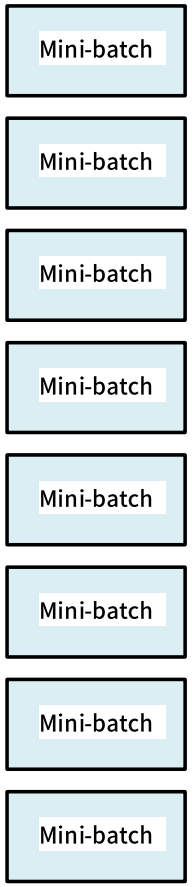
- Batch Gradient Descent
  - 모든 data에 대해서 loss를 계산하여 다 더하고 이를 이용해서 GD
- Stochastic Gradient Descent
  - Data를 1개만 뽑고, 그 1개의 data에 대한 loss를 이용하여 GD
- Mini-batch Gradient Descent
  - Batch/Stochastic의 중간 형태로 data를 n개를 뽑고 그 n개의 data에 대한 loss를 계산하여 다 더한 뒤 이를 이용하여 GD → 제일 많이 쓰는 방법!

Batch / Stochastic / Mini-batch GD

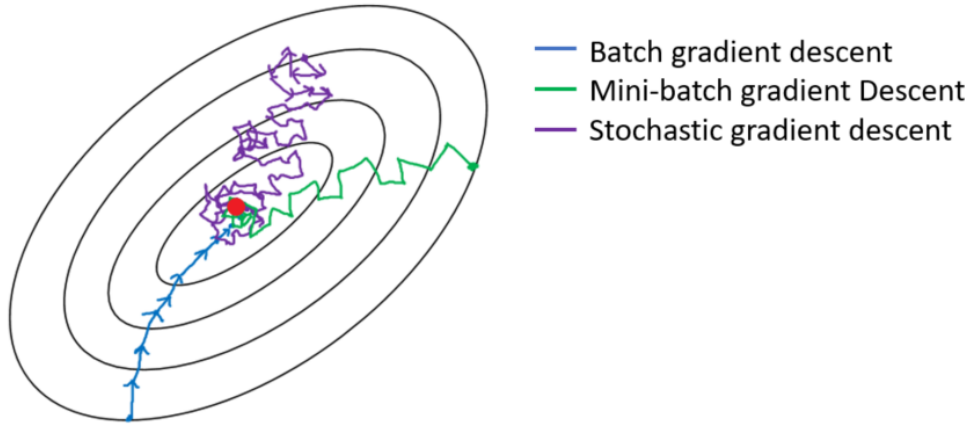
Batch
Gradient Descent



Stochastic
Gradient Descent



Mini-batch
Gradient Descent



Gradient Descent Again

Loss Function의 미분(Gradient)를 이용하여 weight를 update하는 방법

$$w_{new} = w_{old} - \eta \nabla_w L$$

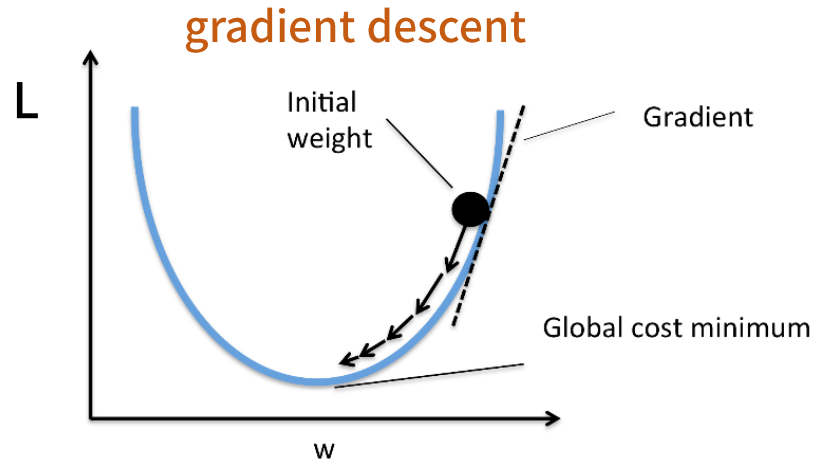
Weight update $= w_{new} - w_{old}$

$$= -\eta \nabla_w L$$

Loss를 감소 시키는 방향 (Descent)

아주 조금씩 이동 (Learning Rate)

미분값 (Gradient)



미분을 계산해봅시다!

$$z11 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + x4 \cdot w14$$

$$a11 = \sigma(z11) = \frac{1}{1 + e^{-z11}}$$

$$z2 = a11 \cdot w21 + a12 \cdot w22 + a13 \cdot w23$$

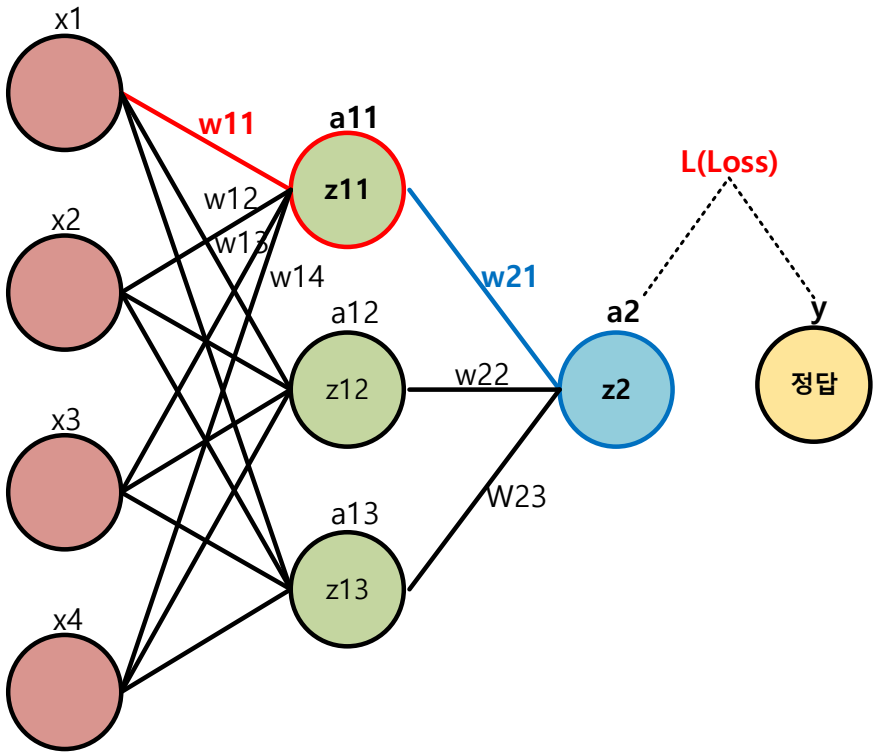
$$a2 = z2$$

$$L = (y - a2)^2$$

일 때,

w11을 update 하기 위해 필요한 미분값

$$\frac{\partial L}{\partial w11} = ??????$$



Back Propagation

$$z11 = x1 \cdot w11 + x2 \cdot w12 + x3 \cdot w13 + x4 \cdot w14$$
$$a11 = \sigma(z11) = \frac{1}{1 + e^{-z11}}$$
$$z2 = a11 \cdot w21 + a12 \cdot w22 + a13 \cdot w23$$
$$a2 = z2$$
$$L = (y - a2)^2$$

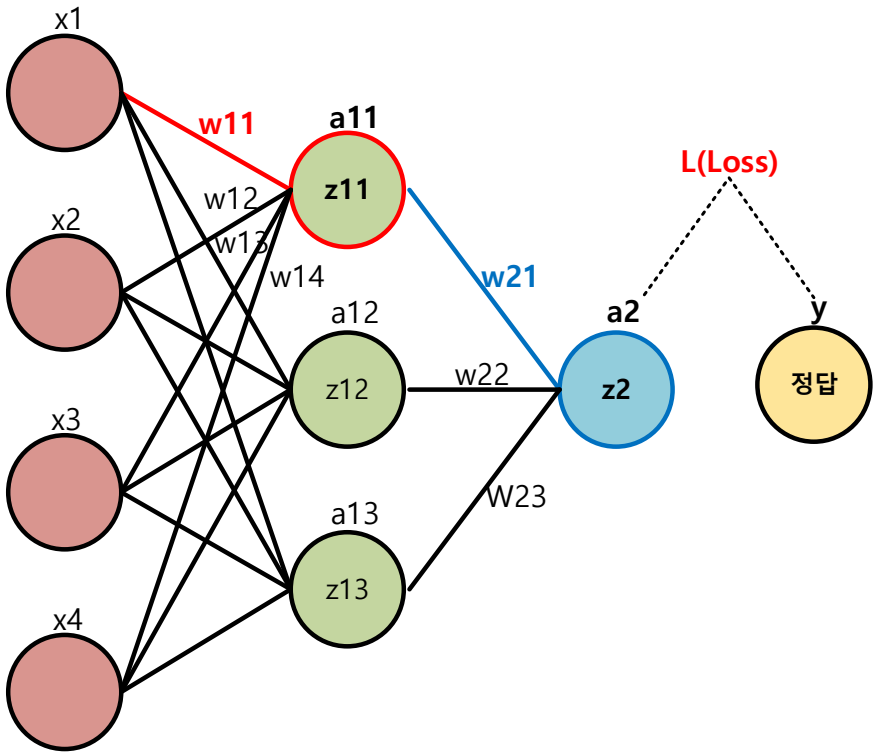
Loss부터 거꾸로 한 단계씩 미분을 해봅시다

$$\partial L/\partial a2 = -2(y - a2)$$
$$\partial a2/\partial z2 = 1$$
$$\partial z2/\partial a11 = w21$$
$$\partial a11/\partial z11 = \sigma(z11) \cdot \left(1 - \sigma(z11)\right)$$
$$\partial z11/\partial w11 = x1$$

이 미분들을 전부 각각 곱하면(chain rule),

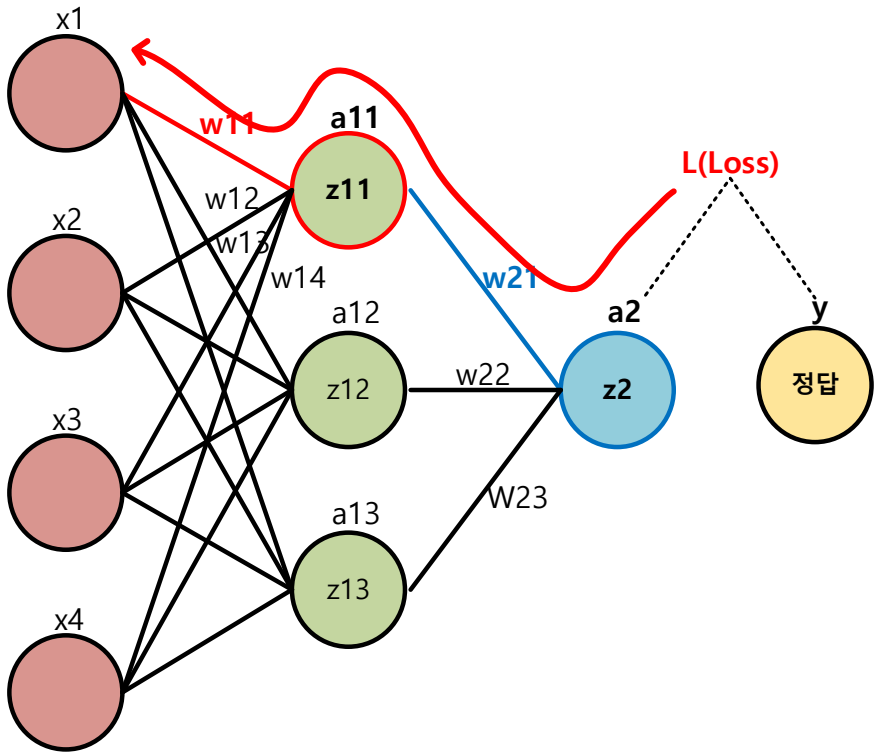
$$\frac{\partial L}{\partial a2} \cdot \frac{\partial a2}{\partial z2} \cdot \frac{\partial z2}{\partial a11} \cdot \frac{\partial a11}{\partial z11} \cdot \frac{\partial z11}{\partial w11} = \frac{\partial L}{\partial w11}$$

우리가 구하려고 했던 미분값



Back Propagation

Loss로부터 거꾸로 한 단계씩 미분 값을 구하고 이 값들을 chain rule에 의하여 곱해가면서 weight에 대한 gradient를 구하는 방법

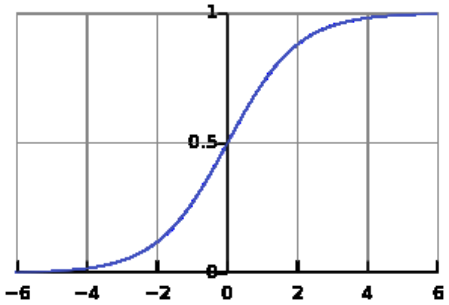


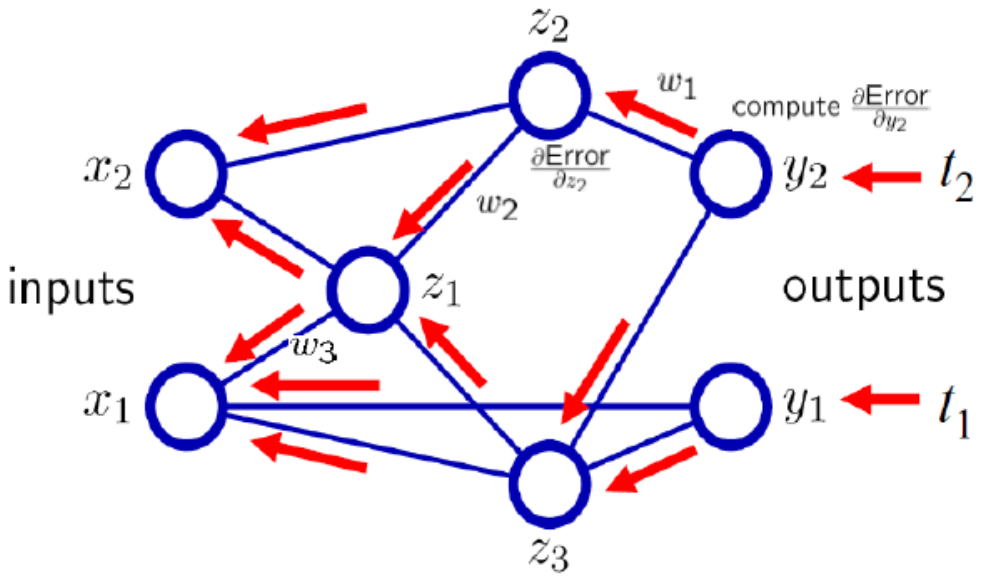
딥러닝을 어렵게 하는 것들

- Vanishing gradient problem
- Overfitting problem
- Get stuck in local minima

Vanishing Gradient Problem

- Gradient 값이 뒤로 전달될 수록 점점 작아짐
- Sigmoid 사용으로 인하여(미분값의 최대 : ¼) 아래쪽 layer 는 학습이 이루어지지 않음





$$\frac{\partial \text{Error}}{\partial w_1} = \frac{\partial \text{Error}}{\partial y_2}\frac{\partial y_2}{\partial w_1} = \frac{\partial \text{Error}}{\partial y_2}\sigma'(\cdot)z_2$$

$$\frac{\partial y_2}{\partial w_1} = \sigma'(\cdot)z_2$$

$$\frac{\partial \text{Error}}{\partial w_2} = \frac{\partial \text{Error}}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \frac{\partial \text{Error}}{\partial y_2}\sigma'(\cdot)w_1\sigma'(\cdot)z_1$$

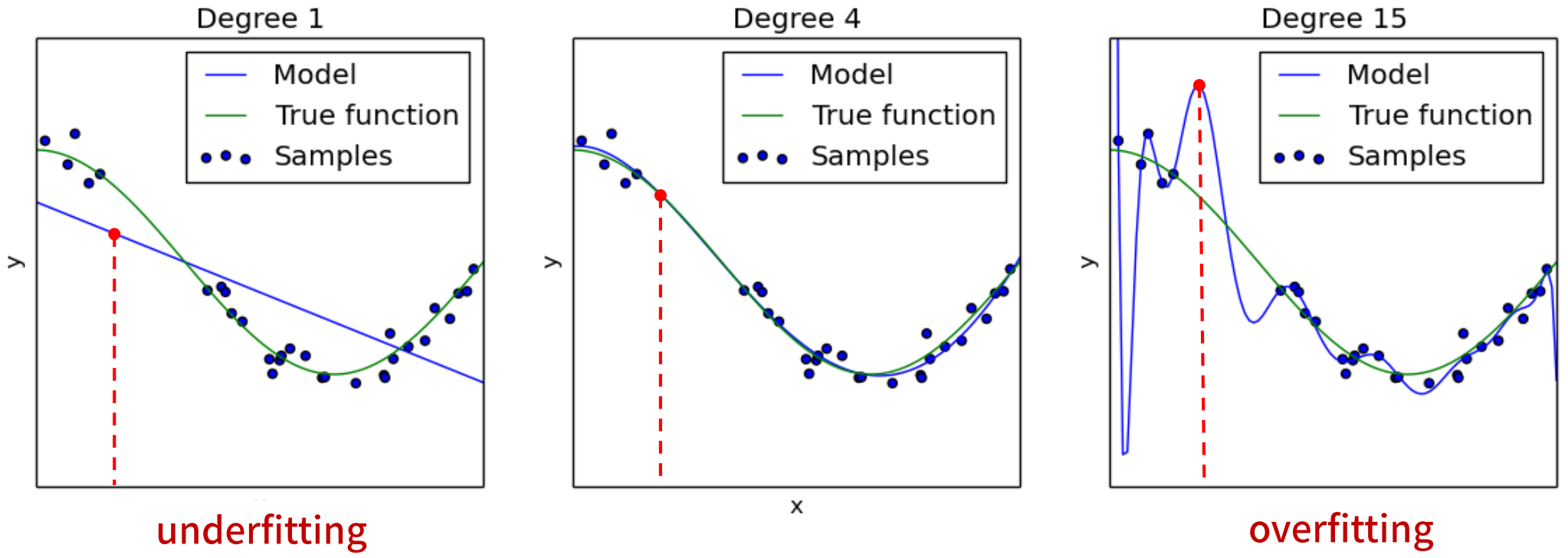
$$\frac{\partial z_2}{\partial w_2} = \sigma'(\cdot)z_1$$

$$\frac{\partial \text{Error}}{\partial w_3} = \sigma'(\cdot)\sigma'(\cdot)\sigma'(\cdot)C$$

$$\sigma'(\cdot) \approx 0$$

Overfitting Problem

- Data가 많지 않은 경우에 발생할 수 있음
- Overfitting : Training data에 너무 최적화(fitting)를 함으로 인해서 일반화(generalization) 성능이 떨어지는 현상



Overfitting Problem

열심히 Neural Network에게 고양이  가르쳤더니…

뚱뚱하니까 고양이 아님

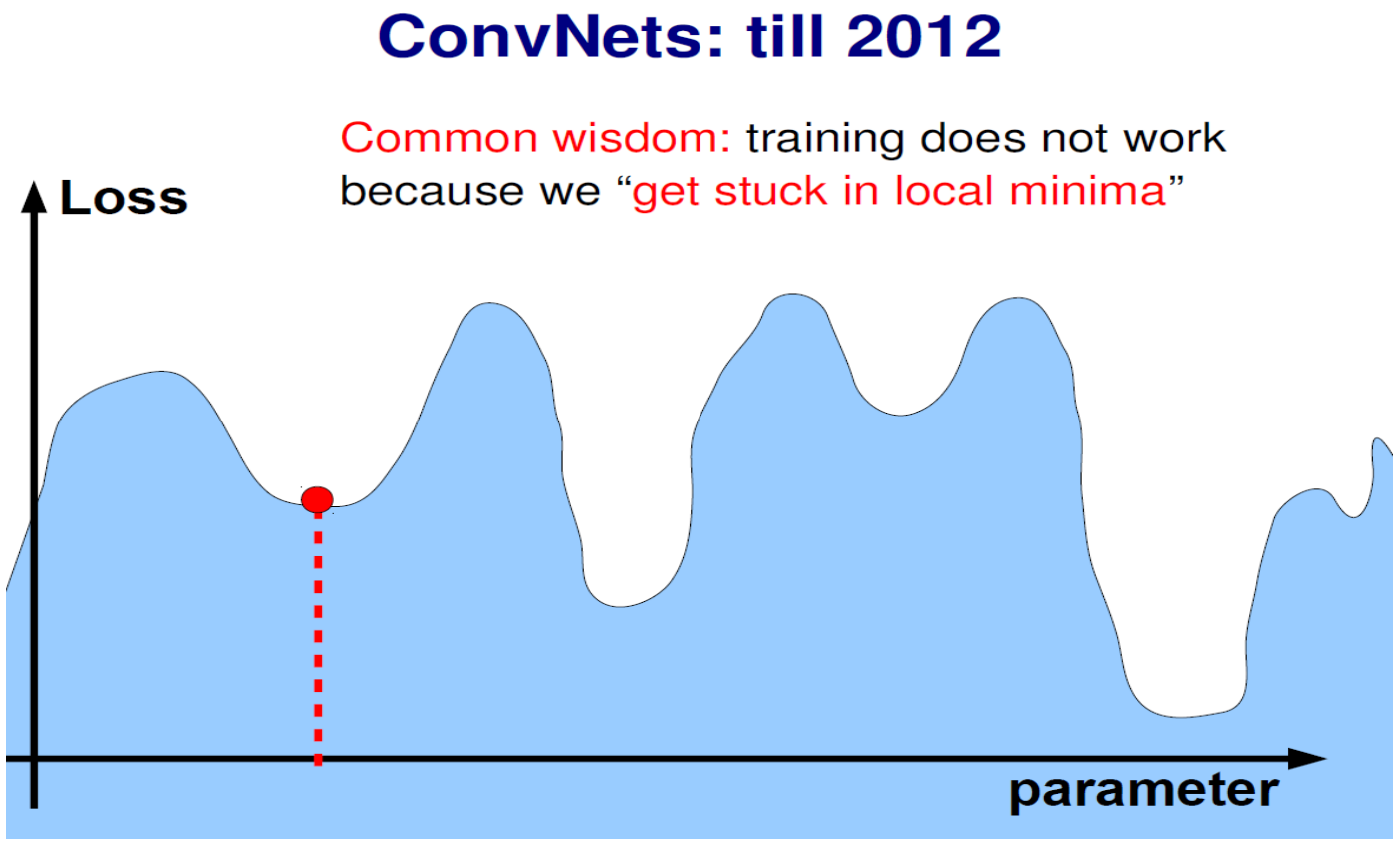
갈색이니까 고양이 아님

귀 쳐졌으니까 고양이 아님

융통성이 하나도 없어짐 → Overfitting!!

Local Minima

- 어디서 시작하느냐에 따라서 잘못하면 local minima에 빠질 위험이 존재



Solutions

- Vanishing gradient problem

→ Sigmoid 말고 ReLU를 쓰자

- Overfitting problem

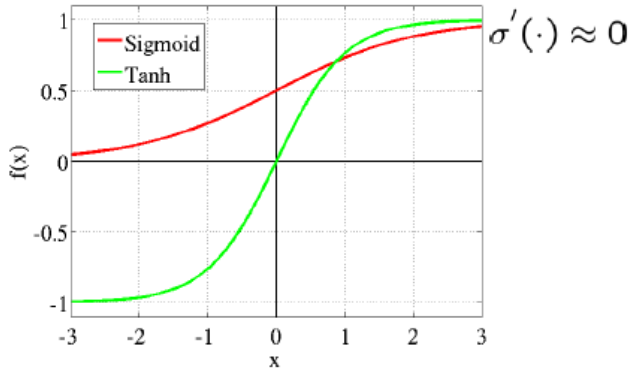
→ Regularization method를 쓰자(예 : dropout)

- Get stuck in local minima

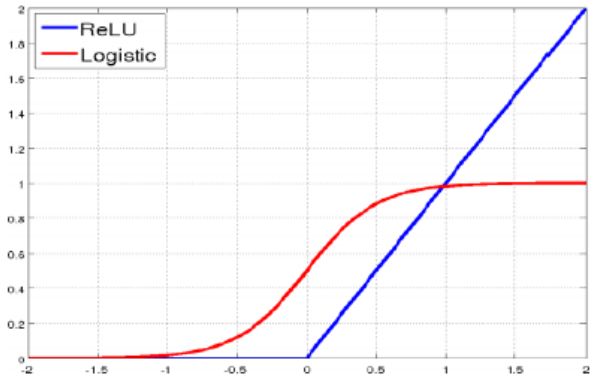
→ Local minima에 빠져도 괜찮다

ReLU : Rectified Linear Unit

- ReLU를 activation function으로 사용 → sparse activation
- ReLU는 미분값이 0 아니면 1 → vanishing gradient 해결

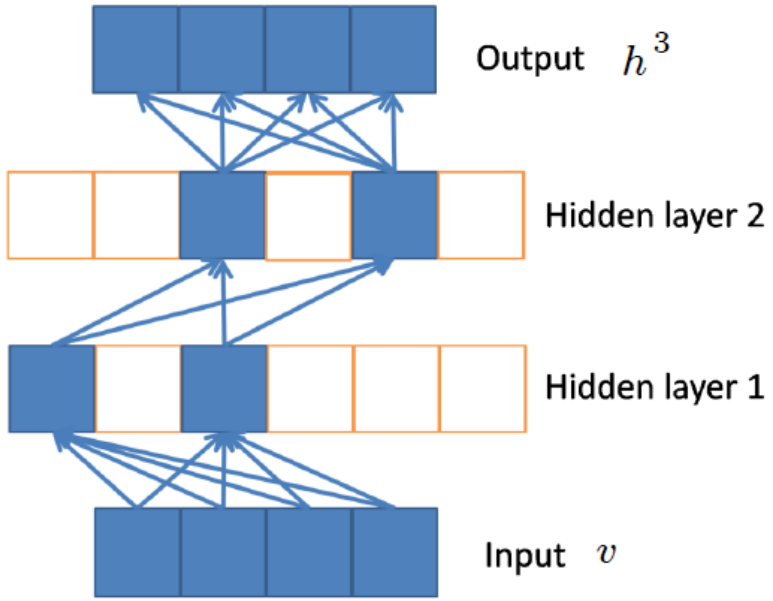


$\sigma'(\cdot) \approx 0$



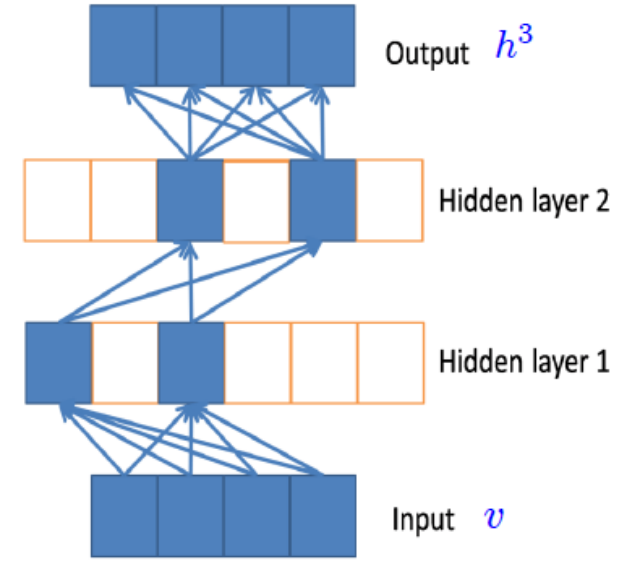
$$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

slope: 1



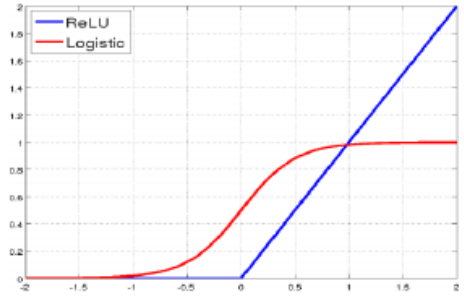
ReLU의 의미

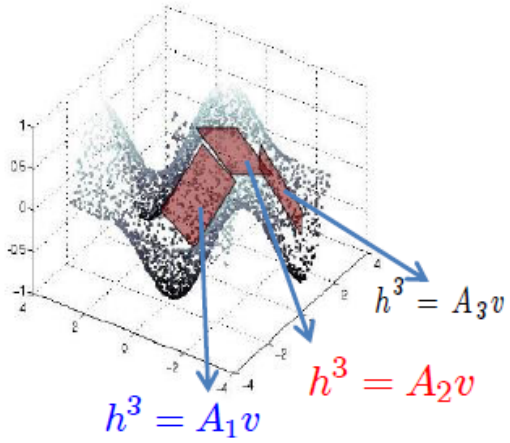
- Piece-wise linear tiling : locally linear mapping

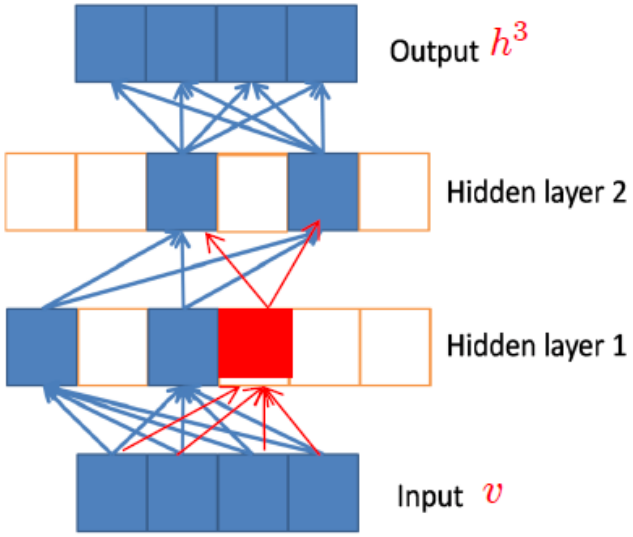


$$h^3 = \begin{bmatrix} \ddots & & \\ & \ddots & \\ & & \ddots \end{bmatrix} \underbrace{\begin{bmatrix} \ddots & \\ & \ddots \end{bmatrix}}_{2\times 2} \underbrace{\begin{bmatrix} \cdots & \cdots \end{bmatrix}}_{2\times 4} v = A_1 v$$

4×2



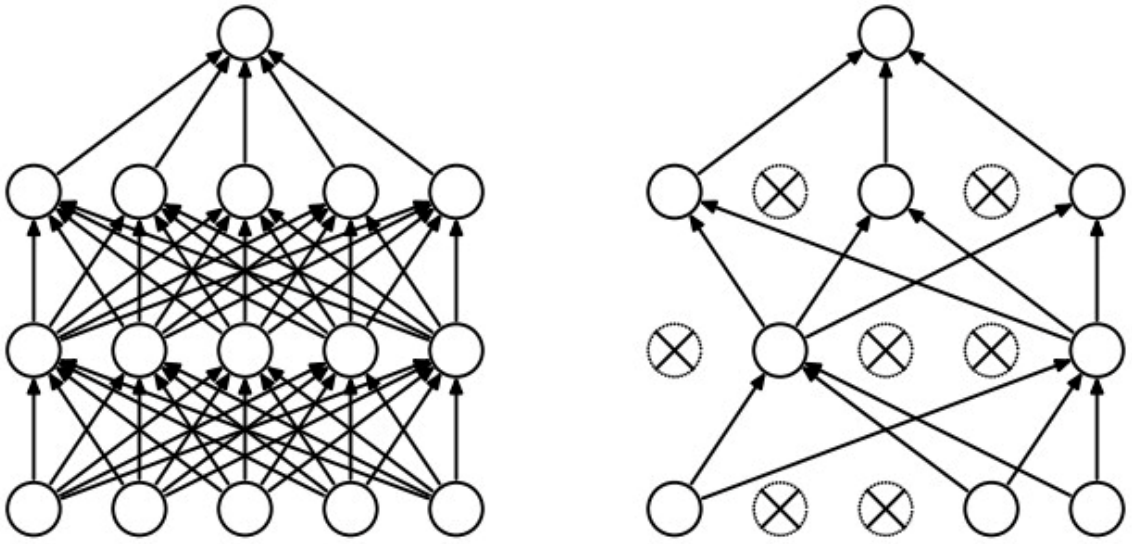




$$h^3 = \underbrace{\begin{bmatrix} \ddots & & \\ & \ddots & \\ & & \ddots \end{bmatrix}}_{4\times 2} \underbrace{\begin{bmatrix} \ddots & \\ & \ddots \end{bmatrix}}_{2\times 3} \underbrace{\begin{bmatrix} \cdots & \cdots \end{bmatrix}}_{3\times 4} v = A_2 v$$

Dropout(Regularization Method)

- 각 학습 단계마다, 특정 확률로(예 : 50%) random하게 hidden layer에 있는 unit들을 없애고 학습하는 방법
- Ensemble 개념을 적용
  - 여러 개의 model을 사용하여 평균값을 쓰면 하나의 model을 쓰는 경우보다 좋음
  - 하나의 model로 비슷한 효과를 낼 수 있는 방법



(a) Standard Neural Net

(b) After applying dropout.

Other Regularization Methods

- Weight Decay(L2 Regularization)

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2$$

- Batch Normalization
  - Benefits of BN
    - Increase learning rate
    - Remove dropout
    - Reduce L2 weight decay
    - Remove LRN

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

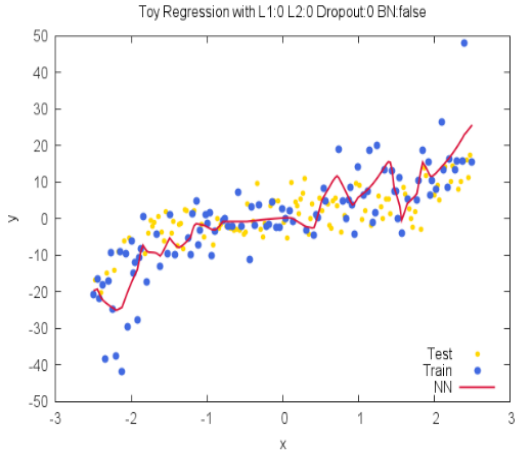
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

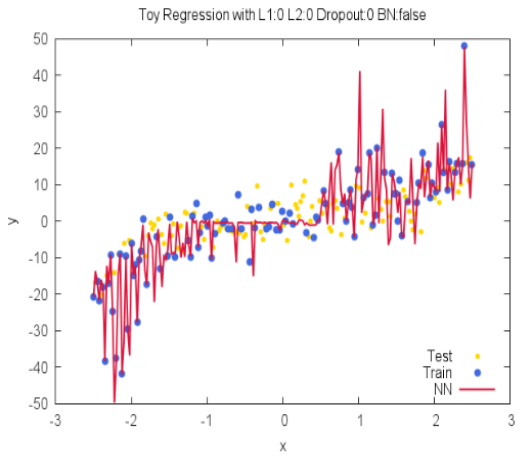
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

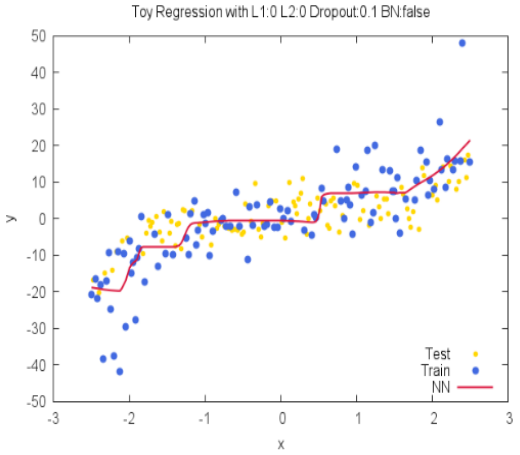
Regularization Methods

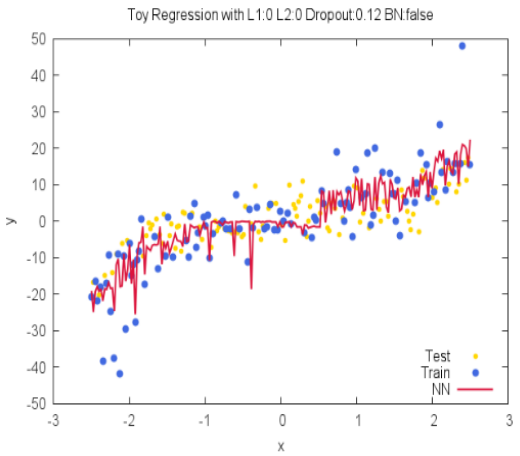
No Regularization



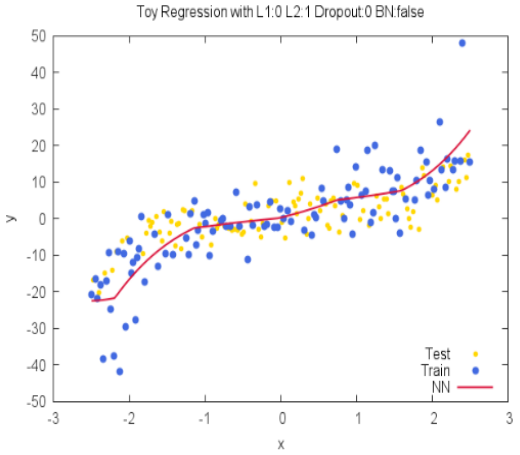


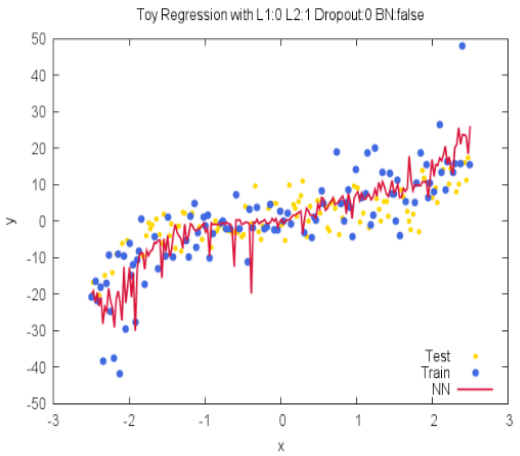
Dropout



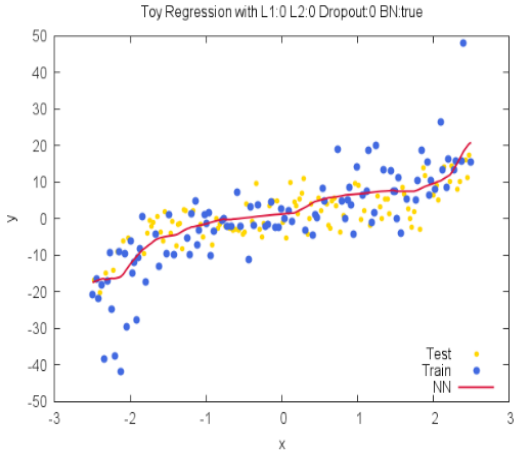


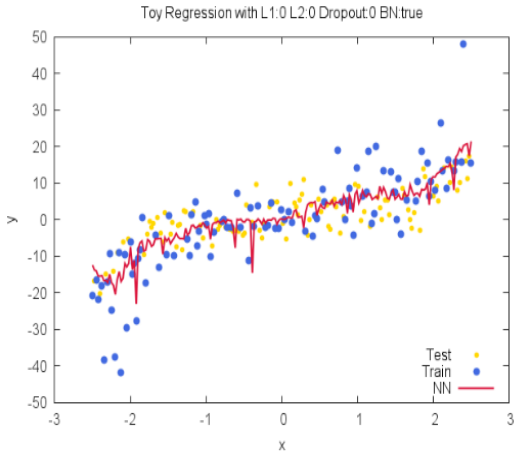
L2 Regularization





Batch Normalization

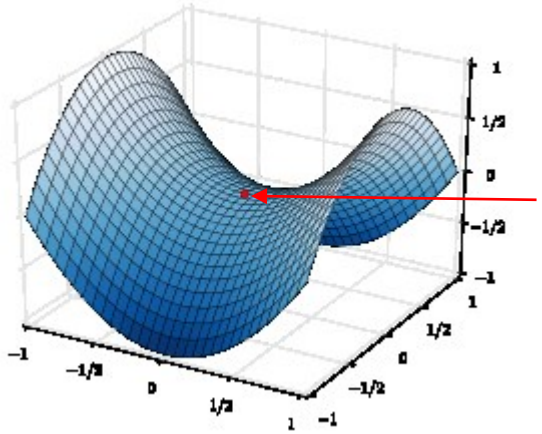


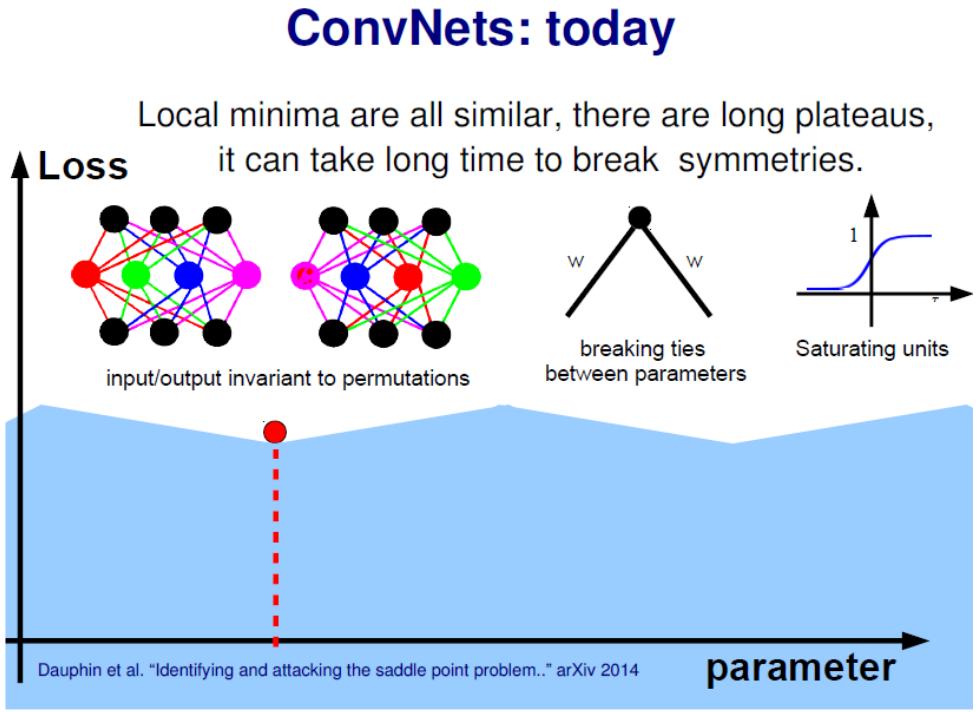


Local Minima

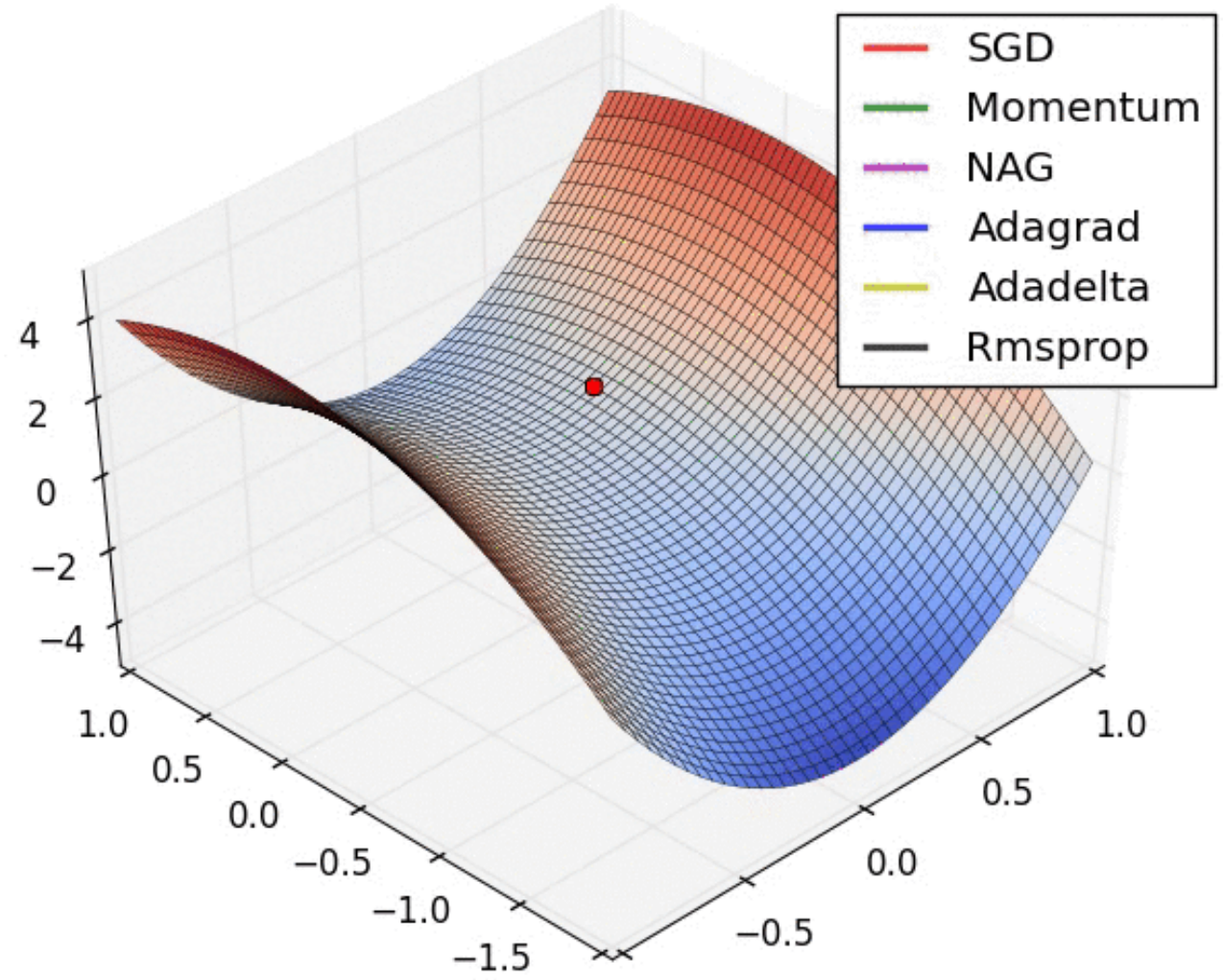
Local Minima에 대하여

- minimum이라고 하는 것은 현재 차원에서 이동할 수 있는 모든 방향으로의 gradient 값이 증가하는 방향이어야 하는데 이런 경우는 확률적으로 희박함
- DNN과 같은 고차원 구조에서는 대부분은 local minima가 아니라 saddle point일 가능성이 높음
- 만약 실제 local minima가 존재한다면 그것은 global minimum과 거의 차이가 없을 가능성이 높음(neural network의 대칭성)





Optimization Methods

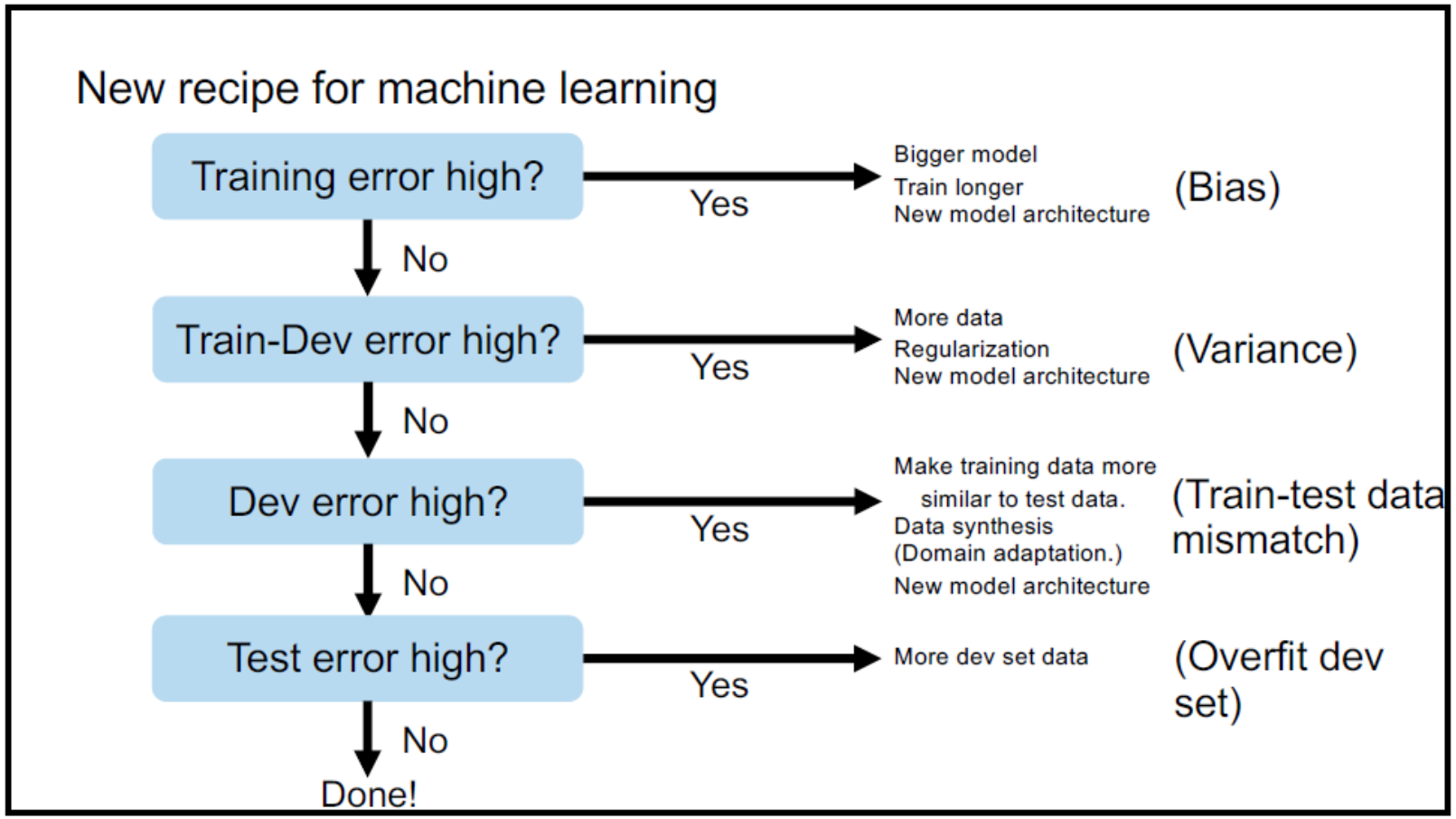


그 밖에 Deep Learning을 가능하게 했던 것들

- Hardware
  - CPUs, GPUs!, ASICs https://youtu.be/-P28LKWTzrI
- Organized Large Datasets
  - ImageNet
- Algorithms and Research
  - Backprop, CNN, LSTM
- Software and Infrastructure
  - Git, AWS, Amazon Mechanical Turk, TensorFlow, …
- Financial Backing of large Companies
  - Google, Facebook, Amazon, …

Data

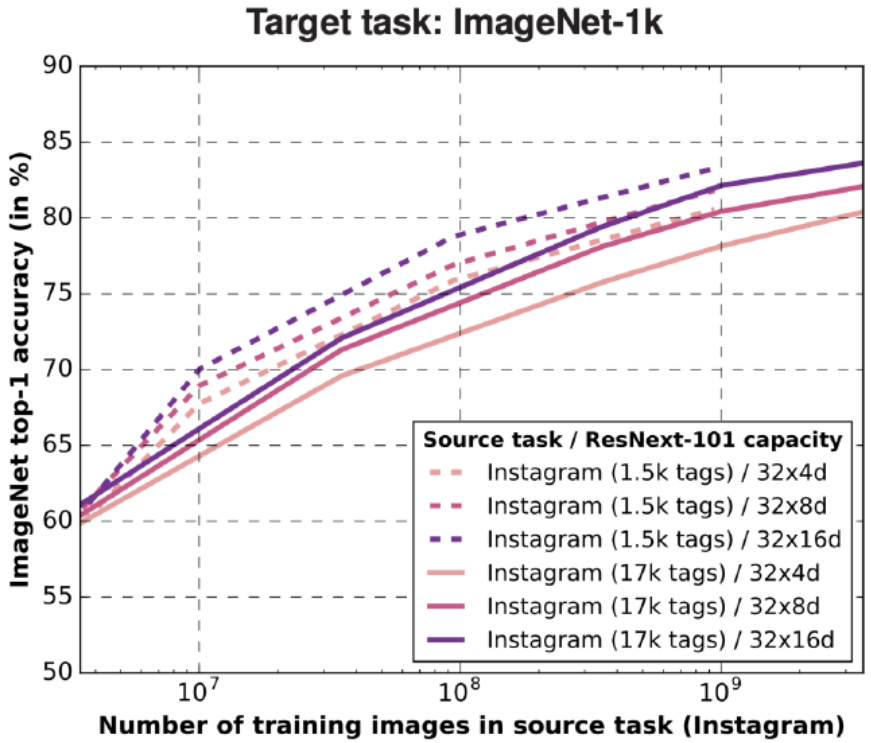
- Data, more data (by Andrew Ng's tutorial @NIPS 2016)



Data is Important!

- It is not questionable that

  more data still improves network performance

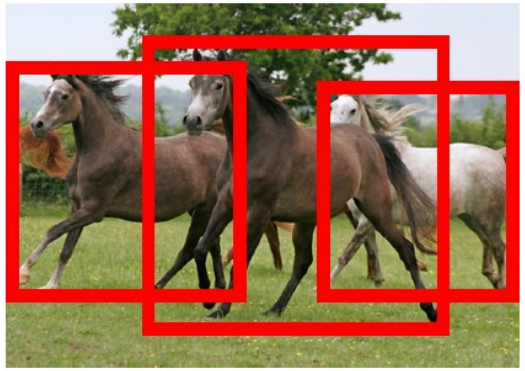


[Mahajan et al., ECCV'18]

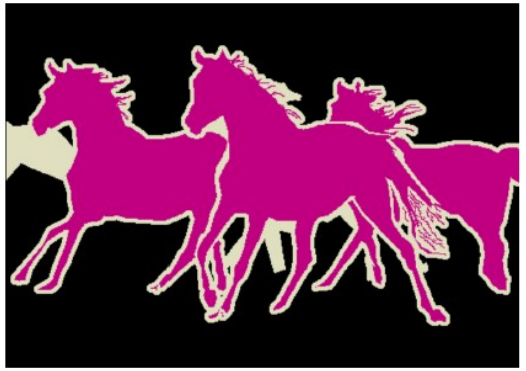
← 천만~10억장

Data is Expensive!

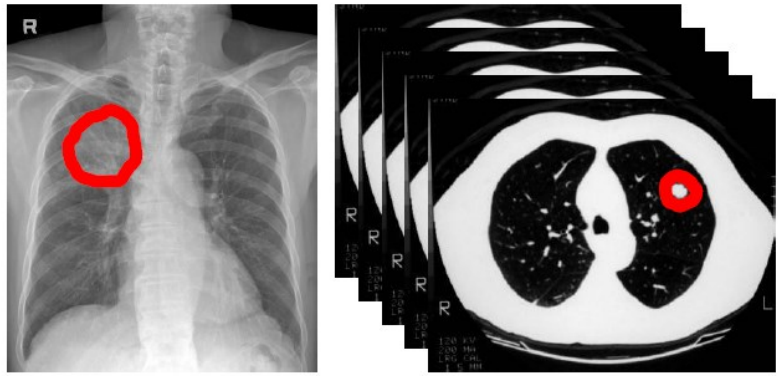
$



$$



$$$



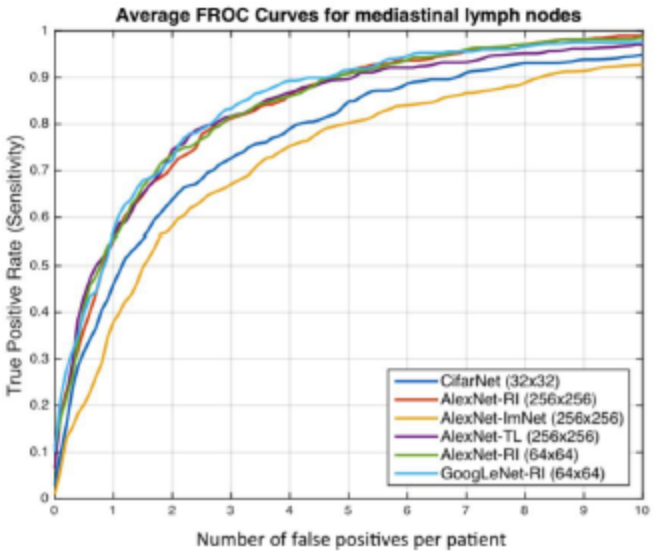
$$$$$

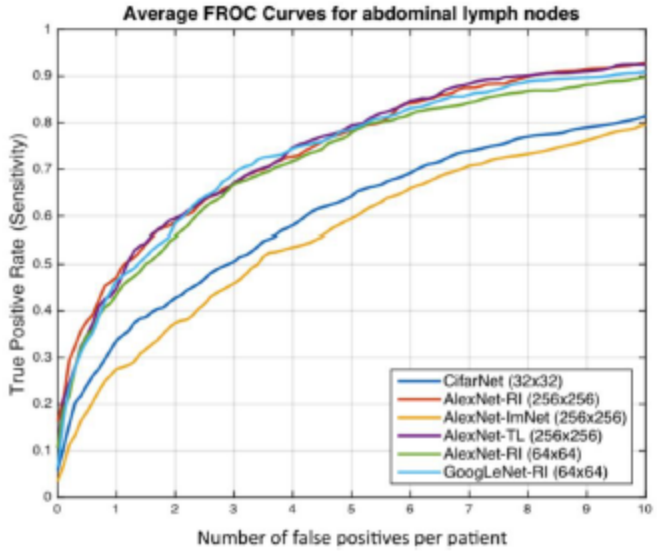
Slide from Donggen Yoo(Lunit)@CVPR 2019

If We Don't Have Enough Data

- Use pre-trained model!
- Transfer learning from other domains
  - Transferred network with 'deep' fine-tuning shows good results

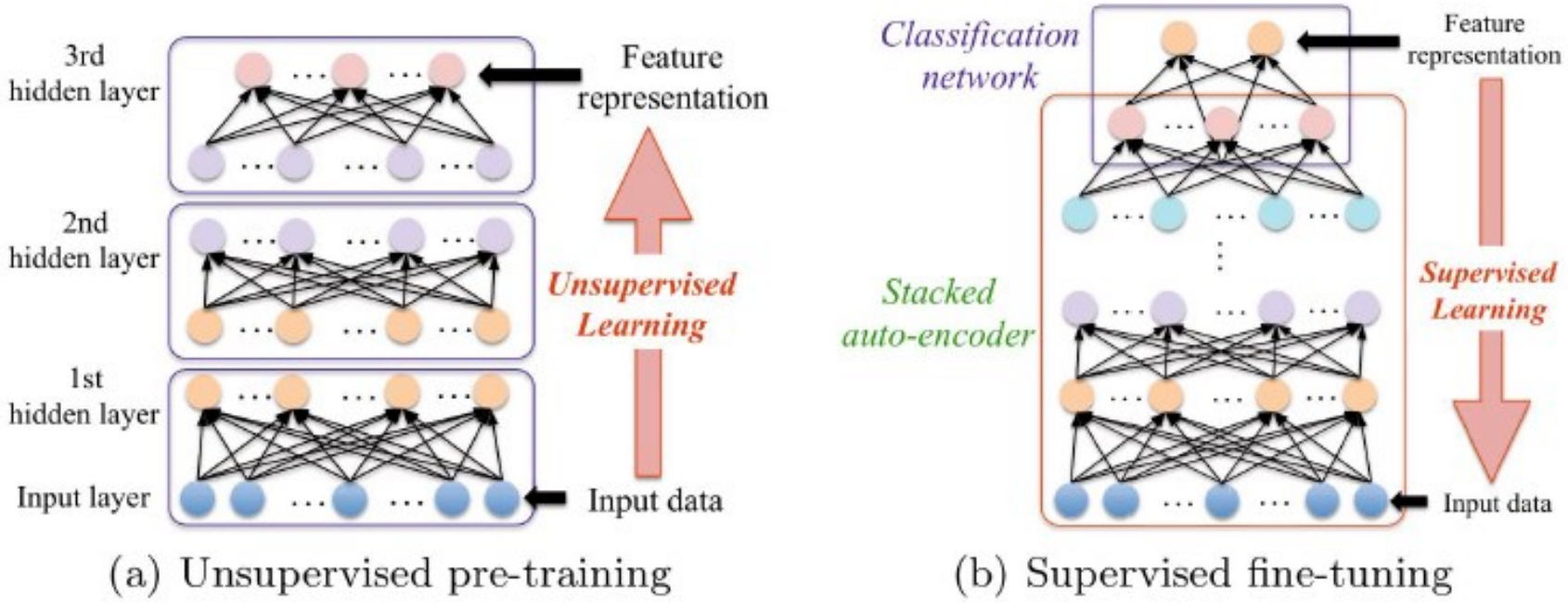
Region	Mediastinum		Abdomen	
Method	AUC	TPR/3FP	AUC	TPR/3FP
[41]	-	0.63	-	0.70
[22]	**0.92**	0.70	**0.94**	**0.83**
[36]	-	**0.78**	-	0.78
CifarNet	0.91	0.70	0.81	0.44
AlexNet-ImNet	0.89	0.63	0.80	0.41
AlexNet-RI-H	0.94	0.79	0.92	0.67
AlexNet-TL-H	0.94	0.81	0.92	0.69
GoogLeNet-RI-H	0.85	0.61	0.80	0.48
GoogLeNet-TL-H	0.94	0.81	**0.92**	**0.70**



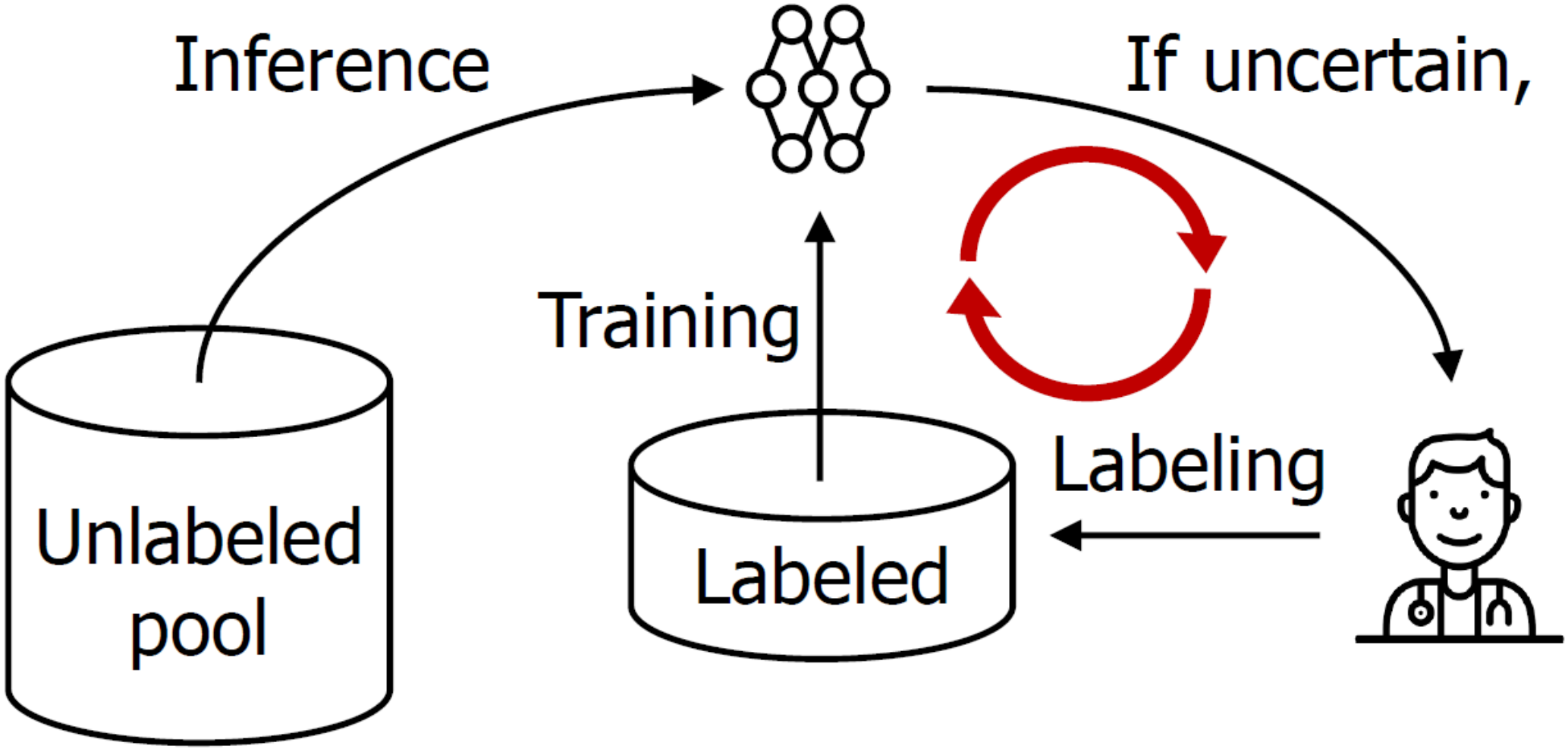


If We Don't Have Enough Data

- Unsupervised pre-training and supervised Fine-tuning
  - Unsupervised training of unlabeled data(using auto-encoder) before supervised learning can make good results



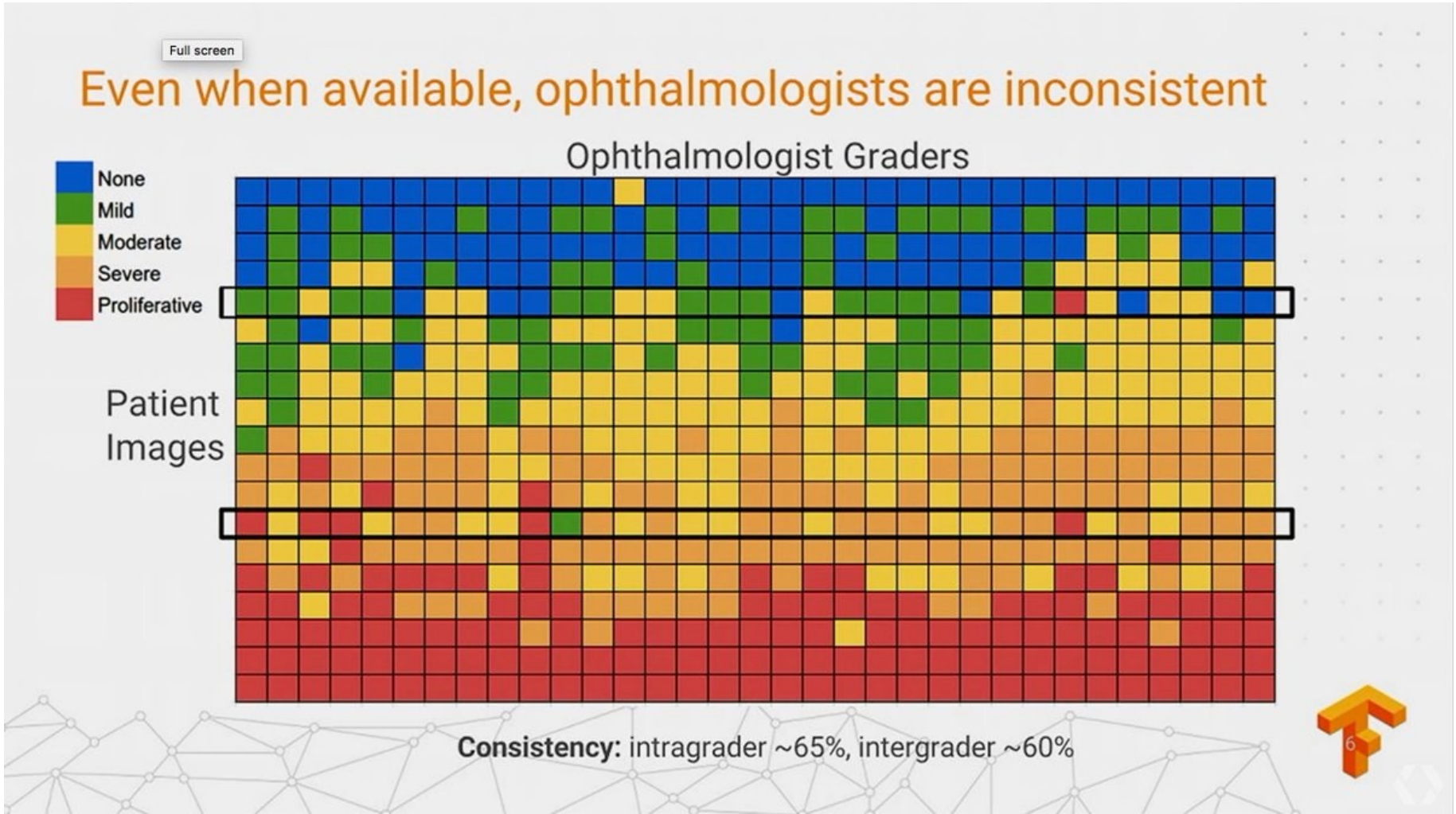
Active Learning



Slide from Donggen Yoo(Lunit)@CVPR 2019

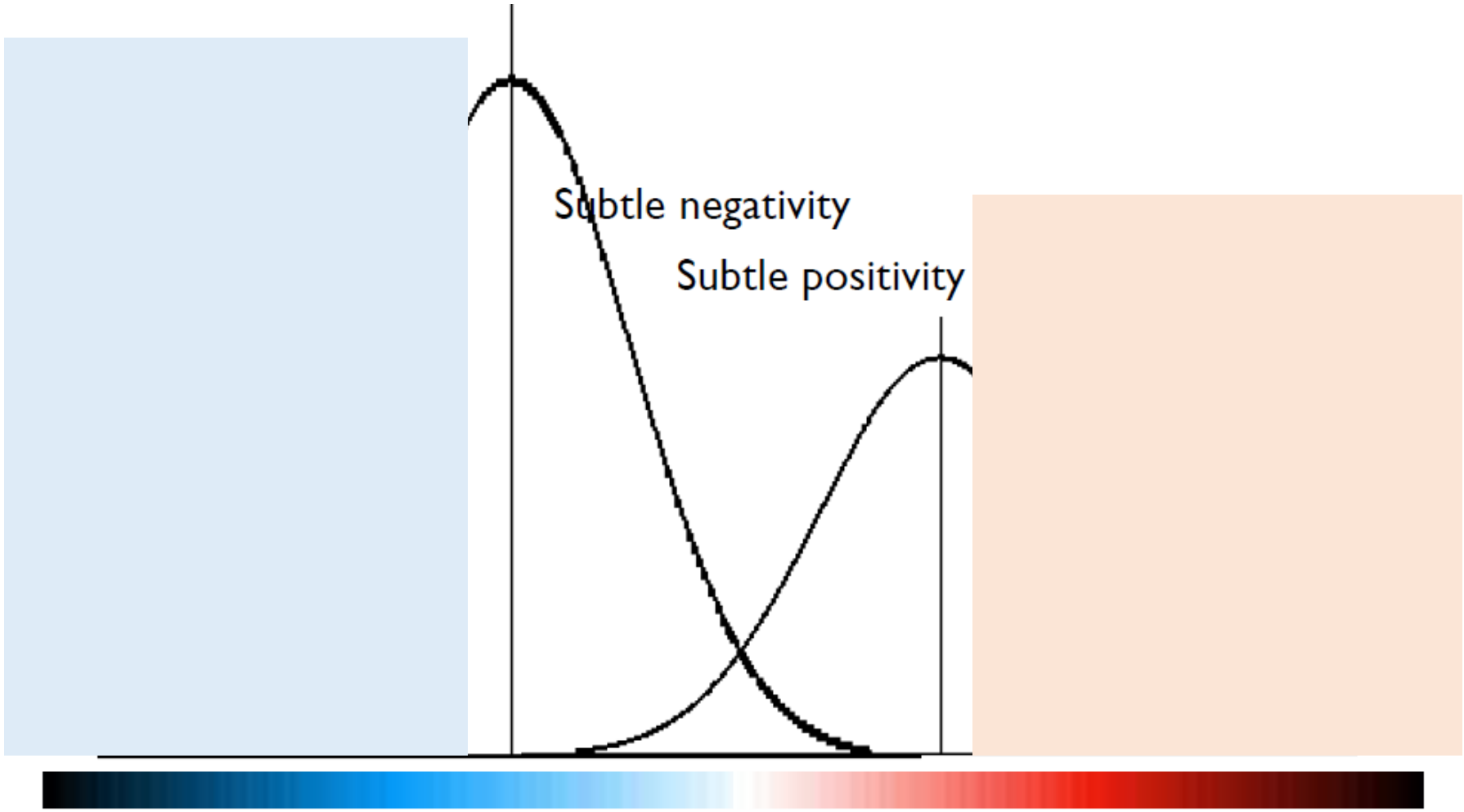
Good Data vs Bad Data

- Our labels are perfect?



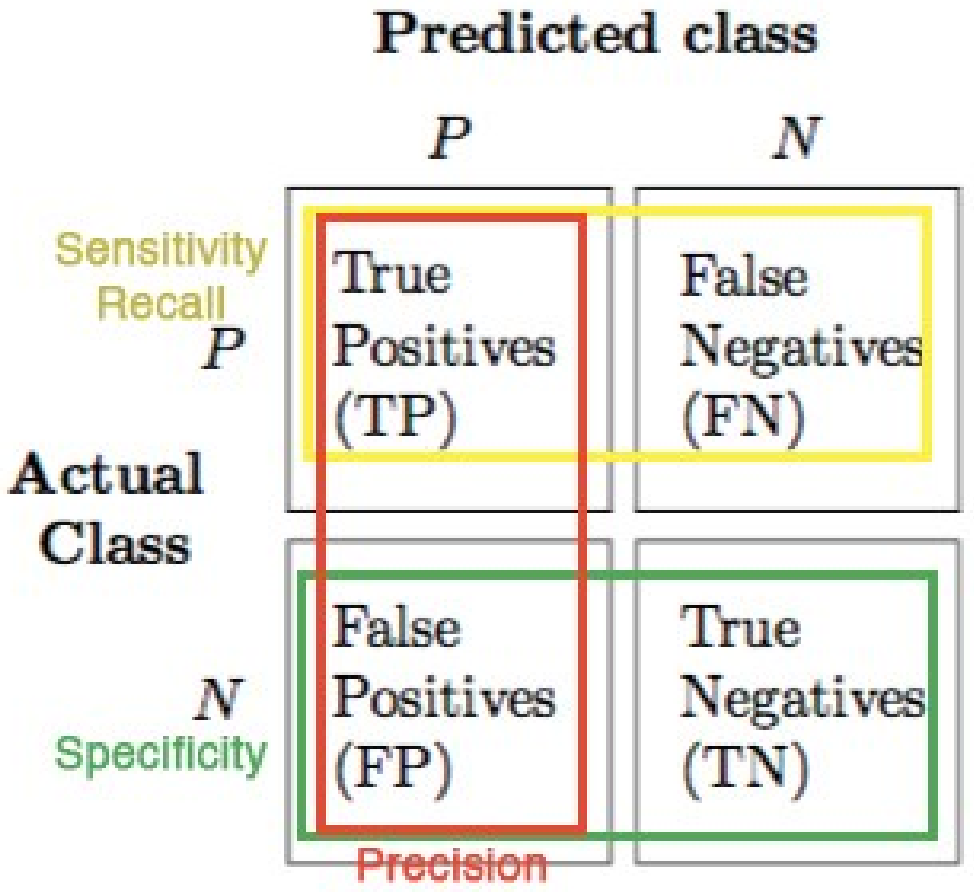
Good Data vs Bad Data

- Our data is unbiased?



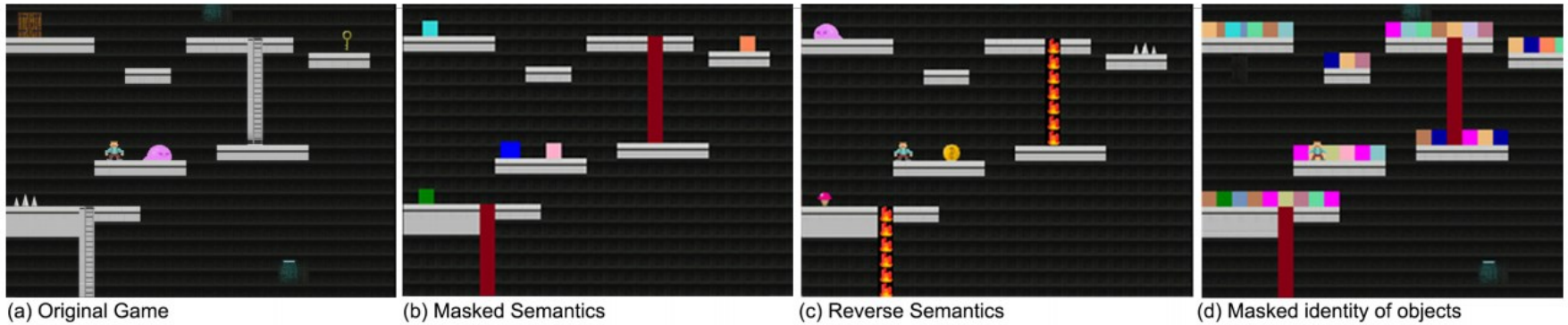
Is Our Model Good Enough?

- Sometimes accuracy is not a good measurement

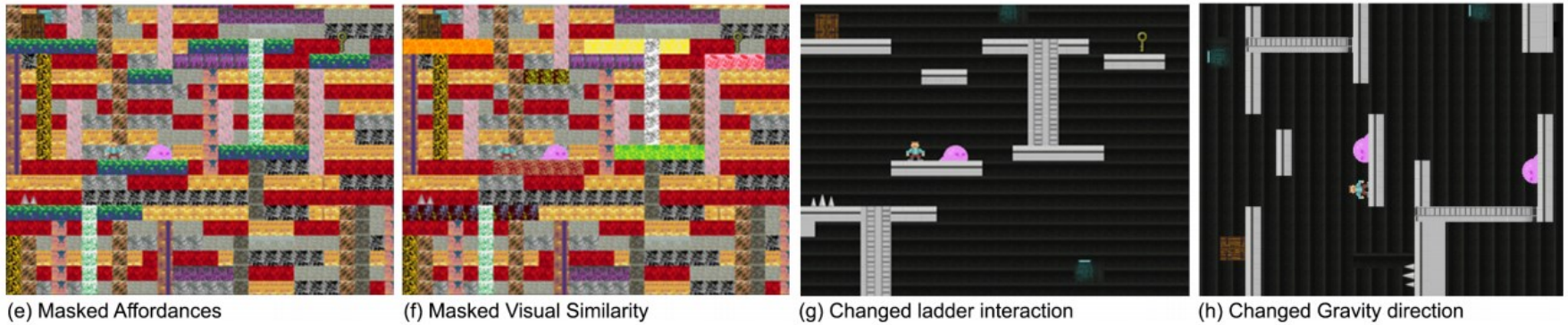


Deep Learning's Challenges – Toward the Real World

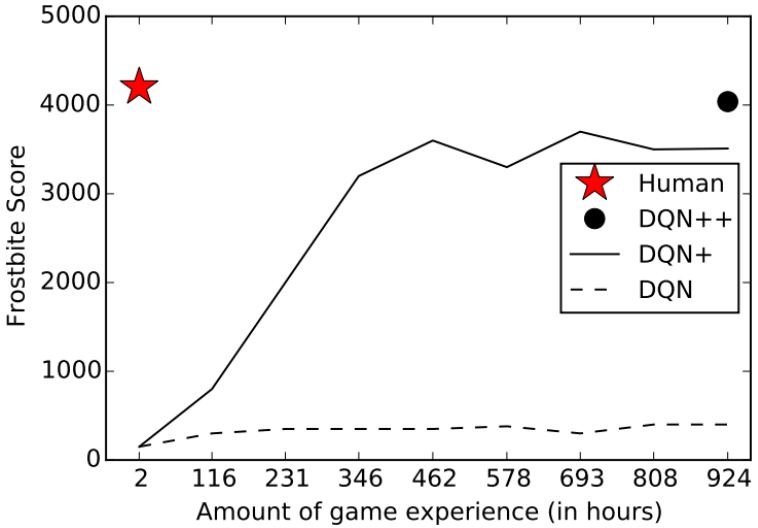
Deep Learning's Challenges – Efficient Training

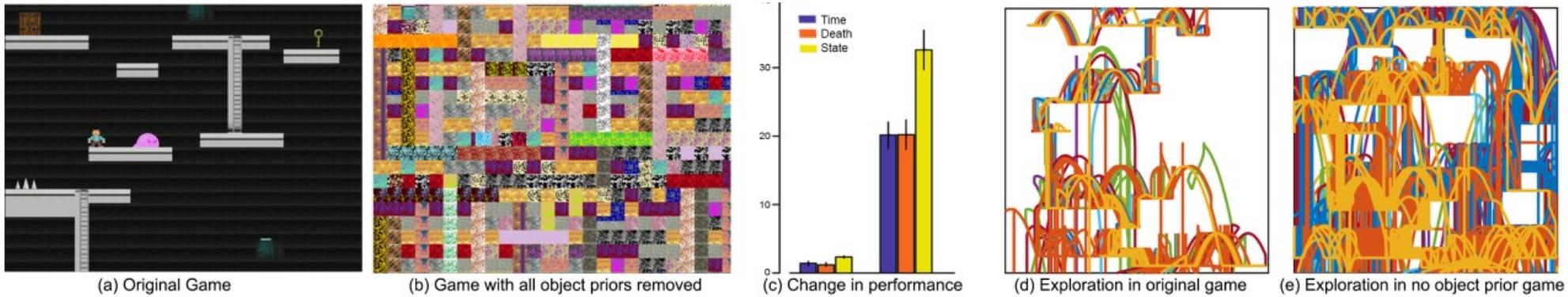


(a) Original Game (b) Masked Semantics (c) Reverse Semantics (d) Masked identity of objects



(e) Masked Affordances (f) Masked Visual Similarity (g) Changed ladder interaction (h) Changed Gravity direction





(a) Original Game (b) Game with all object priors removed (c) Change in performance (d) Exploration in original game (e) Exploration in no object prior game

Deep Learning's Challenges – Unintended Consequences

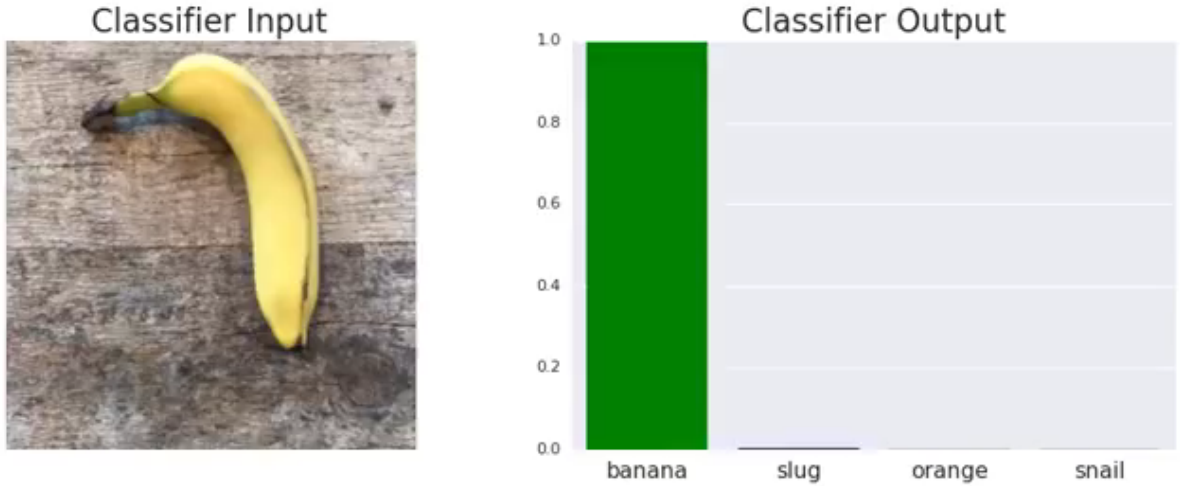
Human Play

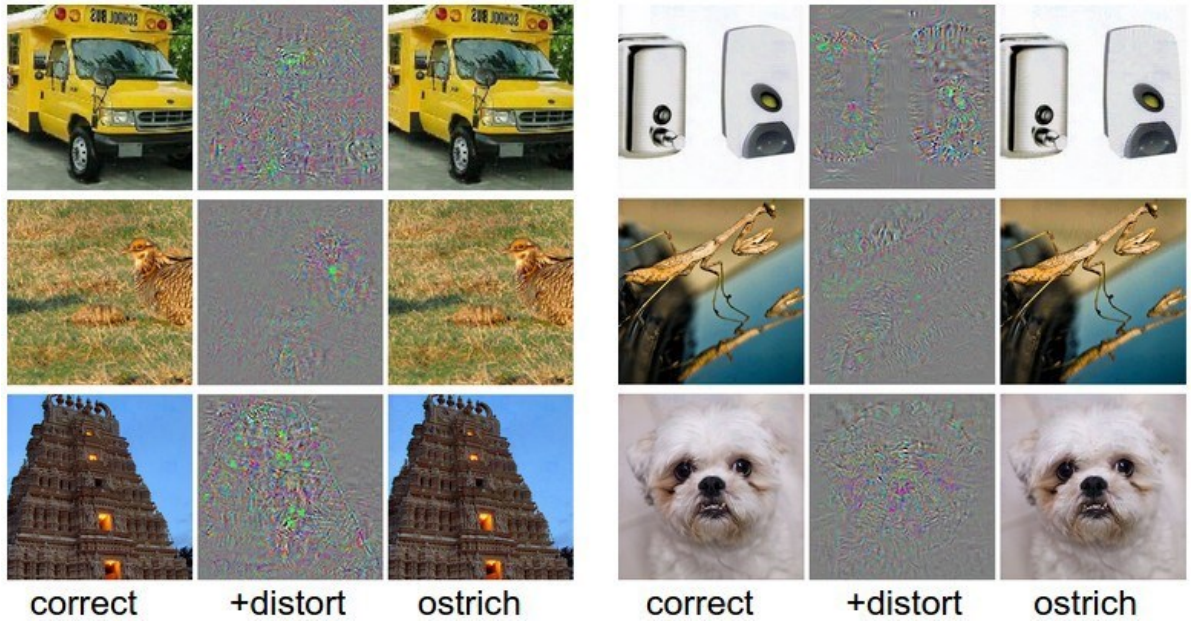
AI Play

Player gets reward based on:

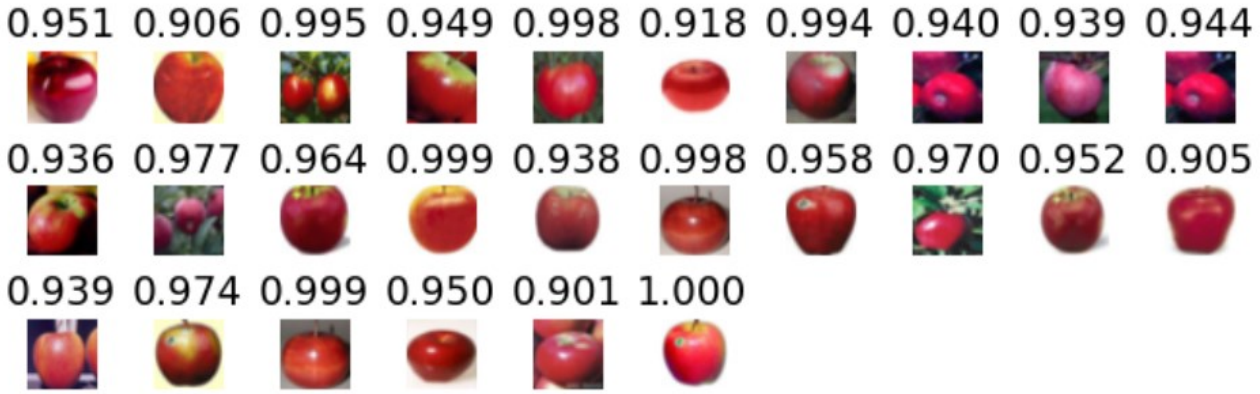
1. Finishing time
2. Finishing position
3. Picking up "turbos"

Deep Learning's Challenges – Adversarial Attack

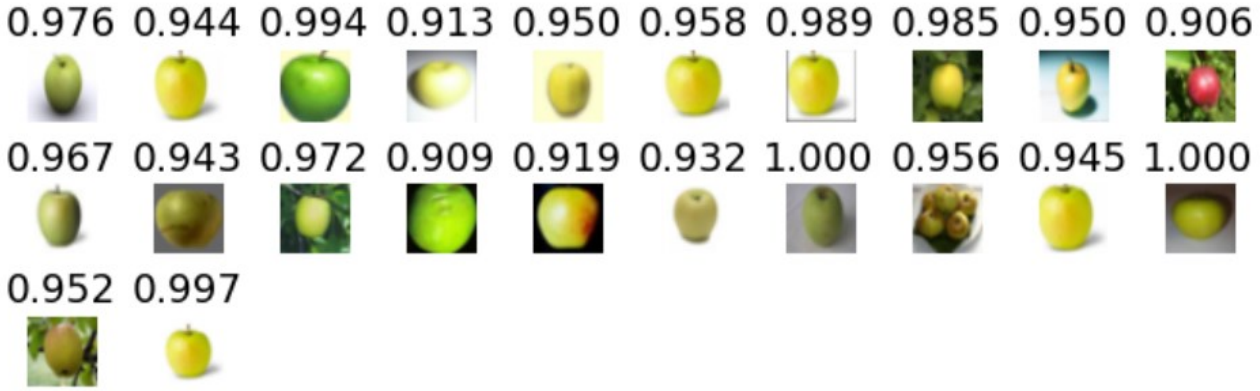




Deep Learning's Challenges – Uncertainty

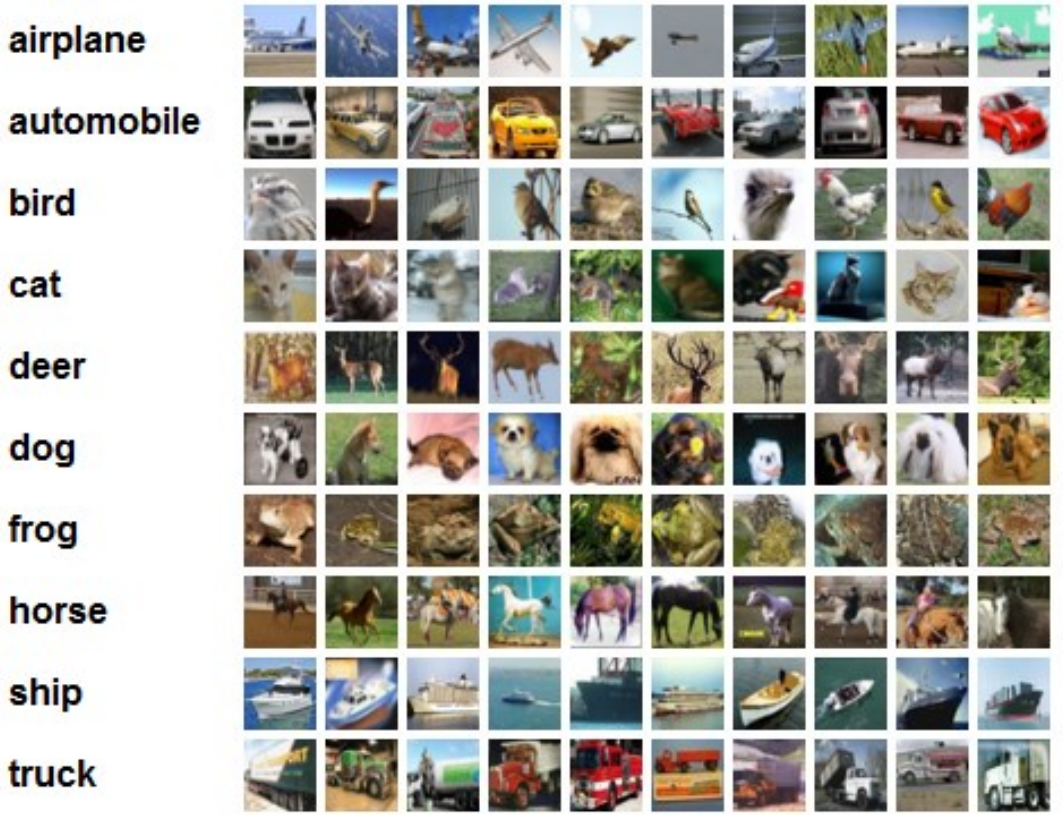


CIFAR-100's *apple* misclassified as CIFAR-10's *automobile* class with $p > 0.9$.



CIFAR-100's *apple* misclassified as CIFAR-10's *frog* class with $p > 0.9$.

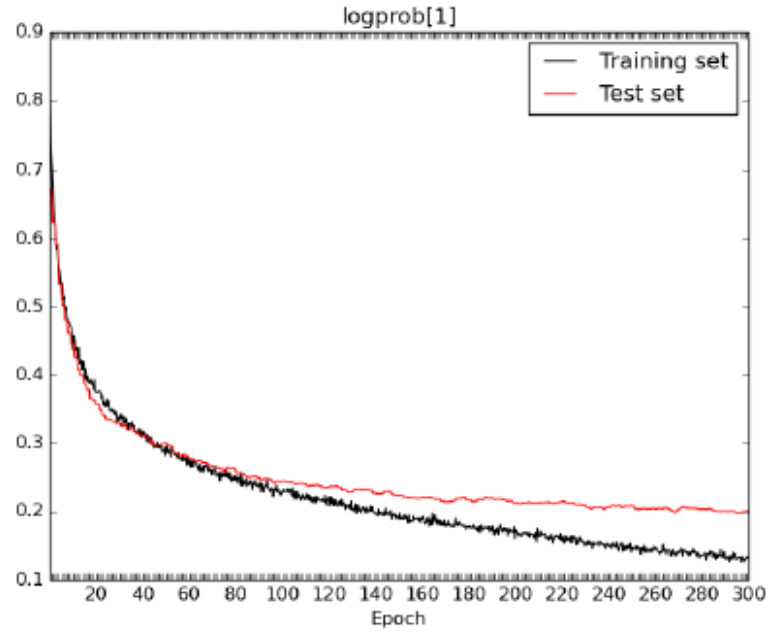
Cifar 10 dataset



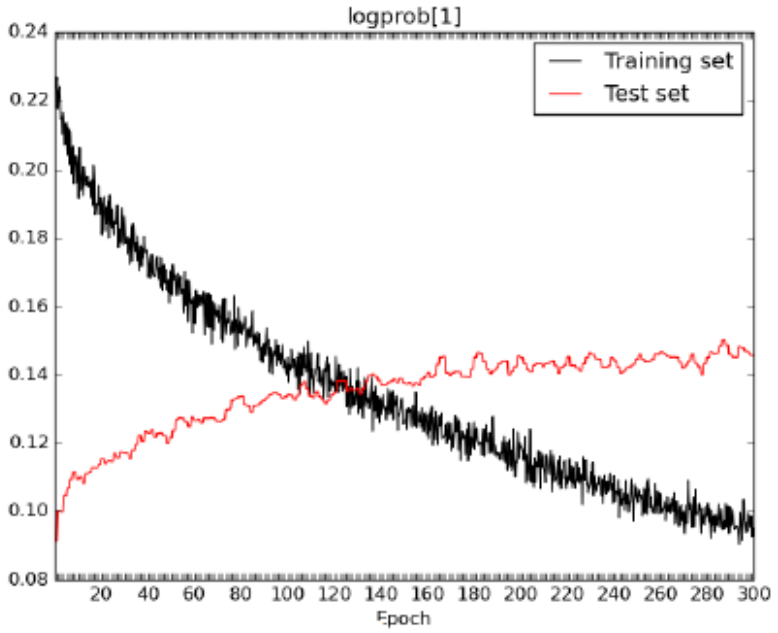
Deep Learning's Challenges – Catastrophic Forgetting Problem

- 학습 데이터를 30,000(group 1), 30,000(group 2)로 구성
- Group 1로 network 학습(test는 group 2)
- 학습된 weights를 initial weights로 하여 group 2를 다시 학습

Error-rate



Error-rate

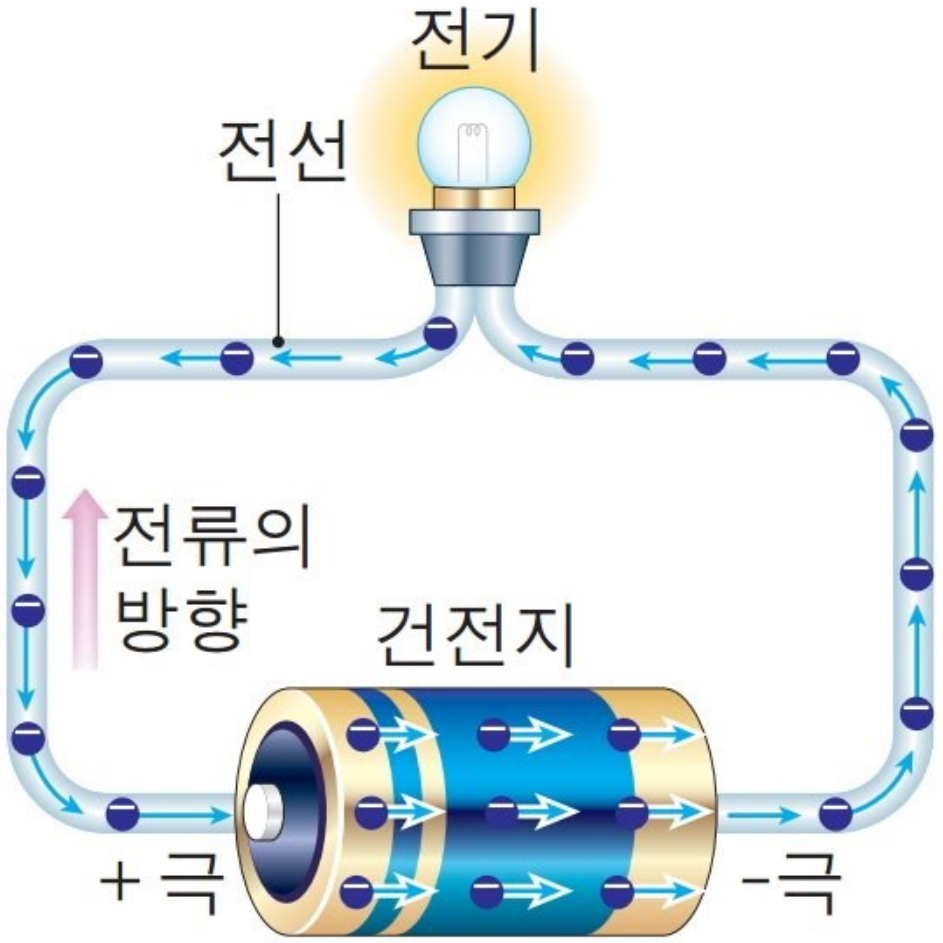


Limitations of Deep Learning

- Not sample efficient
- Compute intensive
- Poor at handling prior knowledge or uncertainty
- Not interpretable
- Suffers from hyper-parameter optimization
- Vulnerable to adversarial attack

Today's Artificial Intelligence?

L'Avion III de Clément Ader, 1897
(Musée du CNAM, Paris)



▲ 전자의 이동과 전류의 방향

From 중학교 3학년 과학3 교과서