# Machine Learning Tutorial

YUN-NUNG (VIVIAN) CHEN
HTTP://VIVIANCHEN.IDV.TW

National Taiwan University

**TSMC**
Jan 23rd, 2016

Slide credit from Hung-Yi Lee and Mark Chang

# Talk Outline



Part I: Introduction to
Machine Learning & Deep Learning

Part II: Variants of Neural Nets

Part III: Beyond Supervised Learning
& Recent Trends

# Talk Outline



Part I: Introduction to Machine Learning & Deep Learning

Part II: Variants of Neural Nets

Part III: Beyond Supervised Learning & Recent Trends

# PART I

Introduction to Machine Learning &
Deep Learning

# Part I: Introduction to ML & DL



- Basic Machine Learning
- Basic Deep Learning
- Toolkits and Learning Recipe

# Part I: Introduction to ML & DL

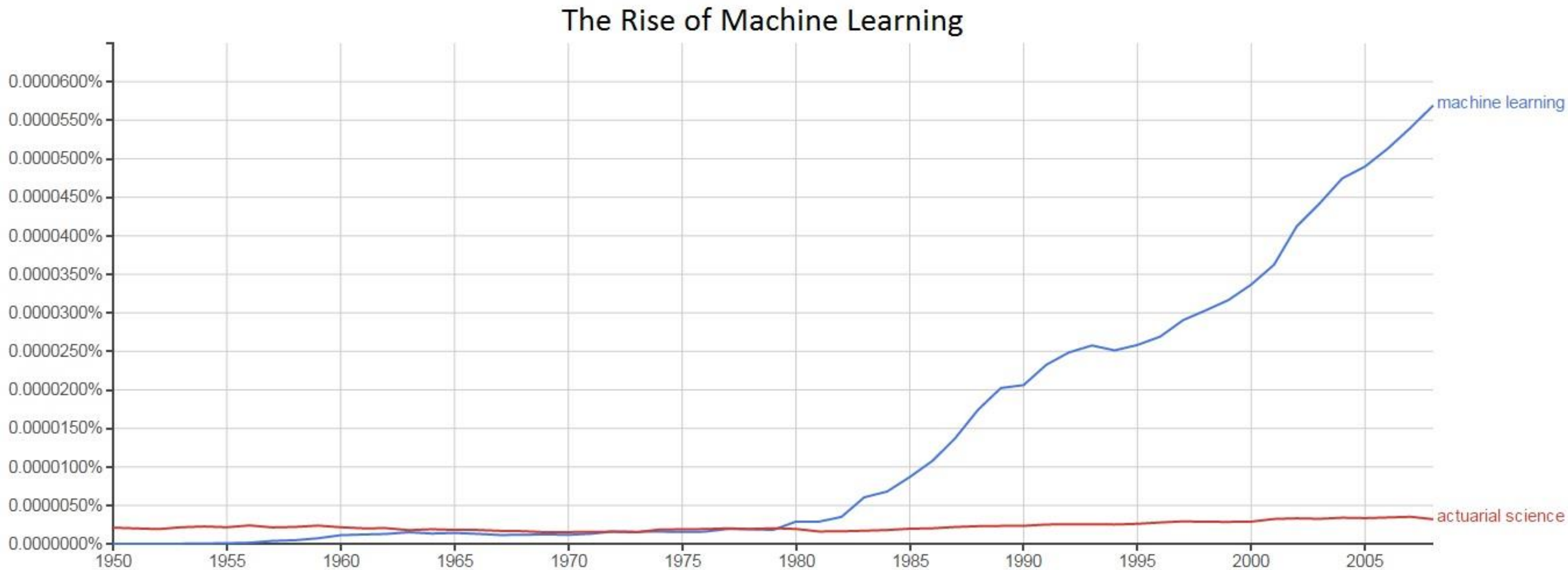


- **Basic Machine Learning**
- Basic Deep Learning
- Toolkits and Learning Recipe

# Machine Learning



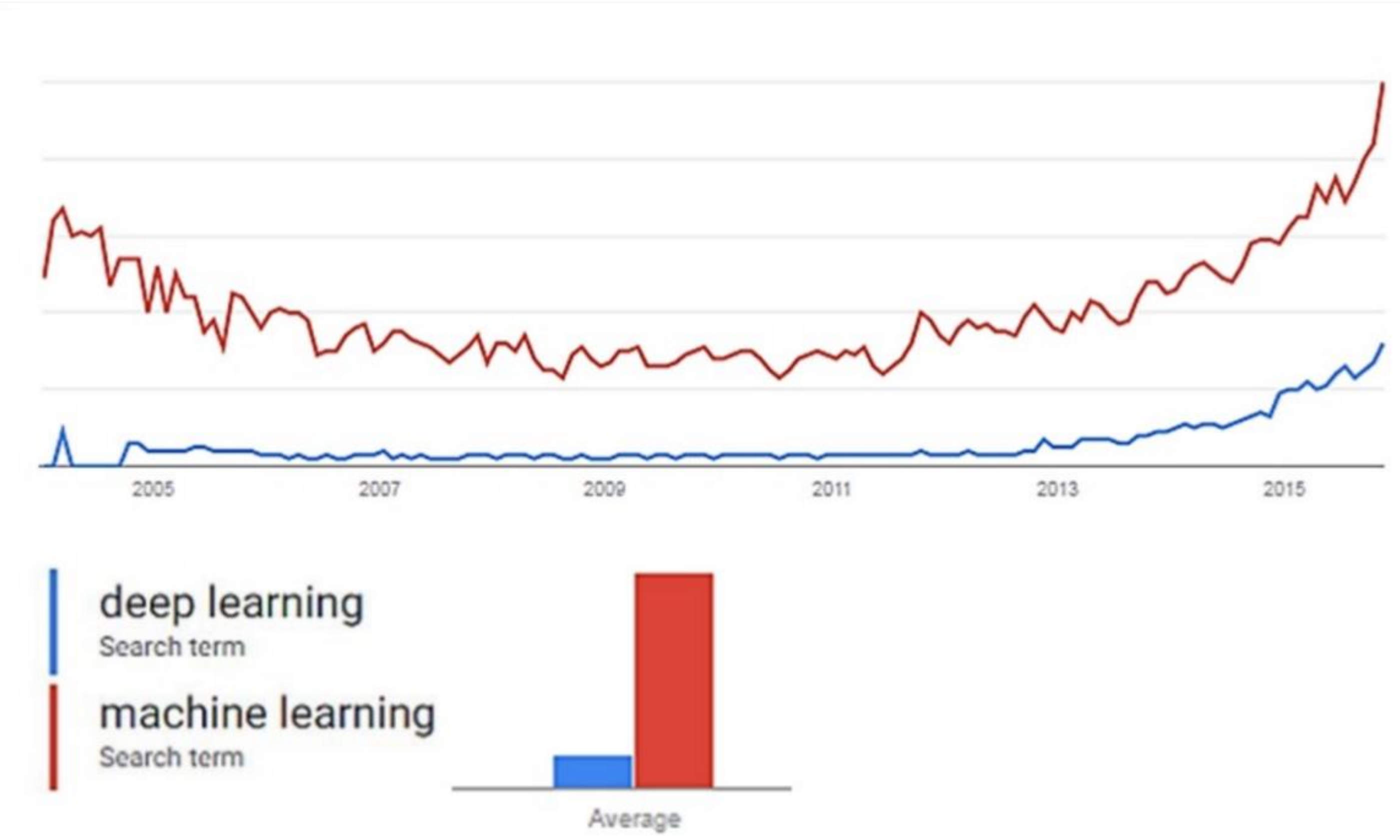
- Machine learning is rising rapidly in recent days




What the y-axis shows is this: of all the bigrams (two word letter combinations) contained in Google's sample of books written in English, what percentage of them are "machine learning" or "actuarial science"?

# Recent Trend





# What Computers Can Do?




Programs can do the things you ask them to do

# Program for Solving Tasks



- Task: predicting positive or negative given a product review

"I love this product!" → program.py → +

"It claims too much." → program.py → -

"It's a little expensive." → program.py → ?

if input contains "love", "like", etc.
output = positive

if input contains "too much", "bad", etc.
output = negative

"台灣第一波上市!" → program.py → 推

"規格好雞肋…" → program.py → 噓

"樓下買了我才考慮" → program.py → ?

Some tasks are complex, and we don't know how to write a program to solve them.

# Learning ≈ Looking for a Function



- Task: predicting positive or negative given a product review

"I love this product!" → $f$ → +

"It claims too much." → $f$ → -

"It's a little expensive." → $f$ → ?

"台灣第一波上市!" → $f$ → 推

"規格好雞肋..." → $f$ → 噓

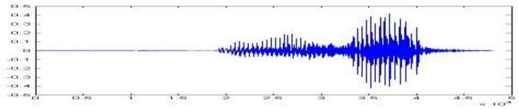
"樓下買了我才考慮" → $f$ → ?

Given a large amount of data, the machine learns what the function ***f*** should be.

# Learning ≈ Looking for a Function



- Speech Recognition

$f($  $)=$ "你好"

- Image Recognition

$f($  $)=$ cat

- Go Playing

$f($  $)=$ 5-5 (next move)

- Dialogue System

$f($ "台積電怎麼去" $)=$ "地址為…
現在建議搭乘計程車"

# Framework

Image Recognition:

$f($  $) =$ "cat"



A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $) =$ "cat"

$f_2($  $) =$ "monkey"

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

# Framework



Image Recognition:

$f($  $)=$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

Goodness of function f

Training Data

$f_1($  $)=$ "cat"

$f_2($  $)=$ "money"

Better!

$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Supervised Learning

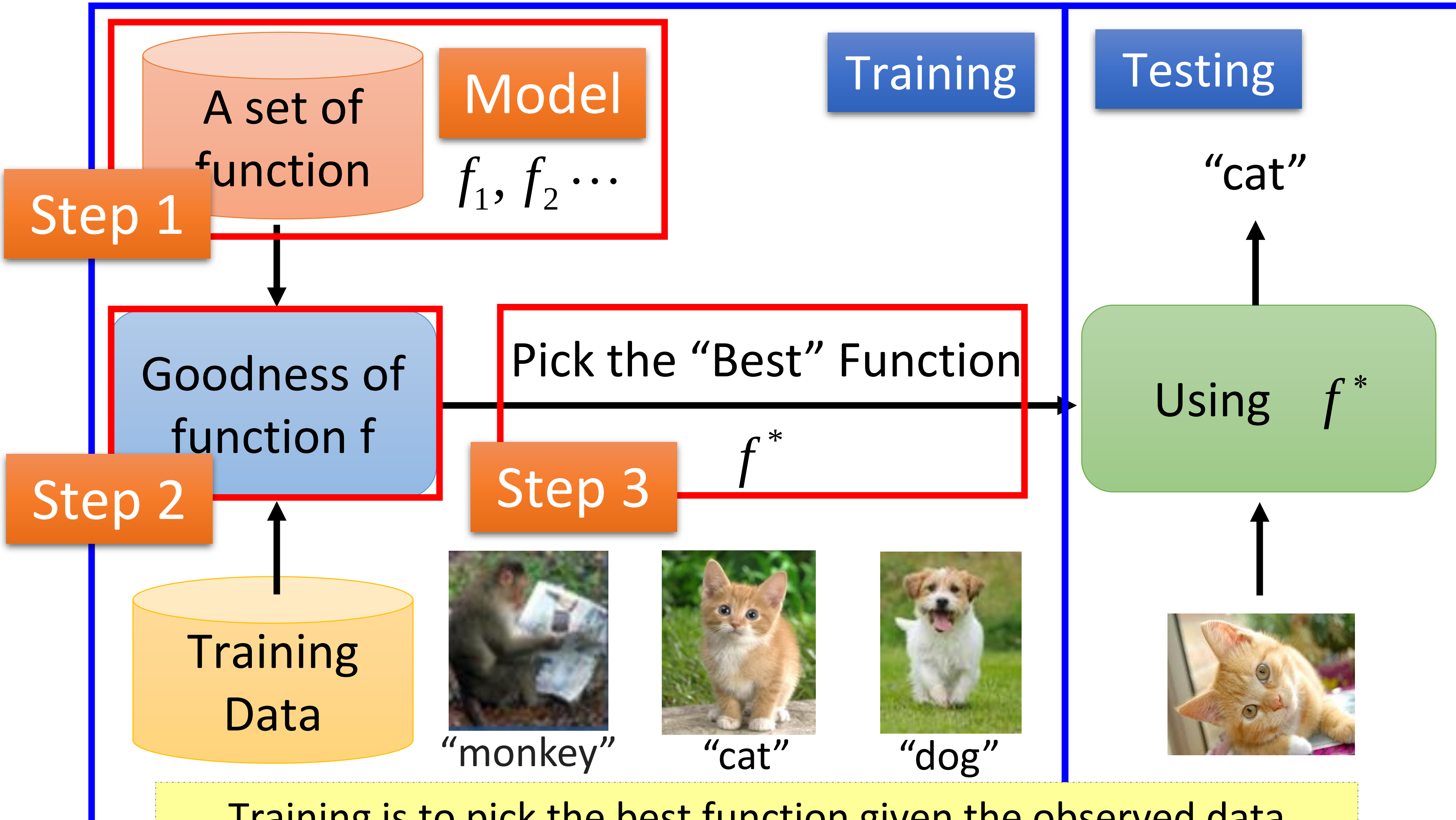
function input:   

function output: "monkey" "cat" "dog"

# Framework

Image Recognition:

$f($  $) =$ "cat"

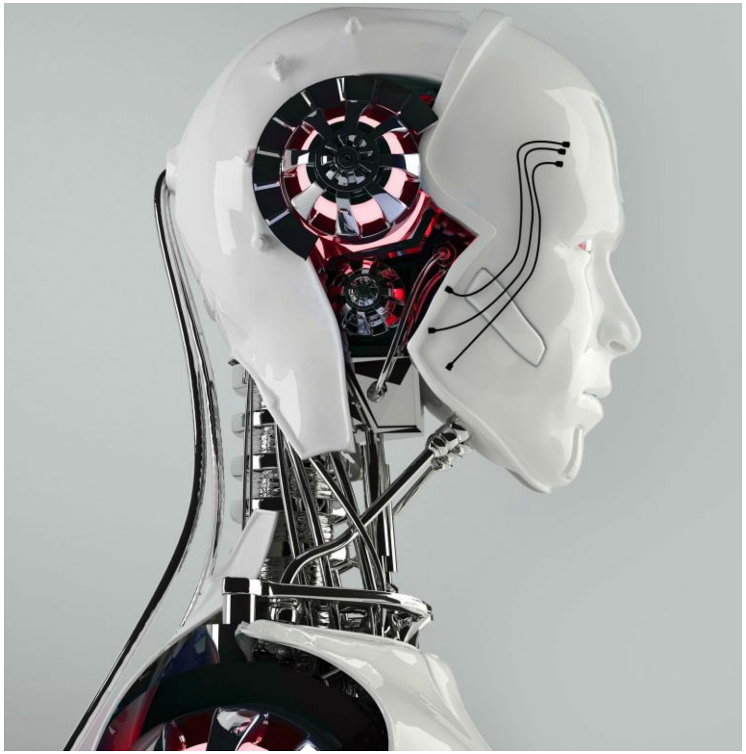



Training is to pick the best function given the observed data
Testing is to predict the label using the learned function

# Why to Learn Machine Learning?



- AI Age
  - AI can work for most of labor work?
  - New job market → AI 訓練師

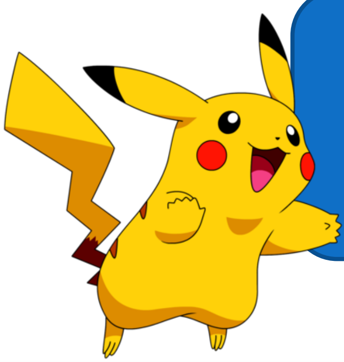


(Machine Learning Expert 機器學習專家、Data Scientist 資料科學家)

# AI 訓練師



機器不是自己會學嗎？
為什麼需要 AI 訓練師



戰鬥是寶可夢在打，
為什麼需要寶可夢訓練師？

# AI 訓練師

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function



- 寶可夢訓練師
  - 挑選適合的寶可夢來戰鬥
    - 寶可夢有不同的屬性
  - 召喚出來的寶可夢不一定聽話
    - E.g. 小智的噴火龍

→ 需要足夠的經驗

- AI 訓練師
  - 在 step 1，AI訓練師要挑選合適的模型
    - 不同模型適合處理不同的問題
  - 不一定能在 step 3 找出 best function
    - E.g. Deep Learning

→ 需要足夠的經驗

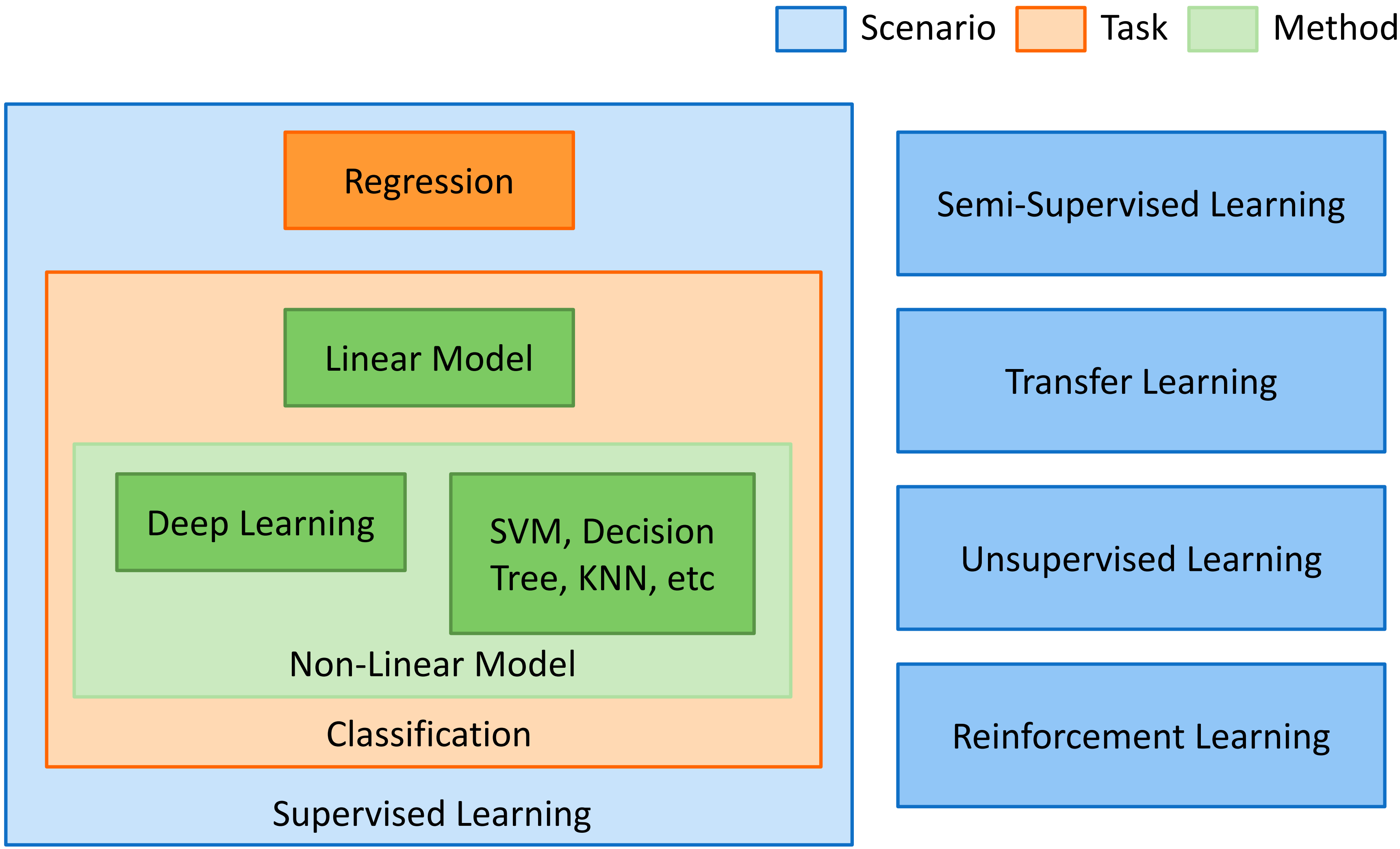
# AI 訓練師



- 厲害的 AI ， AI 訓練師功不可沒
- 讓我們一起朝 AI 訓練師之路邁進

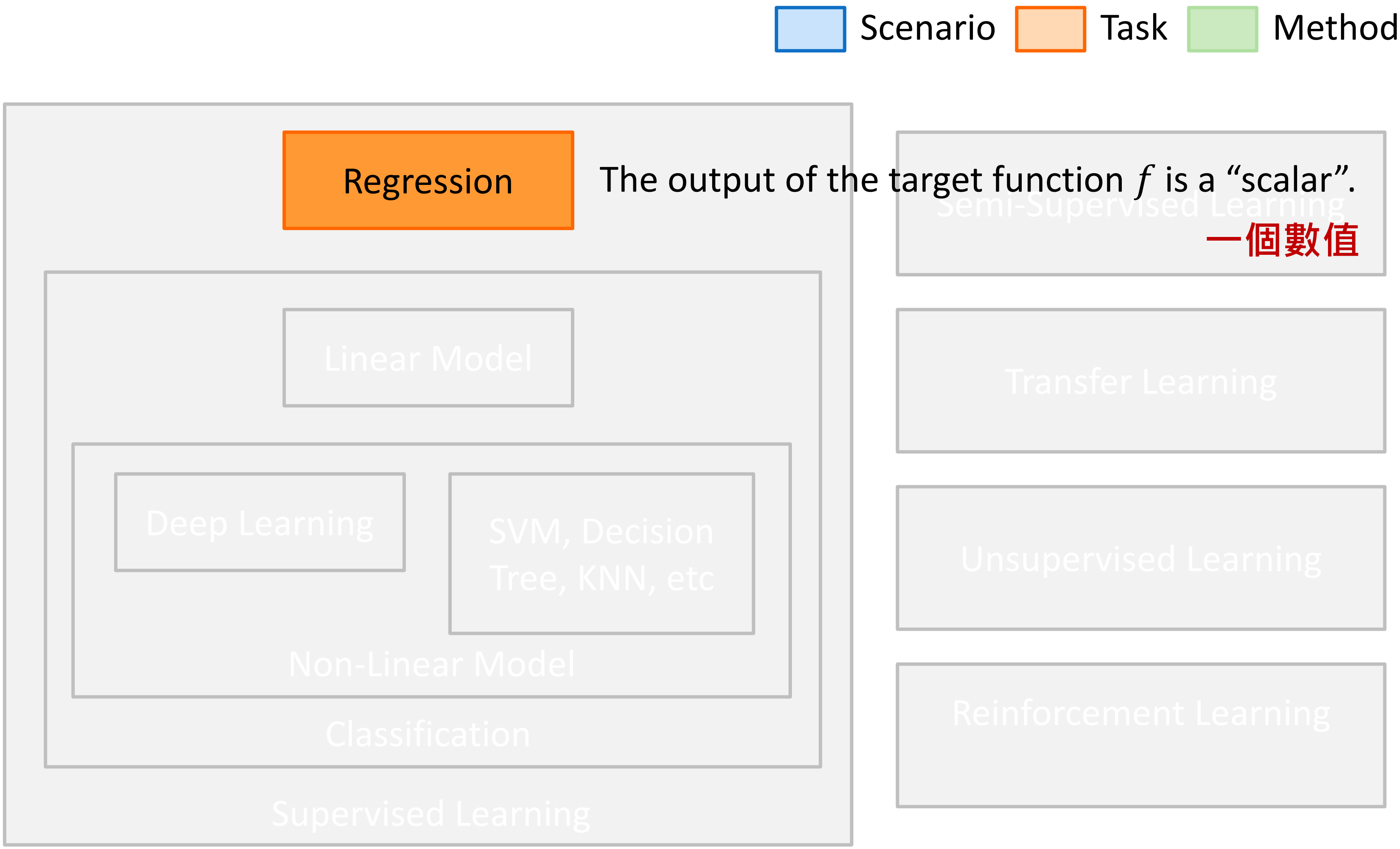
# Machine Learning Map





# Machine Learning Map





# Regression



- Stock Market Forecast

$f($  $) =$ Dow Jones Industrial Average at tomorrow

- Self-driving Car

$f($  $) =$ 方向盤角度

- Recommendation

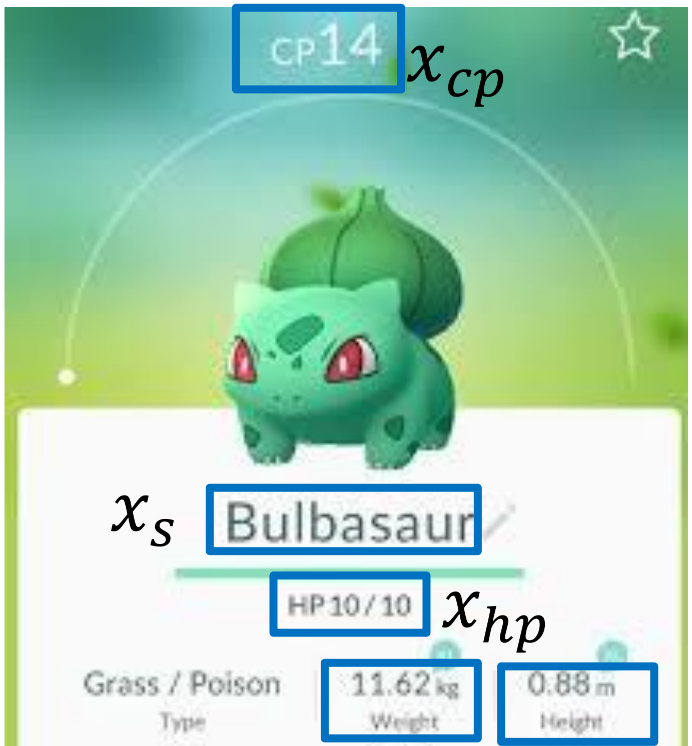
$f($ 使用者 A 商品 B $) =$ 購買可能性

# Example Application



- Estimating the Combat Power (CP) of a pokemon after evolution

$f($




$x_{cp}$ $x_s$ $x_{hp}$ $x_w$ $x_h$

$x$

$) =$ CP after evolution $y$

# Step 1: Model



$y = b + w \cdot x_{\mathrm{cp}}$

A set of function

Model

$f_1, f_2 \cdots$

w and b are parameters (can be any value)

$f_1$: $y = 10.0 + 9.0 \cdot x_{cp}$

$f_2$: $y = 9.8 + 9.2 \cdot x_{cp}$

$f_3$: $y = -0.8 - 1.2 \cdot x_{cp}$

...... infinite

$f($ 

$x) =$ CP after evolution $y$

Linear model: $y = b + \sum w_i x_i$

$x_i$: an attribute of input x — feature

$w_i$: weight, b: bias

# Step 2: Goodness of Function



$y = b + w \cdot x_{\mathrm{cp}}$

A set of function

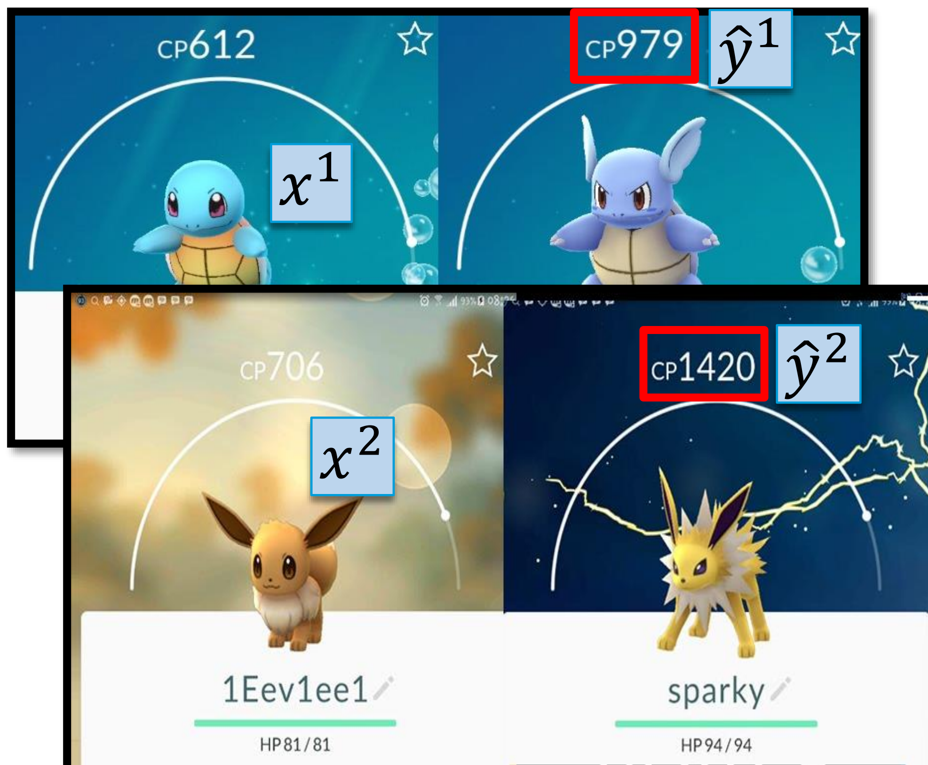
Model

$f_1, f_2 \cdots$

Training Data

function input:

function output (scalar):

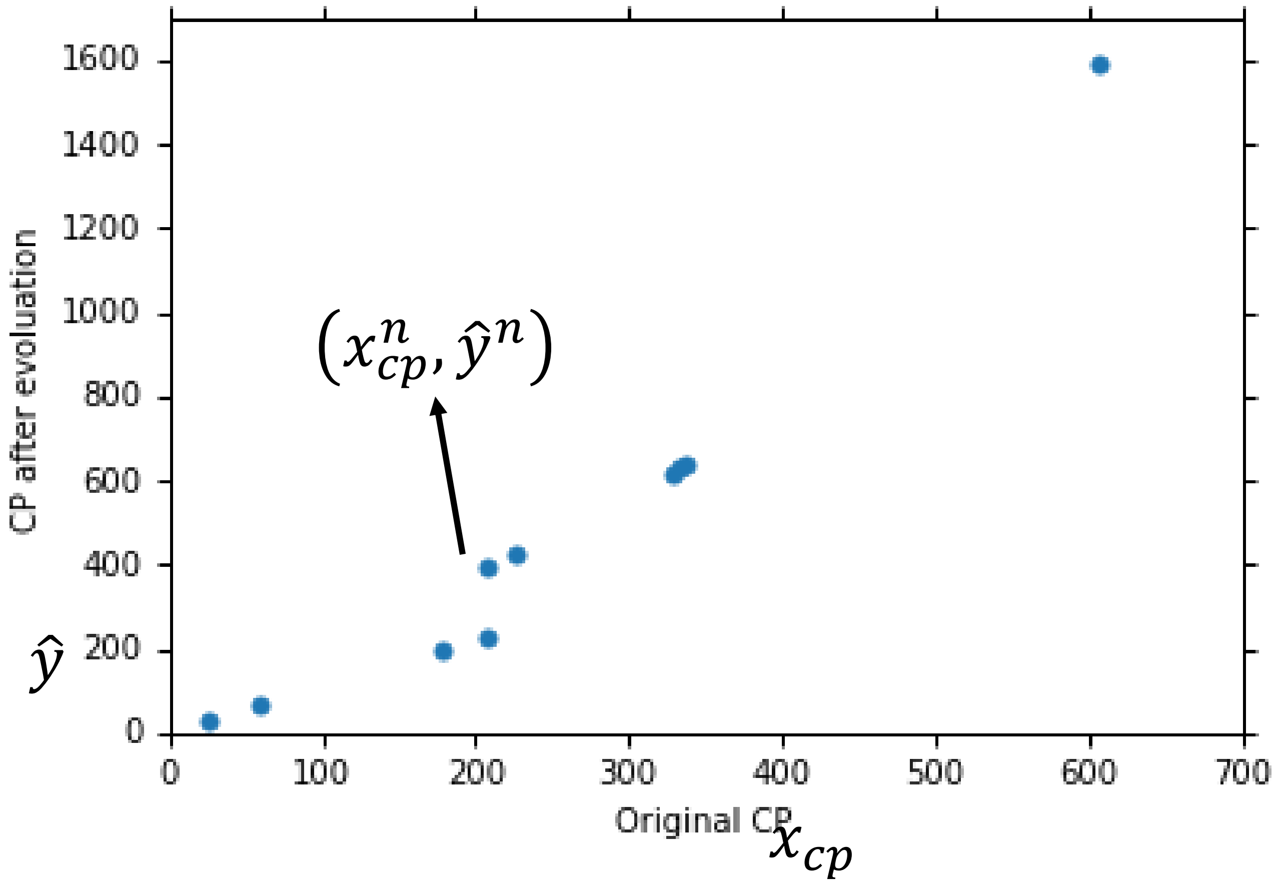


# Step 2: Goodness of Function



- Training data
  - $1^{st}$ pokemon: $(x^1, \hat{y}^1)$
  - $2^{nd}$ pokemon: $(x^2, \hat{y}^2)$
  
  ⋮
  
  - $10^{th}$ pokemon: $(x^{10}, \hat{y}^{10})$

This is real data.

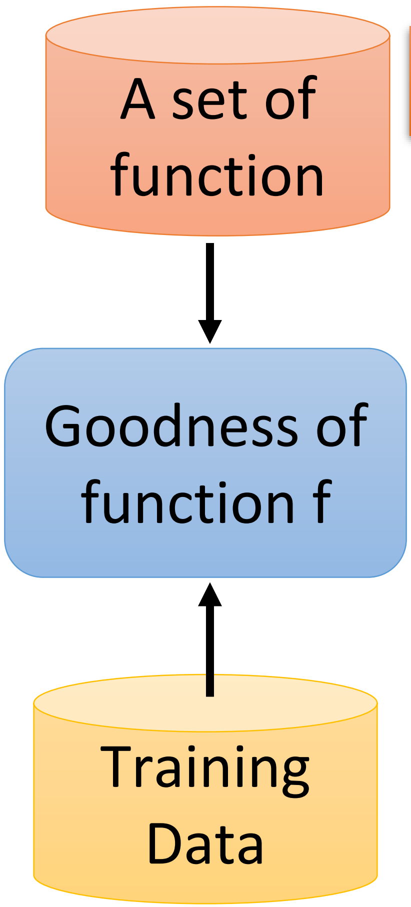



Source: https://www.openintro.org/stat/data/?data=pokemon

# Step 2: Goodness of Function



$y = b + w \cdot x_{\text{cp}}$




Model

$f_1, f_2 \cdots$

Loss function $L$:

Input: a function, output: how bad it is

$$\mathrm{L}(f) = \mathrm{L}(w, b)$$

Estimated y based on input function

$$= \sum_{n=1}^{10} \boxed{\left( \hat{y}^n - \left( b + w \cdot x_{cp}^n \right) \right)^2}$$

Estimation error

Sum over examples

# Step 2: Goodness of Function

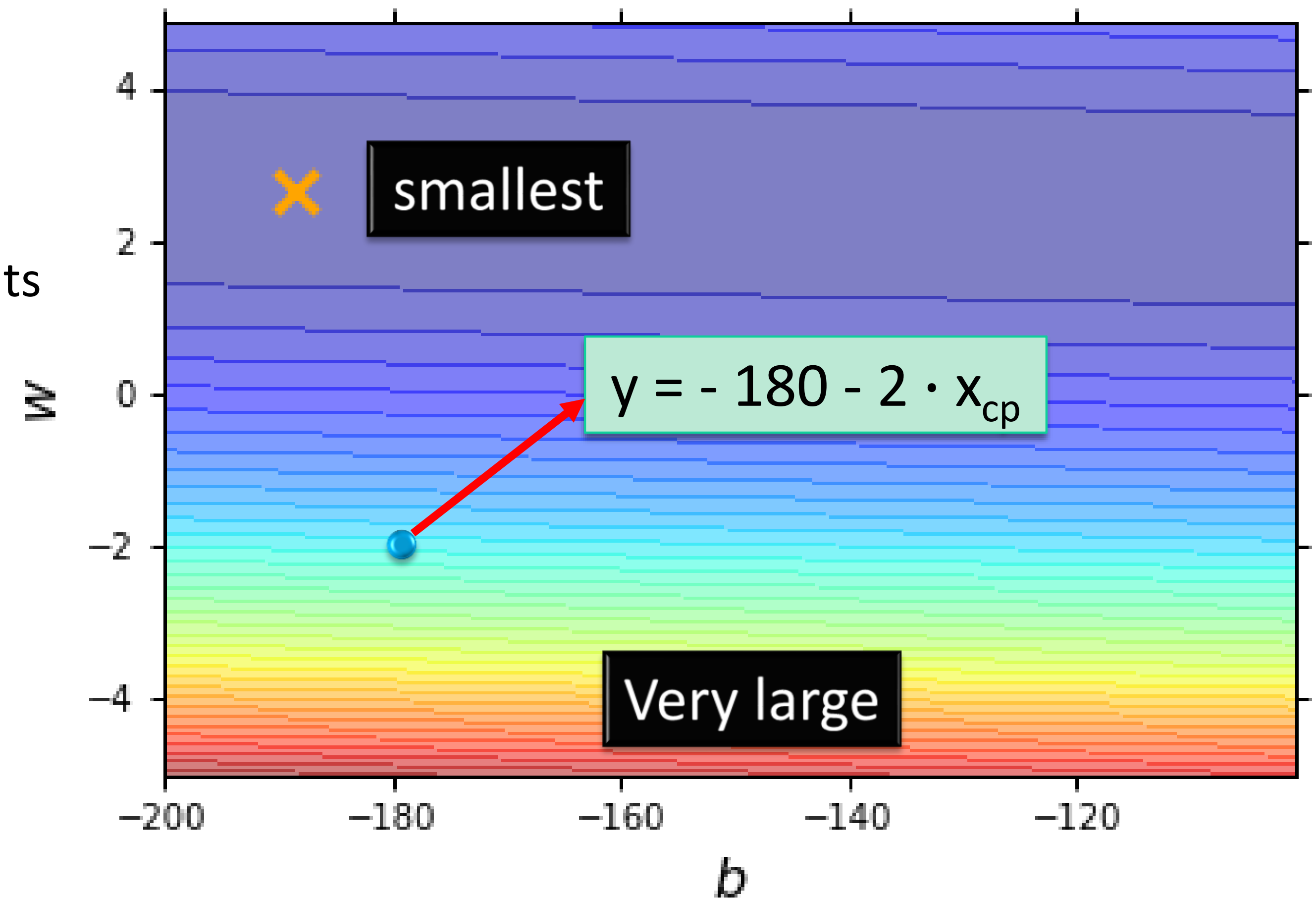


- Loss Function

$$L(w,b)=\sum_{n=1}^{10}\left(\hat{y}^n-\left(b+w\cdot x_{cp}^n\right)\right)^2$$

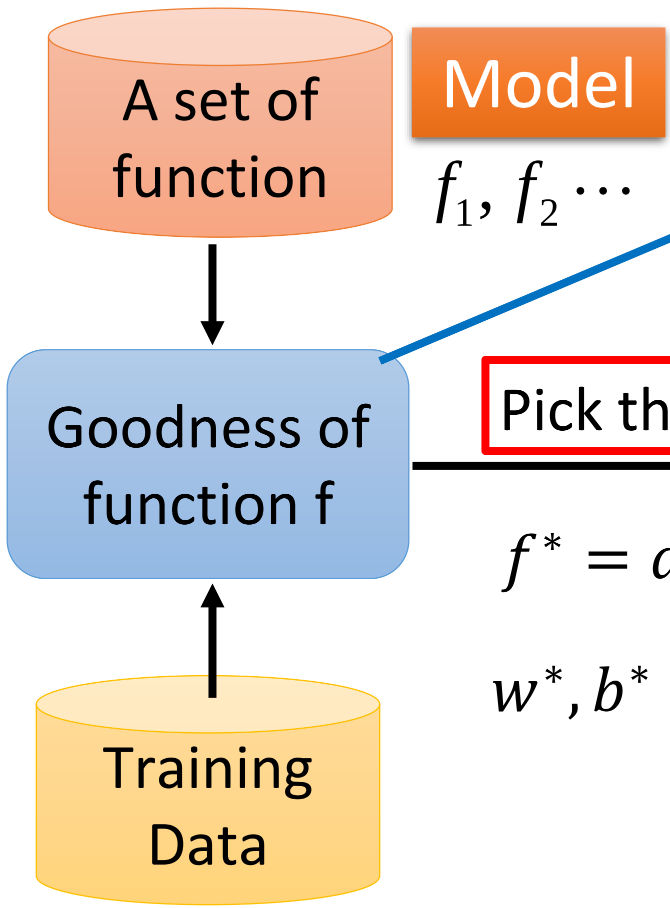
Each point in the figure is a function

The color represents $L(w,b)$



# Step 3: Best Function






$$L(w,b) = \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n)\right)^2$$

Pick the "Best" Function

$$f^* = arg \min_f L(f)$$

$$w^*, b^* = arg \min_{w,b} L(w,b)$$

$$= arg \min_{w,b} \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n)\right)^2$$

Gradient Descent

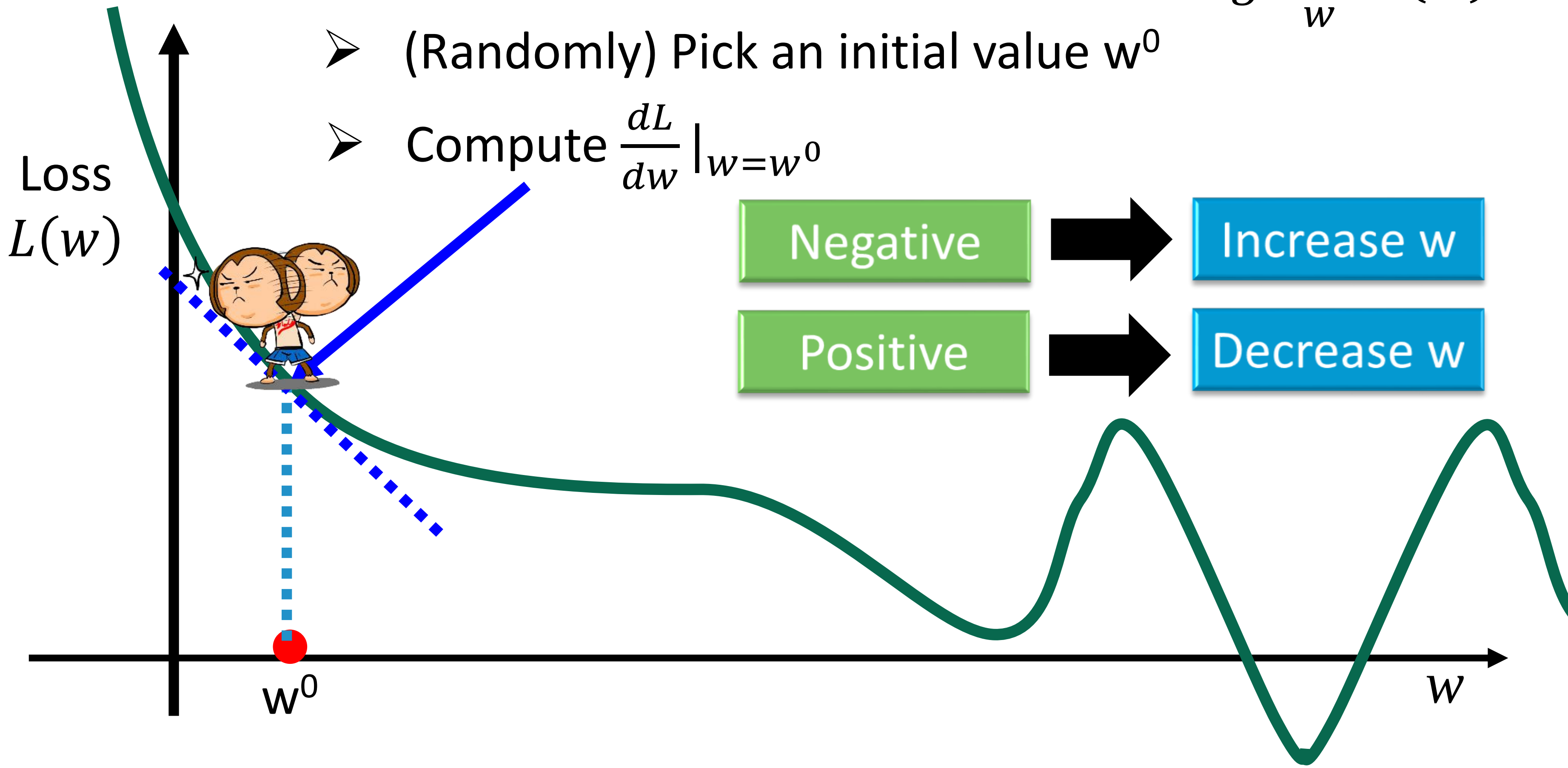
# Step 3: Gradient Descent



- Consider loss function $L(w)$ with one parameter w:

$$w^* = arg\min_w L(w)$$

  - ➢ (Randomly) Pick an initial value $w^0$
  - ➢ Compute $\frac{dL}{dw}|_{w=w^0}$



# Step 3: Gradient Descent

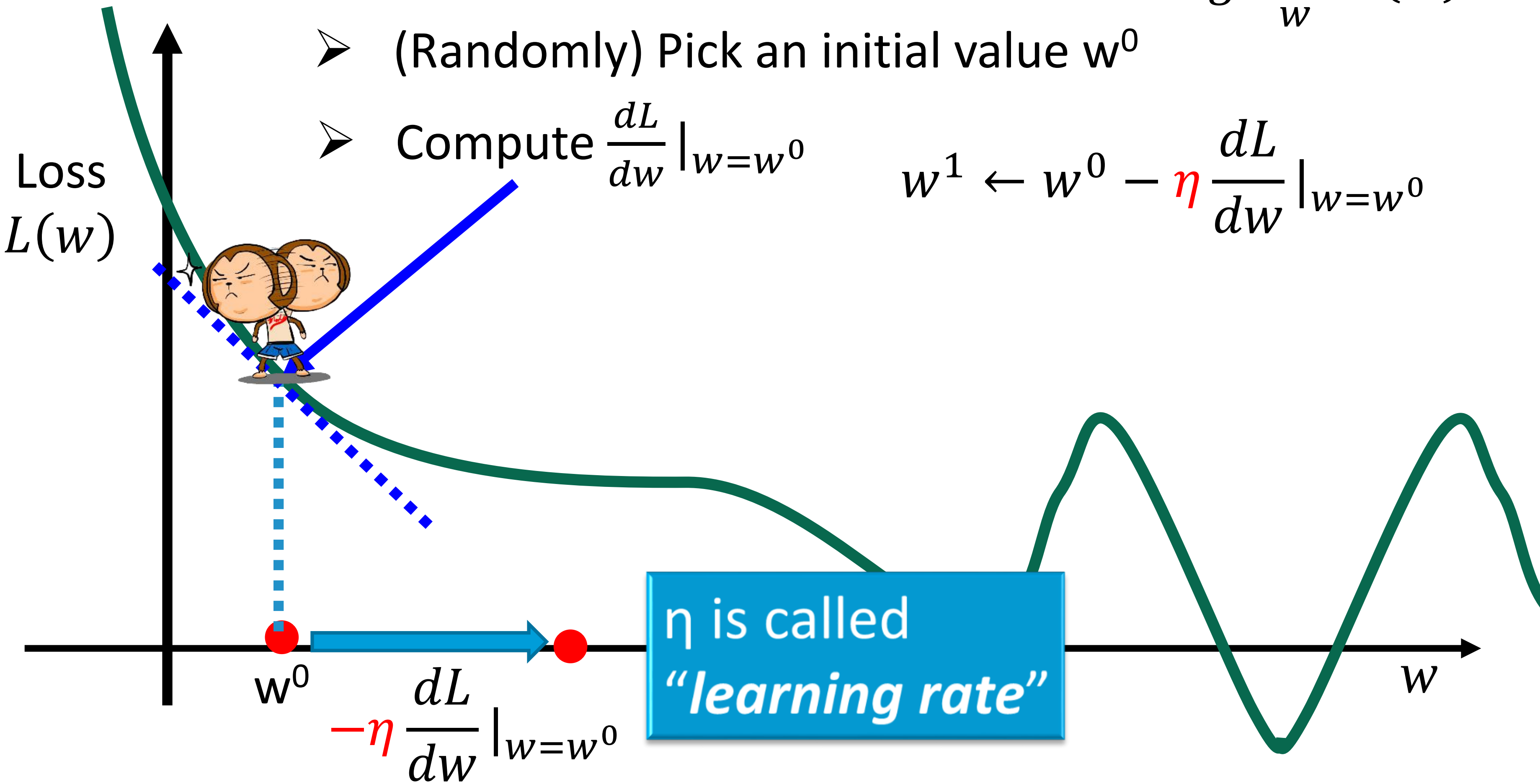


- Consider loss function $L(w)$ with one parameter w:

$$w^* = arg\min_w L(w)$$

  - (Randomly) Pick an initial value $w^0$
  - Compute $\frac{dL}{dw}|_{w=w^0}$

$$w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$$




η is called *"learning rate"*

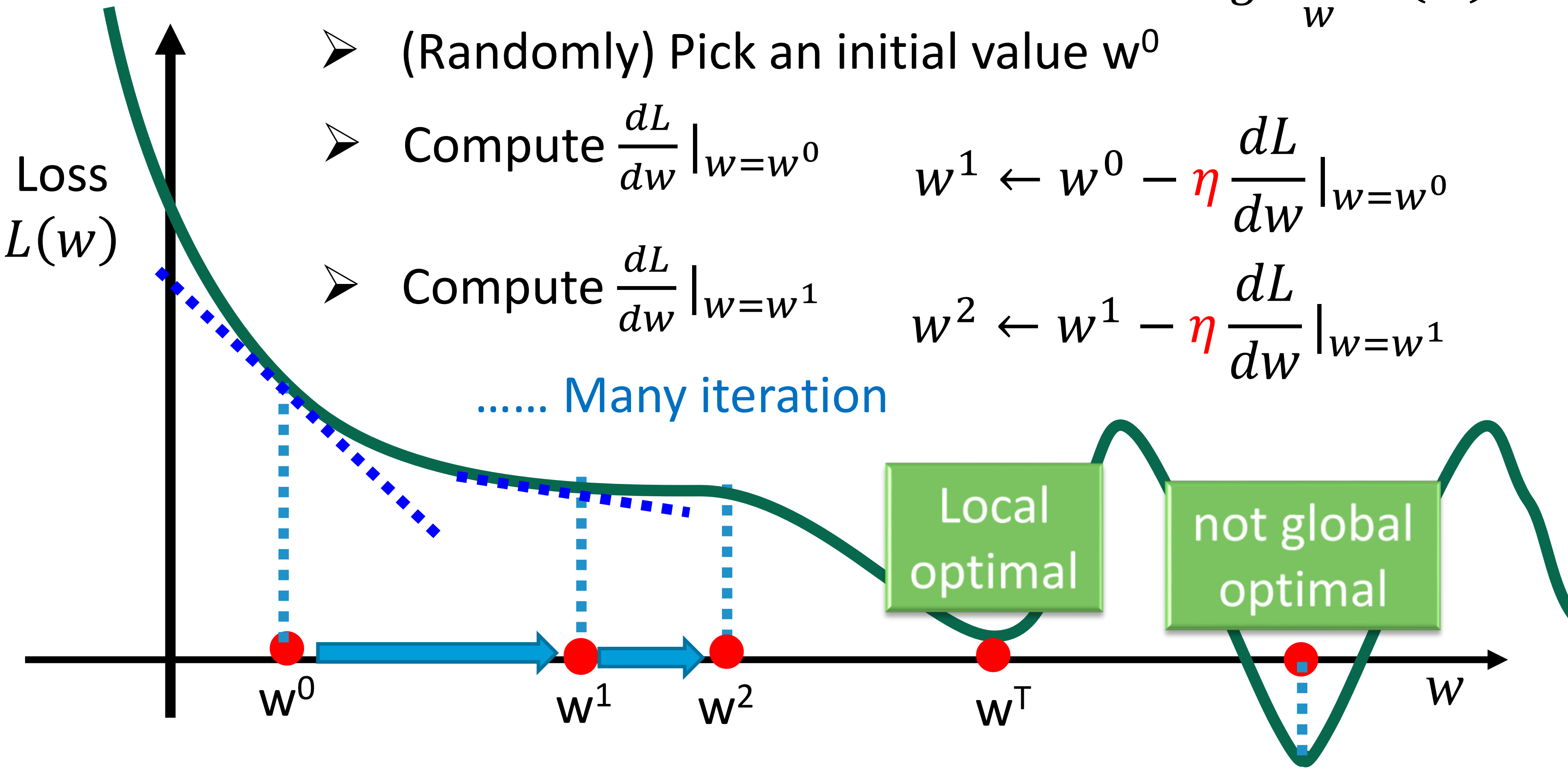
# Step 3: Gradient Descent

- Consider loss function $L(w)$ with one parameter w:

$$w^* = arg \min_w L(w)$$

- (Randomly) Pick an initial value $w^0$
- Compute $\frac{dL}{dw}|_{w=w^0}$ $\quad w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$
- Compute $\frac{dL}{dw}|_{w=w^1}$ $\quad w^2 \leftarrow w^1 - \eta \frac{dL}{dw}|_{w=w^1}$

…… Many iteration



# Step 3: Gradient Descent



- How about two parameters? $w^*, b^* = arg \min_{w,b} L(w, b)$
  - (Randomly) Pick an initial value $w^0$, $b^0$
  - Compute $\frac{\partial L}{\partial w}|_{w=w^0, b=b^0}$, $\frac{\partial L}{\partial b}|_{w=w^0, b=b^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0, b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0, b=b^0}$$

  - Compute $\frac{\partial L}{\partial w}|_{w=w^1, b=b^1}$, $\frac{\partial L}{\partial b}|_{w=w^1, b=b^1}$

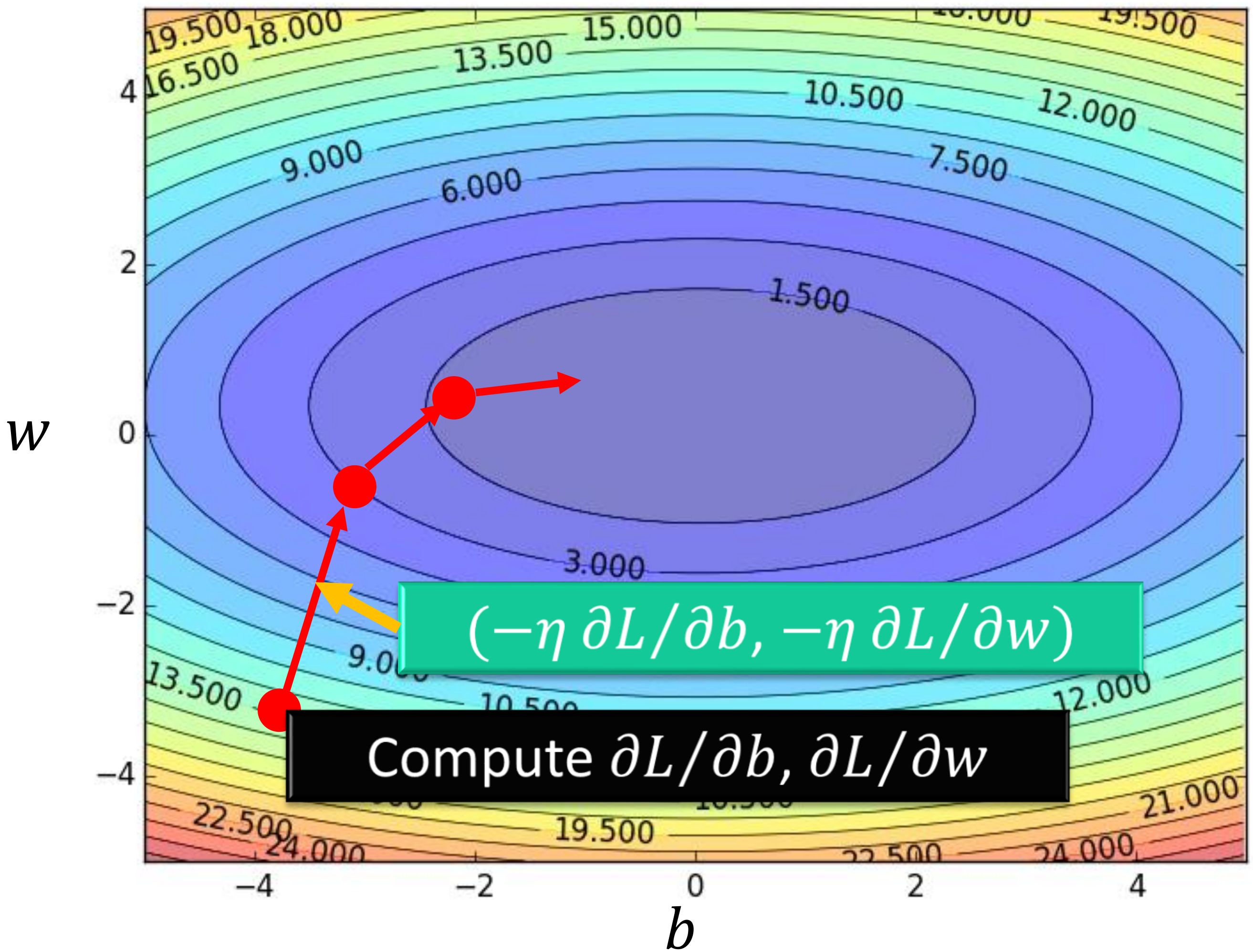
$$w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^1, b=b^1} \qquad b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^1, b=b^1}$$

$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$ gradient

# Step 3: Gradient Descent



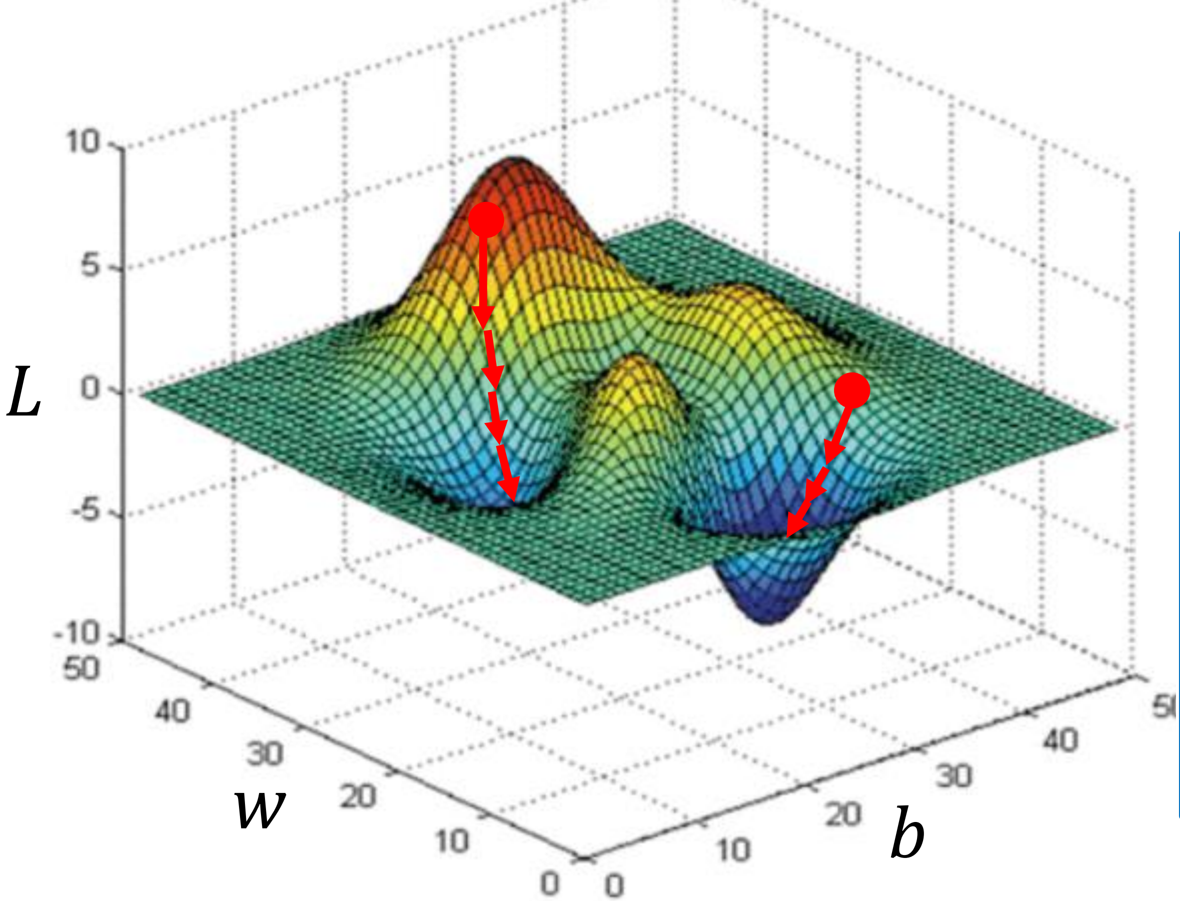
Color: Value of loss $L(w, b)$



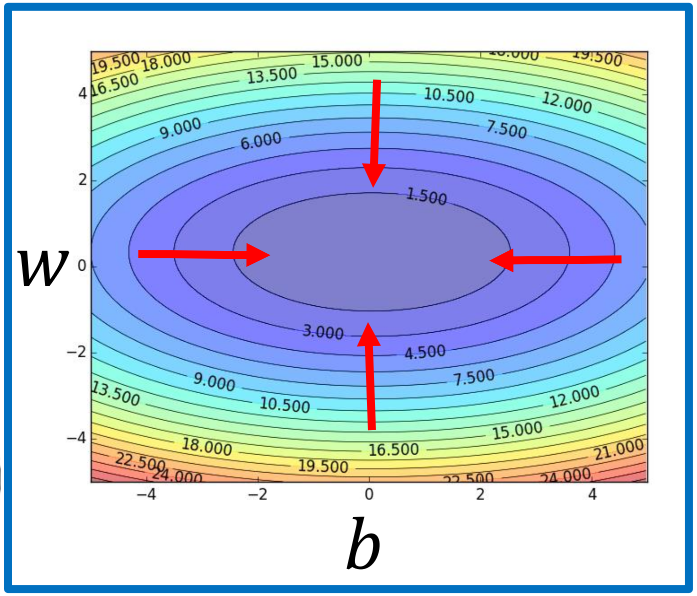
# Step 3: Gradient Descent



- Local optimal
  - Loss function is convex in linear regression




Linear regression → No local optimal



# Step 3: Gradient Descent



- Formulation of $\partial L/\partial w$ and $\partial L/\partial b$

$$L(w,b) = \sum_{n=1}^{10} \left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)^2$$

$$\frac{\partial L}{\partial w} =? \sum_{n=1}^{10} 2\left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)$$

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

# Step 3: Gradient Descent



- Formulation of $\partial L / \partial w$ and $\partial L / \partial b$

$$L(w, b) = \sum_{n=1}^{10} \left( \hat{y}^n - \left( b + w \cdot x_{cp}^n \right) \right)^2$$

$$\frac{\partial L}{\partial w} =? \sum_{n=1}^{10} 2 \left( \hat{y}^n - \left( b + w \cdot x_{cp}^n \right) \right) \left( -x_{cp}^n \right)$$

$$\frac{\partial L}{\partial b} =? \sum_{n=1}^{10} 2 \left( \hat{y}^n - \left( b + w \cdot x_{cp}^n \right) \right)$$

# Learned Model



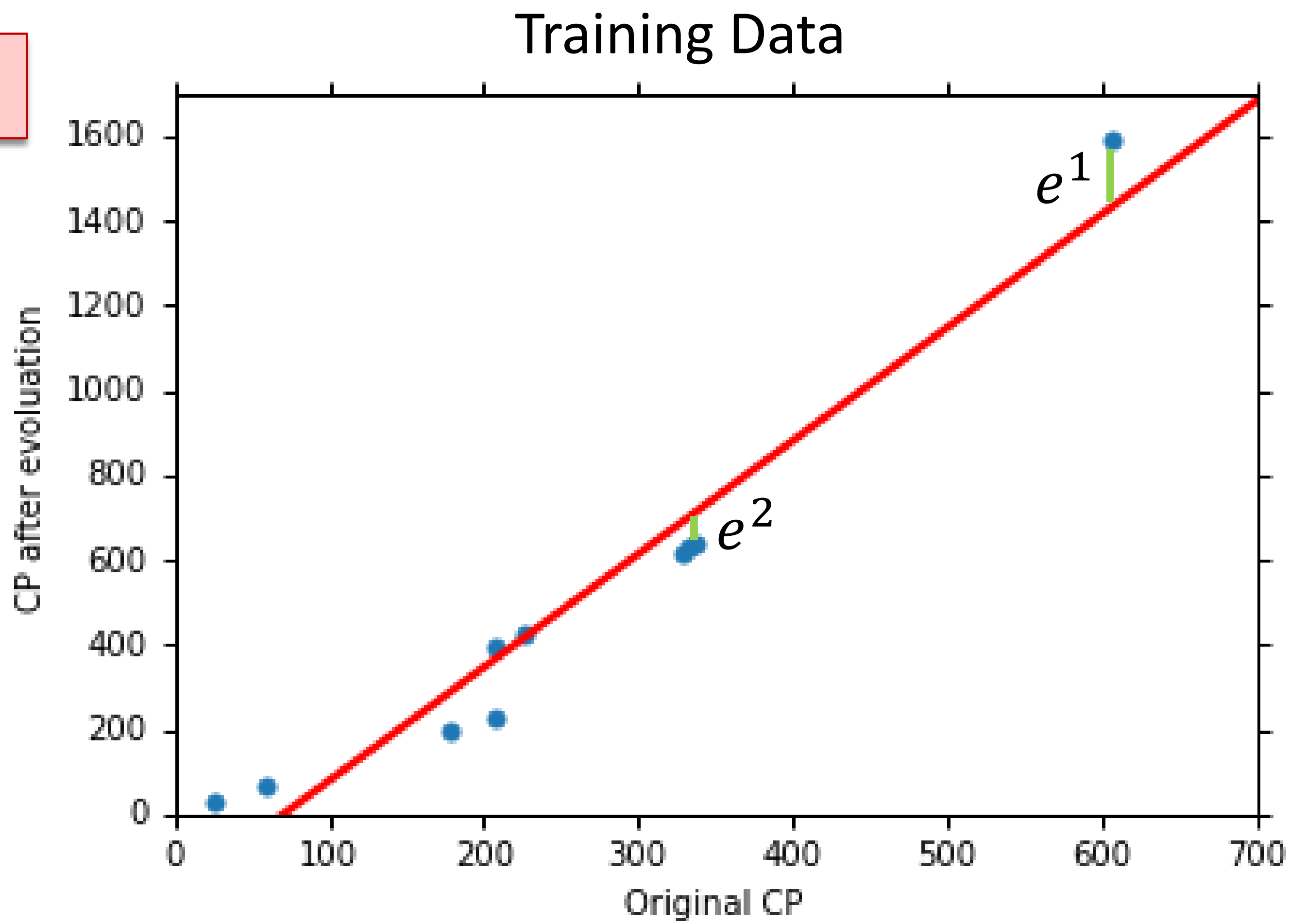
$$y = b + w \cdot x_{\text{cp}}$$

b = -188.4

w = 2.7

Average Error on Training Data

$$= \sum_{n=1}^{10} e^n = 31.9$$



# Model Generalization

What we really care about is the error on new data (testing data)



$$y = b + w \cdot x_{cp}$$

b = -188.4

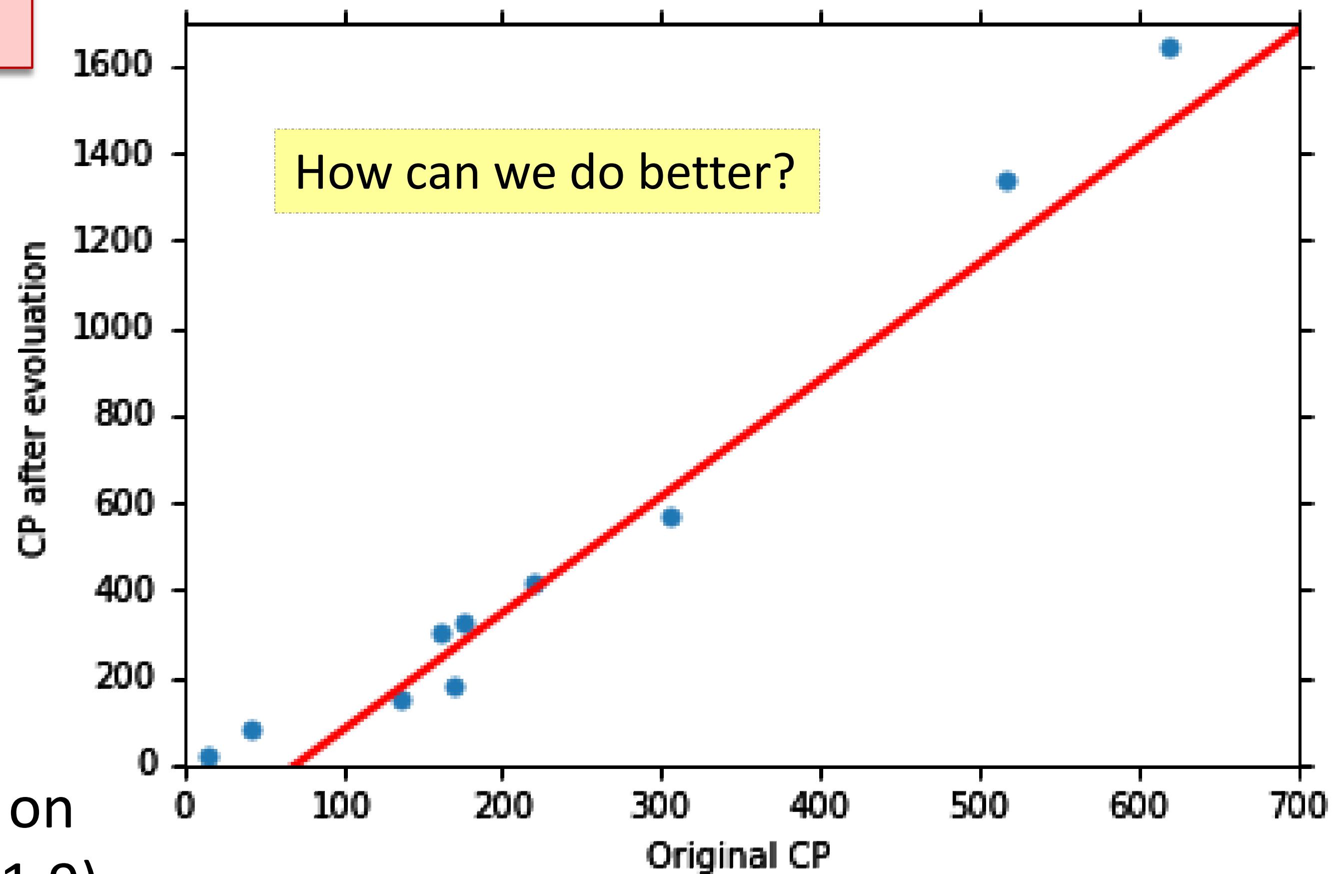
w = 2.7

Average Error on Testing Data

$$= \sum_{n=1}^{10} e^n$$ = 35.0

> Average Error on Training Data (31.9)

Another 10 pokemons as testing data



# Model Generalization



- Select another model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot \left(x_{cp}\right)^2$$

- Best function

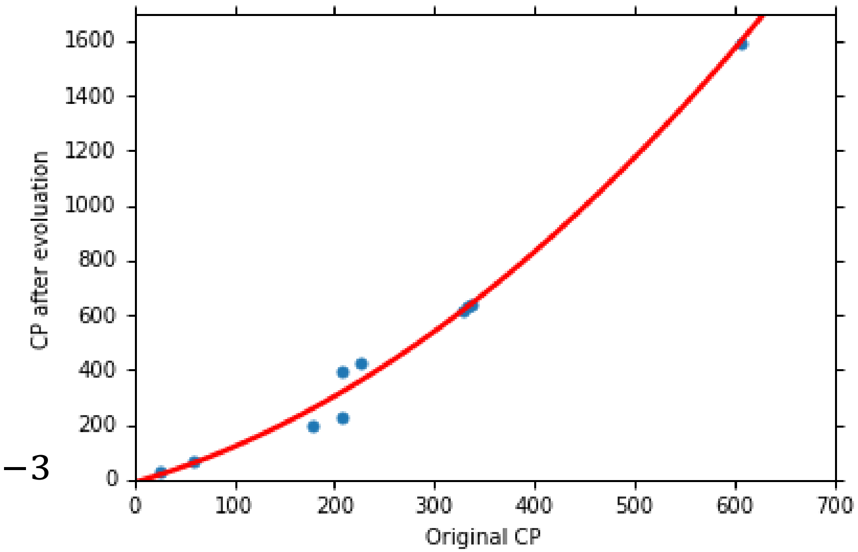
$b = -10.3, w_1 = 1.0, w_2 = 2.7 \times 10^{-3}$

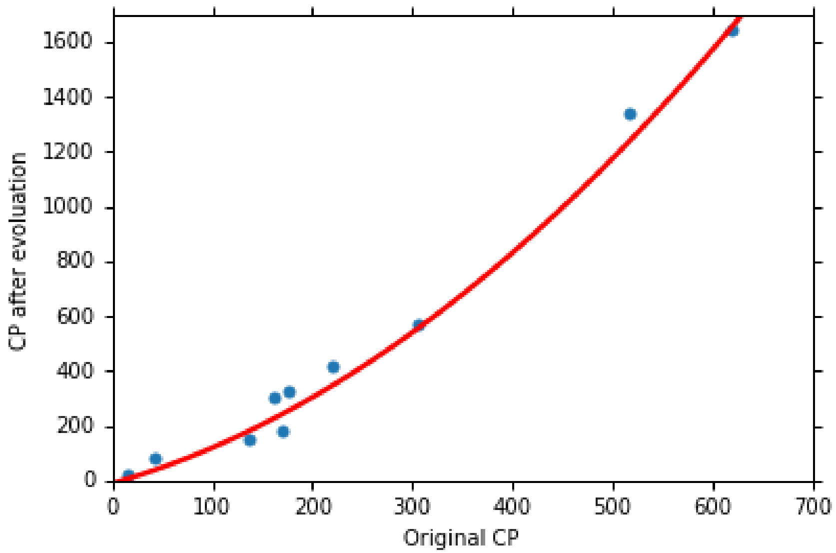
Average Error = 15.4

- Testing

Average Error = 18.4

Better! Could it be even better?



# Model Generalization



- Select another model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$$

- Best function

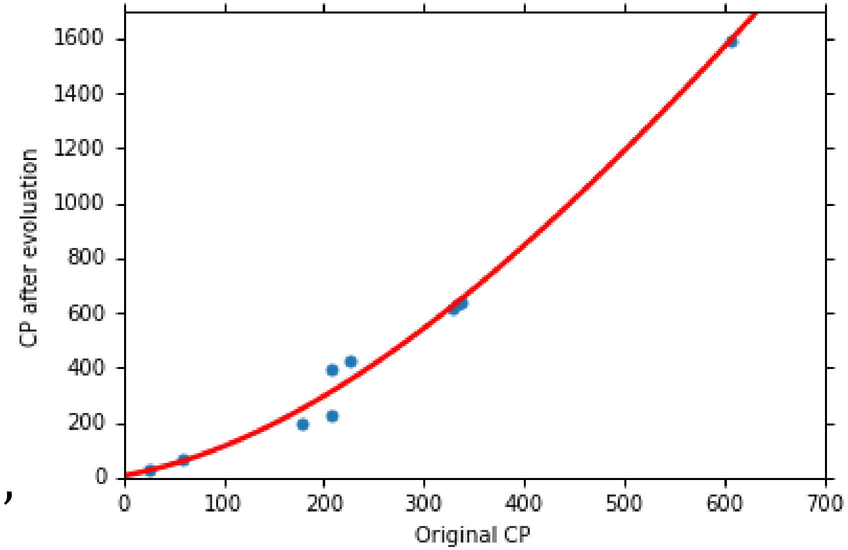
$b = 6.4, w_1 = 0.66, w_2 = 4.3 \times 10^{-3}$, $w_3 = 1.8 \times 10^{-6}$

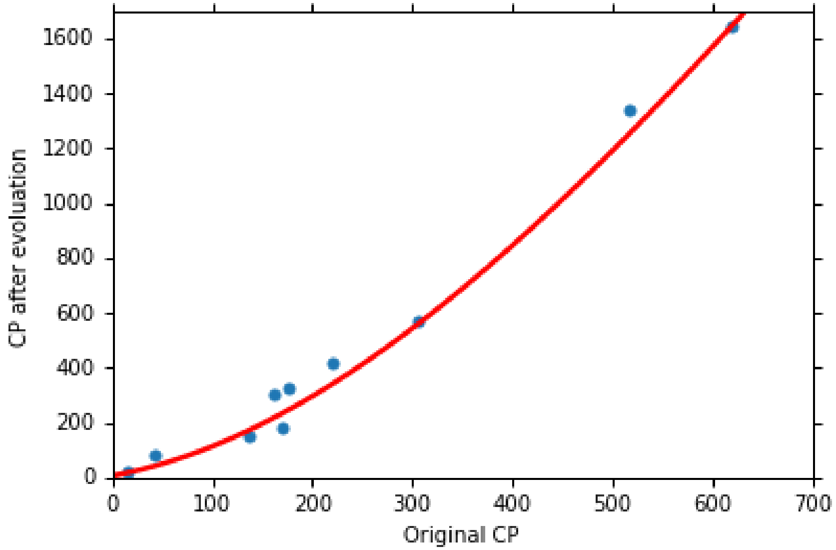
Average Error = 15.3

- Testing

Average Error = 18.1

Slightly better.
How about more complex model?



# Model Generalization



- Select another model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$

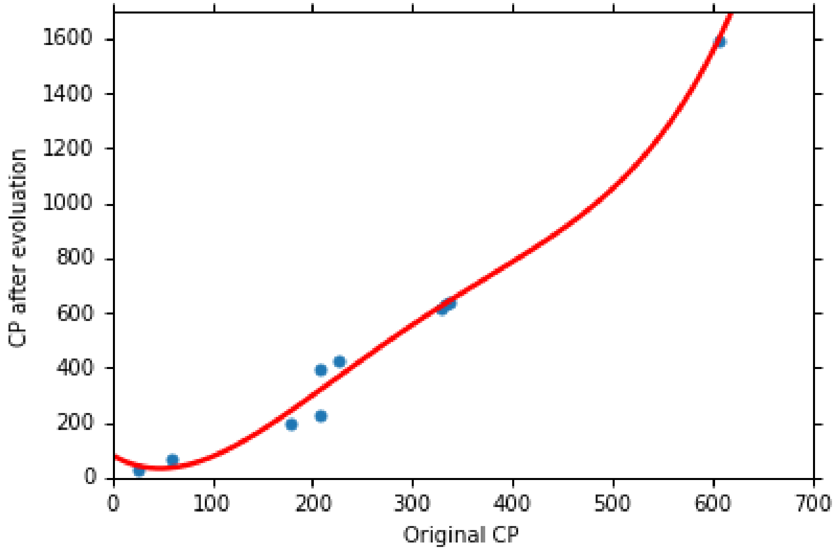
- Best function

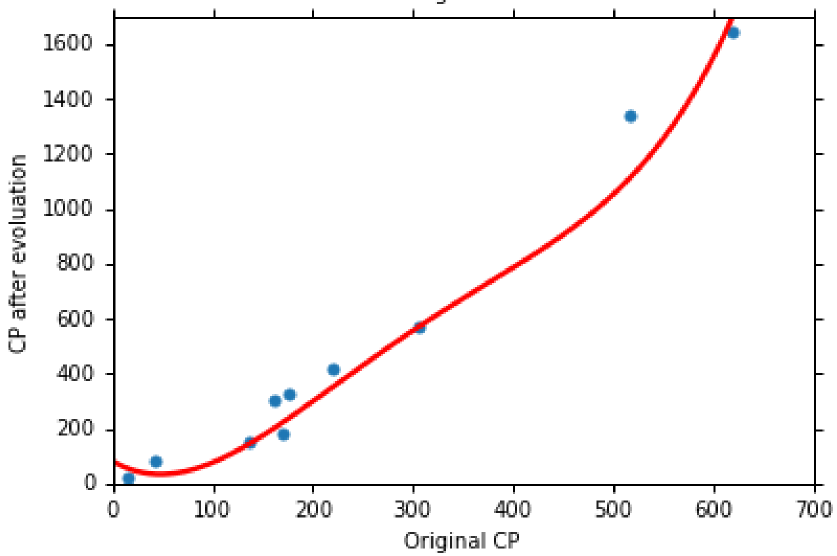
  Average Error = 14.9

- Testing

  Average Error = 28.8

The results become worse



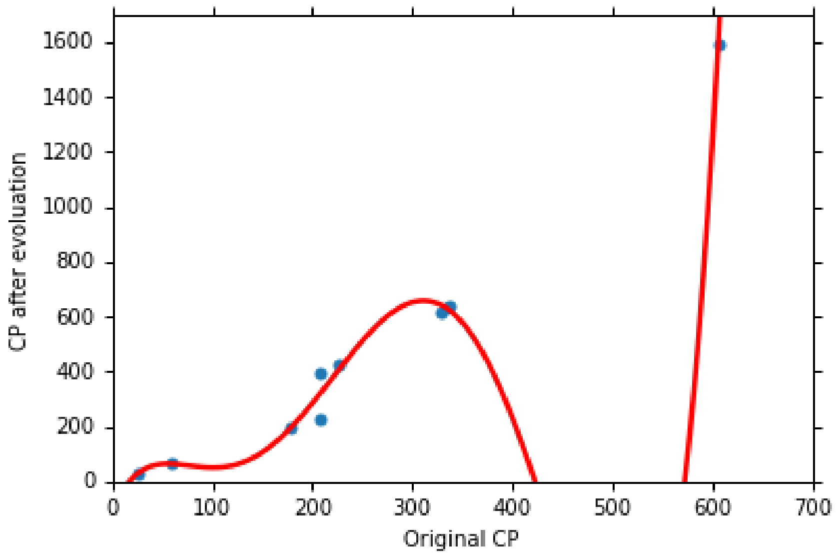



# Model Generalization



- Select another model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$$
$$+w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$$
$$+w_5 \cdot (x_{cp})^5$$

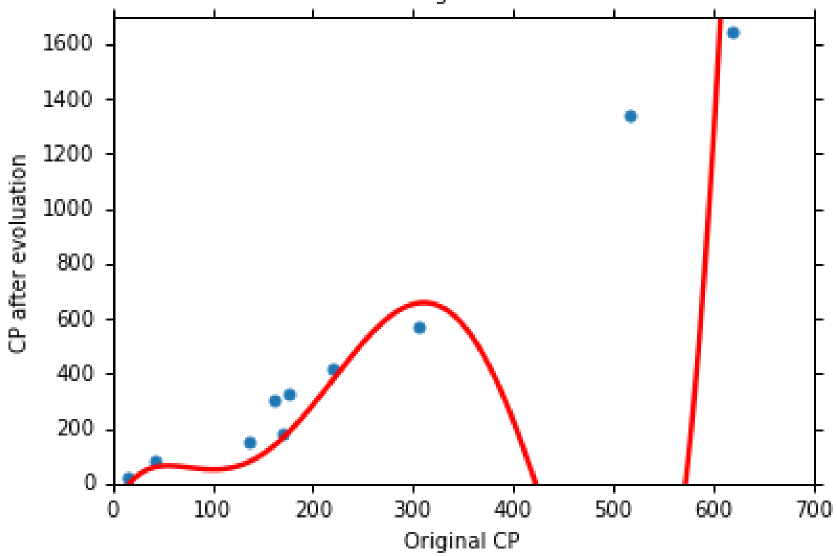



- Best function

Average Error = 12.8

- Testing

Average Error = 232.1

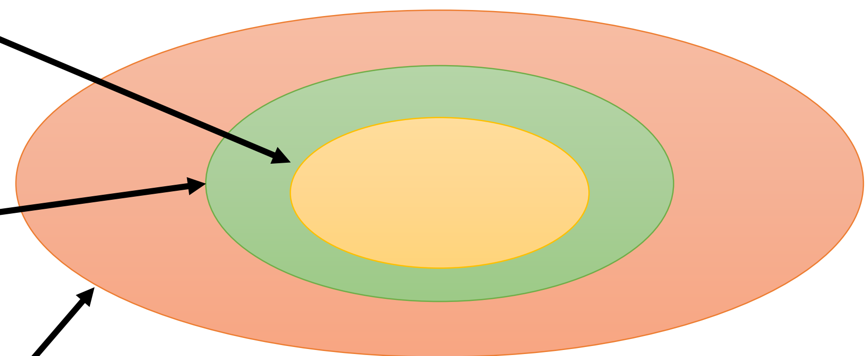



The results are so bad

# Model Selection



1. $y = b + w \cdot x_{cp}$
2. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2$
3. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3$
4. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4$
5. $y = b + w_1 \cdot x_{cp} + w_2 \cdot (x_{cp})^2 + w_3 \cdot (x_{cp})^3 + w_4 \cdot (x_{cp})^4 + w_5 \cdot (x_{cp})^5$

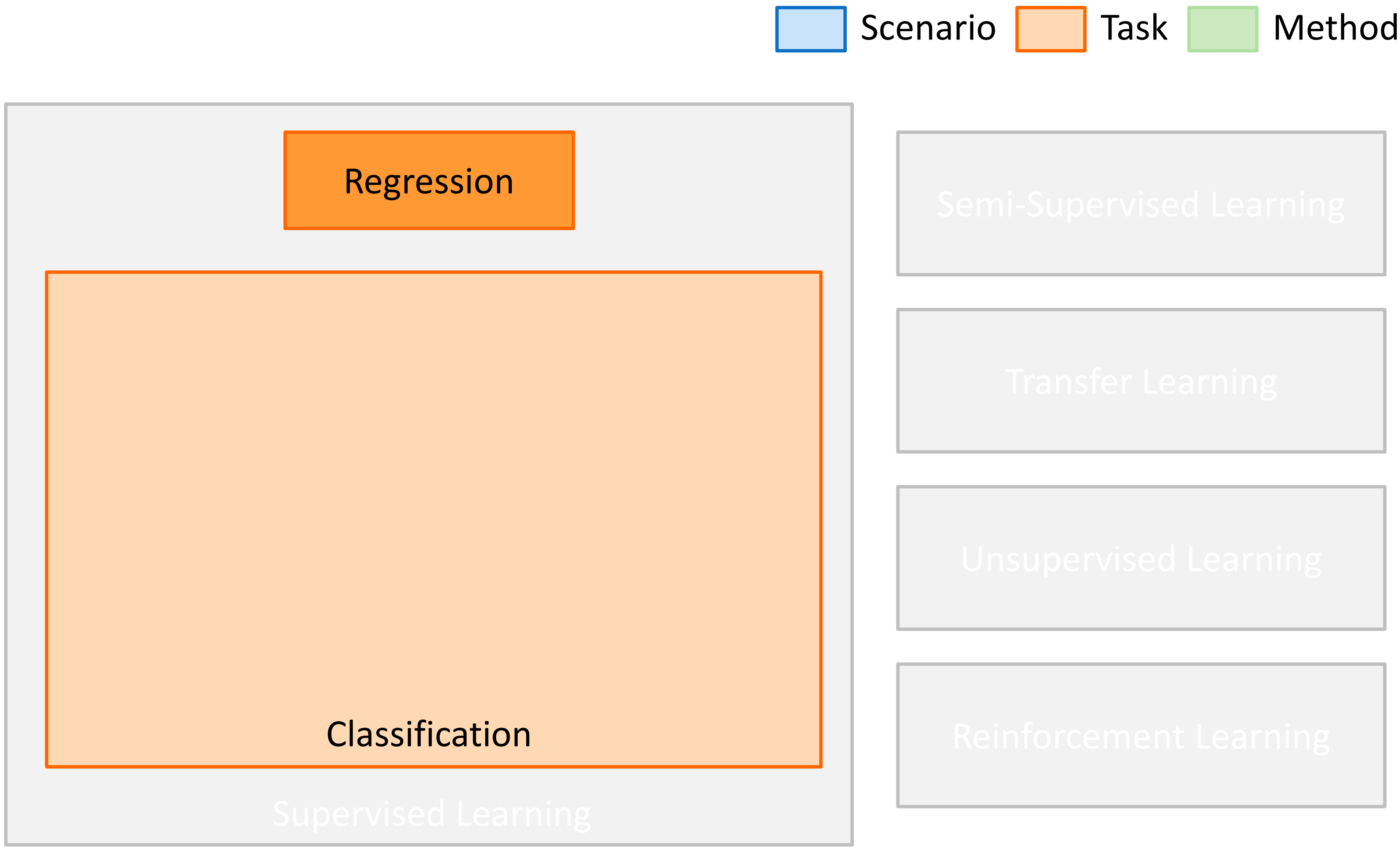



A more complex model yields lower error on training data.

If we can truly find the best function

# Machine Learning Map

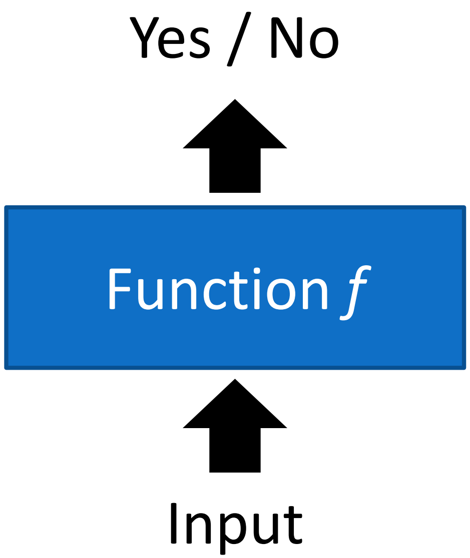




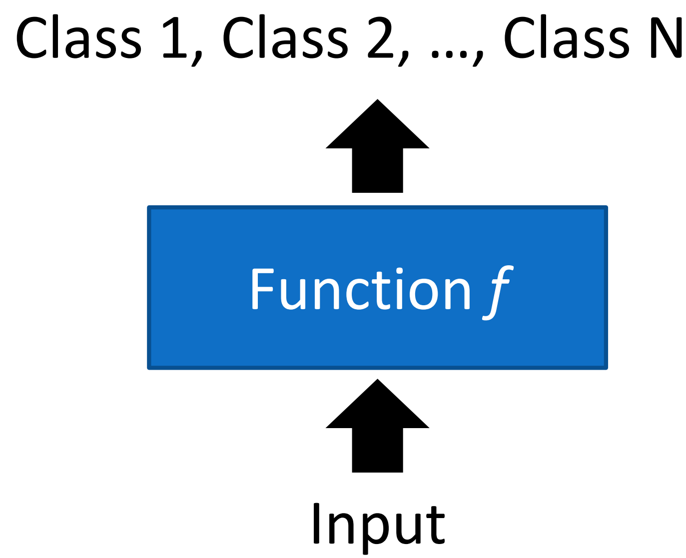
# Classification



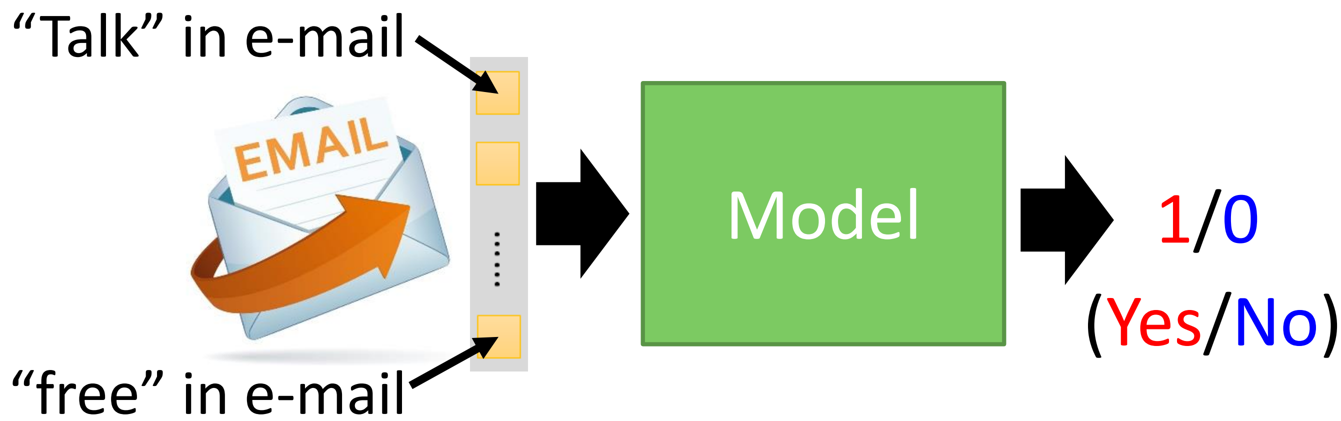
- Binary Classification




- Multi-Class Classification



# Binary Classification – Spam Filtering

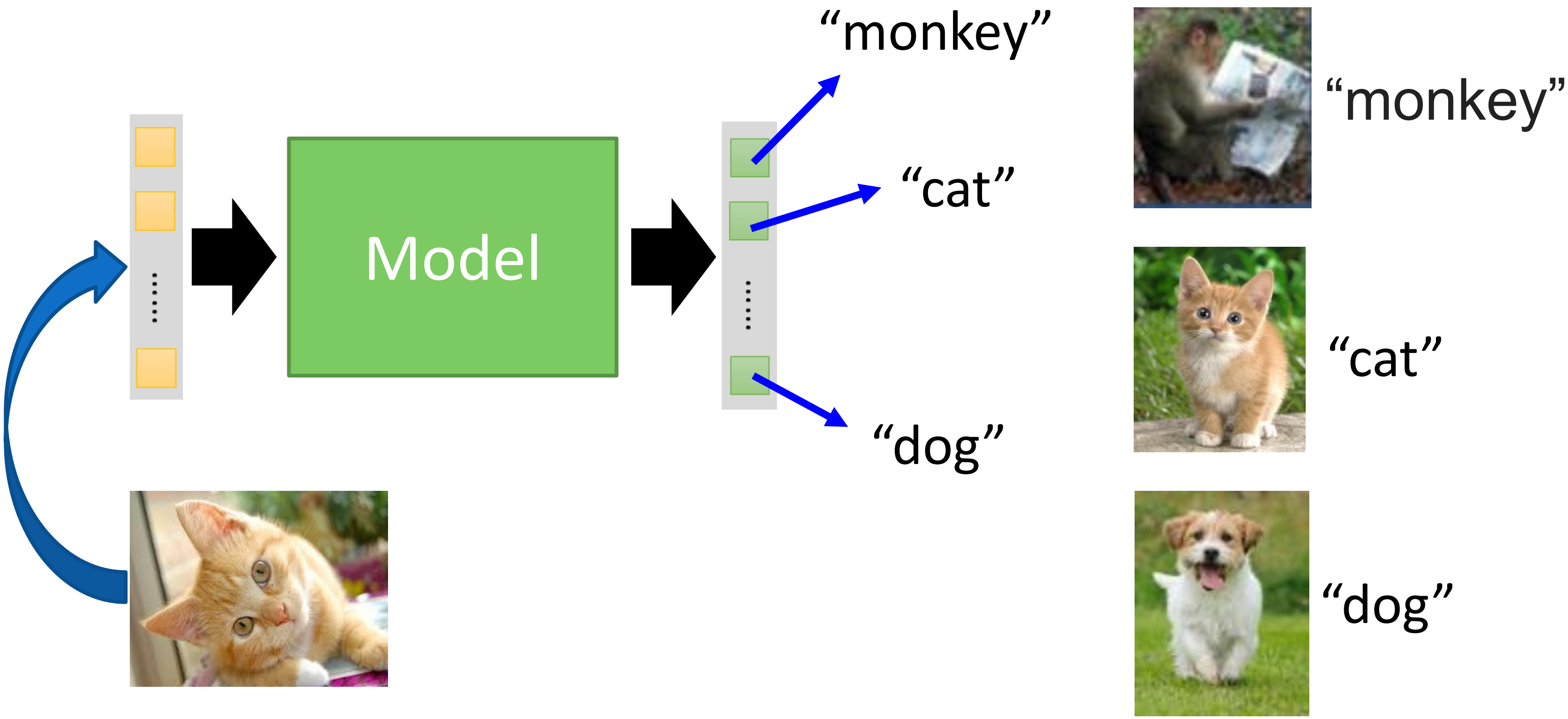




(http://spam-filter-review.toptenreviews.com/)

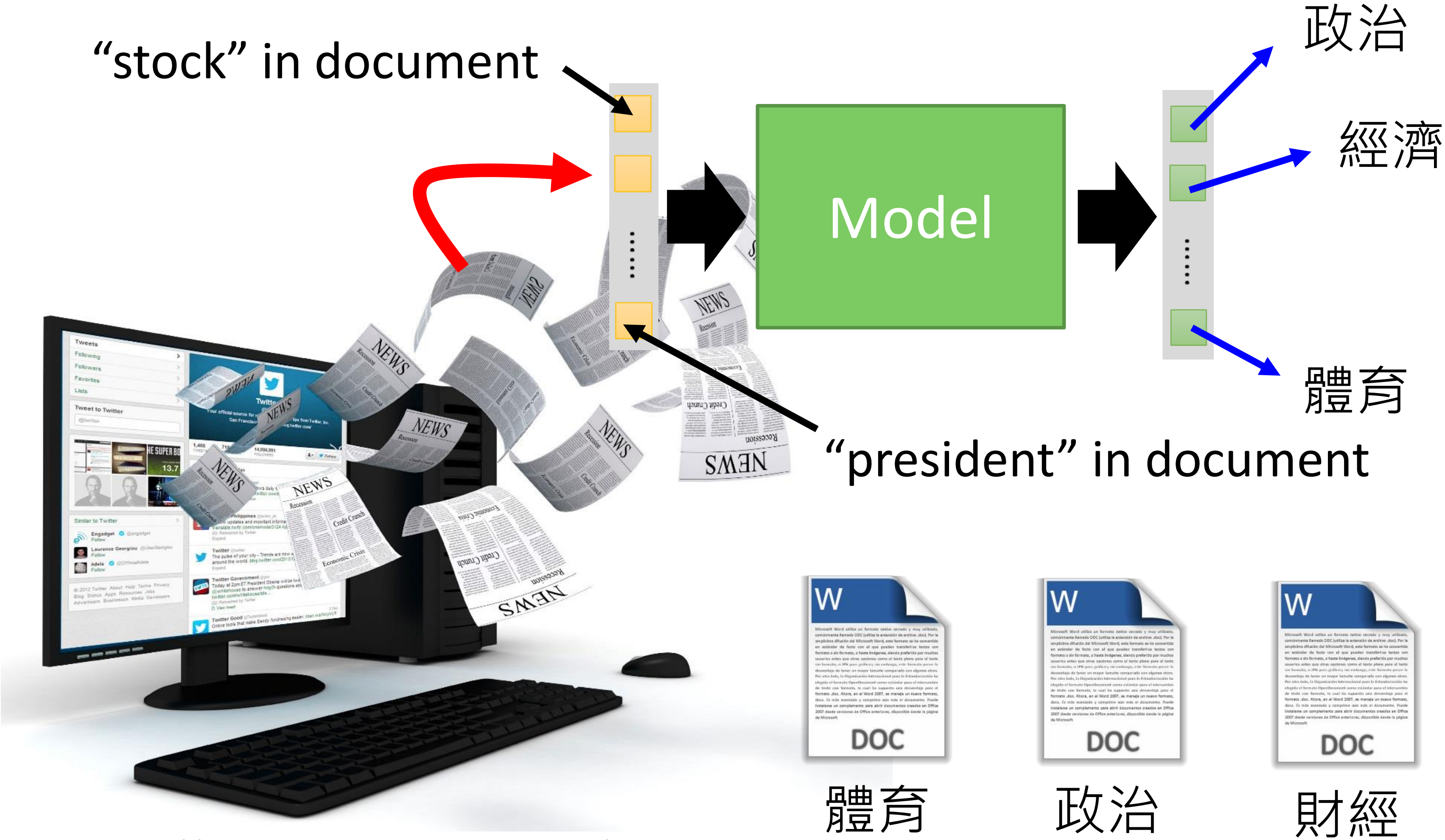
# Multi-Class – Image Recognition





# Multi-Class – Topic Classification

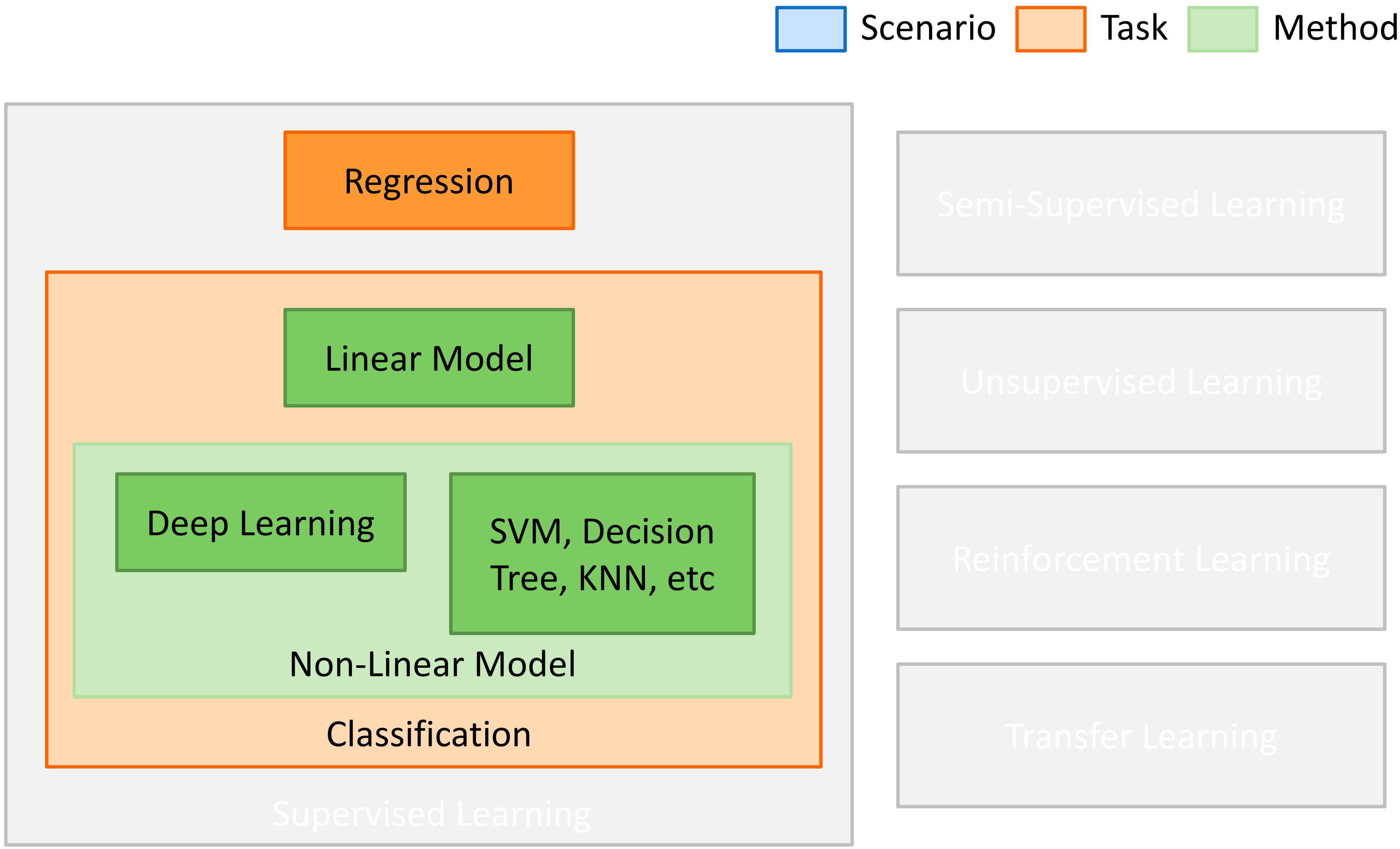





http://top-breaking-news.com/

# Machine Learning Map





# Part I: Introduction to ML & DL

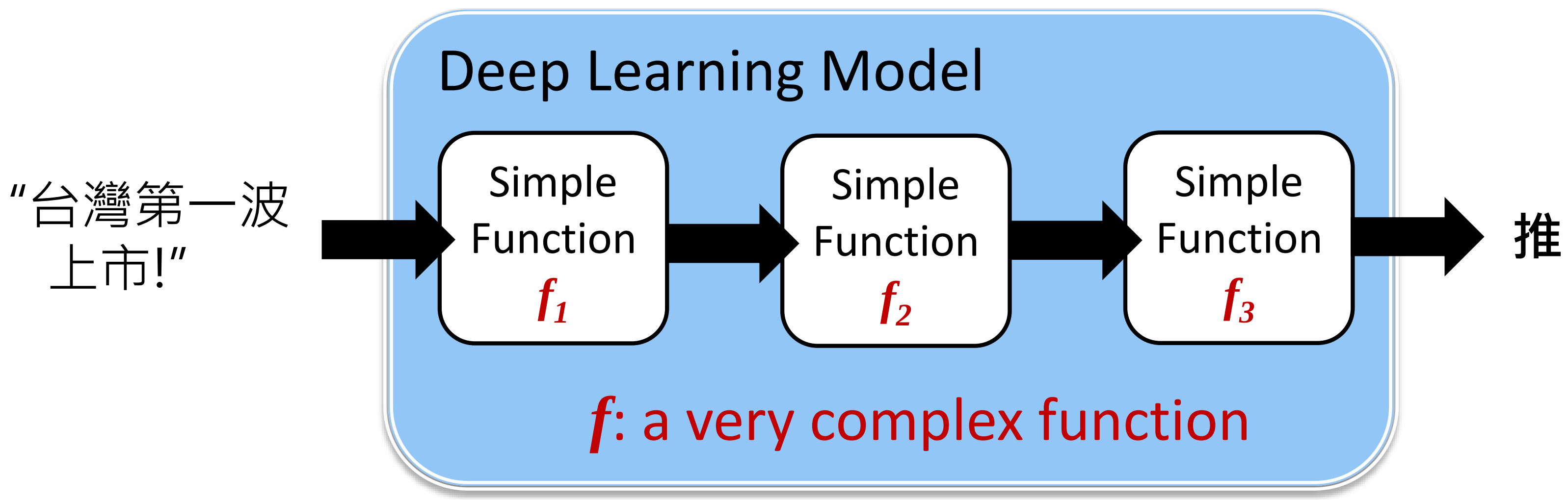


- Basic Machine Learning
- **Basic Deep Learning**
- Toolkits and Learning Recipe

# Stacked Functions Learned by Machine



- Production line (生產線)




End-to-end training: what each function should do is learned automatically

Deep learning usually refers to *neural network* based model

# Three Steps for Deep Learning



Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

# Three Steps for Deep Learning



Step 1: define a set of function

Neural Network

Step 2: goodness of function

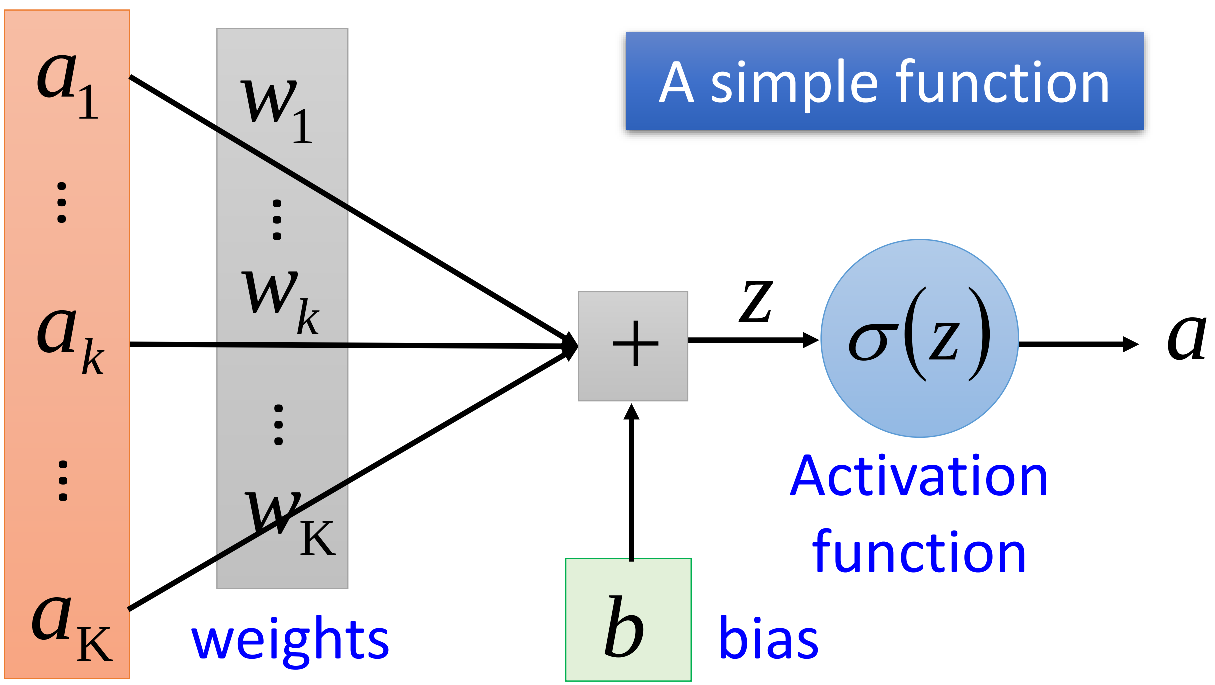
Step 3: pick the best function

# Neural Network



## ***Neuron***

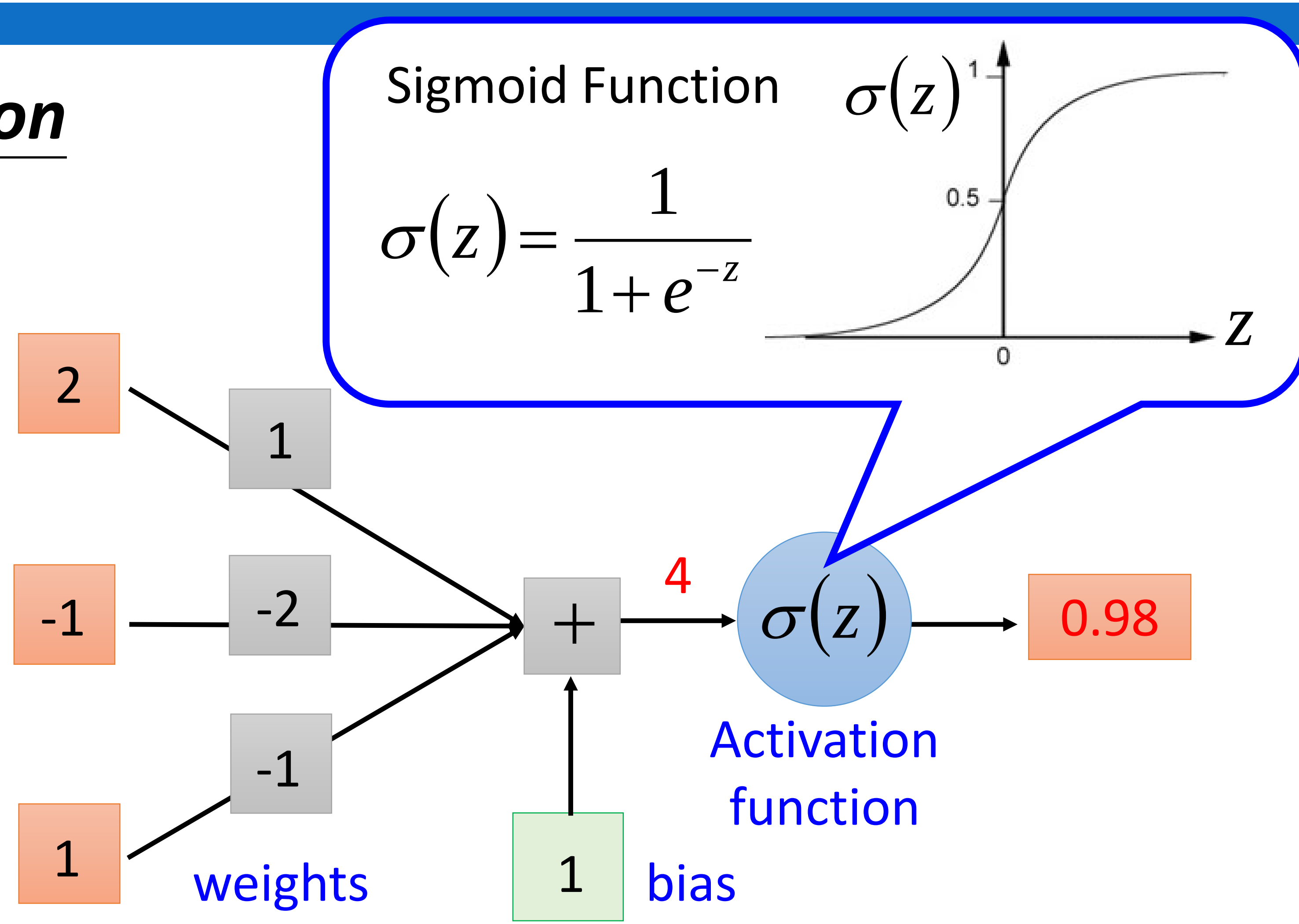
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$



# Neural Network

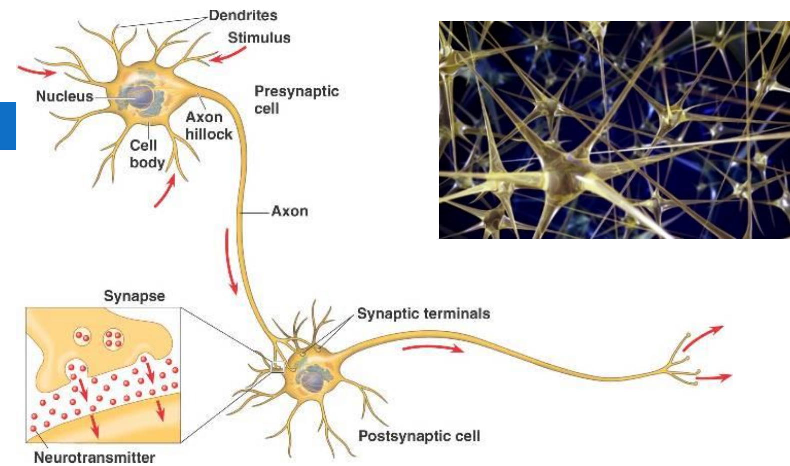


***Neuron***

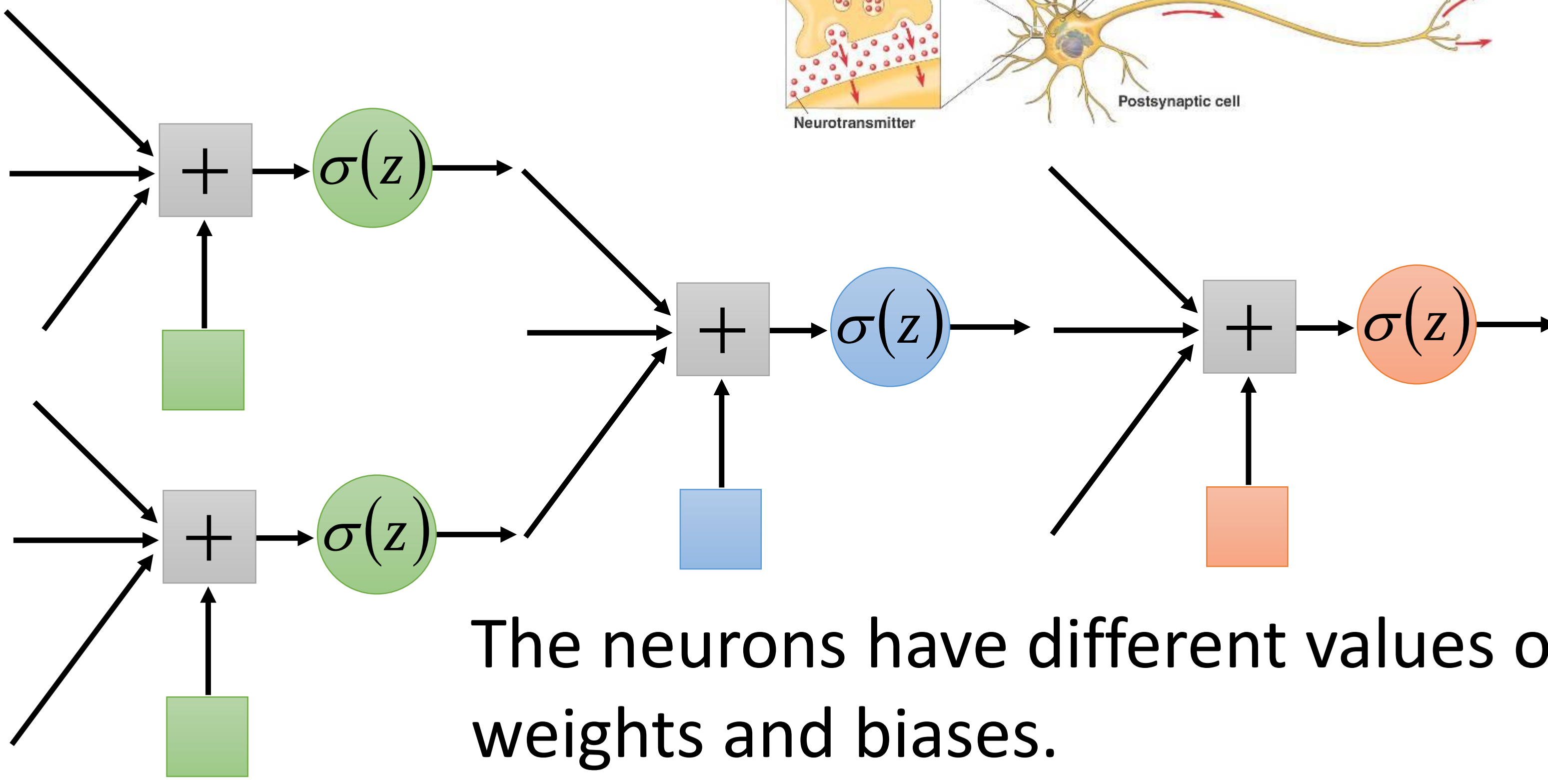


# Neural Network






Different connections lead to different network structures

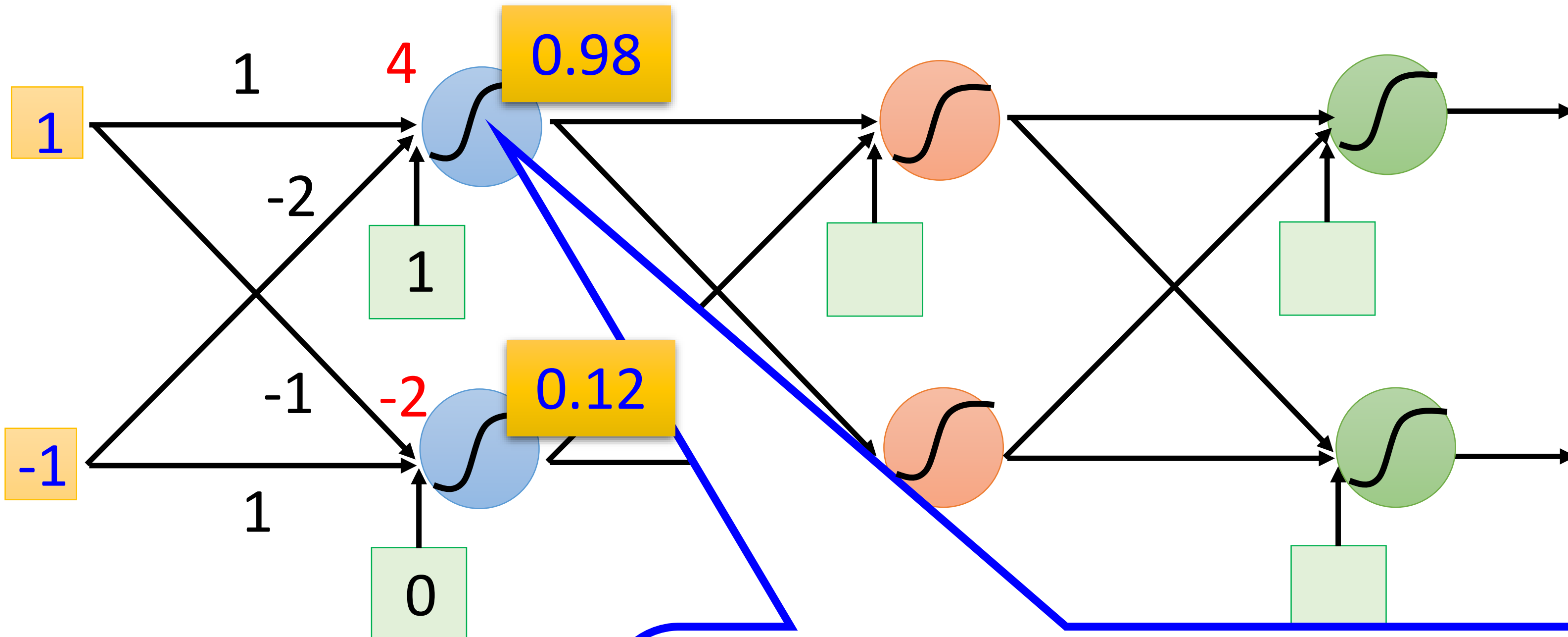



The neurons have different values of weights and biases.

Weights and biases are network parameters $\theta$

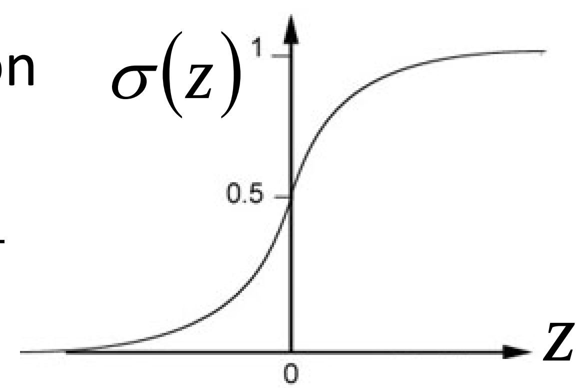
# Fully Connected Feedforward Network





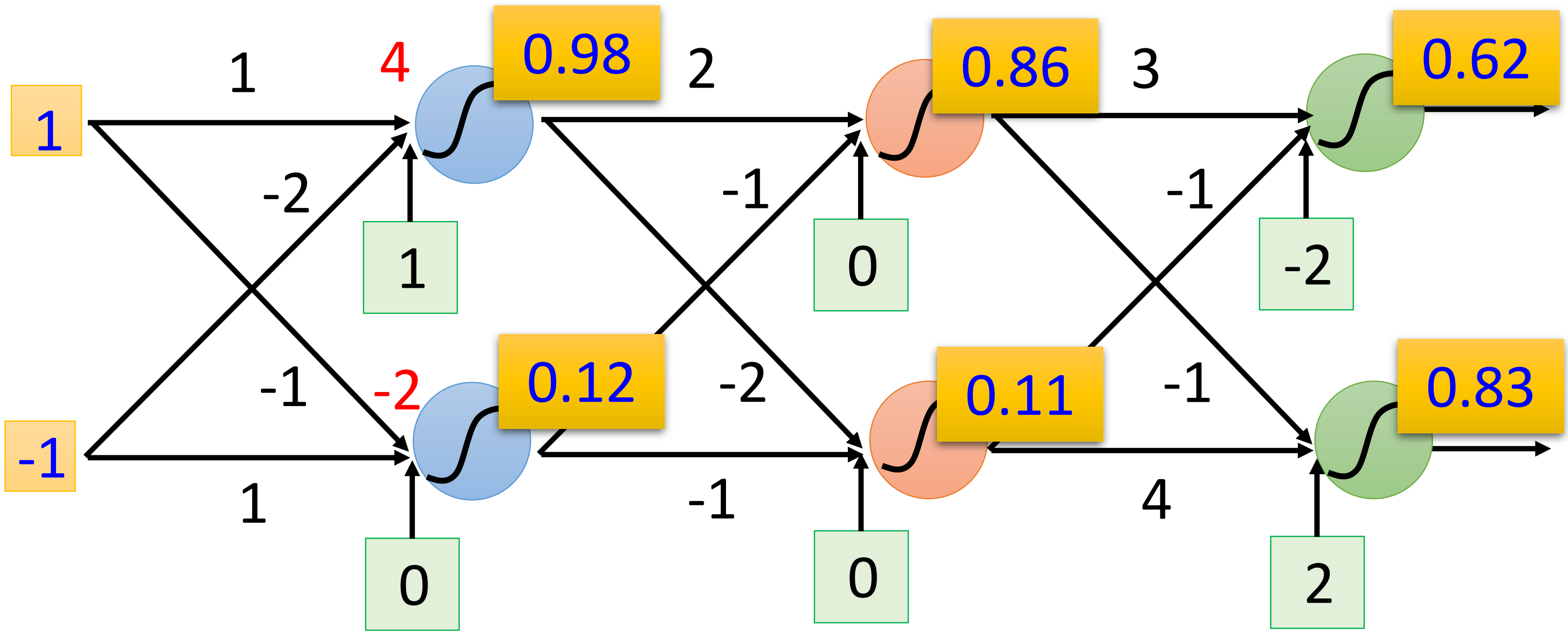

Sigmoid Function

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



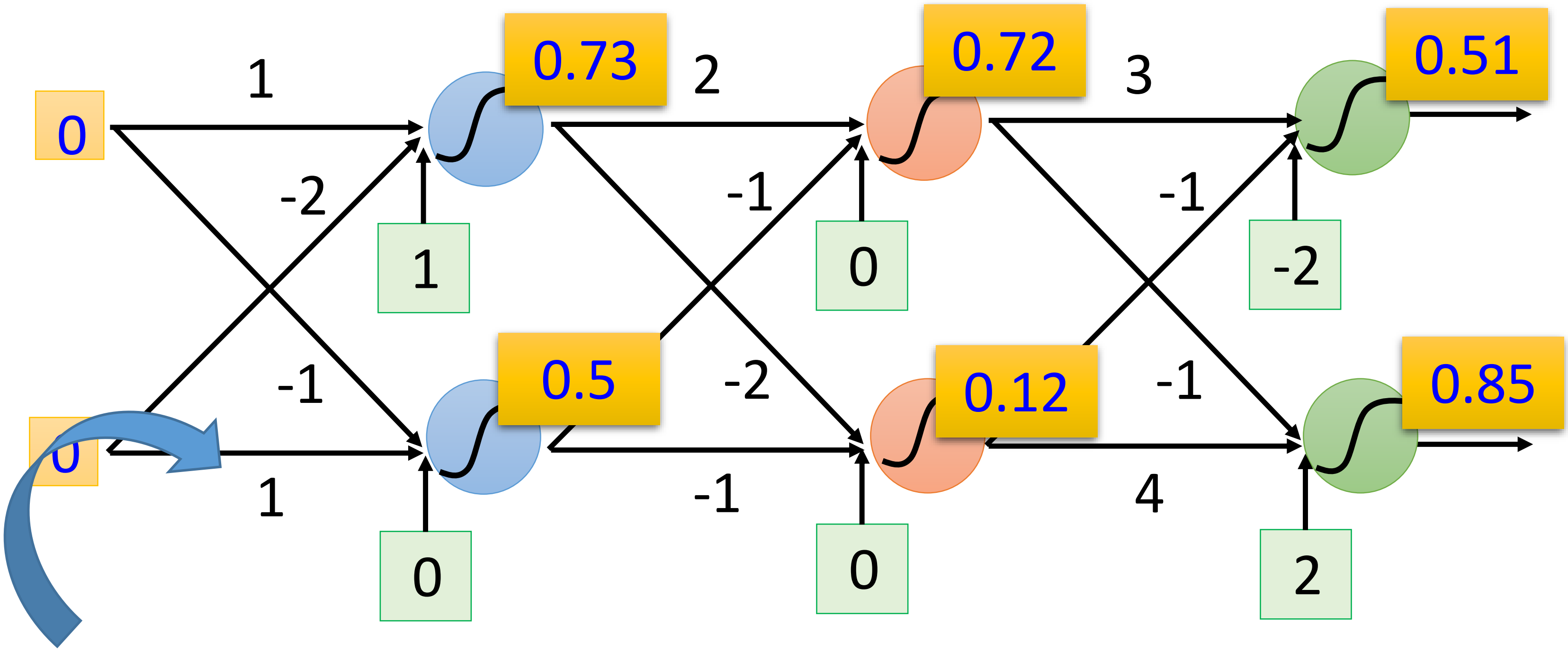
# Fully Connected Feedforward Network





# Fully Connected Feedforward Network






This is a function.

Input vector, output vector

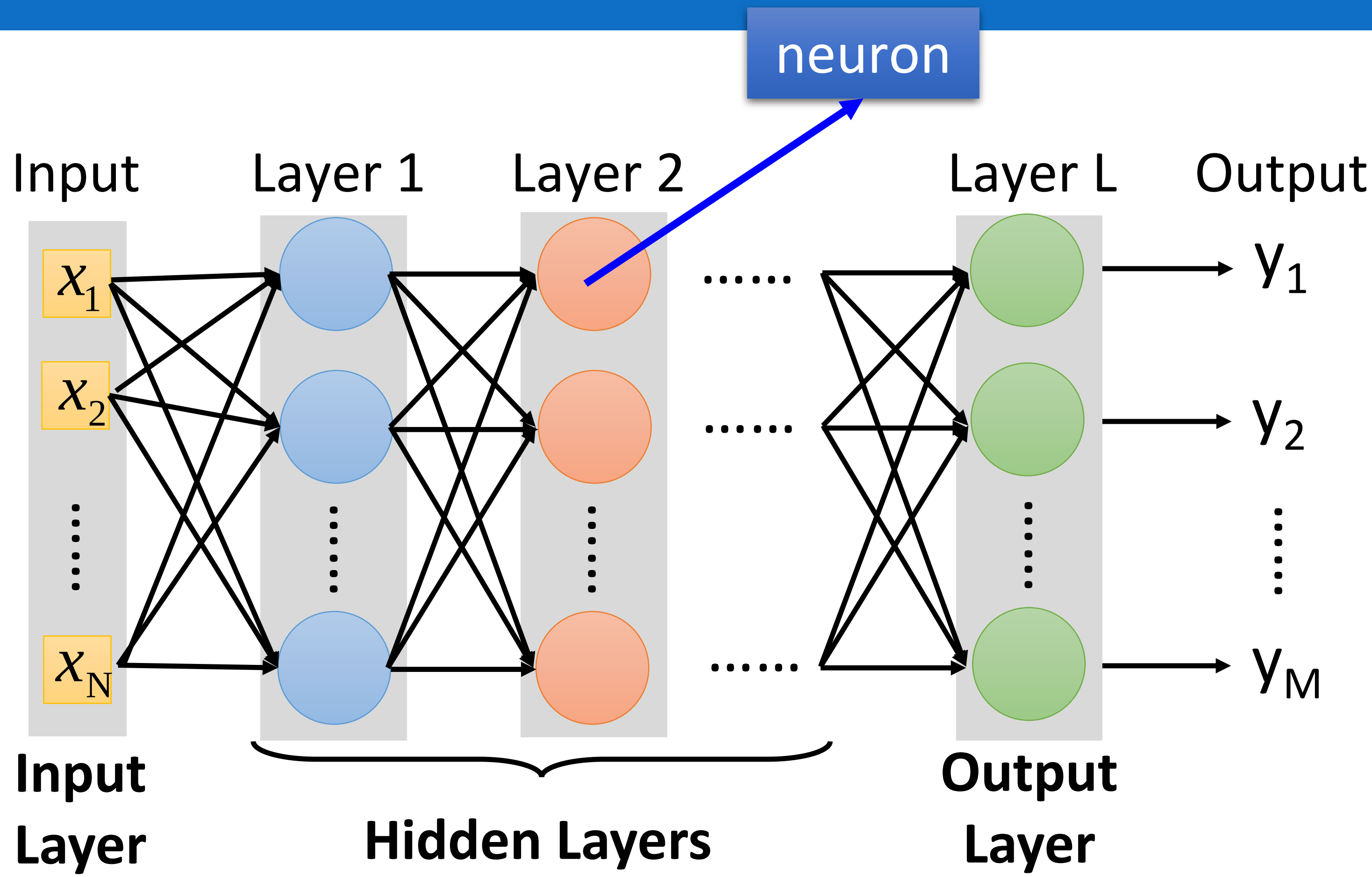
$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix}\quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Given parameters $\theta$, define a function

Given network structure, define ***a function set***

# Fully Connect Feedforward Network






Deep means many hidden layers

# Why Deep? Universality Theorem



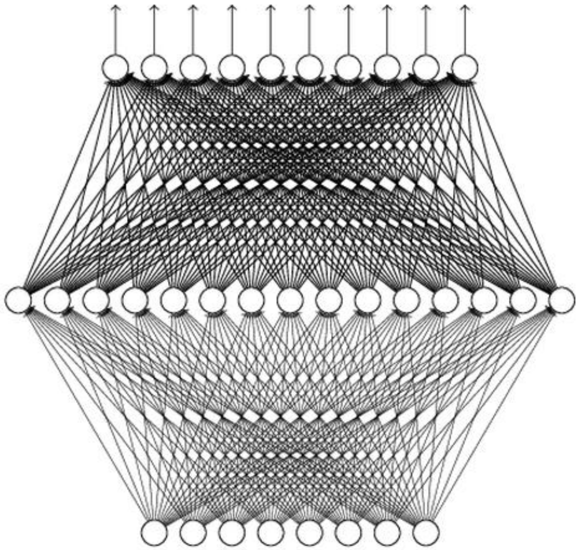
- Any continuous function $f$

$$f : R^N \rightarrow R^M$$

can be realized by a network with only hidden layer

  - (given **enough** hidden neurons)

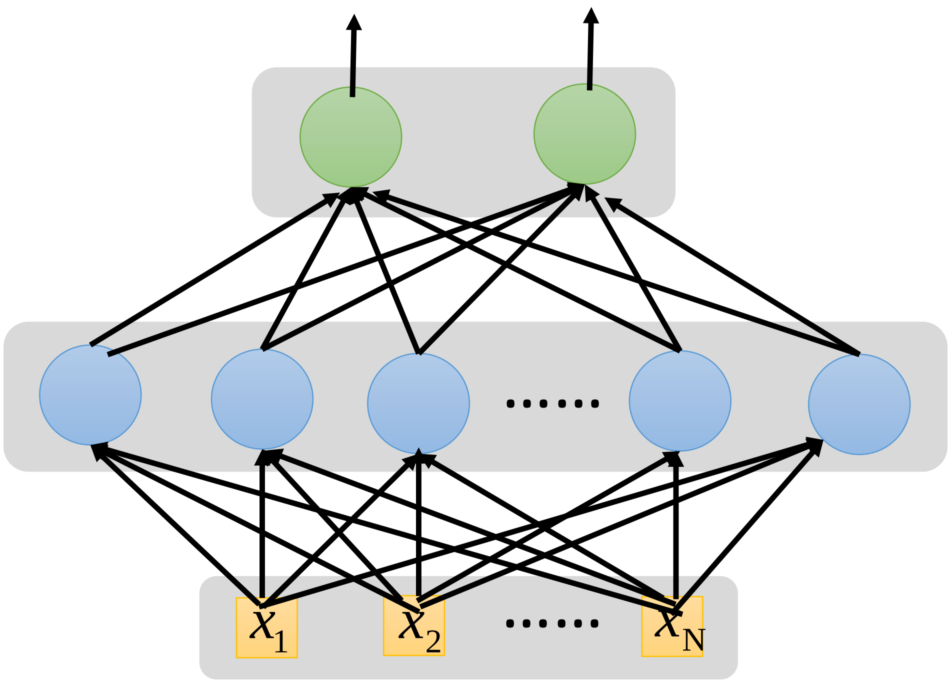
Why "deep" not "fat"?

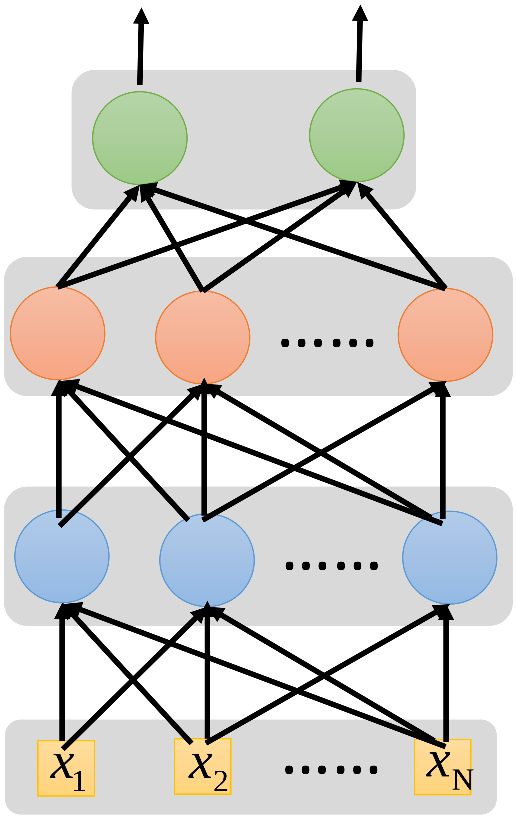


# Fat + Shallow v.s. Thin + Deep



- Two networks with the same number of parameters



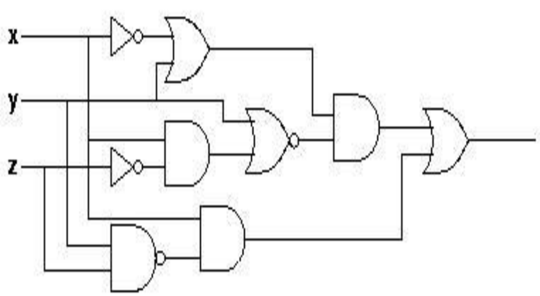



# Why Deep



- Logic circuits
  - Consists of **gates**
  - **A two layers of logic gates** can represent **any Boolean function.**
  - Using multiple layers of logic gates to build some functions are much simpler

→ less gates needed




- Neural network
  - consists of **neurons**
  - **A hidden layer network** can represent **any continuous function.**
  - Using multiple layers of neurons to represent some functions are much simpler

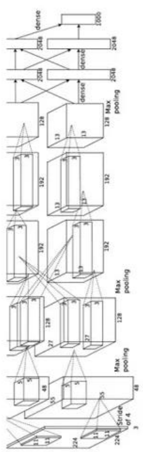
→ less parameters → less data?

# Deep = Many Hidden Layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers

16.4%



AlexNet (2012)

19 layers

7.3%

VGG (2014)

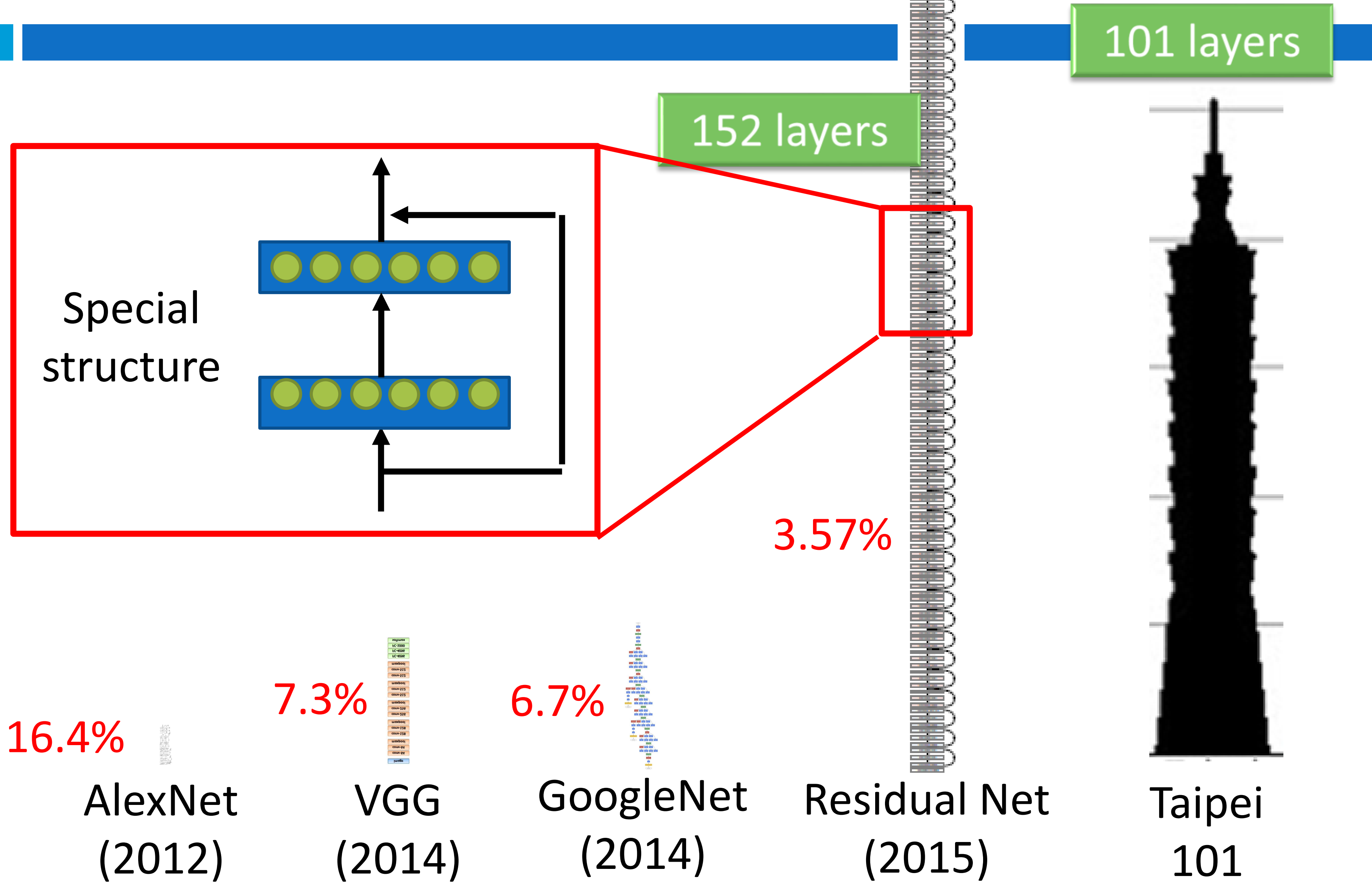
22 layers

6.7%

GoogleNet (2014)

# Deep = Many Hidden Layers



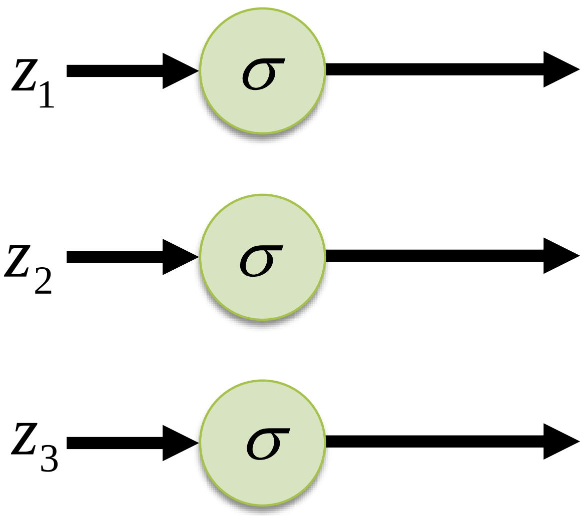


# Output Layer



- Softmax layer as the output layer

***<u>Ordinary Layer</u>***




$$y_1 = \sigma(z_1)$$

$$y_2 = \sigma(z_2)$$

$$y_3 = \sigma(z_3)$$

In general, the output of network can be any value.

May not be easy to interpret

# Output Layer

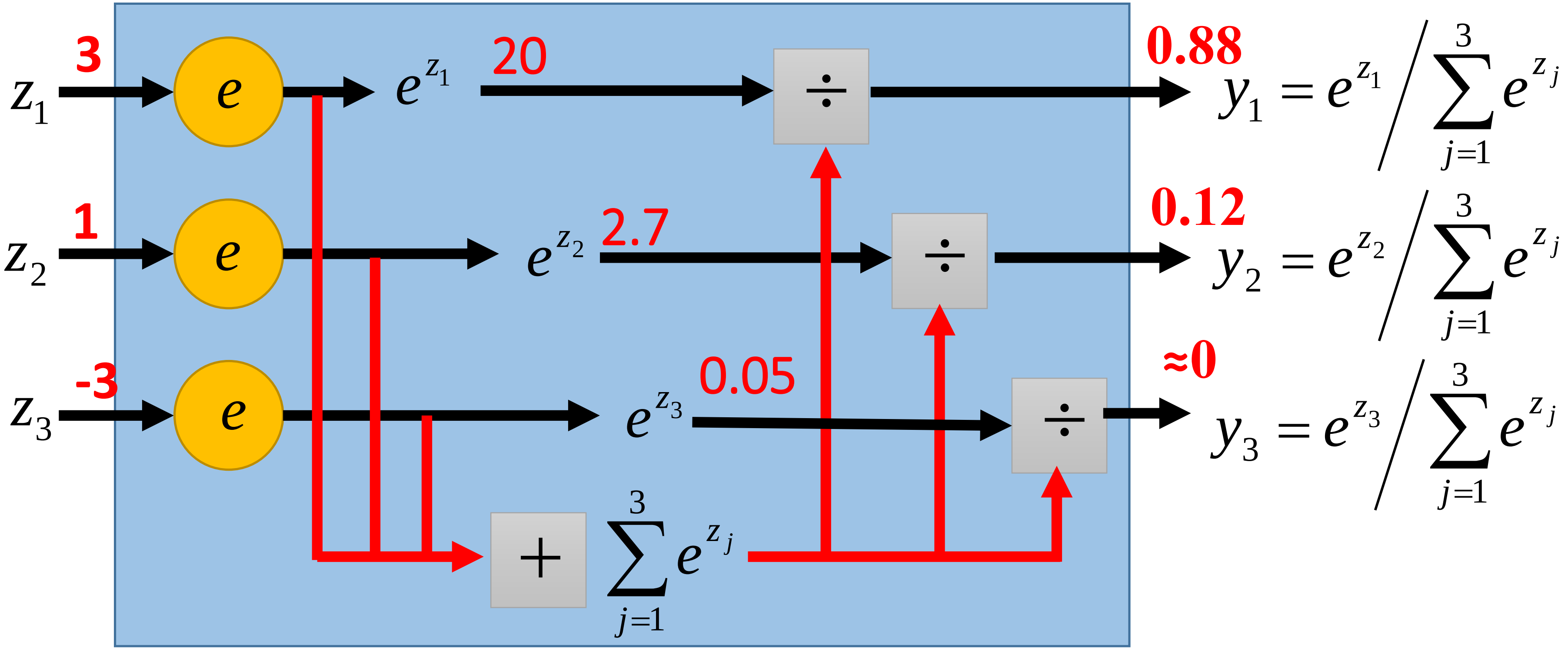


- Softmax layer as the output layer

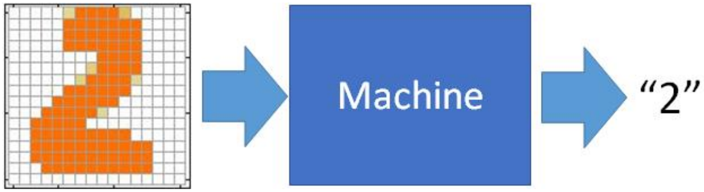
***Probability*:**

- $1 > y_i > 0$
- $\sum_i y_i = 1$

***Softmax Layer***

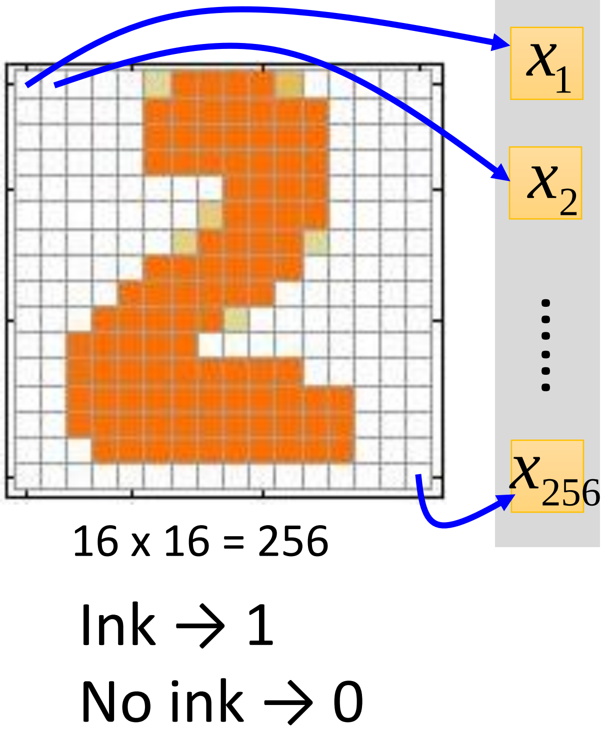


# Example Application

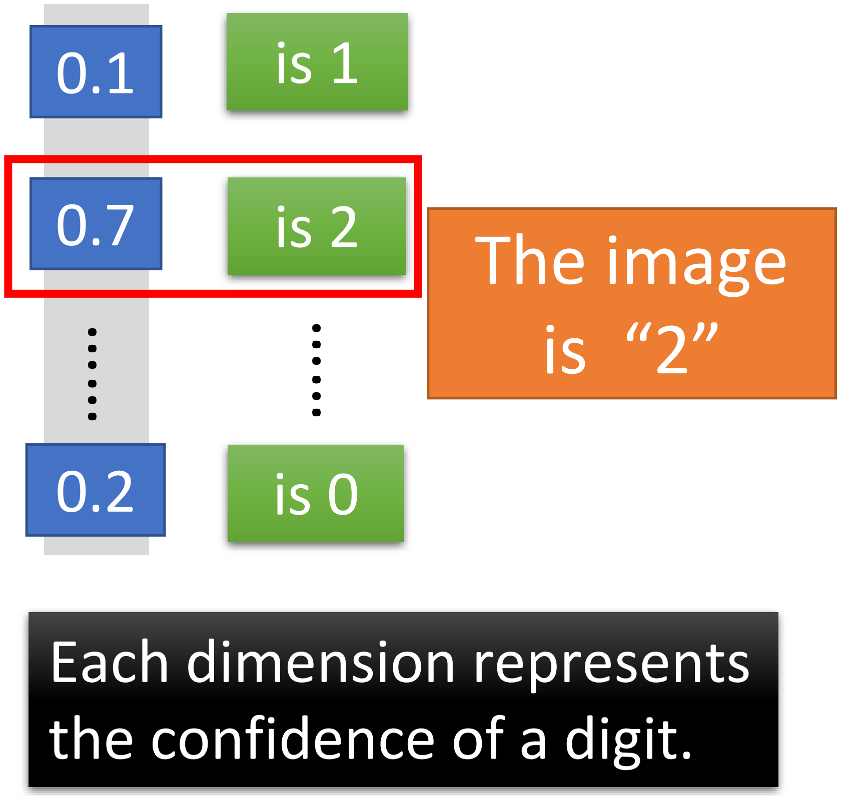





- **Input**



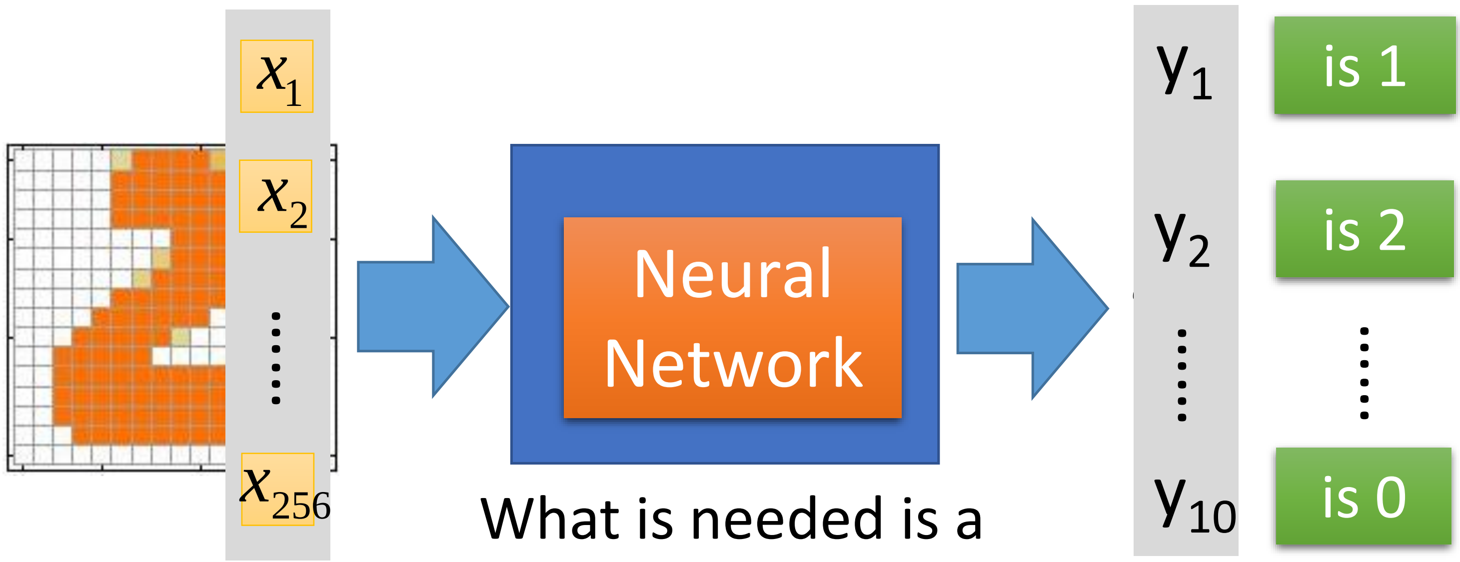

- **Output**



# Example Application



- Handwriting Digit Recognition



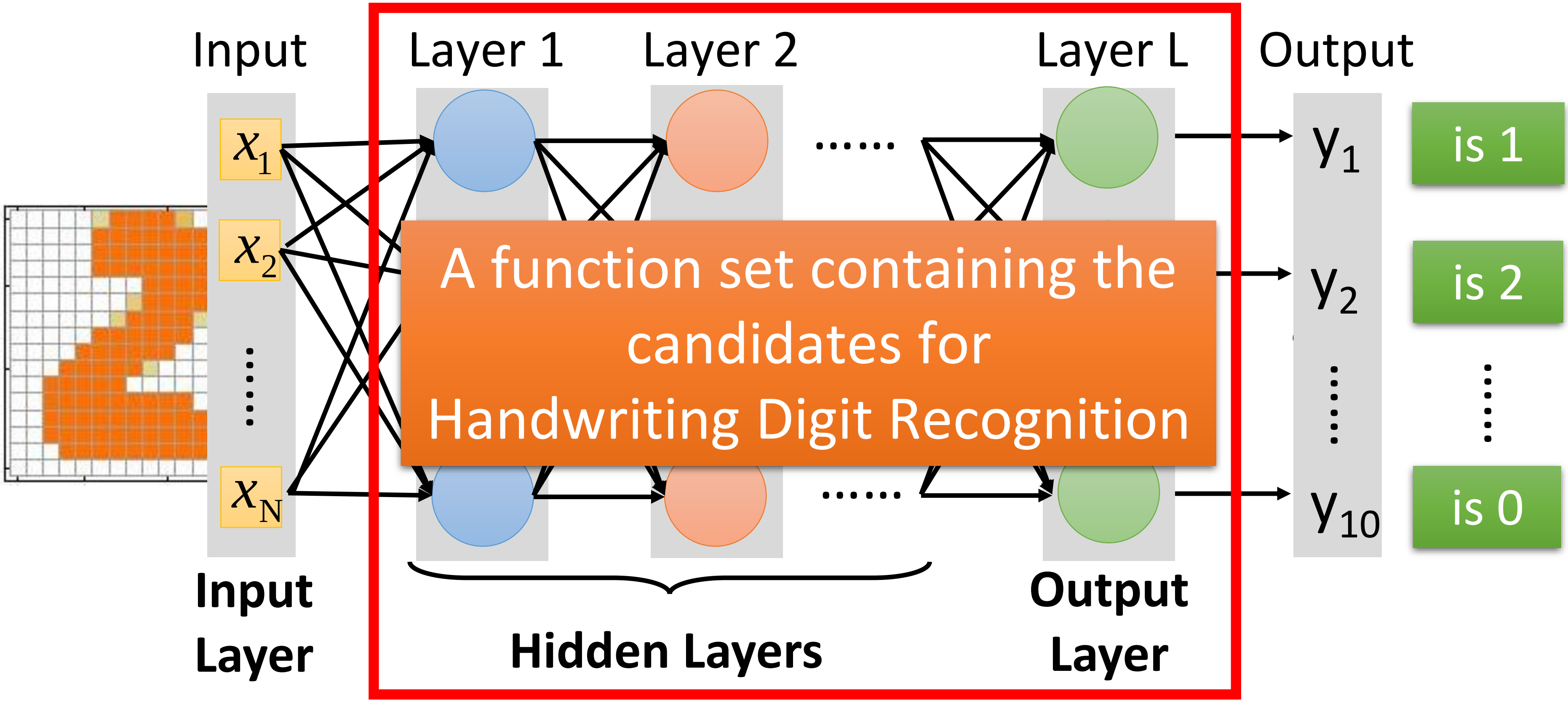

What is needed is a function ……

Input:
256-dim vector

output:
10-dim vector

# Example Application






You need to decide the network structure to let a good function in your function set.

# FAQ



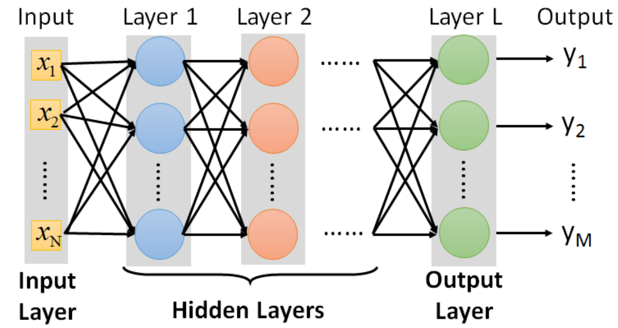
- Q: How many layers? How many neurons for each layer?

  Trial and Error + Intuition

- Q: Can we design the network structure?

  Variants of Neural Networks (next lecture)

- Q: Can the structure be automatically determined?
  - Yes, but not widely studied yet.



# Three Steps for Deep Learning



Step 1: define a set of function

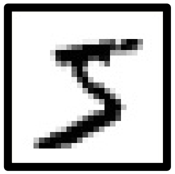
Step 2: goodness of function

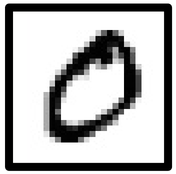
Step 3: pick the best function

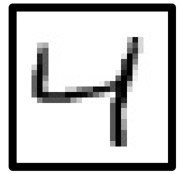
# Training Data

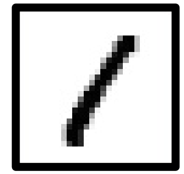


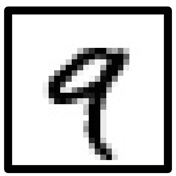
- Preparing training data: images and their labels

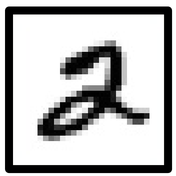

"5"

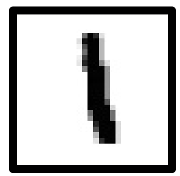

"0"

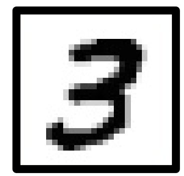

"4"


"1"


"9"


"2"

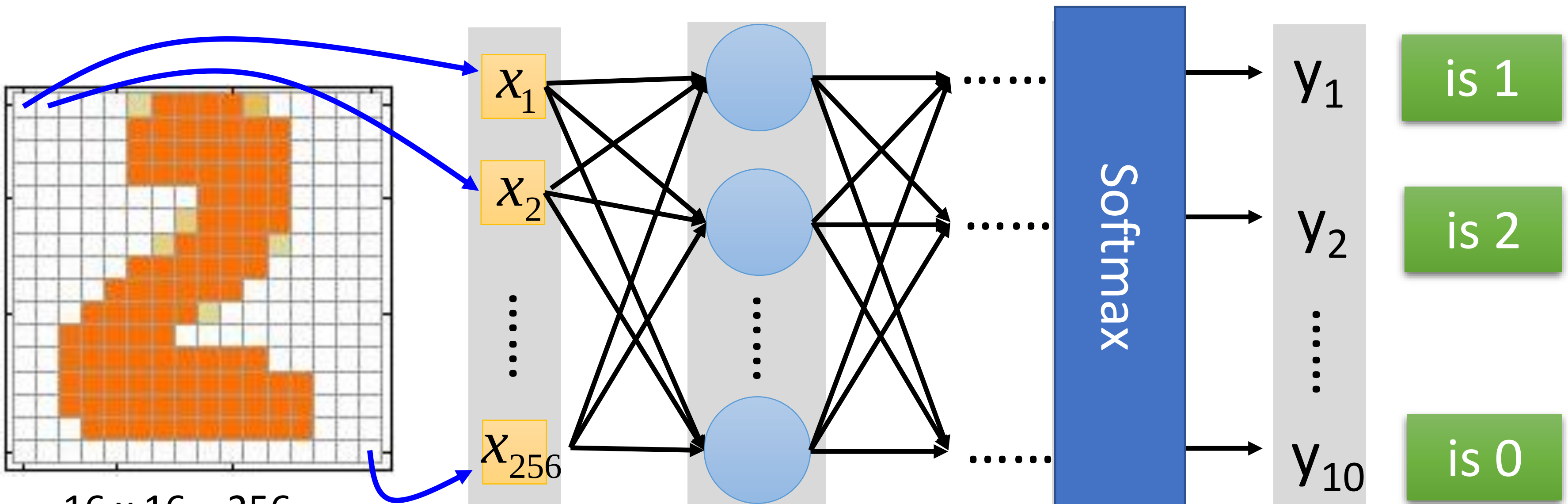

"1"


"3"

The learning target is defined on the training data.

# Learning Target



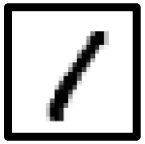



16 x 16 = 256

Ink → 1

No ink → 0

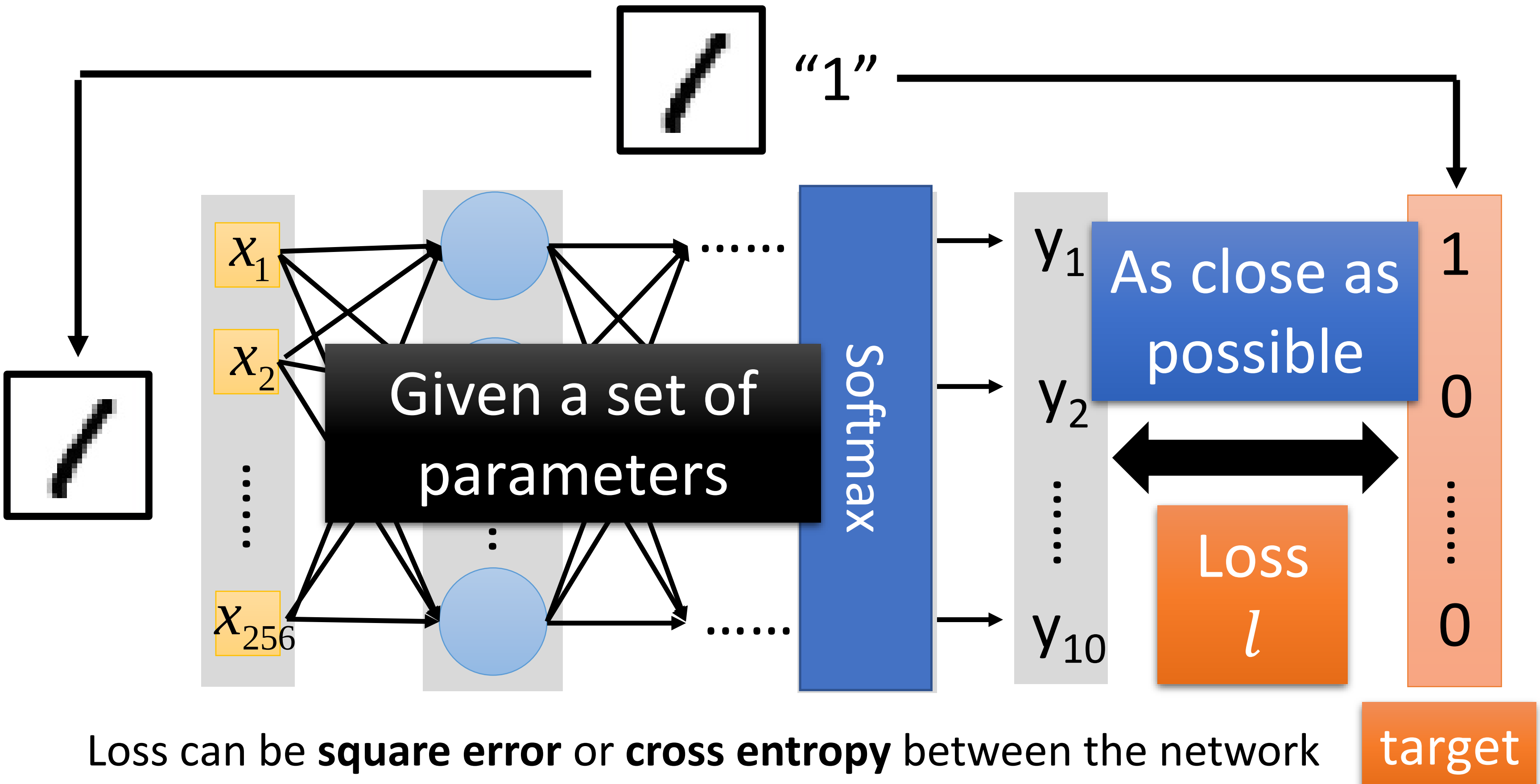
The learning target is ......

Input:  → $y_1$ has the maximum value

Input:  → $y_2$ has the maximum value

# Loss





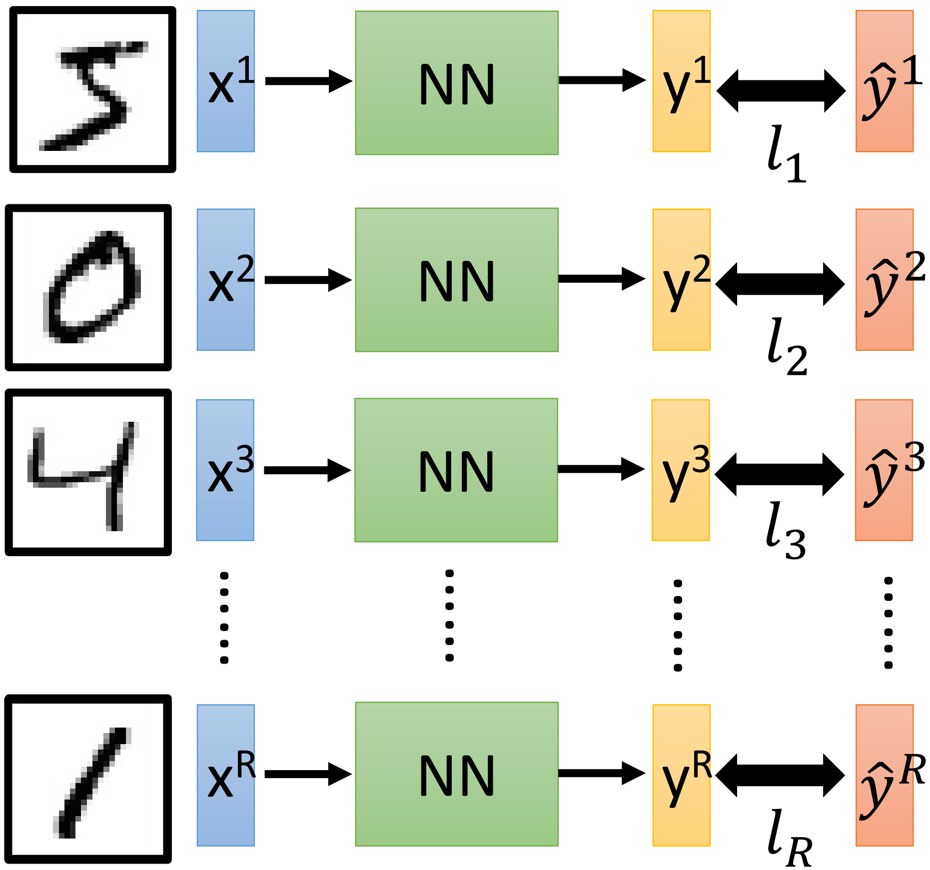

Loss can be **square error** or **cross entropy** between the network output and target

A good function should make the loss of all examples as small as possible.

# Total Loss



- For all training data …




Total Loss:

$$\boxed{L = \sum_{r=1}^{R} l_r}$$

As small as possible

Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters $\theta^*$*** that minimize total loss L

# Three Steps for Deep Learning



Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

# How to pick the best function



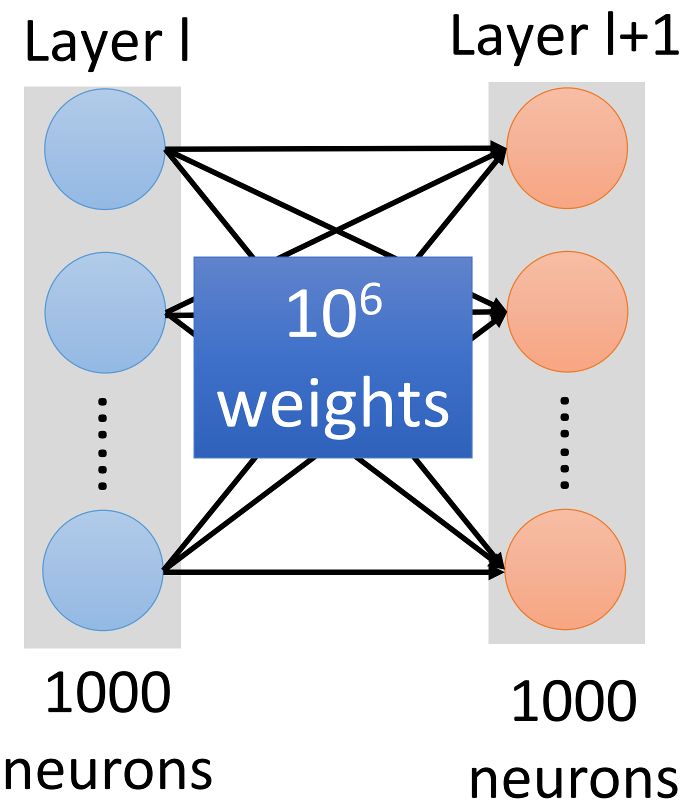
Find ***network parameters $\theta^*$*** that minimize total loss L

Enumerate all possible values

Networ[illegible]rame[illegible]s $\theta =$
$\{w_1, w_2, w_3$[illegible]$_1, b_2, b_3, \cdots\}$

Millio[illegible]r p[illegible]meters

E.g. speech recognition: 8 layers and 1000 neurons each layer



# Gradient Descent

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

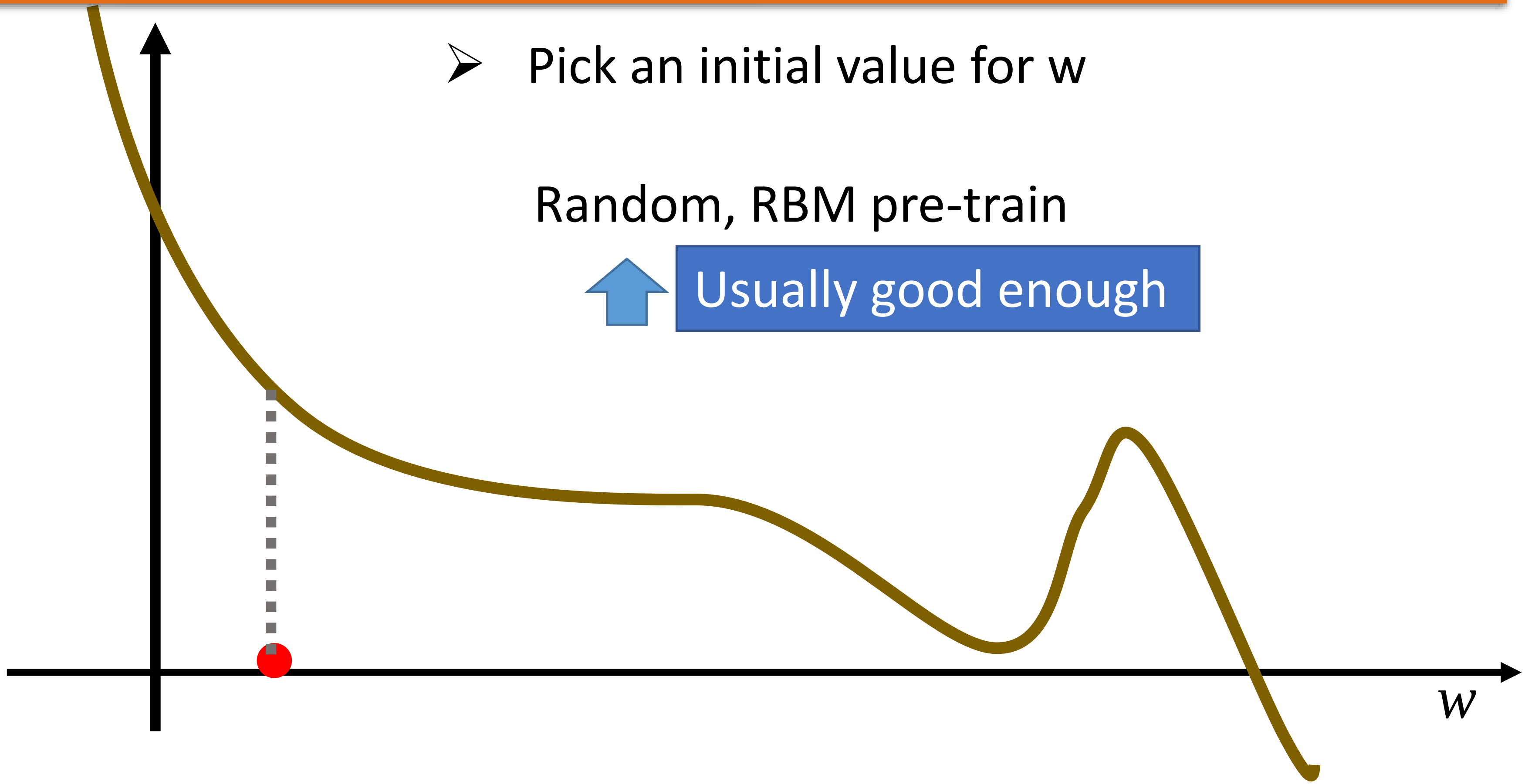


Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L

- Pick an initial value for w

Random, RBM pre-train

Usually good enough



# Gradient Descent

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

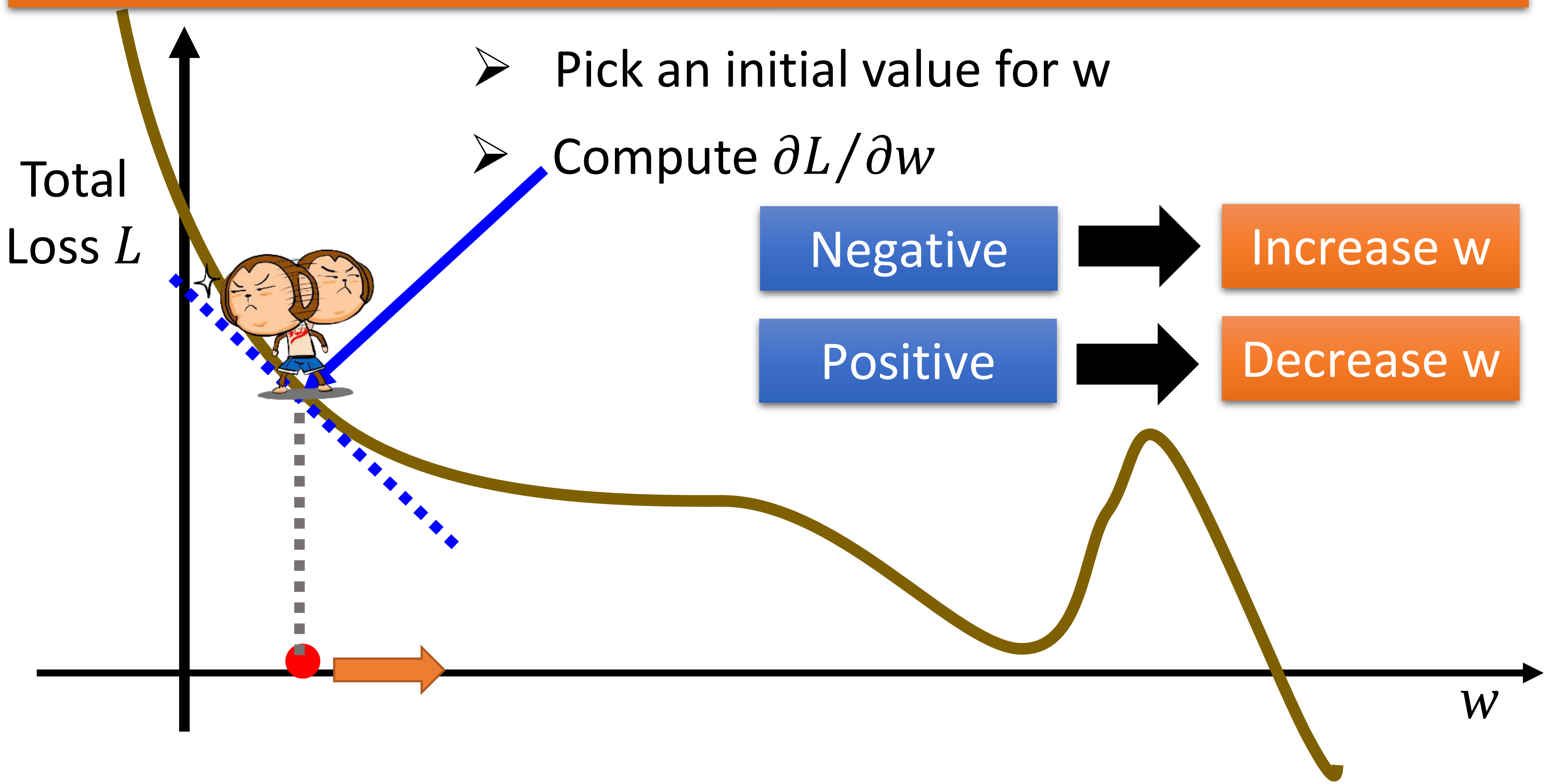


Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L

- Pick an initial value for w
- Compute $\partial L / \partial w$

Negative → Increase w

Positive → Decrease w

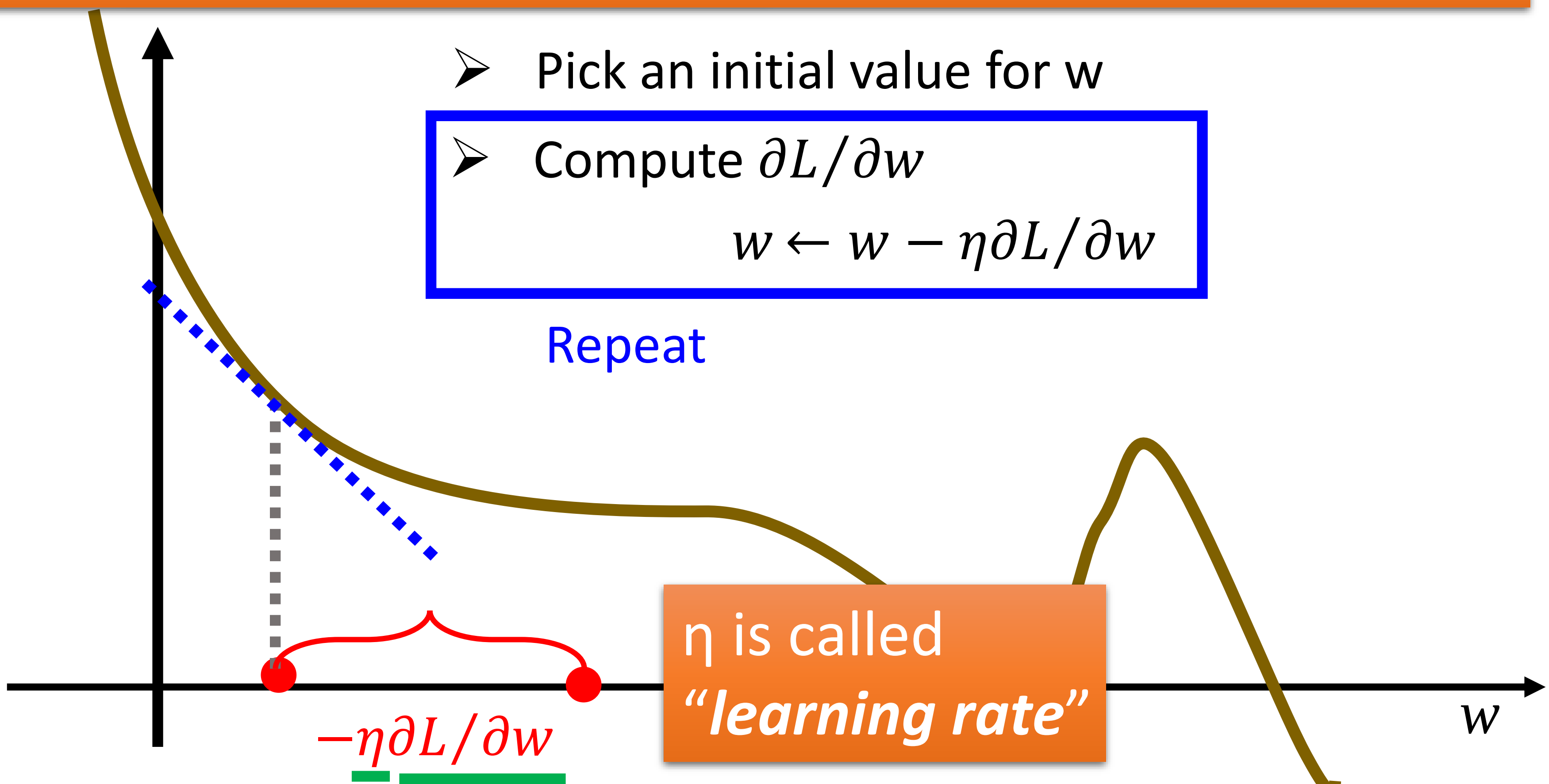


# Gradient Descent

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



Find ***network parameters*** $\theta^*$ that minimize total loss L



# Gradient Descent

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



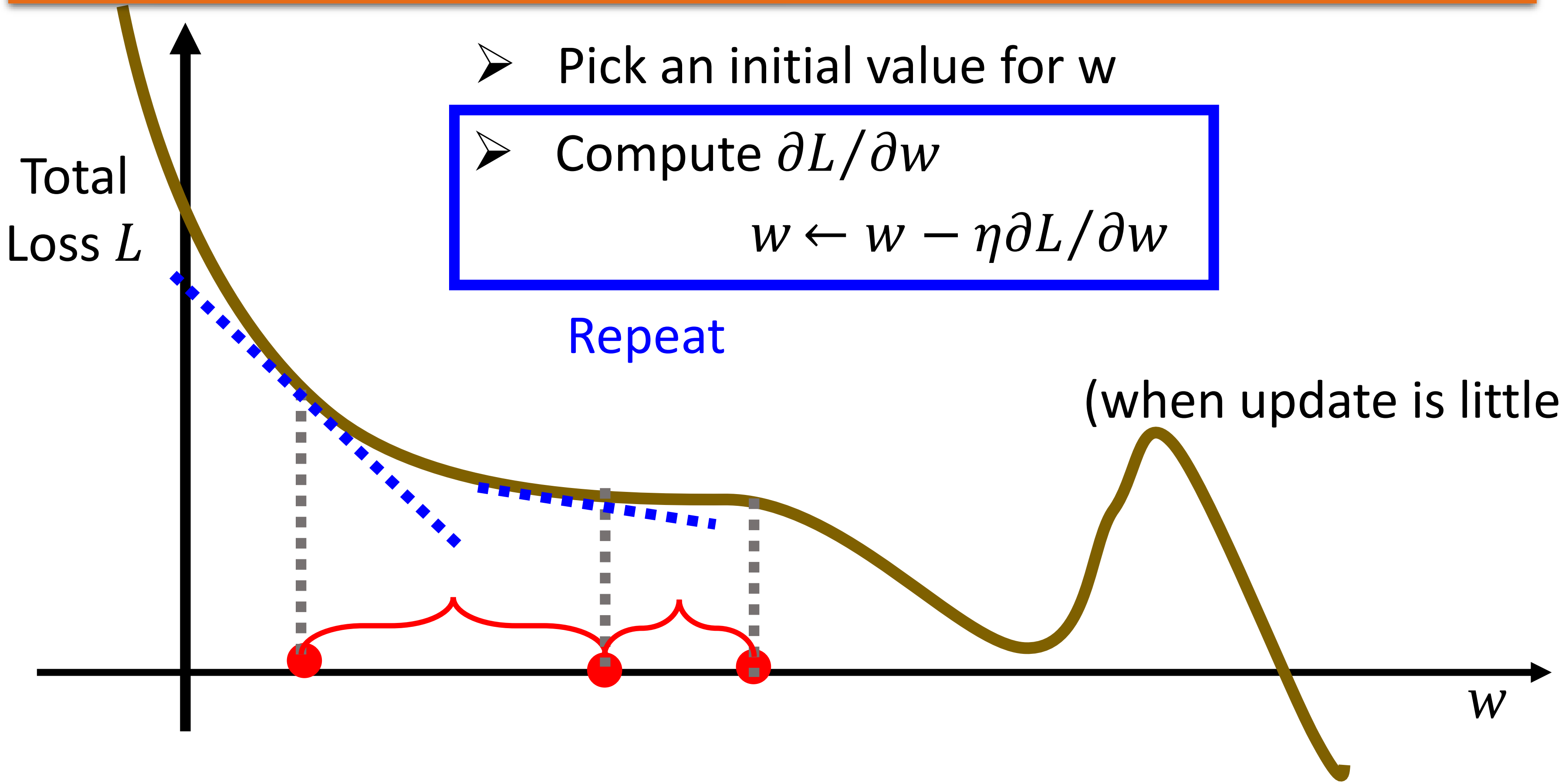
Find ***network parameters*** $\theta^*$ that minimize total loss L

- Pick an initial value for w
- Compute $\partial L / \partial w$

$$w \leftarrow w - \eta \partial L / \partial w$$

Repeat

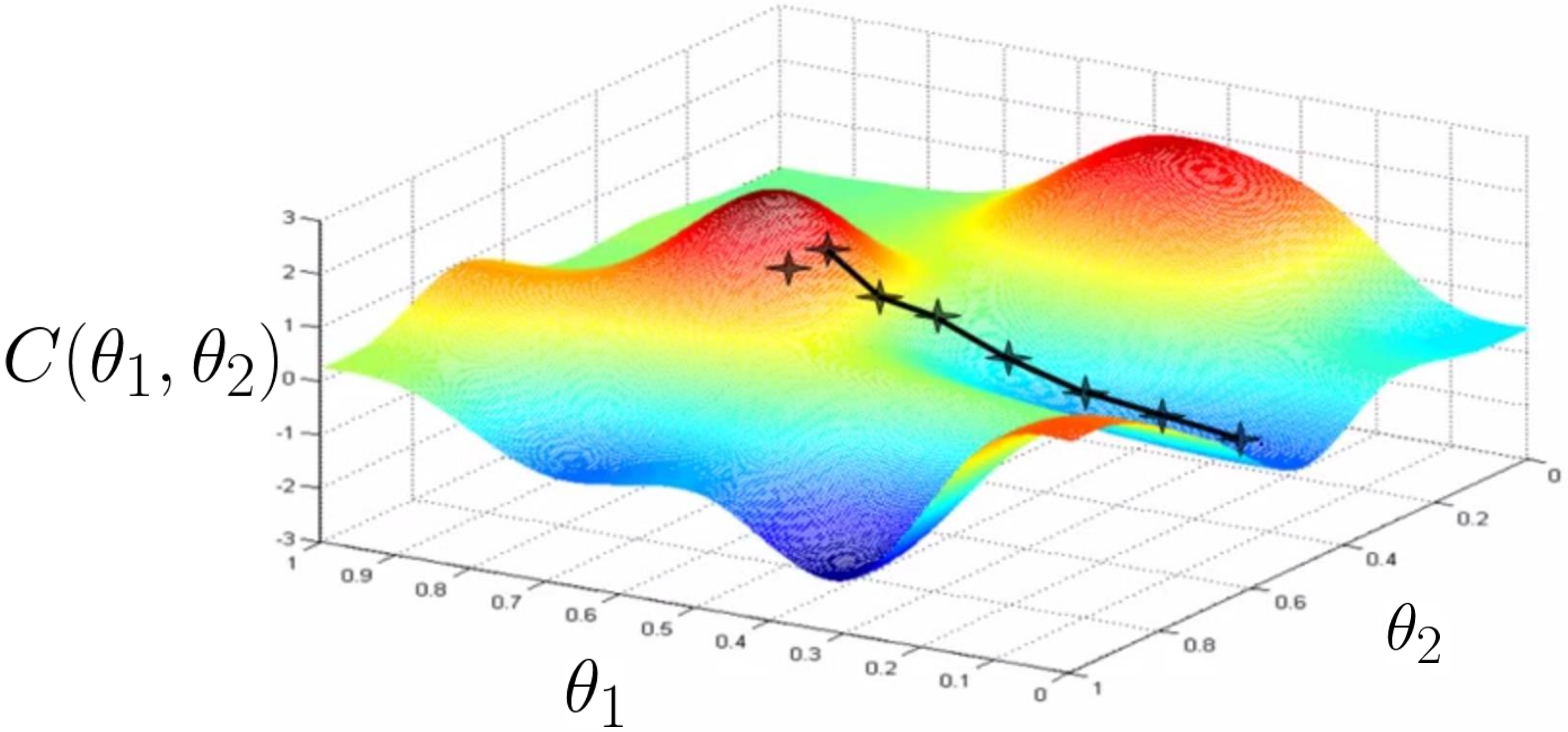
(when update is little)



# Gradient Descent

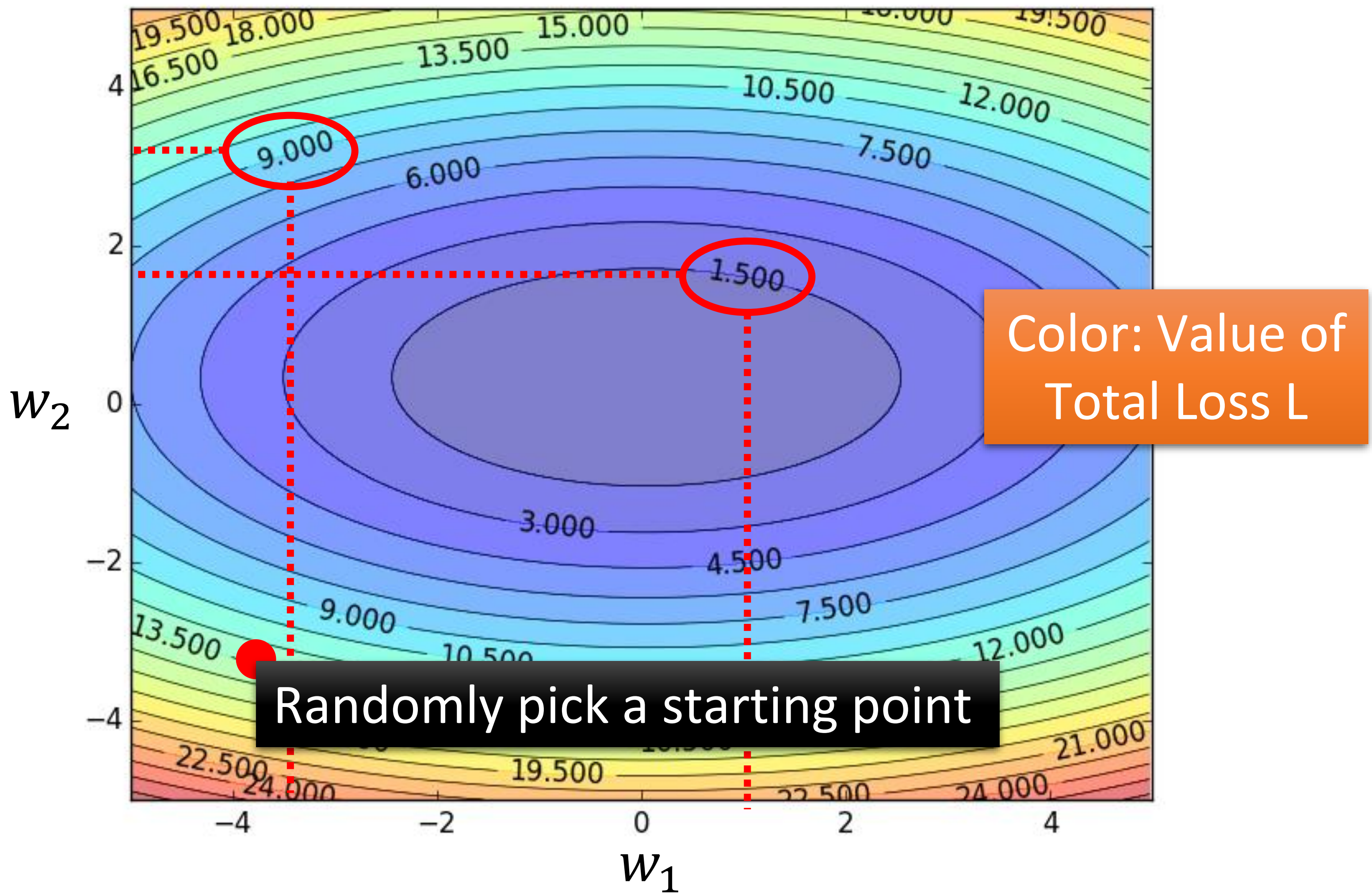


- Assume that $\theta$ has two variables $\{\theta_1, \theta_2\}$



# Gradient Descent

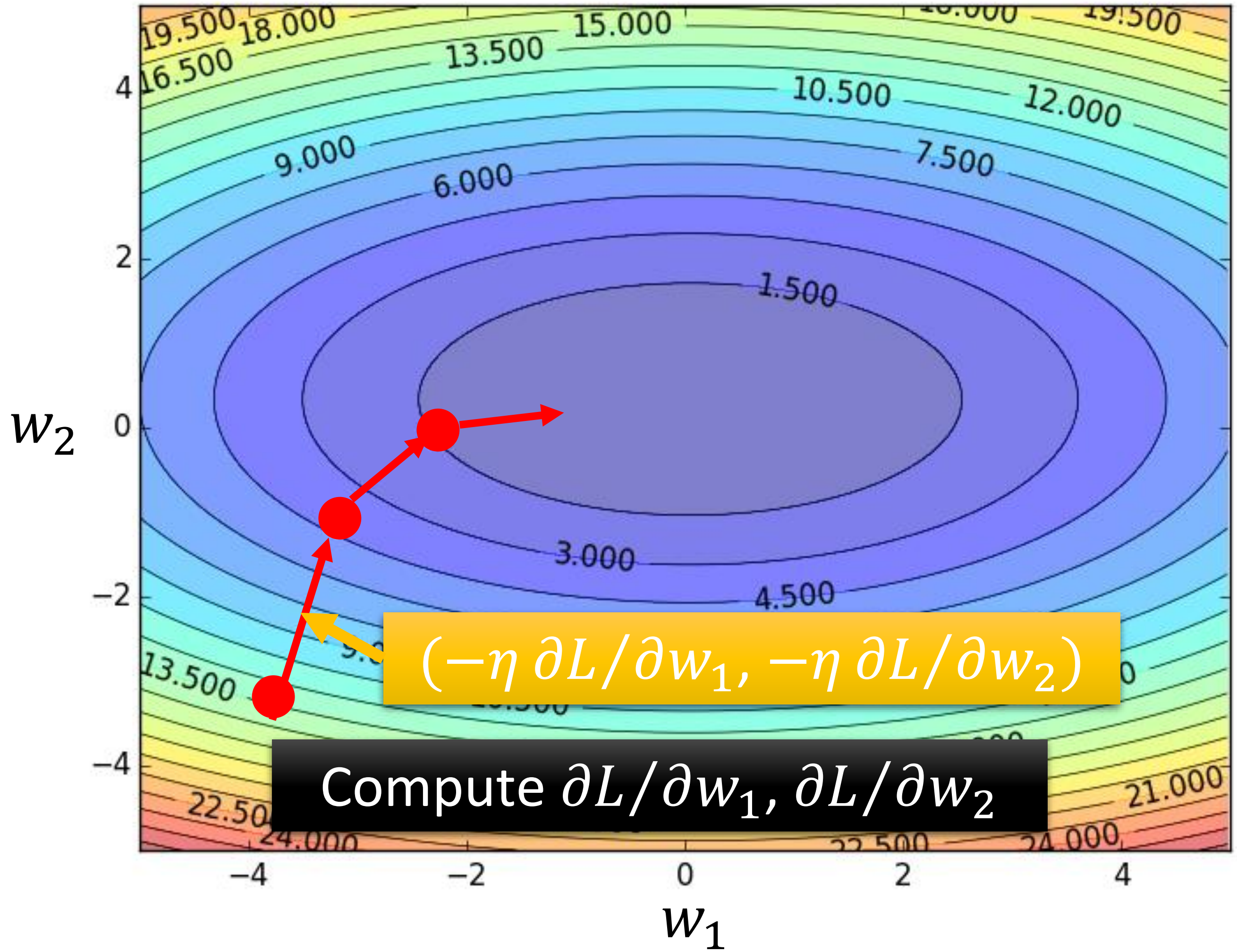




# Gradient Descent

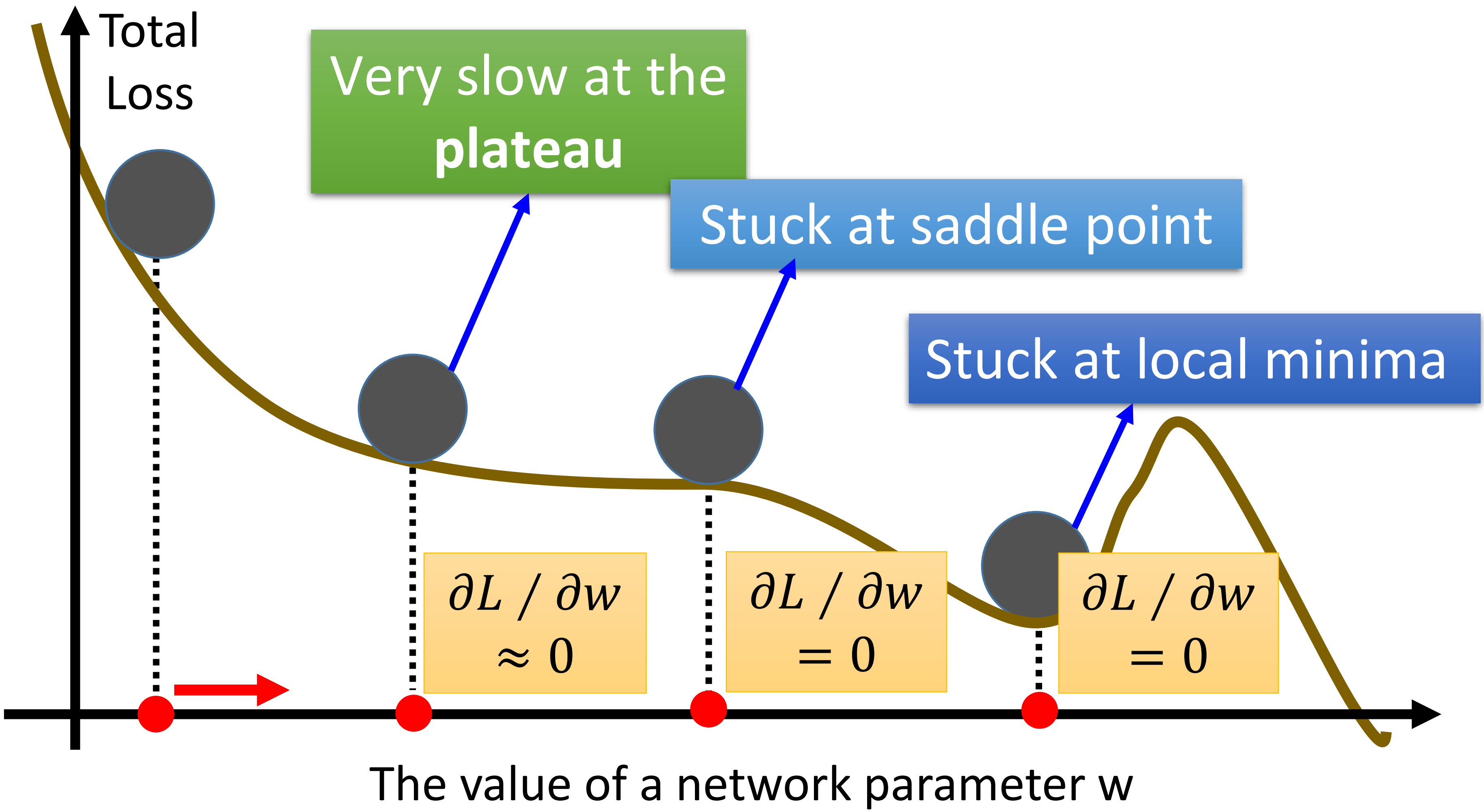
Hopfully, we would reach a minima .....





# Local Minima

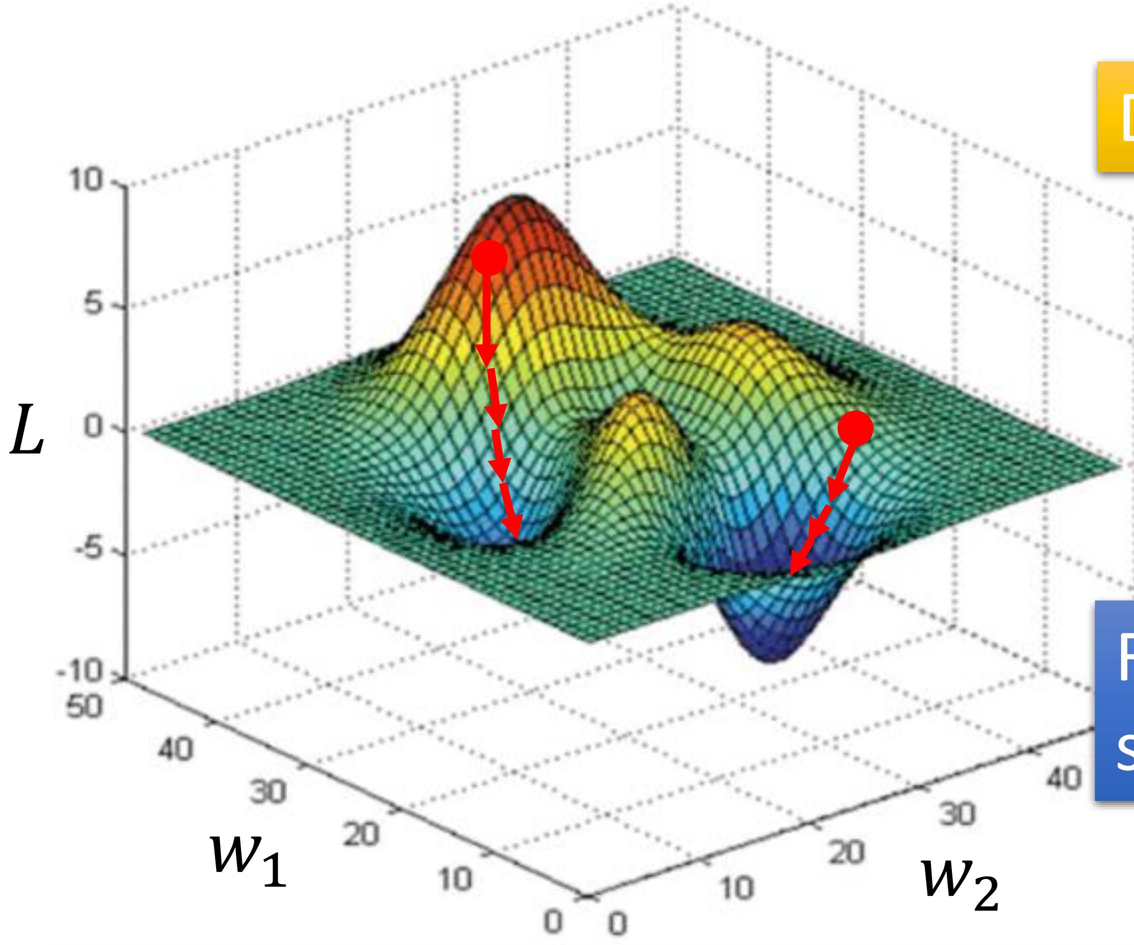




# Local Minima



- Gradient descent never guarantee global




Different initial point

Reach different minima, so different results

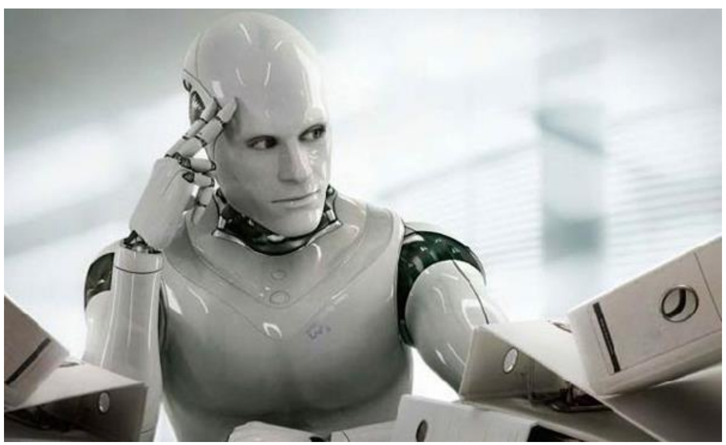
# Gradient Descent



This is the “learning” of machines in deep learning ……

Even AlphaGo using this approach.

People image ……



Actually …..


I hope you are not too disappointed :p

# Part I: Introduction to ML & DL



- Basic Machine Learning
- Basic Deep Learning
- **Toolkits and Learning Recipe**

# Deep Learning Toolkit



- Backpropagation: an efficient way to compute $\partial L / \partial w$ in neural network

TensorFlow

torch

theano

Caffe

Microsoft CNTK

Chainer

Deep Learning library produced by Amazon
DSSTNE

mxnet

# Three Steps for Deep Learning



Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……

Now If you want to find a function

If you have lots of function input/output (?) as training data

→ You can use deep learning

# Keras



TensorFlow or theano

Very flexible

Need some effort to learn

Interface of TensorFlow or Theano

keras

Easy to learn and use

(still have some flexibility)

You can modify it if you can write TensorFlow or Theano

# Keras



- François Chollet is the author of Keras.
  - He currently works for Google as a deep learning engineer and researcher.
- Keras means *horn* in Greek
- Documentation: http://keras.io/
- Example
  - https://github.com/fchollet/keras/tree/master/examples
- Step-by-step lecture by Prof. Hung-Yi Lee
  - Slide
    http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/Keras.pdf
  - Lecture recording:
    https://www.youtube.com/watch?v=qetE6uUoLQA

# 使用 Keras 心得

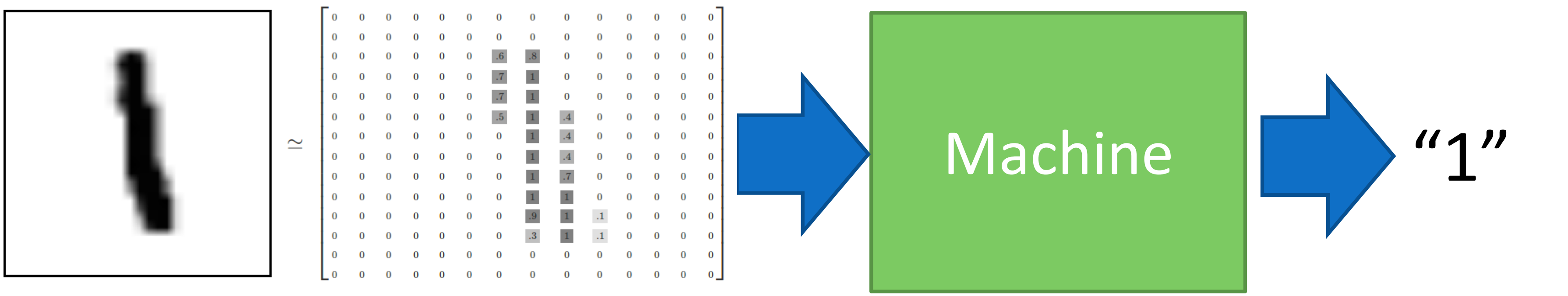



感謝 沈昇勳 同學提供圖檔
Deep Learning研究生
朋友覺得我在
我媽覺得我在
大眾覺得我在
指導教授覺得我在
我以為我在
事實上我在

# Example Application



- Handwriting Digit Recognition




28 x 28

MNIST Data: http://yann.lecun.com/exdb/mnist/

“Hello world” for deep learning

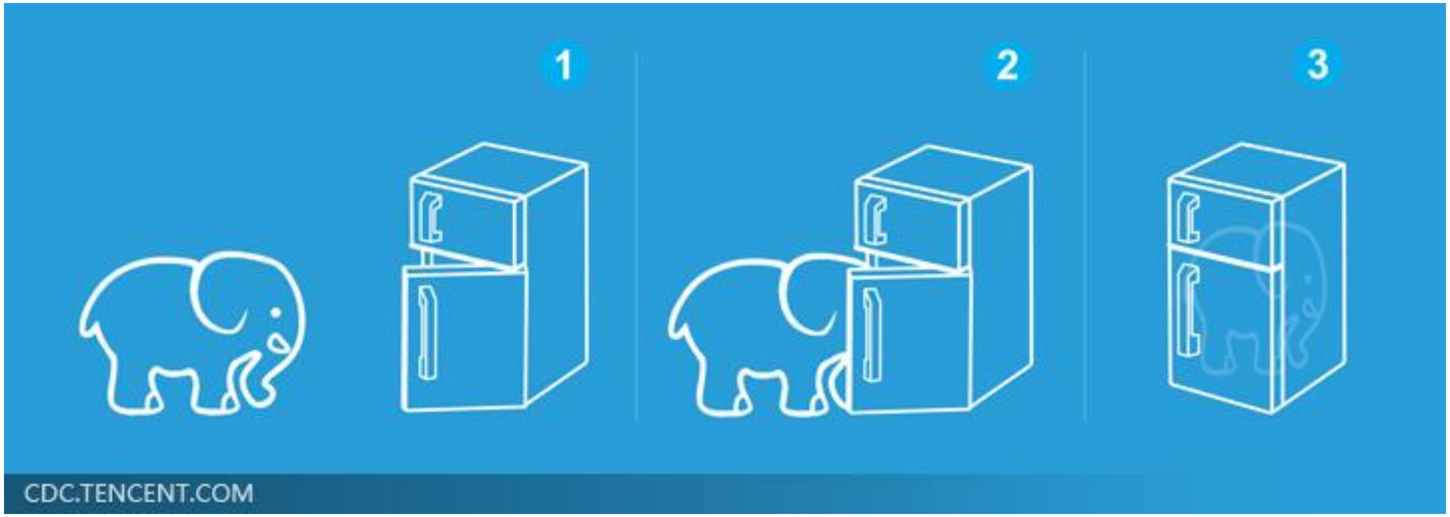
Keras provides data sets loading function: http://keras.io/datasets/

# Three Steps for Deep Learning



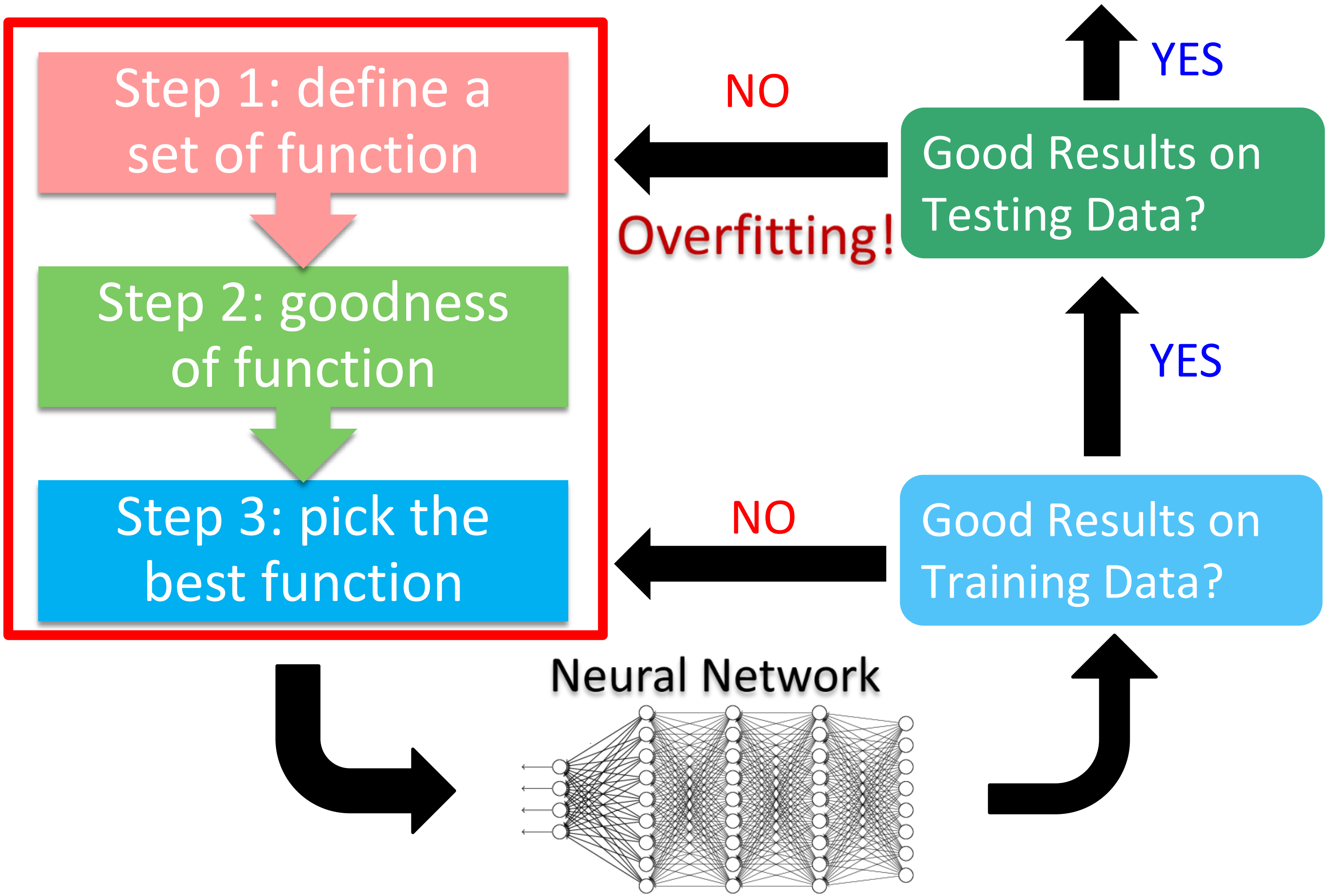
Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



# Learning Recipe





# Overfitting




- Possible solutions
  - more training samples
  - some tips: dropout, etc.

# Learning Recipe



YES

Good Results on Testing Data?

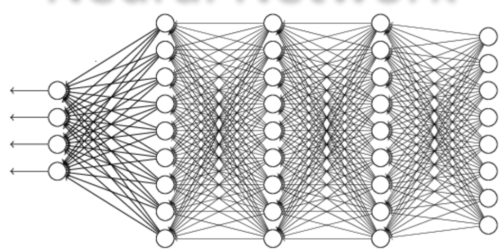
YES

Different approaches for different problems

e.g. dropout for good results on testing data

Good Results on Training Data?

Neural Network



# Learning Recipe



- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Training Data?

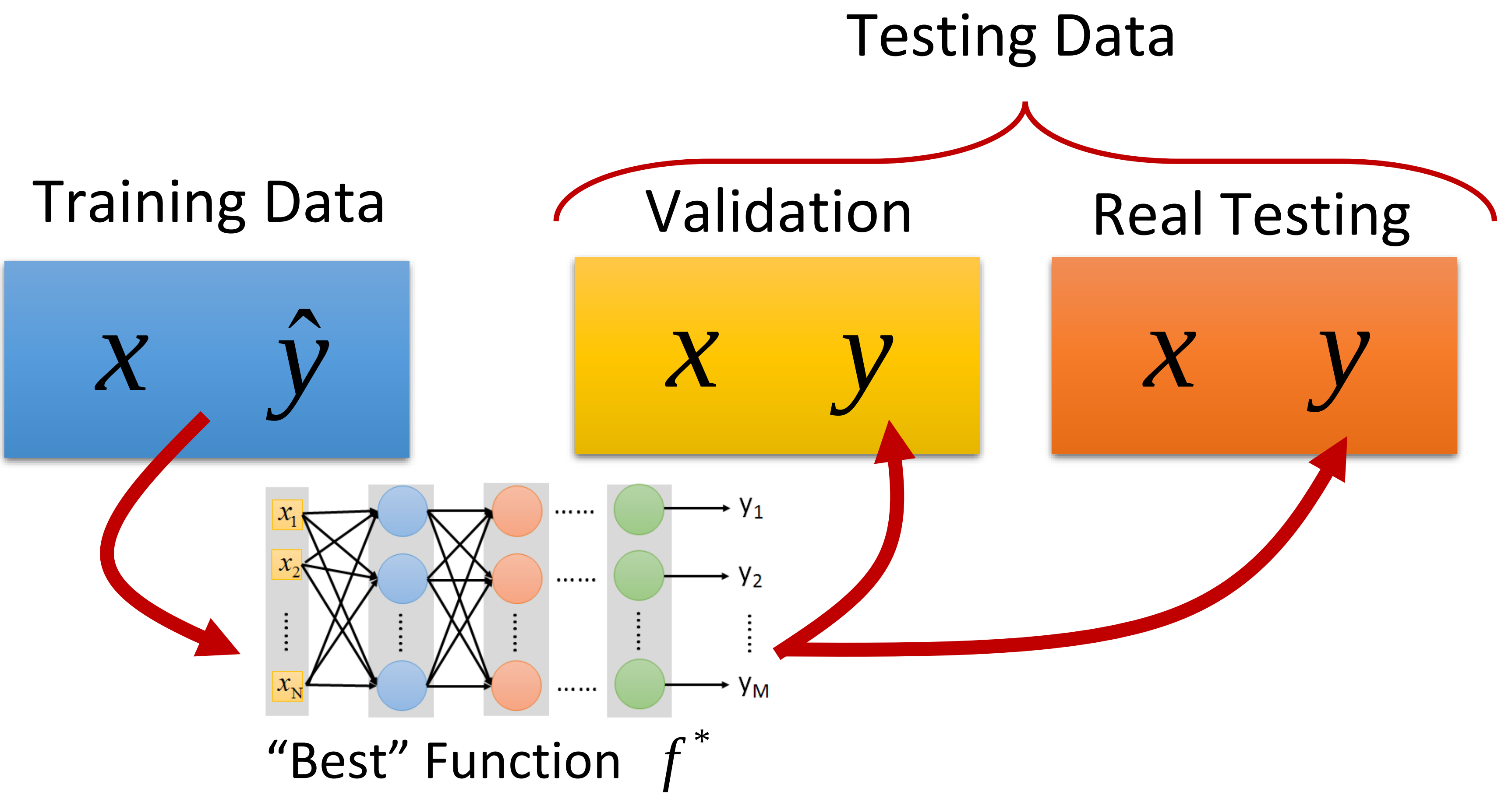
YES

Good Results on Testing Data?

YES

# Learning Recipe





# Learning Recipe



Testing Data

Training Data

$x$ $\hat{y}$

Validation

$x$ $y$

immediately know the performance

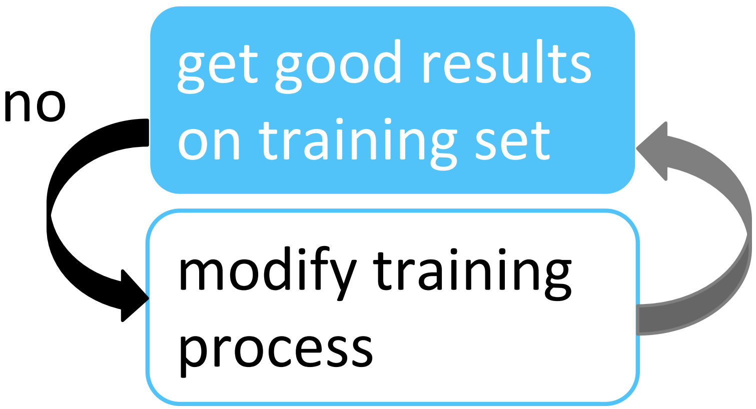
Real Testing

$x$ $y$

Do not know the performance

# Learning Recipe





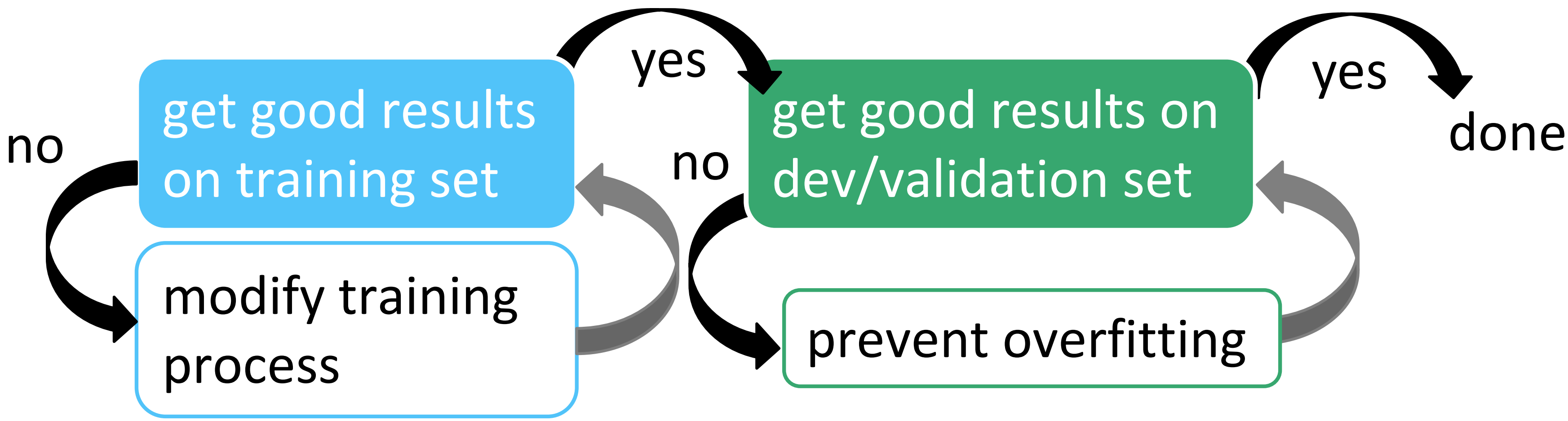

- Possible reasons
  - no good function exists: bad hypothesis function set

    → reconstruct the model architecture
  - cannot find a good function: local optima

    → change the training strategy

# Learning Recipe



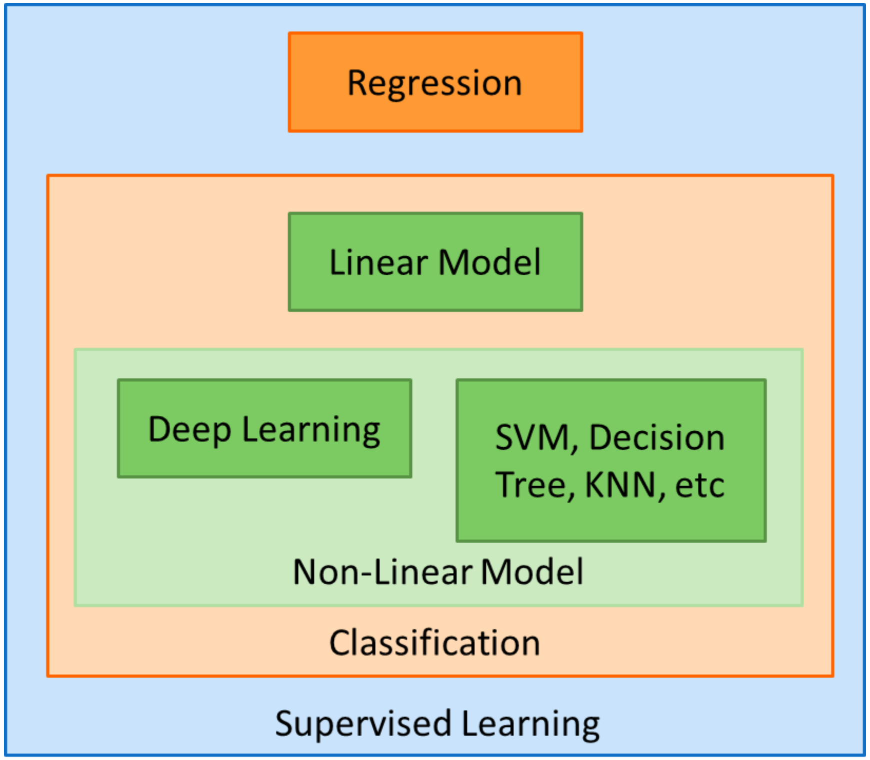



Better performance on training but worse performance on dev → overfitting

# Concluding Remarks



- Basic Machine Learning
  1. Define a set of functions
  2. Measure goodness of functions
  3. Pick the best function
- Basic Deep Learning
  - Stacked functions



# Talk Outline



Part I: Introduction to
Machine Learning & Deep Learning

Part II: Variants of Neural Nets

Part III: Beyond Supervised Learning
& Recent Trends

# PART II

Variants of Neural Networks

# PART II: Variants of Neural Networks



- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)

# PART II: Variants of Neural Networks



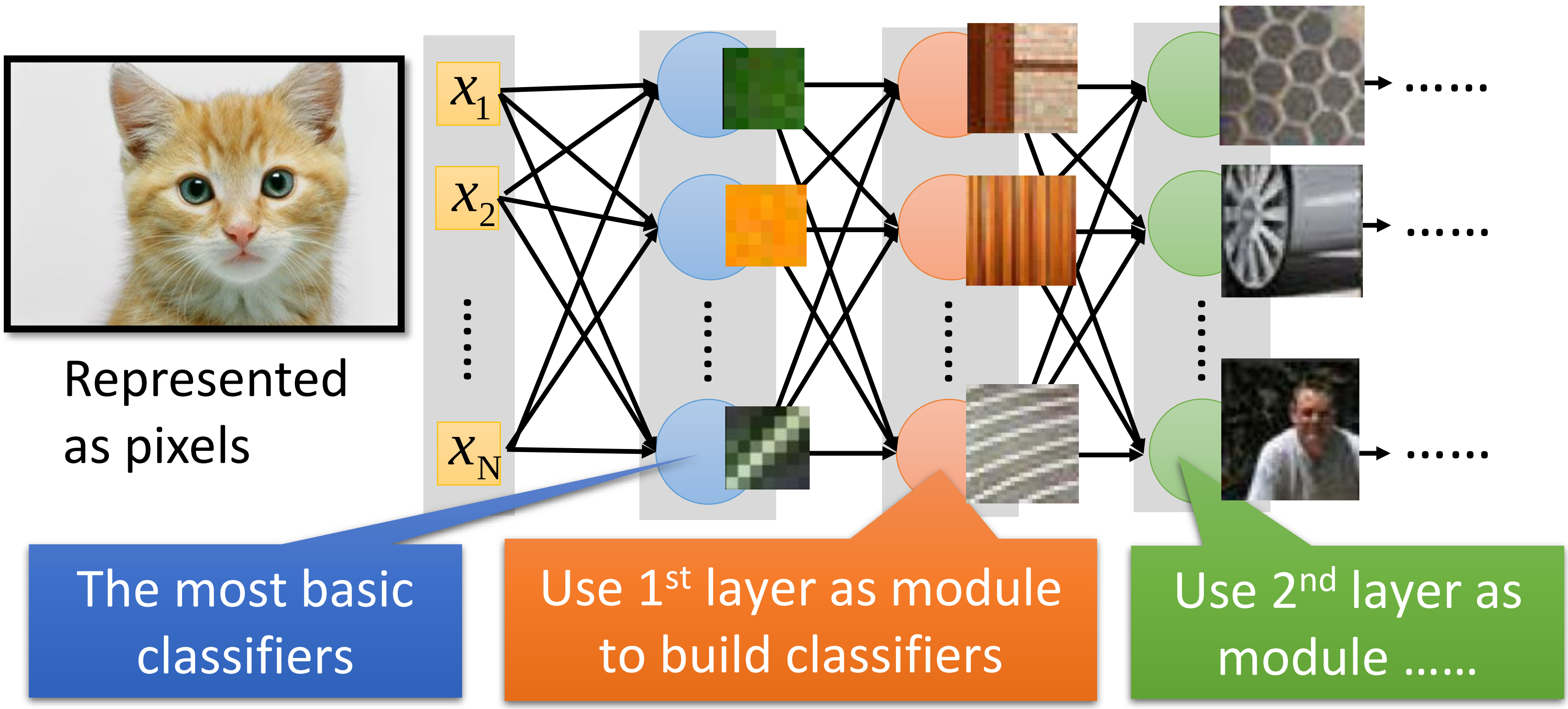
- **Convolutional Neural Network (CNN)**
- Recurrent Neural Network (RNN)

Widely used in image processing

# Why CNN for Image

(Zeiler, M. D., *ECCV 2014*)






Can the network be simplified by considering the properties of images?

# Why CNN for Image



- Some patterns are much smaller than the whole image

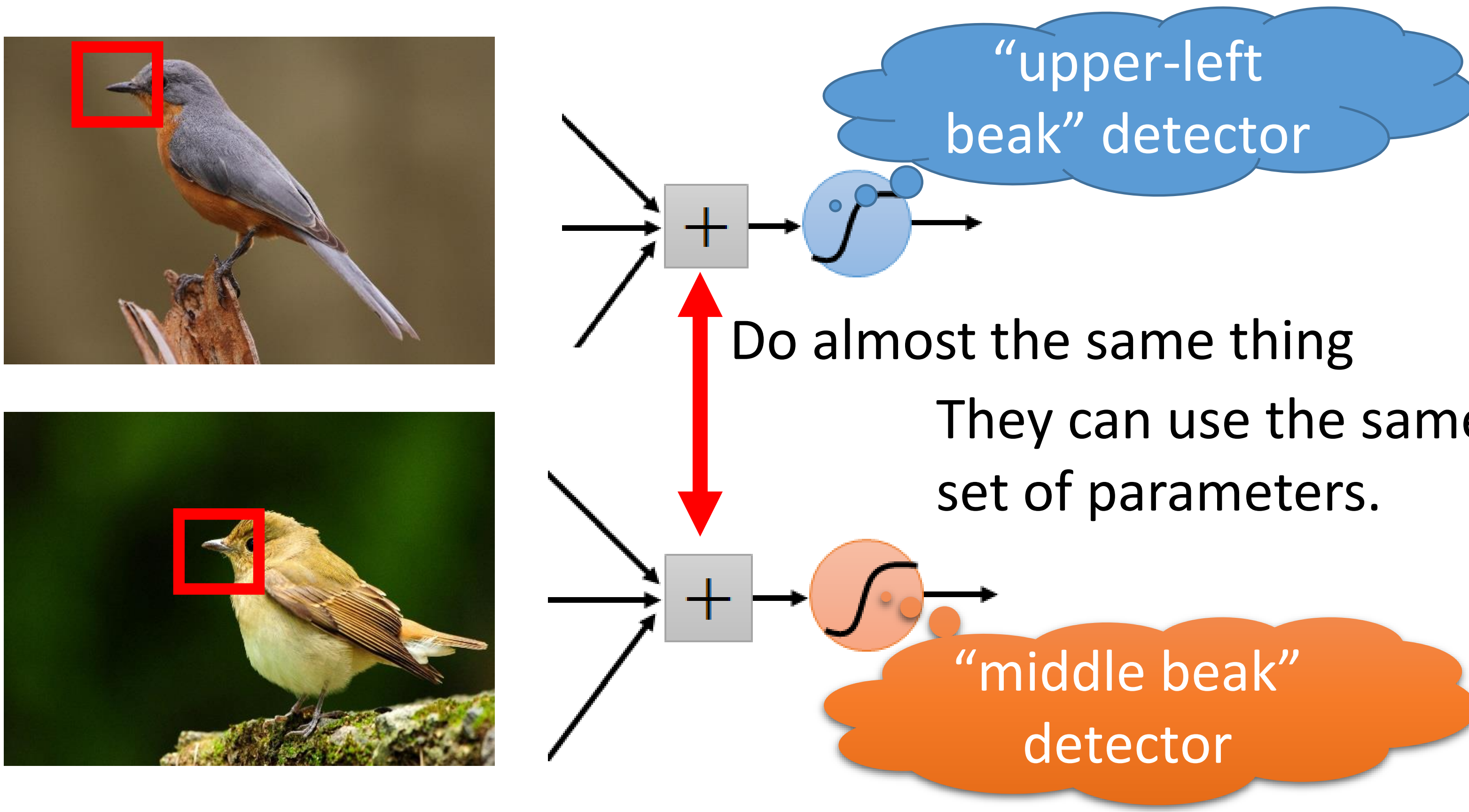
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

# Why CNN for Image



- The same patterns appear in different regions.



# Why CNN for Image



- Subsampling the pixels will not change the object

bird

subsampling

bird

We can subsample the pixels to make image smaller

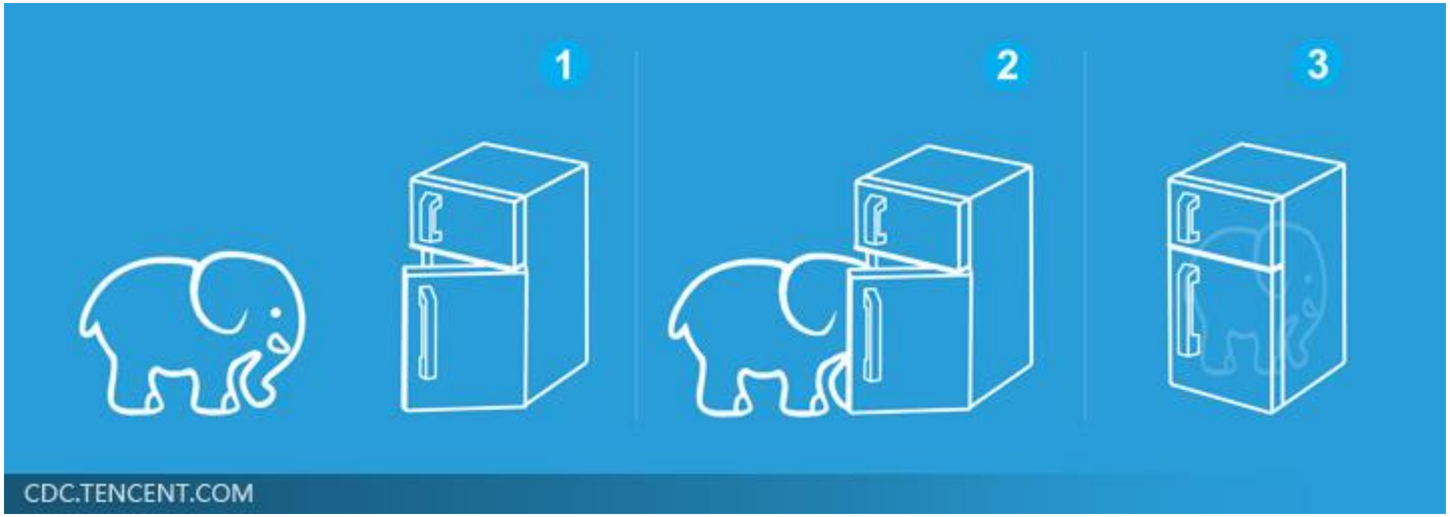
Less parameters for the network to process the image

# Three Steps for Deep Learning



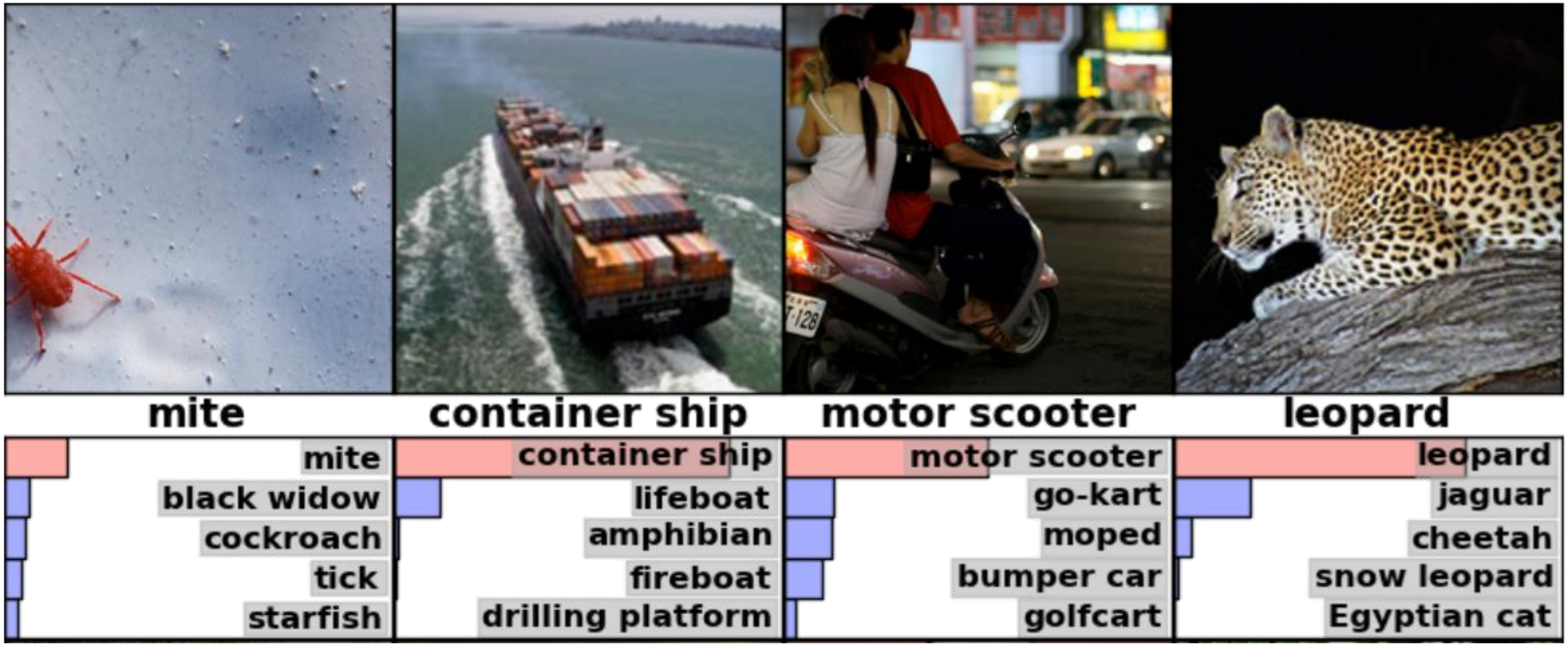
Step 1: Convolutional Neural Network → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



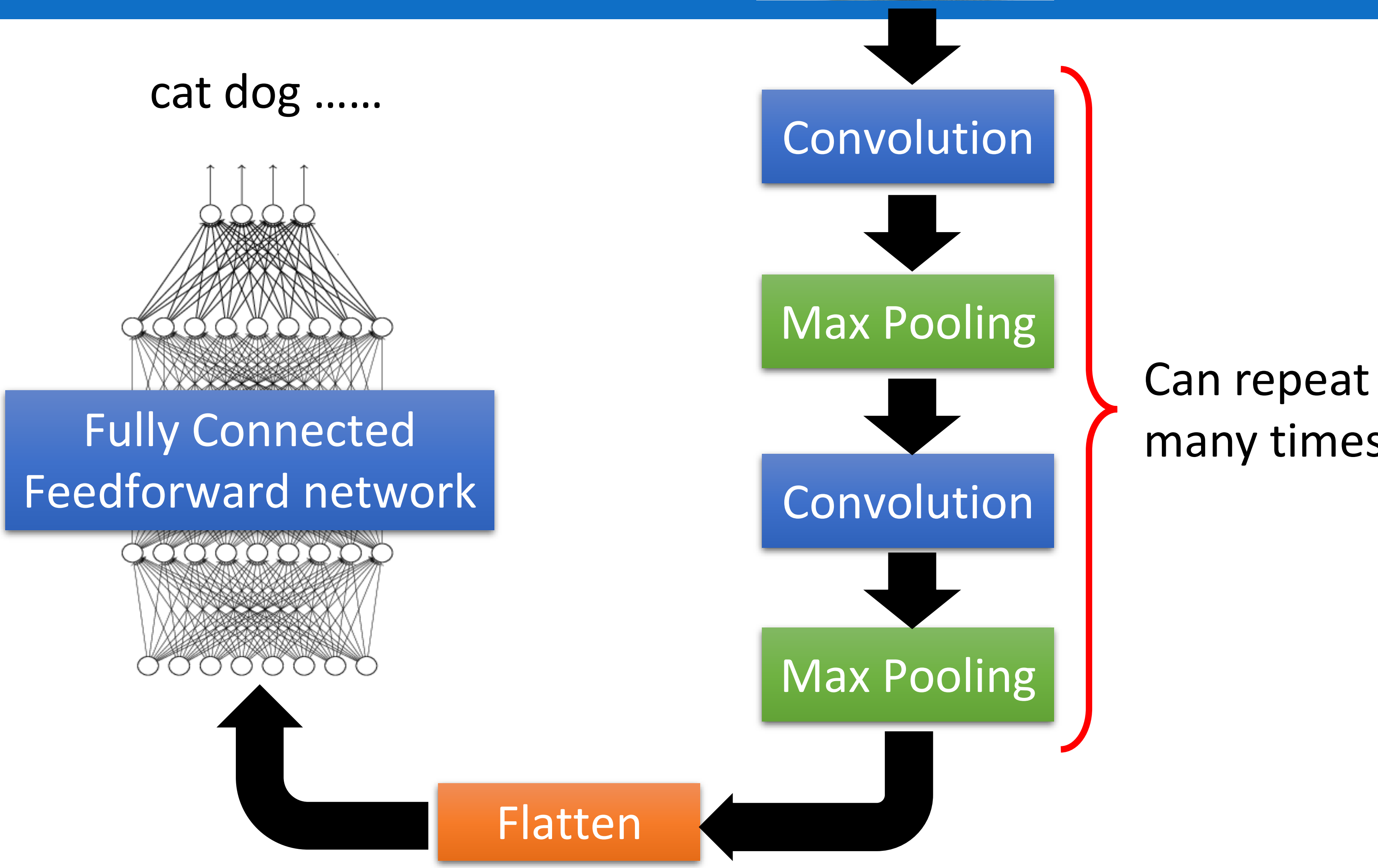
# Image Recognition






http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

# The Whole CNN



# The Whole CNN

**Property 1**

- Some patterns are much smaller than the whole image

**Property 2**

- The same patterns appear in different regions

**Property 3**

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

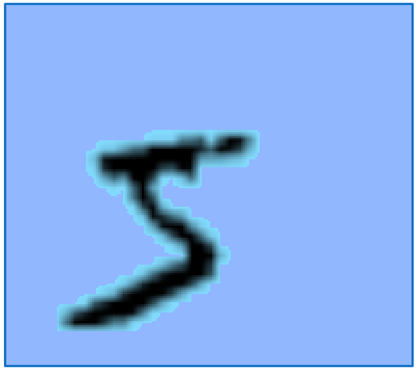
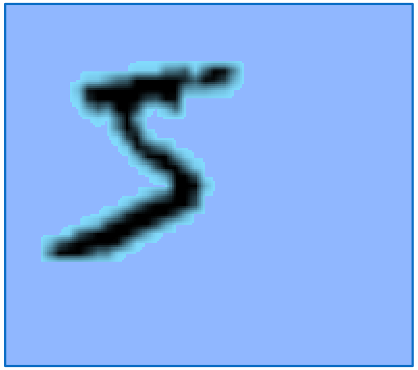
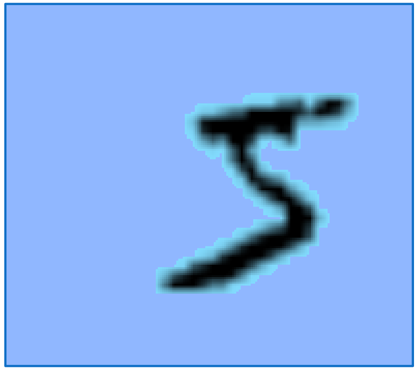
Max Pooling

Can repeat many times

Flatten

# Image Recognition

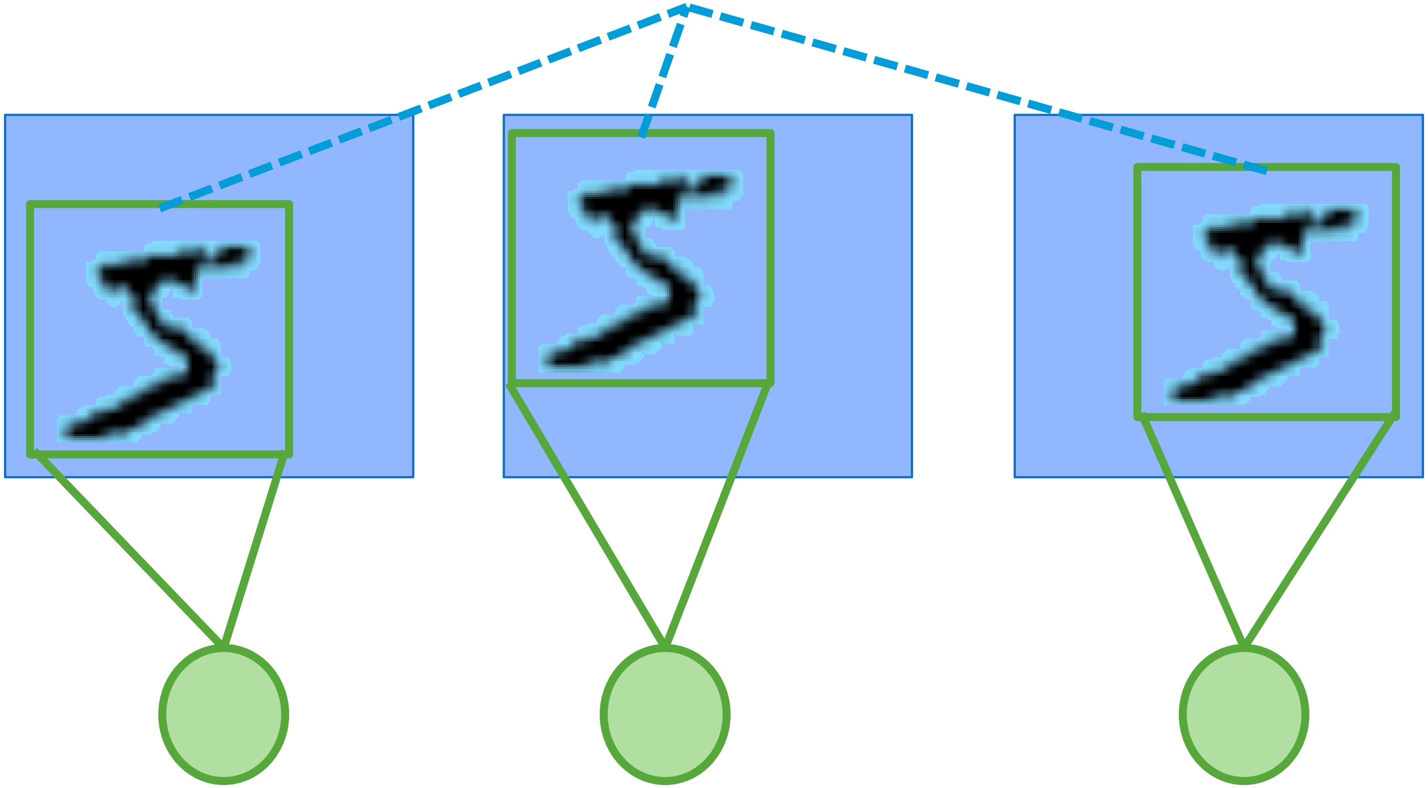


# Local Connectivity



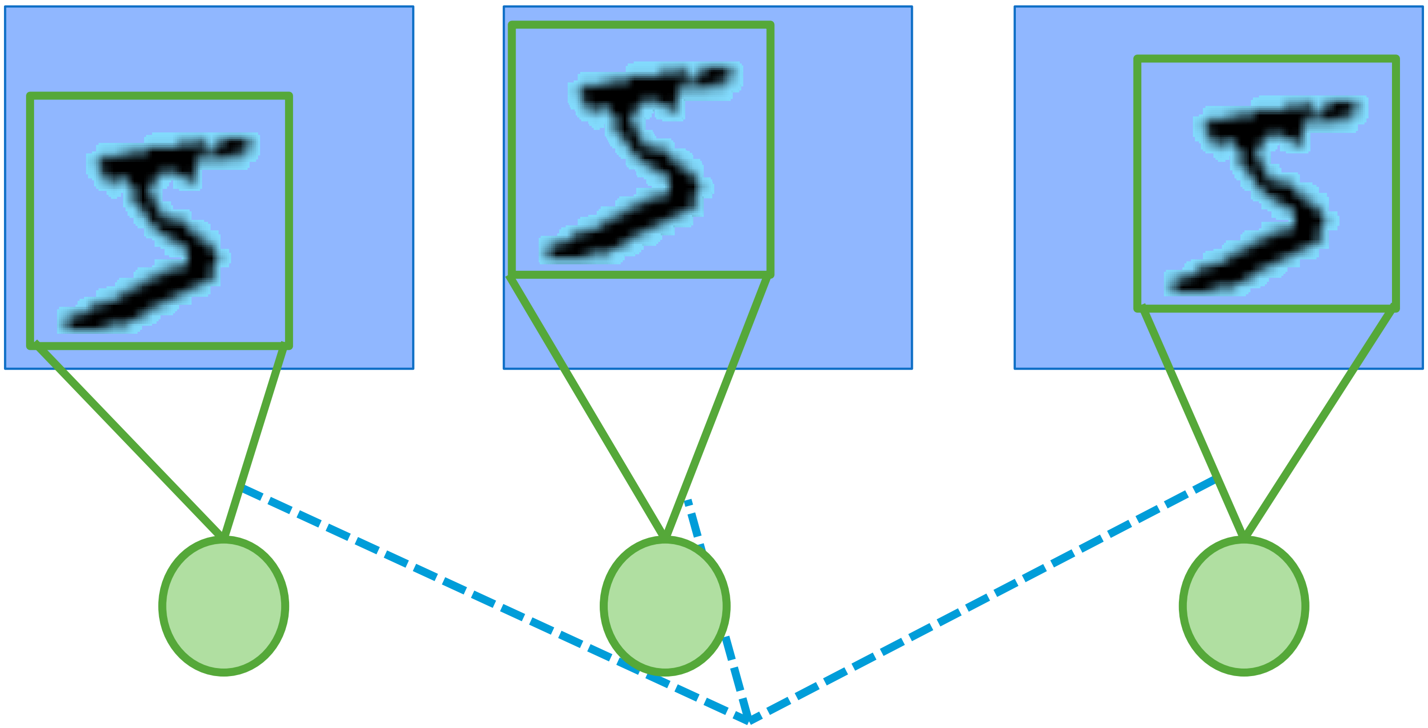
Neurons connect to a small region



# Parameter Sharing



- The same feature in different positions

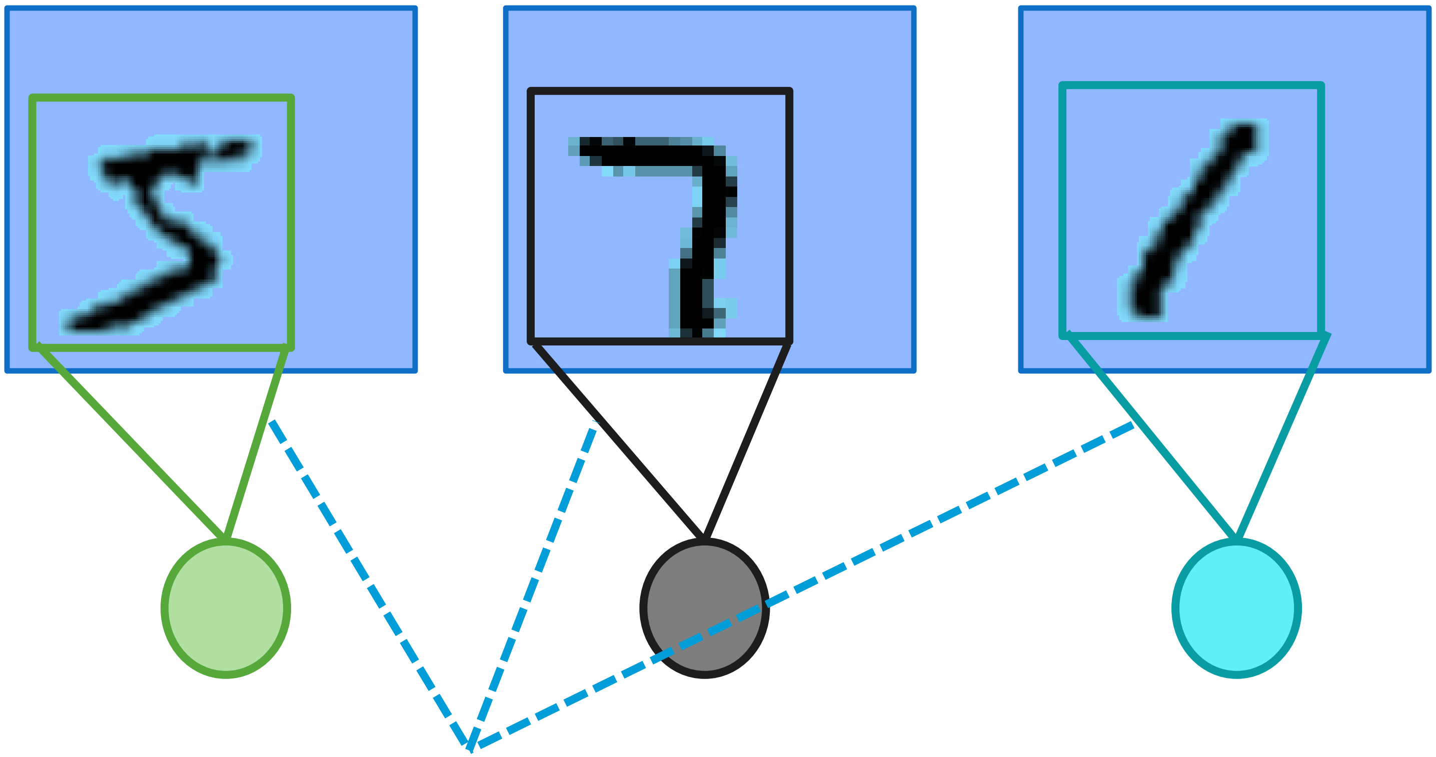


Neurons share the same weights

# Parameter Sharing



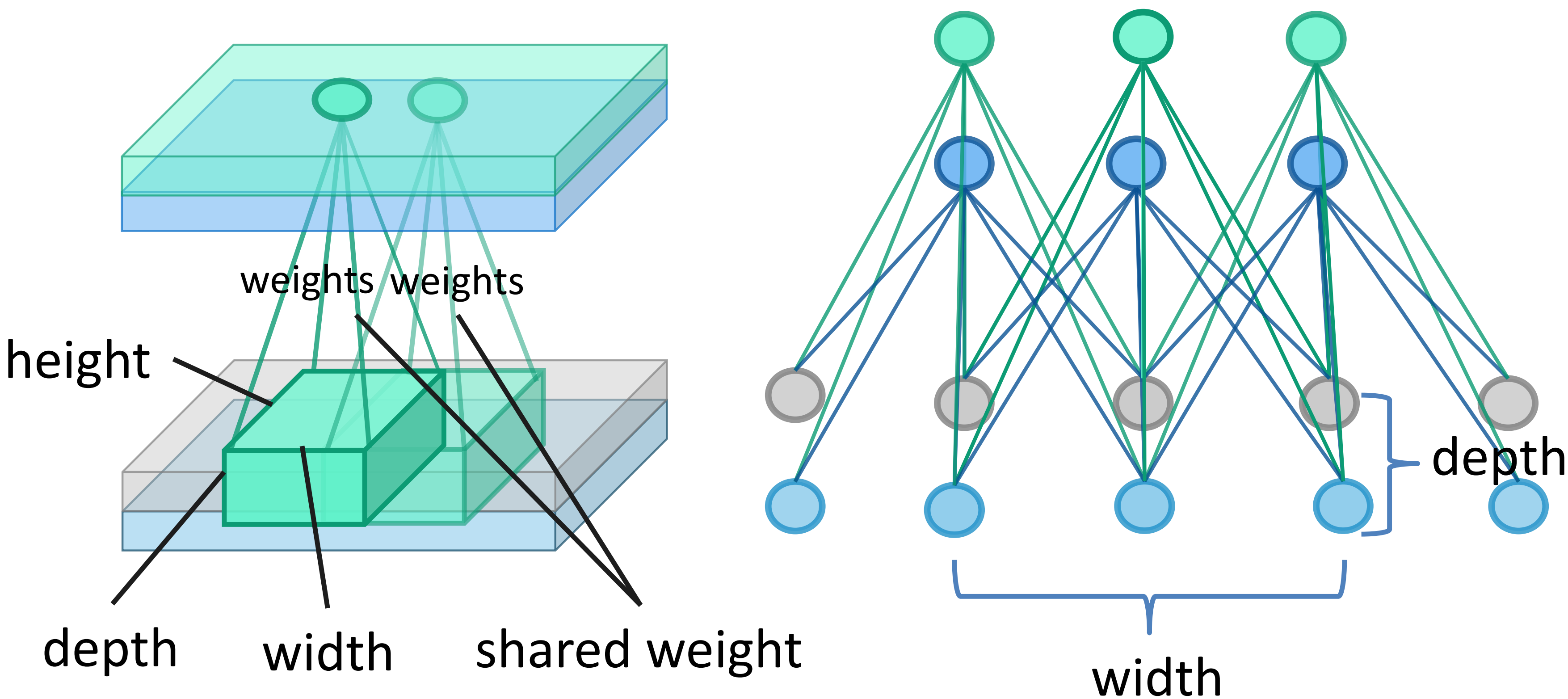
- Different features in the same position



Neurons have different weights

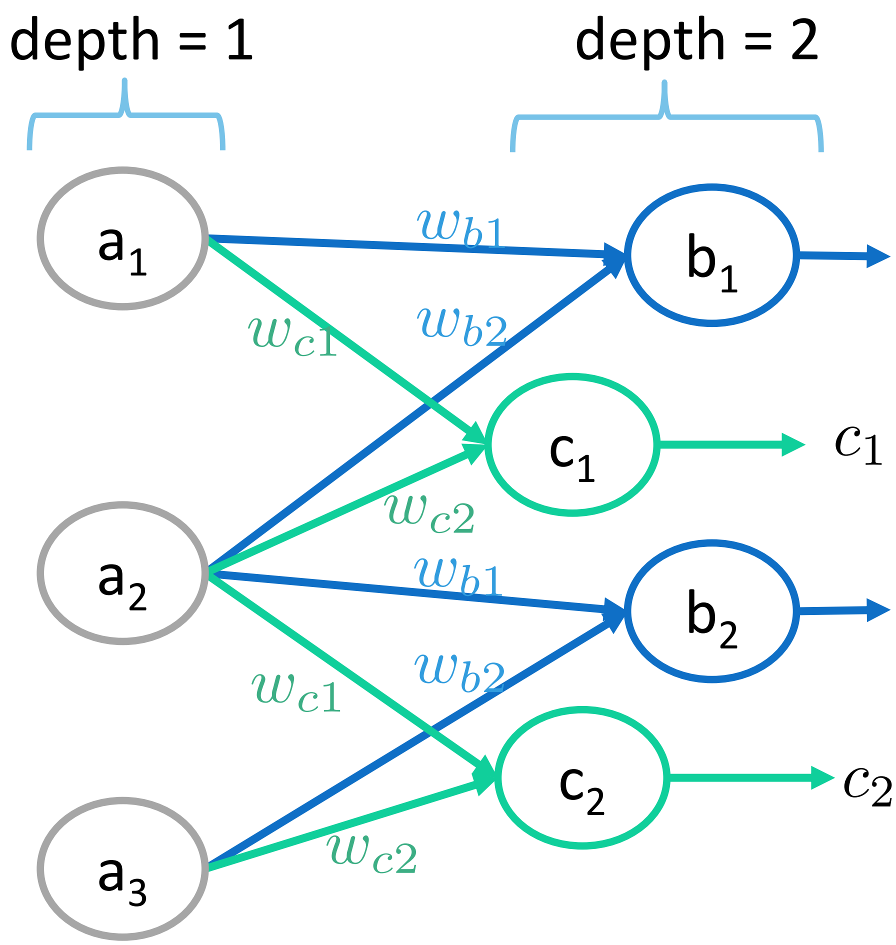
# Convolutional Layers





# Convolutional Layers






$b_1 = w_{b1} a_1 + w_{b2} a_2$

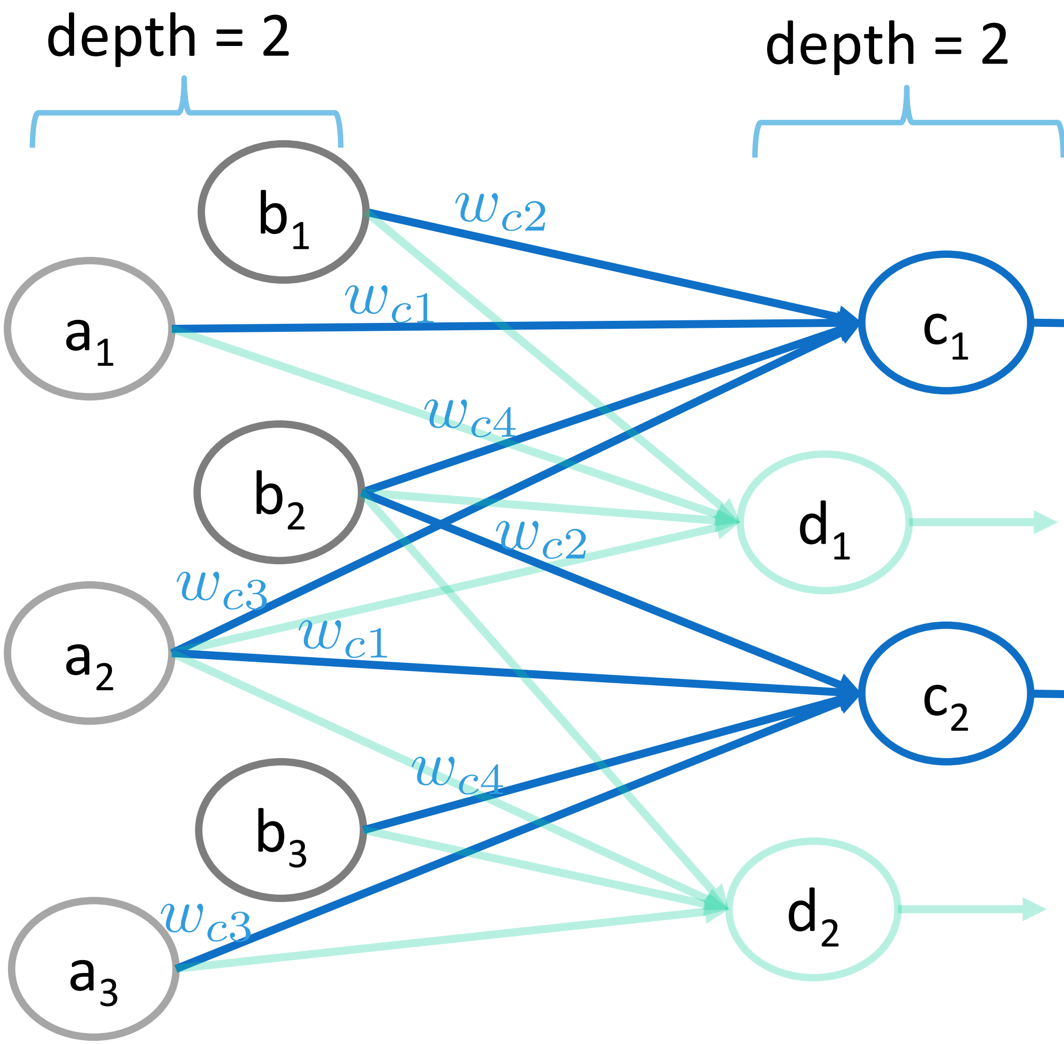
$c_1 = w_{c1} a_1 + w_{c2} a_2$

$b_2 = w_{b1} a_2 + w_{b2} a_3$

$c_2 = w_{c1} a_2 + w_{c2} a_3$

# Convolutional Layers



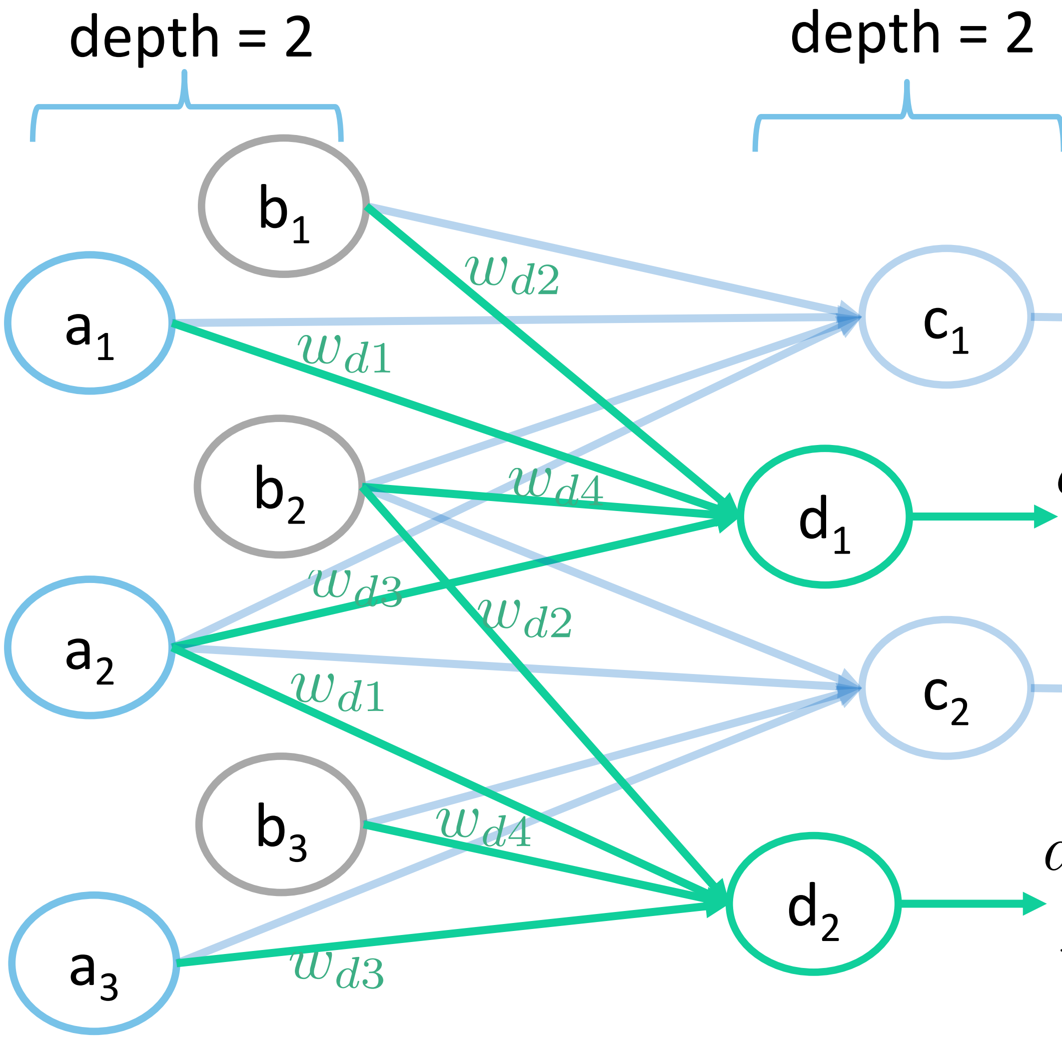



$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

# Convolutional Layers






$$c_1 = a_1 w_{c1} + b_1 w_{c2} + a_2 w_{c3} + b_2 w_{c4}$$

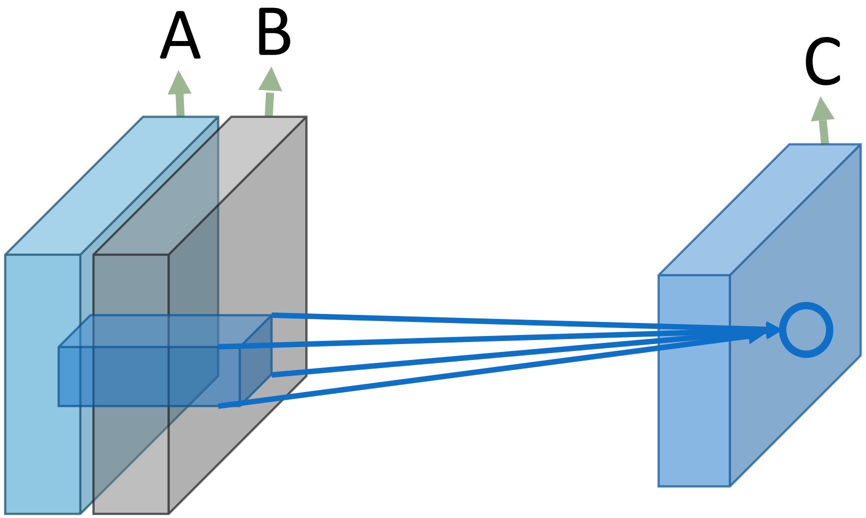
$$d_1 = a_1 w_{d1} + b_1 w_{d2} + a_2 w_{d3} + b_2 w_{d4}$$

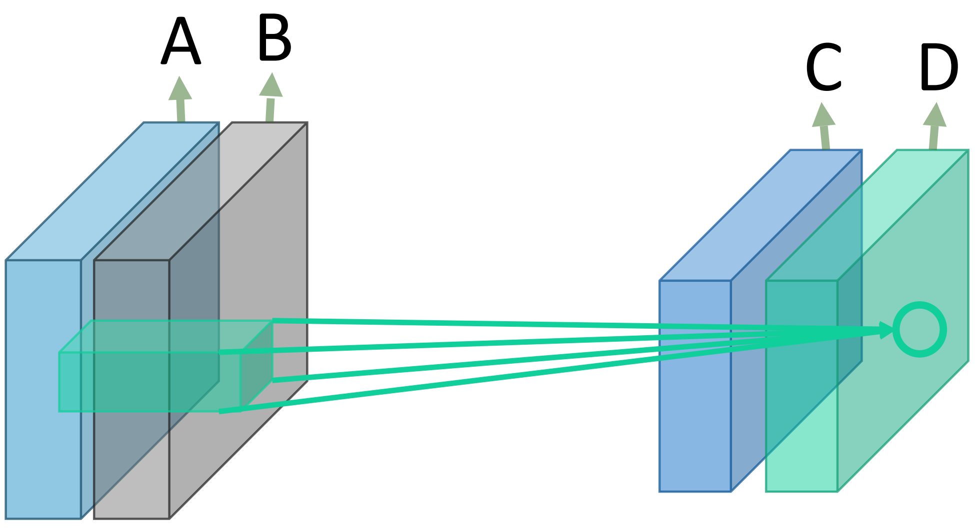
$$c_2 = a_2 w_{c1} + b_2 w_{c2} + a_3 w_{c3} + b_3 w_{c4}$$

$$d_2 = a_2 w_{d1} + b_2 w_{d2} + a_3 w_{d3} + b_3 w_{d4}$$

# Convolutional Layers

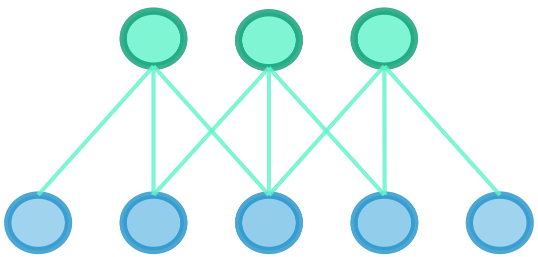




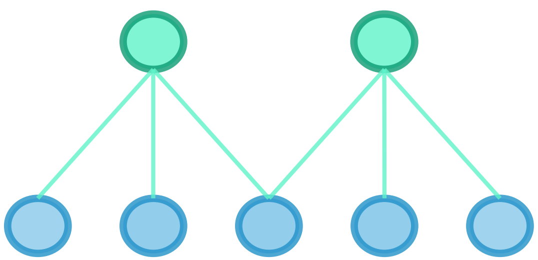



# Hyper-parameters of CNN

- Stride

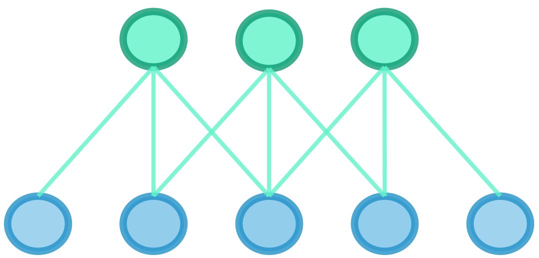


Stride = 1

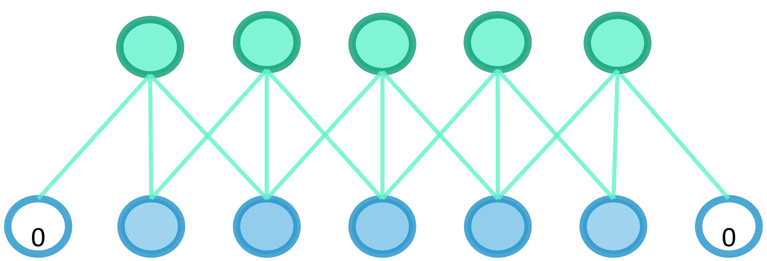


Stride = 2

- Padding

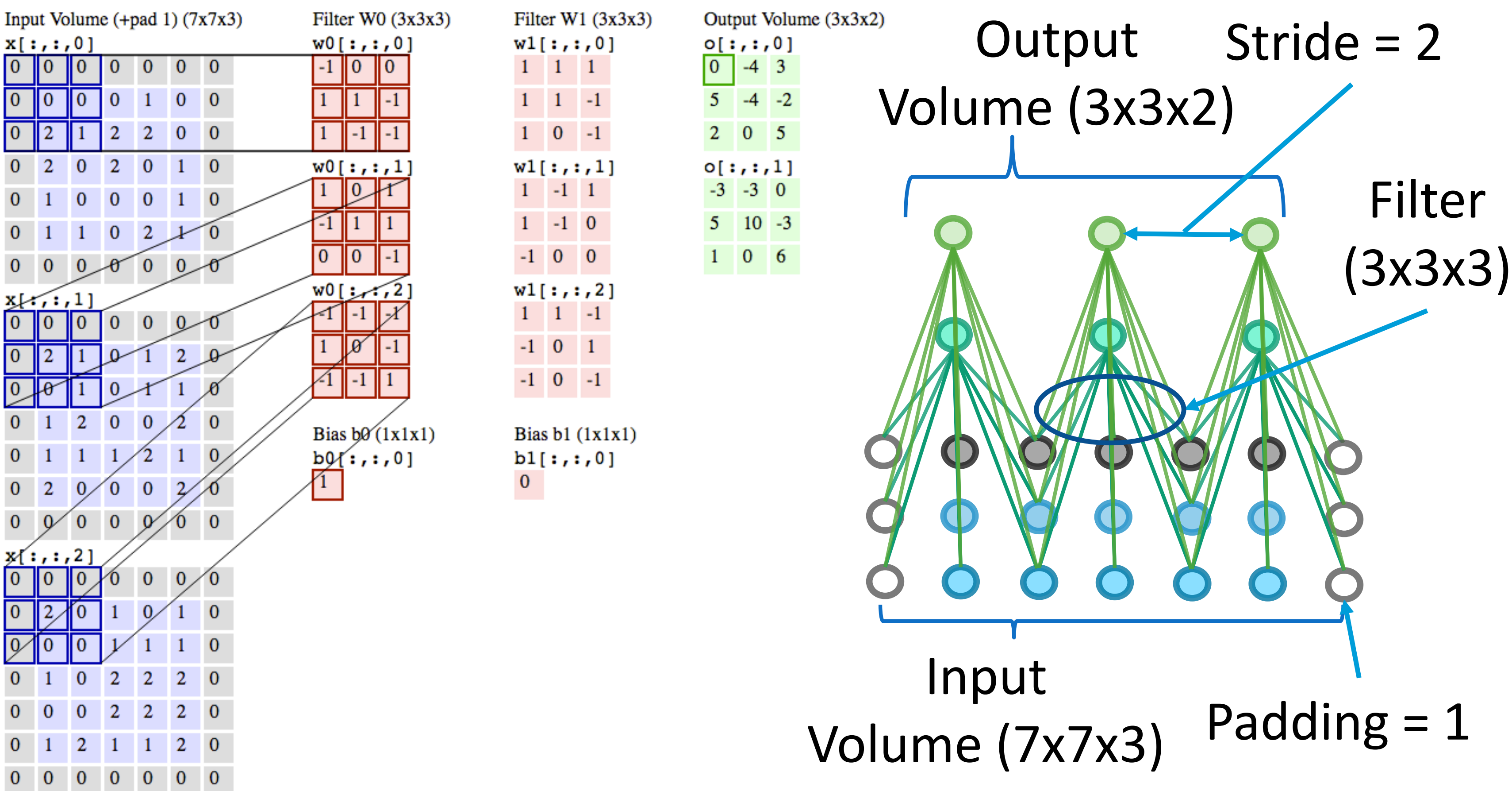


Padding = 0




Padding = 1

# Example




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

# Convolutional Layers



**Input Volume (+pad 1) (7x7x3)**

x[:,:,0]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 2 | 1 | 2 | 2 | 0 | 0 |
| 0 | 2 | 0 | 2 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,1]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 1 | 0 | 1 | 2 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 2 | 0 | 0 | 2 | 0 |
| 0 | 1 | 1 | 1 | 2 | 1 | 0 |
| 0 | 2 | 0 | 0 | 0 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x[:,:,2]

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 2 | 2 | 2 | 0 |
| 0 | 0 | 0 | 2 | 2 | 2 | 0 |
| 0 | 1 | 2 | 1 | 1 | 2 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Filter W0 (3x3x3)**

w0[:,:,0]

| | | |
|---|---|---|
| -1 | 0 | 0 |
| 1 | 1 | -1 |
| 1 | -1 | -1 |

w0[:,:,1]

| | | |
|---|---|---|
| 1 | 0 | 1 |
| -1 | 1 | 1 |
| 0 | 0 | -1 |

w0[:,:,2]

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 1 | 0 | -1 |
| -1 | -1 | 1 |

**Bias b0 (1x1x1)**

b0[:,:,0]

| |
|---|
| 1 |

**Filter W1 (3x3x3)**

w1[:,:,0]

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | -1 |
| 1 | 0 | -1 |

w1[:,:,1]

| | | |
|---|---|---|
| 1 | -1 | 1 |
| 1 | -1 | 0 |
| -1 | 0 | 0 |

w1[:,:,2]

| | | |
|---|---|---|
| 1 | 1 | -1 |
| -1 | 0 | 1 |
| -1 | 0 | -1 |

**Bias b1 (1x1x1)**

b1[:,:,0]

| |
|---|
| 0 |

**Output Volume (3x3x2)**

o[:,:,0]

| | | |
|---|---|---|
| 0 | -4 | 3 |
| 5 | -4 | -2 |
| 2 | 0 | 5 |

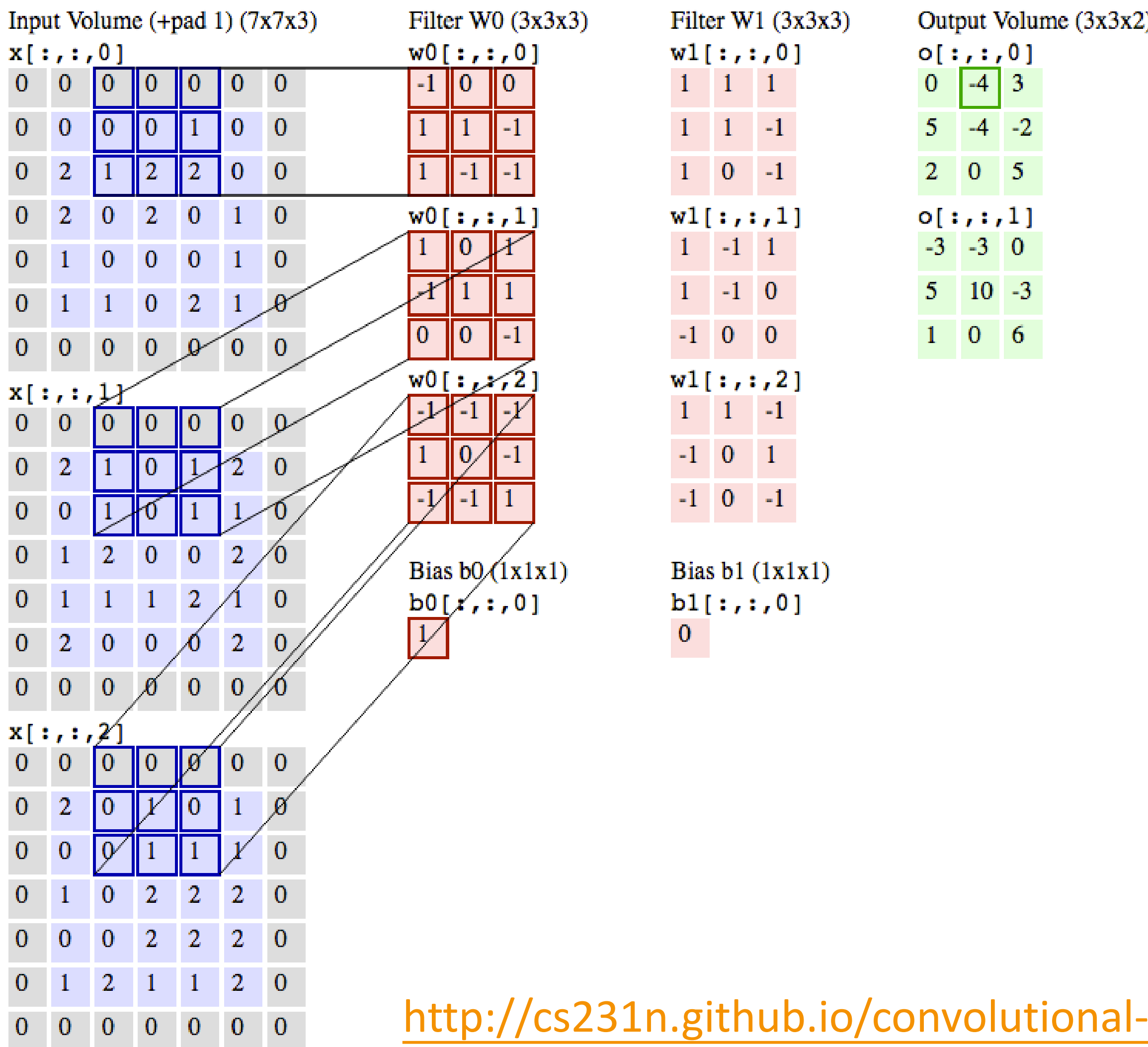
o[:,:,1]

| | | |
|---|---|---|
| -3 | -3 | 0 |
| 5 | 10 | -3 |
| 1 | 0 | 6 |

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

# Convolutional Layers

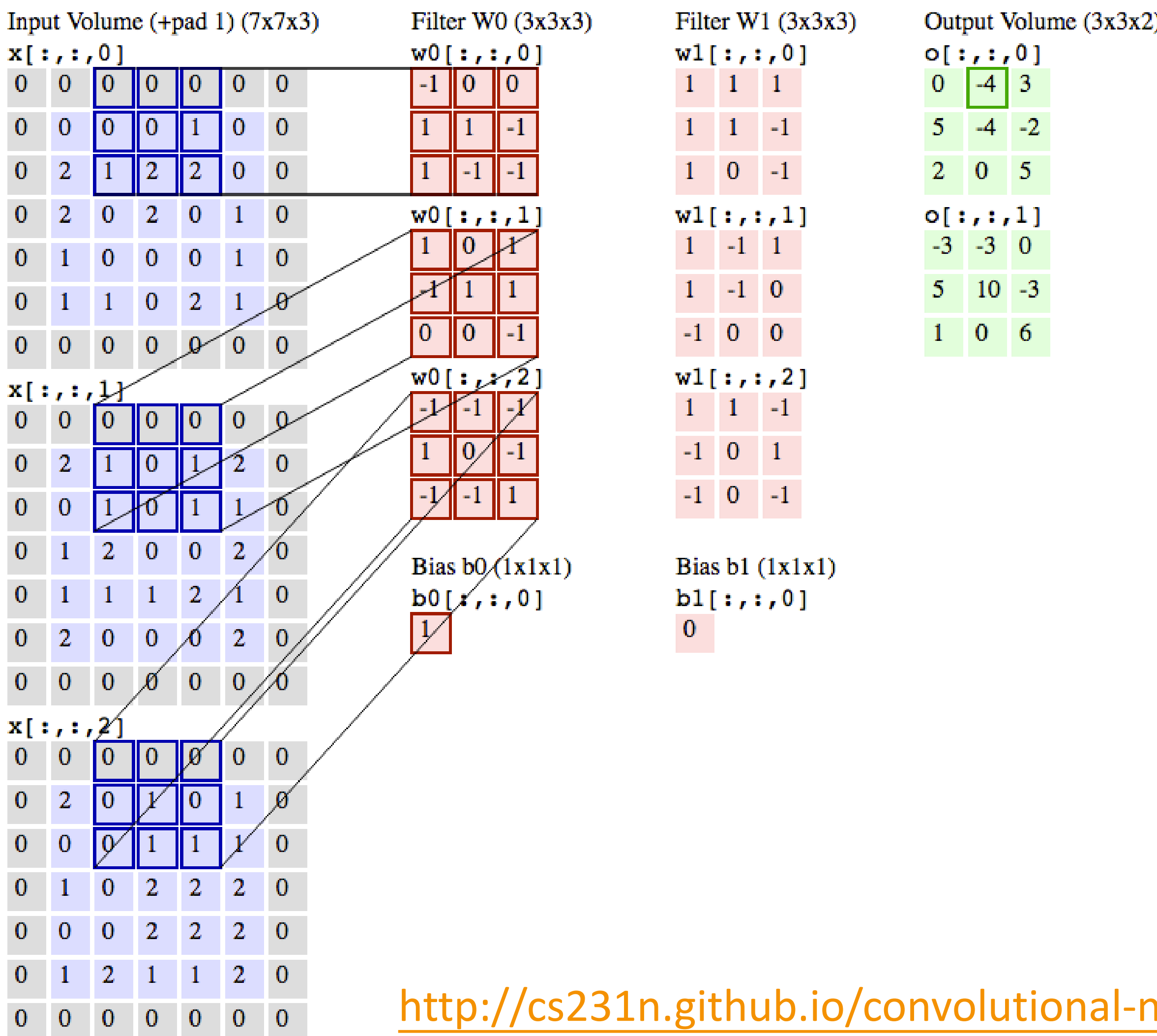





http://cs231n.github.io/convolutional-networks/

# Convolutional Layers

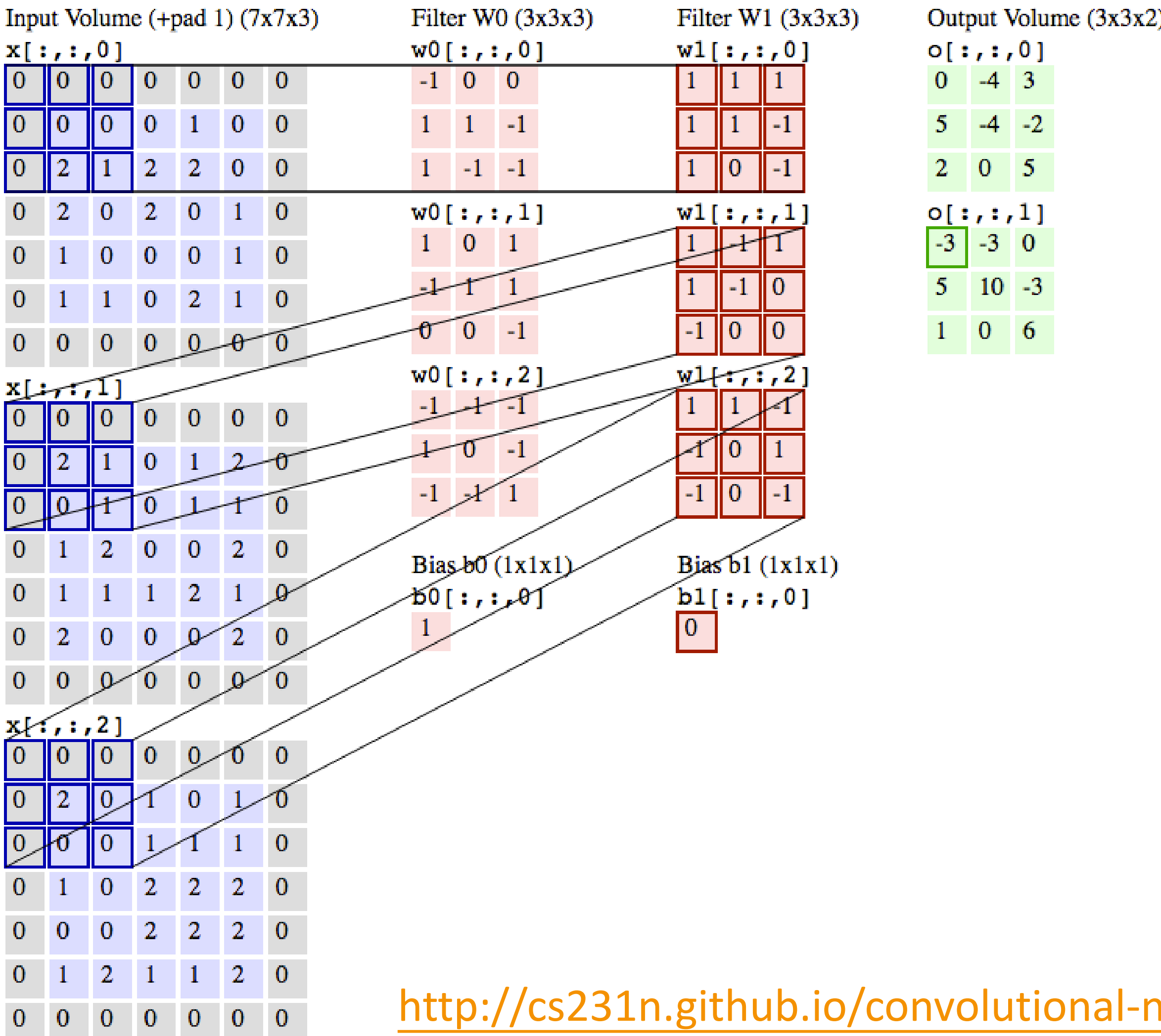





http://cs231n.github.io/convolutional-networks/

# Convolutional Layers

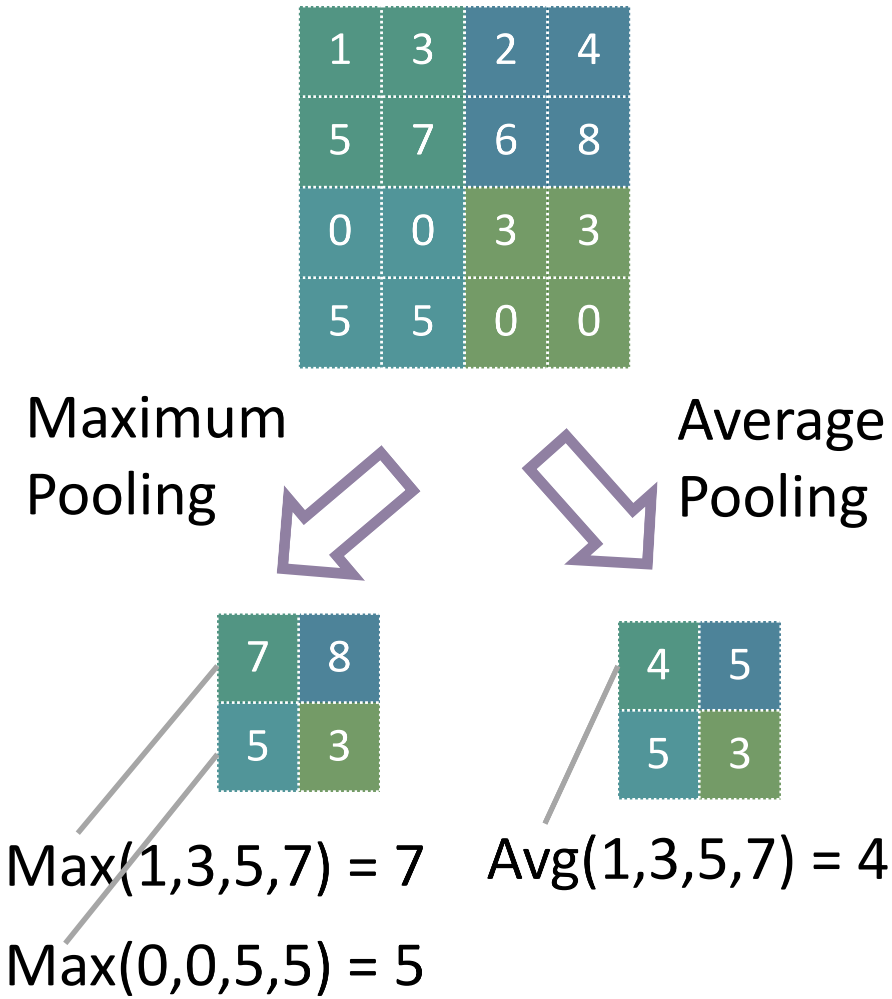


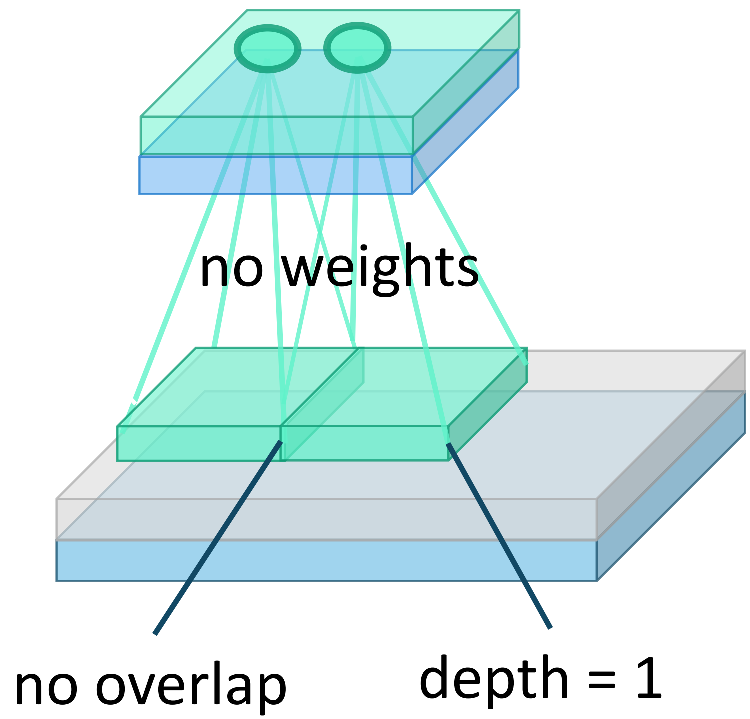



http://cs231n.github.io/convolutional-networks/

# Pooling Layer

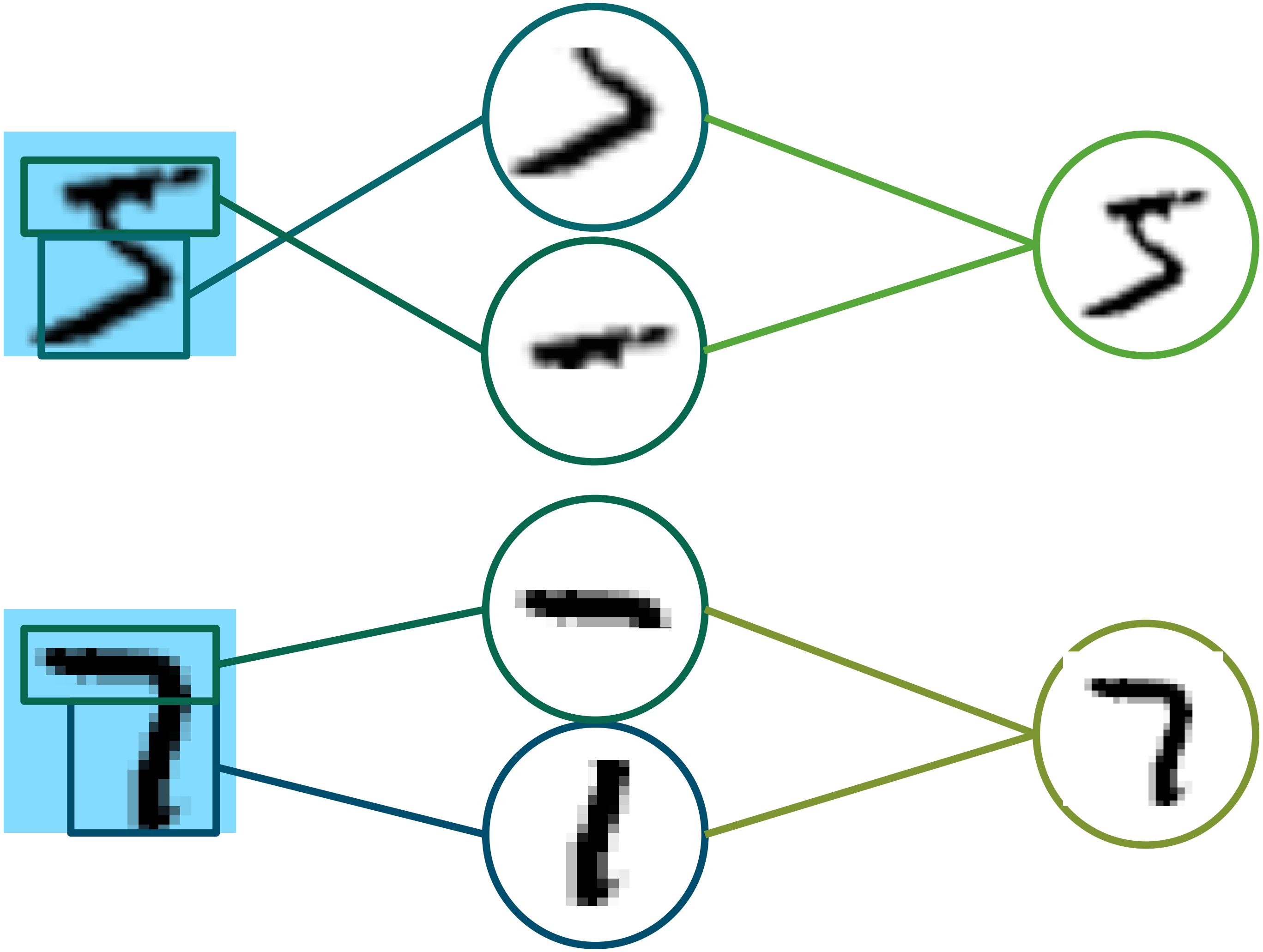




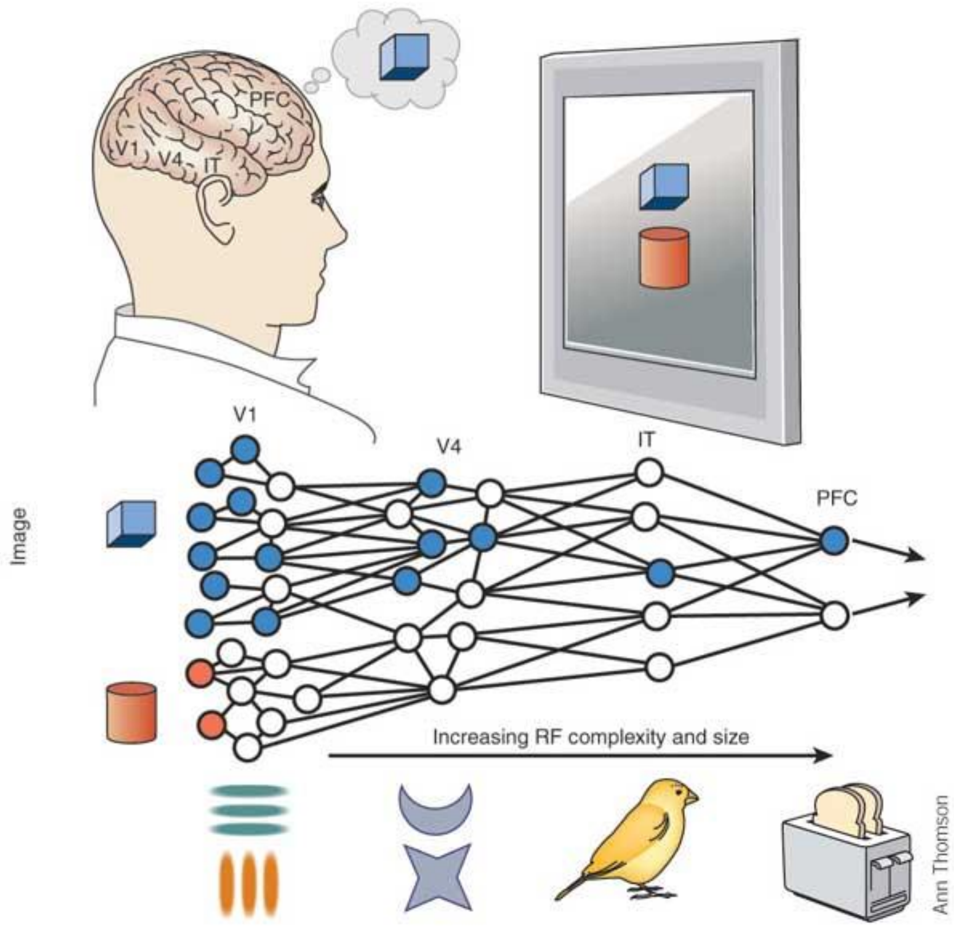
# Why “Deep” Learning?





# Visual Perception of Human

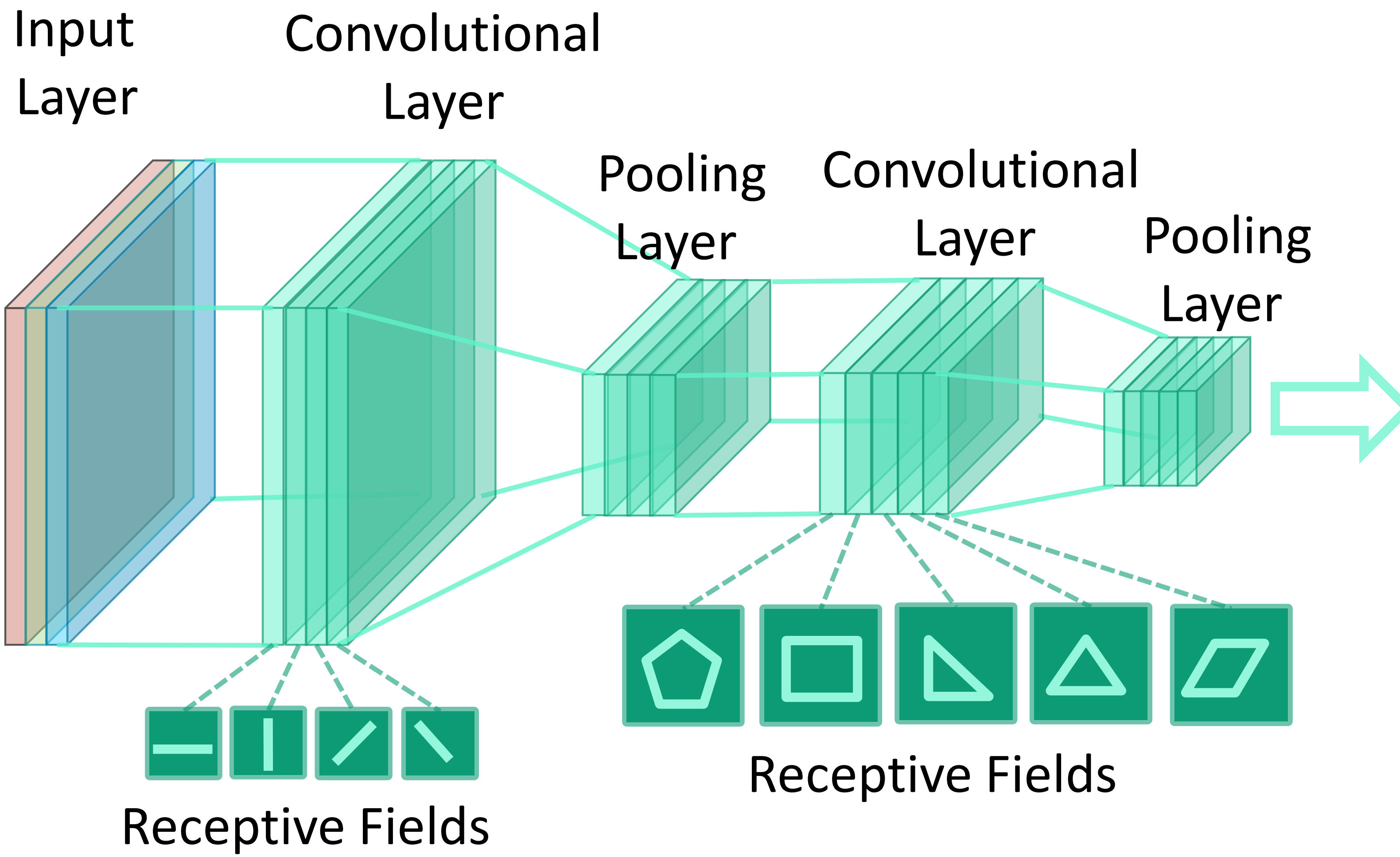





http://www.nature.com/neuro/journal/v8/n8/images/nn0805-975-F1.jpg

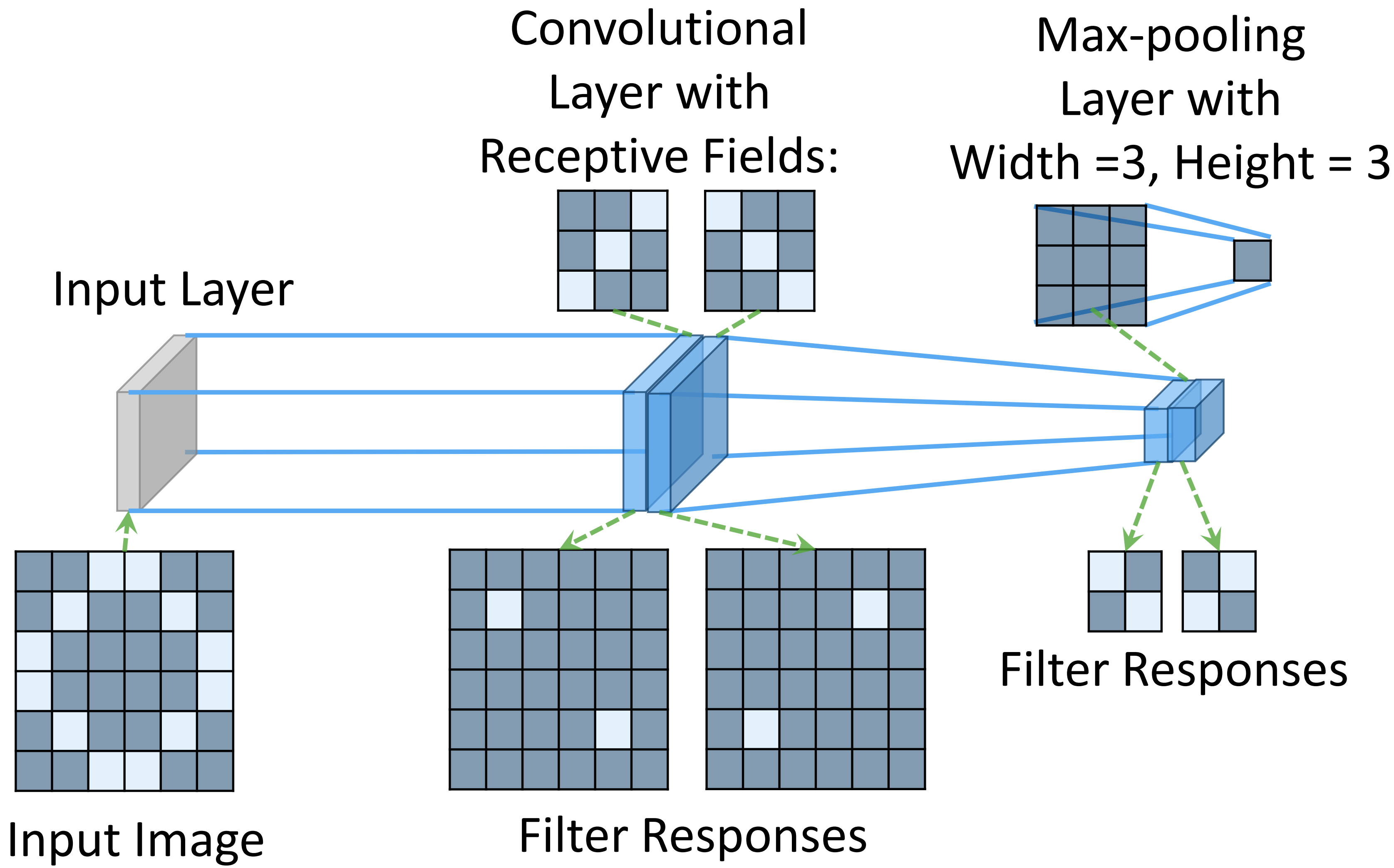
# Visual Perception of Computer





# Visual Perception of Computer

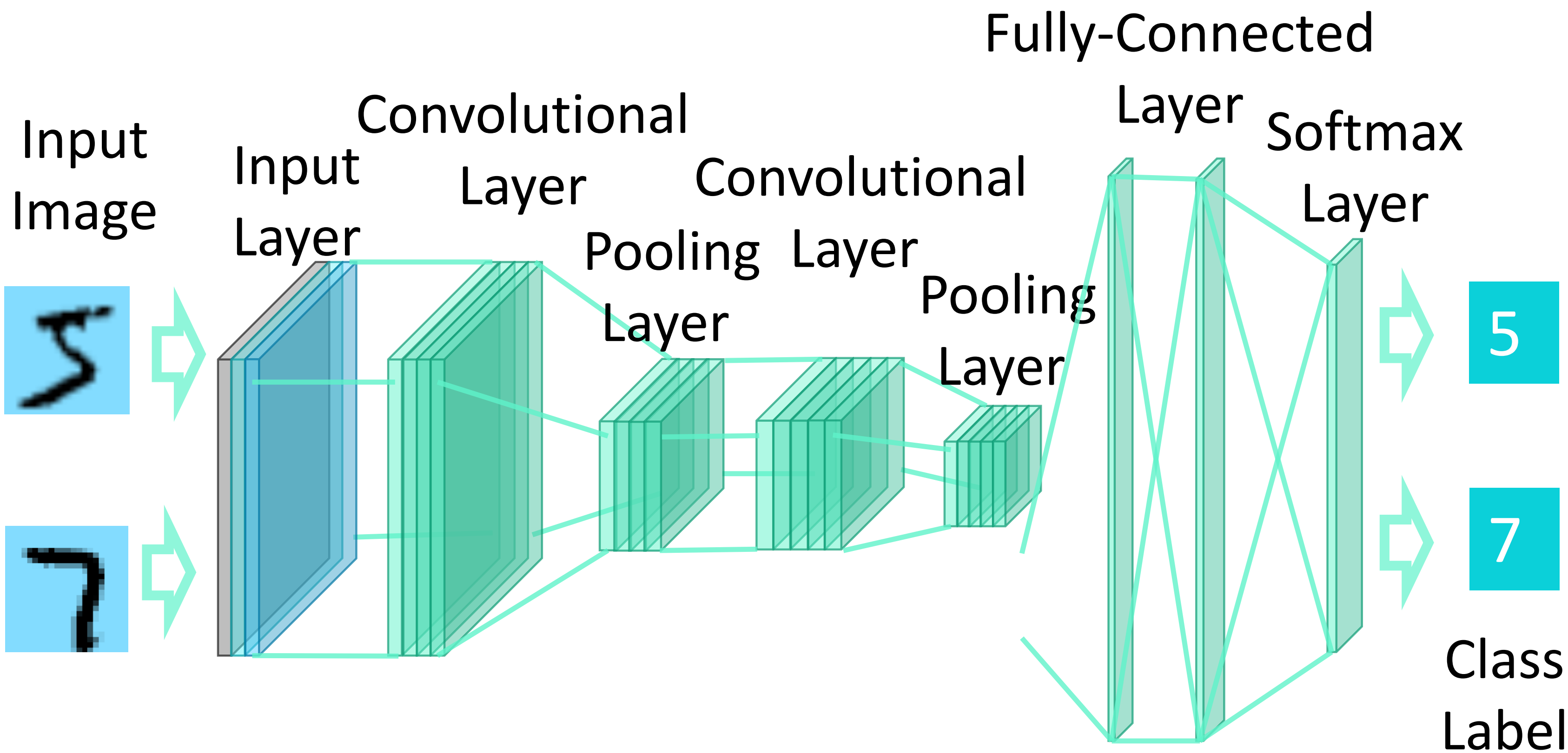




# Fully-Connected Layer



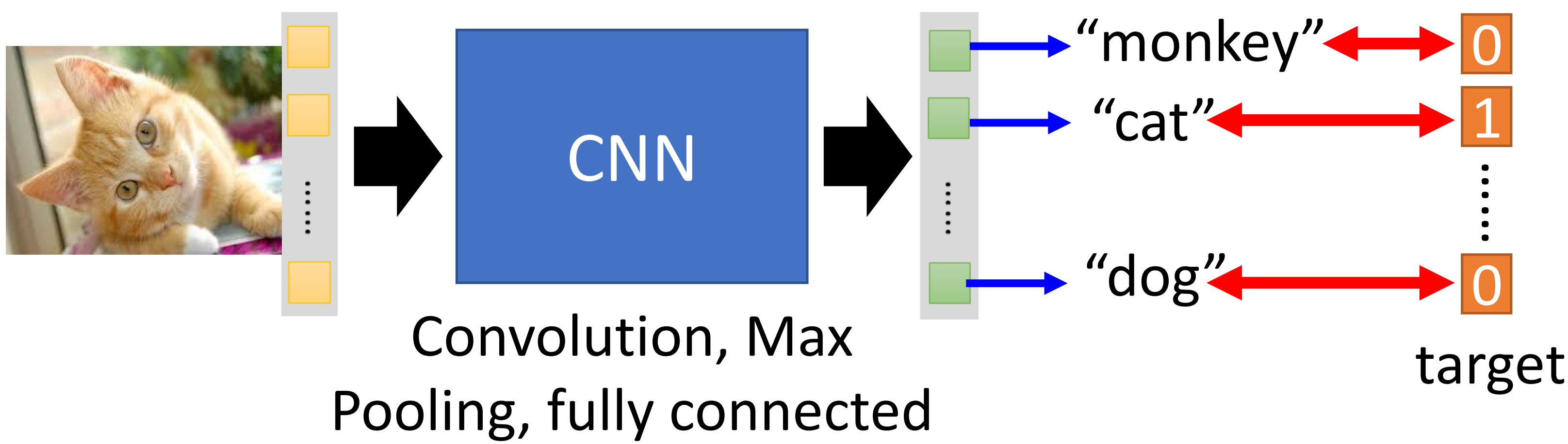
- Fully-Connected Layers : Global feature extraction
- Softmax Layer: Classifier



# Convolutional Neural Network

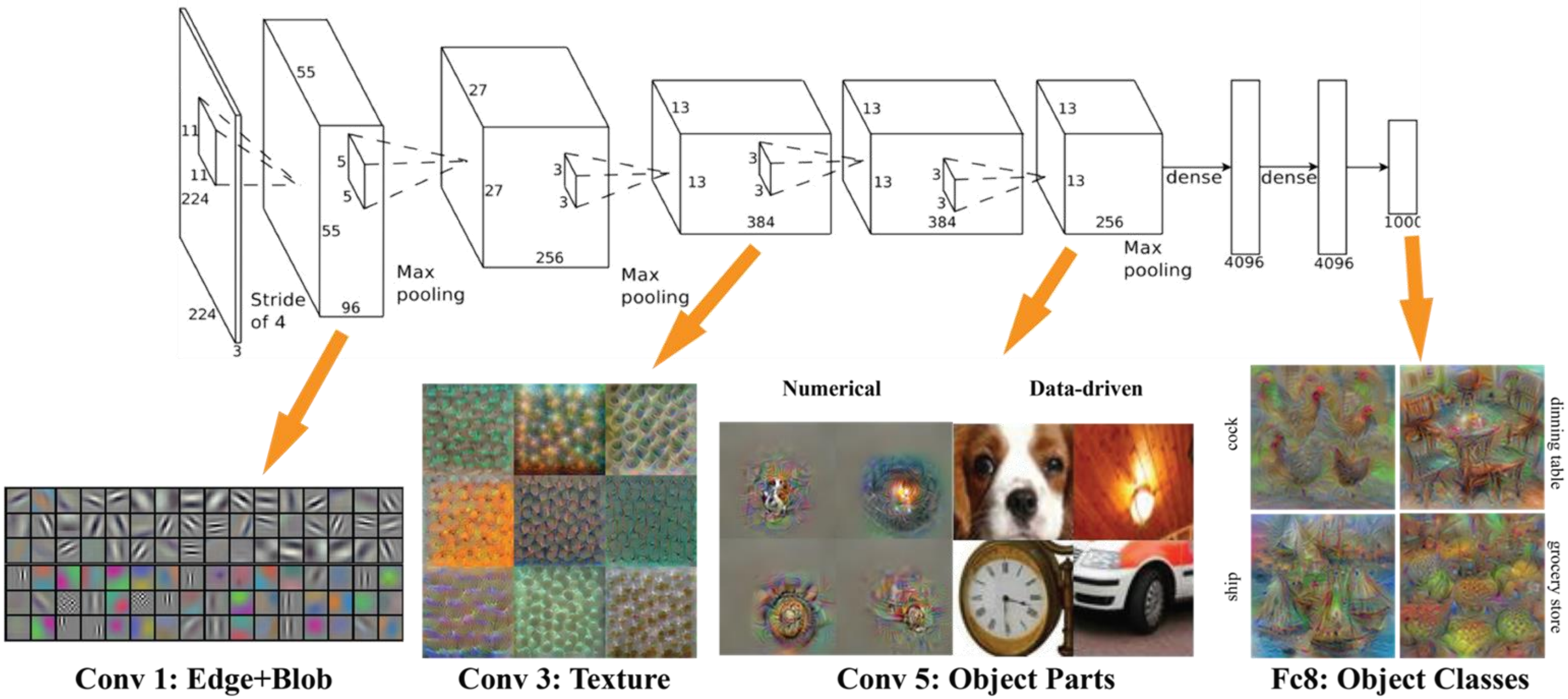


Step 1: Convolutional Neural Network → Step 2: goodness of function → Step 3: pick the best function



# What CNN Learned

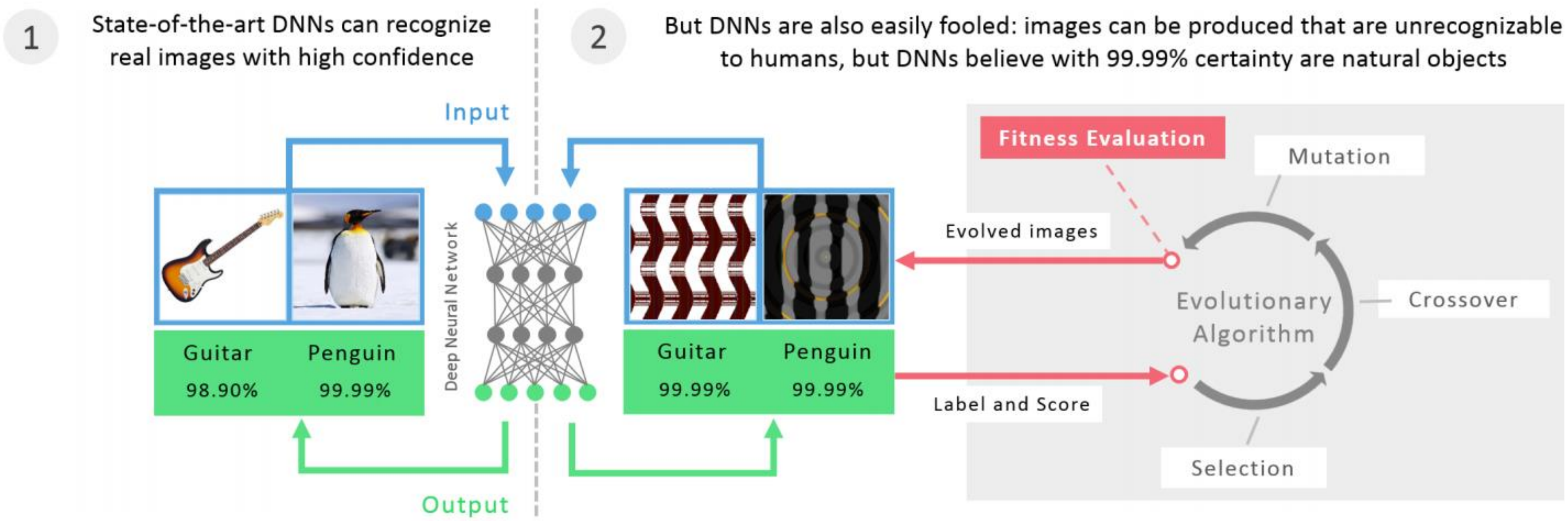
- Alexnet
  - http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf

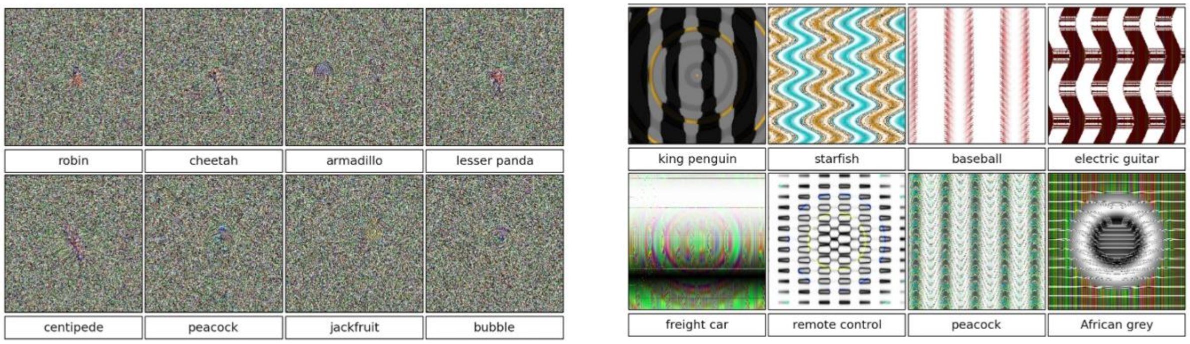



http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

# DNN are easily fooled

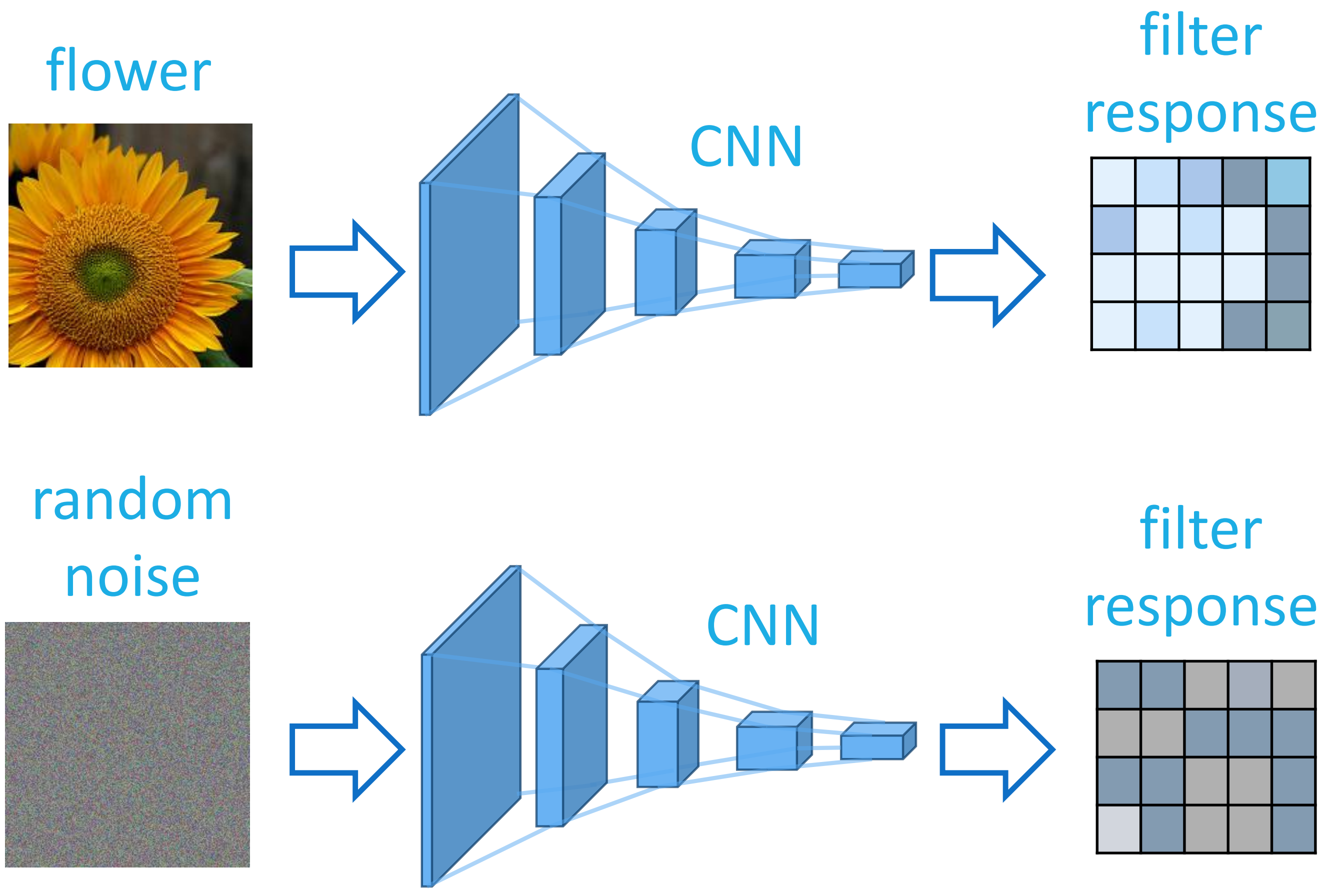




# Visualizing CNN



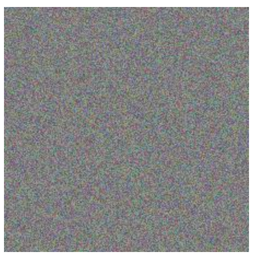


# Gradient Ascent

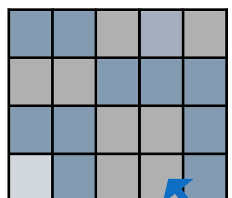


- Magnify the filter response

random noise: $\mathbf{x}$



filter response: $\mathbf{f}$

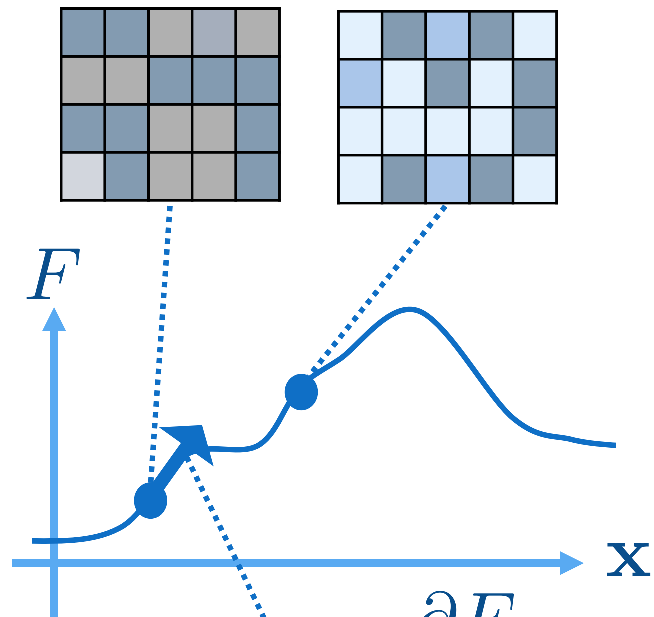


$f_{i,j}$

score: $F = \sum_{i,j} f_{i,j}$

lower score

higher score



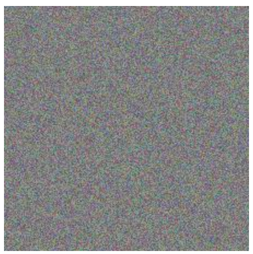

gradient: $\dfrac{\partial F}{\partial \mathbf{x}}$

# Gradient Ascent

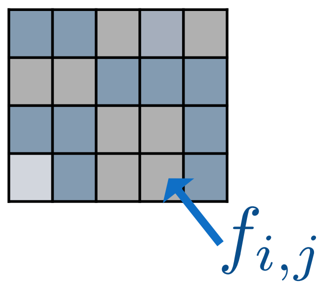


- Magnify the filter response

random noise: $\mathbf{x}$



filter response: $\mathbf{f}$



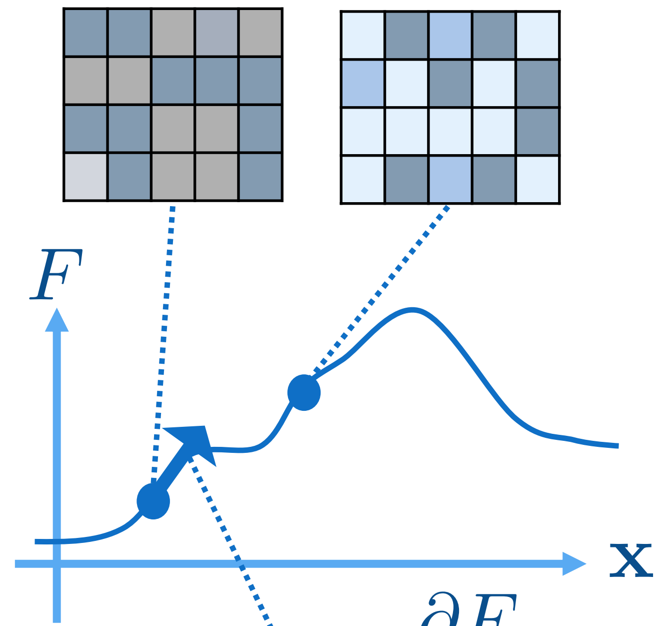

update $\mathbf{x}$

$$\mathbf{x} \leftarrow \mathbf{x} + \eta \frac{\partial F}{\partial \mathbf{x}}$$

learning rate

lower score

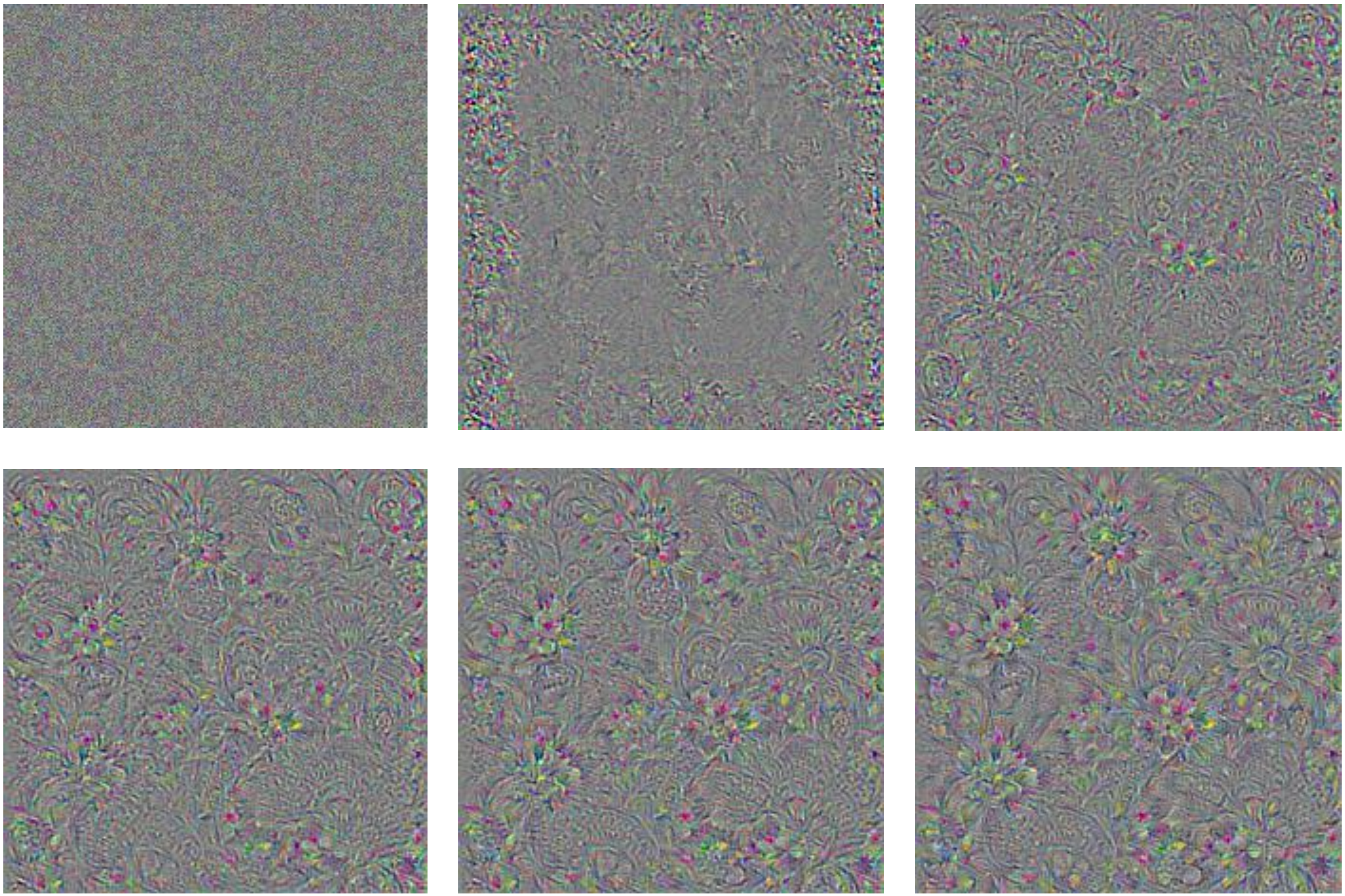
higher score




gradient: $\frac{\partial F}{\partial \mathbf{x}}$

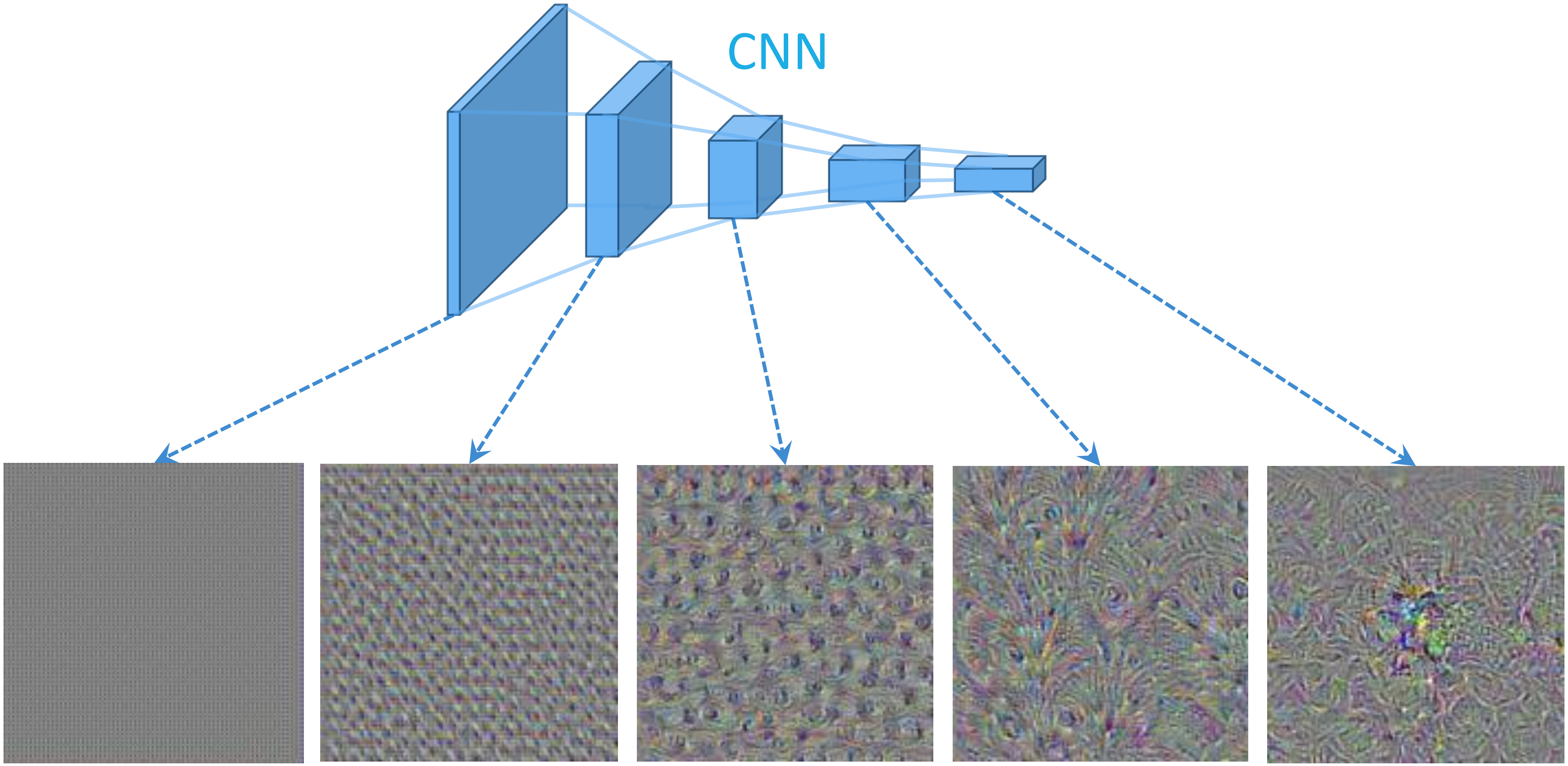
# Gradient Ascent





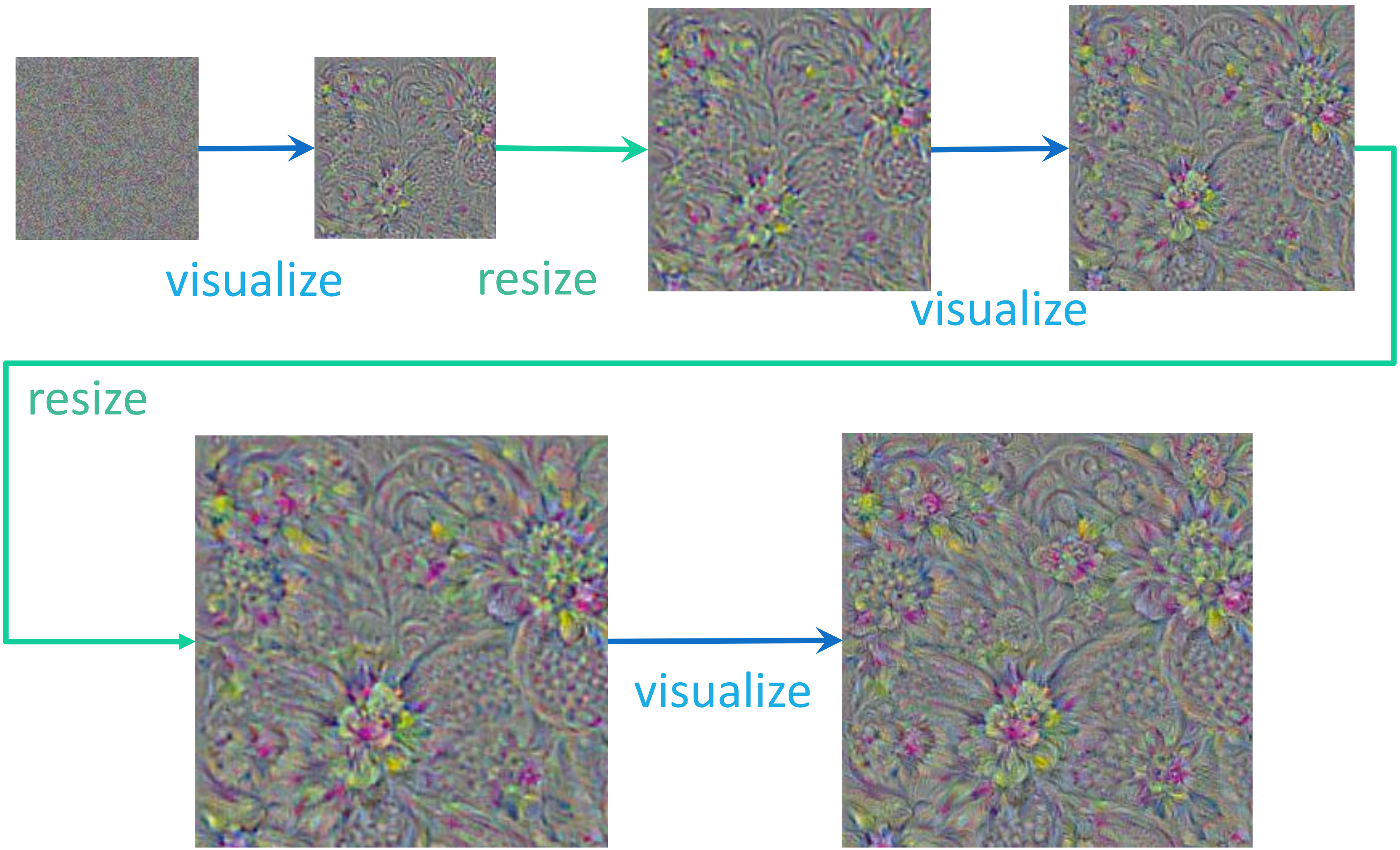
# Different Layers of Visualization





# Multiscale Image Generation

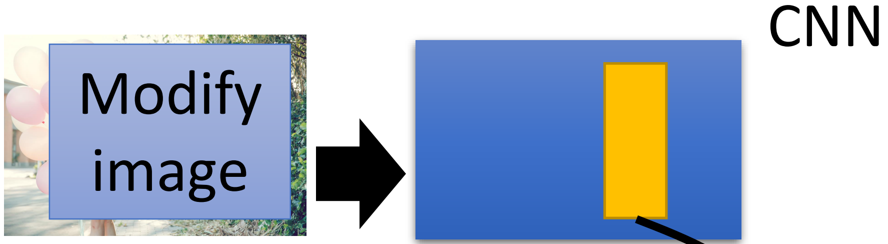




# Multiscale Image Generation



# Deep Dream






- Given a photo, machine adds what it sees ……

$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$


http://deepdreamgenerator.com/

# Deep Dream

http://deepdreamgenerator.com/



- Given a photo, machine adds what it sees ……

# Deep Style

http://deepdreamgenerator.com/



- Given a photo, make its style like famous paintings

# Deep Style

http://deepdreamgenerator.com/

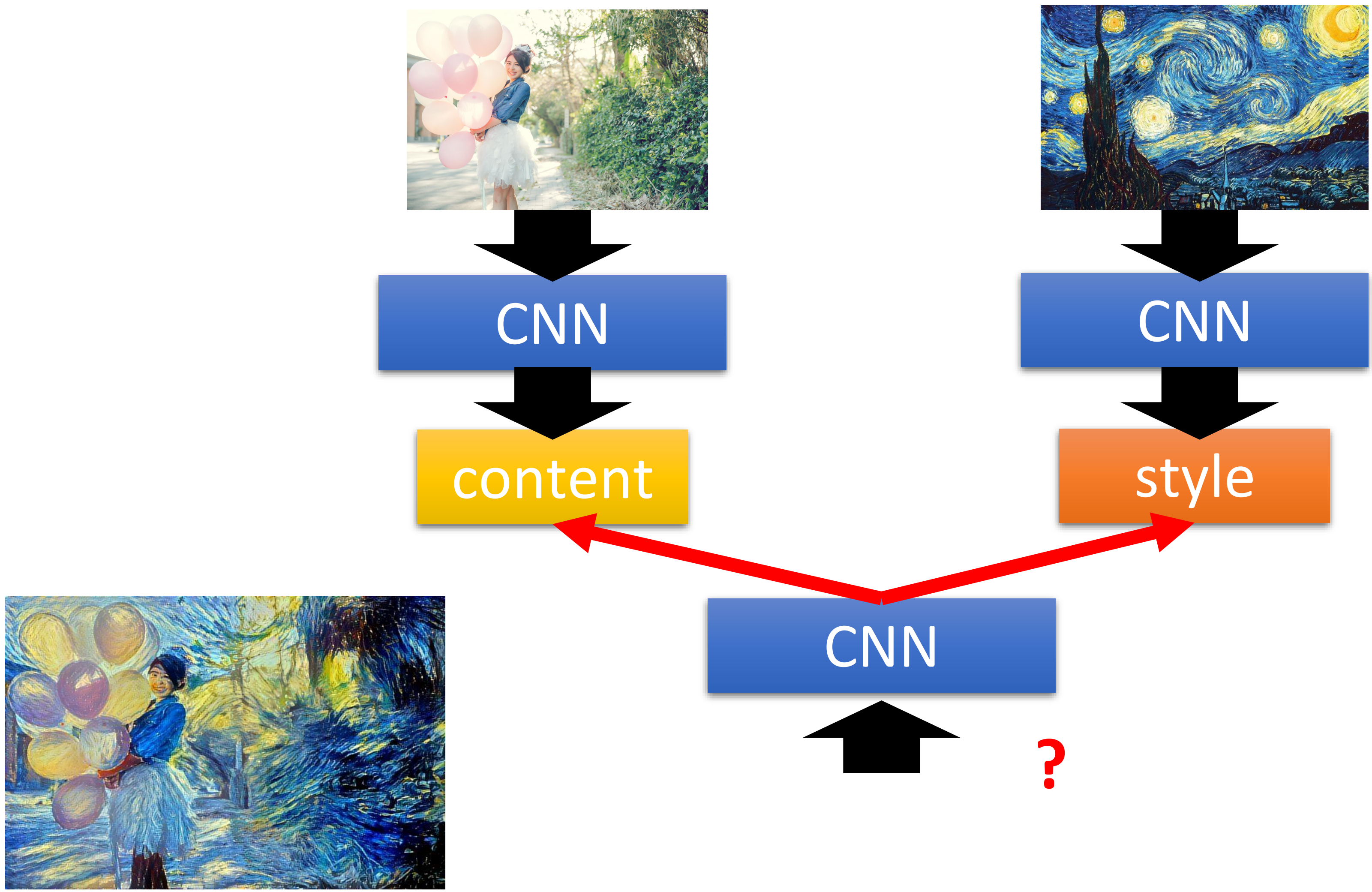


- Given a photo, make its style like famous paintings

# Deep Style

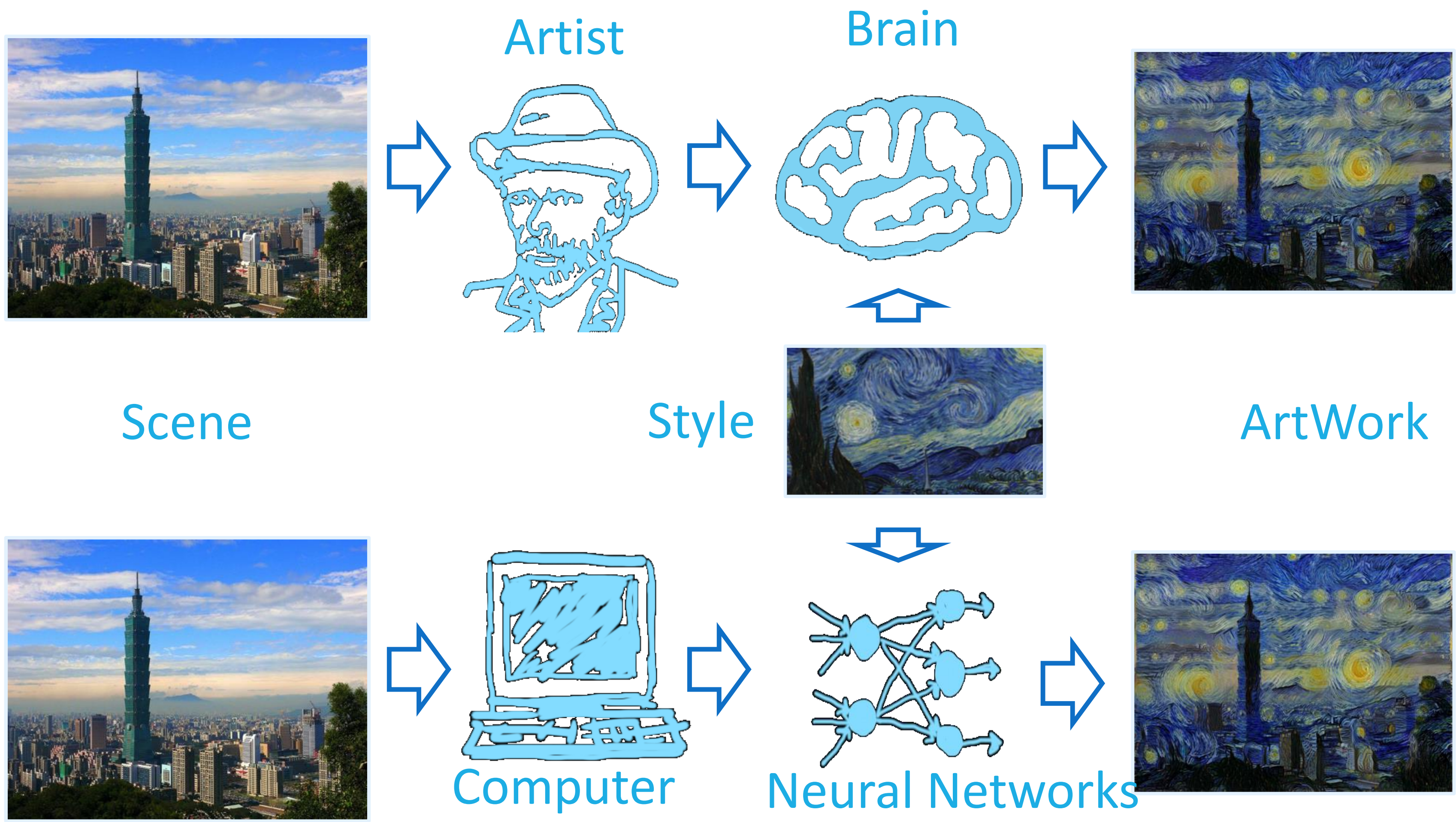
A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576





# Neural Art Mechanism





# Go Playing



19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

none: 0

Fully-connected feedforward network can be used

But CNN performs much better.

# More Application: Playing Go



Training: record of previous plays 黑: 5之五 → 白: 天元 → 黑: 五之5 …


CNN

Target:
“天元” = 1
else = 0


CNN

Target:
“五之 5” = 1
else = 0

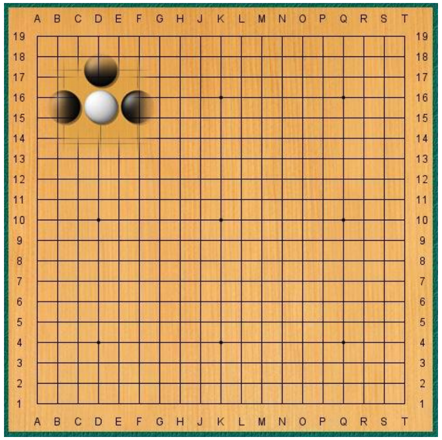
# Why CNN for playing Go?

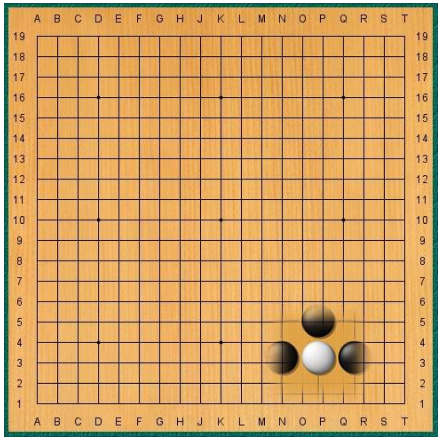


- Some patterns are much smaller than the whole image

Alpha Go uses 5 x 5 for first layer

- The same patterns appear in different regions





# Why CNN for playing Go?



- Subsampling the pixels will not change the object

Max Pooling

How to explain this???

**Neural network architecture.** The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a $23 \times 23$ image, then convolves $k$ filters of kernel size $5 \times 5$ 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 $21 \times 21$ image, then convolves $k$ filters of kernel size $3 \times 3$ with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size $1 \times 1$ with strid … max func-

tion. The … Extended

Alpha Go does not use Max Pooling ……

Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

# PART II: Variants of Neural Networks

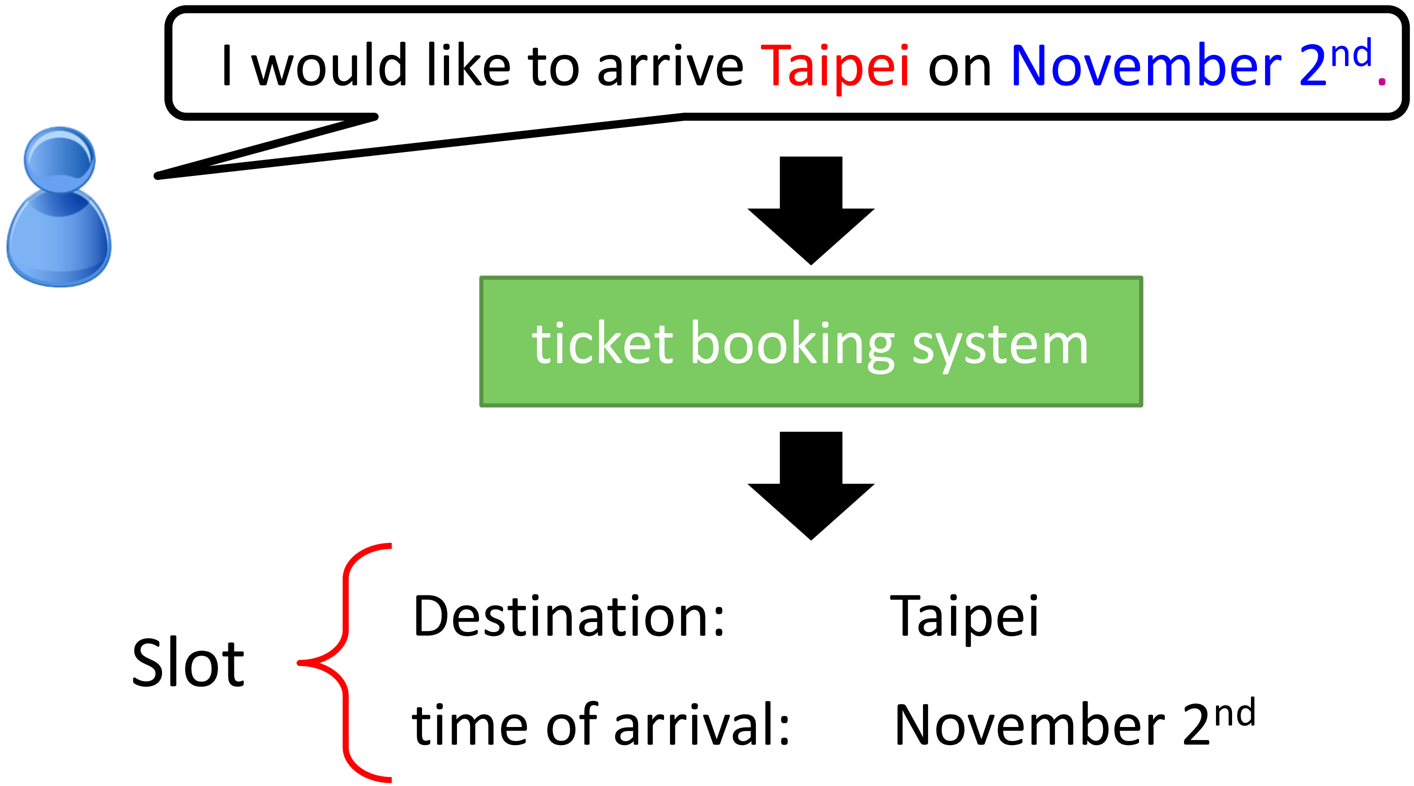


- Convolutional Neural Network (CNN)
- **Recurrent Neural Network (RNN)**

Neural Network with Memory

# Example Application

- Slot Filling



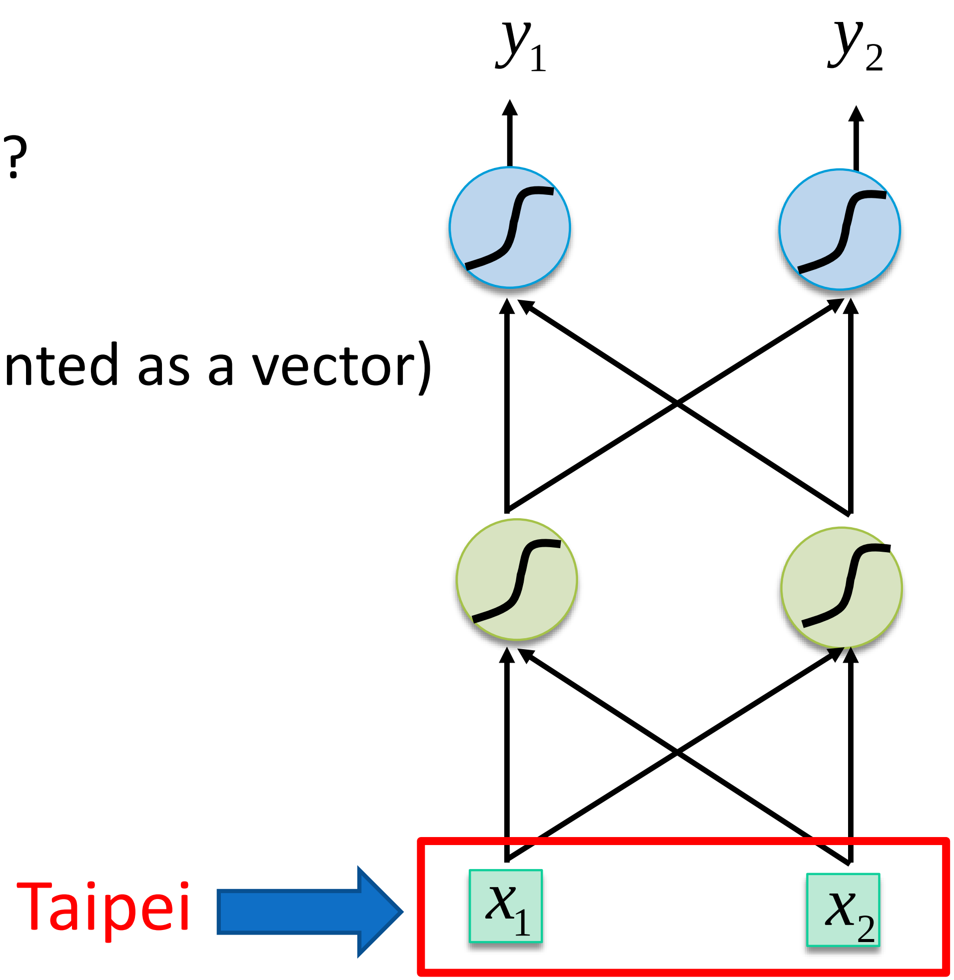
# Example Application



Solving slot filling by feedforward network?

Input: a word

(Each word is represented as a vector)



# 1-of-N encoding



How to represent each word as a vector?

***1-of-N Encoding*** lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [ 1 0 0 0 0]

bag = [ 0 1 0 0 0]

cat = [ 0 0 1 0 0]

dog = [ 0 0 0 1 0]

elephant = [ 0 0 0 0 1]

# Example Application



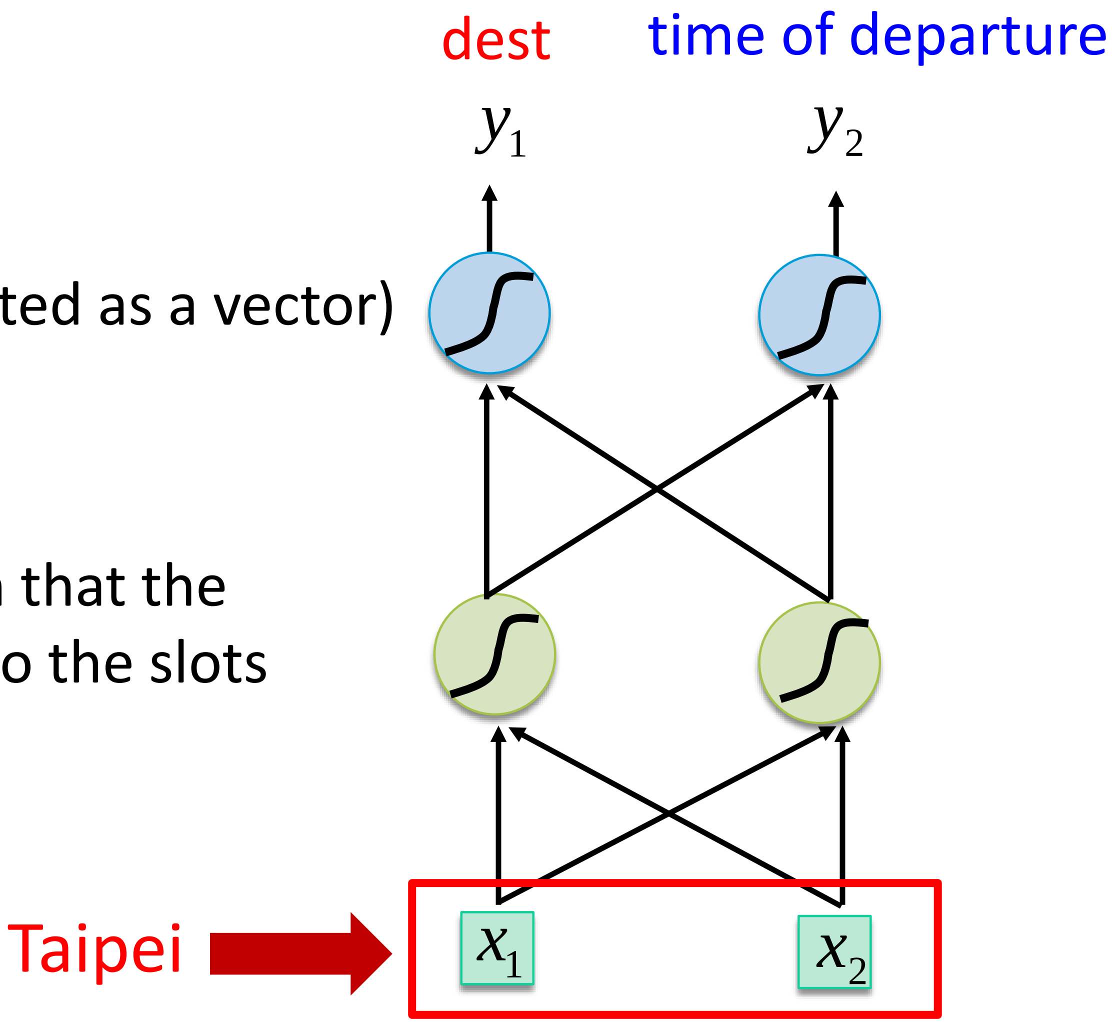
Solving slot filling by feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

probability distribution that the input word belonging to the slots



# Example Application



arrive Taipei on November 2nd

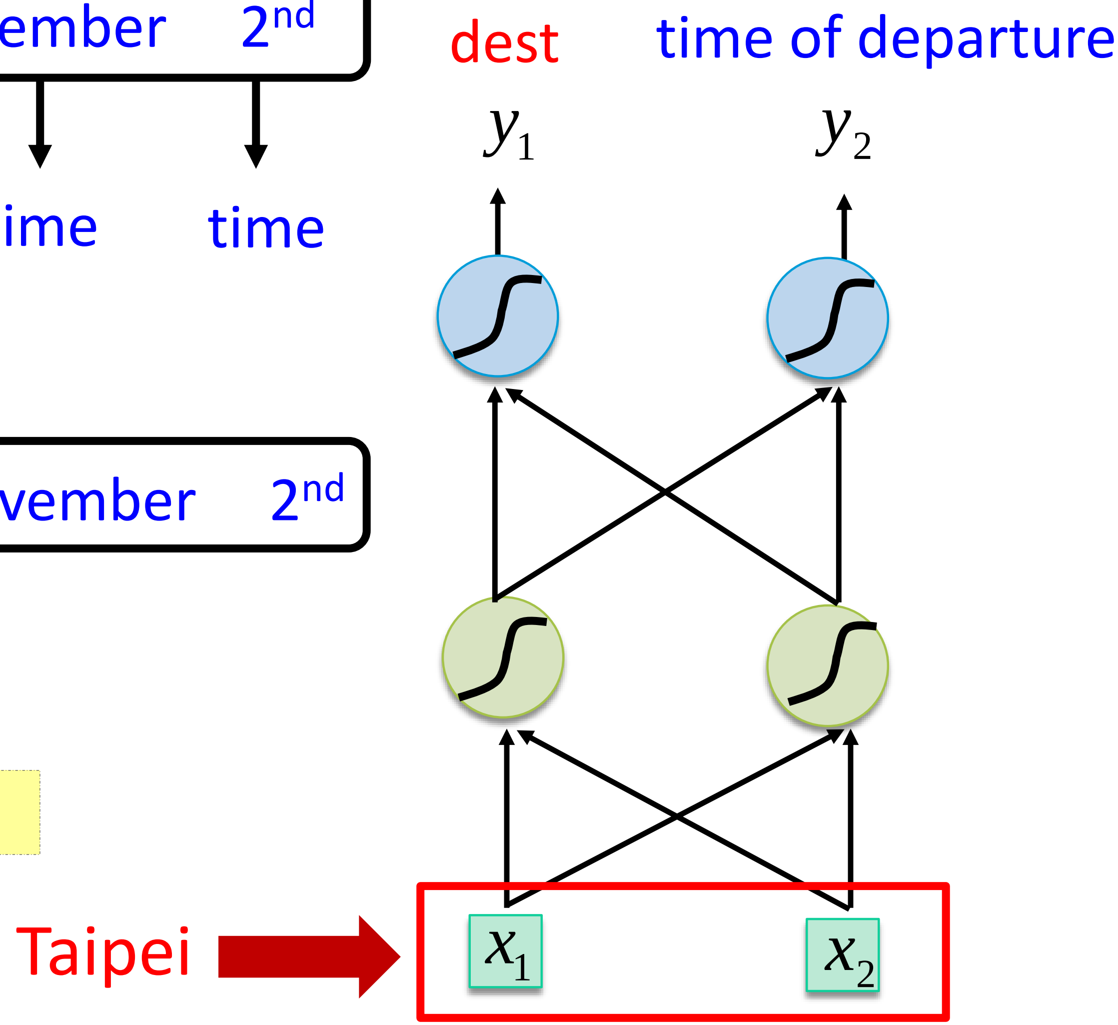
other dest other time time

Problem?

leave Taipei on November 2nd

place of departure

Neural network needs memory!

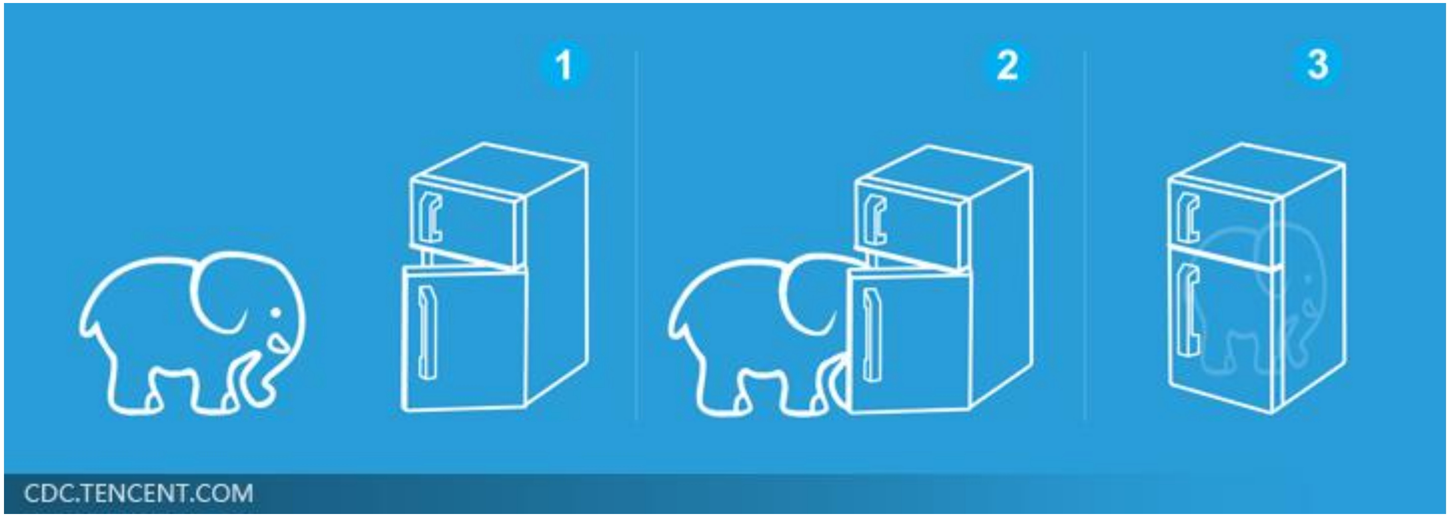


# Three Steps for Deep Learning



Step 1: Recurrent Neural Network → Step 2: goodness of function → Step 3: pick the best function

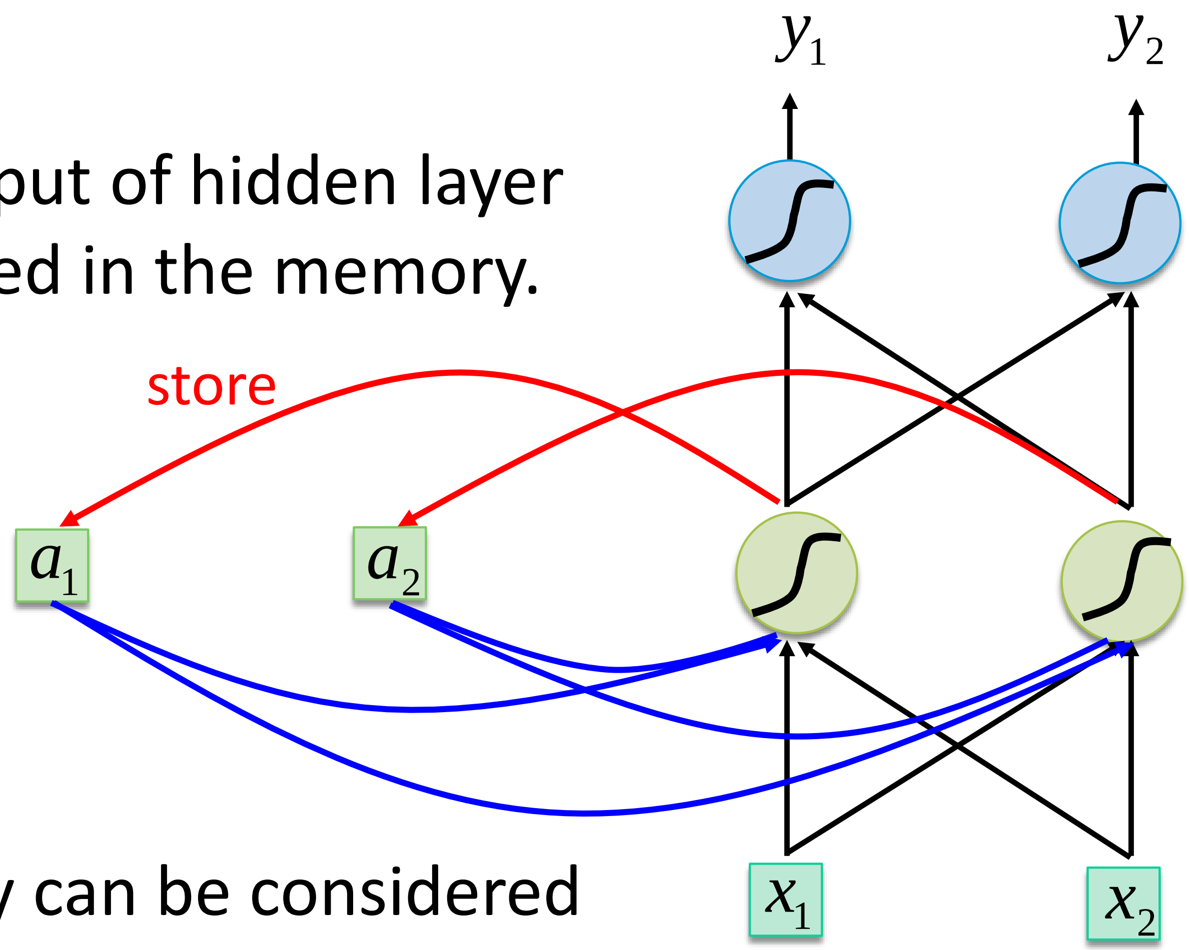
Deep Learning is so simple ……



# Recurrent Neural Network (RNN)



The output of hidden layer are stored in the memory.




Memory can be considered as another input.

# RNN

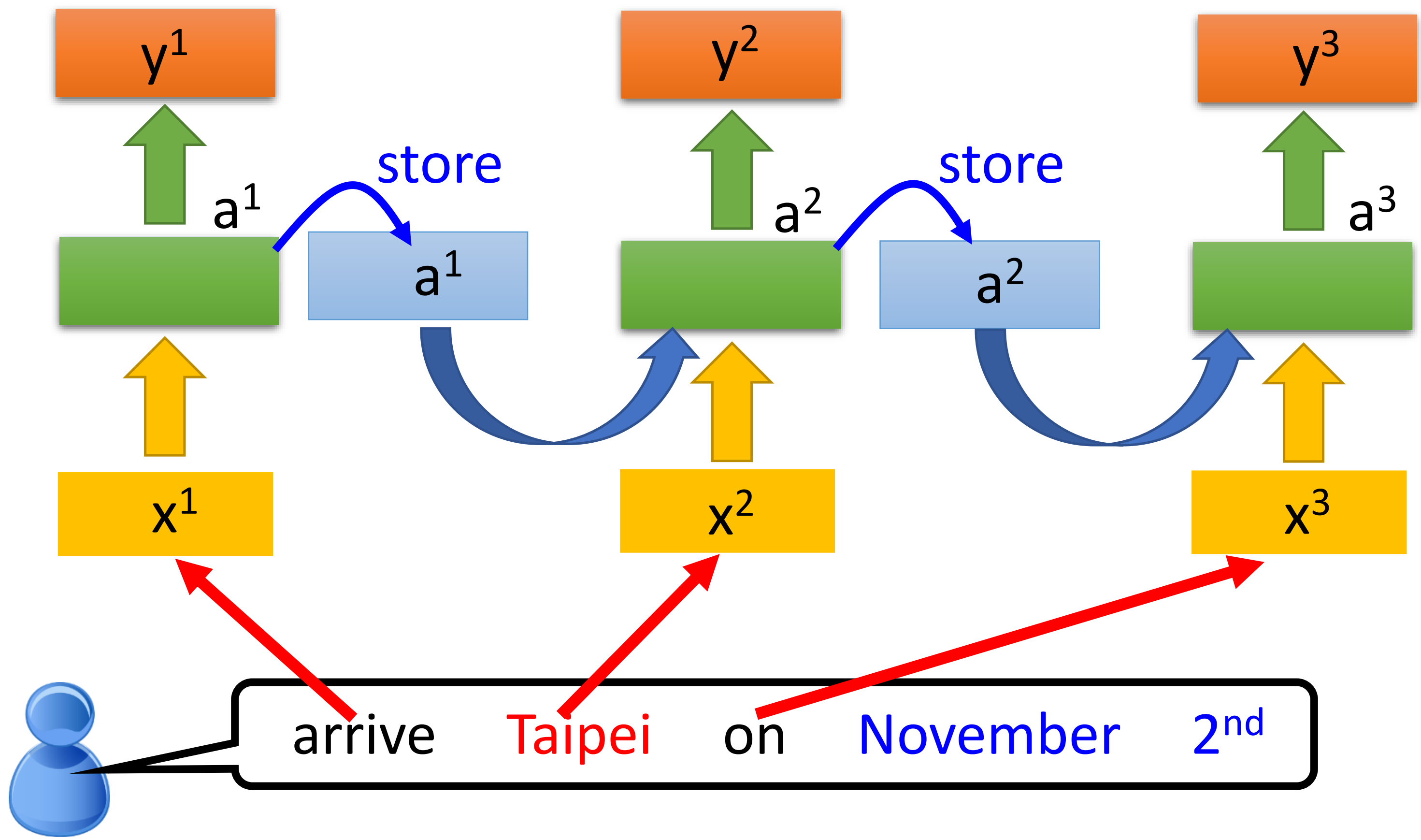
The same network is used again and again.



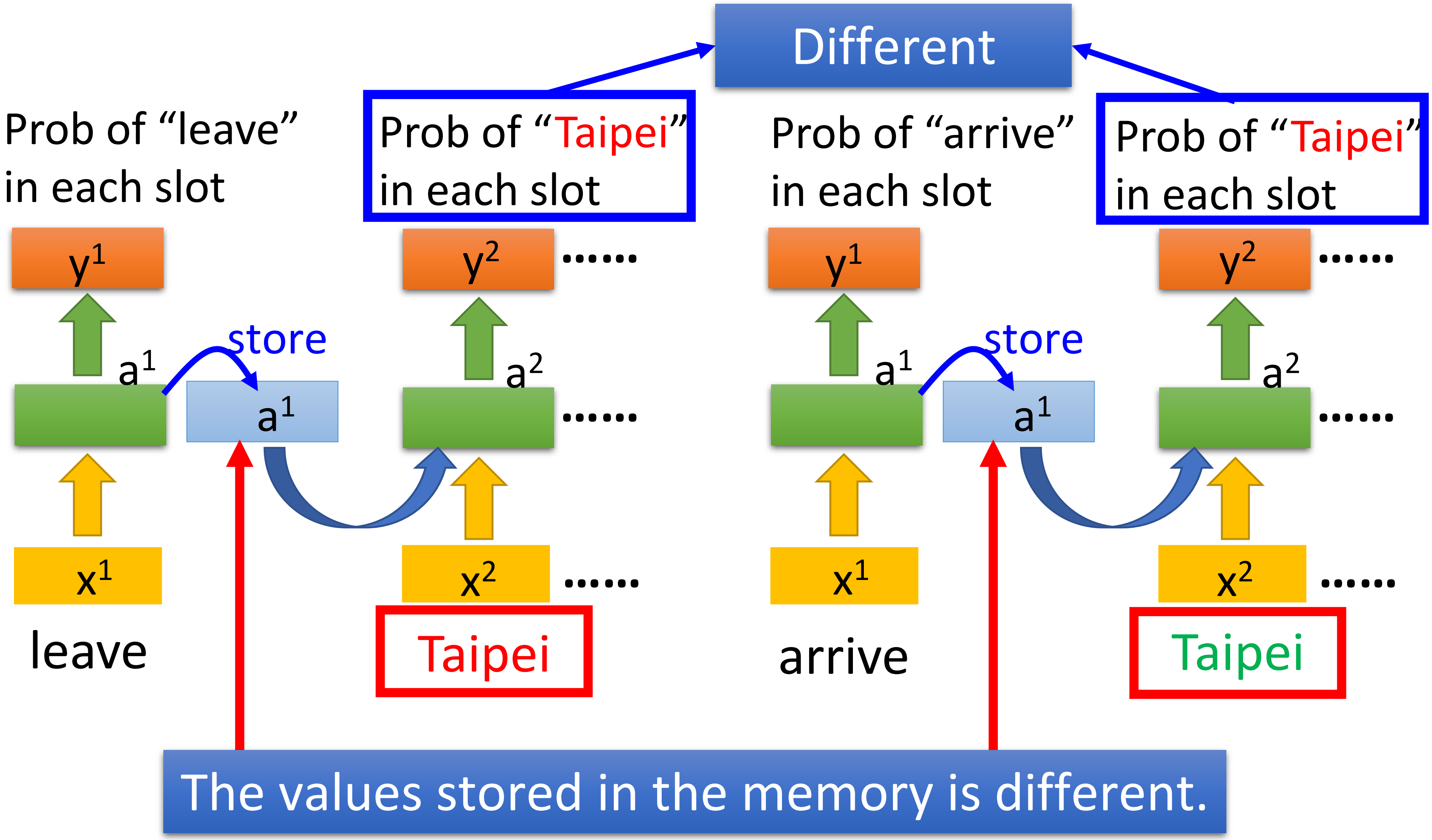
Probability of "arrive" in each slot

Probability of "Taipei" in each slot

Probability of "on" in each slot

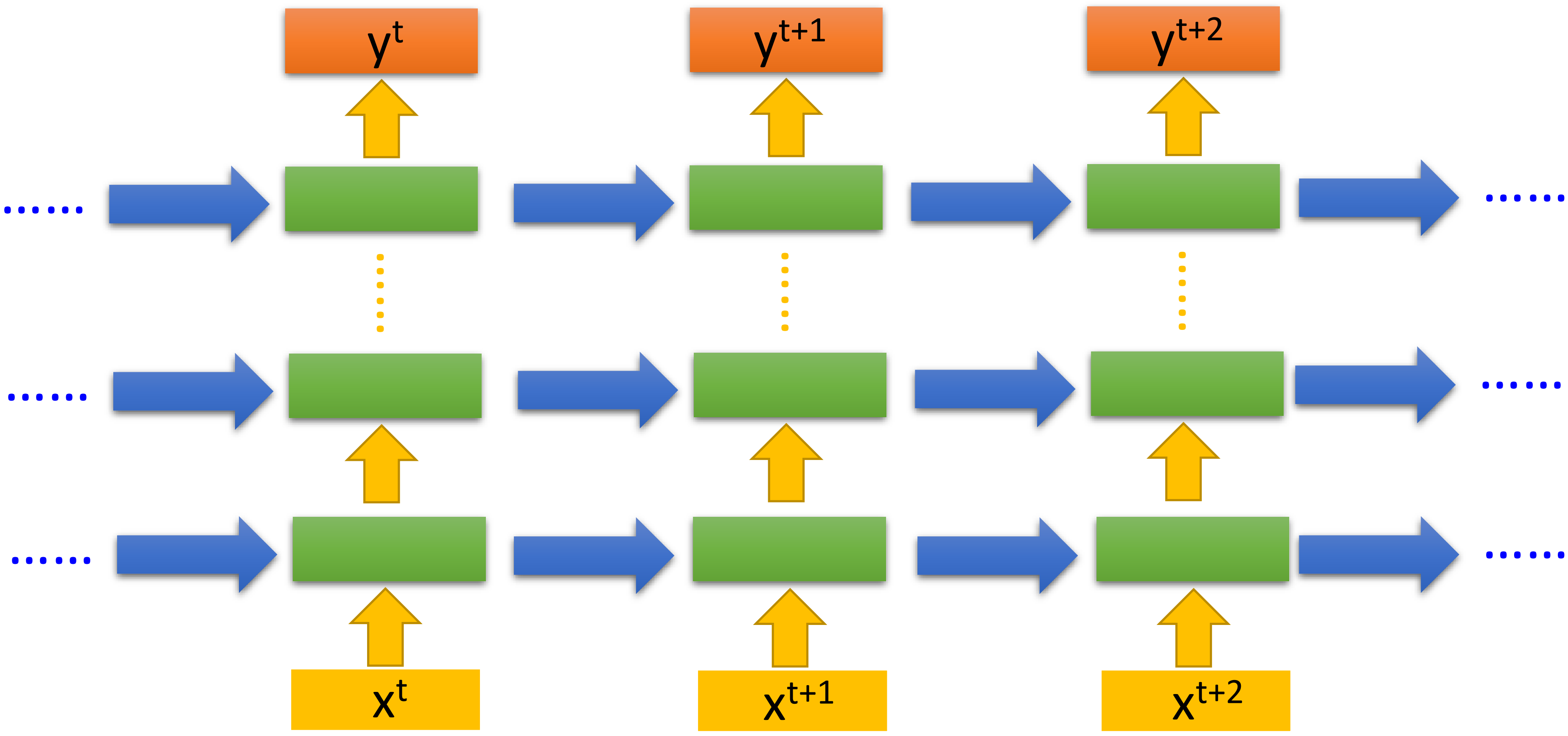


# RNN


Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
y1
y2
......
y1
y2
......
store
a1
a1
a2
......
store
a1
a1
a2
......
x1
x2
......
x1
x2
......
leave
Taipei
arrive
Taipei
The values stored in the memory is different.

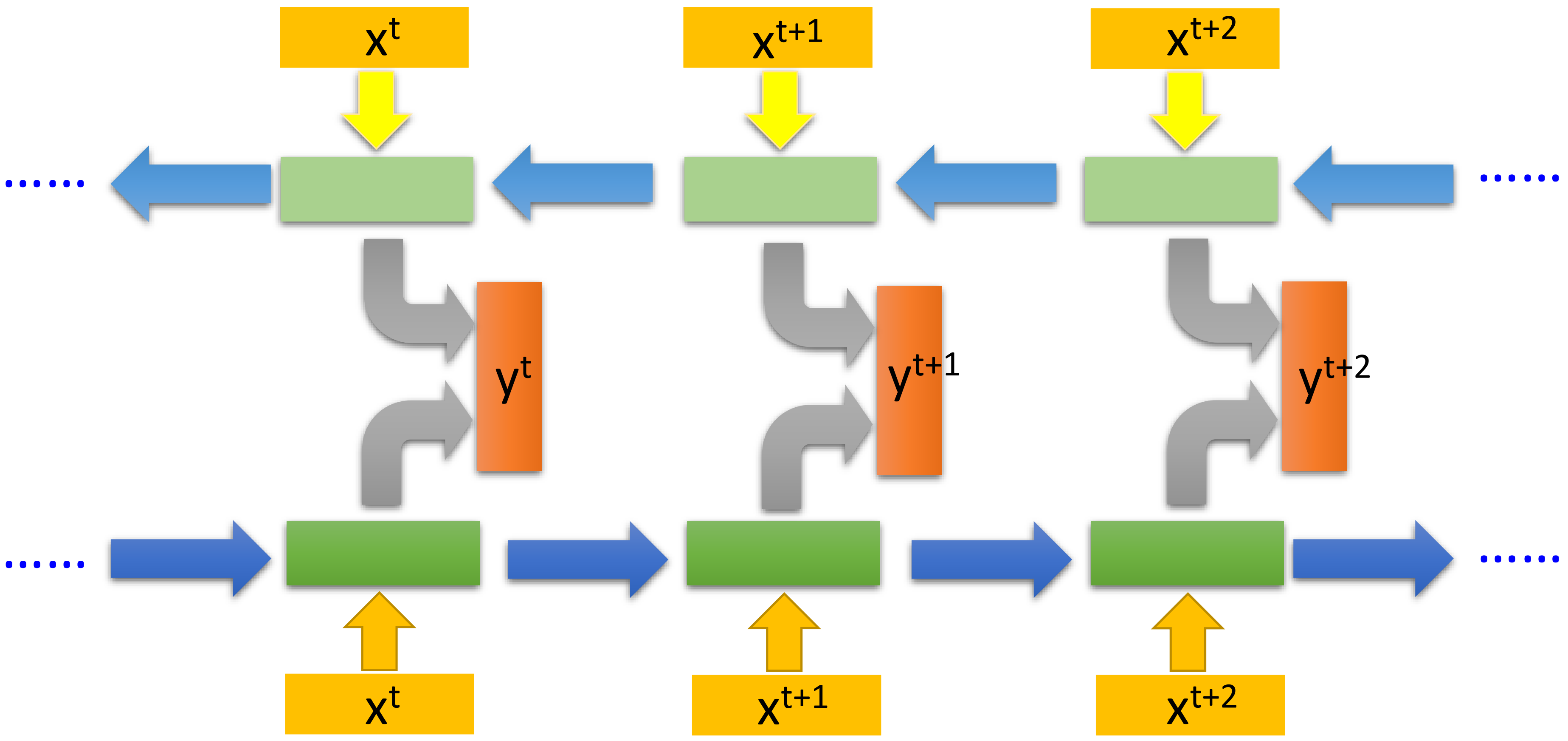
# Deep RNN





# Bidirectional RNN





# RNN

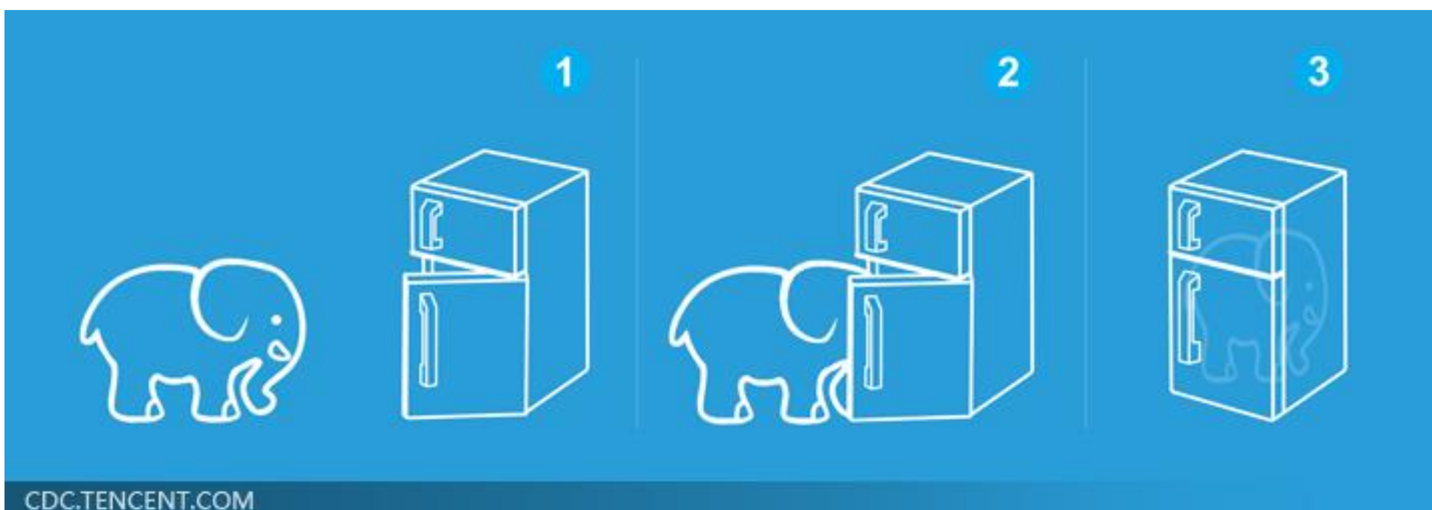


Step 1: Recurrent Neural Network

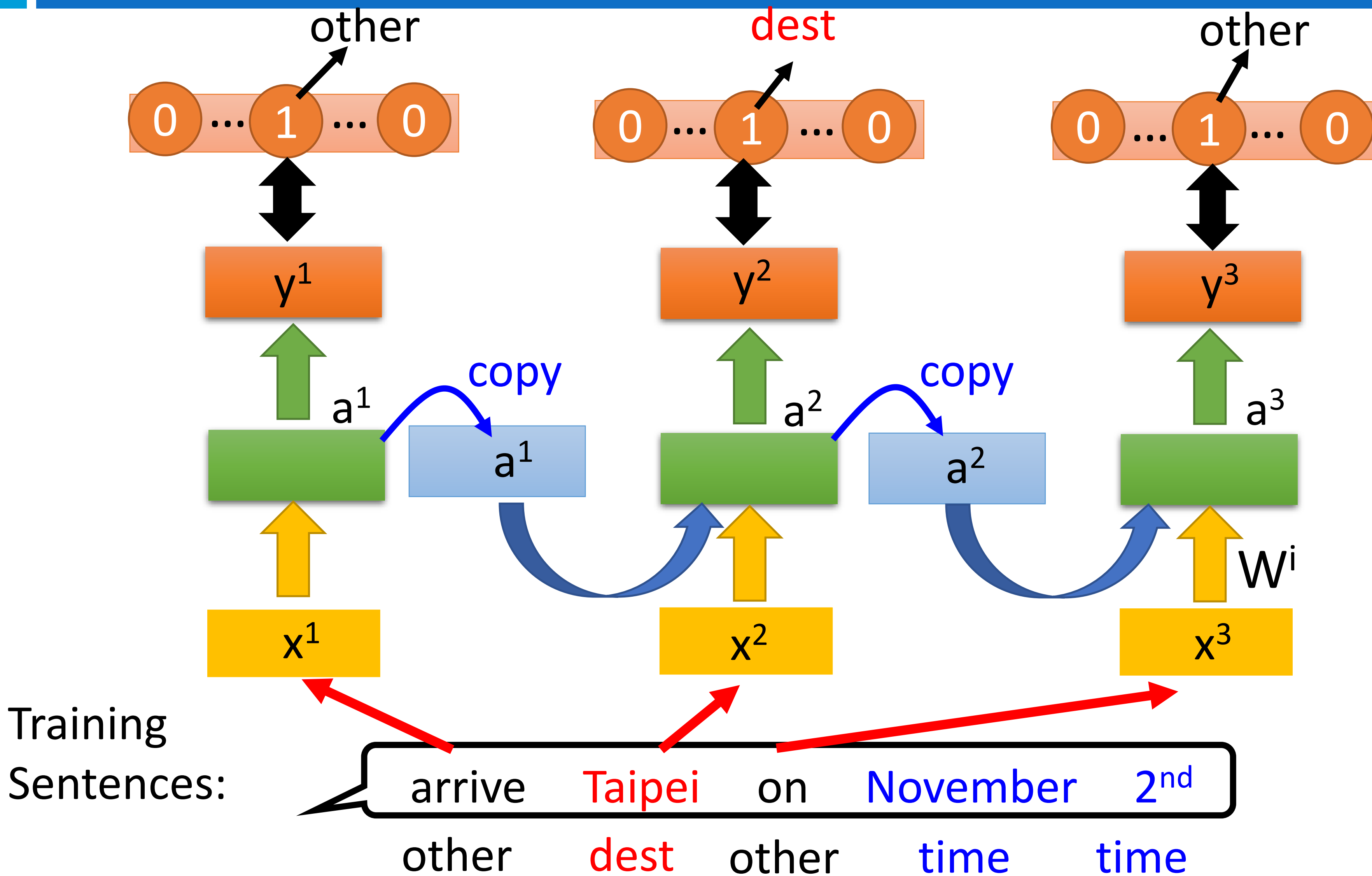
Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……

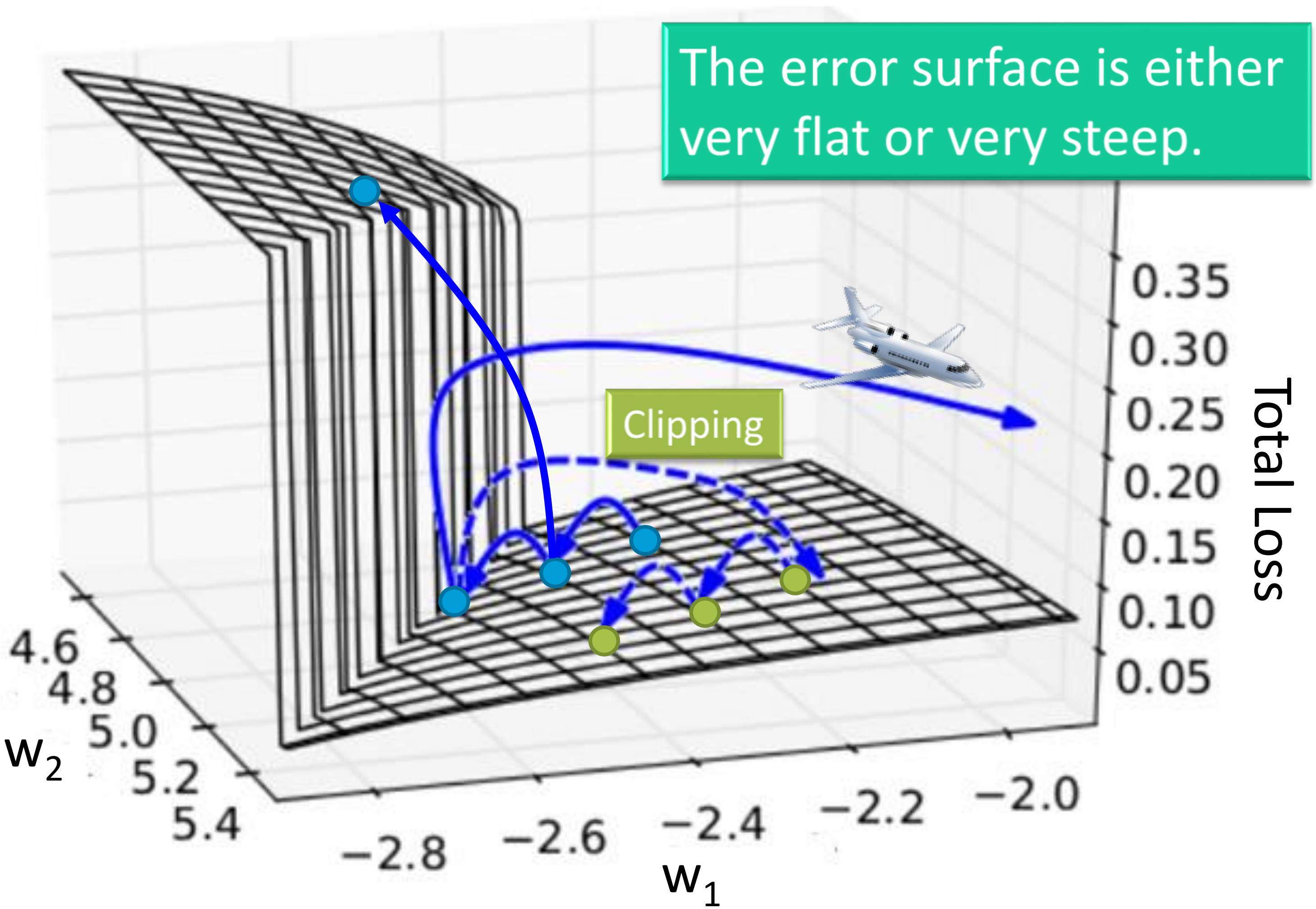


# Learning Target



# Rough Error Surface






[Razvan Pascanu, ICML'13]

# Rough Error Surface



$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

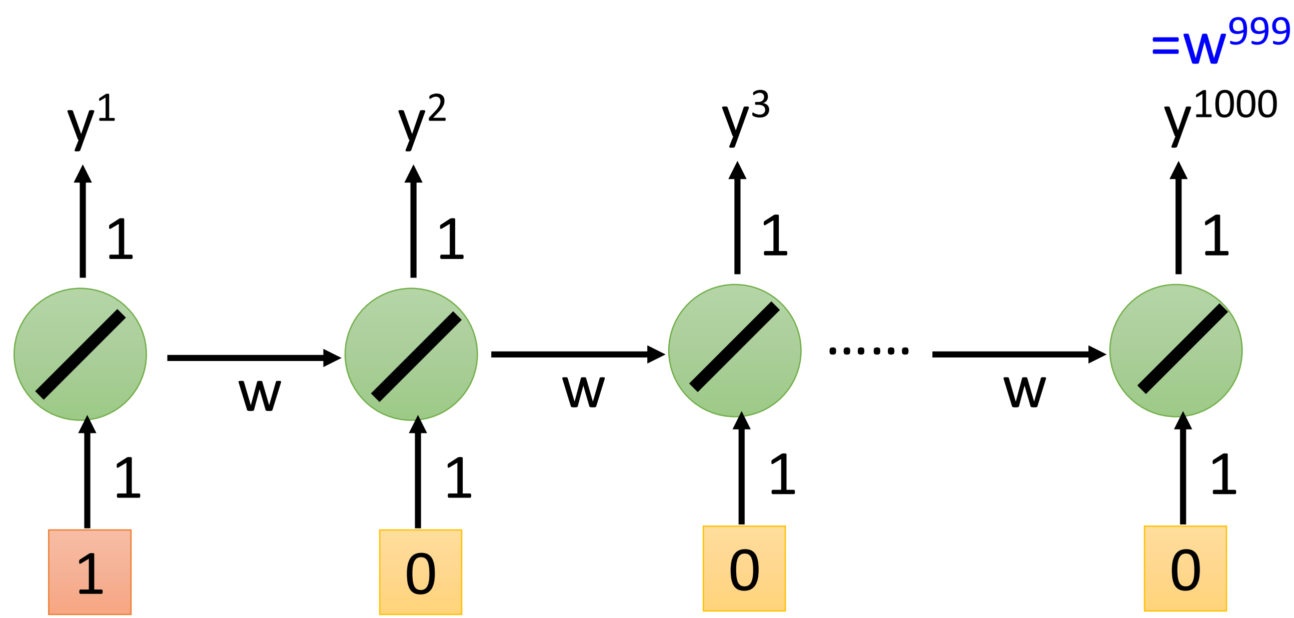
Large $\partial L / \partial w$ → Small Learning rate?

$w = 0.99 \Rightarrow y^{1000} \approx 0$

$w = 0.01 \Rightarrow y^{1000} \approx 0$

small $\partial L / \partial w$ → Large Learning rate?

***Toy Example***



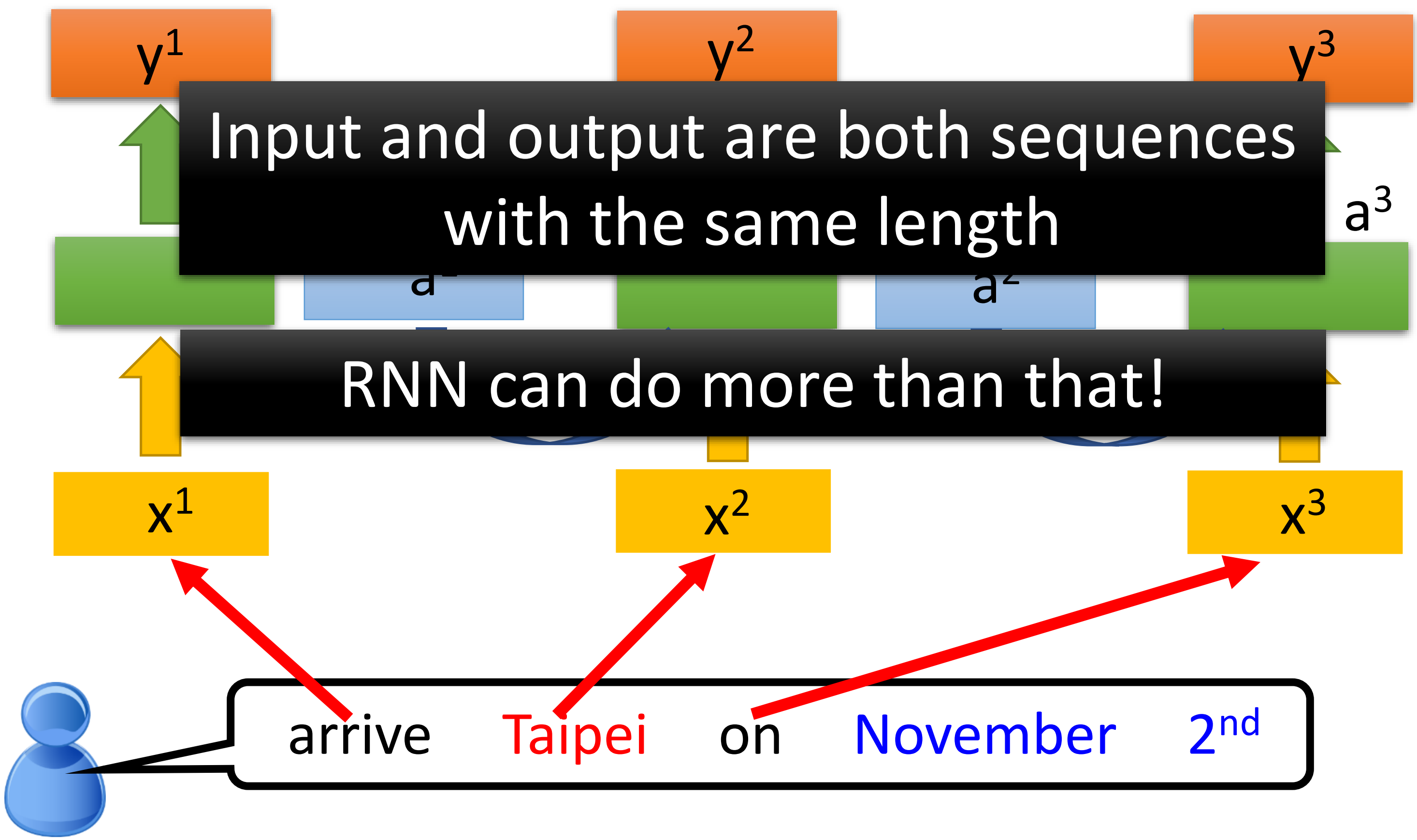
# RNN Applications



Probability of “arrive” in each slot

Probability of “Taipei” in each slot

Probability of “on” in each slot

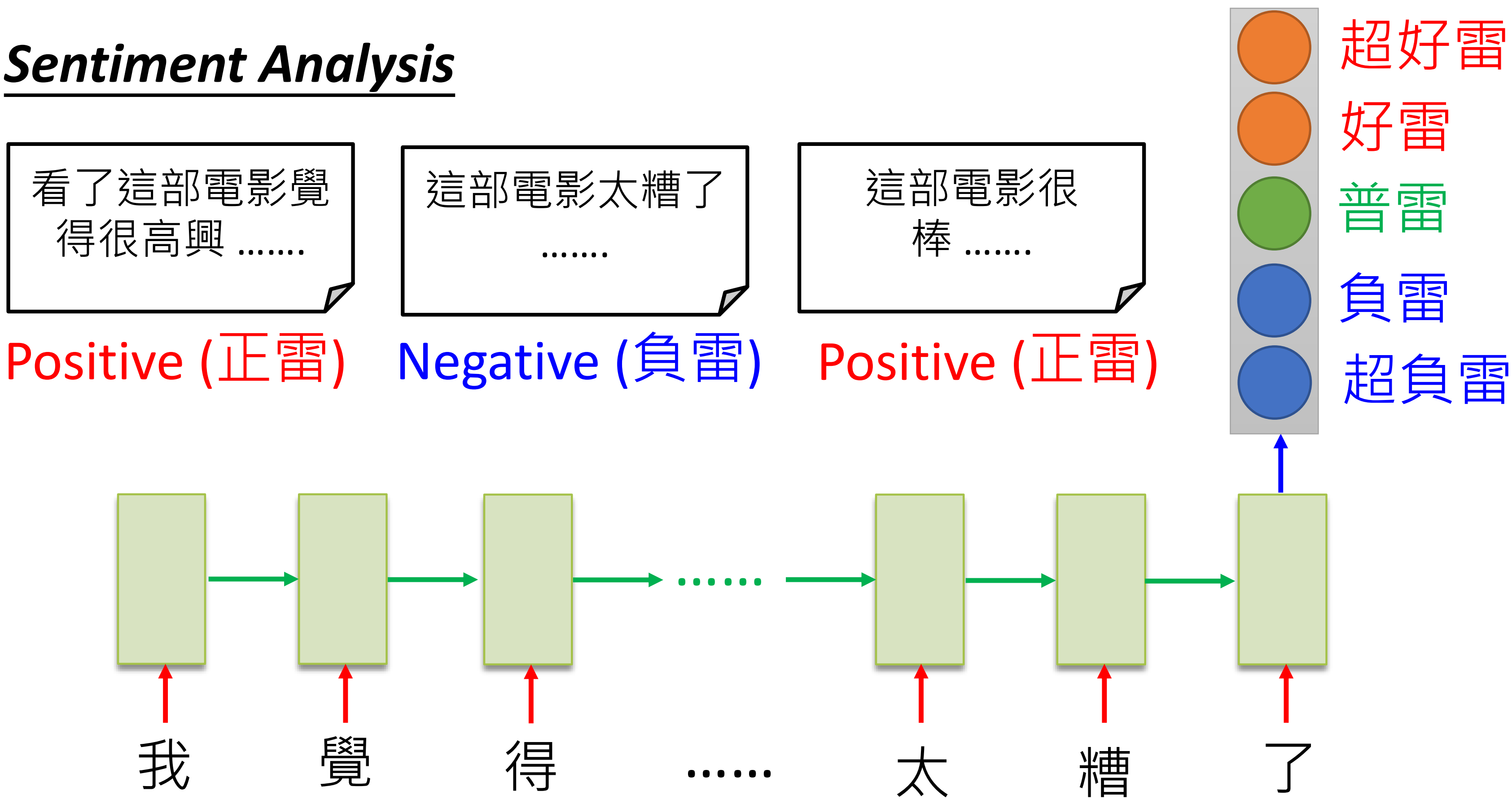


# Many-to-One



- Input is a vector sequence, but output is only one vector

***Sentiment Analysis***



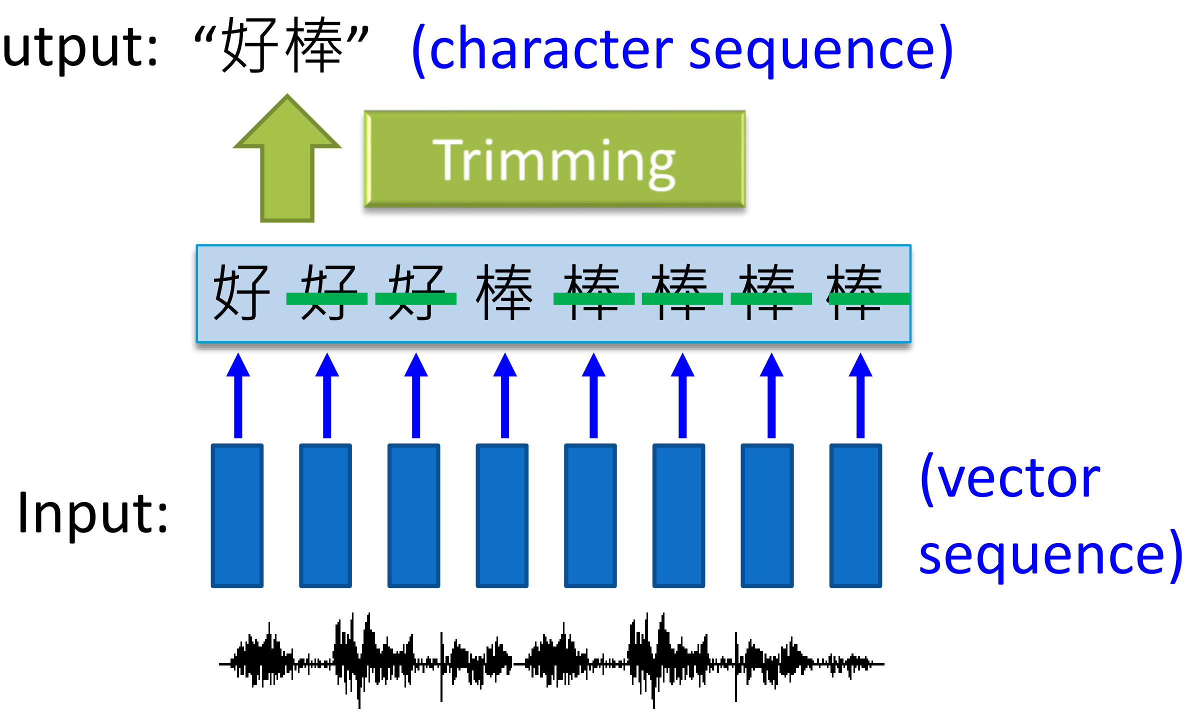
# Many-to-Many (Output is shorter)



- Both input and output are both sequences, ***but the output is shorter.***
  - E.g. ***Speech Recognition***

Problem?

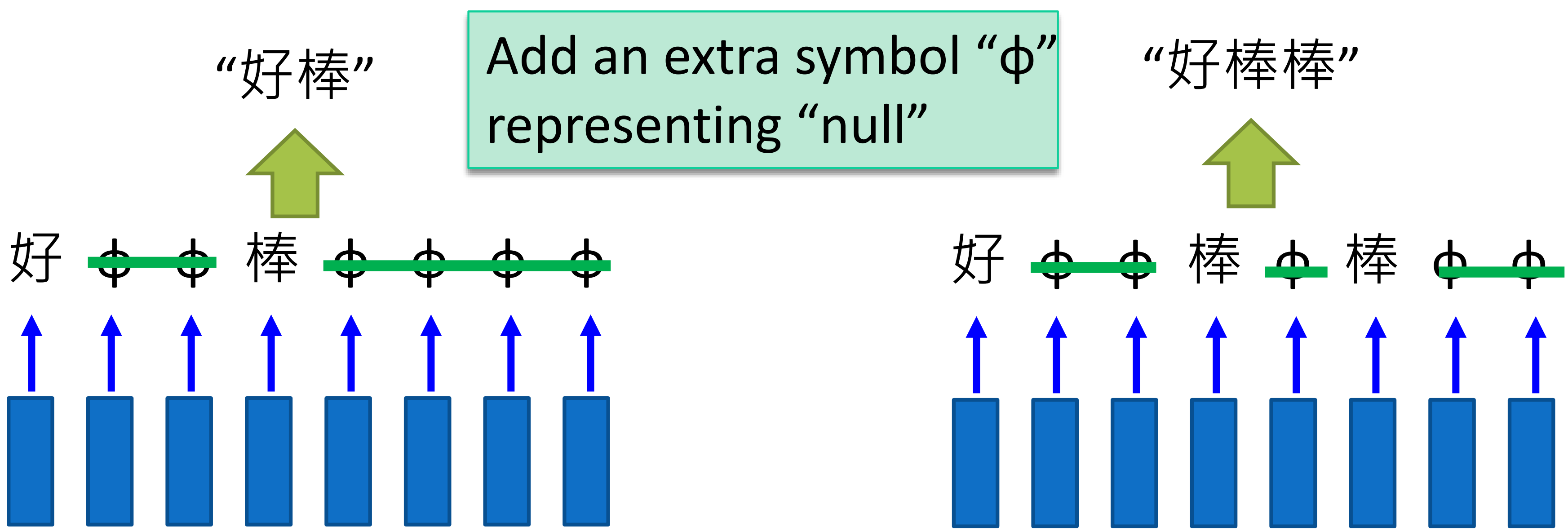
Why can't it be “好棒棒”



# Many-to-Many (Output is shorter)



- Both input and output are both sequences, ***but the output is shorter.***
- Connectionist Temporal Classification (CTC)

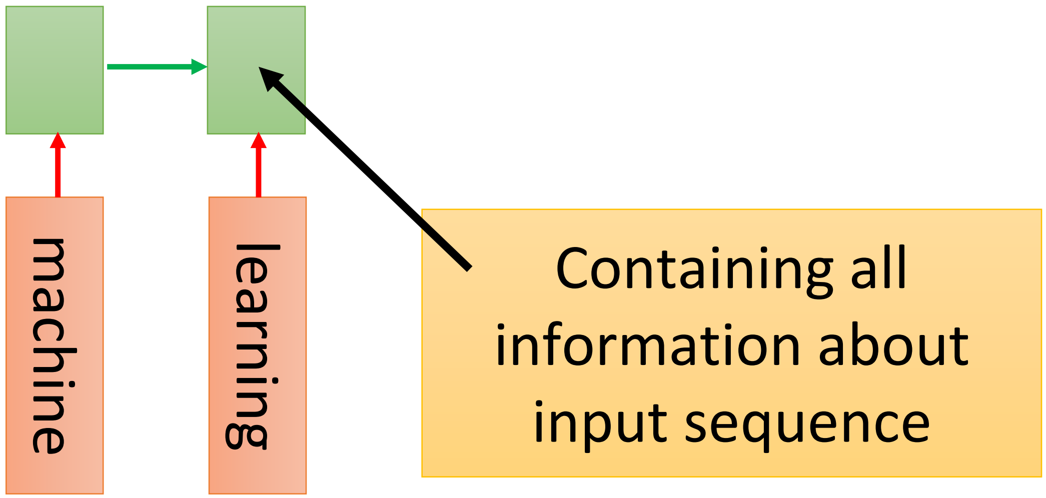



[Alex Graves, ICML'06][Alex Graves, ICML'14][Haşim Sak, Interspeech'15][Jie Li, Interspeech'15][Andrew Senior, ASRU'15]

# Many-to-Many (Output has no limitation)



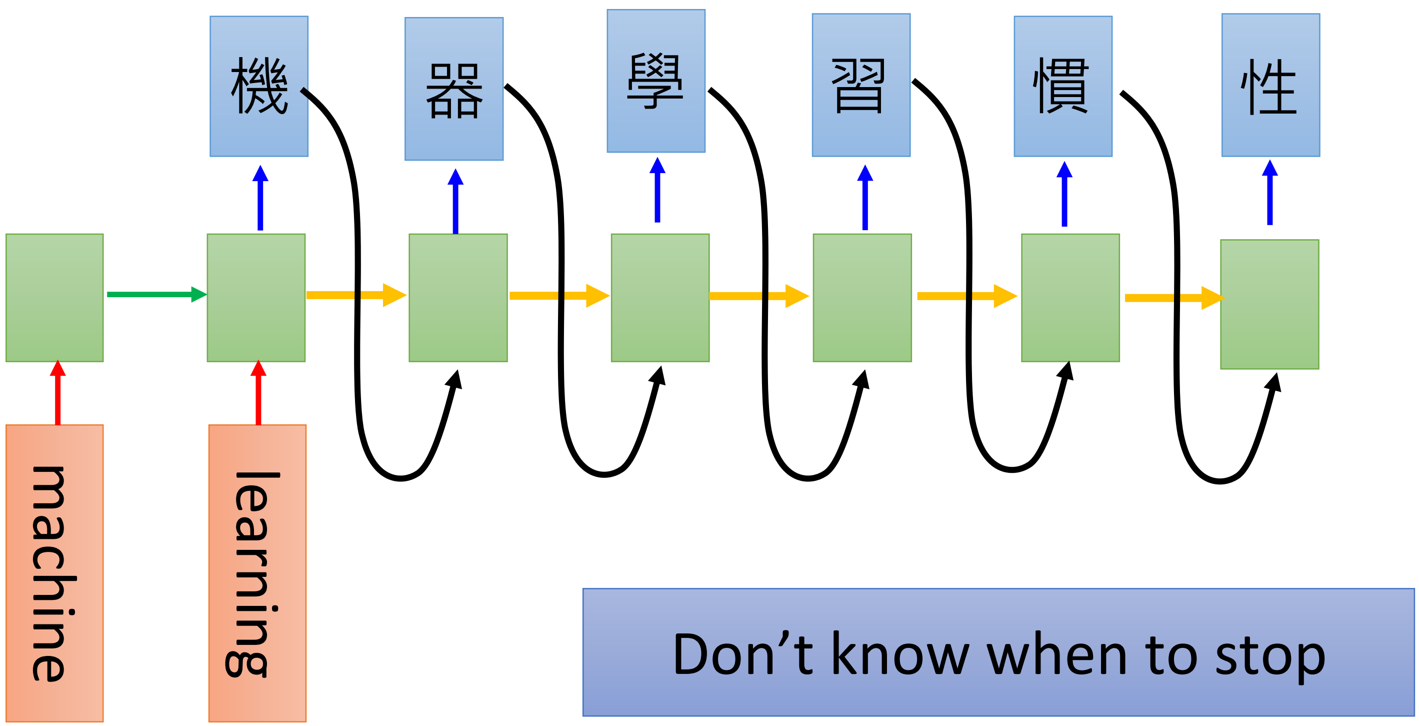
- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



# Many-to-Many (Output has no limitation)

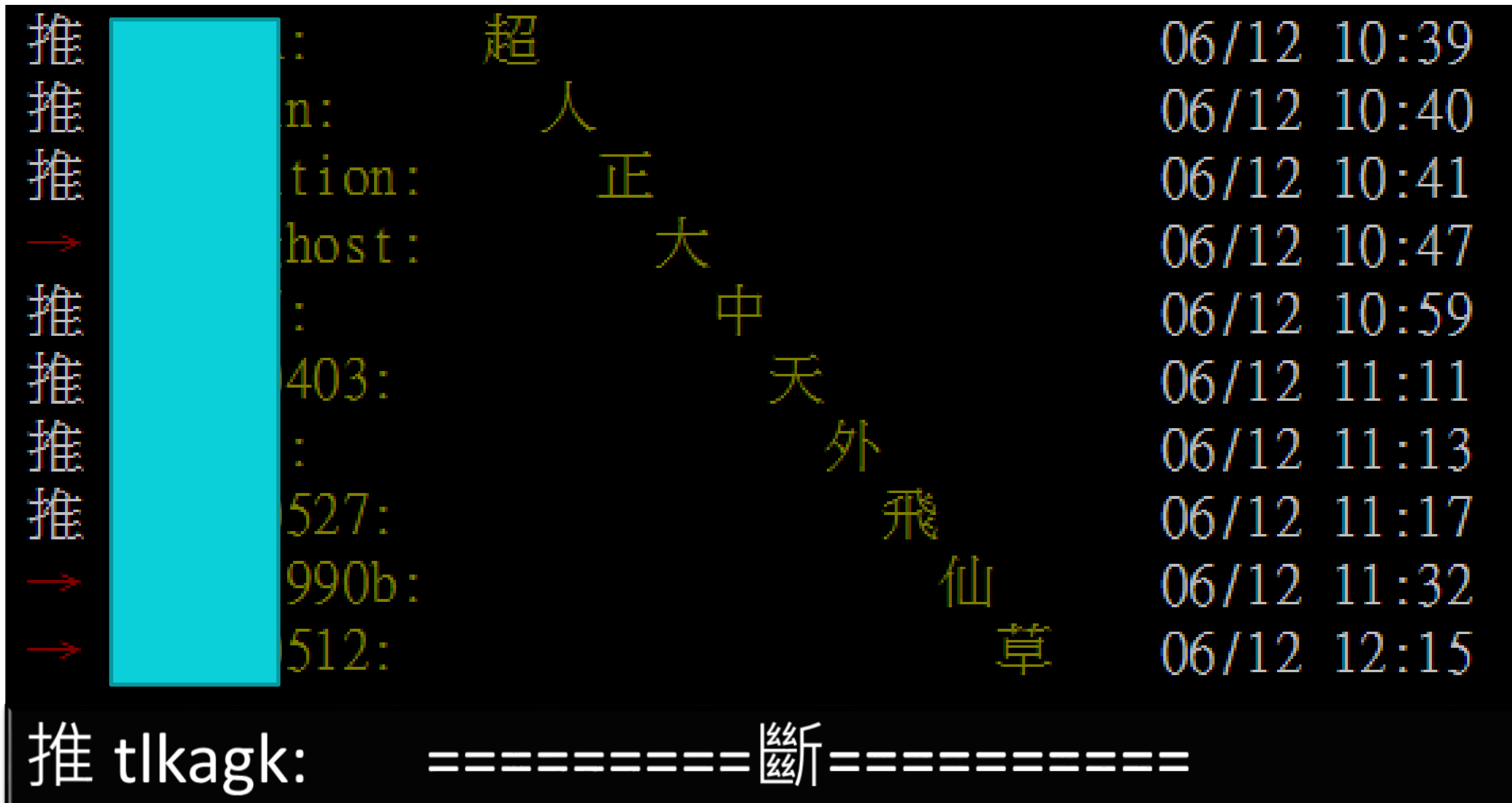


- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



# Many-to-Many (Output has no limitation)



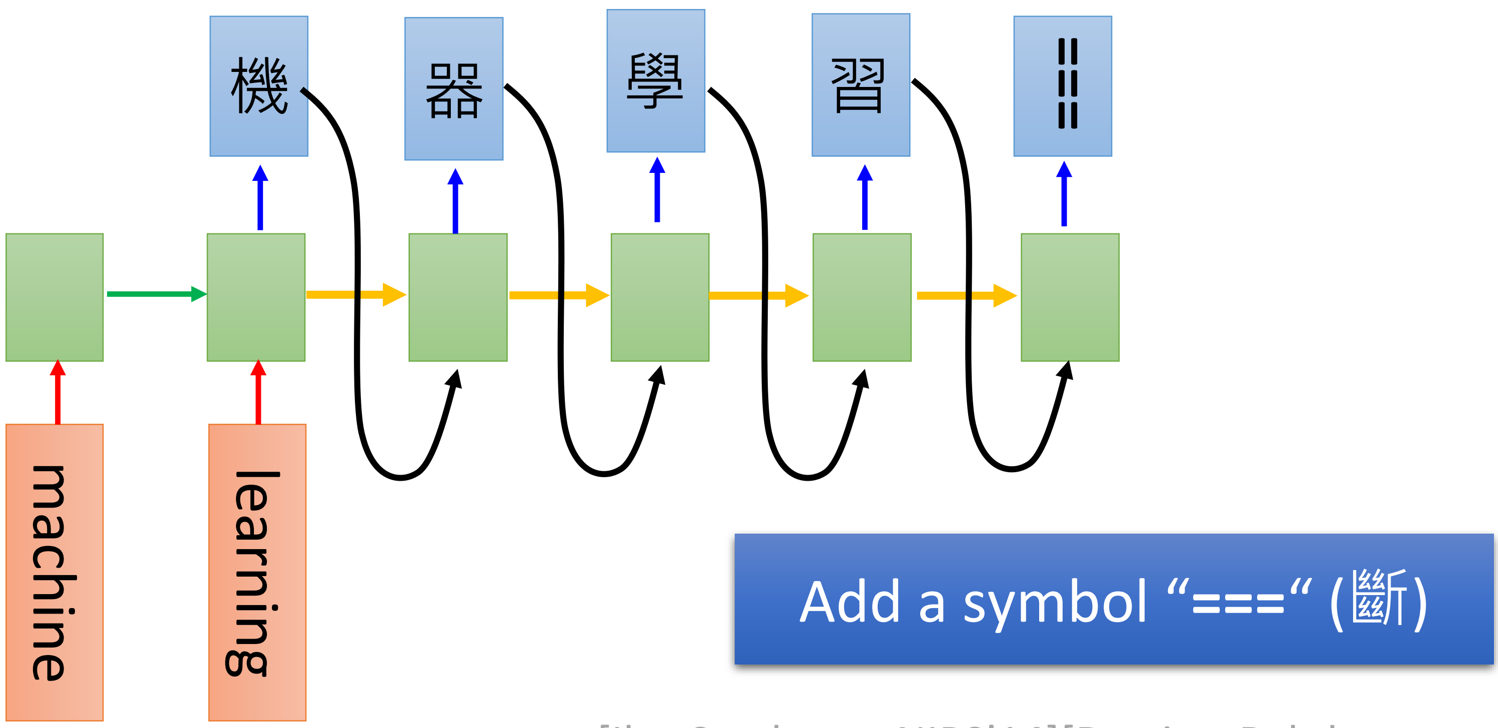



http://zh.pttpedia.wikia.com/wiki/%E6%8E%A5%E9%BE%8D%E6%8E%A8%E6%96%87 (鄉民百科)

# Many-to-Many (Output has no limitation)



- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)

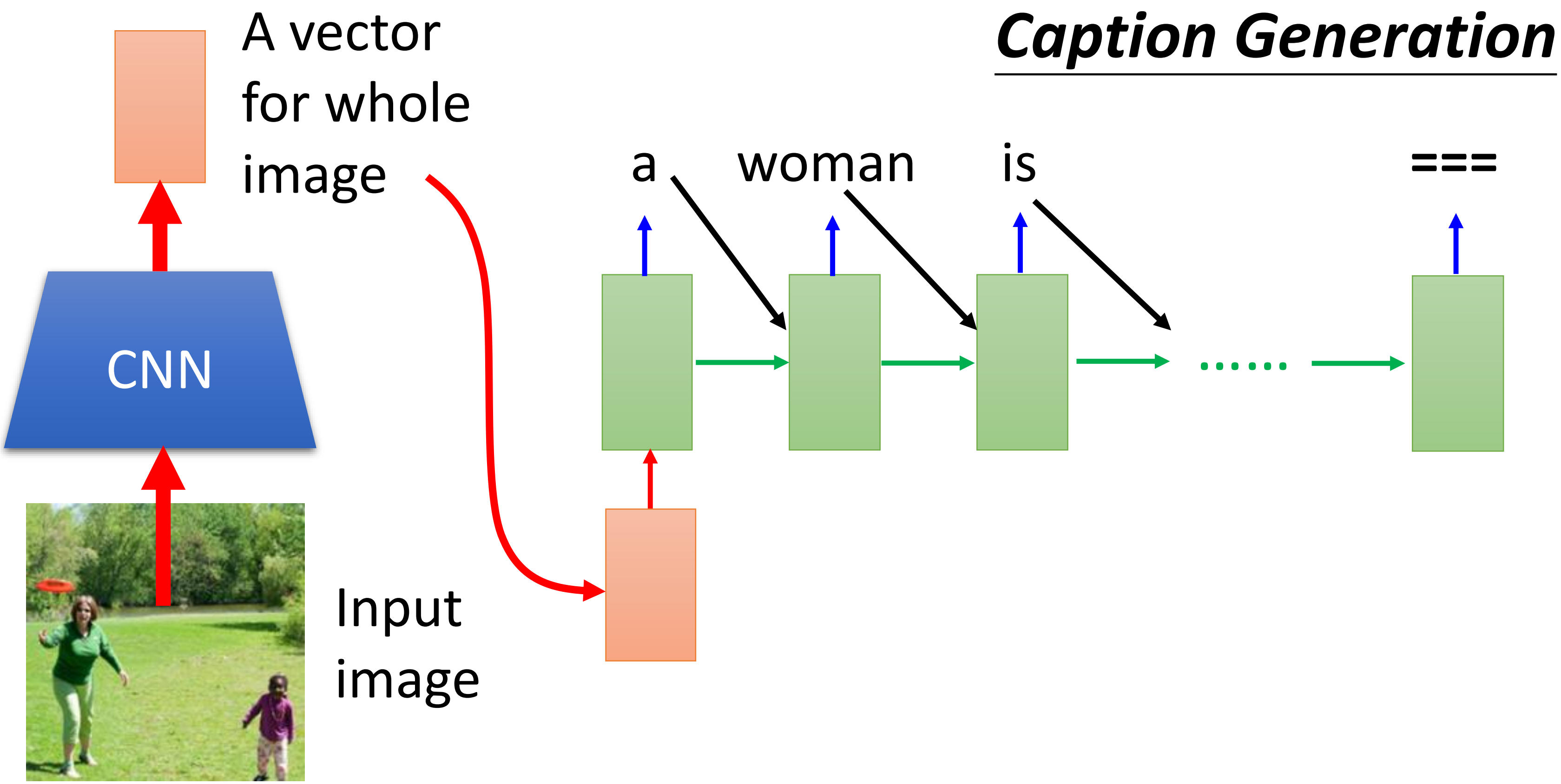



[Ilya Sutskever, NIPS'14][Dzmitry Bahdanau, arXiv'15]

# Image Caption Generation



- Input an image, but output a sequence of words




[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]

# Image Caption Generation



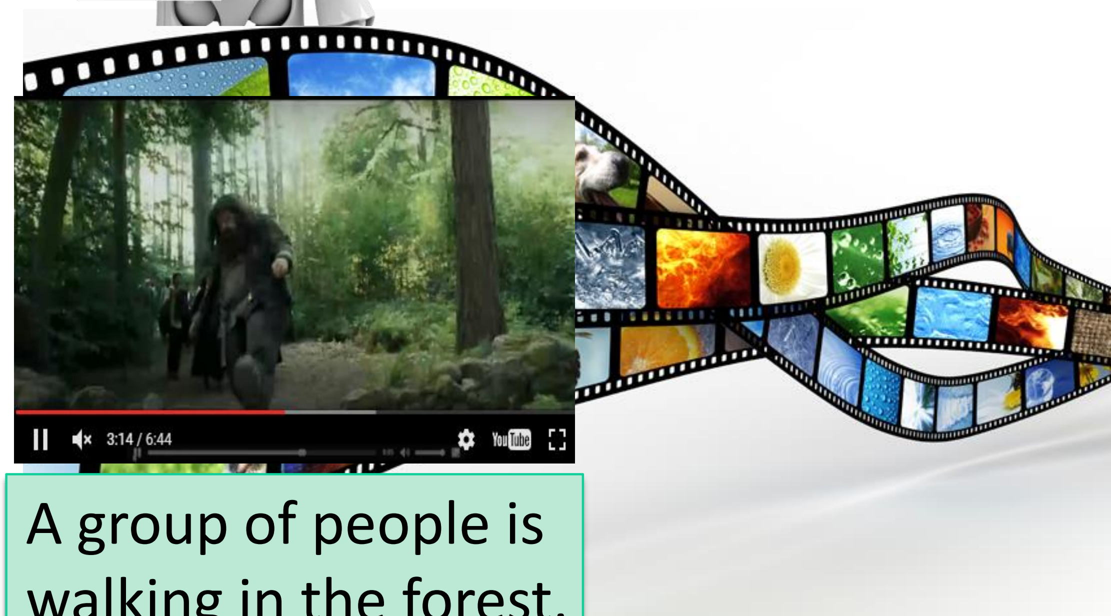
# Video Caption Generation



Video

A girl is running.

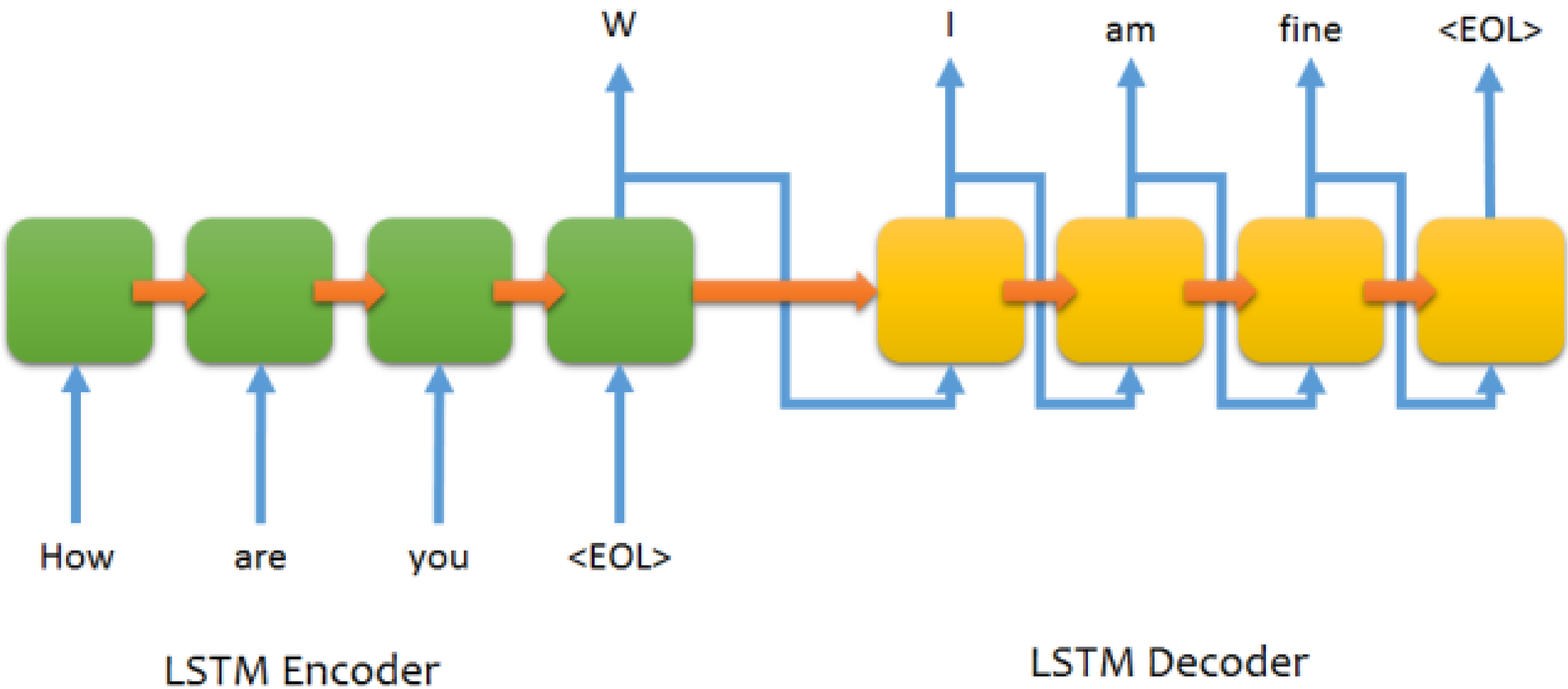

A group of people is knocked by a tree.




A group of people is walking in the forest.

# Chit-Chat Bot






電視影集 (~40,000 sentences)、美國總統大選辯論

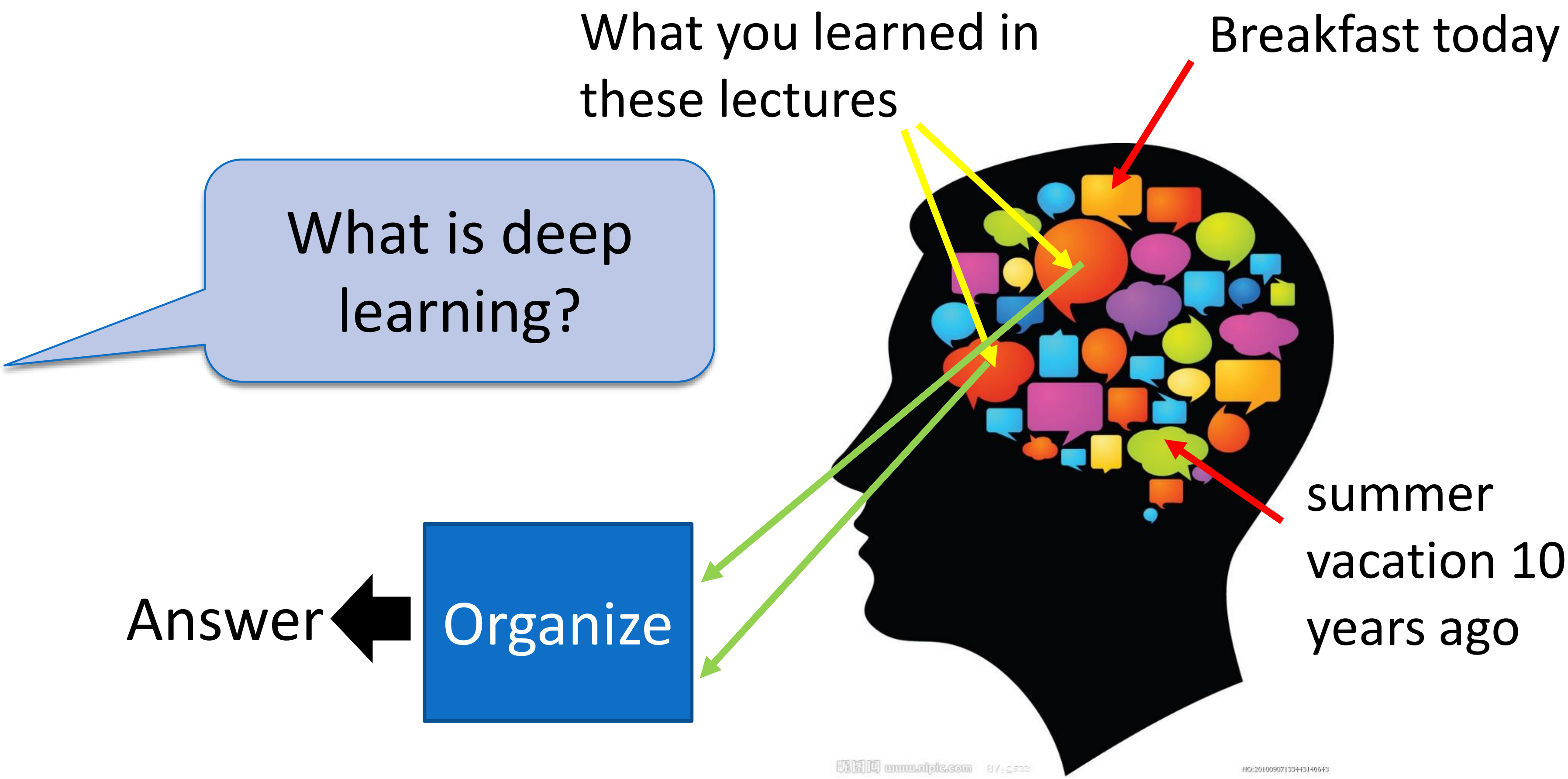
# Sci-Fi Short Film - SUNSPRING




https://www.youtube.com/watch?v=LY7x2Ihqj

# Attention and Memory

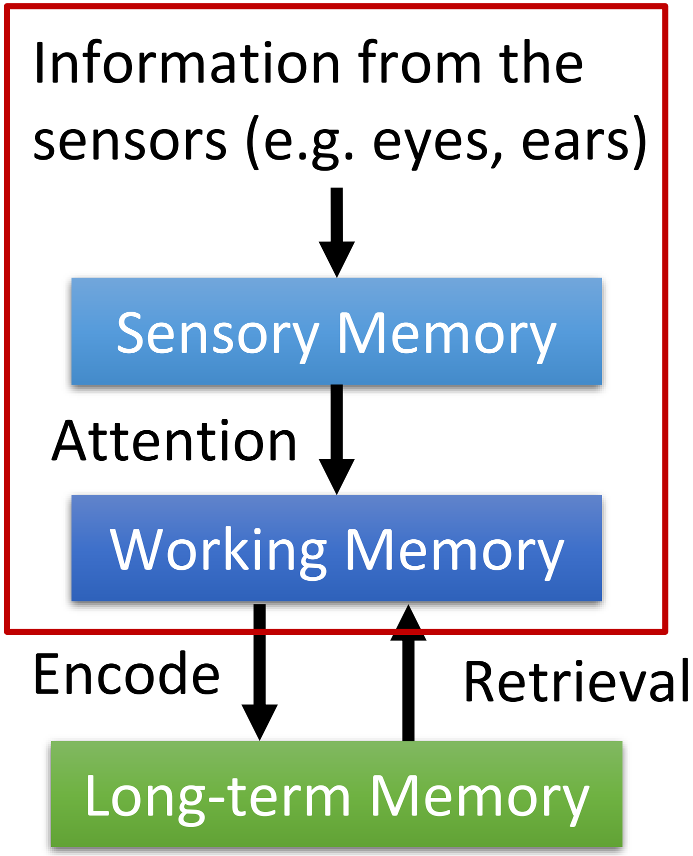





http://henrylo1605.blogspot.tw/2015/05/blog-post_56.html

# Attention on Sensory Info





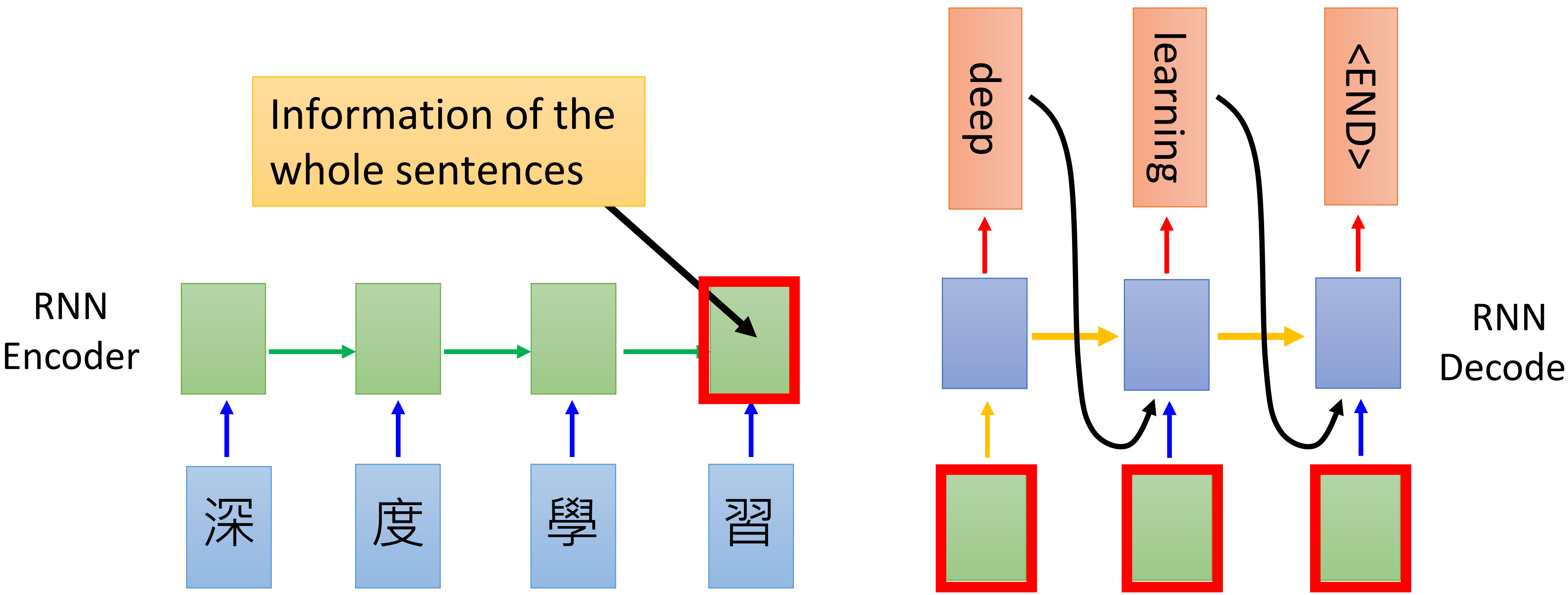

When the input is a very long sequence or an image

→ Pay attention on partial of the input object each time

# Machine Translation

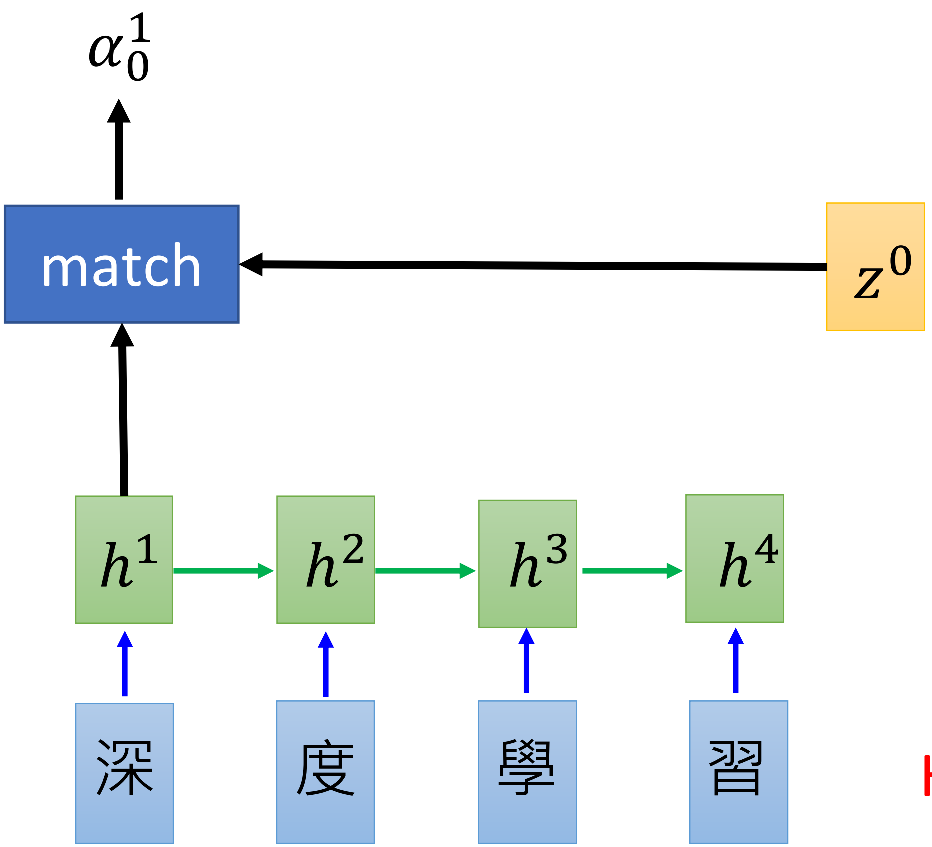


- Sequence-to-sequence learning: both input and output are both sequences ***with different lengths***.
- E.g. 深度學習 → deep learning



# Machine Translation with Attention





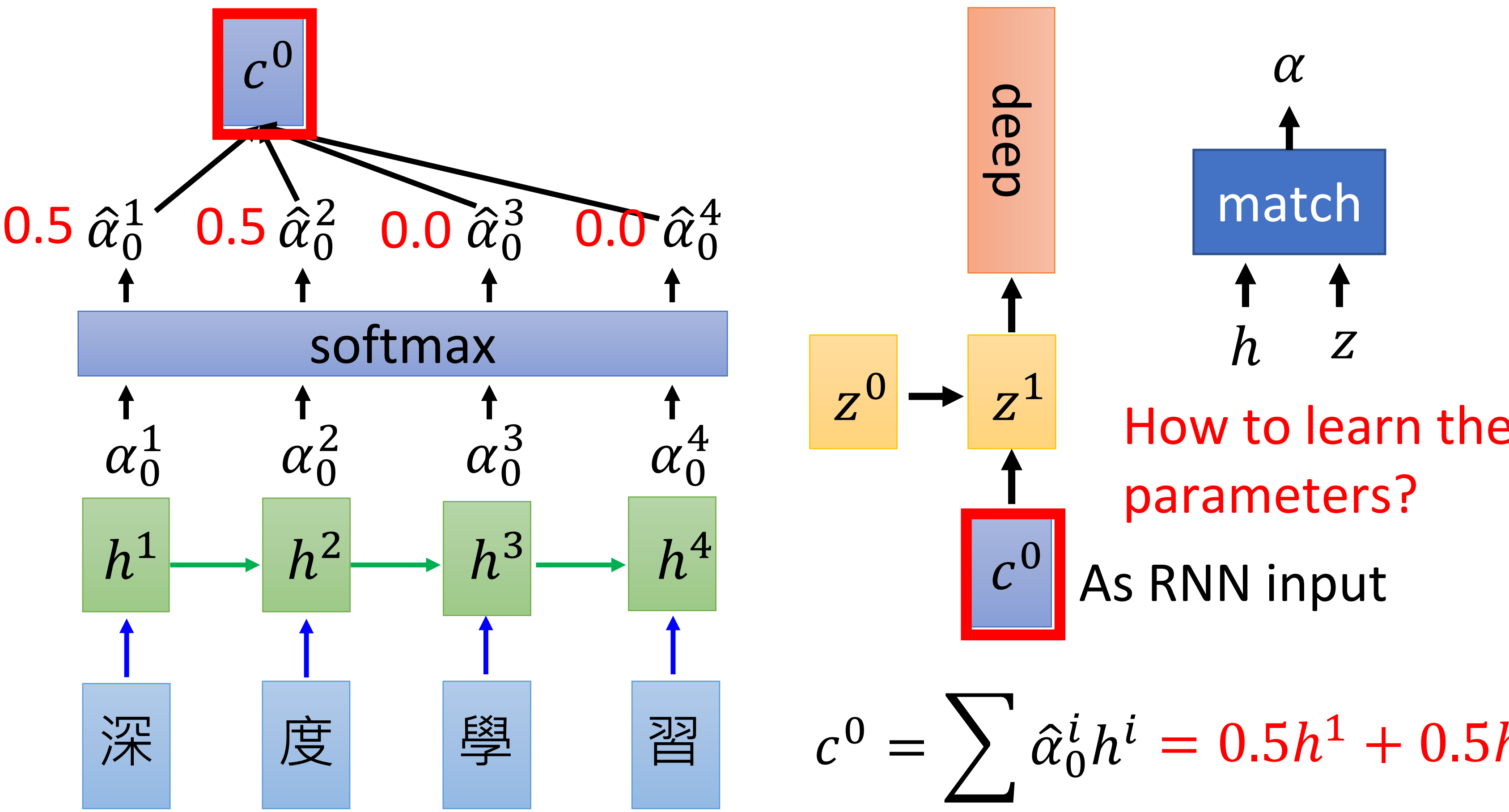

What is match ?

- Cosine similarity of $z$ and $h$
- Small NN whose input is $z$ and $h$, output a scalar
- $\alpha = h^T W z$

How to learn the parameters?

# Machine Translation with Attention

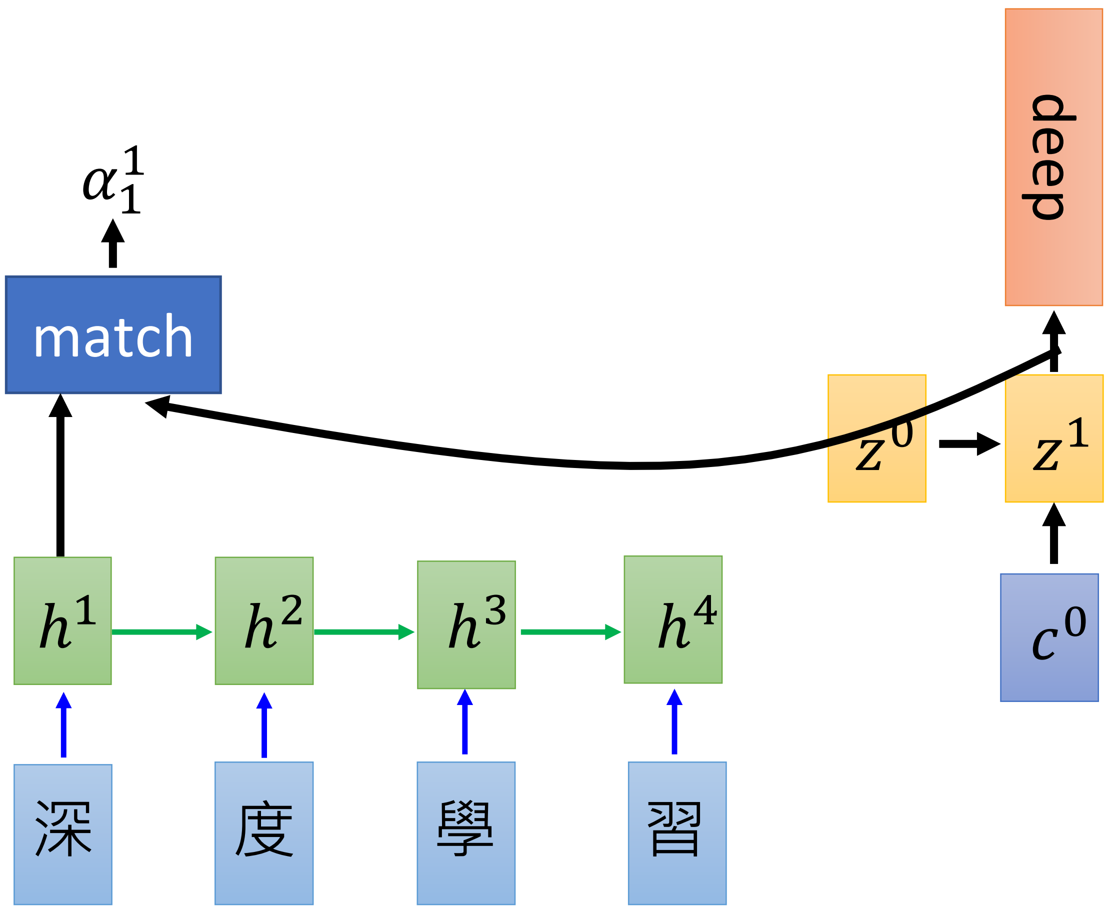





$$c^0 = \sum \hat{\alpha}_0^i h^i = 0.5h^1 + 0.5h^2$$

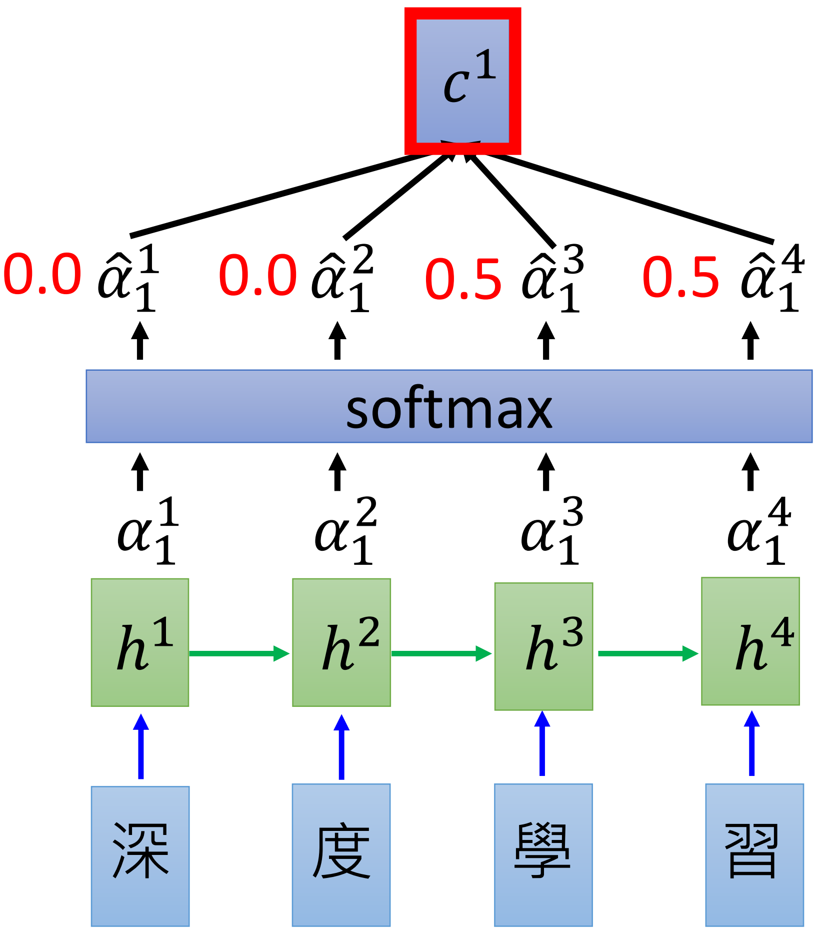
# Machine Translation with Attention

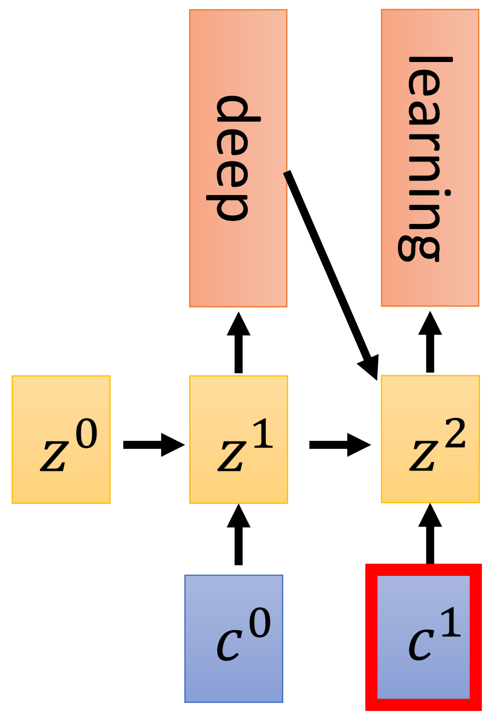




# Machine Translation with Attention



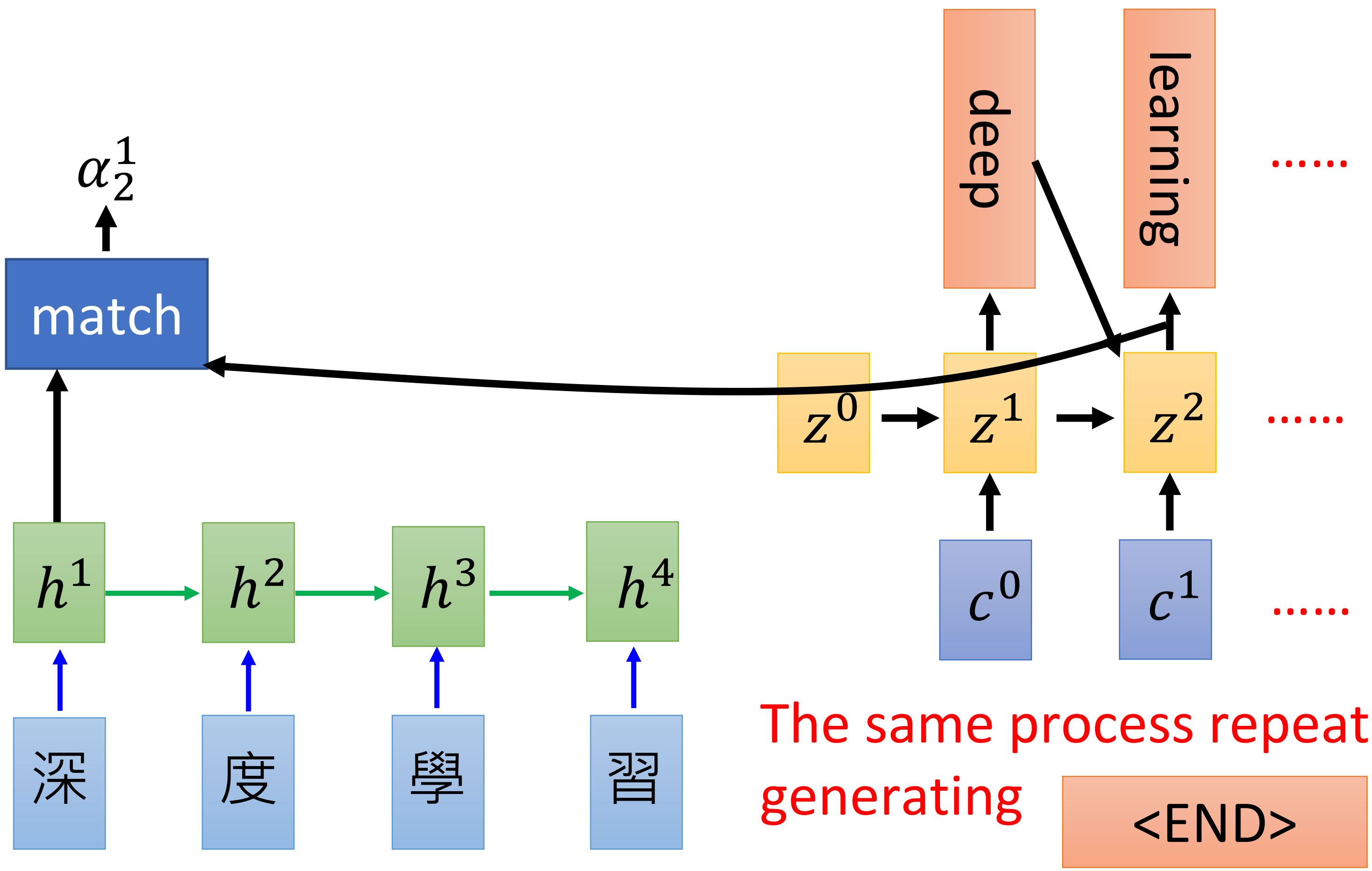



$$c^1 = \sum \hat{\alpha}_1^i h^i = 0.5h^3 + 0.5h^4$$

# Machine Translation with Attention

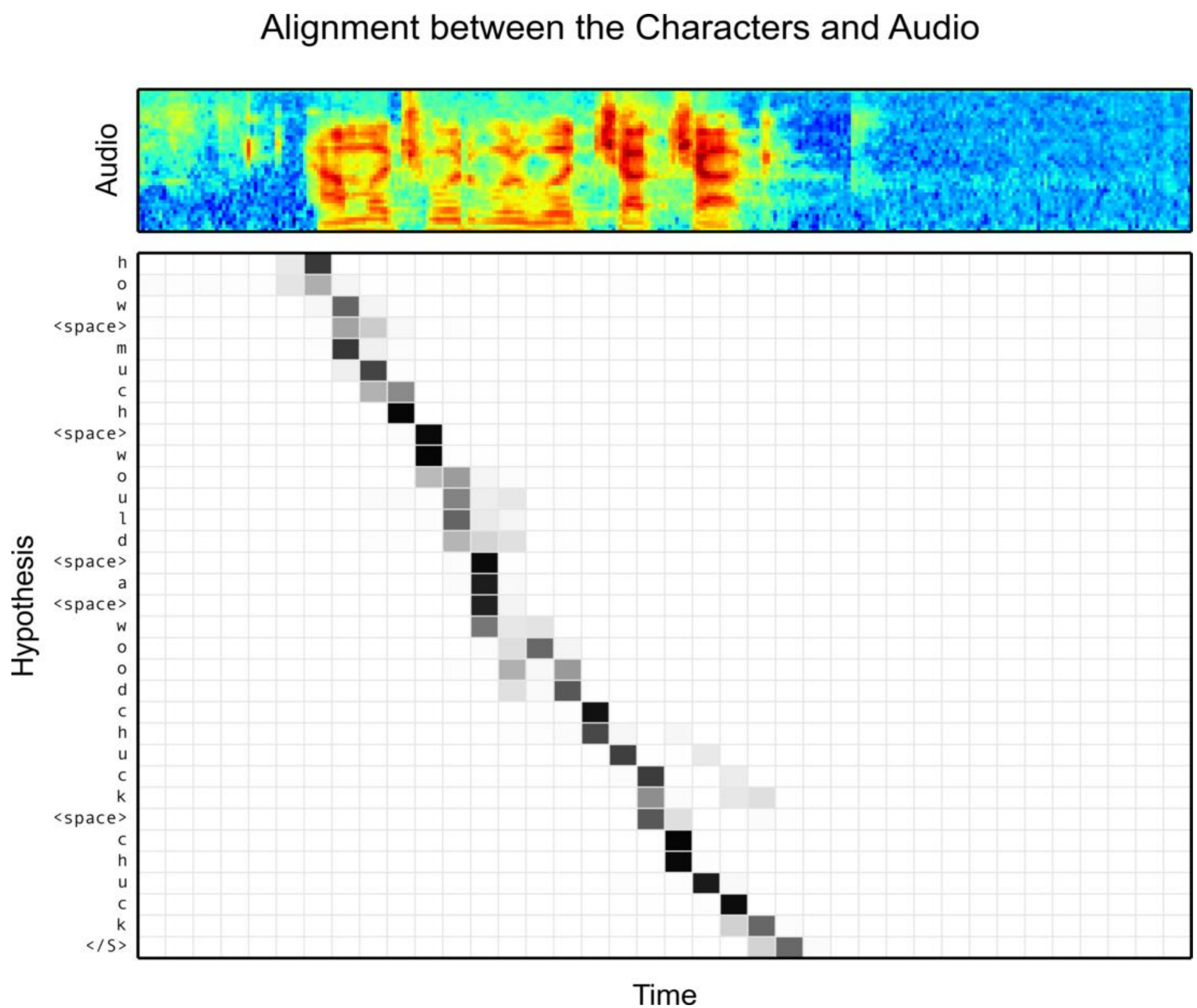





The same process repeat until generating

# Speech Recognition with Attention



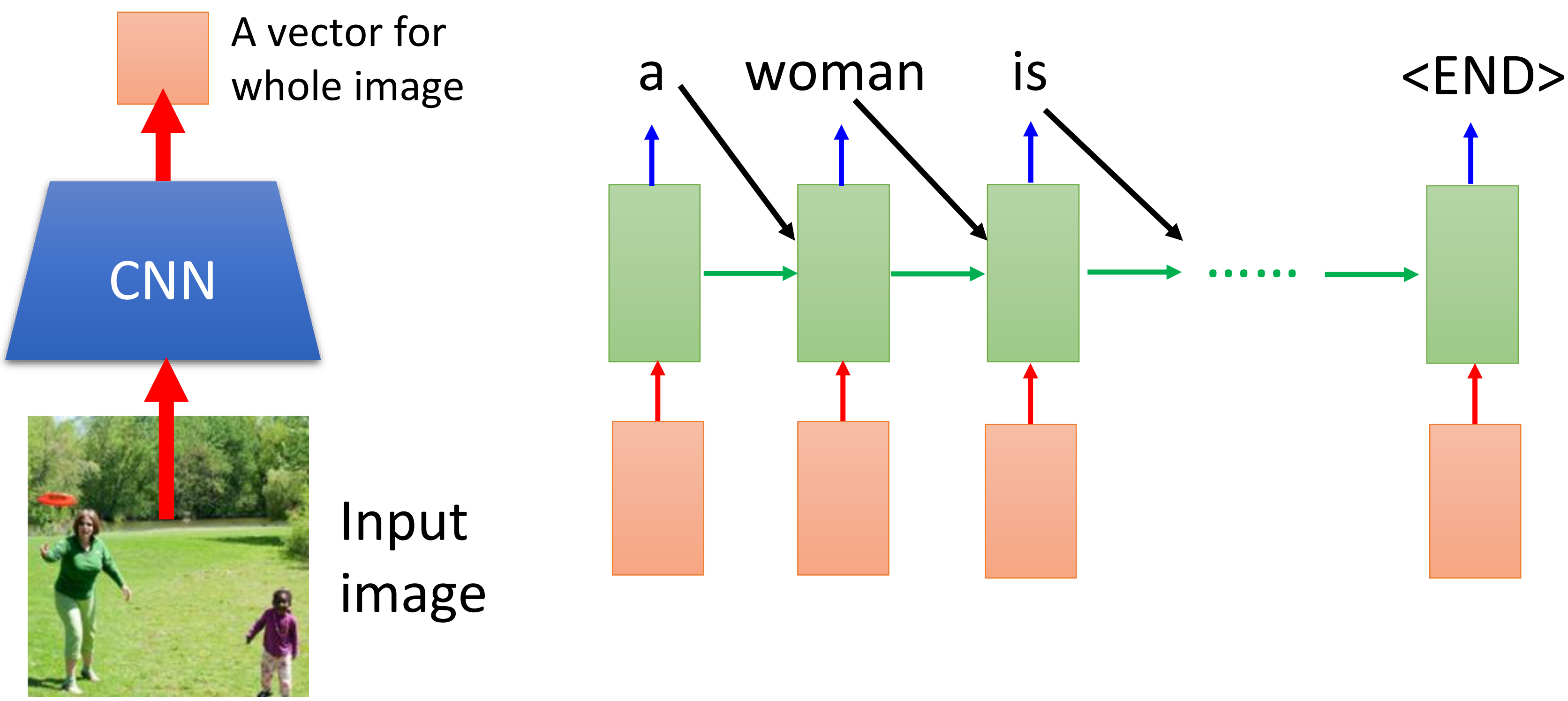



Chan et al., "Listen, Attend and Spell", arXiv, 2015 .

# Image Captioning

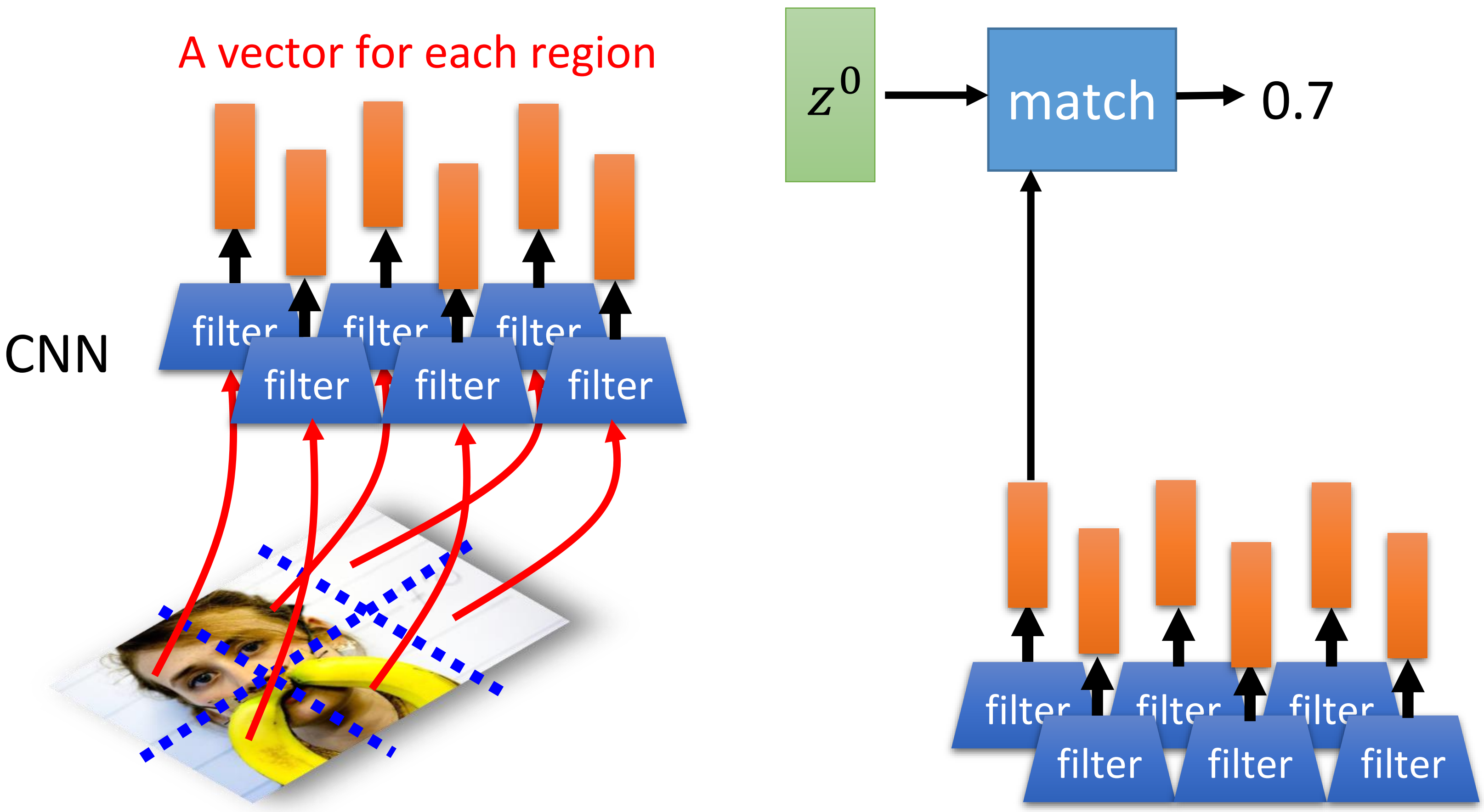


- Input: image
- Output: word sequence



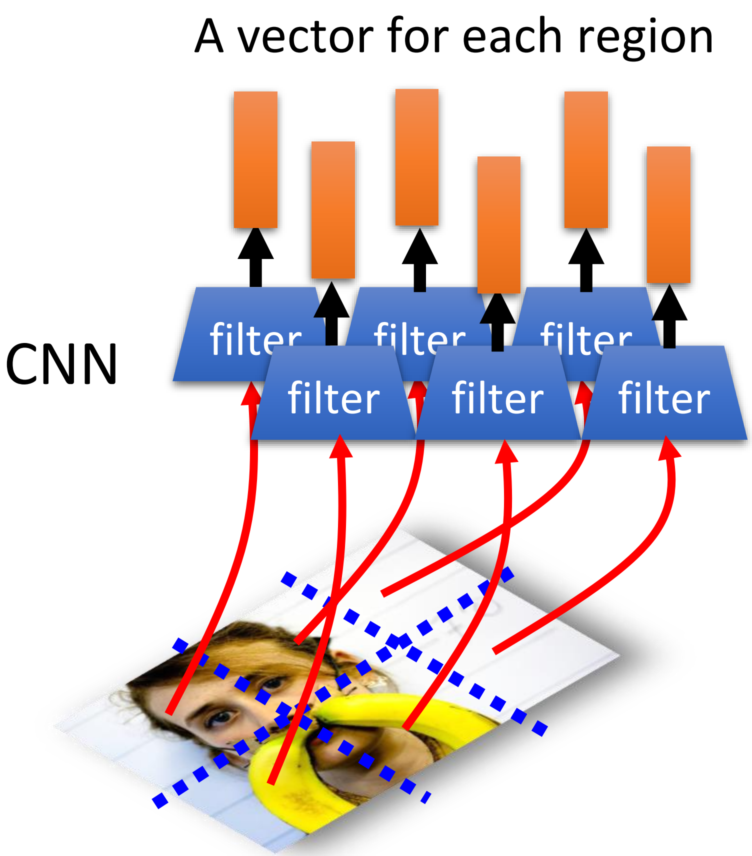
# Image Captioning with Attention

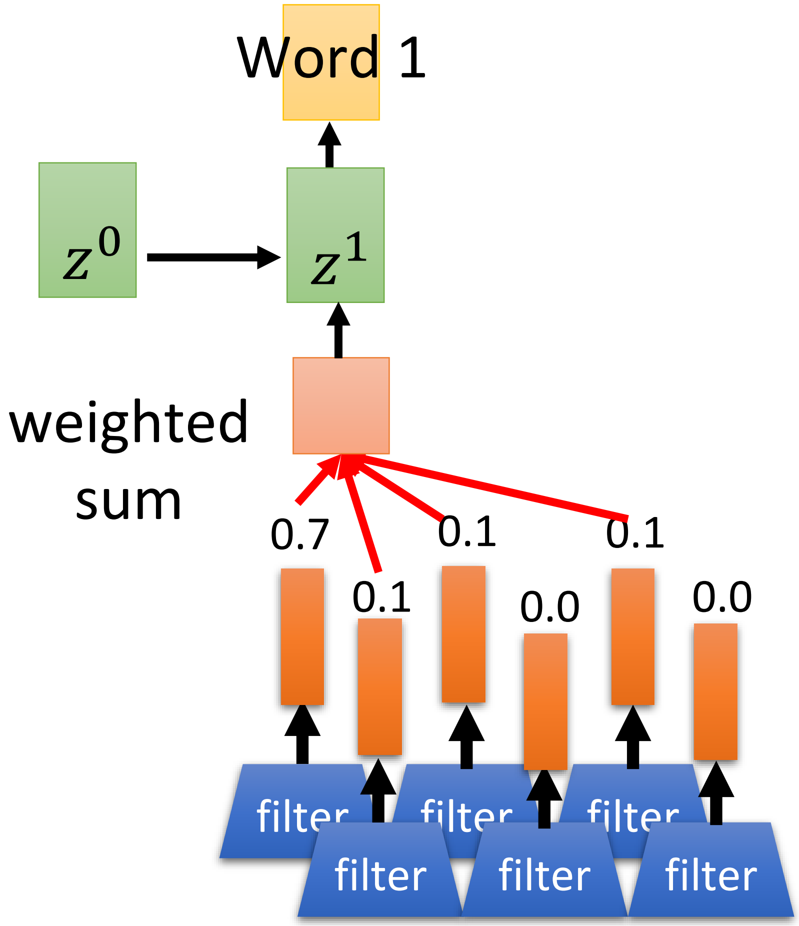




# Image Captioning with Attention

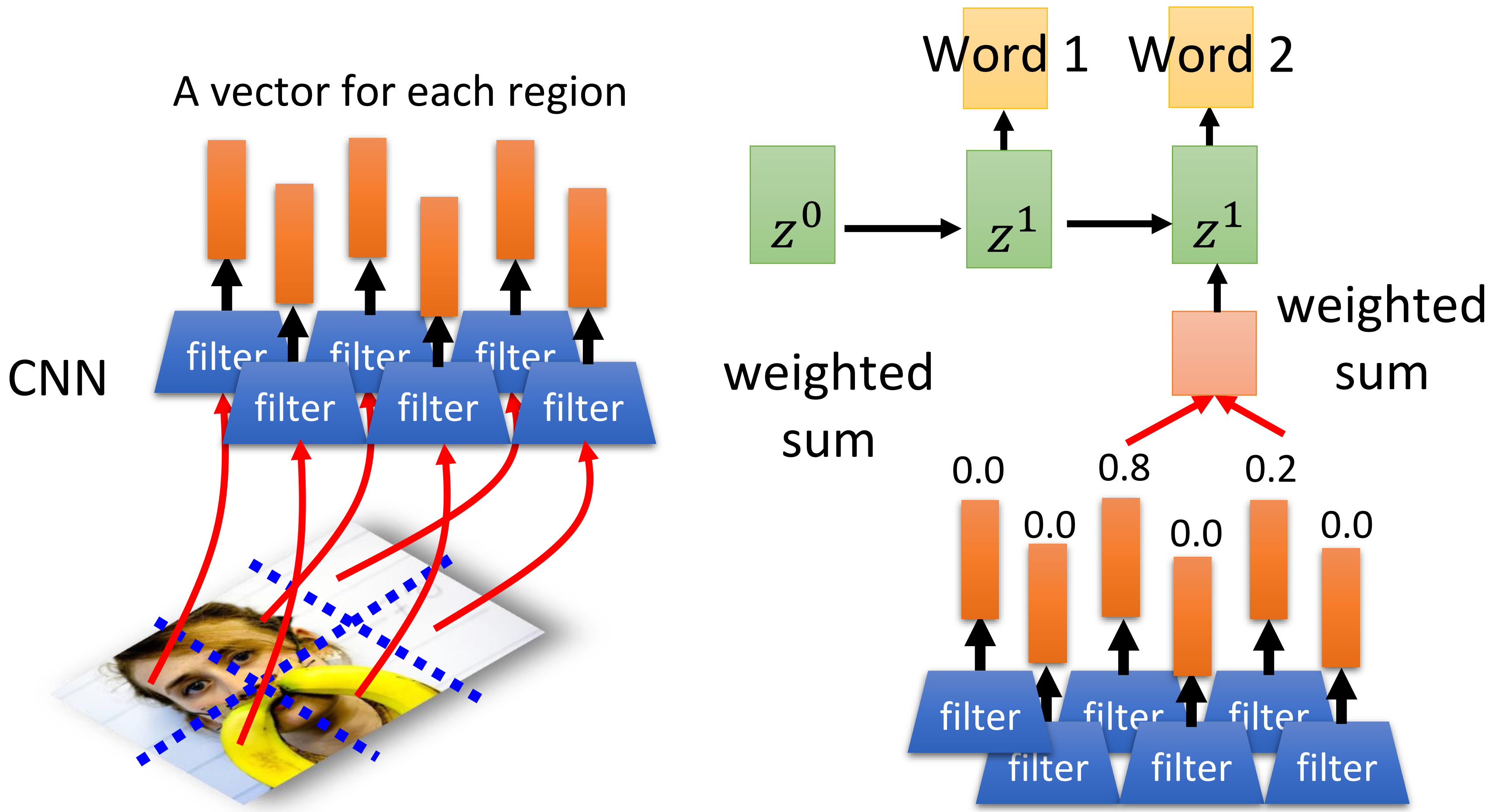




# Image Captioning with Attention





# Image Captioning

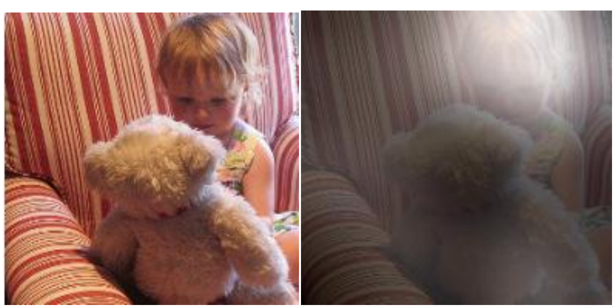


- Good examples

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.


A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

# Image Captioning



- Bad examples

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

# Video Captioning



***Ref:*** A man and a woman ride a motorcycle

A **man** and a **woman** are **talking** on the **road**

# Video Captioning



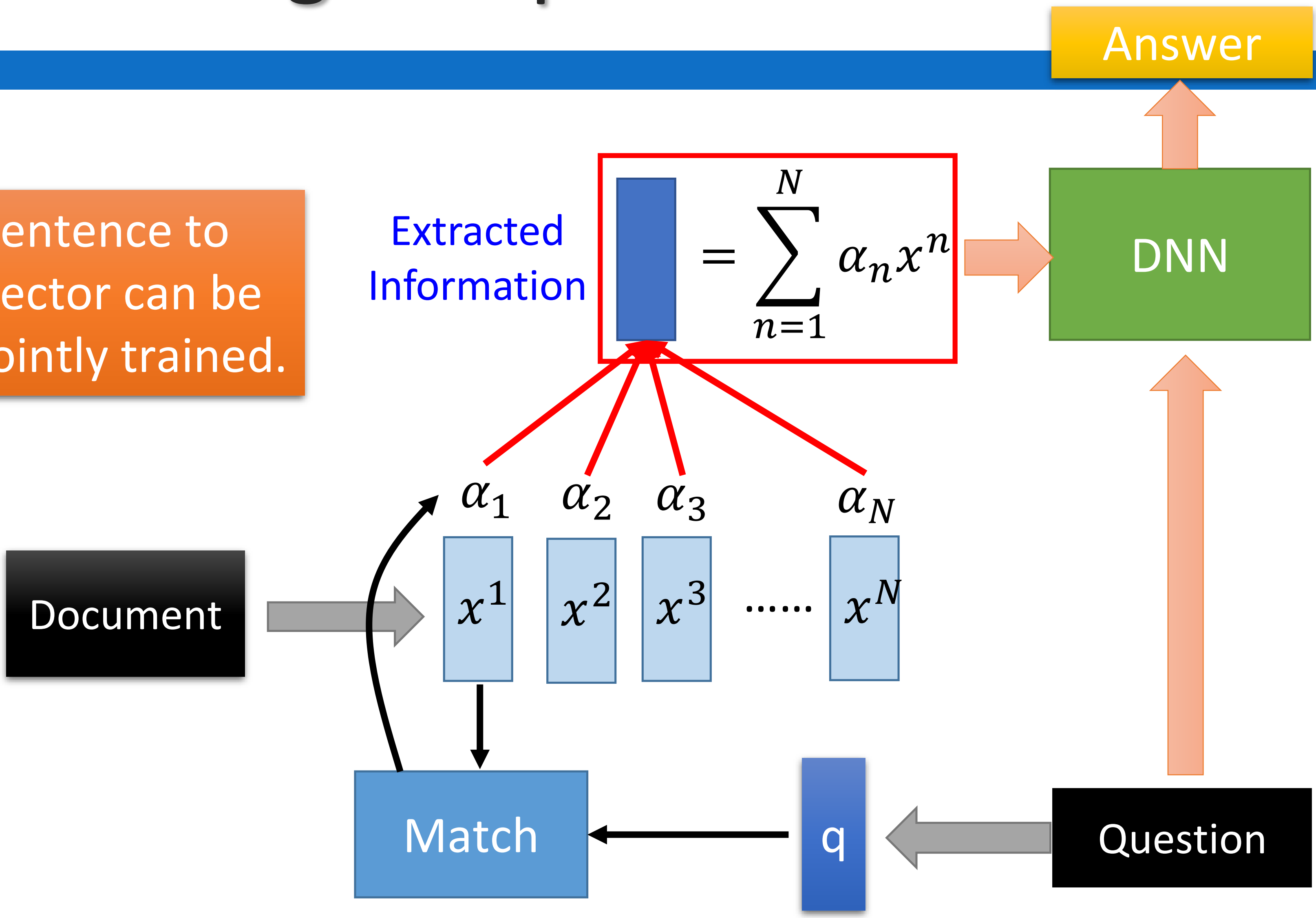
***Ref:*** A woman is frying food

**Someone** is **frying** a **fish** in a **pot**

# Reading Comprehension

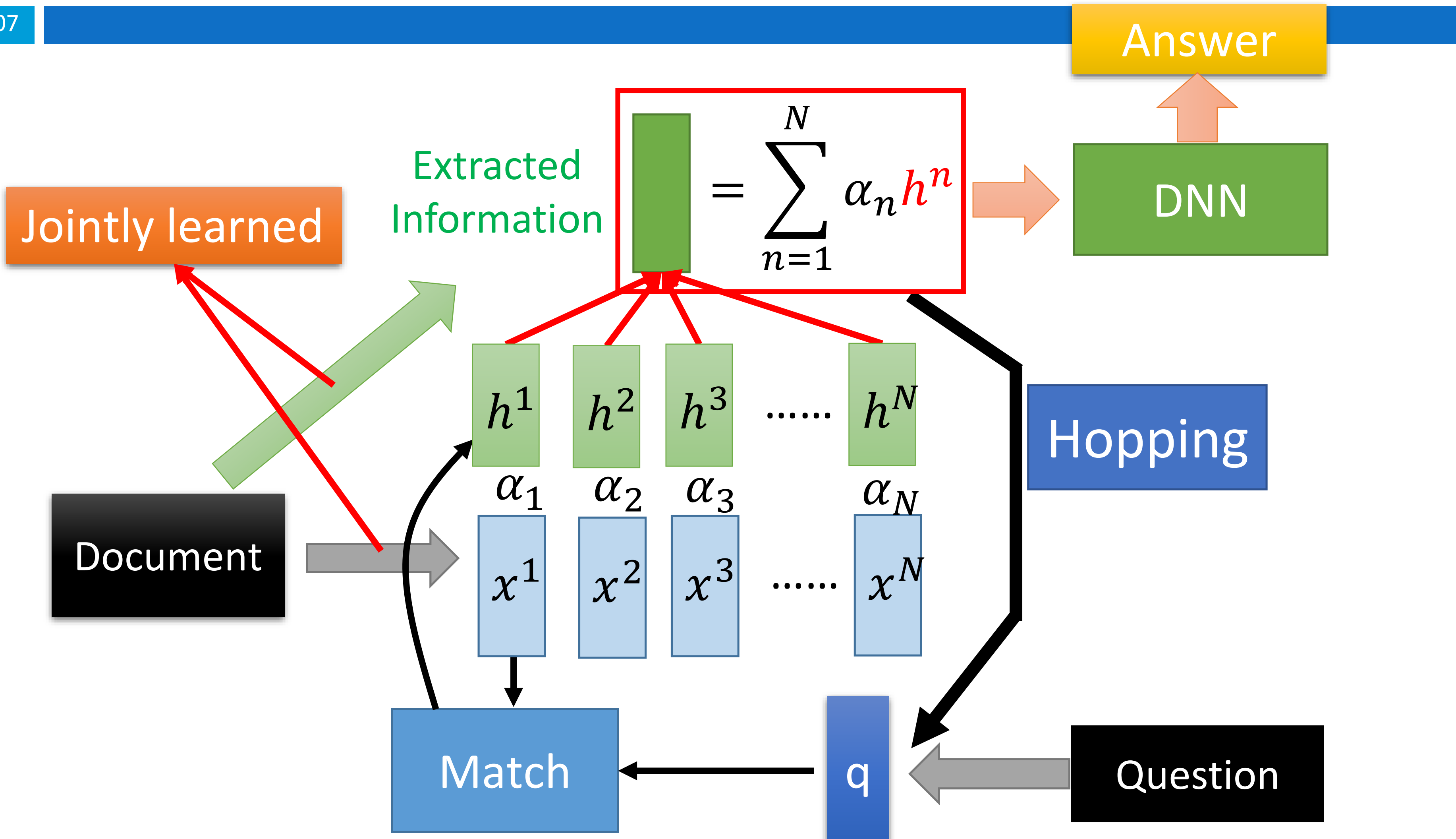


Sentence to vector can be jointly trained.



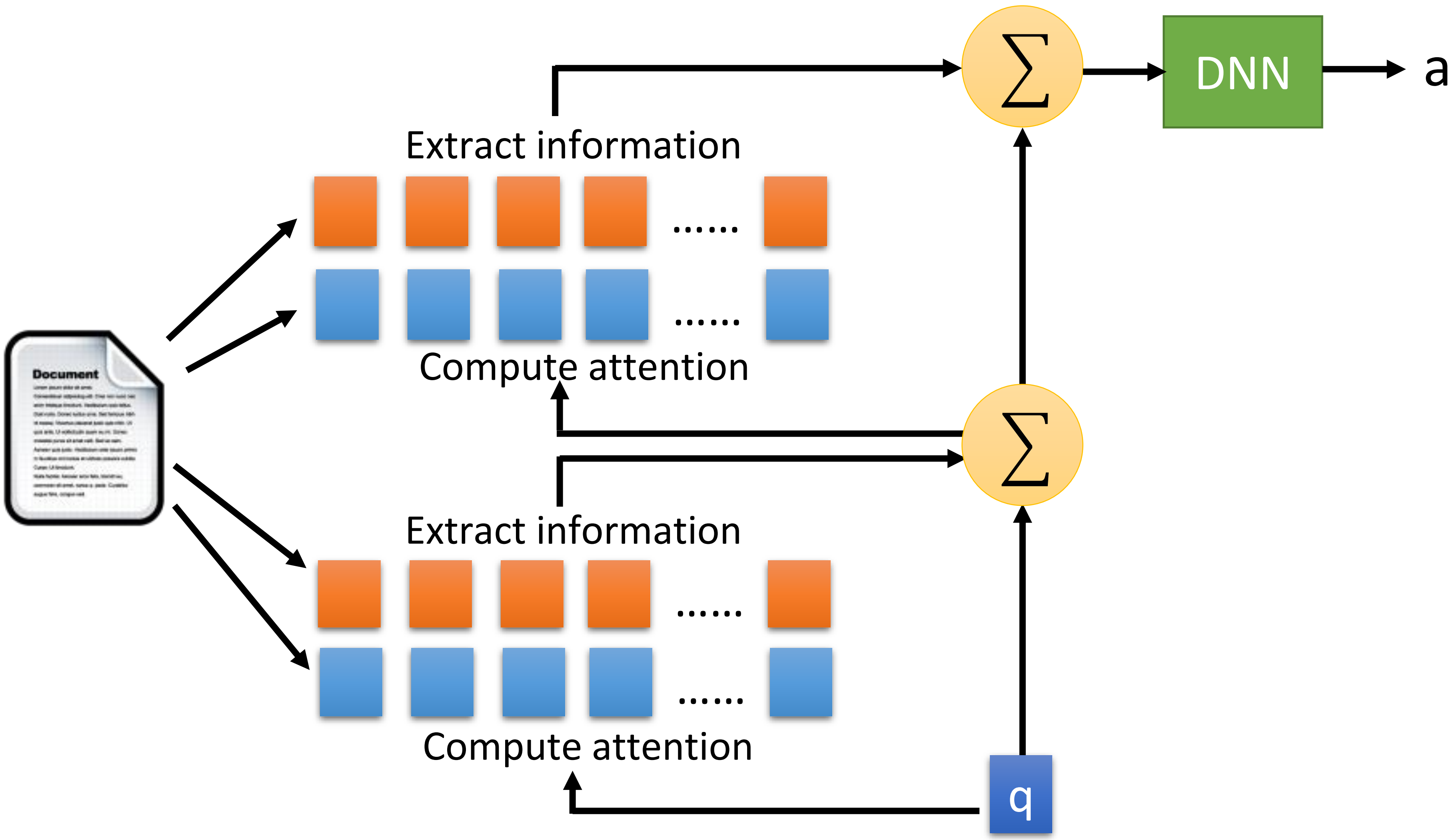
# Reading Comprehension





# Memory Network





# Memory Network

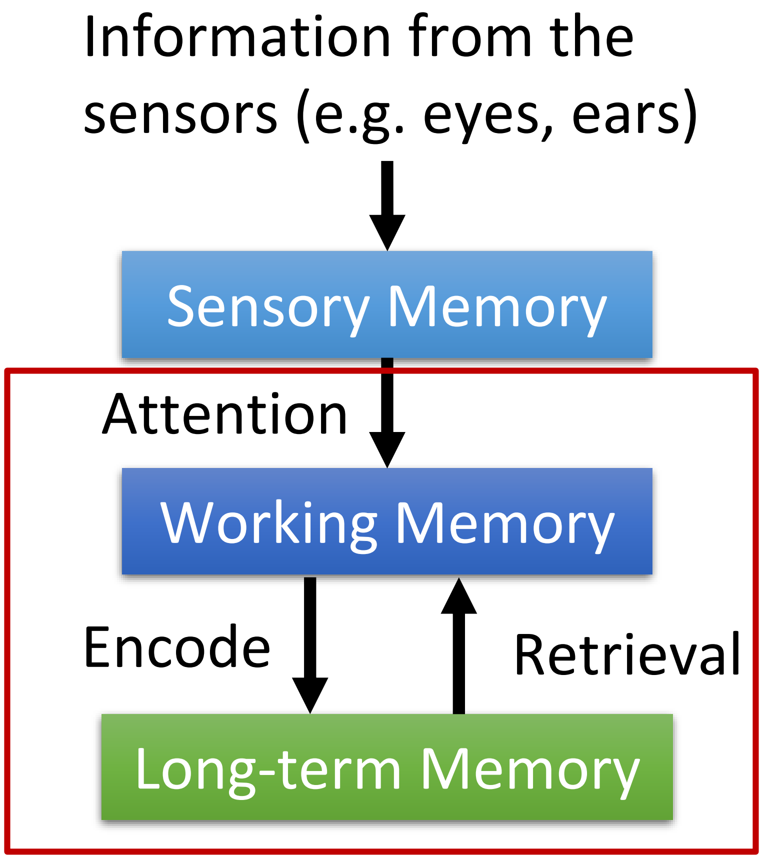


- Muti-hop performance analysis

| Story (16: basic induction) | Support | Hop 1 | Hop 2 | Hop 3 |
|---|---|---|---|---|
| Brian is a frog. | yes | 0.00 | 0.98 | 0.00 |
| Lily is gray. | | 0.07 | 0.00 | 0.00 |
| Brian is yellow. | yes | 0.07 | 0.00 | 1.00 |
| Julius is green. | | 0.06 | 0.00 | 0.00 |
| Greg is a frog. | yes | 0.76 | 0.02 | 0.00 |
| **What color is Greg? Answer: yellow Prediction: yellow** | | | | |

# Attention on Memory






When the input is a very long sequence or an image

→ Pay attention on partial of the input object each time

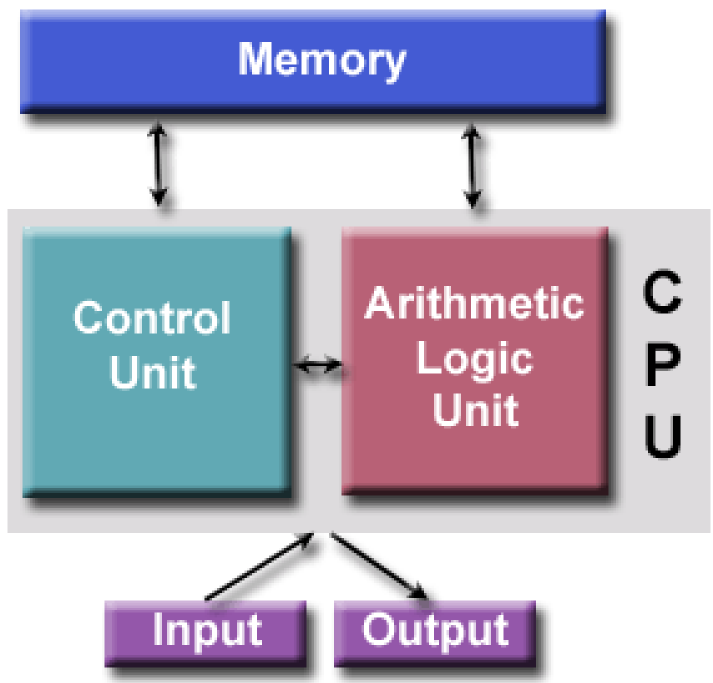
In RNN/LSTM, larger memory implies more parameters

→ Increasing memory size will not increasing parameters

# Neural Turing Machine



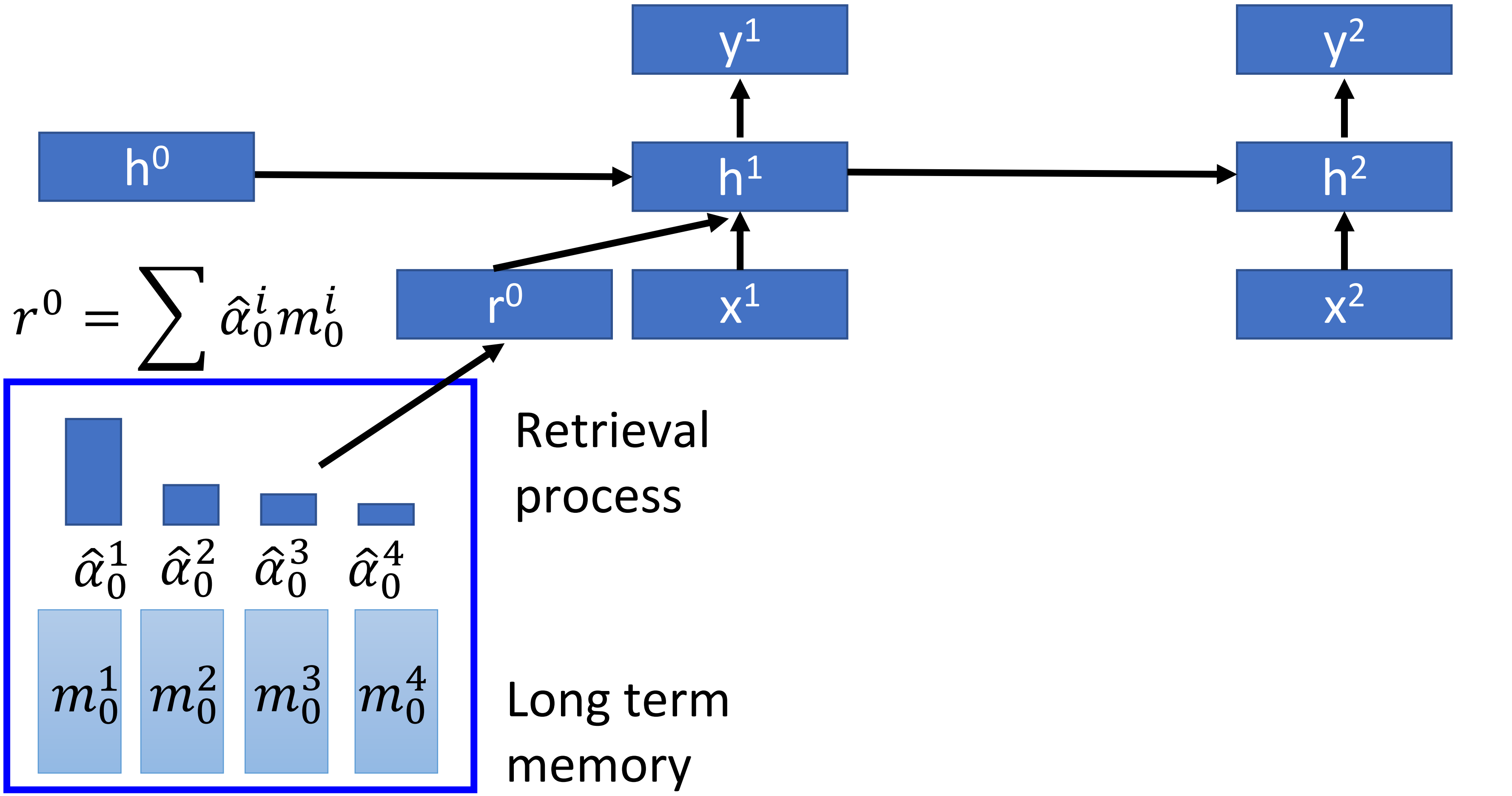
- Von Neumann architecture
- Neural Turing Machine is an advanced RNN/LSTM.




Zhang et al., "Structured Memory for Neural Turing Machines," arXiv, 2015.

# Neural Turing Machine

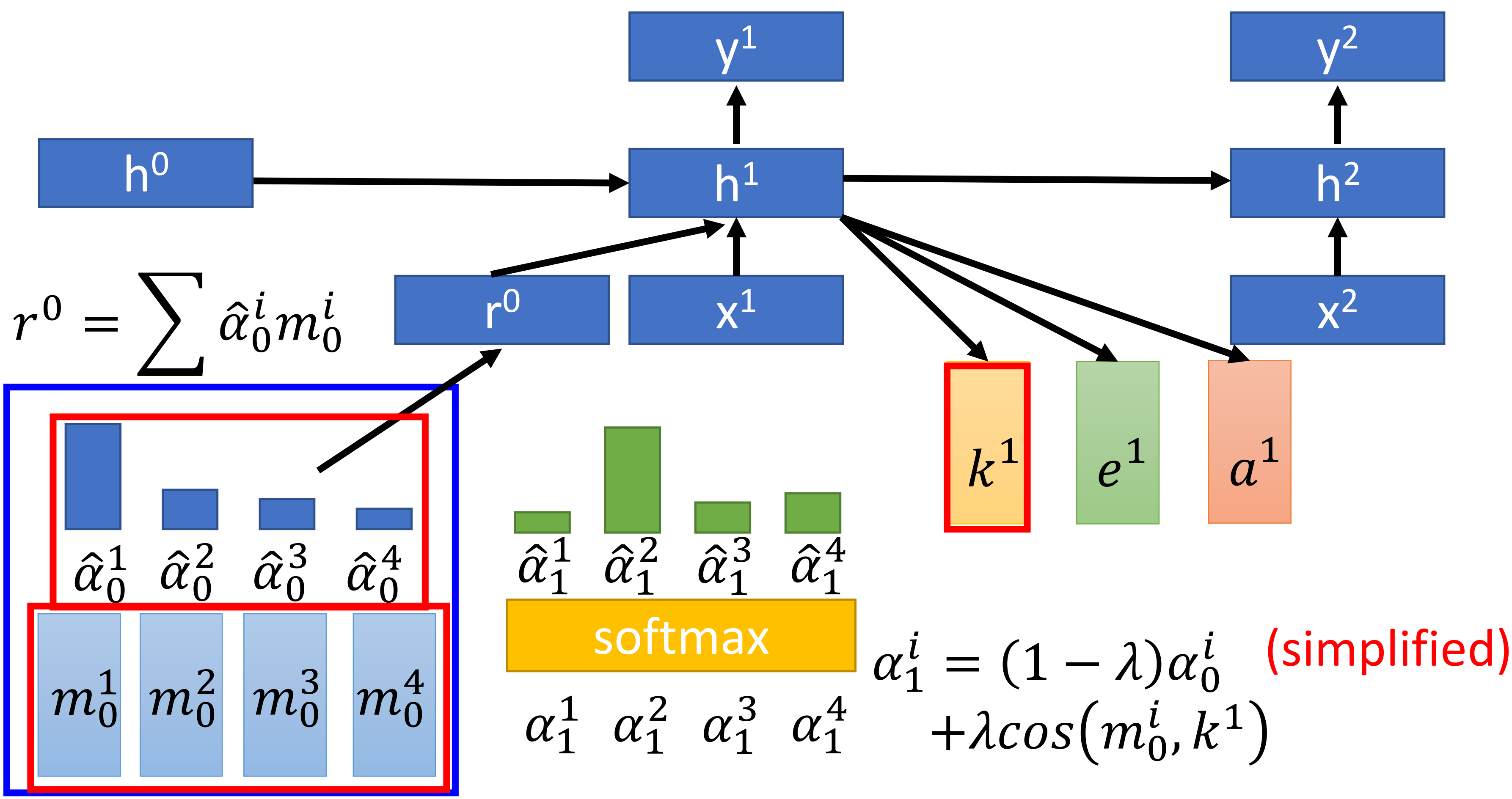





Zhang et al., “Structured Memory for Neural Turing Machines,” arXiv, 2015.

# Neural Turing Machine






Zhang et al., "Structured Memory for Neural Turing Machines," arXiv, 2015.

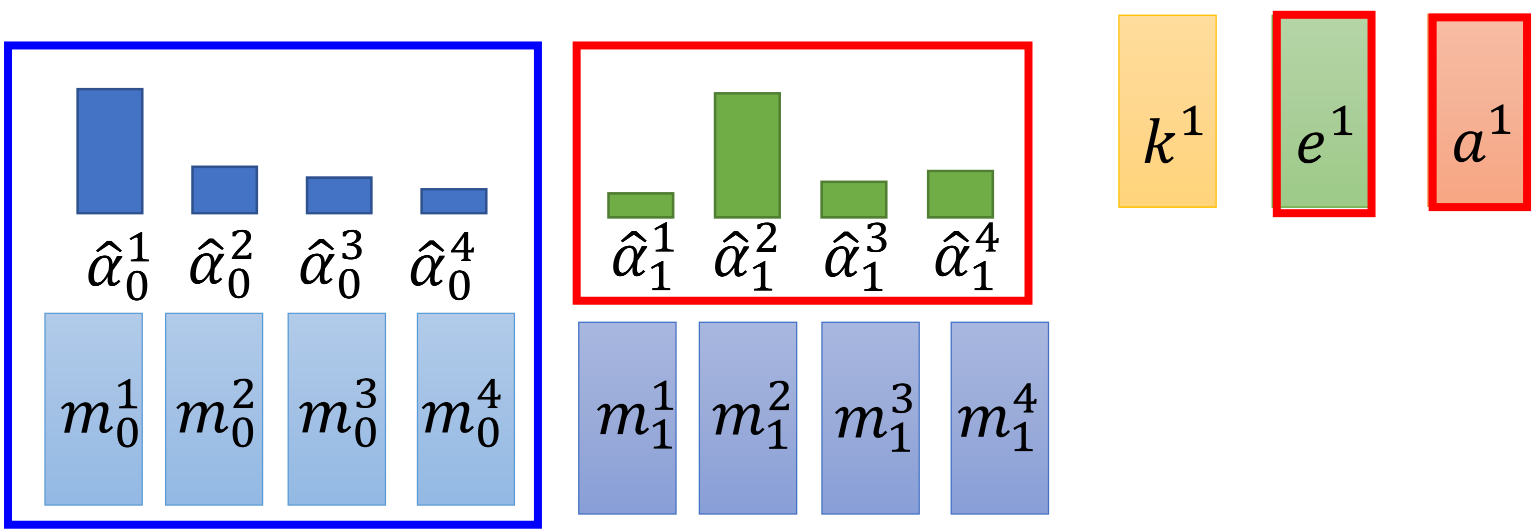
# Neural Turing Machine



$$m_1^i = m_0^i * \left(1 - \hat{\alpha}_1^i \, e^1\right) + \hat{\alpha}_1^i \, a^1$$

(element-wise)

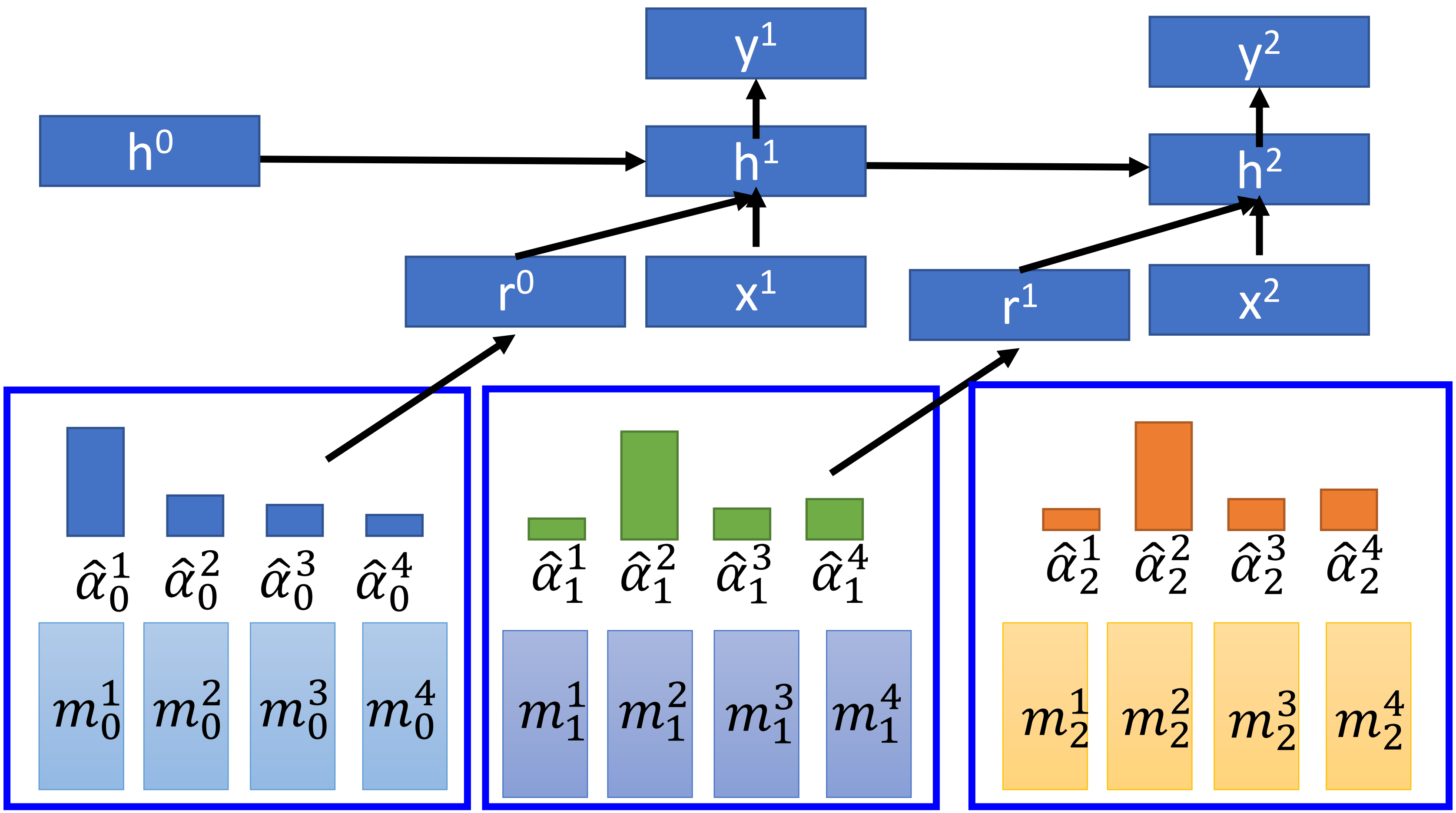
Encode process




Zhang et al., “Structured Memory for Neural Turing Machines,” arXiv, 2015.

# Neural Turing Machine



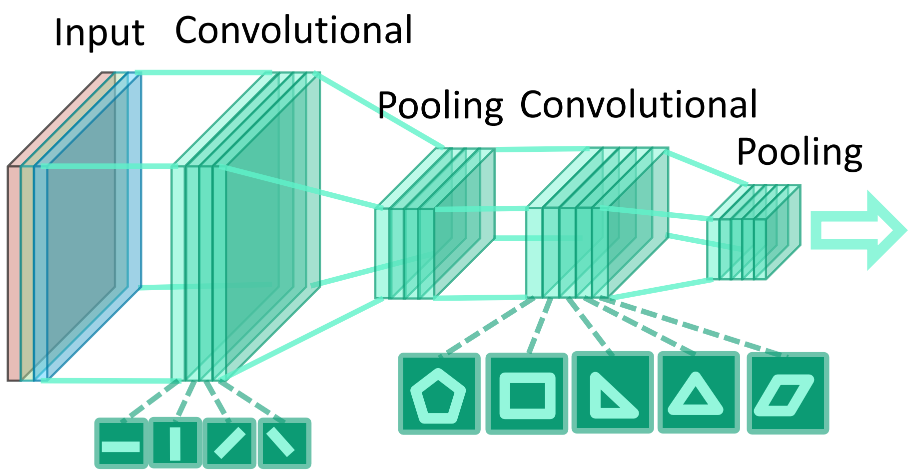



Zhang et al., “Structured Memory for Neural Turing Machines,” arXiv, 2015.

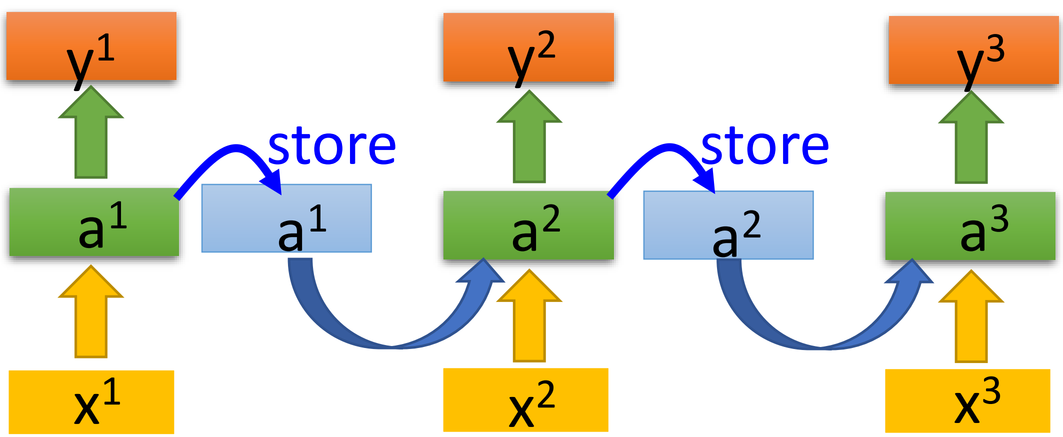
# Concluding Remarks



- Convolutional Neural Network (CNN)




- Recurrent Neural Network (RNN)



# Talk Outline



Part I: Introduction to
Machine Learning & Deep Learning

Part II: Variants of Neural Nets

Part III: Beyond Supervised Learning
& Recent Trends

# PART III

Beyond Supervised Learning & Recent Trend

# Introduction

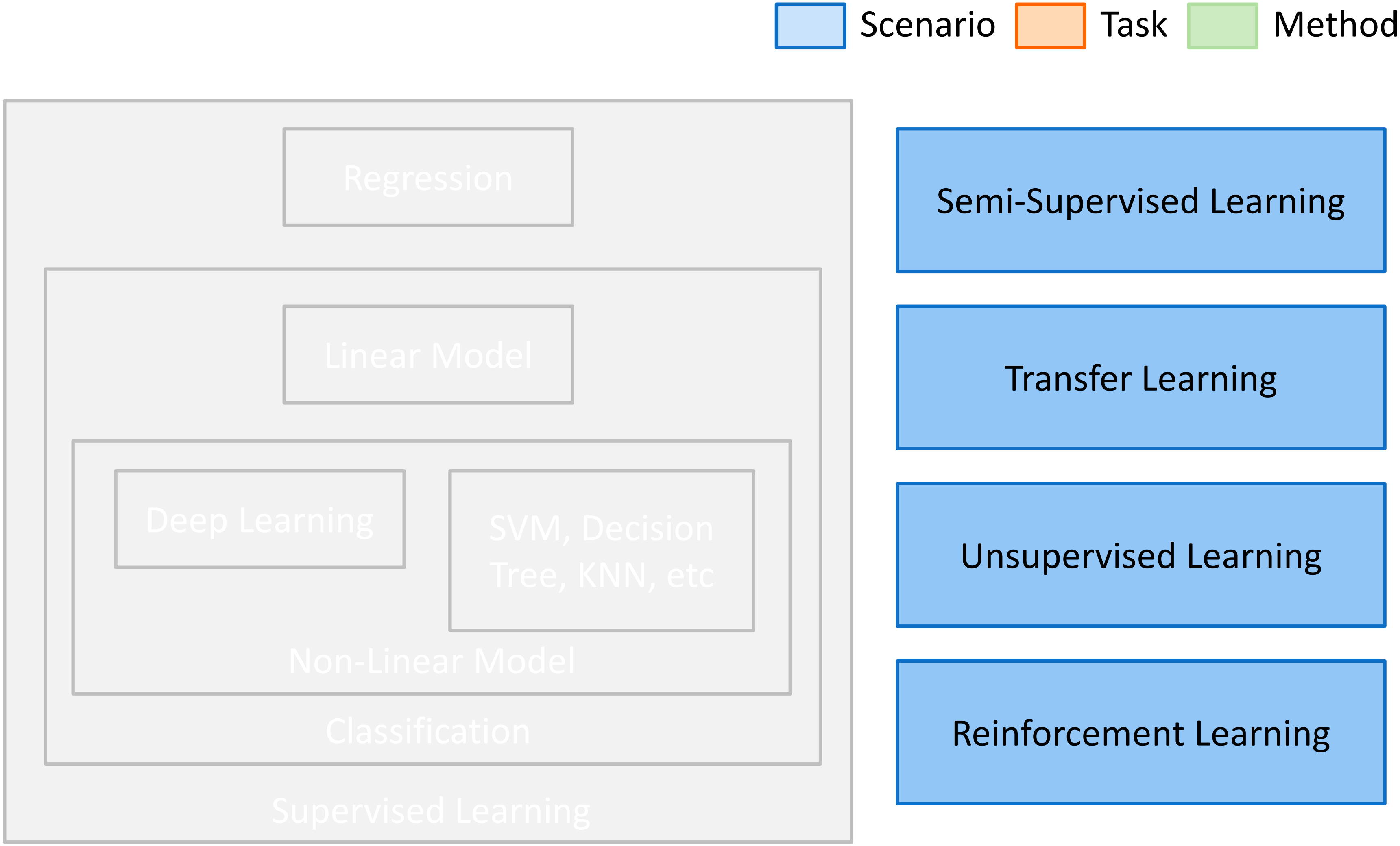


- Big data ≠ Big annotated data
- Machine learning techniques include:
  - Supervised learning (if we have labelled data)
  - Reinforcement learning (if we have an environment for reward)
  - Unsupervised learning (if we do not have labelled data)

What can we do if there is no sufficient labelled training data?

# Machine Learning Map





# Outline



- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Outline



- **Semi-Supervised Learning**
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Semi-Supervised Learning



Labelled data

cat

dog

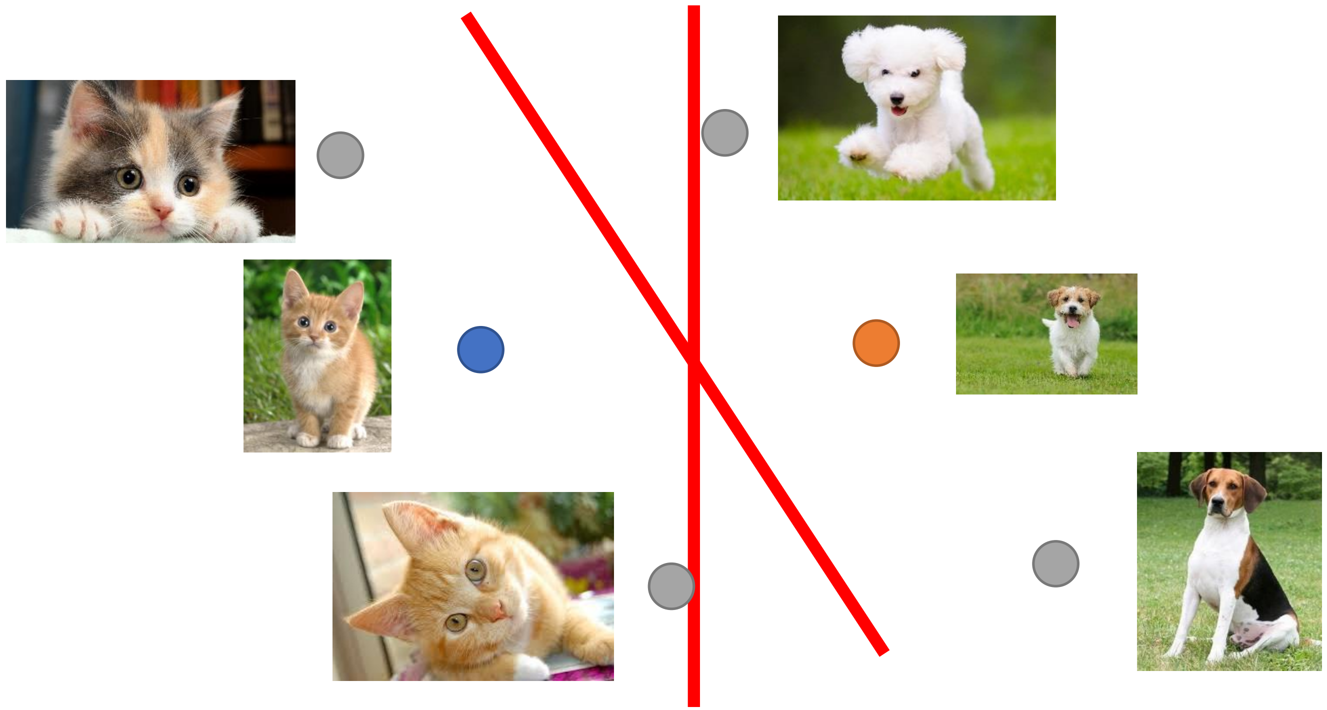
Unlabeled data

(Image of cats and dogs without labeling)

# Semi-Supervised Learning



- Why semi-supervised learning helps?



The distribution of the unlabeled data provides some cues

# Outline



- Semi-Supervised Learning
- **Transfer Learning**
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Transfer Learning



Labelled data

cat

dog

Labeled data

elephant

elephant

tiger

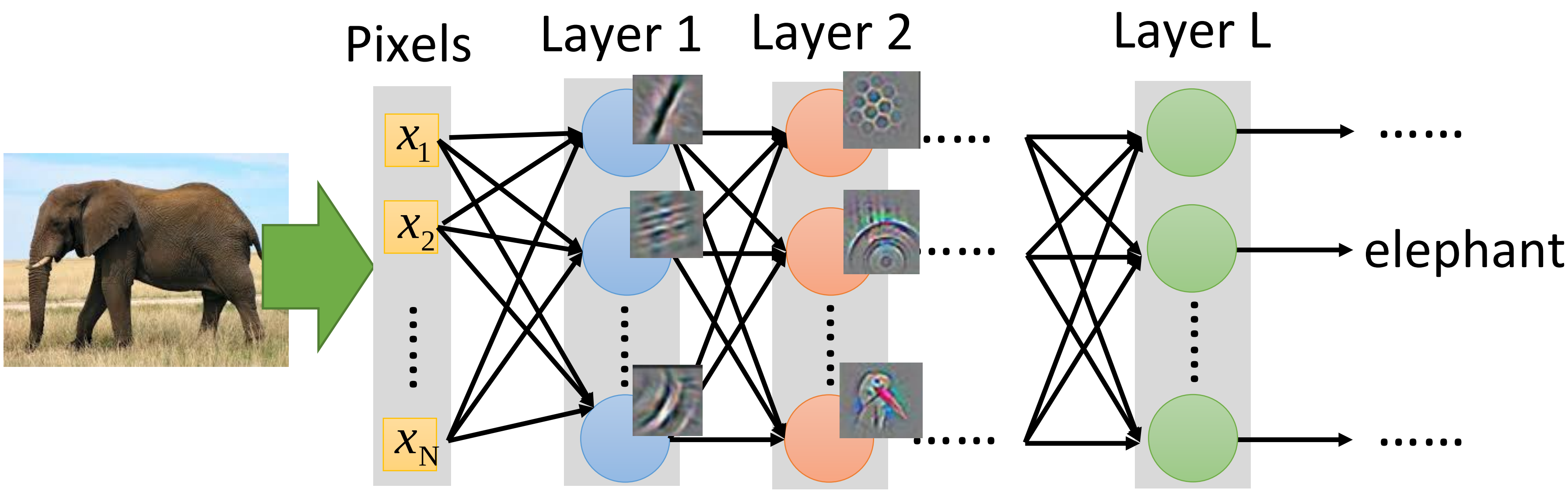
tiger

Not related to the task considered

# Transfer Learning



- Widely used on image processing
  - Using sufficient labeled data to learn a CNN
  - Using this CNN as feature extractor

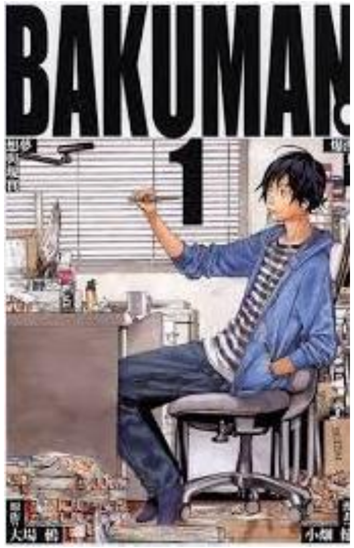


# Transfer Learning Example



| ***研究生 online*** | ***漫畫家 online*** |
| --- | --- |
| 研究生 | 漫畫家 |
| 指導教授 | 責編 |
| 跑實驗 | 畫分鏡 |
| 投稿期刊 | 投稿 jump |

研究生生存守則




爆漫王

# Outline



- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Outline



- Semi-Supervised Learning
- Transfer Learning
- **Unsupervised Learning**
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Unsupervised Learning



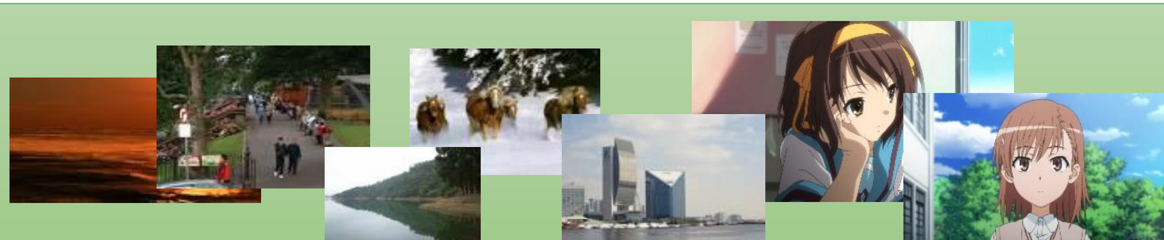
- The unlabeled data sometimes is not related to the task

Labelled data

cat

dog

Unlabeled data



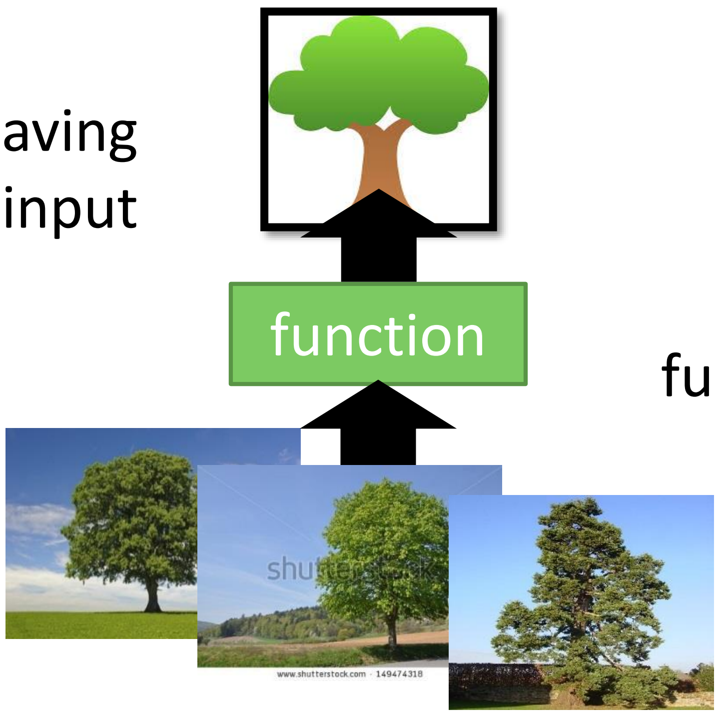
(Just crawl millions of images from the Internet)

# Unsupervised Learning



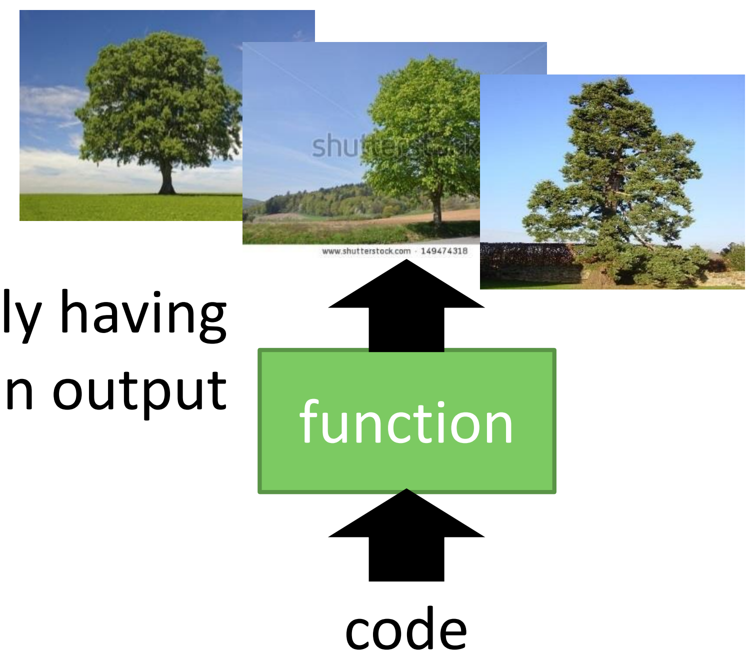
- 化繁為簡

only having
function input




- 無中生有

only having
function output



# Unsupervised Learning

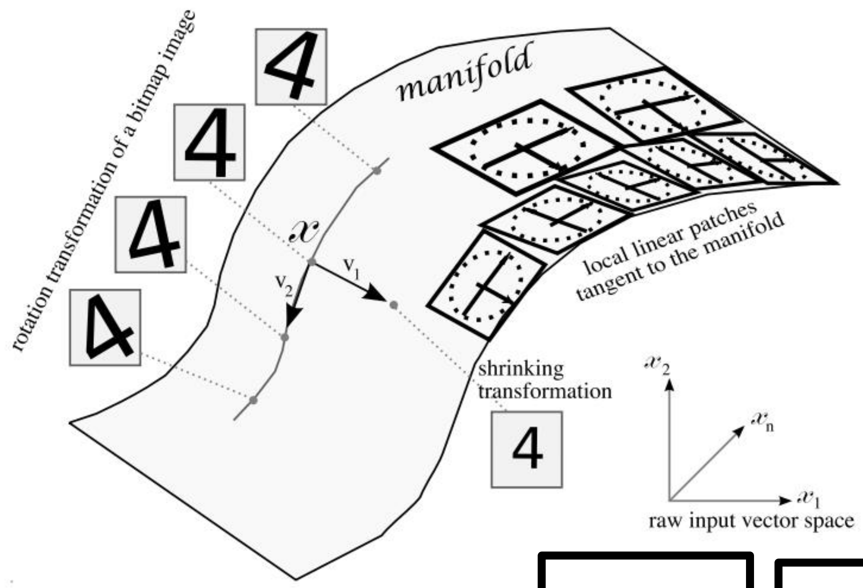


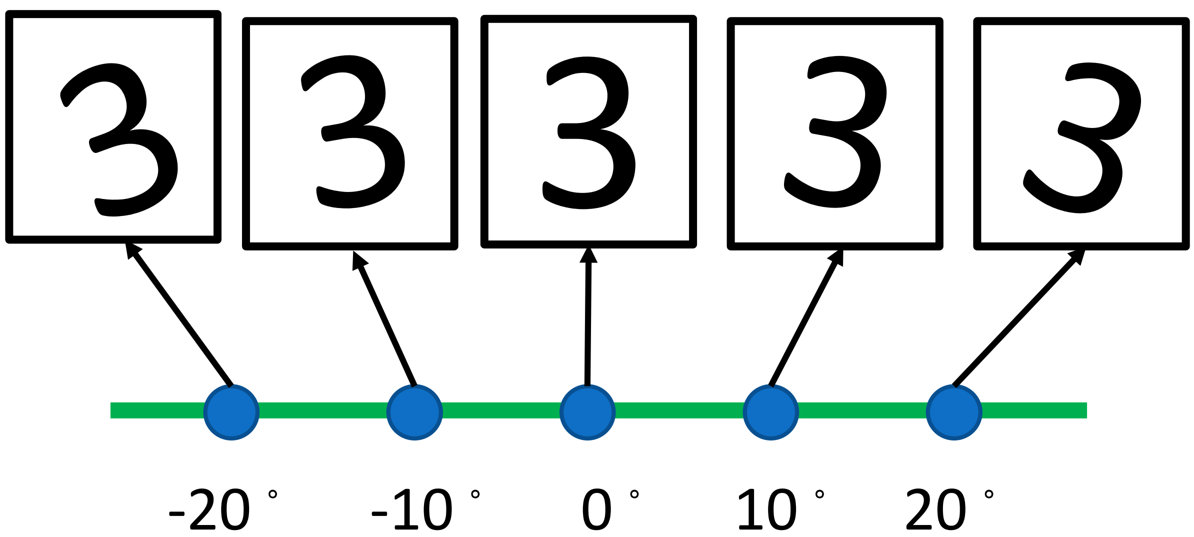
- How does self-taught learning work?
- Why does unlabeled and unrelated data help the tasks?

Finding latent factors that control the observations

# Latent Factors for Handwritten Digits

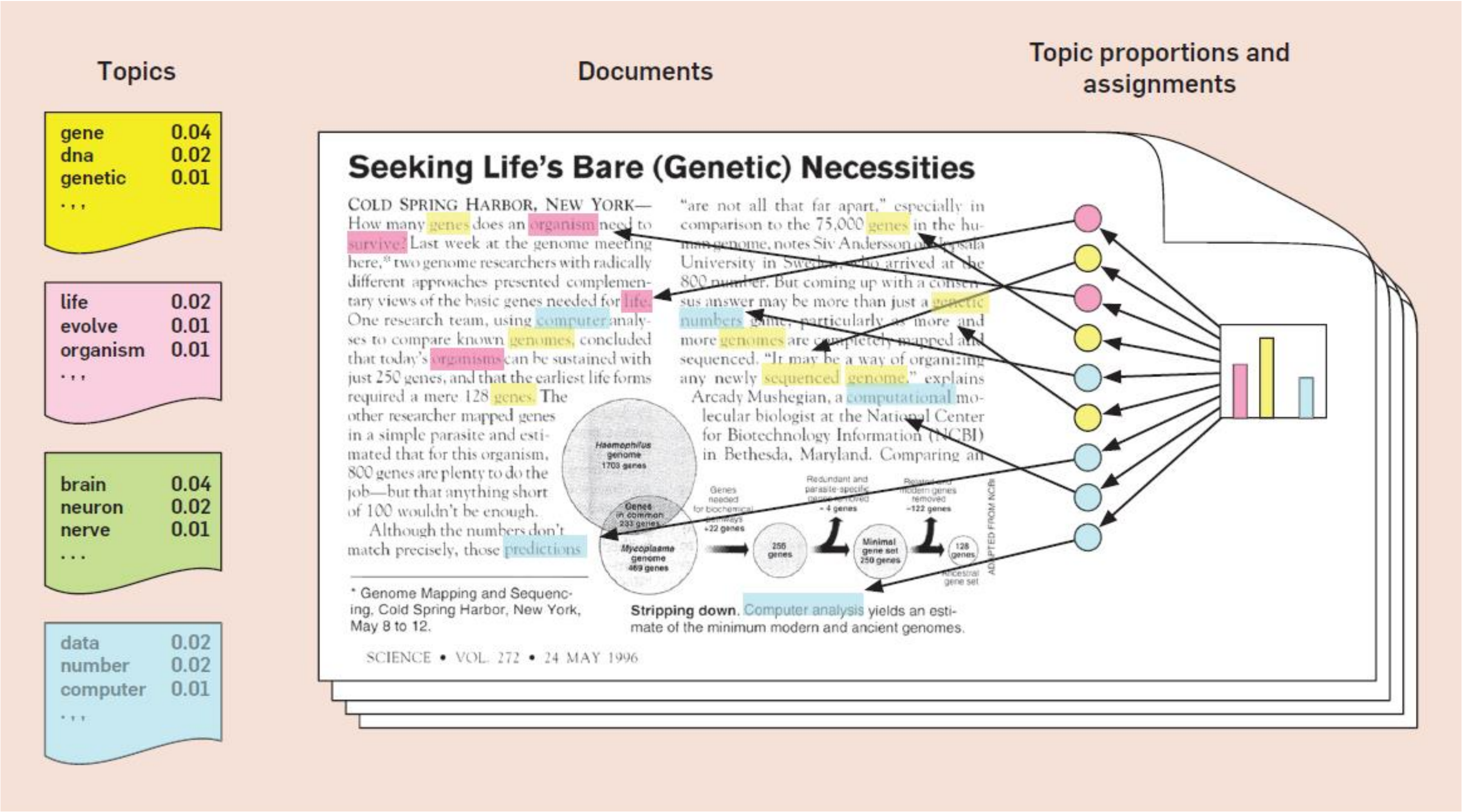




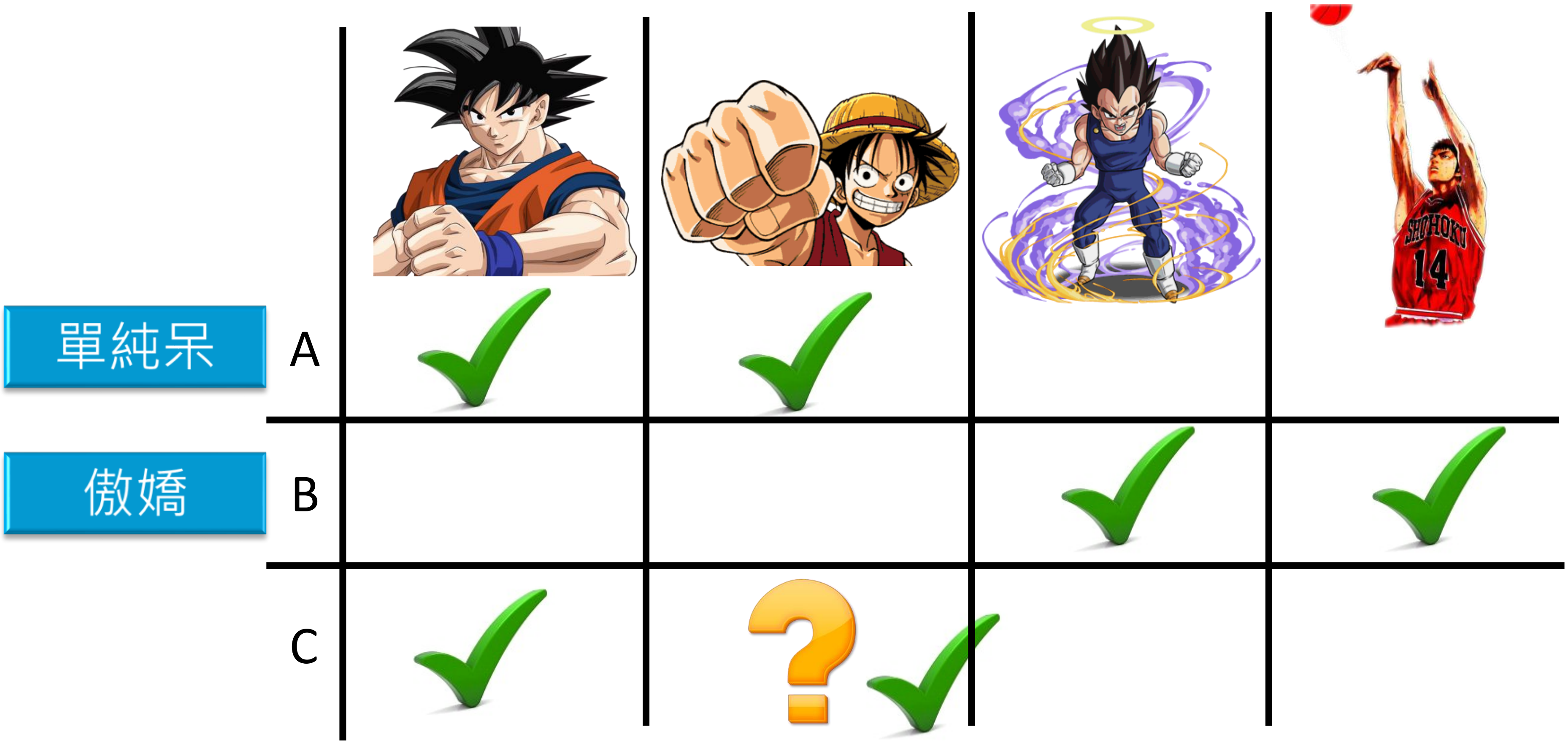
# Latent Factors for Documents





# Latent Factors for Recommendation





# Latent Factor Exploitation



- Handwritten digits

The handwritten images are composed of **strokes**

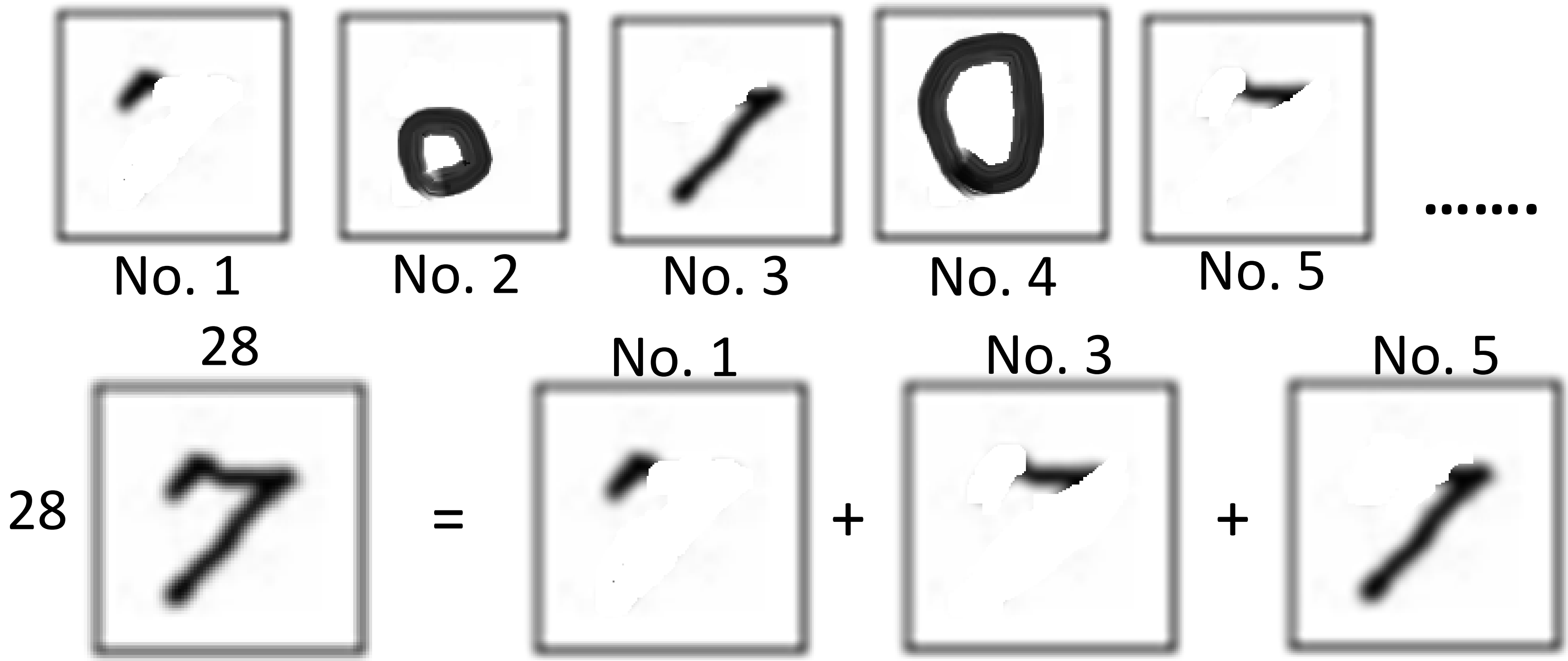
***<u>Strokes (Latent Factors)</u>***

No. 1 No. 2 No. 3 No. 4 No. 5 .......

# Latent Factor Exploitation



***Strokes (Latent Factors)***




Represented by
28 X 28 = 784 pixels

[1 0 1 0 1 0 .......]
(simpler representation)

# Outline



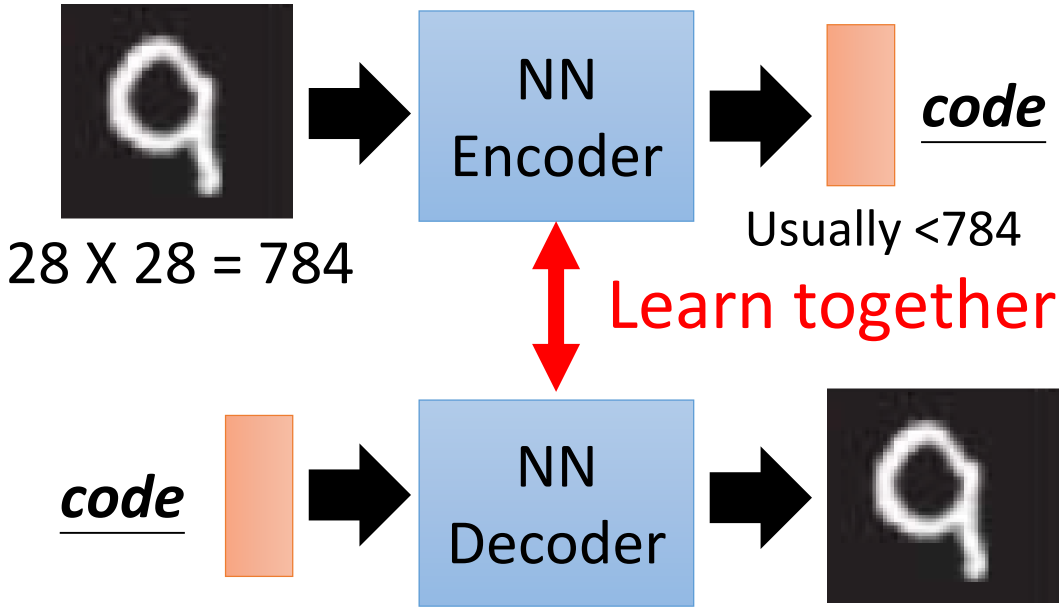
- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Autoencoder



- Represent a digit using 28 X 28 dimensions
- Not all 28 X 28 images are digits

Idea: represent the images of digits in a more compact way

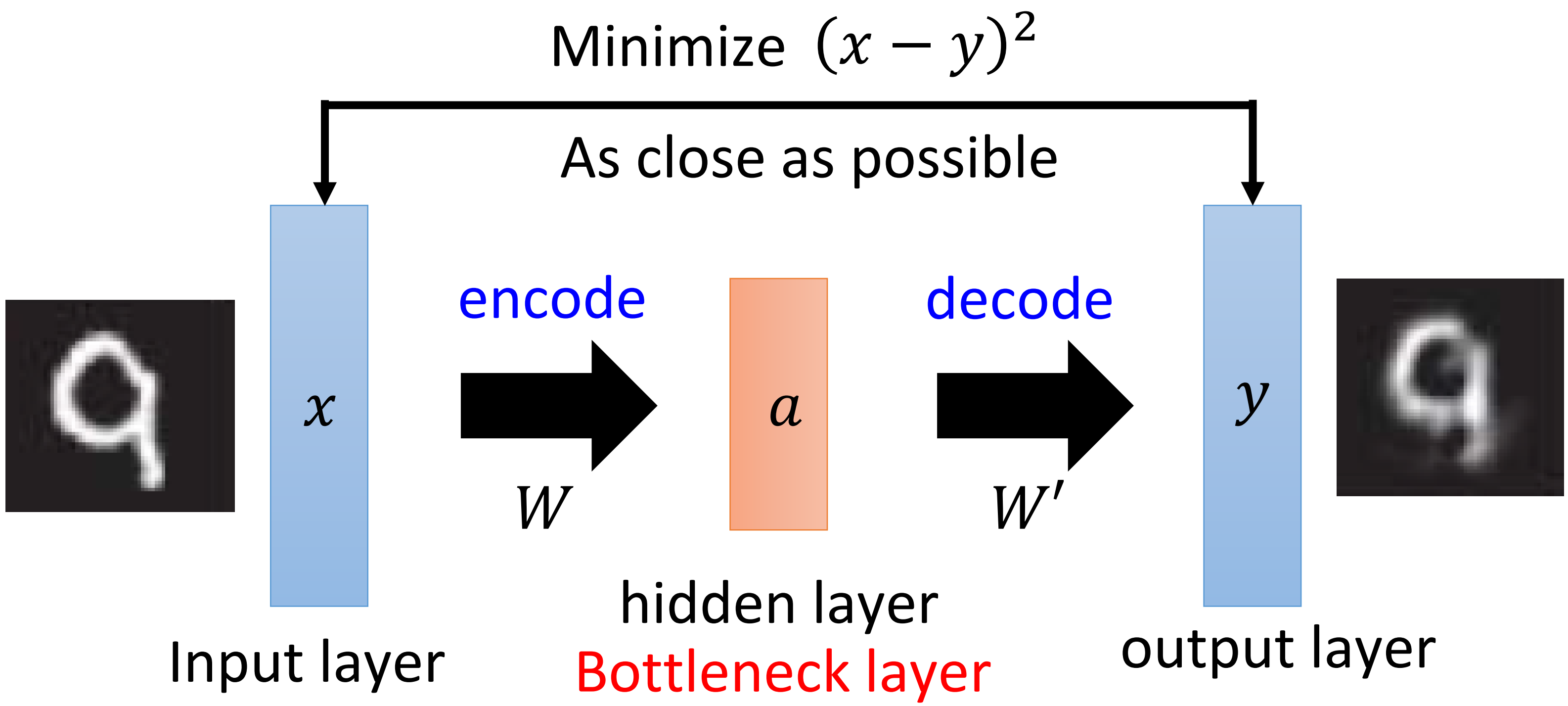



Compact representation of the input object

Can reconstruct the original object

# Autoencoder





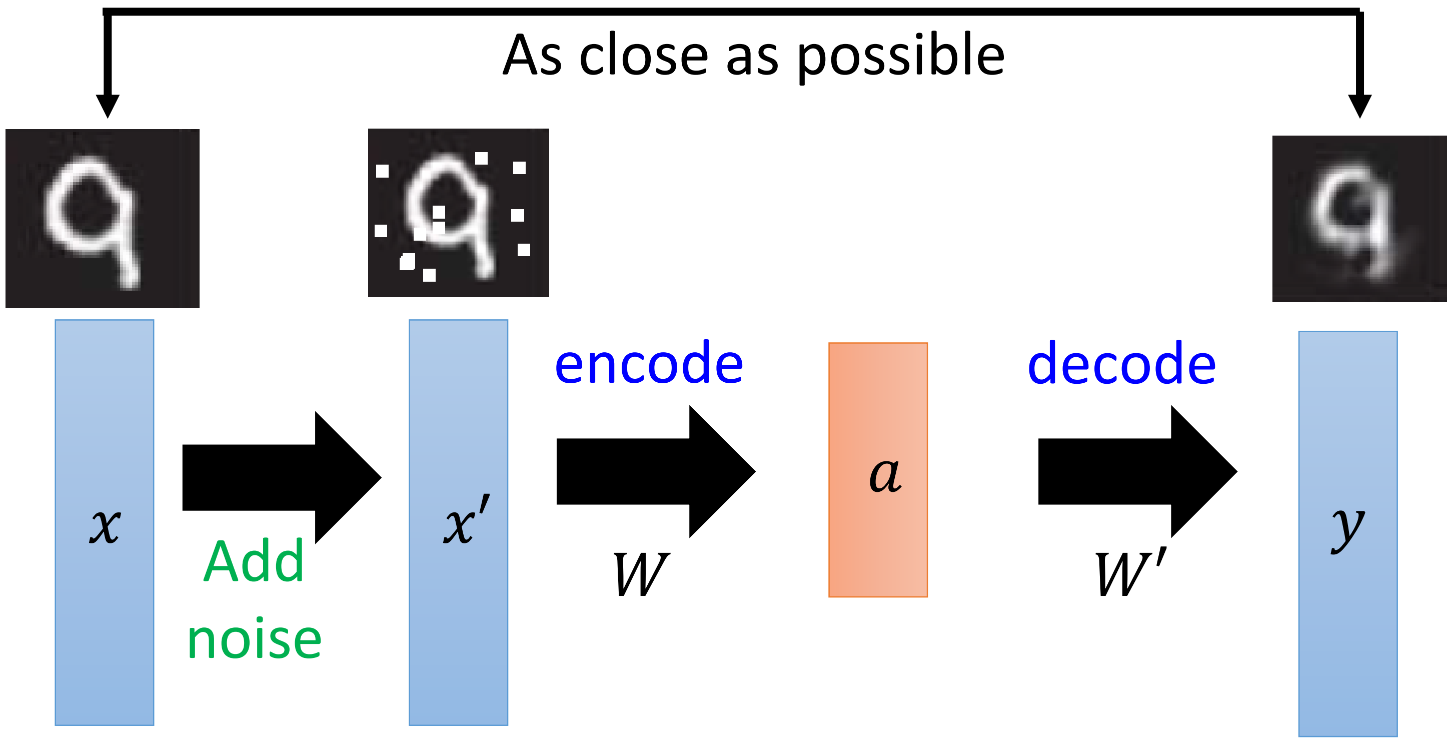

$a = \sigma(Wx + b)$ $\quad$ $y = \sigma(W'a + b')$

Output of the hidden layer is the code

# Autoencoder

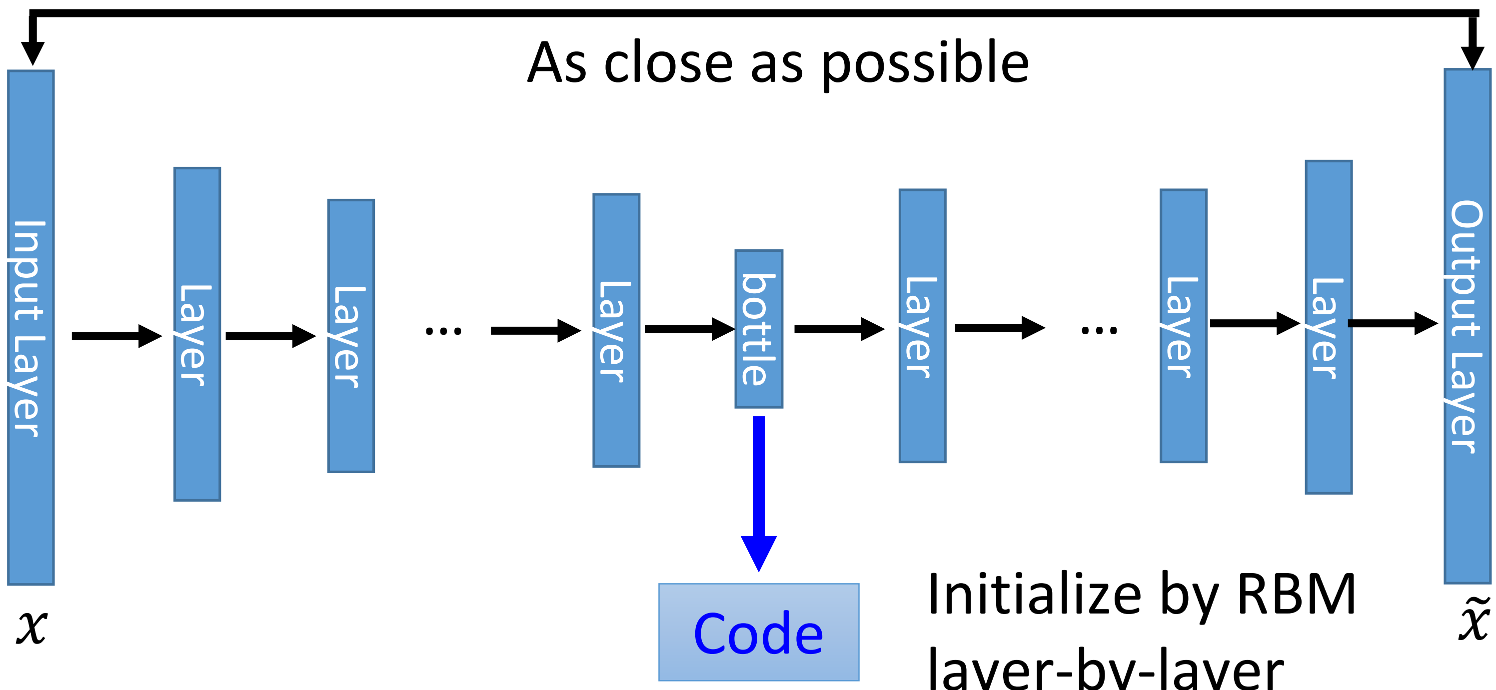


- De-noising auto-encoder



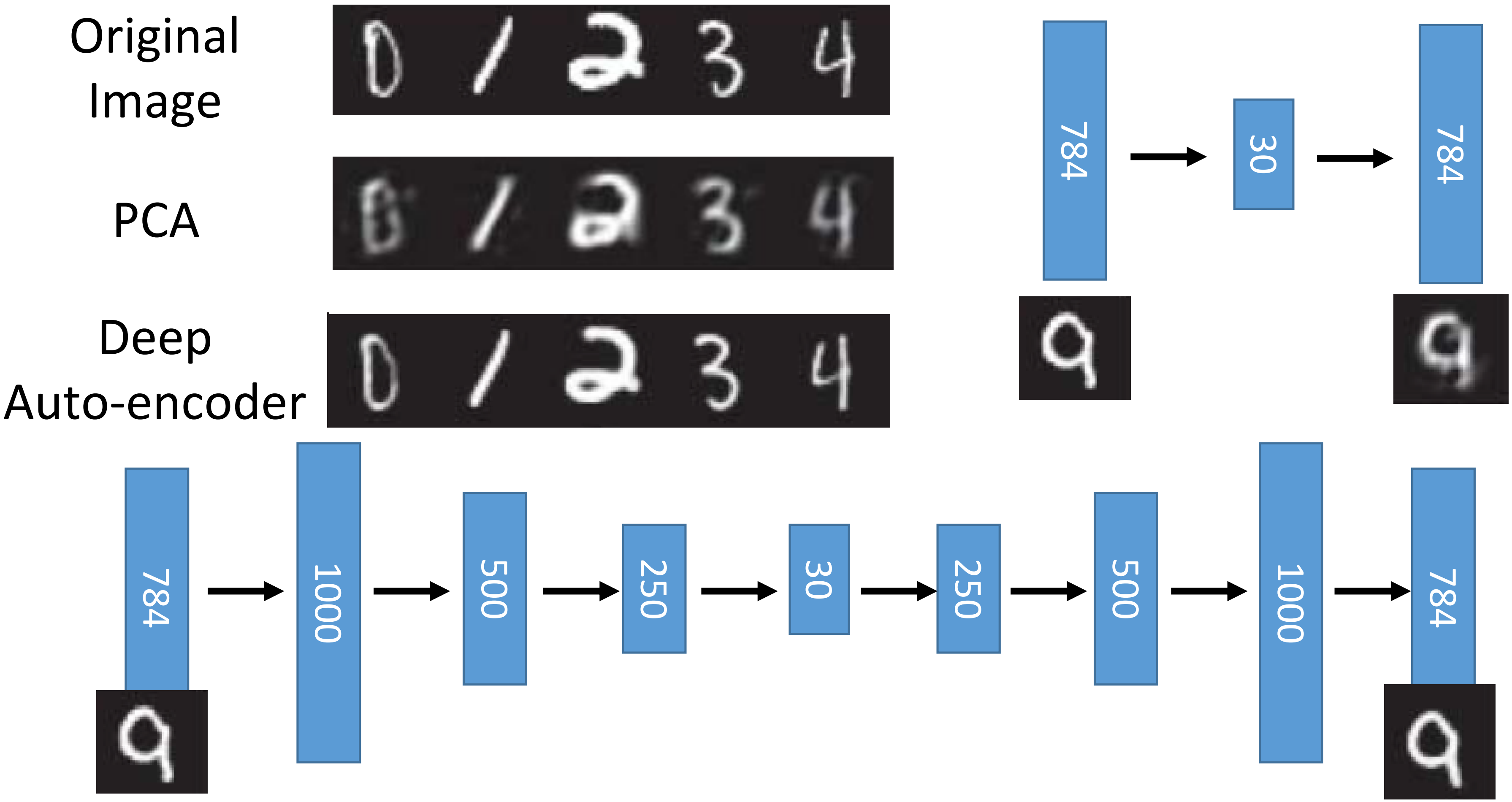
# Deep Autoencoder





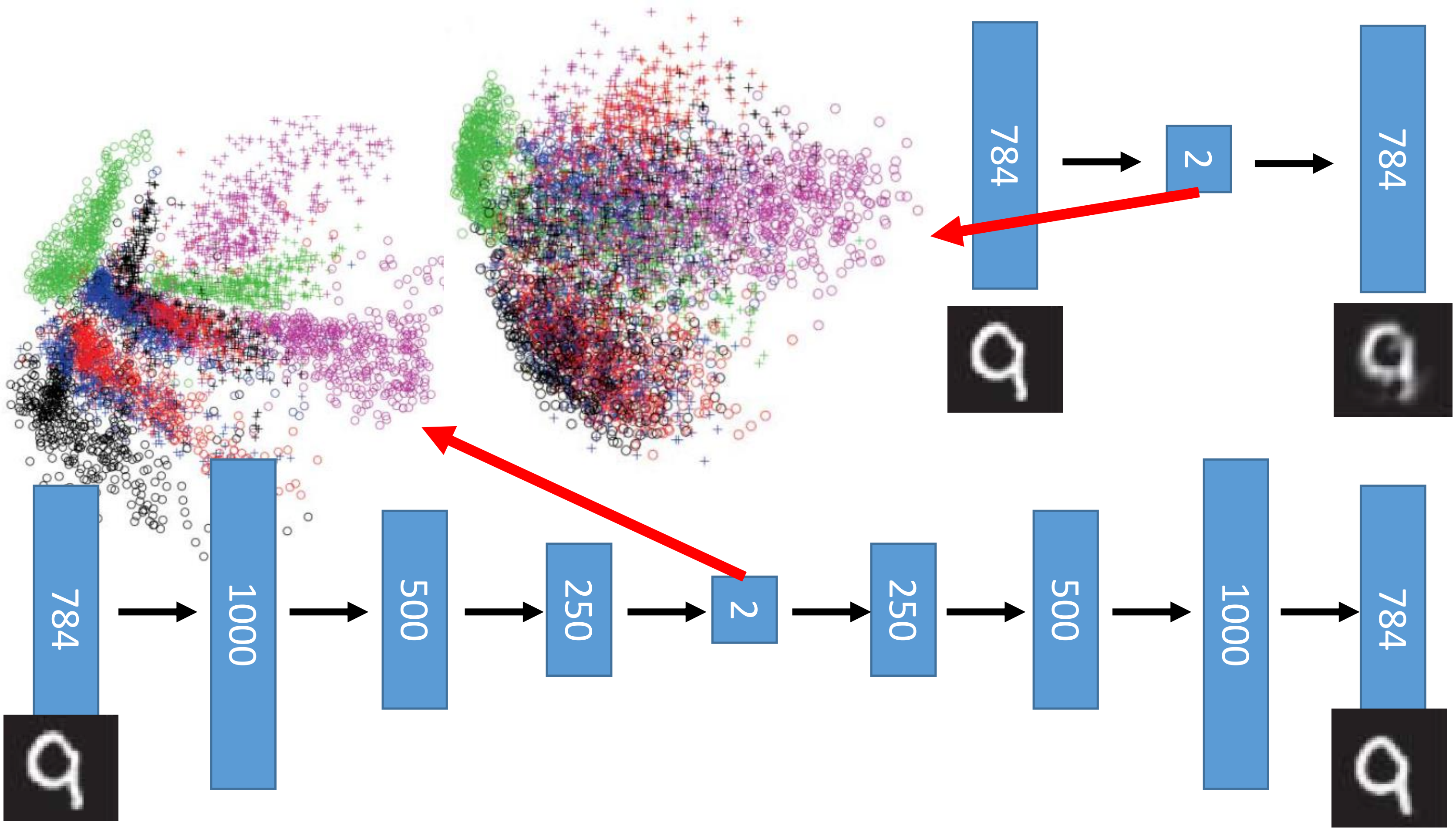
# Deep Autoencoder





# Feature Representation

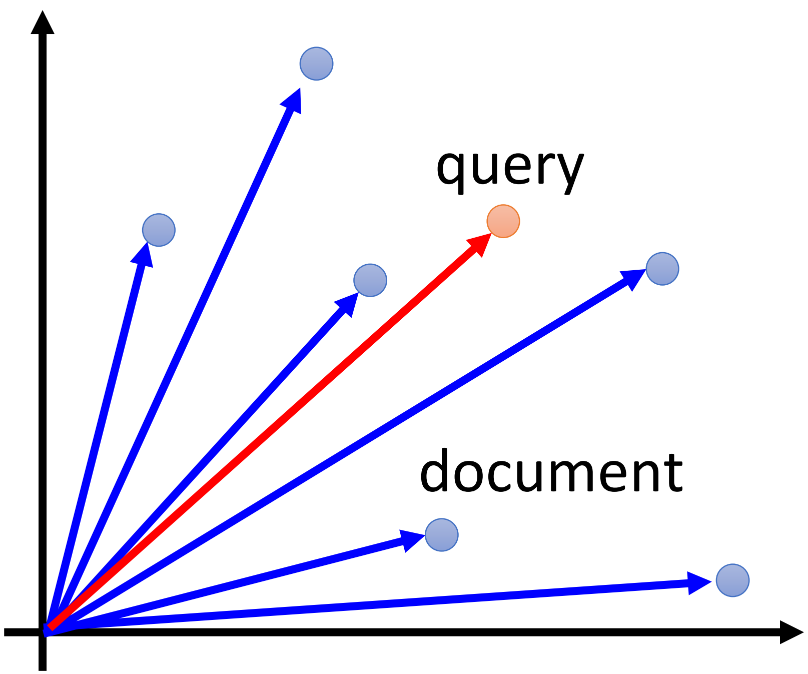




# Auto-encoder – Text Retrieval

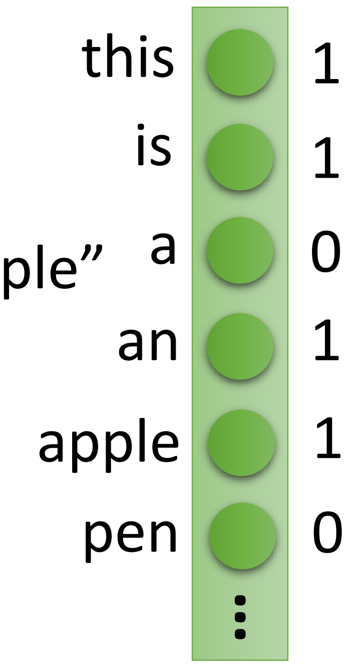


***Vector Space Model***




***Bag-of-word***

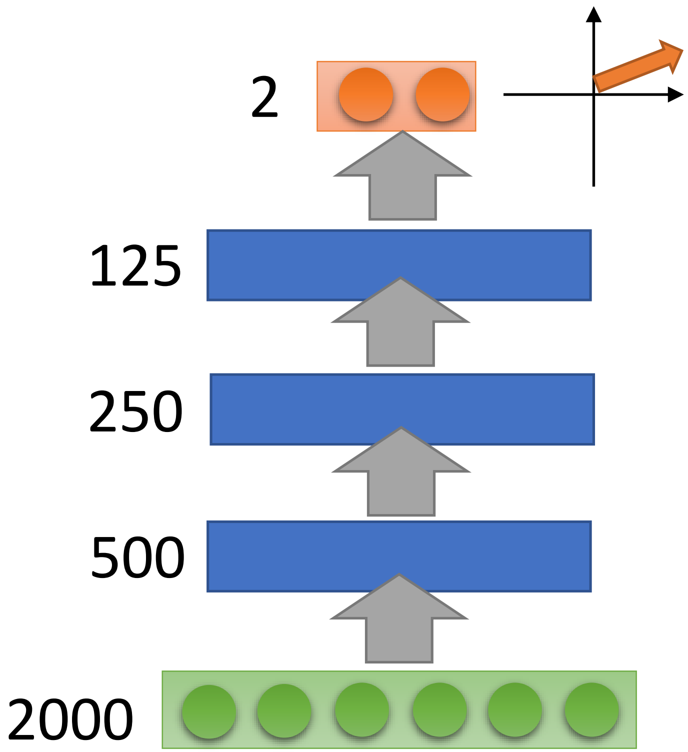
word string:
"This is an apple"

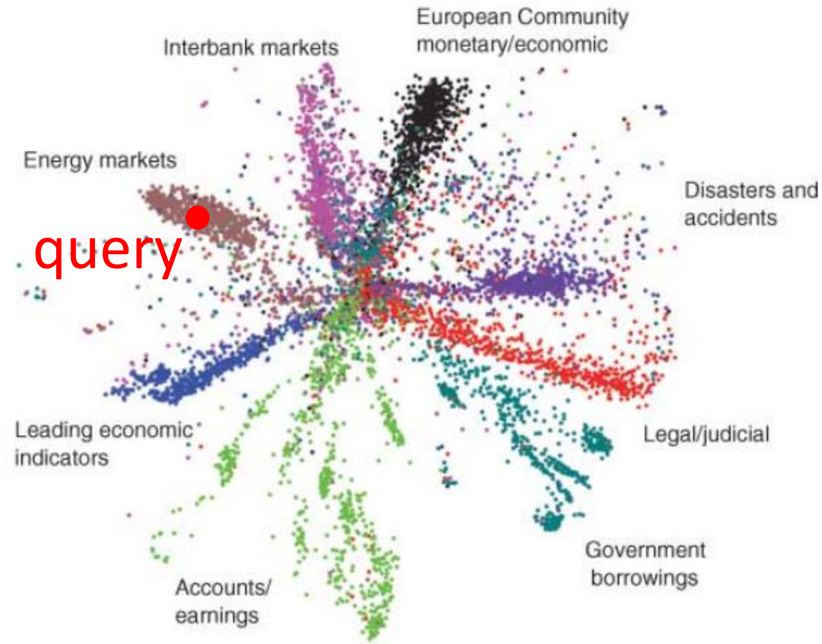



Semantics are not considered

# Autoencoder – Text Retrieval






The documents talking about the same thing will have close code

Bag-of-word (document or query)

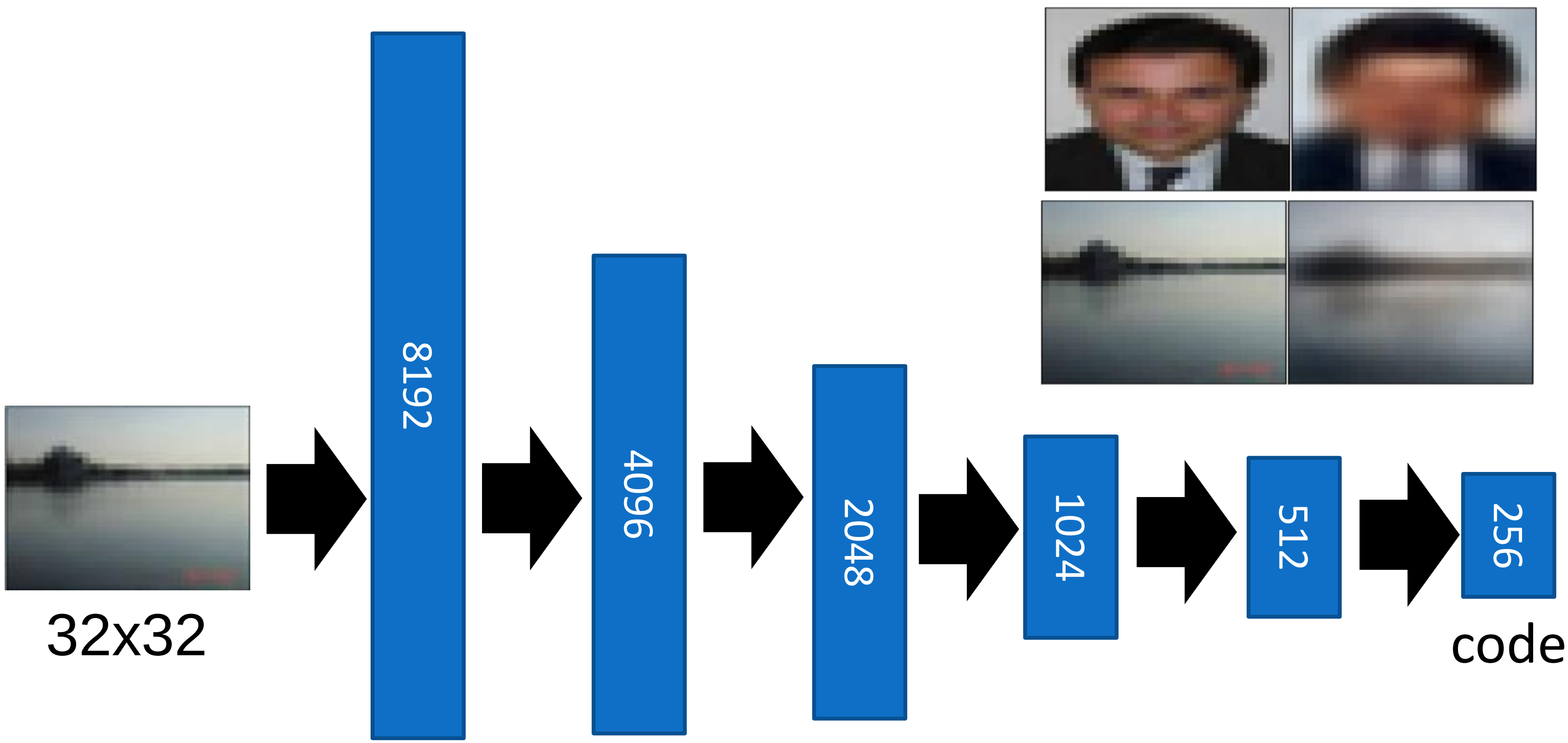
# Autoencoder – Similar Image Retrieval



- Retrieved using Euclidean distance in pixel intensity space

# Autoencoder – Similar Image Retrieval



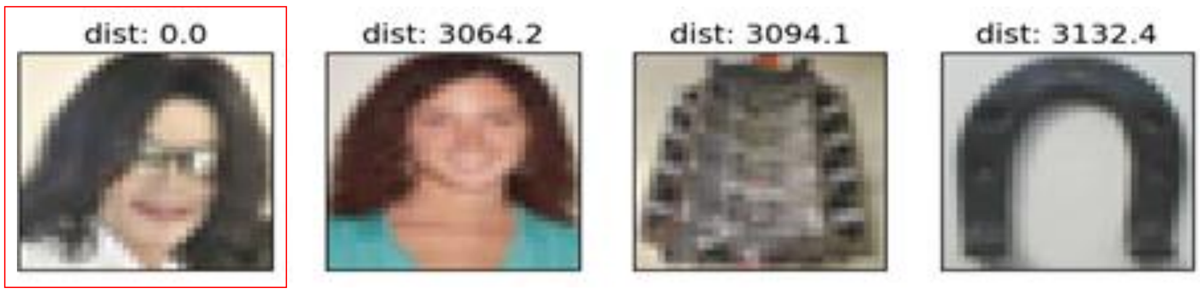



(crawl millions of images from the Internet)

# Autoencoder – Similar Image Retrieval



- Images retrieved using Euclidean distance in pixel intensity space



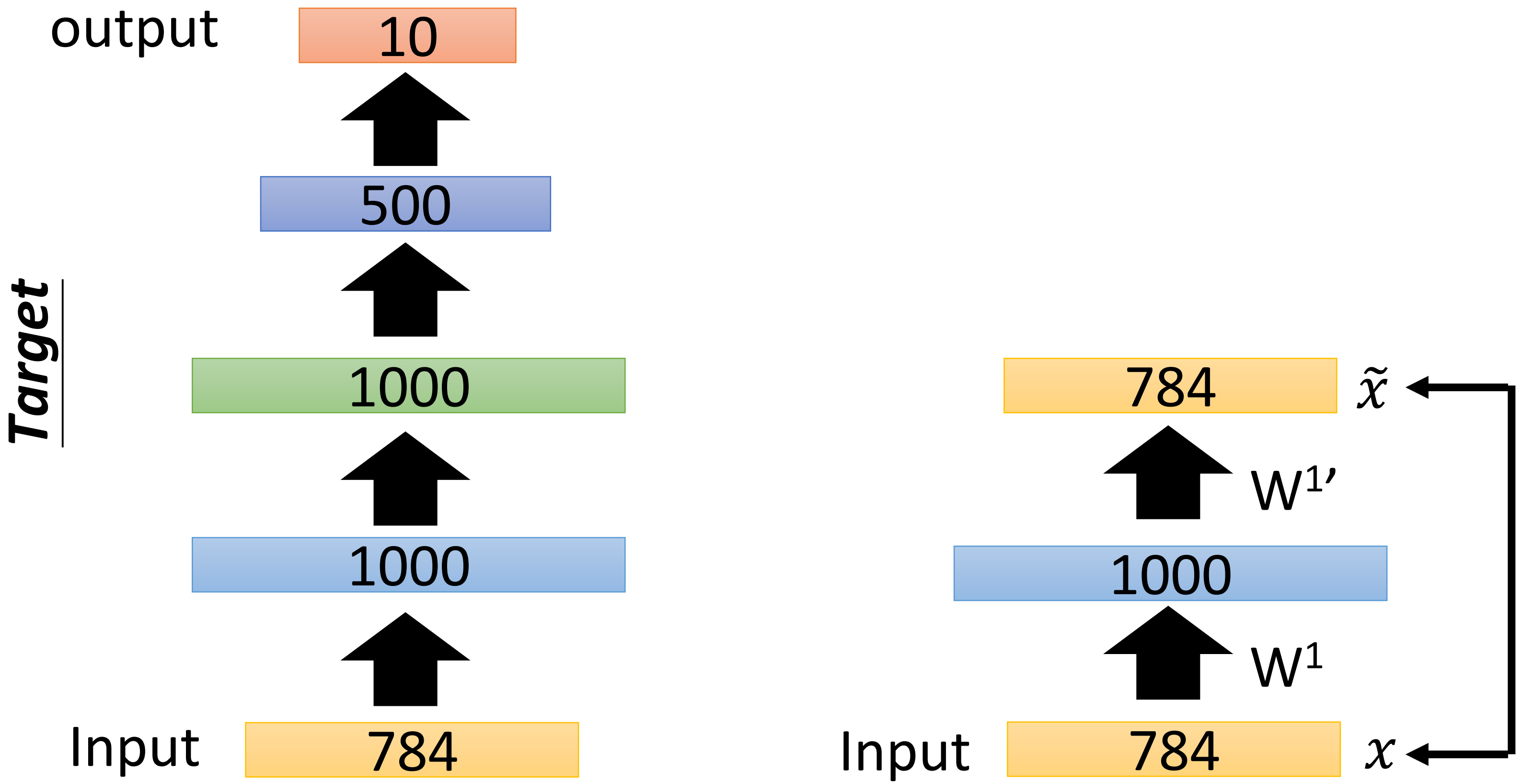

- Images retrieved using 256 codes


Learning the useful latent factors

# Autoencoder for DNN Pre-Training



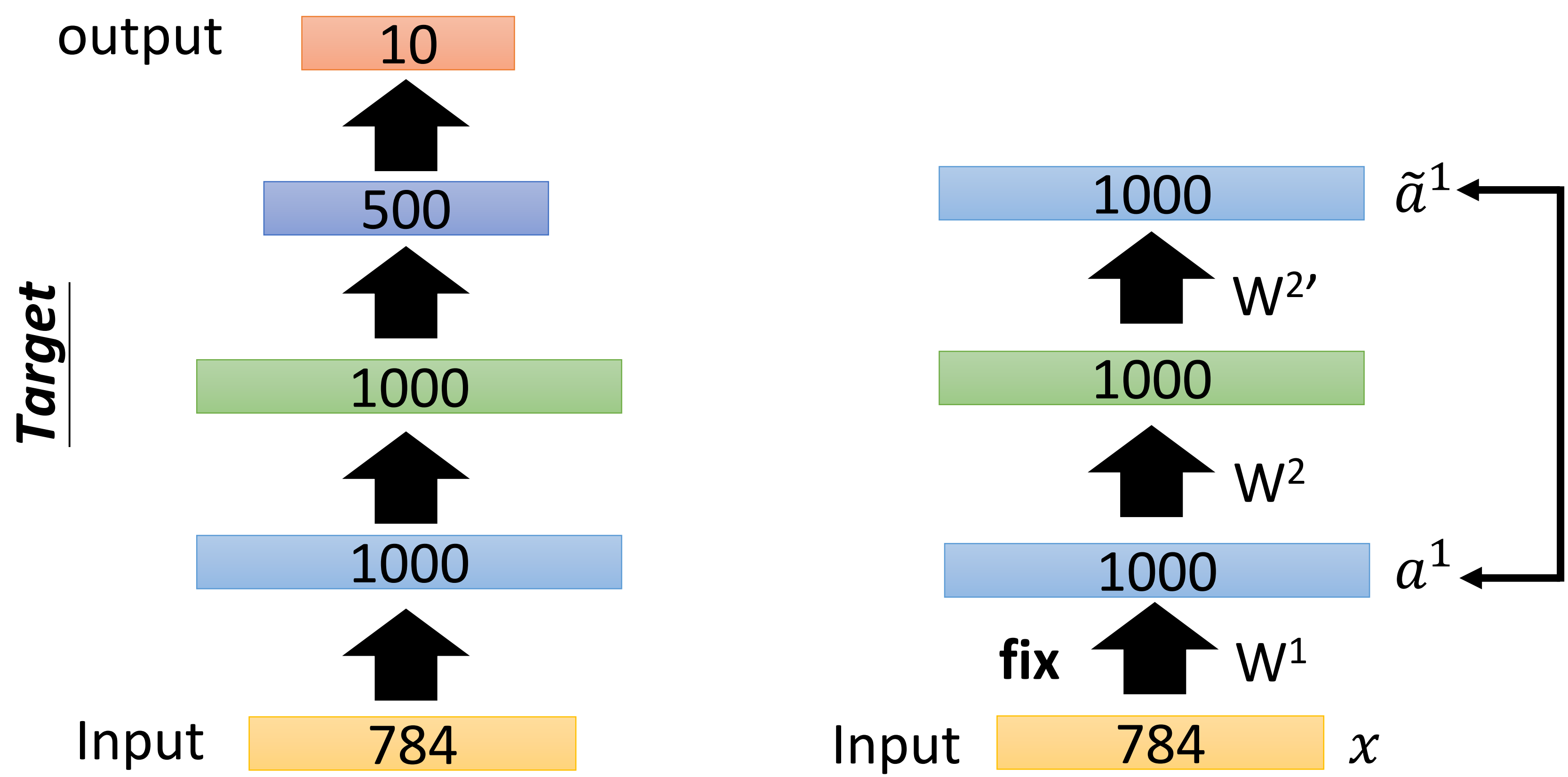
- Greedy layer-wise pre-training *again*



# Autoencoder for DNN Pre-Training



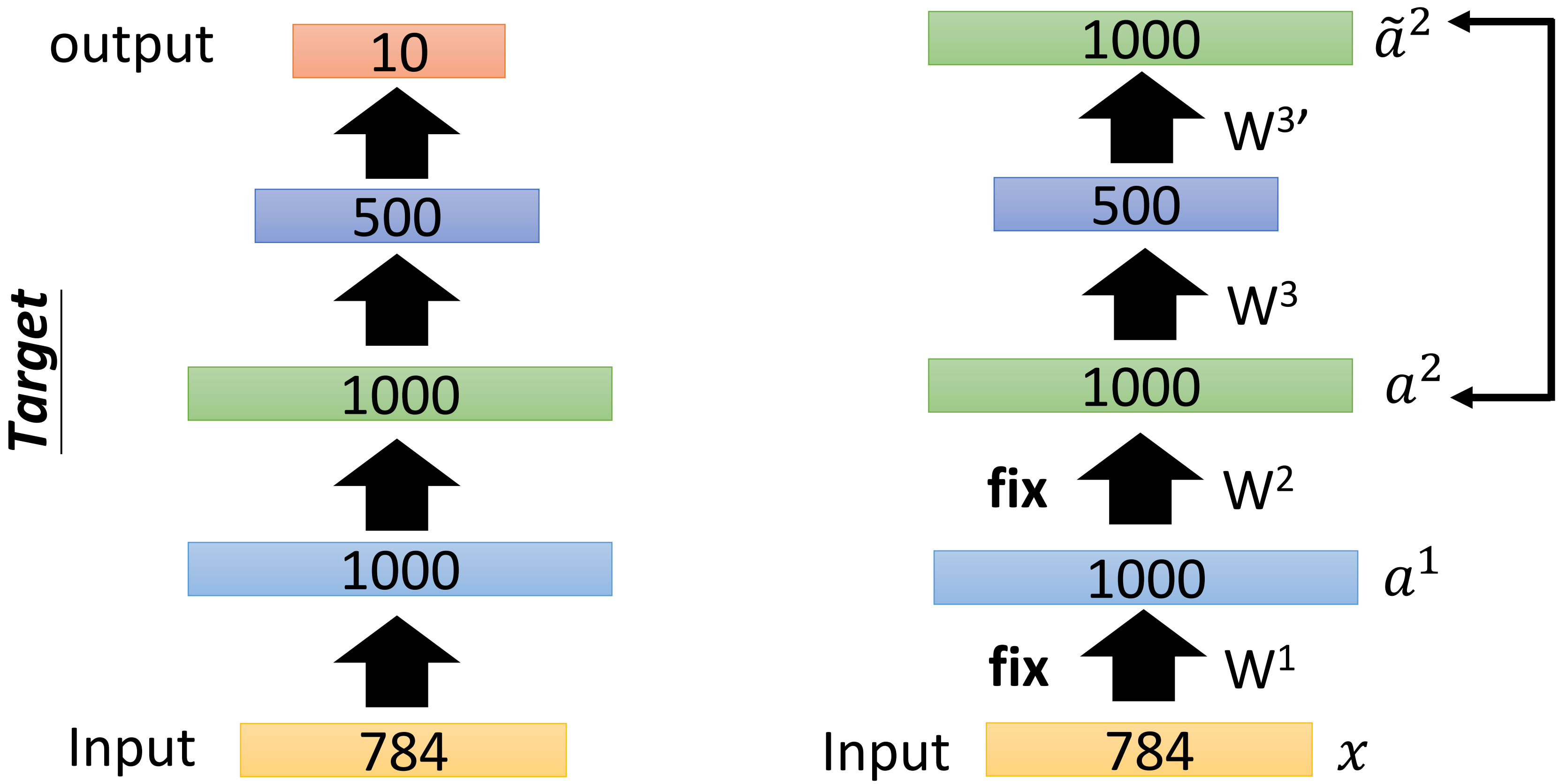
- Greedy layer-wise pre-training *again*



# Autoencoder for DNN Pre-Training



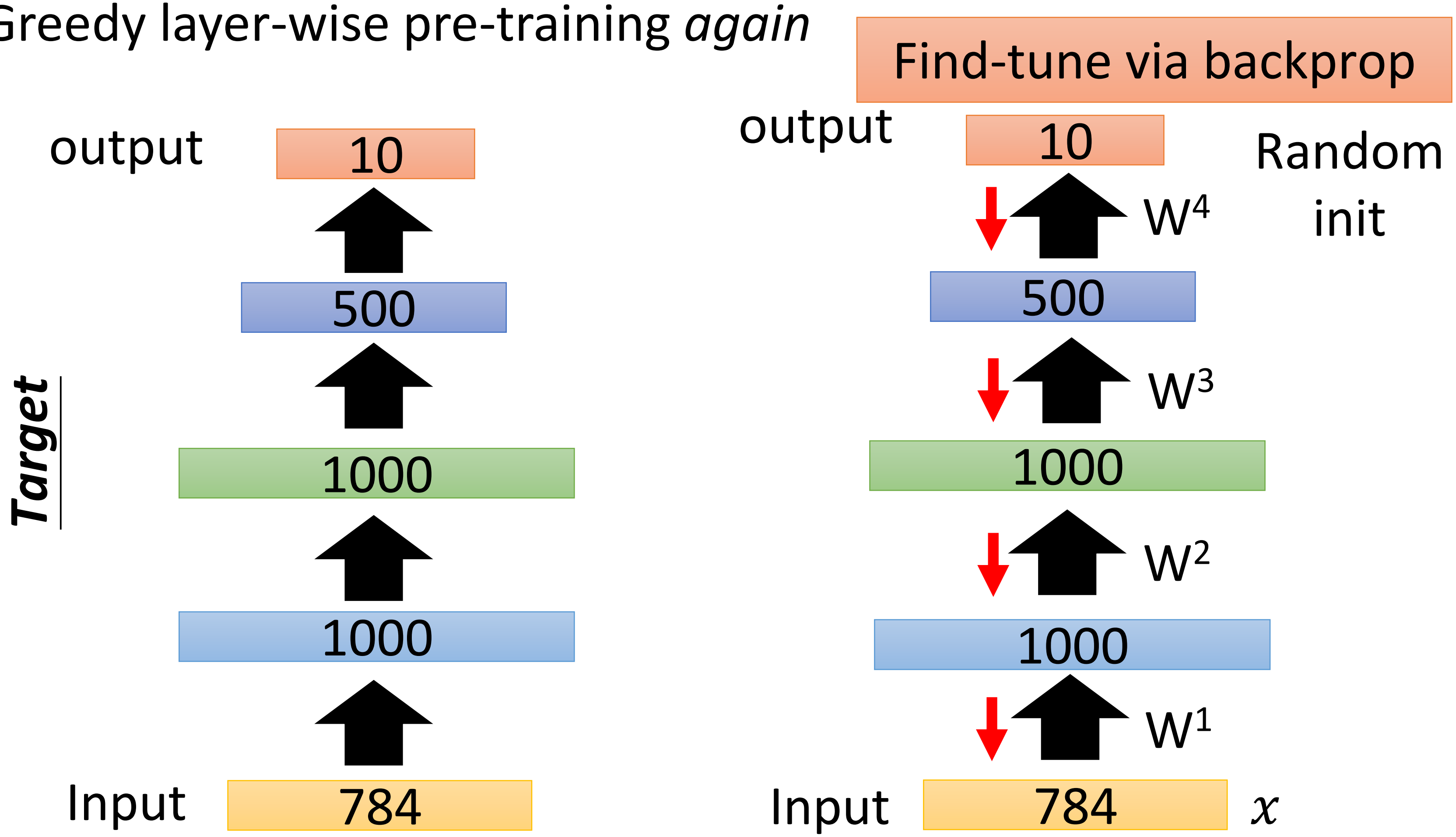
- Greedy layer-wise pre-training *again*



# Autoencoder for DNN Pre-Training



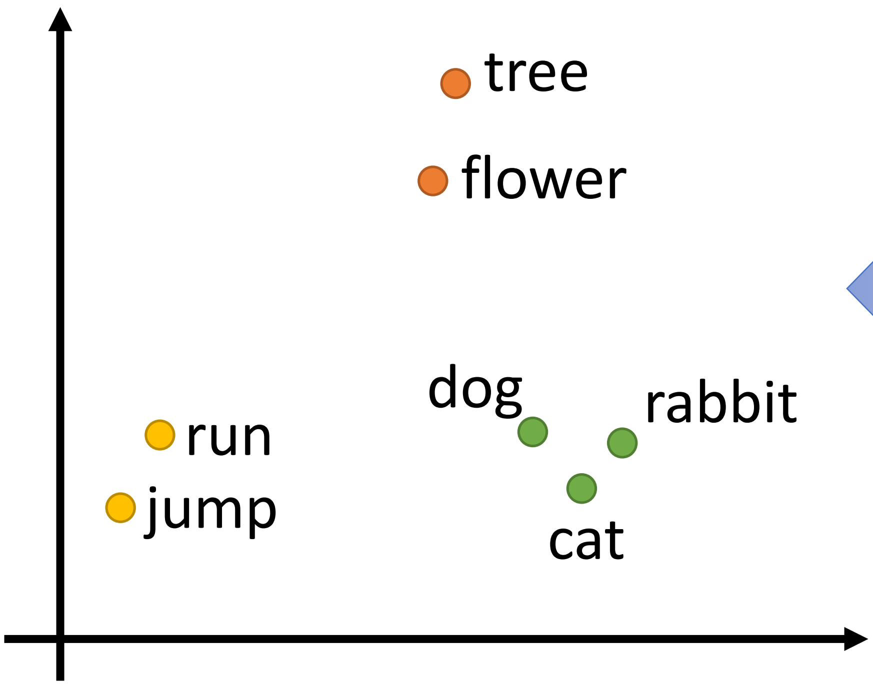
- Greedy layer-wise pre-training *again*

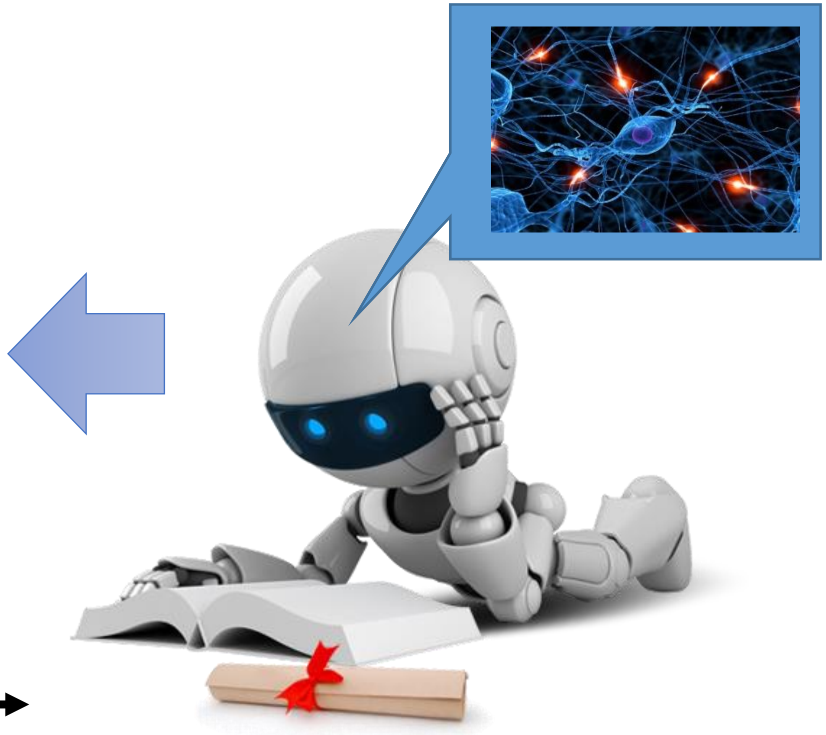


# Word Vector/Embedding



- Machine learn the meaning of words from reading a lot of documents without supervision



# Word Embedding



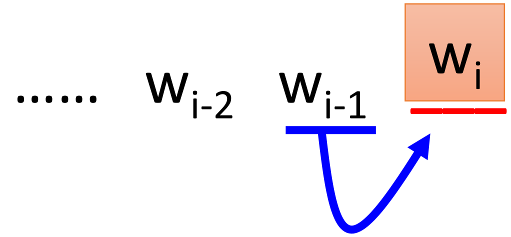
- Machine learn the meaning of words from reading a lot of documents without supervision
- A word can be understood by its context

蔡英文、馬英九 are something very similar

You shall know a word by the company it keeps

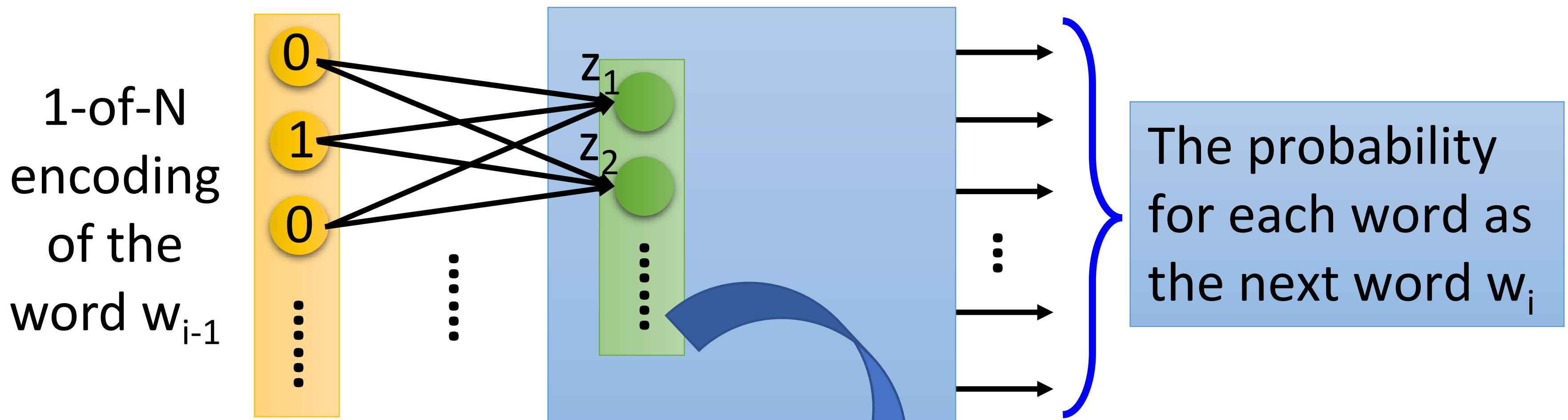
馬英九 520宣誓就職

蔡英文 520宣誓就職

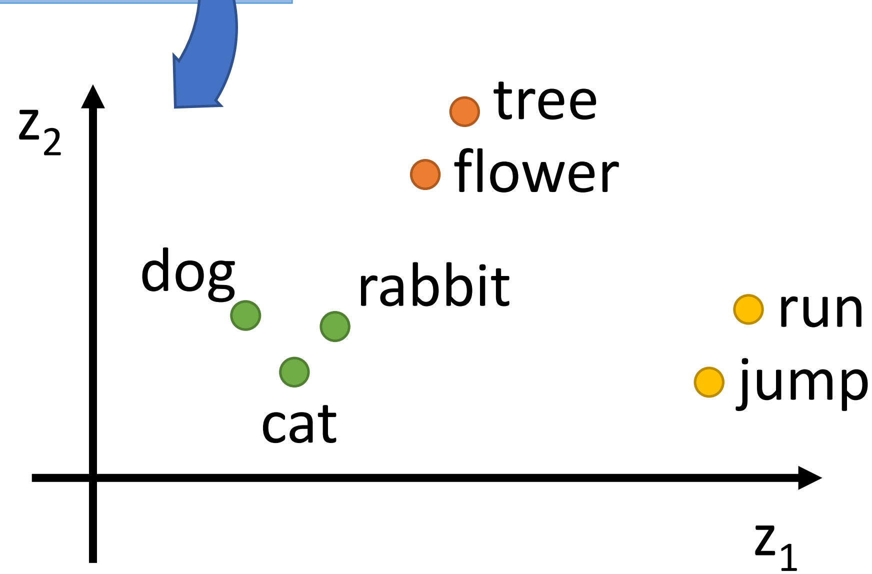



# Prediction-Based






- Take out the input of the neurons in the first layer
- Use it to represent a word w
- Word vector, word embedding feature: V(w)

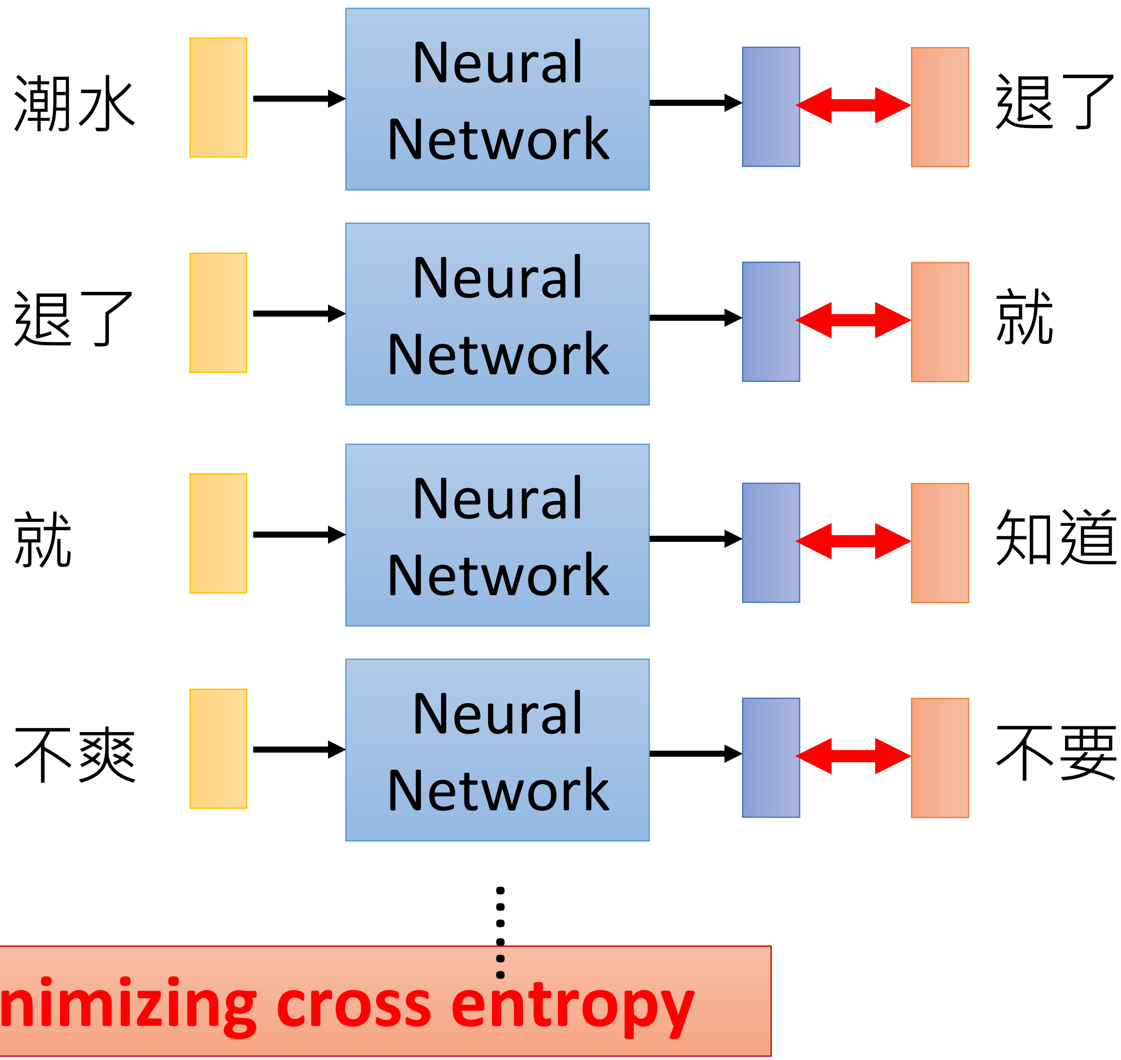


# Prediction-Based



Collect data:

潮水 退了 就 知道 …
不爽 不要 買 …
公道價 八萬 一 …
………

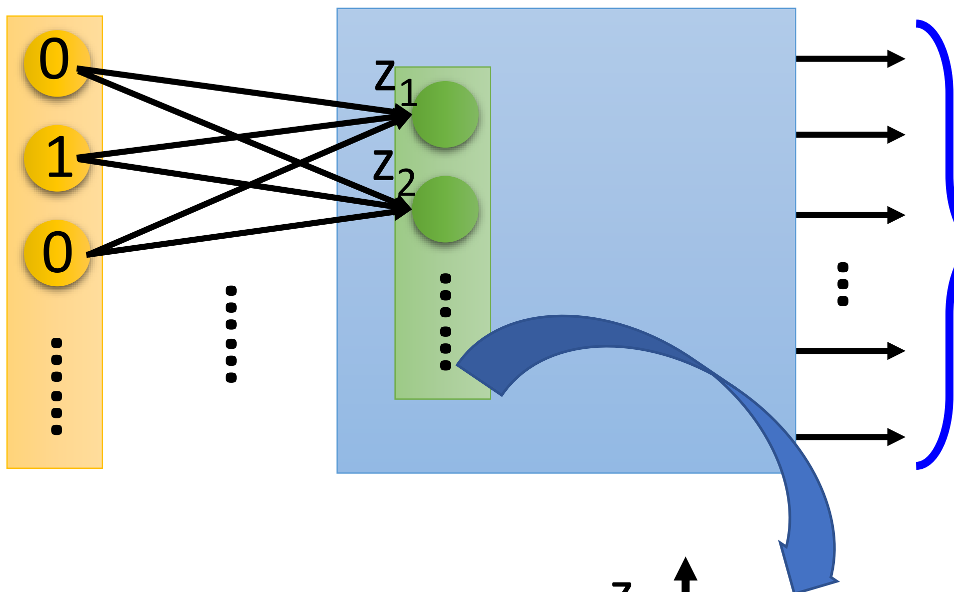


# Prediction-Based



You shall know a word by the company it keeps

蔡英文
or
馬英九




The probability for each word as the next word $w_i$

"宣誓就職" should have large probability

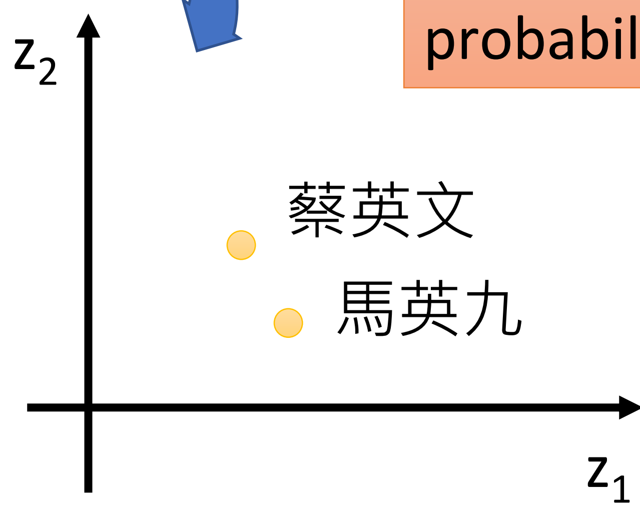
Training text:

…… 蔡英文 宣誓就職 ……

$w_{i-1}$ $w_i$

…… 馬英九 宣誓就職 ……

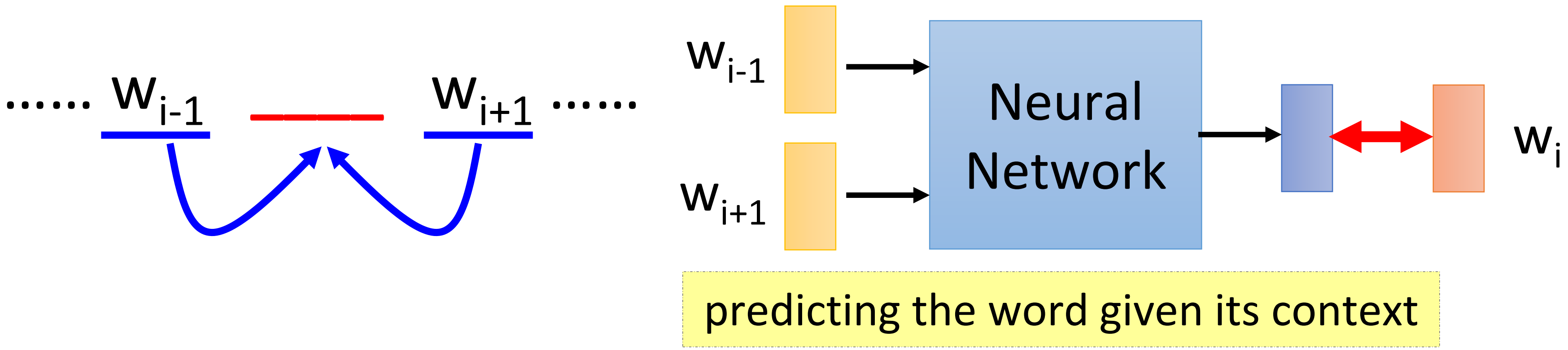
$w_{i-1}$ $w_i$



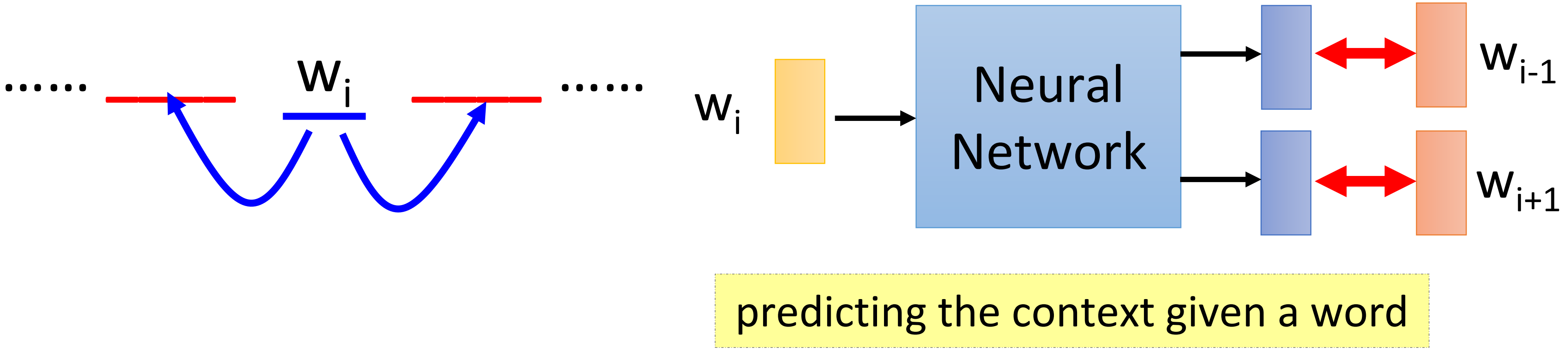
# Various Architectures



- Continuous bag of word (CBOW) model




- Skip-gram

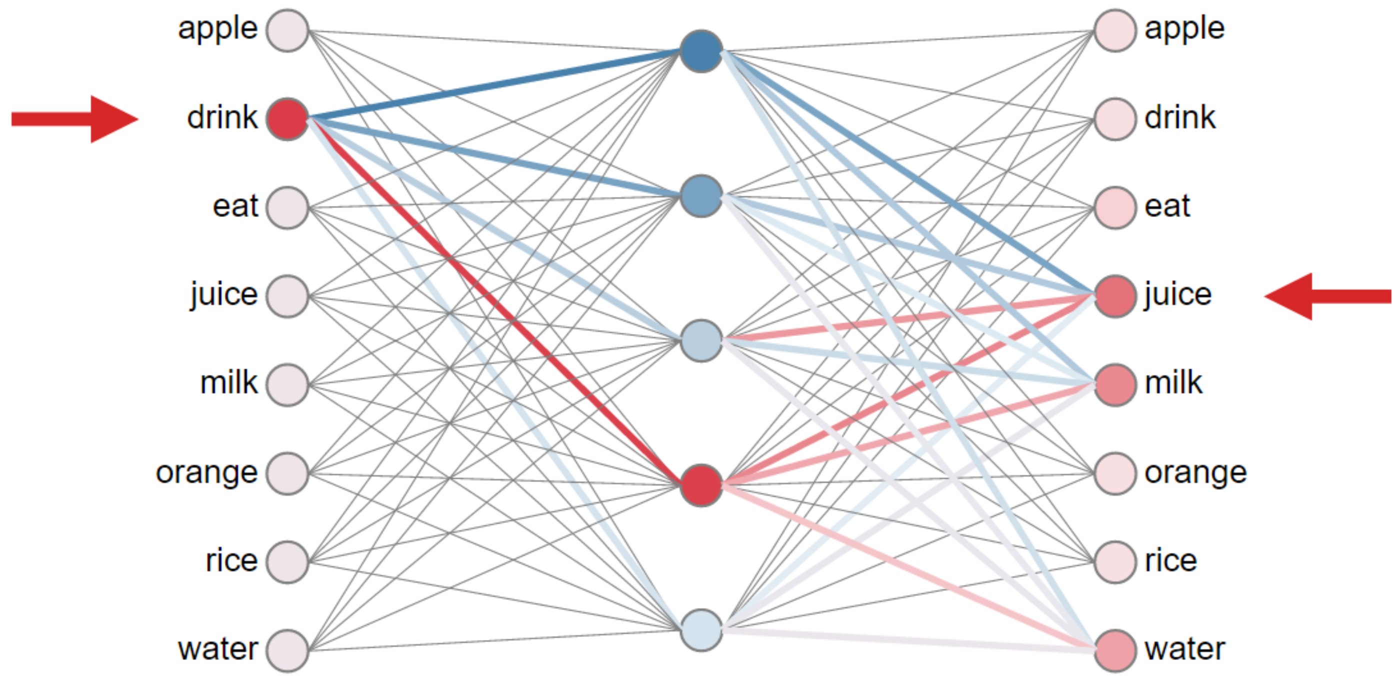


# Word2Vec LM



- Goal: predicting the next words given the proceeding contexts

$$p(w_{t+1} \mid w_t)$$



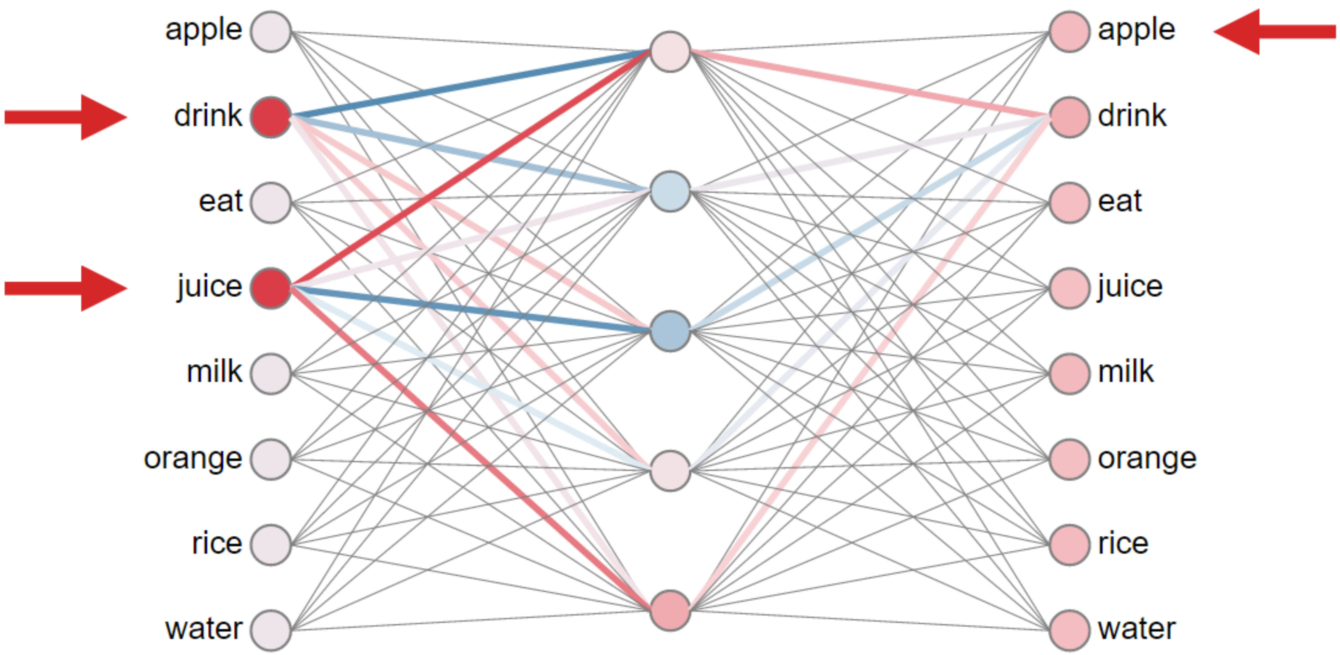

https://ronxin.github.io/wevi/

# Word2Vec CBOW



- Goal: predicting the target word given the surrounding words

$$p(w_t \mid w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots, w_{t+m})$$

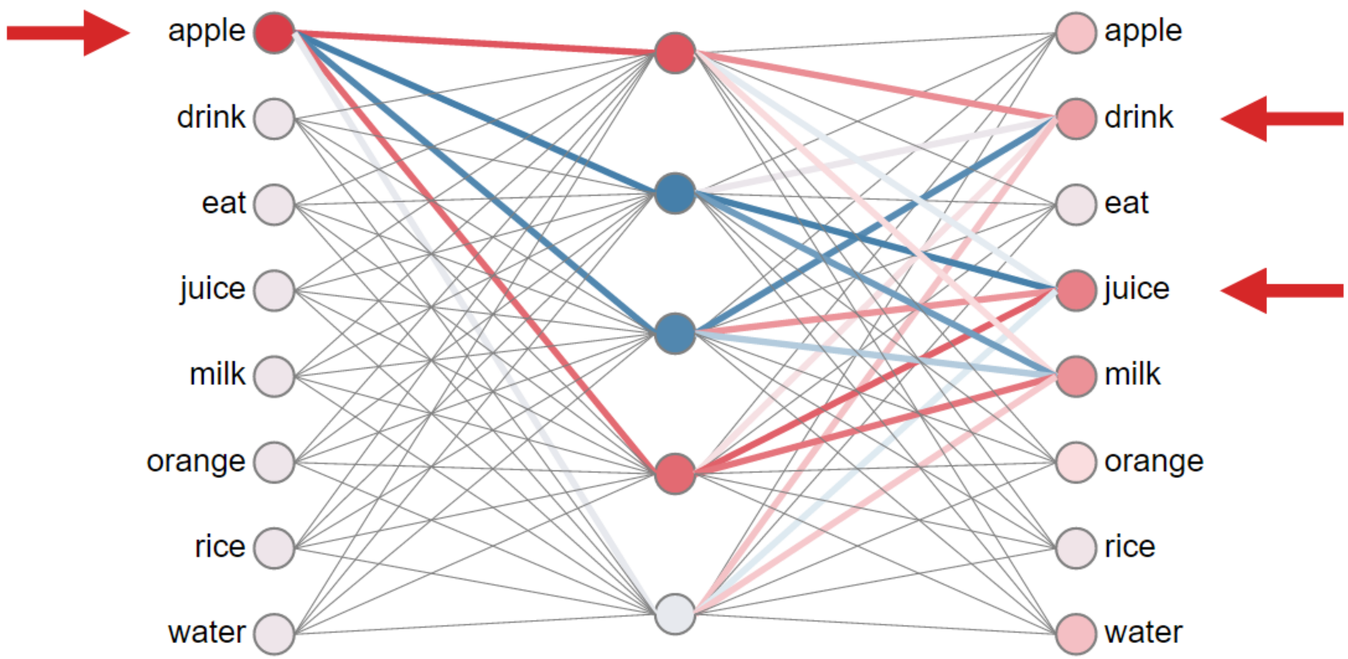



https://ronxin.github.io/wevi/

# Word2Vec Skip-Gram



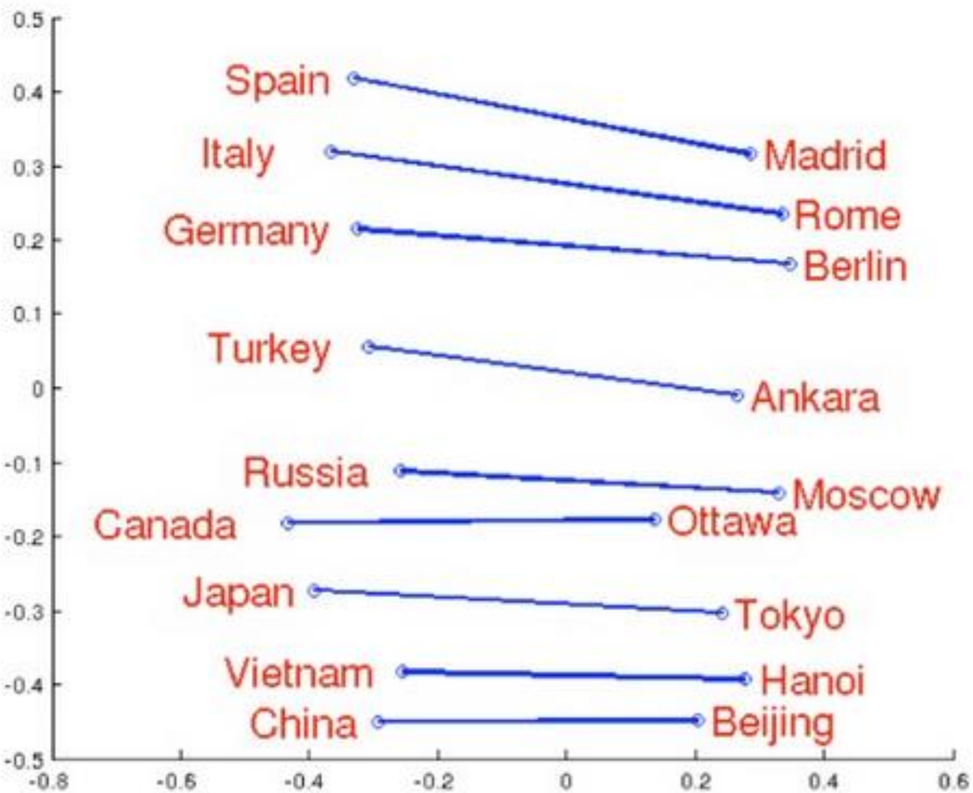
- Skip-gram training data:
  apple|drink^juice,orange|eat^apple,rice|drink^juice,juice|drink^milk,milk|drink^rice,water|drink^milk,juice|orange^apple,juice|apple^drink,milk|rice^drink,drink|milk^water,drink|water^juice,drink|juice^water

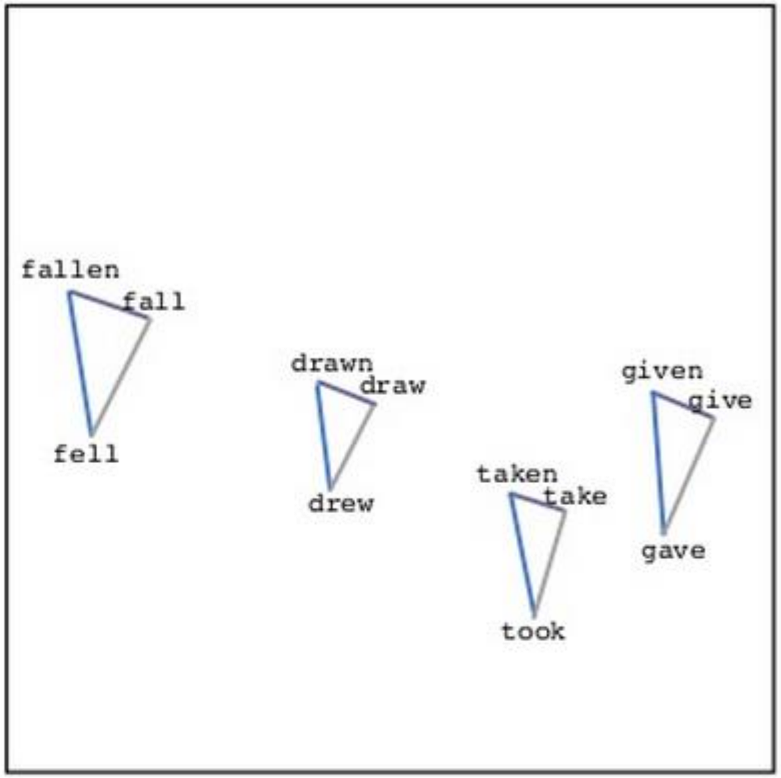



https://ronxin.github.io/wevi/

# Word Embedding






http://www.slideshare.net/hustwj/cikm-keynotenov2014

# Word Embedding



- Characteristics

$$V(hotter) - V(hot) \approx V(bigger) - V(big)$$
$$V(Rome) - V(Italy) \approx V(Berlin) - V(Germany)$$
$$V(king) - V(queen) \approx V(uncle) - V(aunt)$$

- Solving analogies

Rome : Italy = Berlin : ?

Compute $\underline{V(Berlin) - V(Rome) + V(Italy)}$

Find the word w with the closest V(w)

$$V(Germany) \approx V(Berlin) - V(Rome) + V(Italy)$$

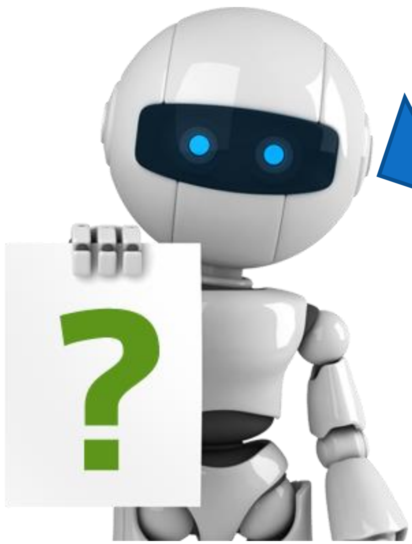
# Outline



- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Creation



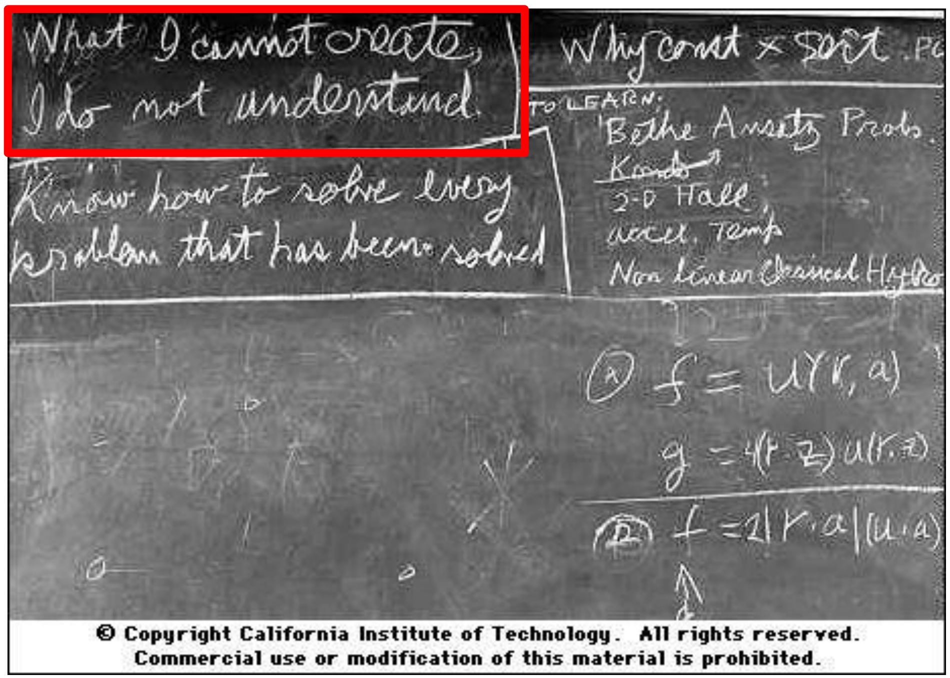


Draw something!

# Creation



- Generative Models
  - https://openai.com/blog/generative-models/




What I cannot create,
I do not understand.

**Richard Feynman**

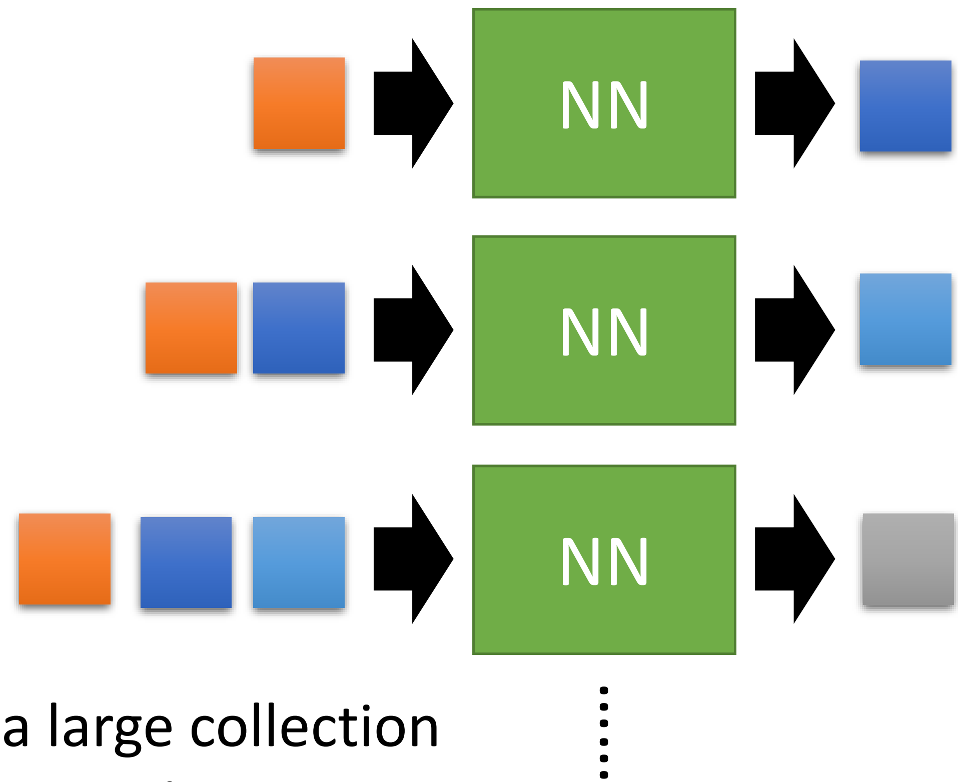
https://www.quora.com/What-did-Richard-Feynman-mean-when-he-said-What-I-cannot-create-I-do-not-understand

# PixelRNN



- To create an image, generating a pixel each time

E.g. 3 x 3 images

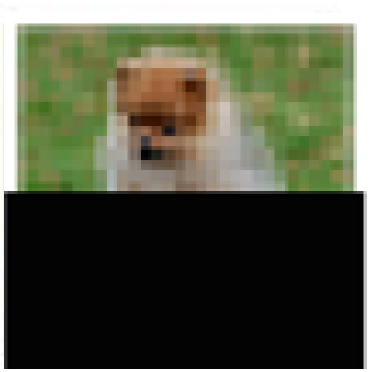



Can be trained just with a large collection of images without any annotation

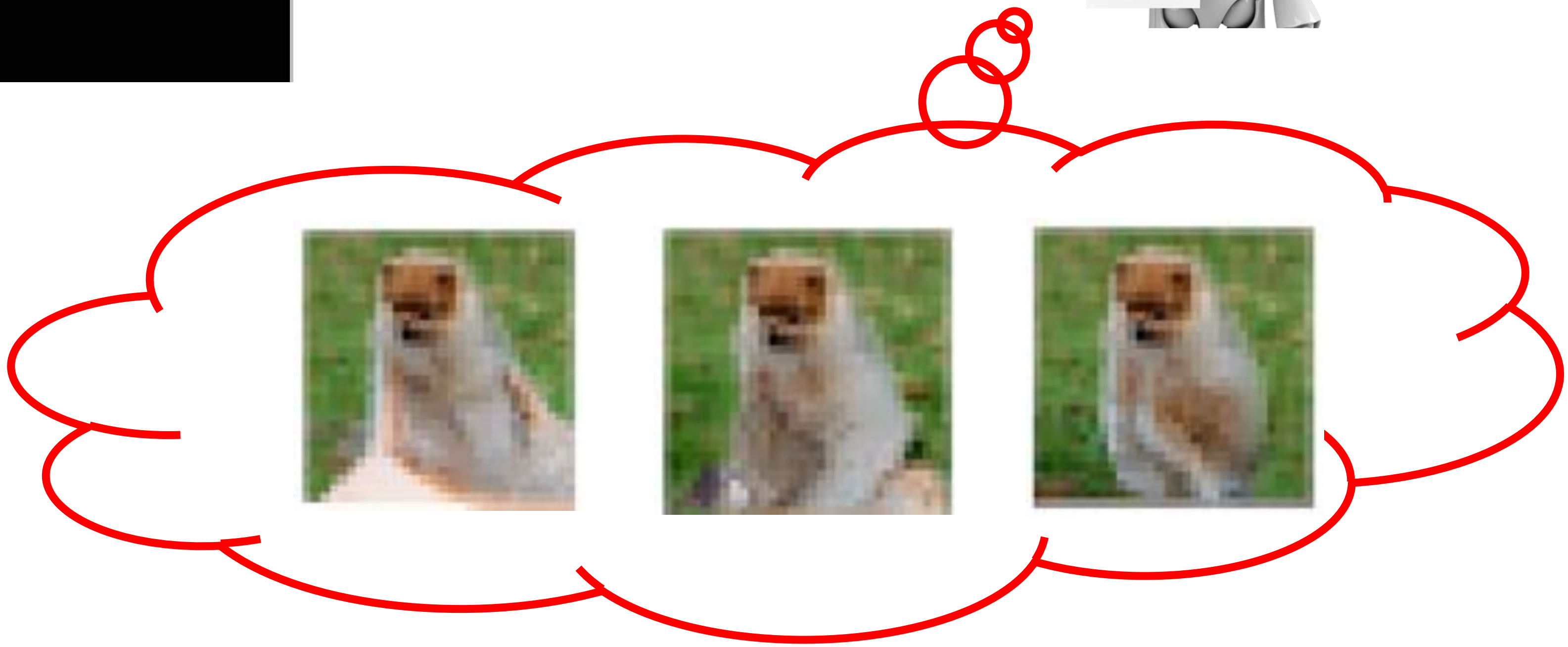
Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, Pixel Recurrent Neural Networks, arXiv preprint, 2016

# PixelRNN





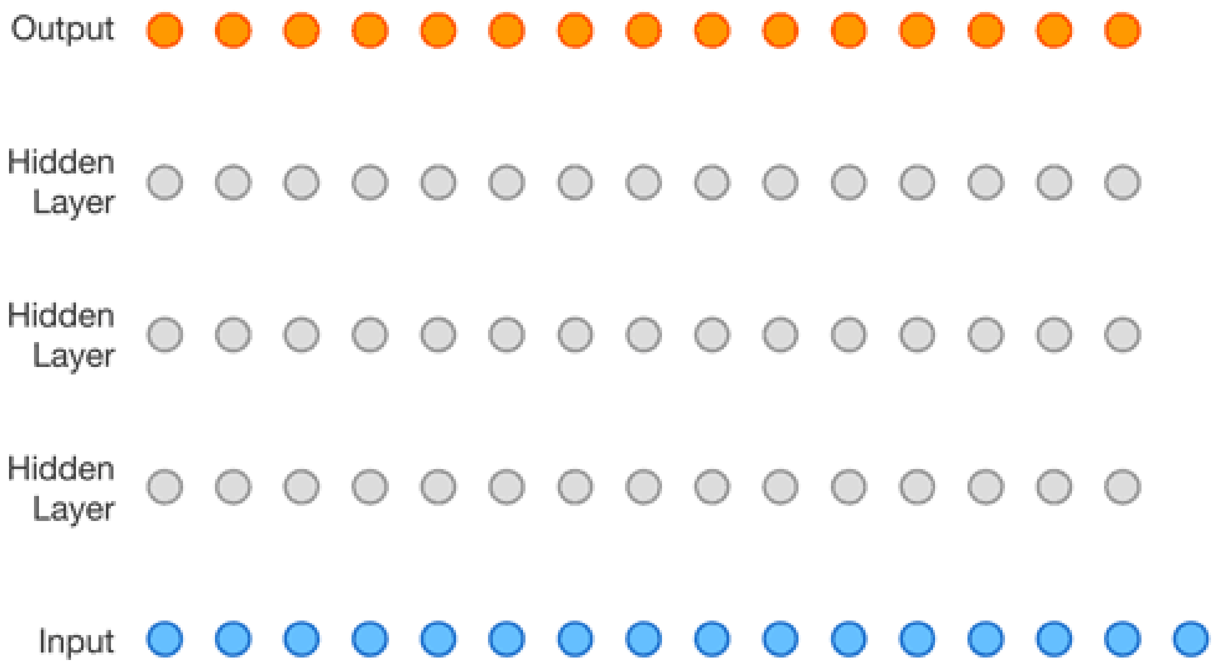
Real
World



Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, Pixel Recurrent Neural Networks, arXiv preprint, 2016

# PixelRNN – beyond Image






Audio: Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, WaveNet: A Generative Model for Raw Audio, arXiv preprint, 2016

Video: Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, Koray Kavukcuoglu, Video Pixel Networks , arXiv preprint, 2016

# Generative Adversarial Network (GAN)



## What are some recent and potentially upcoming breakthroughs in unsupervised learning?

**Yann LeCun**, Director of AI Research at Facebook and Professor at NYU

Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I've listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

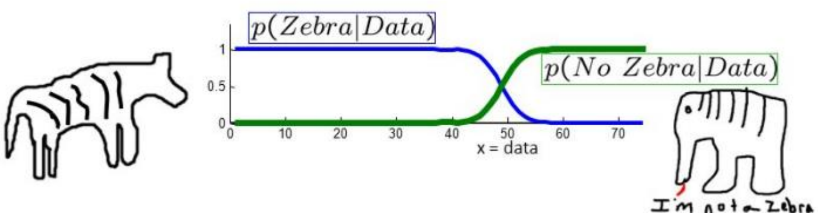
What's missing at the moment is a good understanding of it so we can make it work reliably. It's very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn't true).

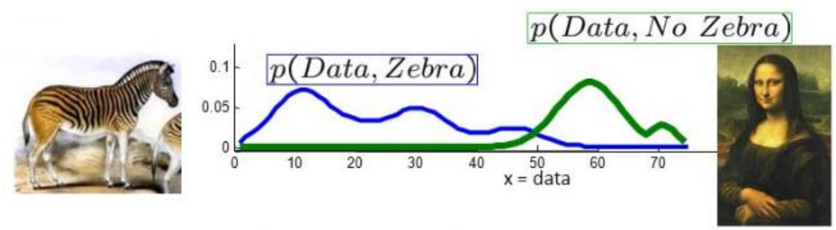
Ref: Generative Adversarial Networks, http://arxiv.org/abs/1406.2661

# Discriminative v.s. Generative Models



- Discriminative
  - learns a function that maps the input data (*x*) to some desired output class label (*y*)
    - directly learn the conditional distribution *P(y|x)*
- Generative
  - tries to learn the joint probability of the input data and labels simultaneously, i.e. *P(x,y)*
    - can be converted to *P(y|x)* for classification via Bayes rule




Advantage: generative models have the potential to <u>understand and explain the underlying structure</u> of the input data even when there are no labels

# 擬態的演化



棕色

葉脈

蝴蝶不是棕色

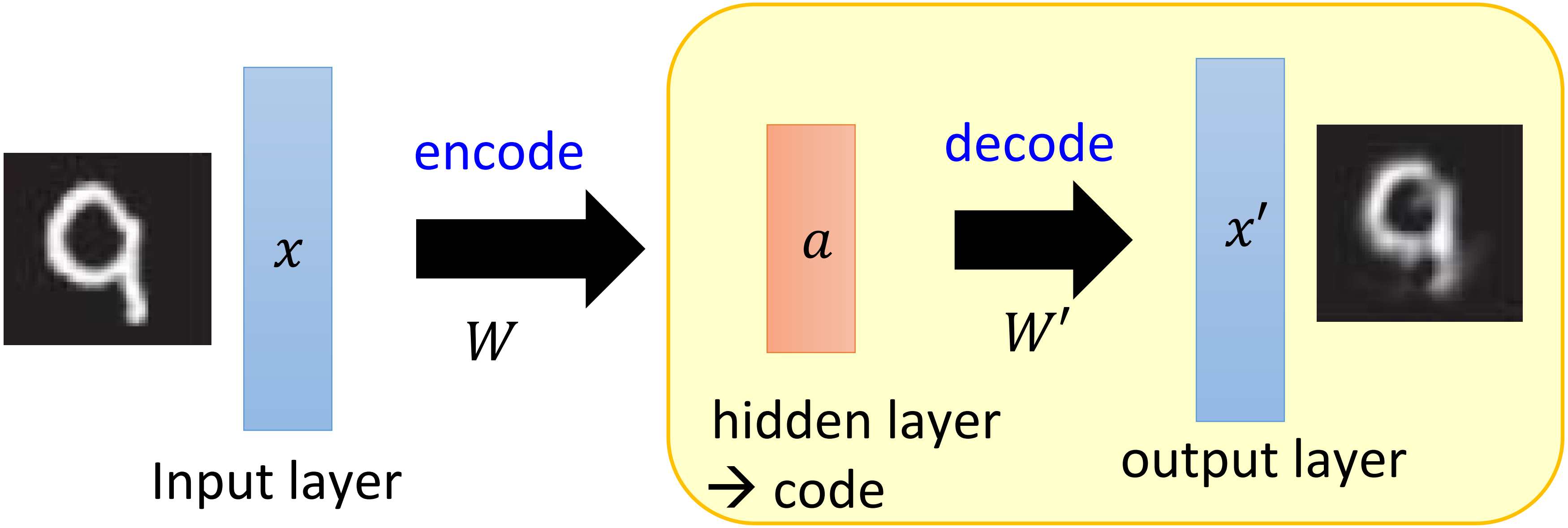
蝴蝶沒有葉脈

........

# Generator



- Decoder from autoencoder as generator

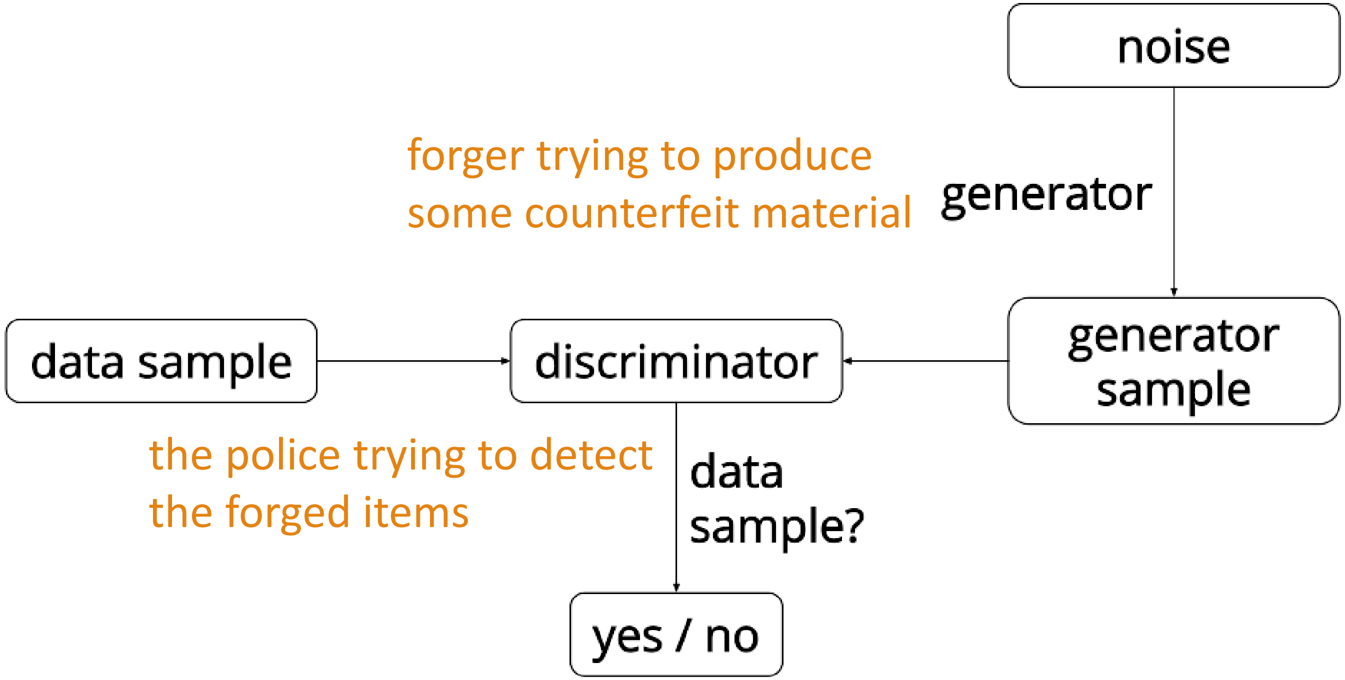



The generator is to generate the data from the code

# Generative Adversarial Networks (GAN)



- Two competing neural networks: generator & discriminator

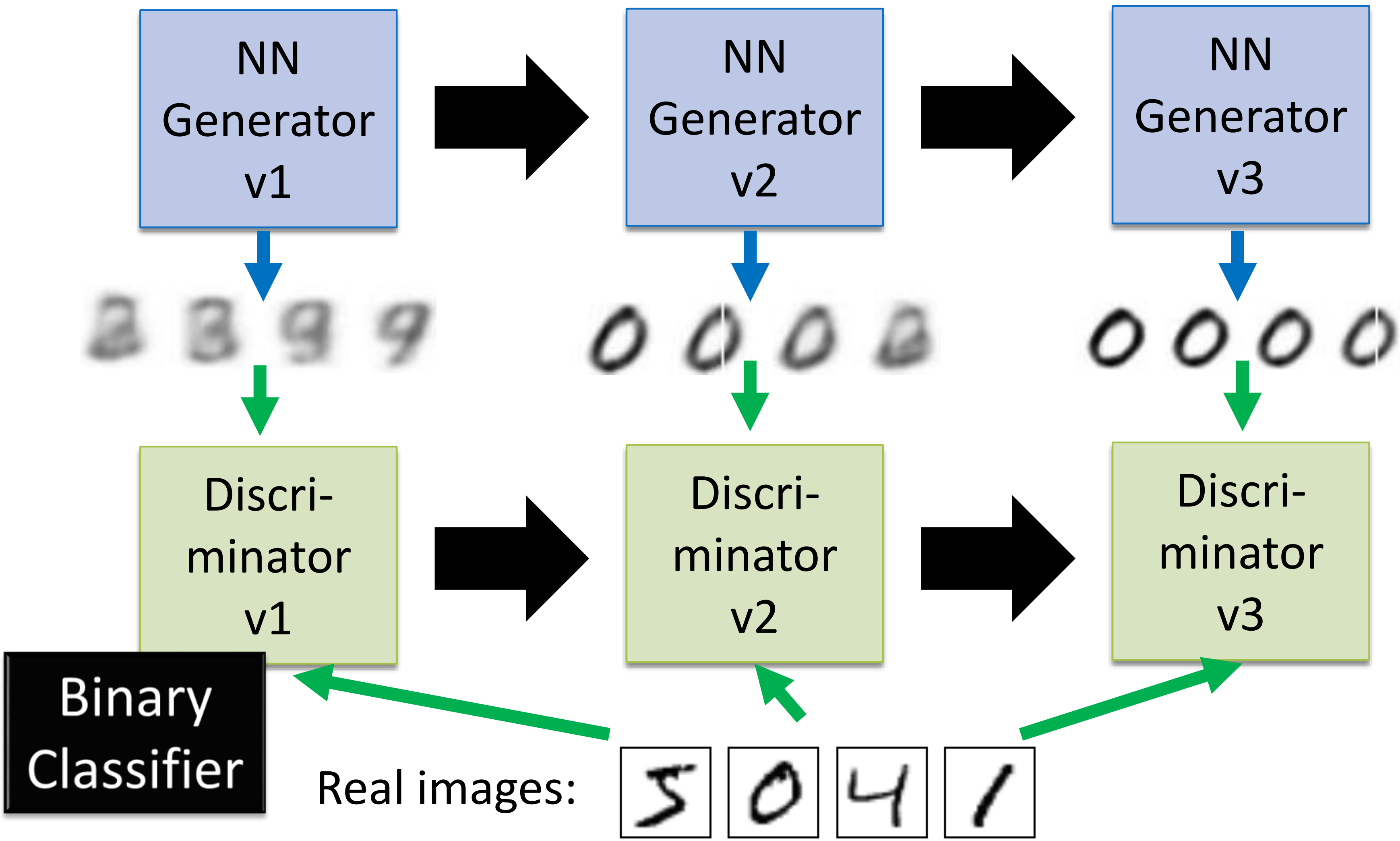



Training two networks jointly → the generator knows how to adapt its parameters in order to produce output data that can fool the discriminator

http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

# Generator Evolution

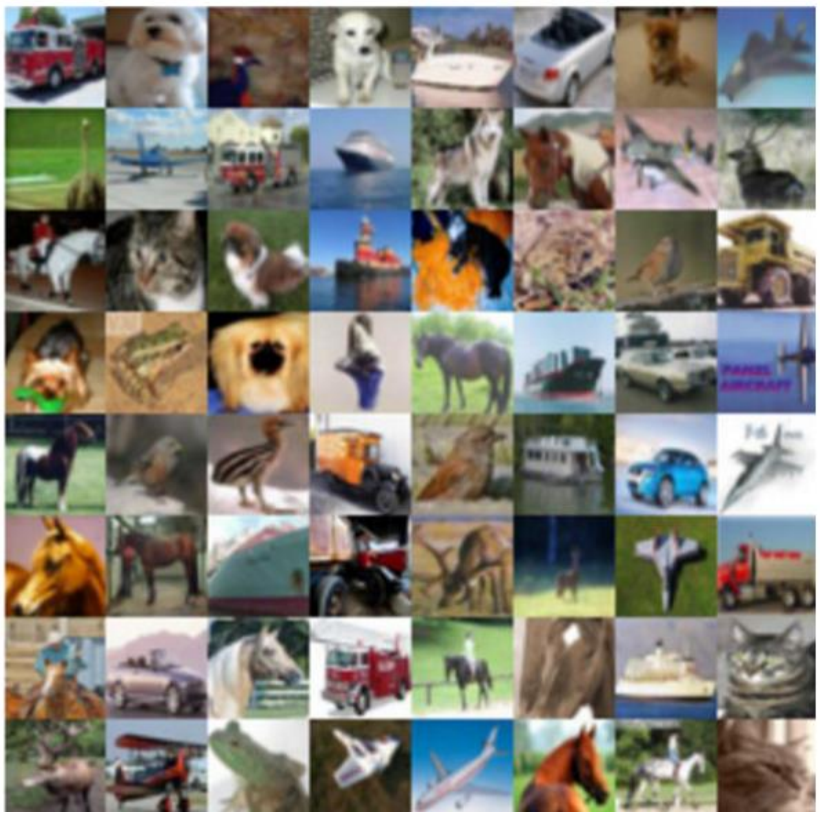


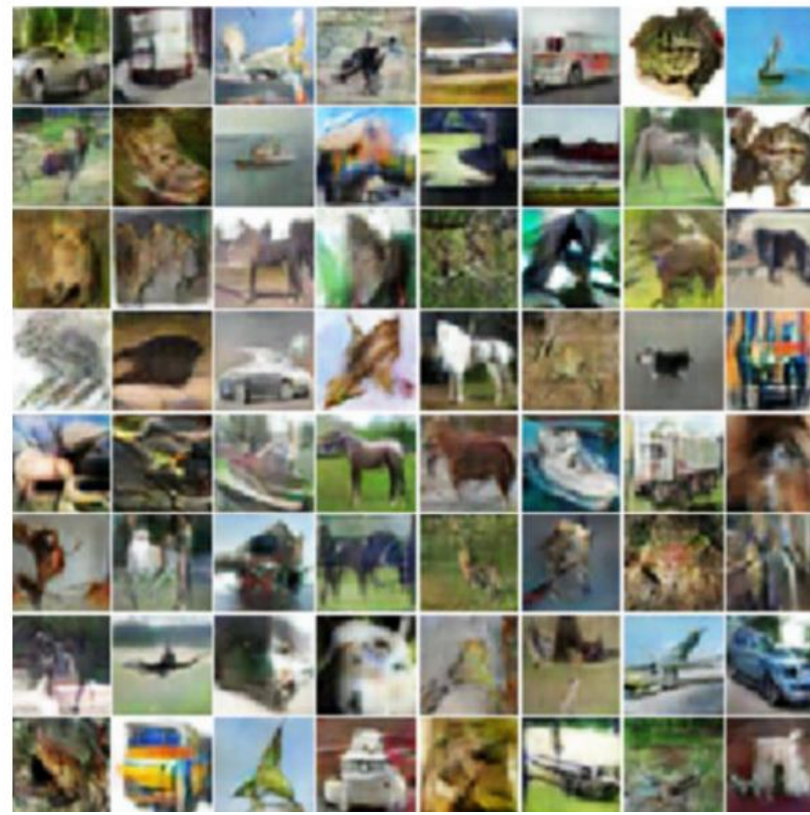


# Cifar-10



- Which one is machine-generated?





https://openai.com/blog/generative-models/

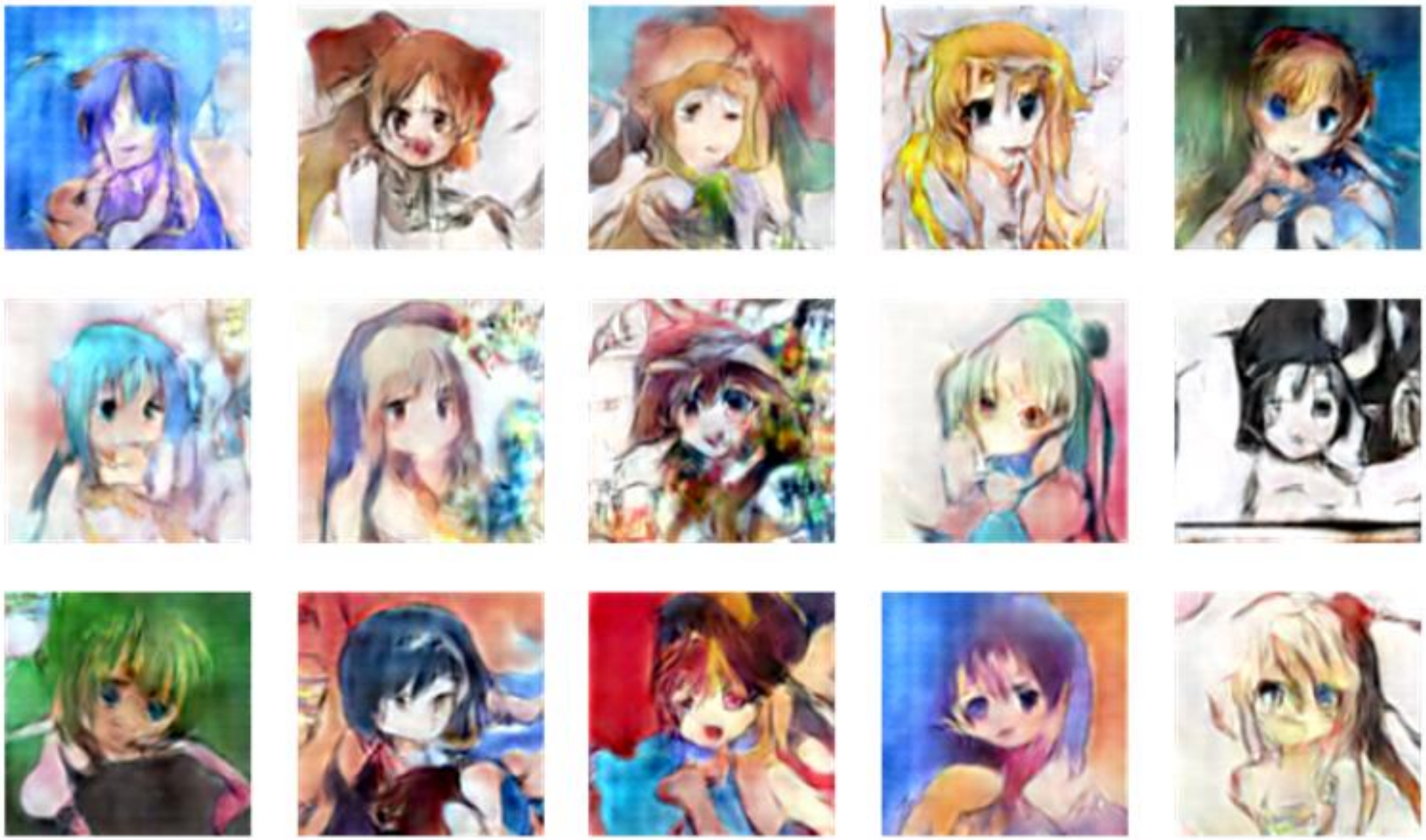
# Generated Bedrooms



Radford et al., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv:1511.06434.

# Comics Drawing





https://github.com/mattya/chainer-DCGAN

# Comics Drawing



元画像　－赤髪＋金髪　－赤目＋青目　＋制服＋セーラー　＋笑顔＋口開き　＋青背景

長髪化ベクトル

一番左のキャラクターが元画像で、
右に行くほど長髪化ベクトルを強く足している

http://qiita.com/mattya/items/e5bfe5e04b9d2f0bbd47

# Pokémon Creation



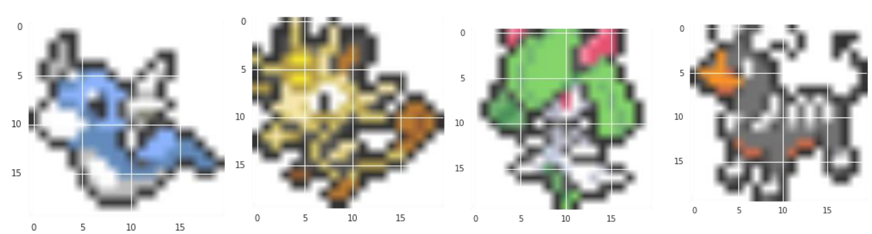
- Small images of 792 Pokémon's
  - Can machine learn to create new Pokémons?

***<u>Don't catch them! Create them!</u>***

- Source of image: http://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_base_stats_(Generation_VI)

Original image is 40 x 40

Making them into 20 x 20

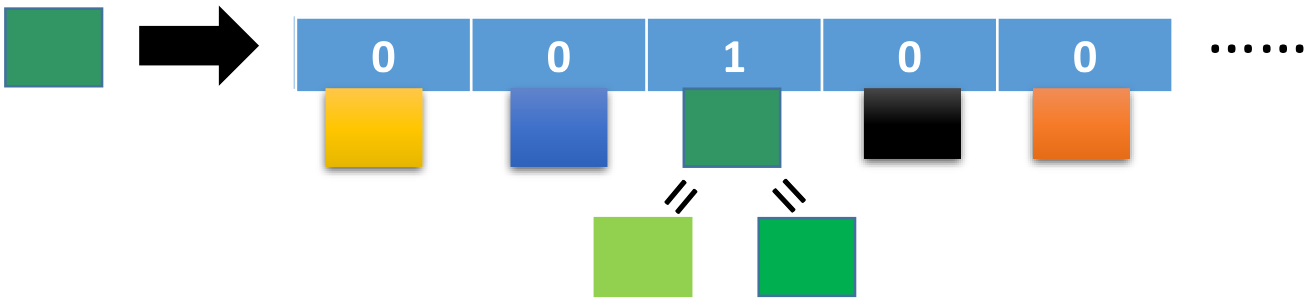


# Pokémon Creation

- Each pixel is represented by 3 numbers (corresponding to RGB)

R=50, G=150, B=100

- Each pixel is represented by a 1-of-N encoding feature

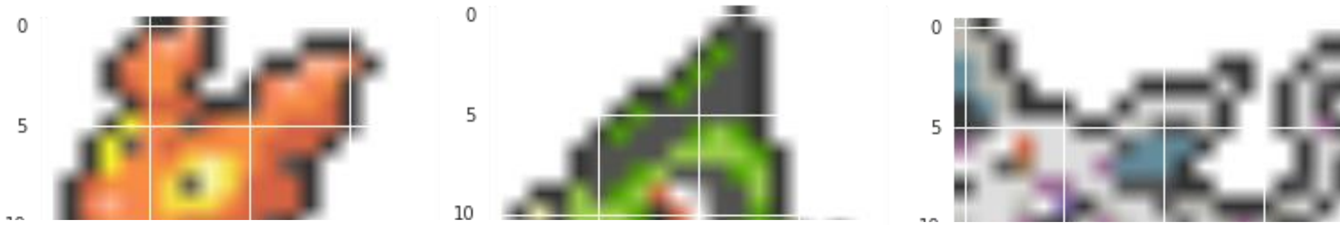



Clustering the similar color → 167 colors in total

# Pokémon Creation



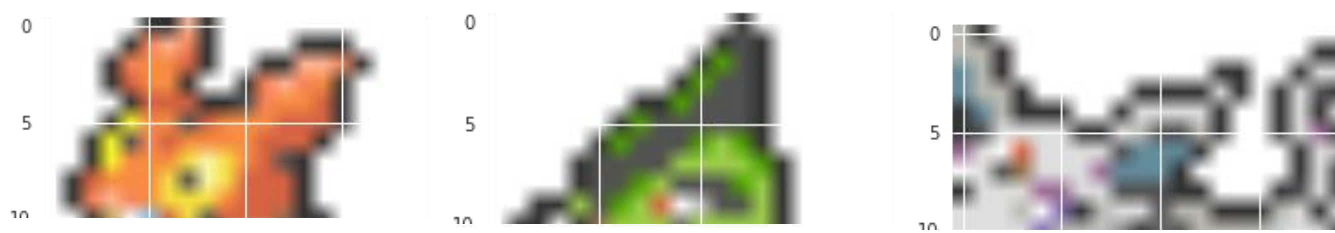
Real Pokémon



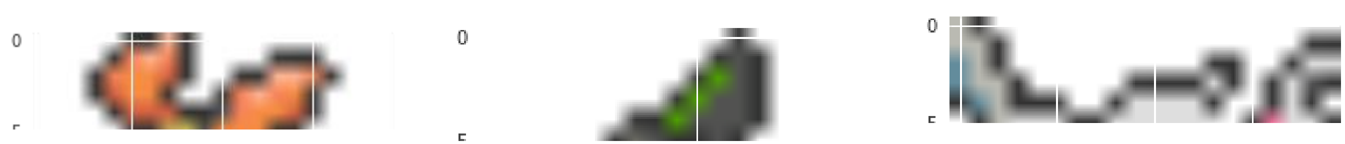
Never seen by machine!

It is difficult to evaluate generation.

Cover 50%



Cover 75%

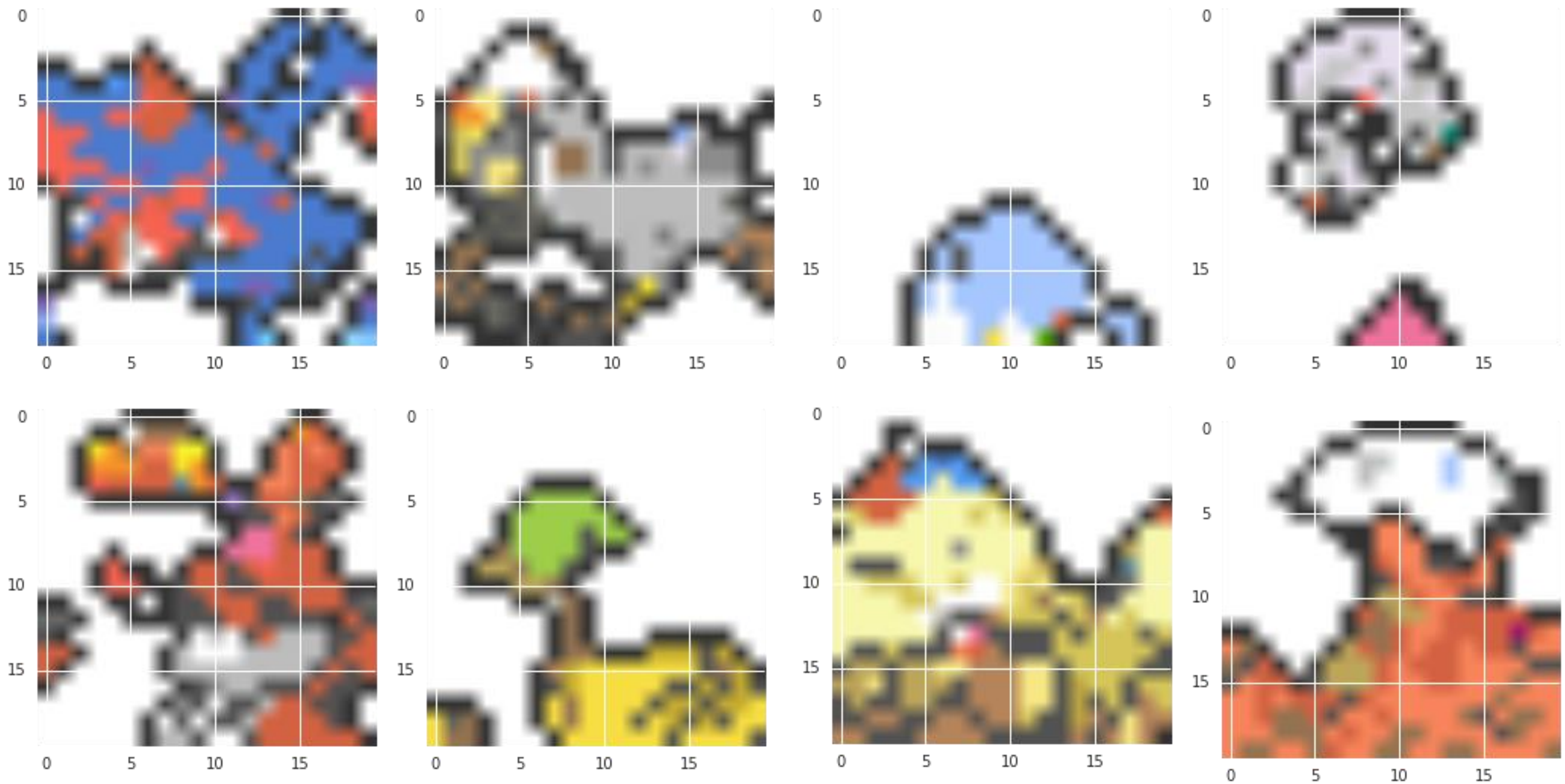


# Pokémon Creation

Drawing from scratch

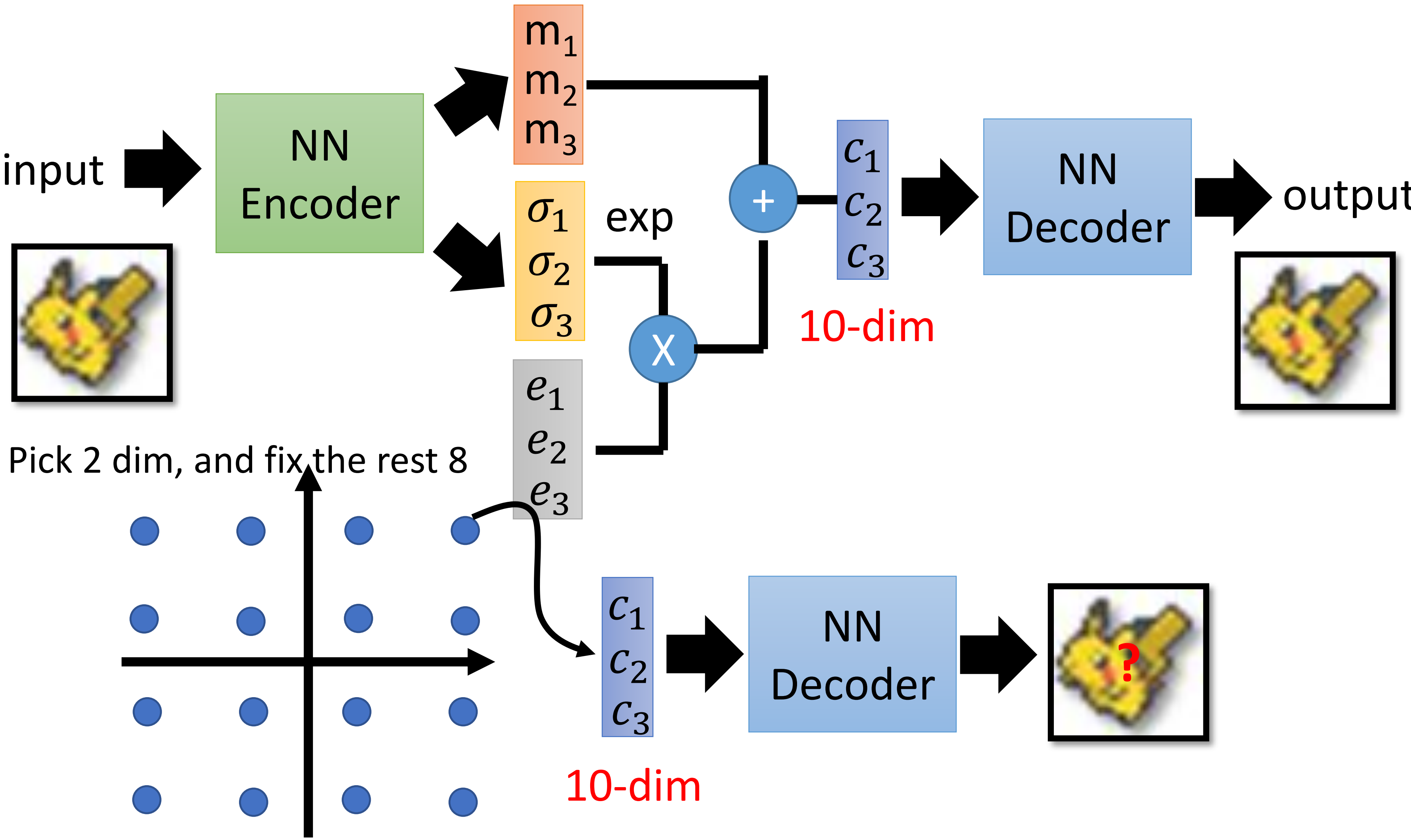
Need some randomness

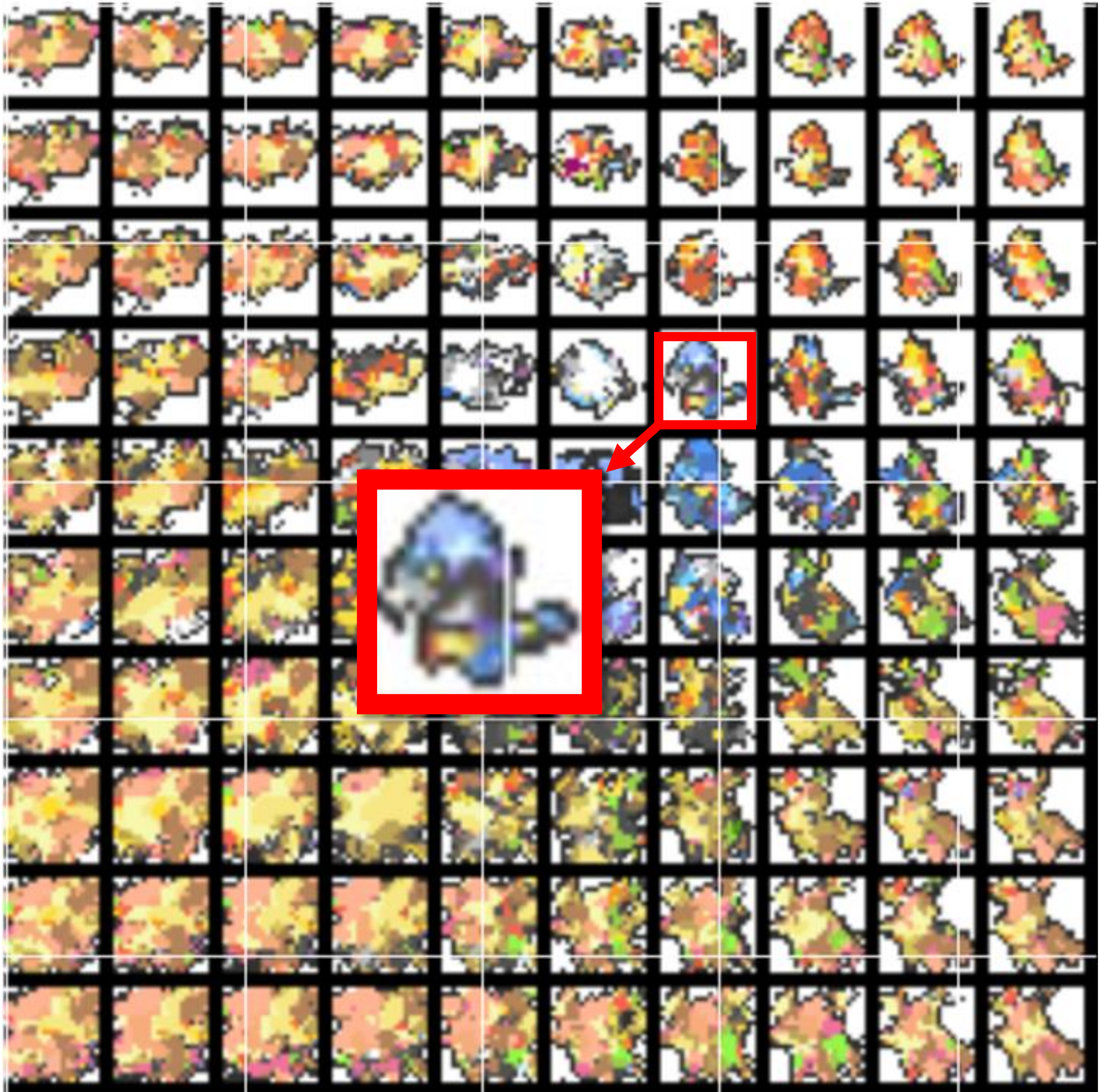




# Pokémon Creation







# Pokémon Creation - Data



- Original image (40 x 40): http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/image.rar
- Pixels (20 x 20): http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/pixel_color.txt
  - Each line corresponds to an image, and each number corresponds to a pixel
    - http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/colormap.txt

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 19 41 34 0 0 19 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 44 74 44 51 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 21 80 80 81 0 0 0 0 0 0 0 0
0 0 0 0 0 1 2 3 18 35 22 0 5 2 0 0 0 0 0 0
93 94 93 93 85 95 38 96 97 98 99 99 67 99 9
0 0 0 0 0 0 1 106 106 106 106 106 61 107 0
```

......

0
1
2

```
255 255 255
53 53 53
49 49 49
186 186 186
51 51 51
54 54 54
187 187 187
83 83 83
50 51 52
251 251 251
52 52 52
```

⋮

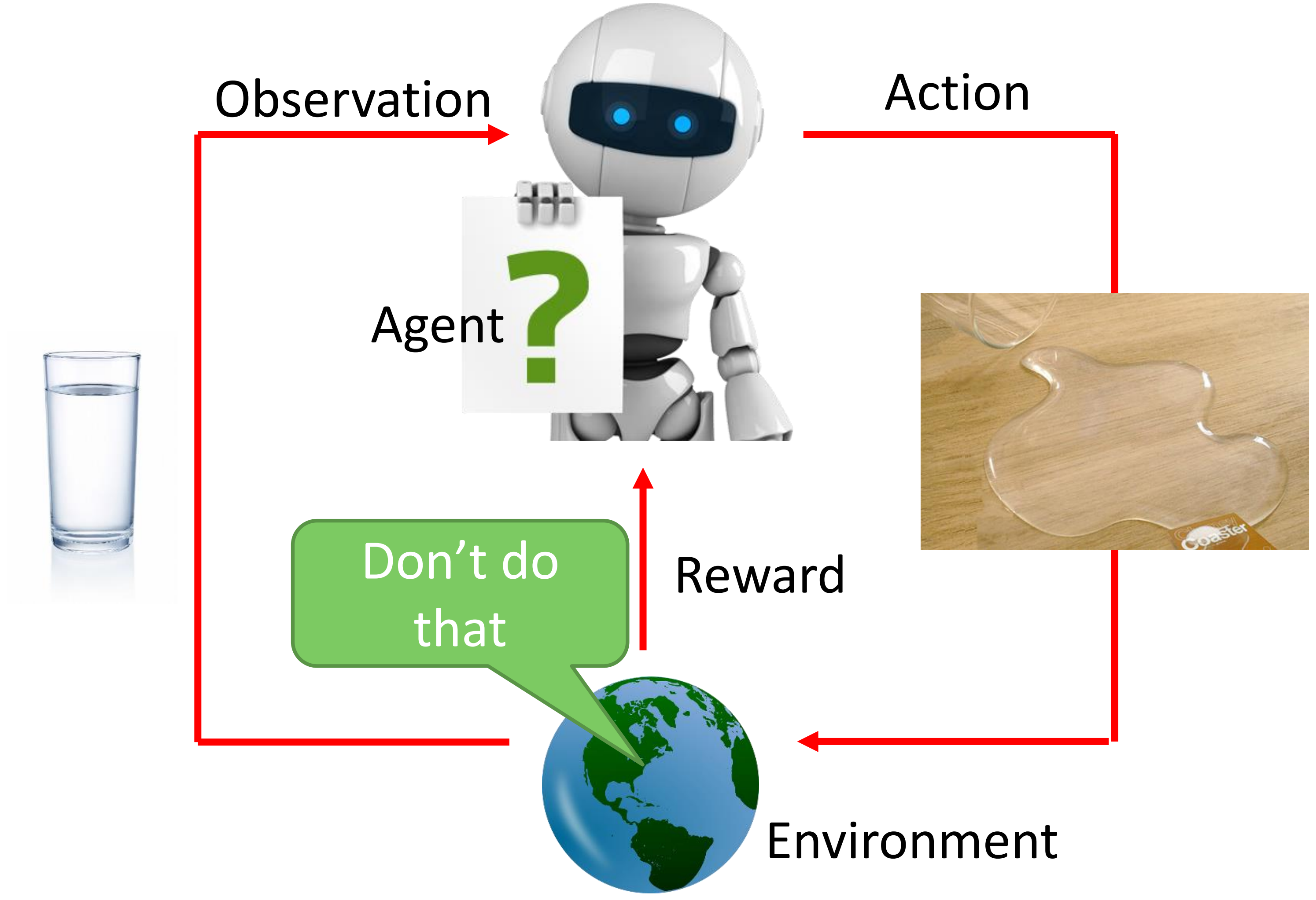
# Outline



- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

# Reinforcement Learning





# Reinforcement Learning




Agent learns to take actions to maximize expected reward.

# Supervised v.s. Reinforcement



- Supervised

Learning from teacher

"Hello" — Say "Hi"

"Bye bye" — Say "Good bye"

- Reinforcement

……. ……. ……

Hello ☺ ……

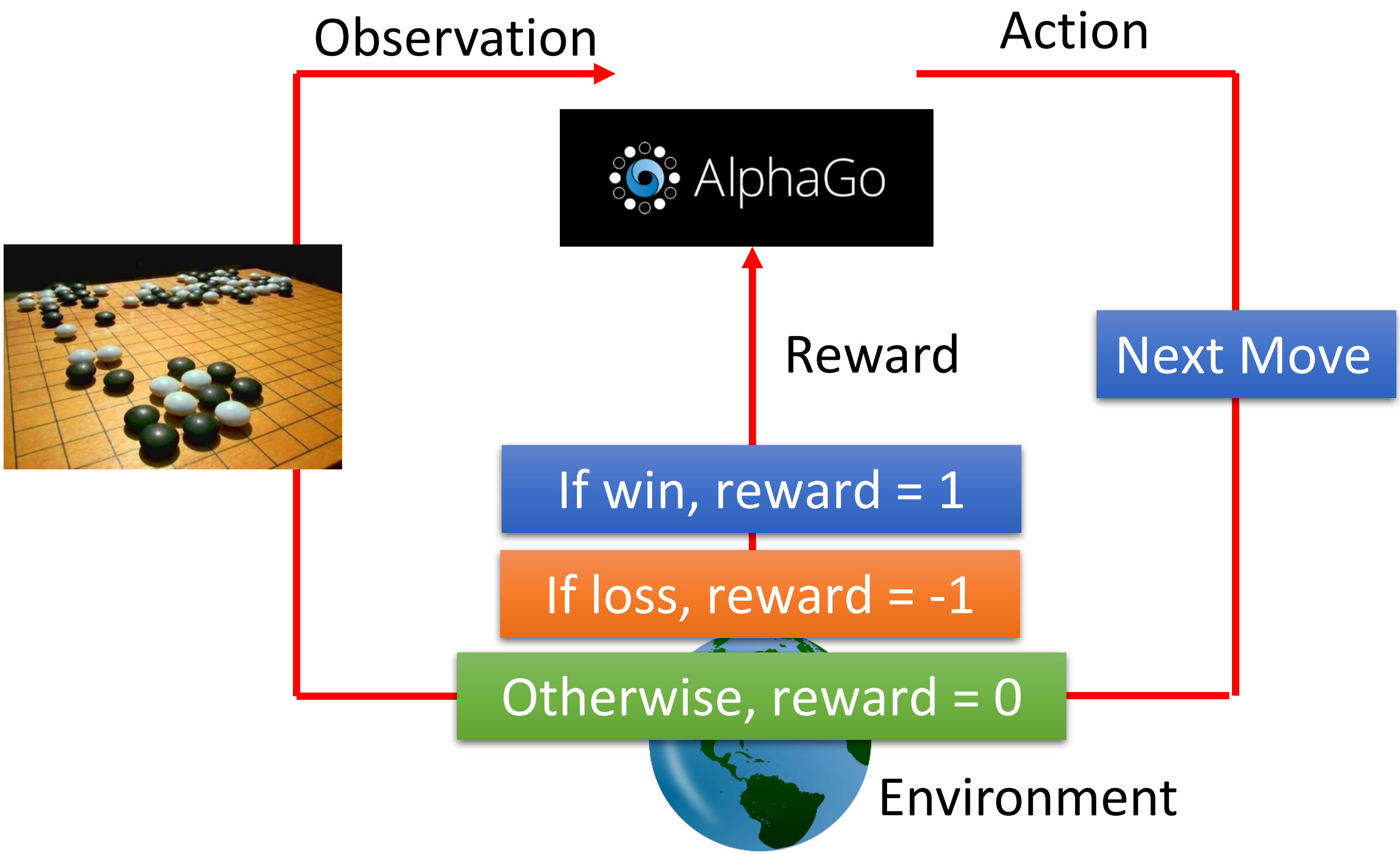
Agent Agent

Bad

Learning from critics

# Scenario of Reinforcement Learning



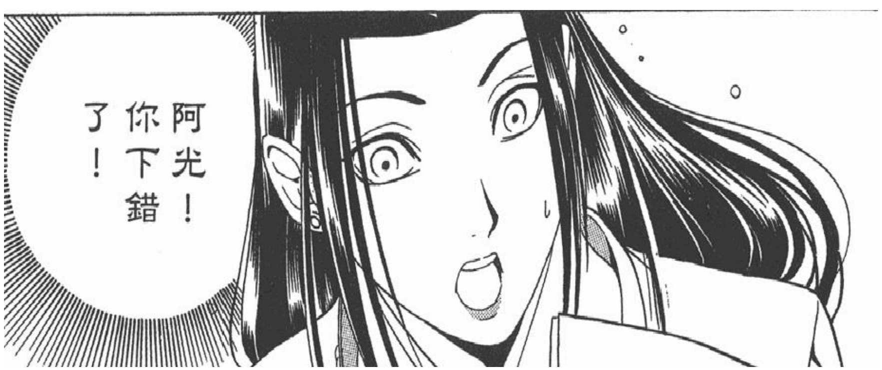



Agent learns to take actions to maximize expected reward.

# Supervised v.s. Reinforcement



- Supervised Learning
  - Training based on supervisor/label/annotation
  - Feedback is instantaneous
  - Time does not matter
- Reinforcement Learning
  - Training only based on reward signal
  - Feedback is delayed
  - Time matters
  - Agent actions affect subsequent data




AlphaGo uses supervised learning + reinforcement learning

# Reinforcement Learning



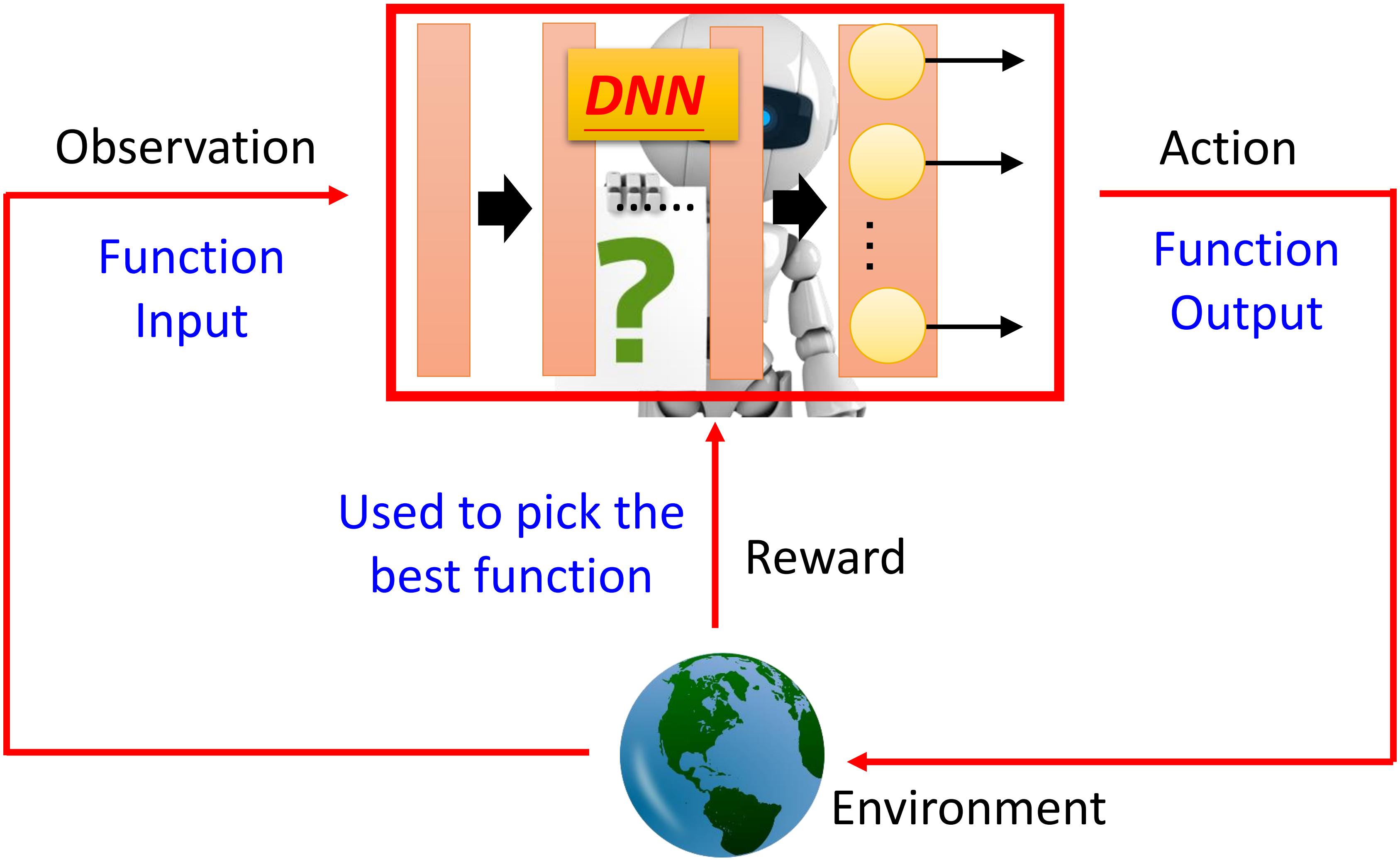
- RL is a general purpose framework for **decision making**
  - RL is for an *agent* with the capacity to *act*
  - Each *action* influences the agent's future *state*
  - Success is measured by a scalar *reward* signal
  - Goal: *select actions to maximize future reward*

# RL Difficulty



- Goal: select actions to maximize total future reward
  - Actions may have long-term consequences
  - Reward may be delayed
  - It may be better to sacrifice immediate reward to gain more long-term reward


# Deep Reinforcement Learning



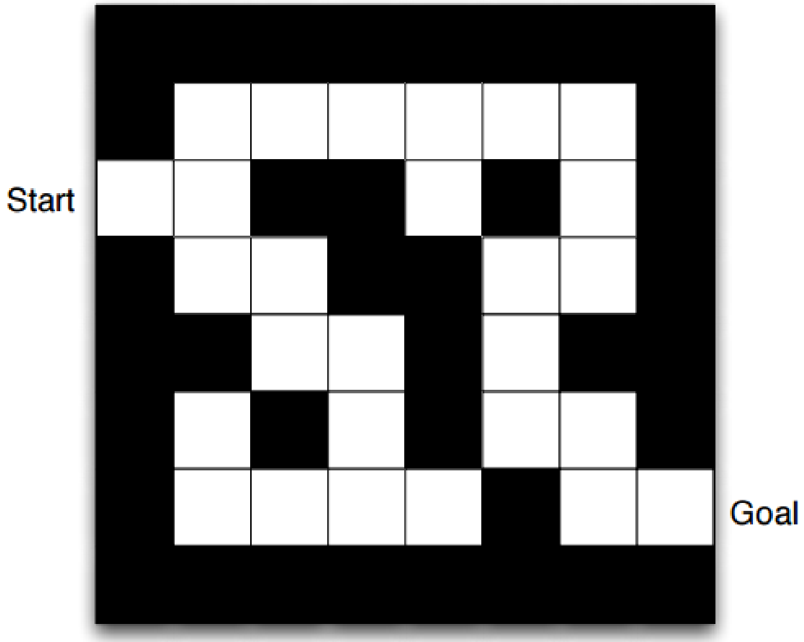
# Major Components in an RL Agent



- An RL agent may include one or more of these components
  - **Policy**: agent's behavior function
  - **Value function**: how good is each state and/or action
  - **Model**: agent's representation of the environment

# Maze Example

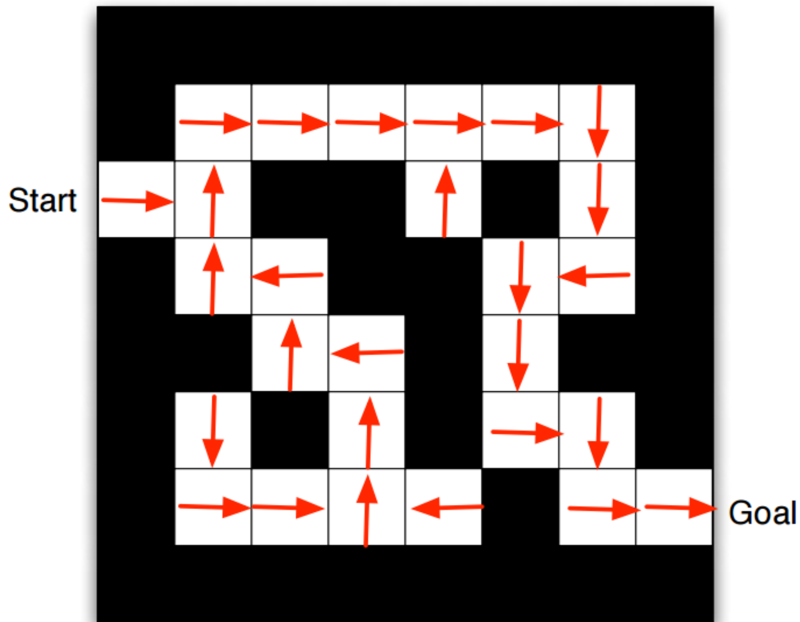





- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent’s location

# Maze Example: Policy





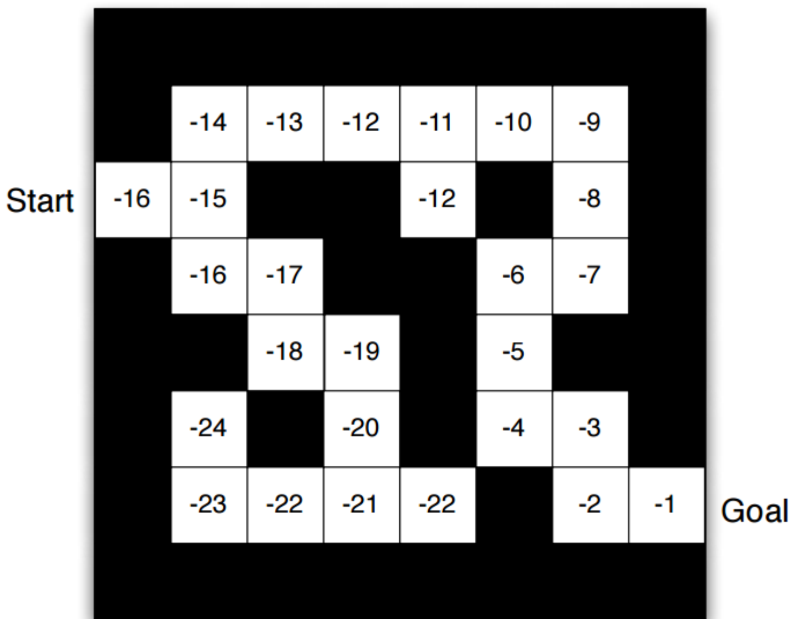

- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent's location

Arrows represent policy *π(s)* for each state *s*

# Maze Example: Value Function



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent's location




Numbers represent value $Q_\pi(s)$ of each state $s$

# E2E RL Task-Completion Bot



- Simulated User
  - Generate interactions based on a predefined fake goal
  - Automatically learn strategy by training on the simulated data

User goal: ***Two tickets for "the witch" tomorrow 9:30 PM at regal meridian 16, Seattle.***

**RULE BASED SYSTEM**
User: Can I get 2 tickets for the witch?
Agent: Which theater do you want?
User: Regal meridian 16
Agent: What time would you like to see it?
User: 9:30 pm
Agent: What date would you like to see it?
User: Tomorrow!
Agent: How many people are you?
User: Just two adults.
Agent: What is your location?
User: Seattle!
Agent: What kind of video format do you like?
User: Oh, any kind is okay.
Agent: Do you have any price constraint?
User: No, I don't care.
Agent: Great - I was able to purchase 2 tickets for you to see the witch tomorrow at regal meridian 16 theater in Seattle at 9:30 pm.
User: Thank you.
Agent: Thank you!

**REINFORCEMENT LEARNING SYSTEM**
User: Can I get 2 tickets for the witch?
Agent: What date would you like to see it?
User: Tomorrow
Agent: Which theater do you prefer?
User: Regal meridian 16
Agent: What is your location?
User: Seattle
Agent: What time would you like to see it?
User: 9:30 pm.
Agent: Great - I was able to purchase 2 tickets for you to see the witch tomorrow at regal meridian 16 theater in Seattle at 9:30 pm.
User: Thanks.
Agent: Thanks!

# More applications



- AlphaGo, Playing Video Games, Dialogue
- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI
  - http://www.bloomberg.com/news/articles/2016-07-19/google-cuts-its-giant-electricity-bill-with-deepmind-powered-ai

# Concluding Remarks



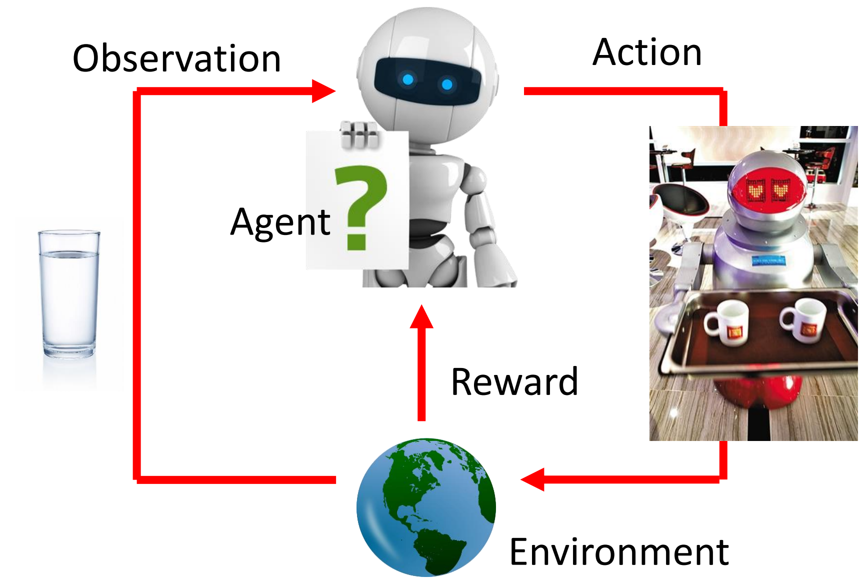
- Semi-Supervised Learning
- Transfer Learning
- Unsupervised Learning
  - 化繁為簡 Representation Learning
  - 無中生有 Generative Model
- Reinforcement Learning

Semi-Supervised Learning

Transfer Learning

Unsupervised Learning

Reinforcement Learning



# 如何成為武林高手



- 內外兼修
  - 內功充沛，恃強克弱
  - 招數精妙，以快打慢
- Machine Learning & Deep Learning 也需要內外兼修
  - 內力：運算資源
  - 招數：各種技巧
- 內力充沛,平常的招式也有可能發會巨大的威力