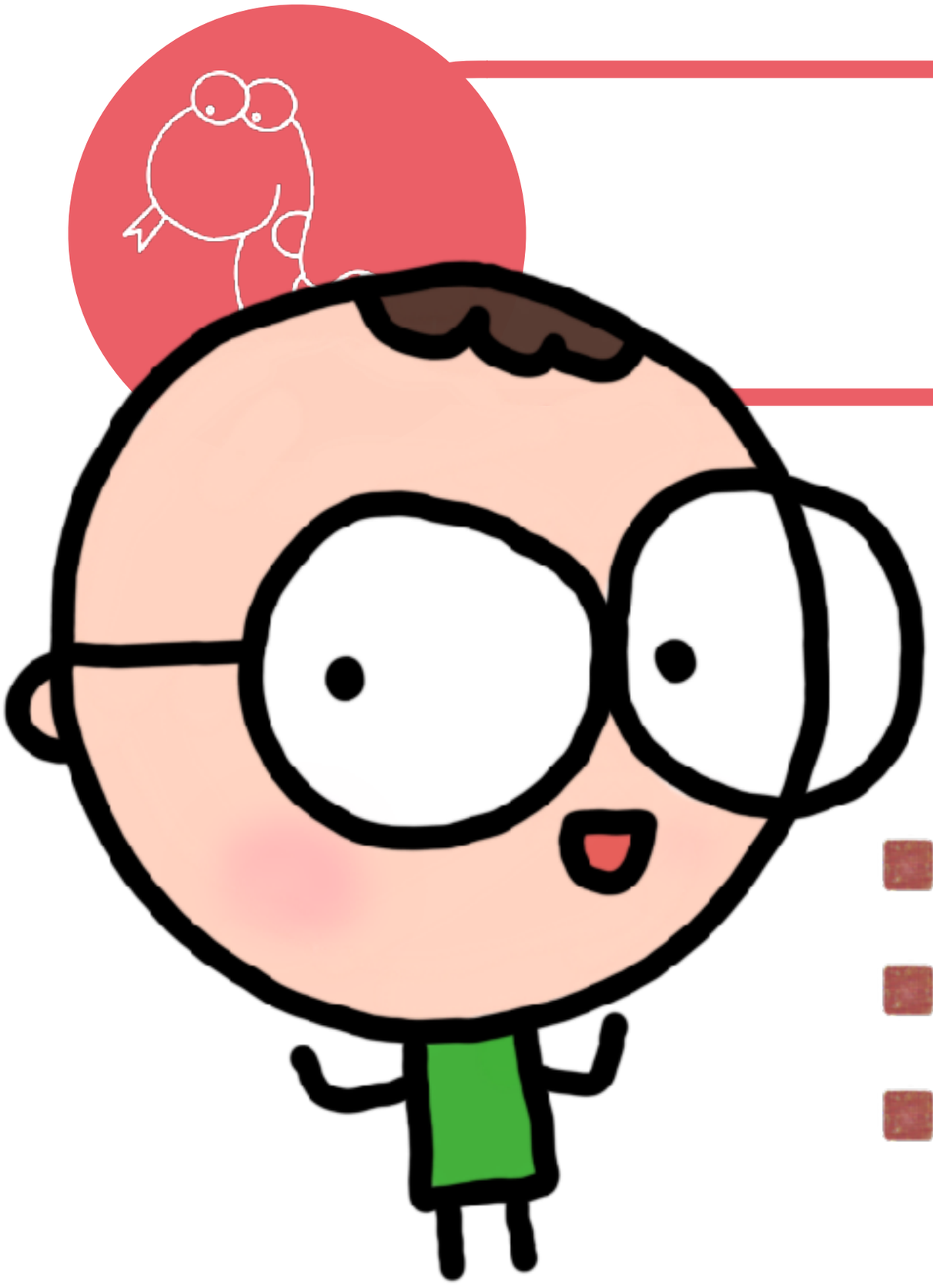
NTHU 2018

Seminar in Information Systems and Applications

Deep Learning and Design Thinking

Yen-lung Tsai
Department of Mathematical Sciences
National Chengchi University



Outline

- Asking a good question
- Learning Neural Networks
- Applying Design Thinking

1

Asking a Good Question

What we want to do is just transforming the real world problem to a function. Then we use deep learning techniques to find the function.

Functions are solution manuals

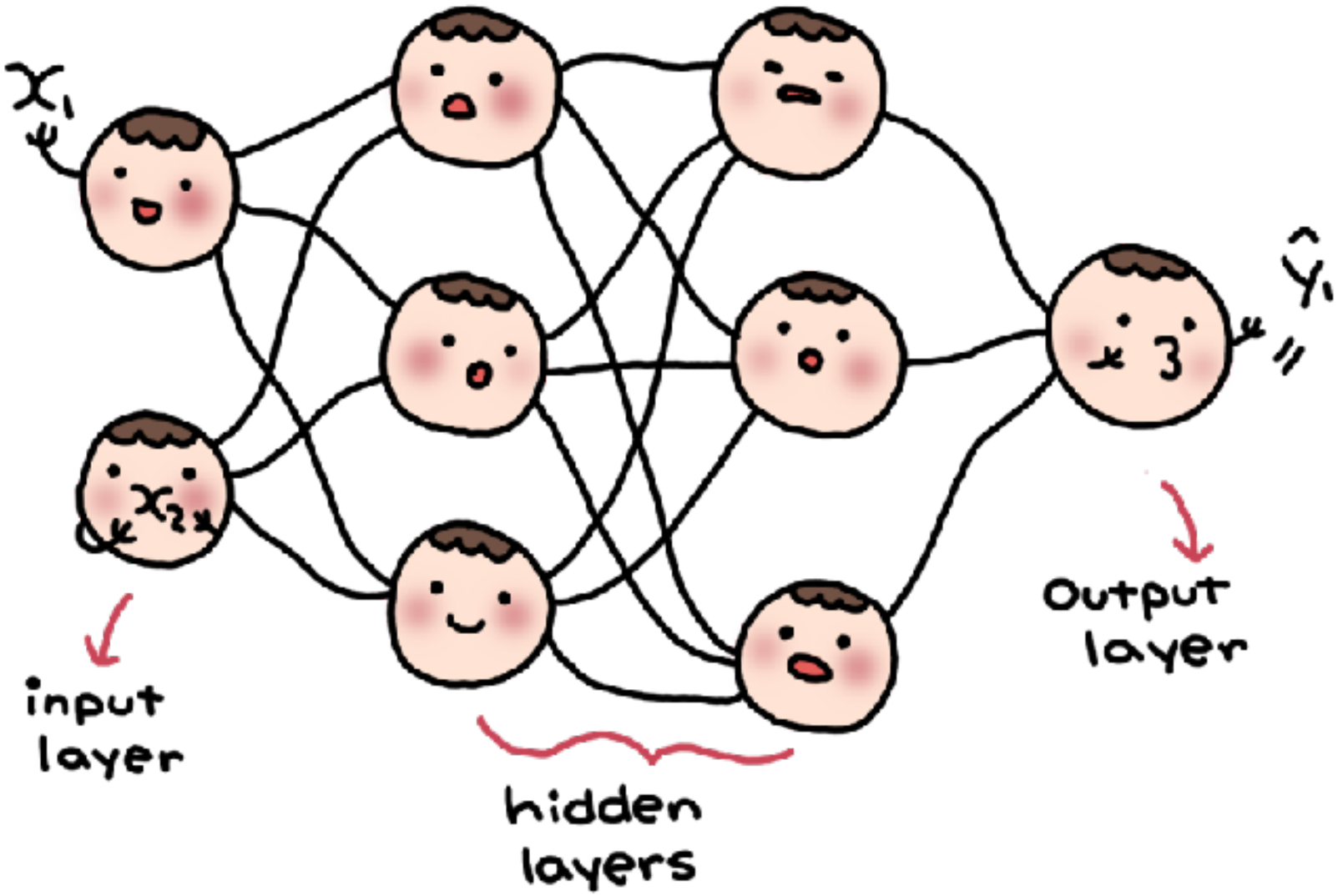
I wan to know what the animal in the picture is.

Question	Answer
	Formosan black bear
	Python
⋮	

We have partial answers

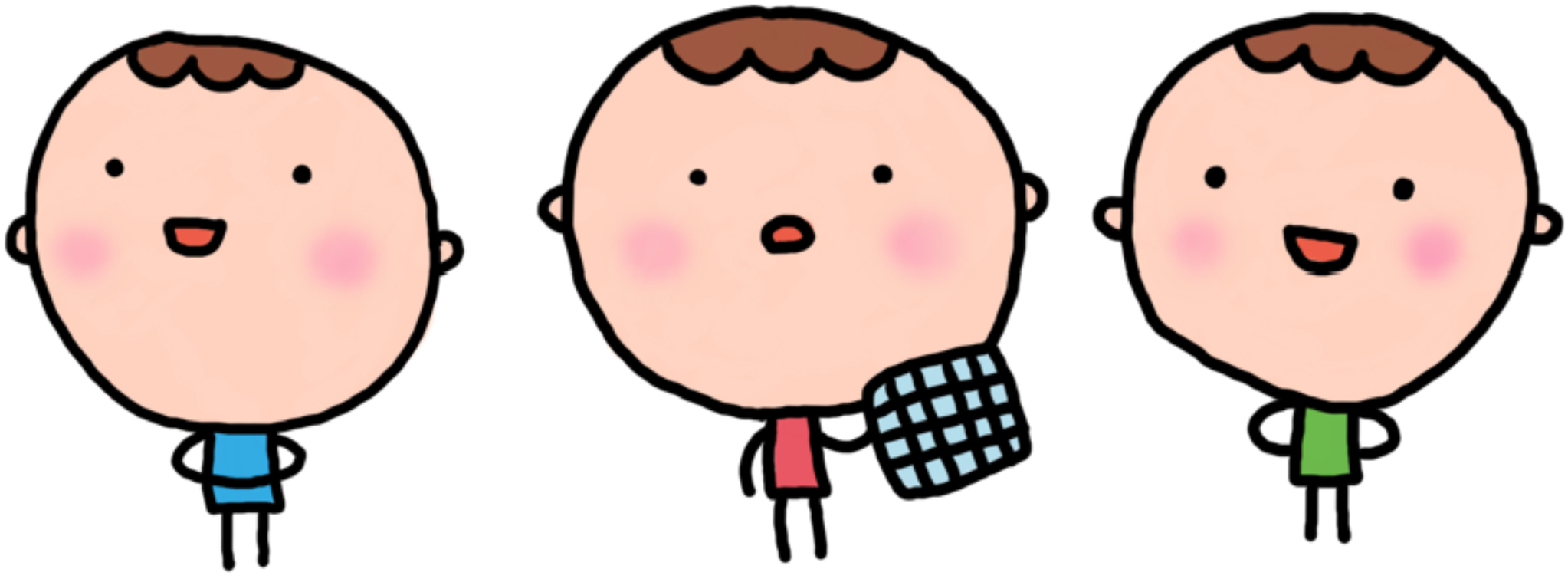
Quesiton	Answer
	Formosan black bear
	?
	Python

There might be infinitely many possible cases that we haven’t seen beore



Finding our functions by Neural Networks!

“Big 3” of Deep Learning (Neural Works)



Standard NN **CNN** **RNN**

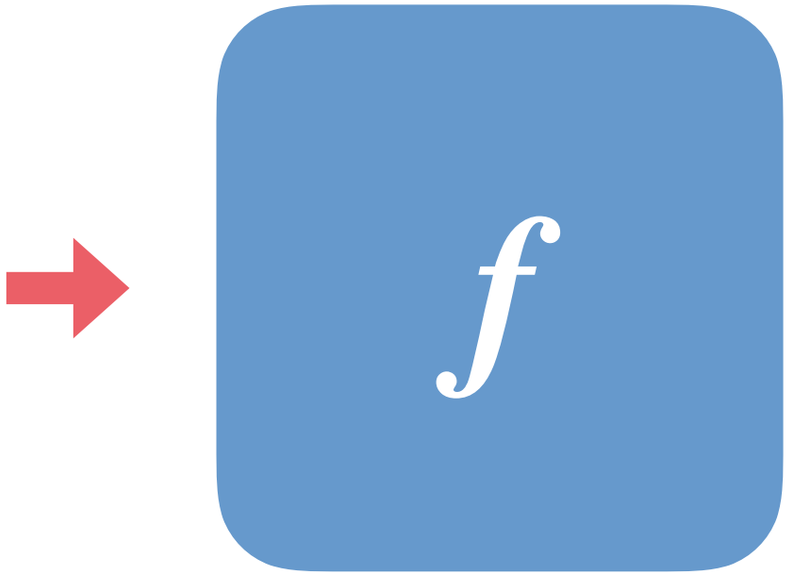
1

Asking a Question

Seeing an animal in wild, we want to know what is is?

2

Transforming our question into a function



Formosan black bear

3

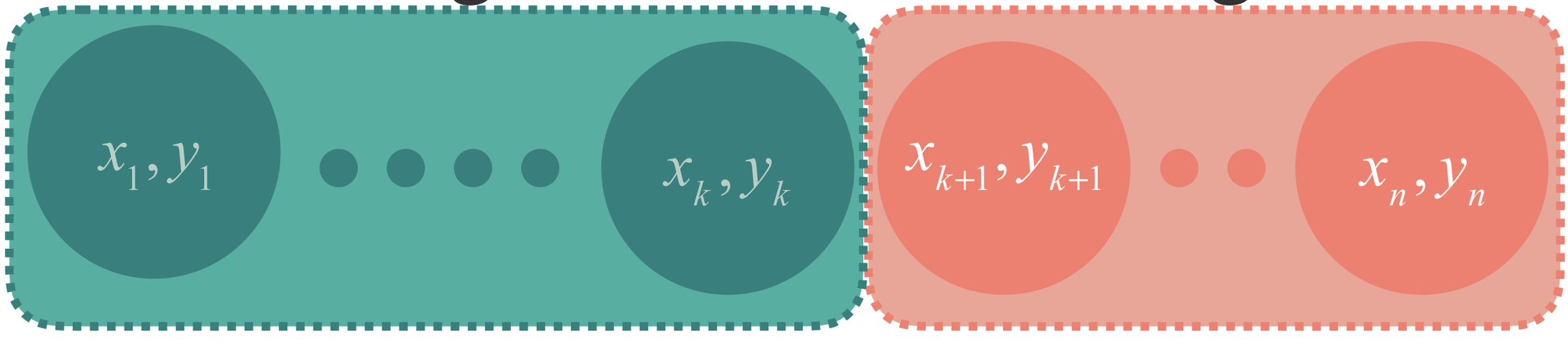
Preparing Data Sets for Training


, “Formosan black bear”), (

, “Python”), …

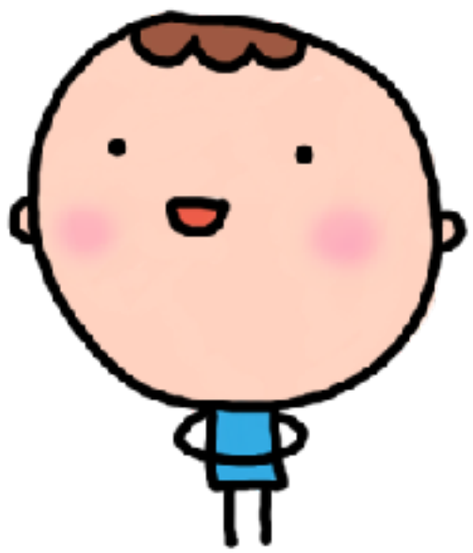
Training Data

Testing Data



4 Constructing NN

Standard NN



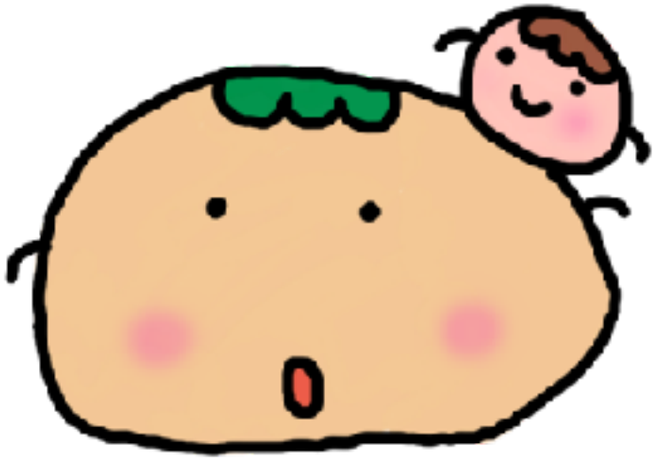
CNN

RNN

VAE

Capsule

Reinforcement Learning (RL)



Generative Adversarial Network (GAN)

Once we decide our structure of neural network, we have a set of parameters need to adjust.

$$\theta = \{w_i, b_j\}$$

Once we determine the parameters, we have a function:

$$f_\theta$$

5

Learning

The learning is sent to our neural network using our training data, adjusting our parameters, and then using a loss function to see how much difference we have and the observed value.

$$L(\theta)$$

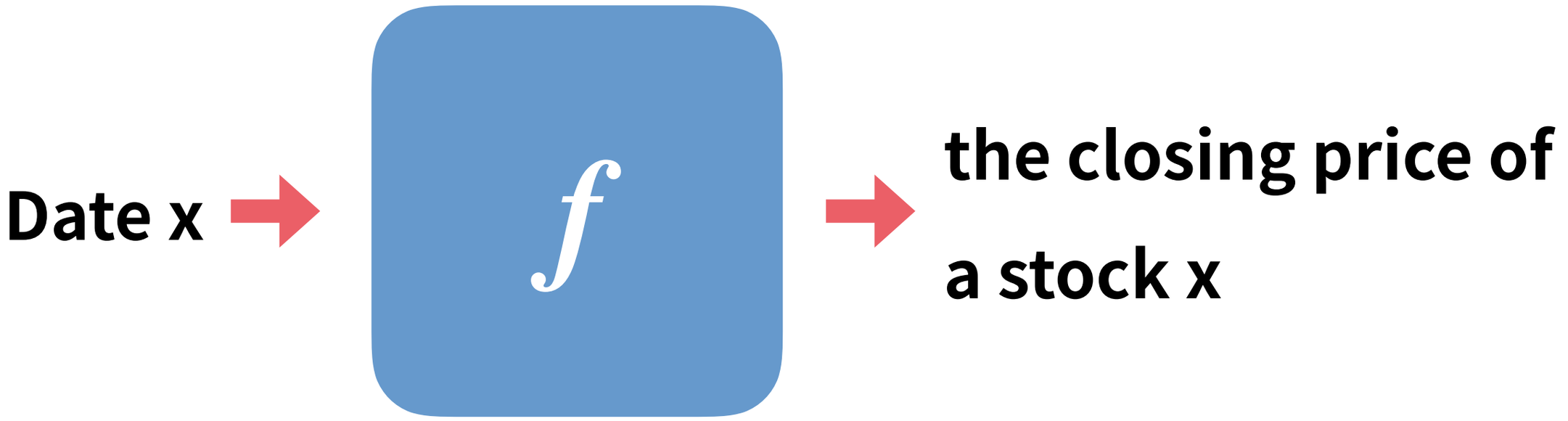
Basically, we use the method called:

gradient descent

For Neural Networks, the method is also called:

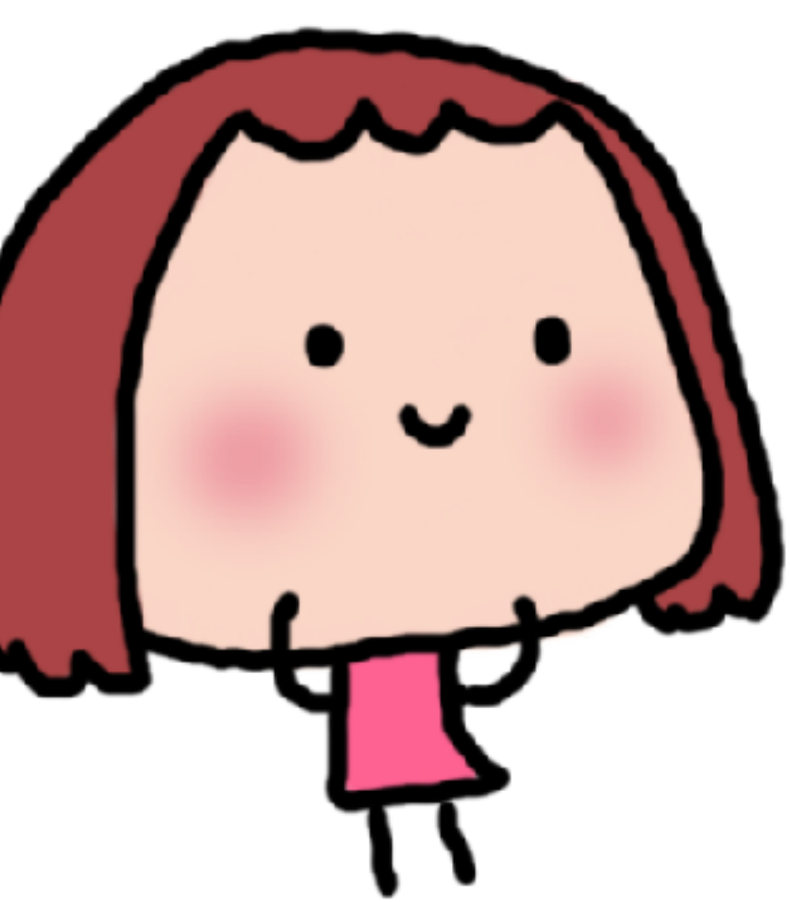
backpropagation

I want to know the closing price of a stock tomorrow.


Date x
f
the closing price of a stock x

NN, CNN, RNN

$x_{t-1}, x_{t-2}, x_{t-3},$
x_{t-4}, x_{t-5}



f

x_t

Use the prices of the previous week to predict the price for the next day.

I want to know how may homers a particular MLB player can hit in 2018 season.

15 features!

[Age, G, PA, AB, R, H, 2B, 3B, HR,

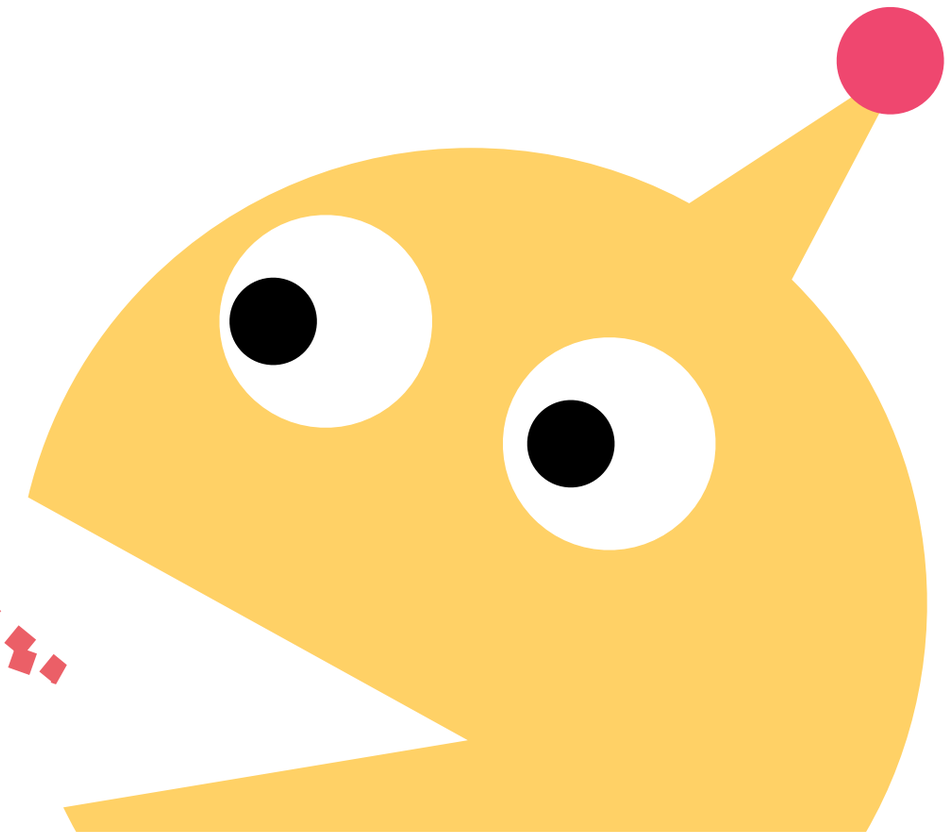
RBI, SB, BB, SO, OPS+, TB]

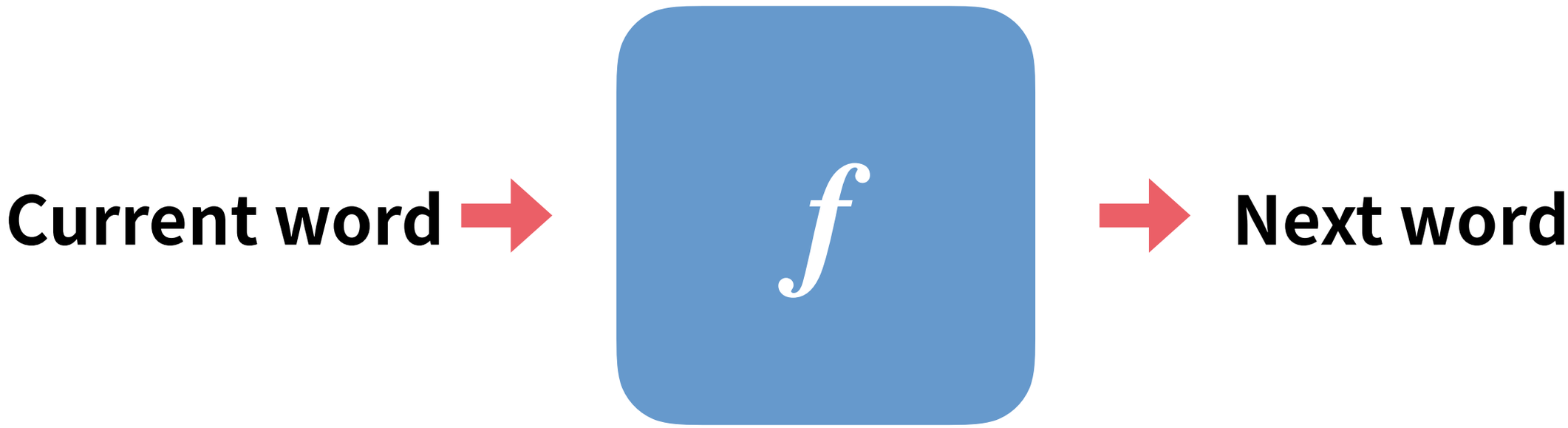
Player preformance in the year of t-1

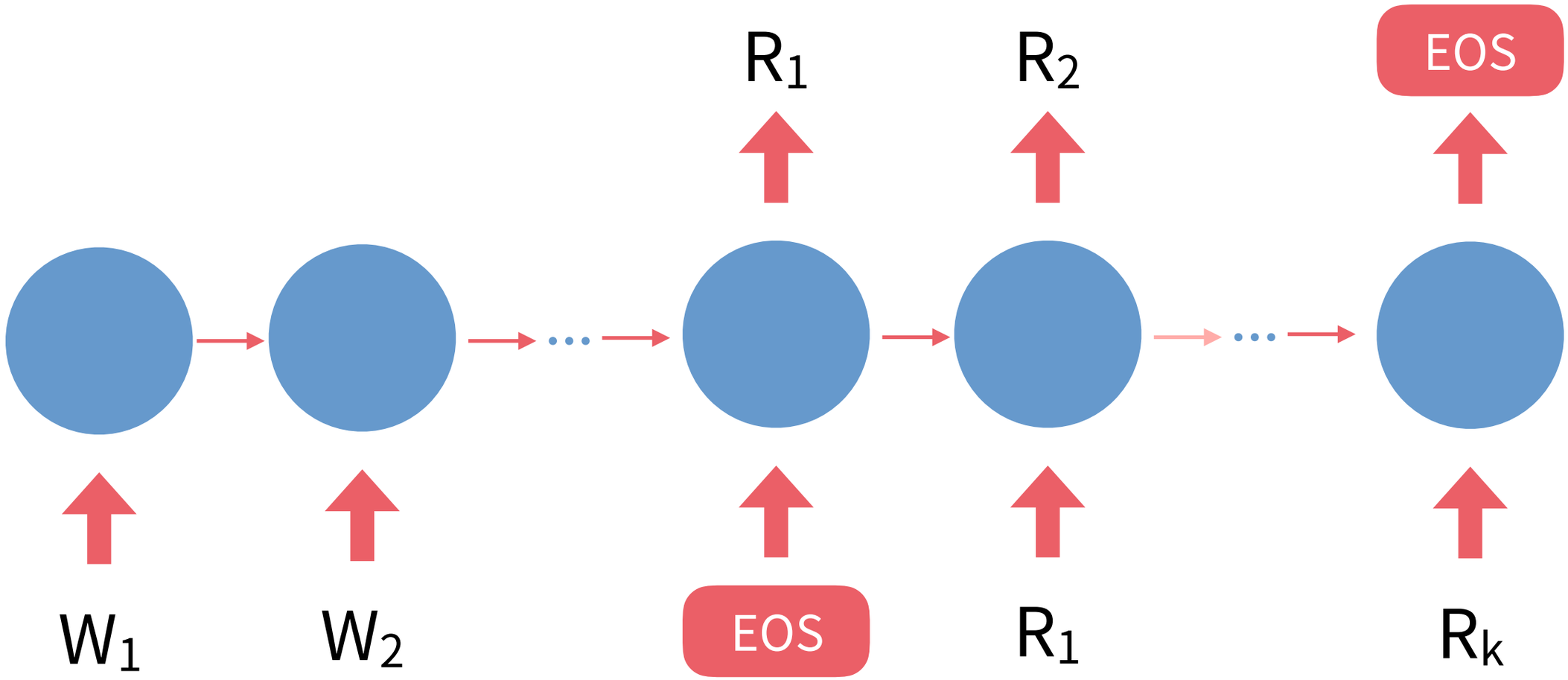
RNN

Number of homers in the year of t

Chatbot

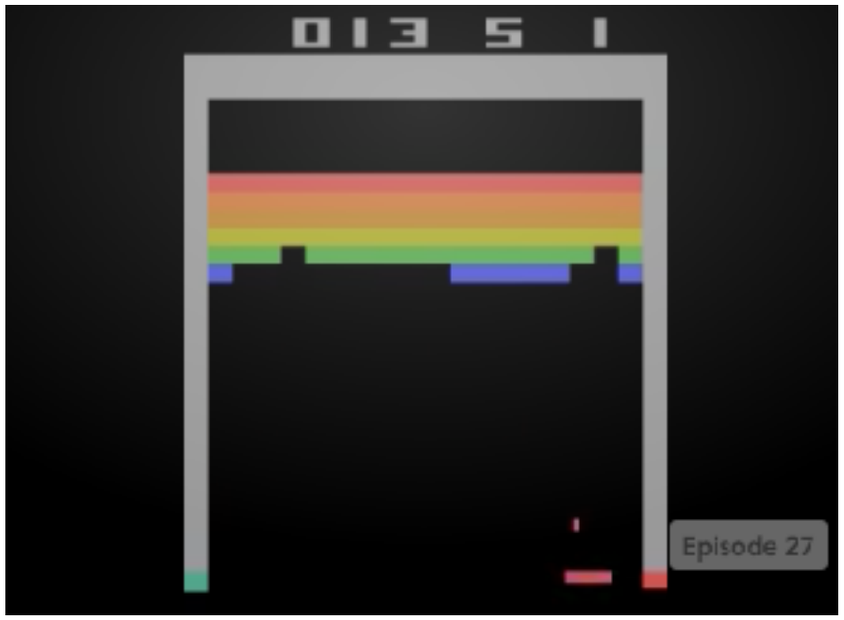


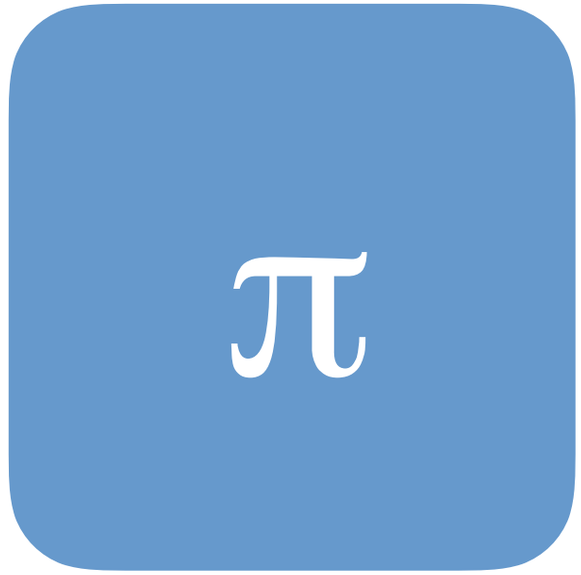

Current word
f
Next word



Note that in this mode, each input and output is not of fixed length!

I want to use AI to play games (drive a car, brew coffee...)

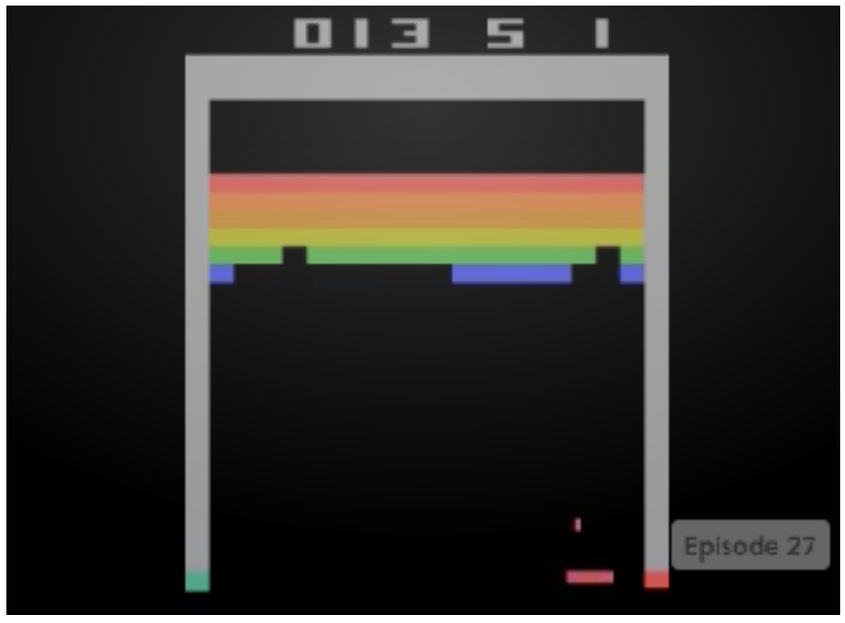




The best action

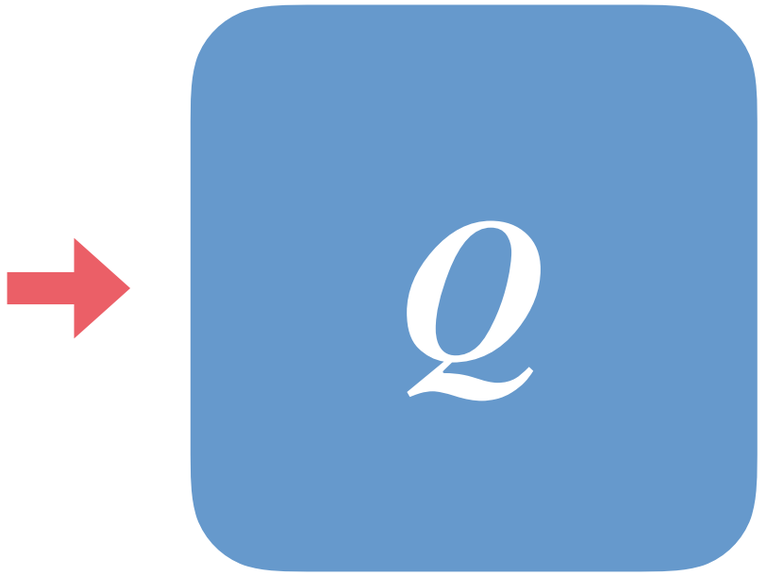
CNN + NN

Reinforcement Learning



+

Left



Score

some characters are missing in a font I like…

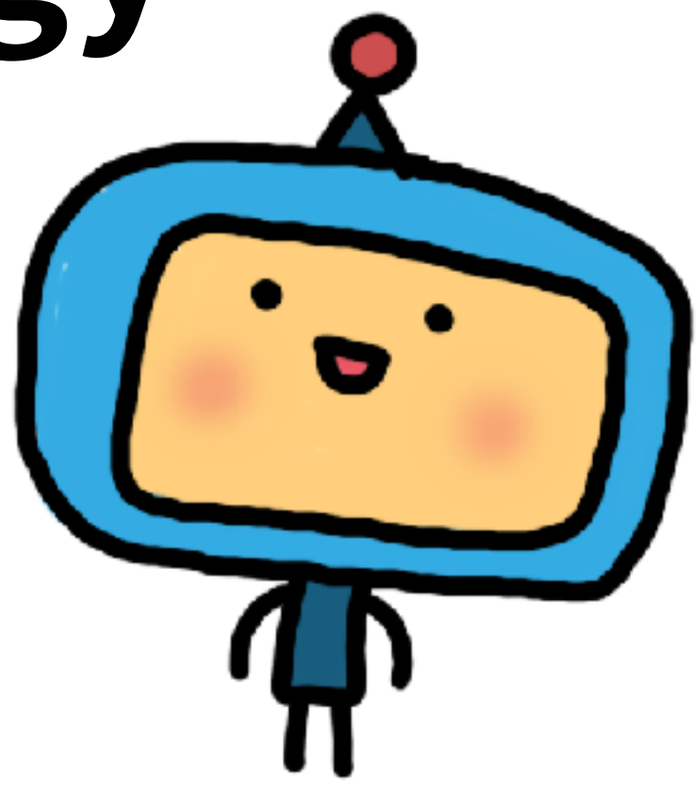
Font A

Font B

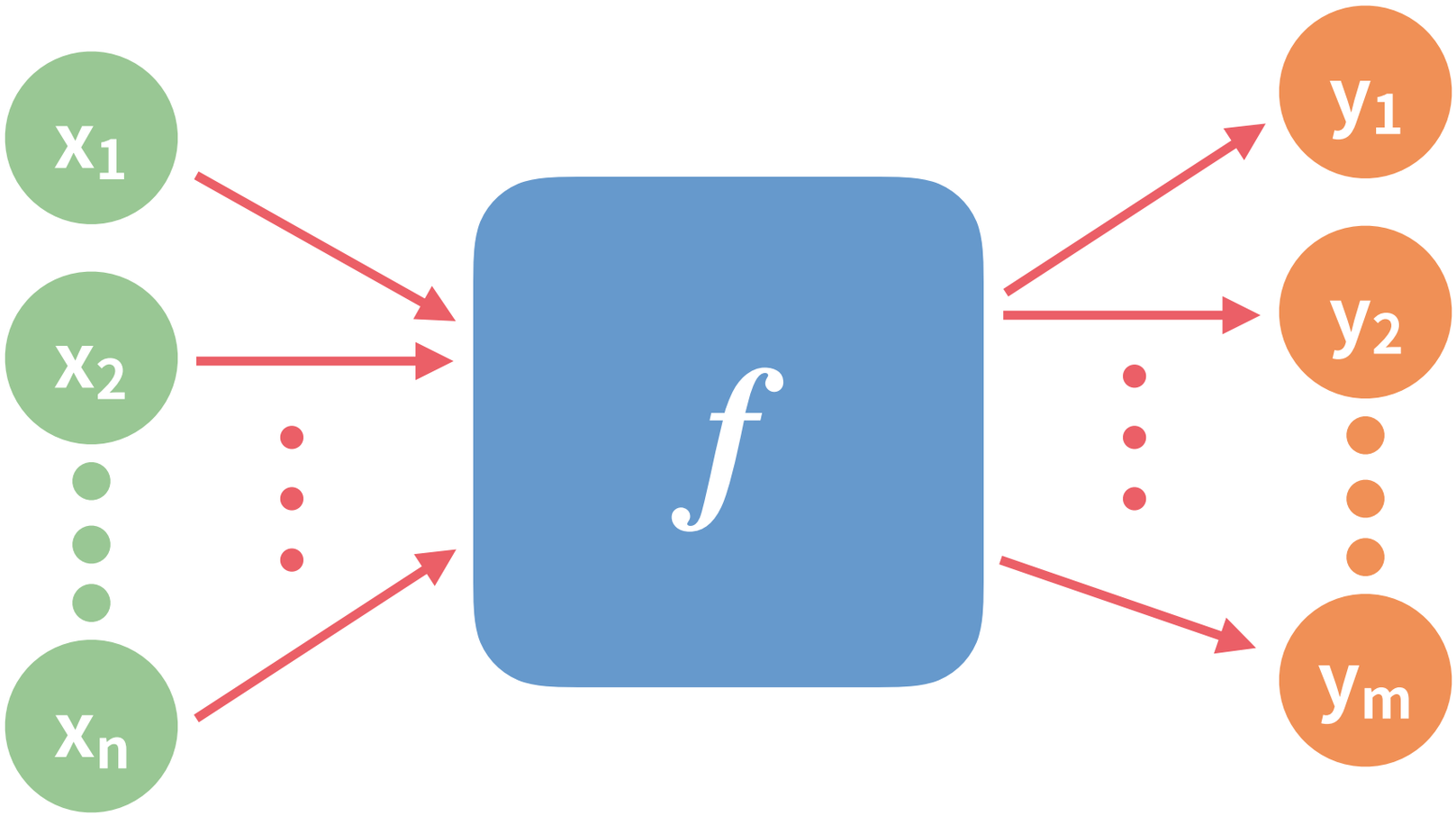
CNN, VAE, GAN

2

AI Core Technology - Principles of Neural Networks



Remember we just have to learn a function



Three Steps of Learning Functions

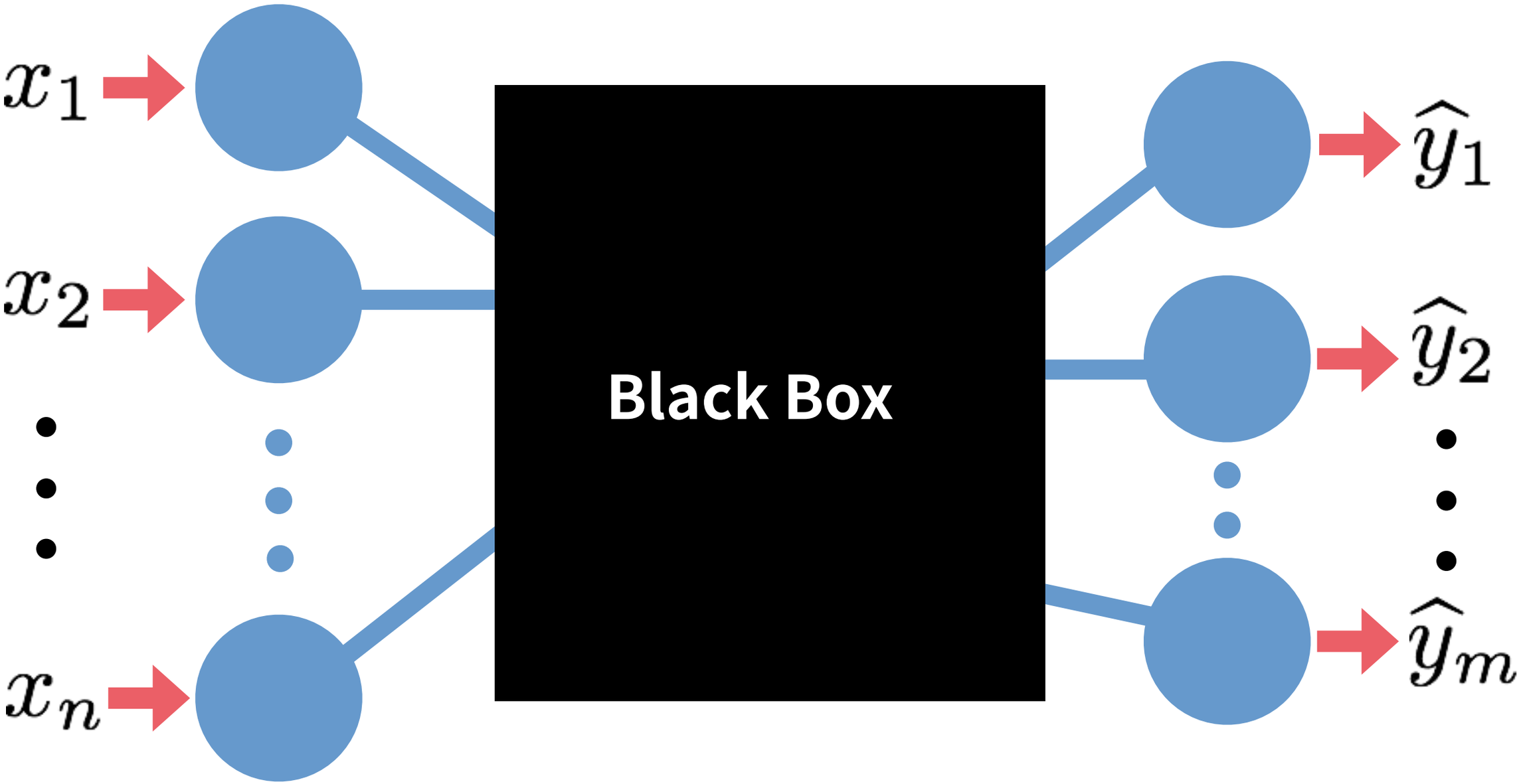
- Transfer a real world problem into a function.
- Collect training materials that we know "correct answers."
- Find the function!

Black Box Learning

There are really techniques for learning arbitrary functions

neural networks

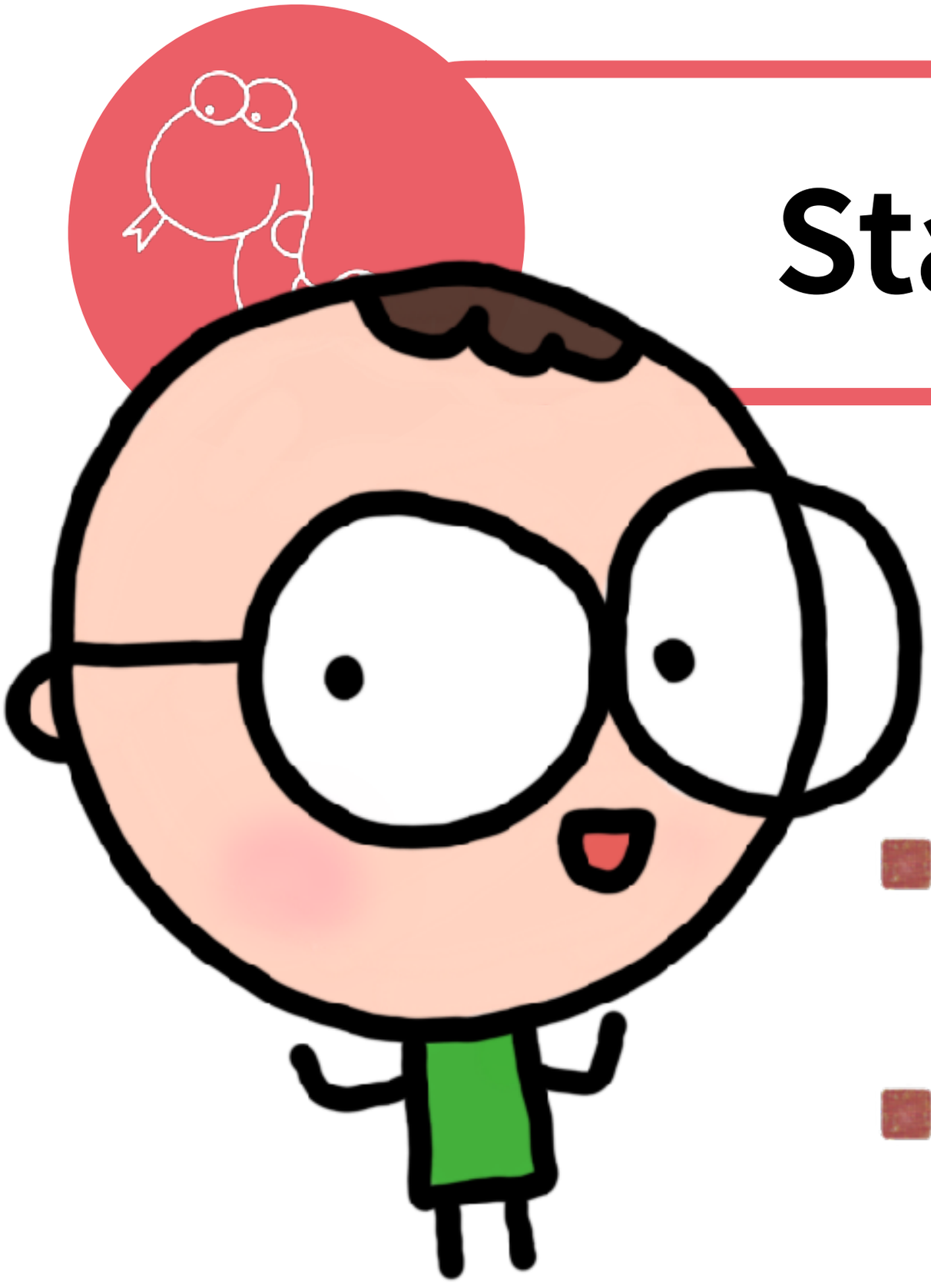
In the 1980-1990 or so, it is a pretty fancy stuff.


x_1
x_2
x_n
Black Box
$\widehat{y}_1$
$\widehat{y}_2$
$\widehat{y}_m$
Input Layer
Hidden Layer
Output Layer

What is powerful is that neural networks will learn everything!

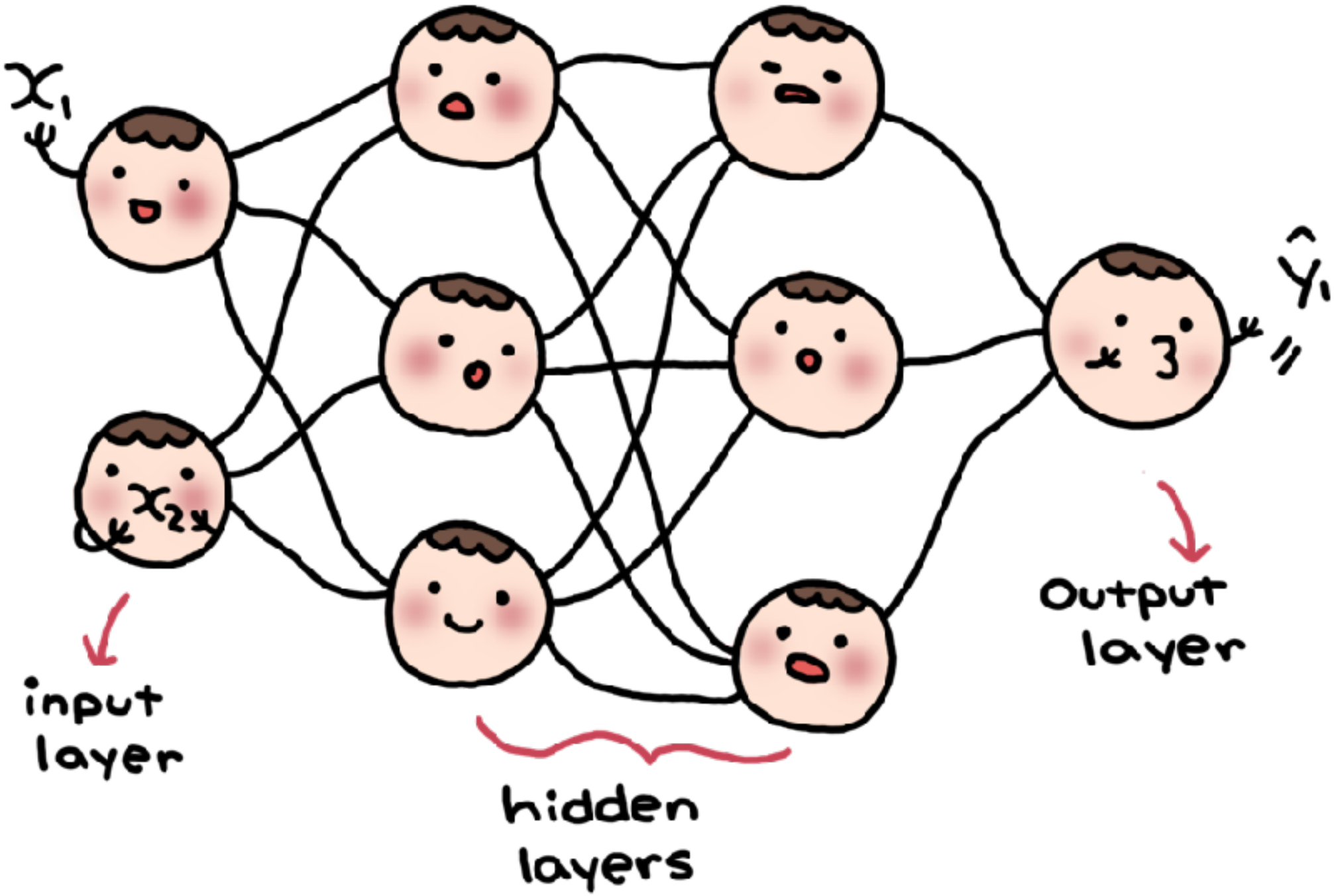
And you don't have to tell it what the function should look like: linear, quadratic polynomial, and so on.

Open the Black Box!

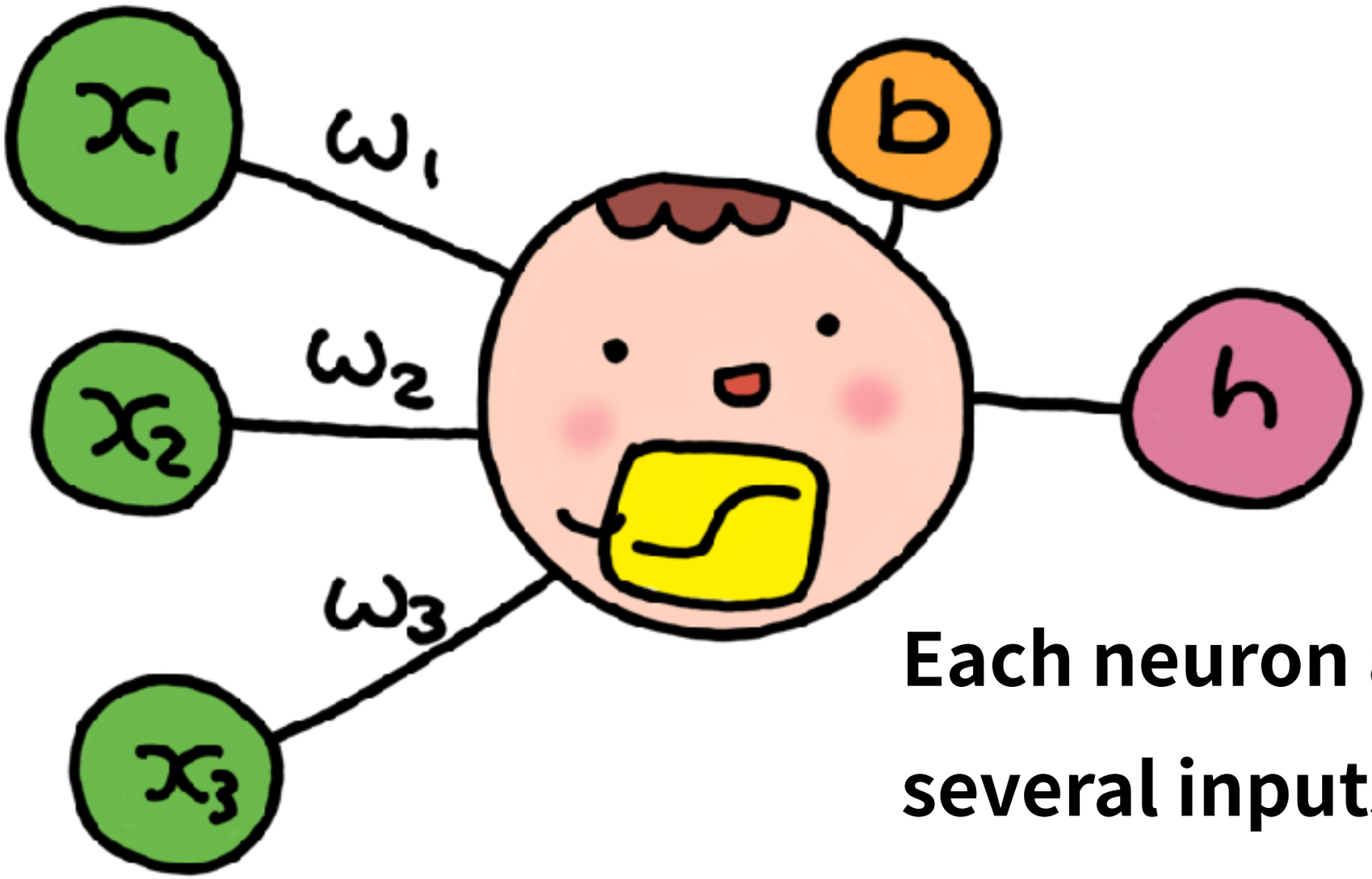


Standard NN

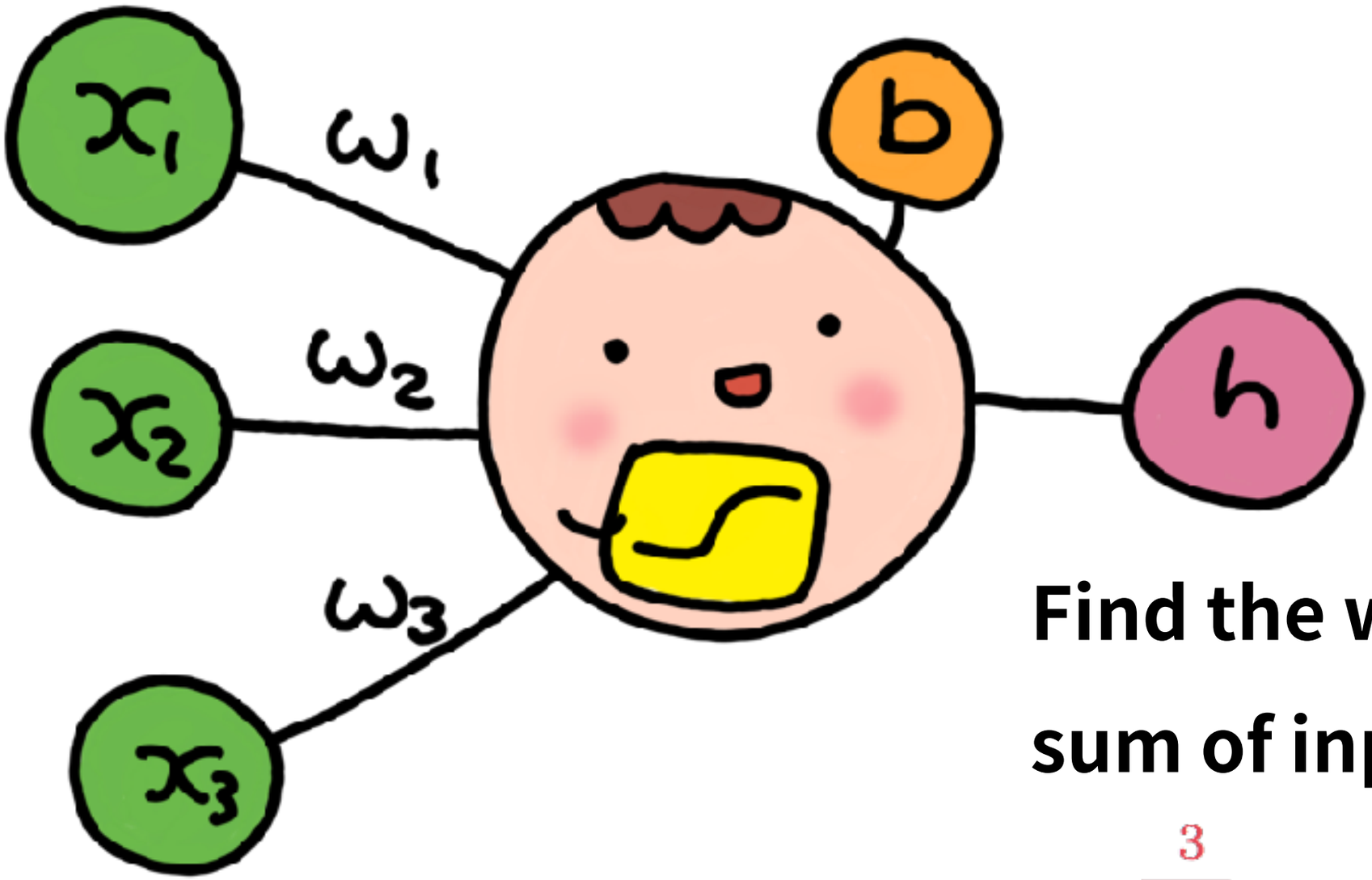
- Fully Connected Neural Networks
- Very popular since 1980s


x_1
x_2
$\hat{y}_1$
input layer
hidden layers
Output layer

The action of every neuron is basically the same!

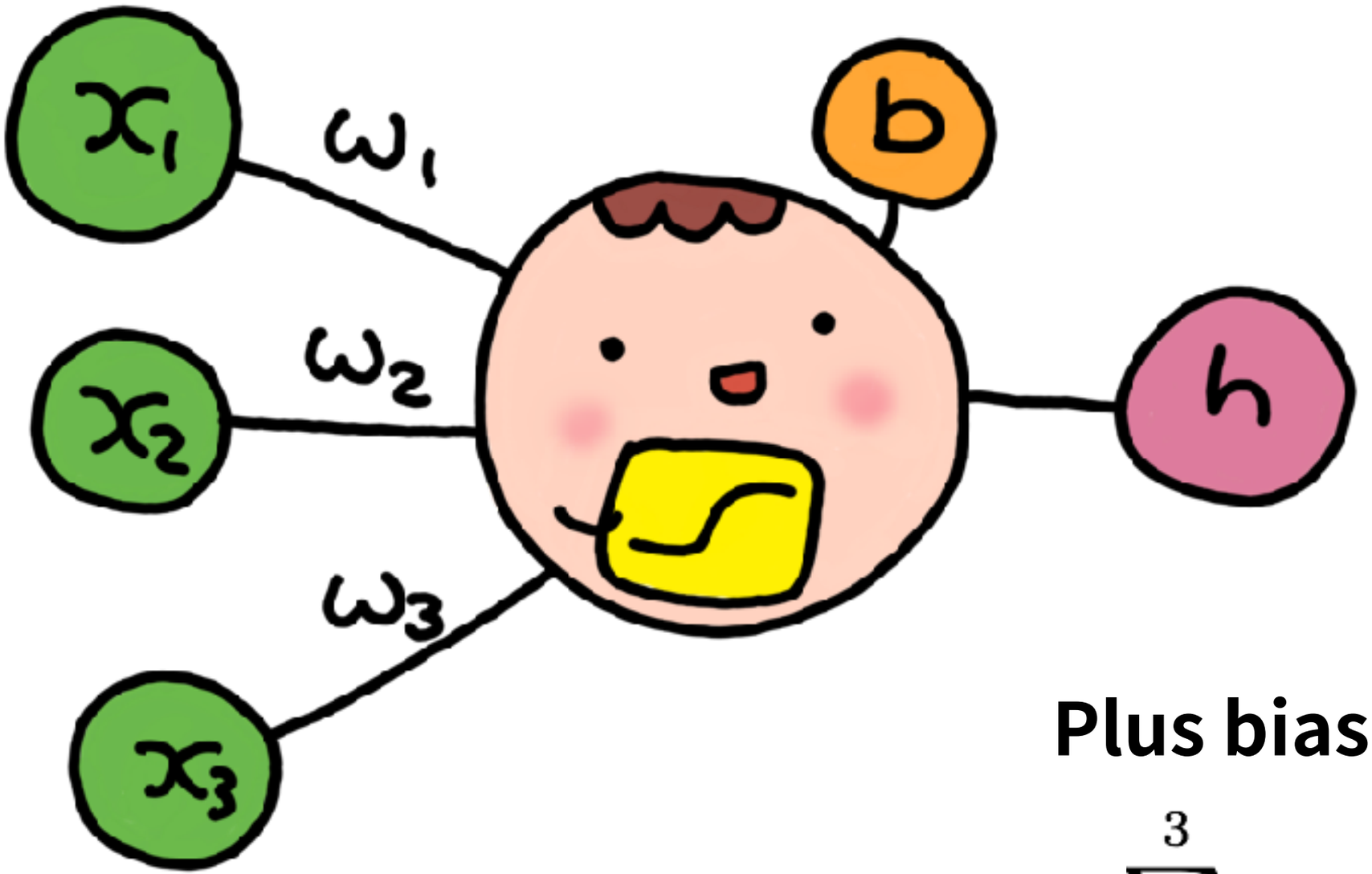


Each neuron accepts several inputs and then sends one output.



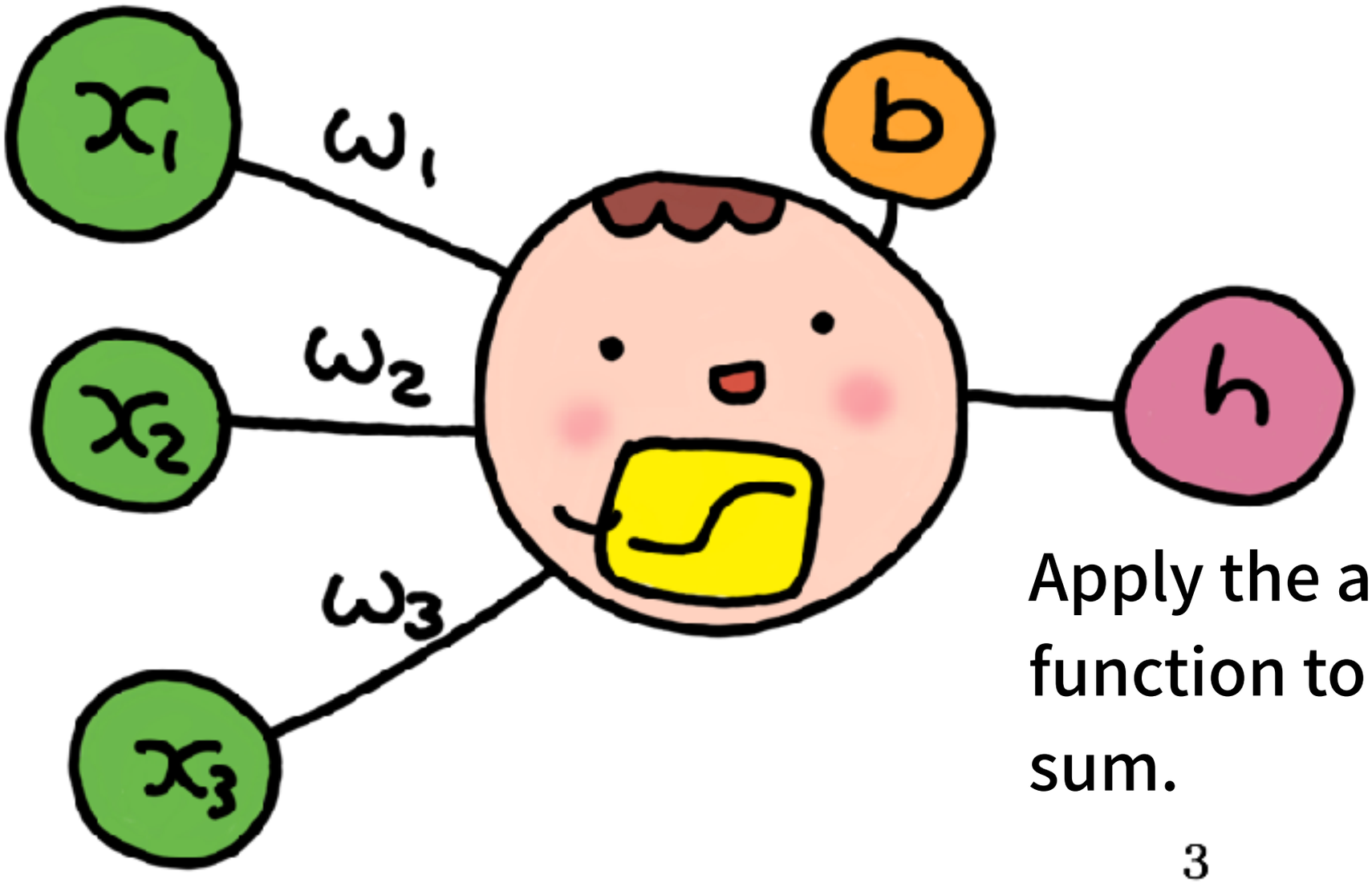
Find the weighted sum of inputs.

$$\sum_{i=1}^{3} w_i x_i$$



Plus bias.

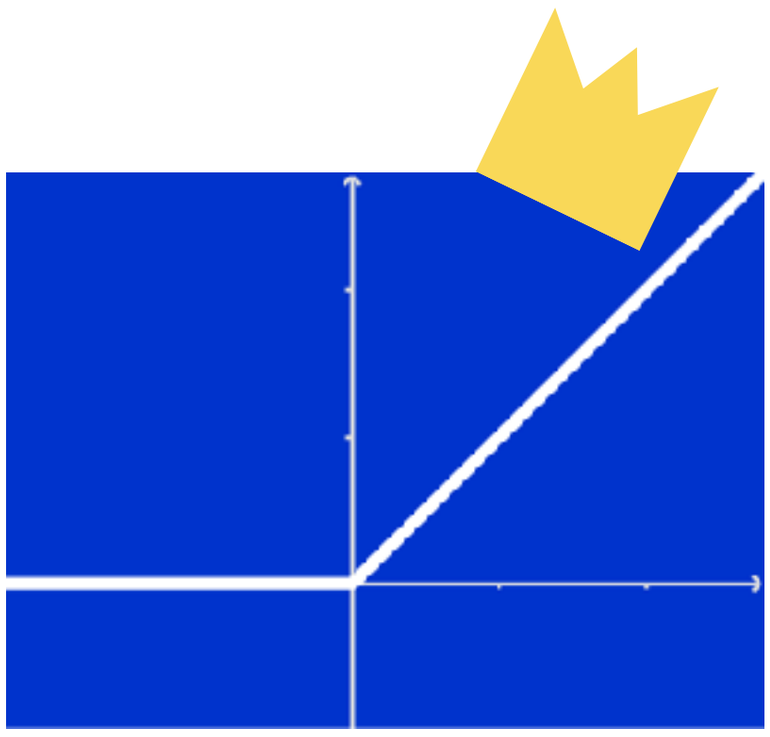
$$\sum_{i=1}^{3} w_i x_i + b$$



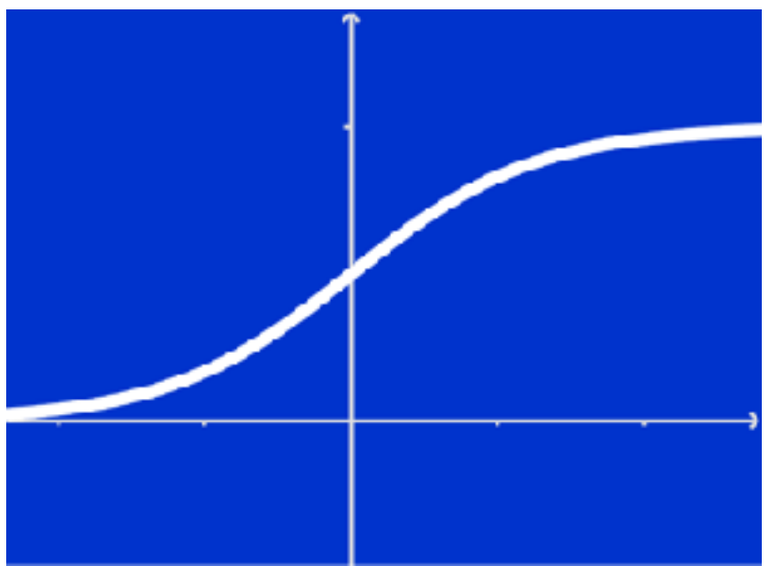
Apply the activation function to the weighted sum.

$$\varphi\left(\sum_{i=1}^{3} w_i x_i + b\right) = h$$

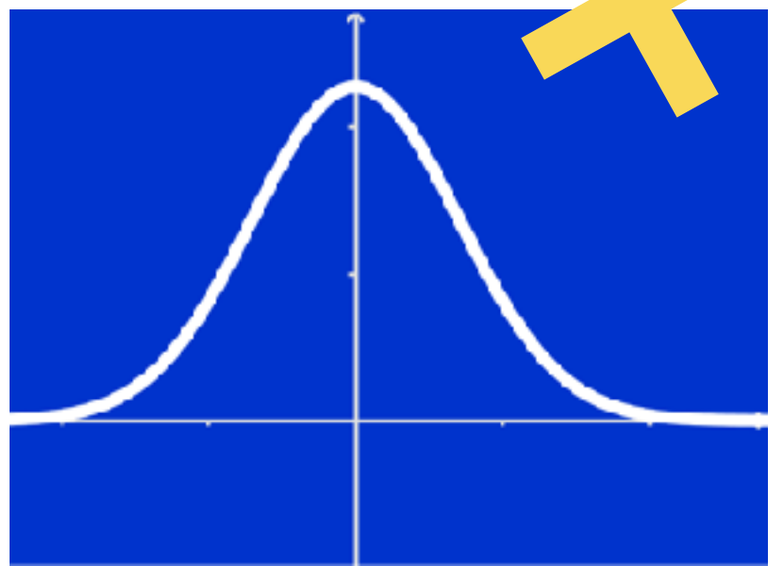
Popular activation functions



ReLU

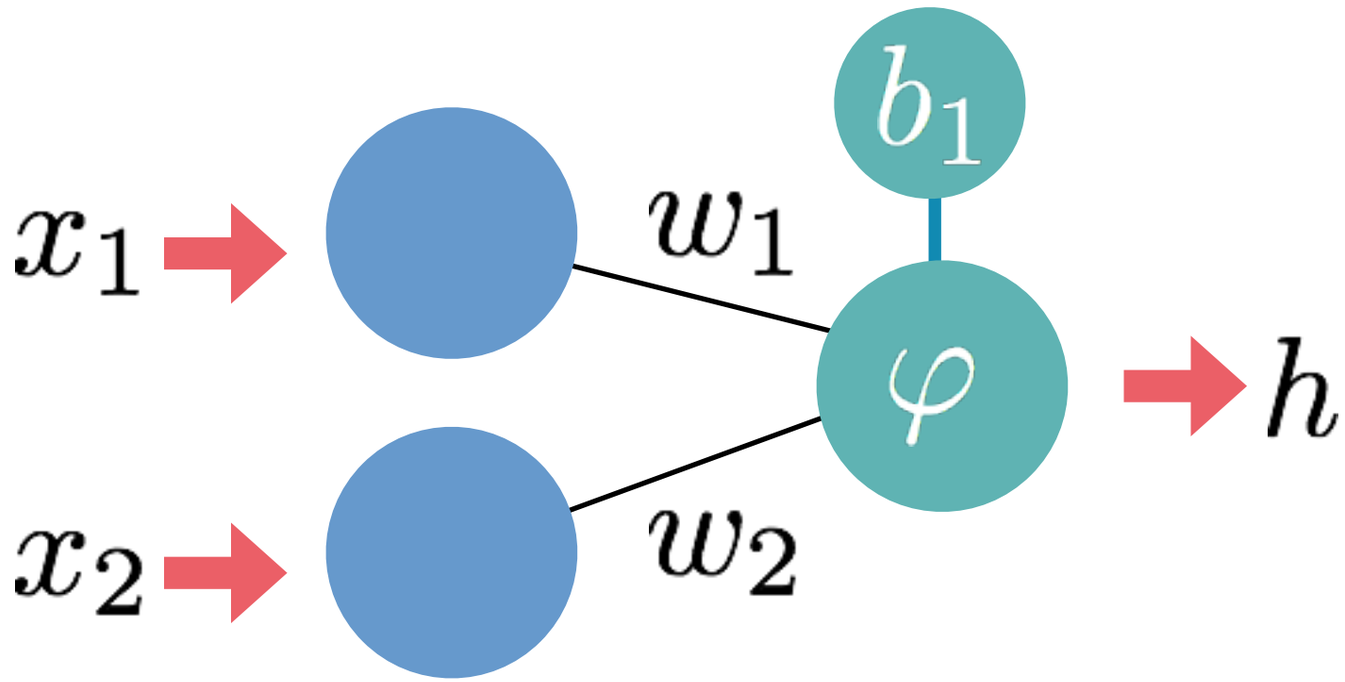


Sigmoid



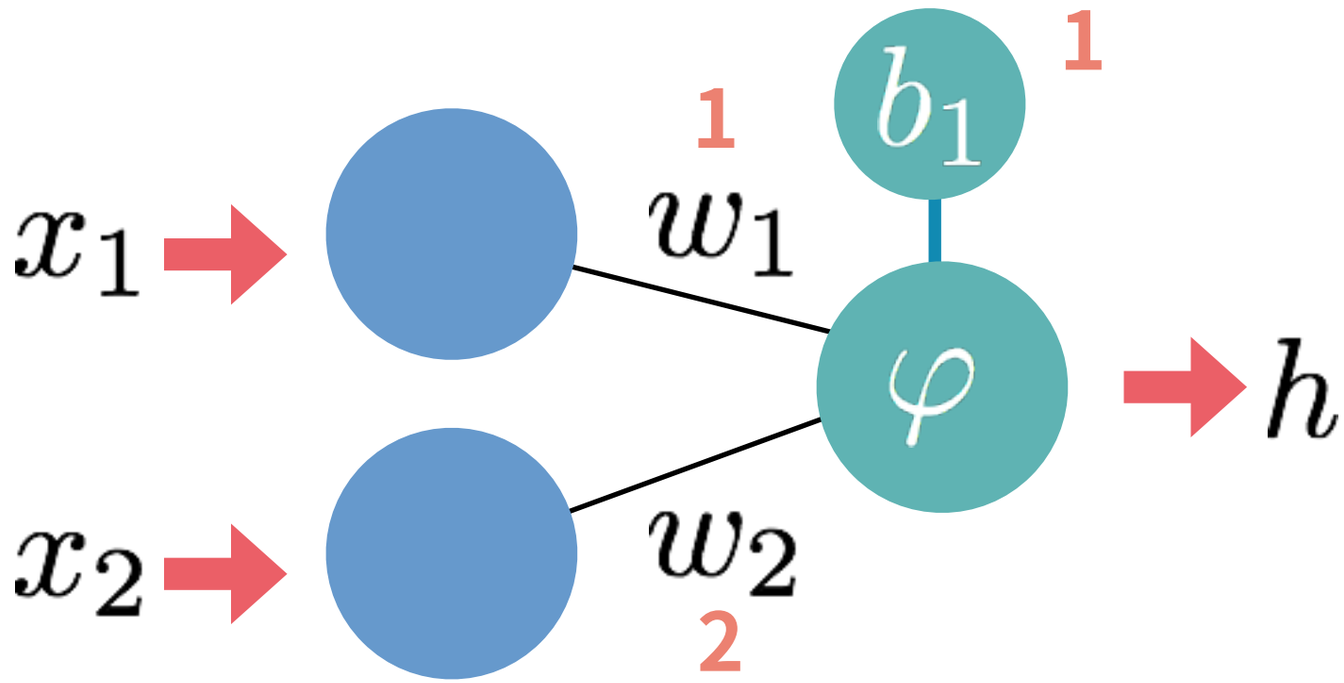
Gaussian

Parameters are weights, biases



$$\varphi(w_1 x_1 + w_2 x_2 + b_1) = h$$

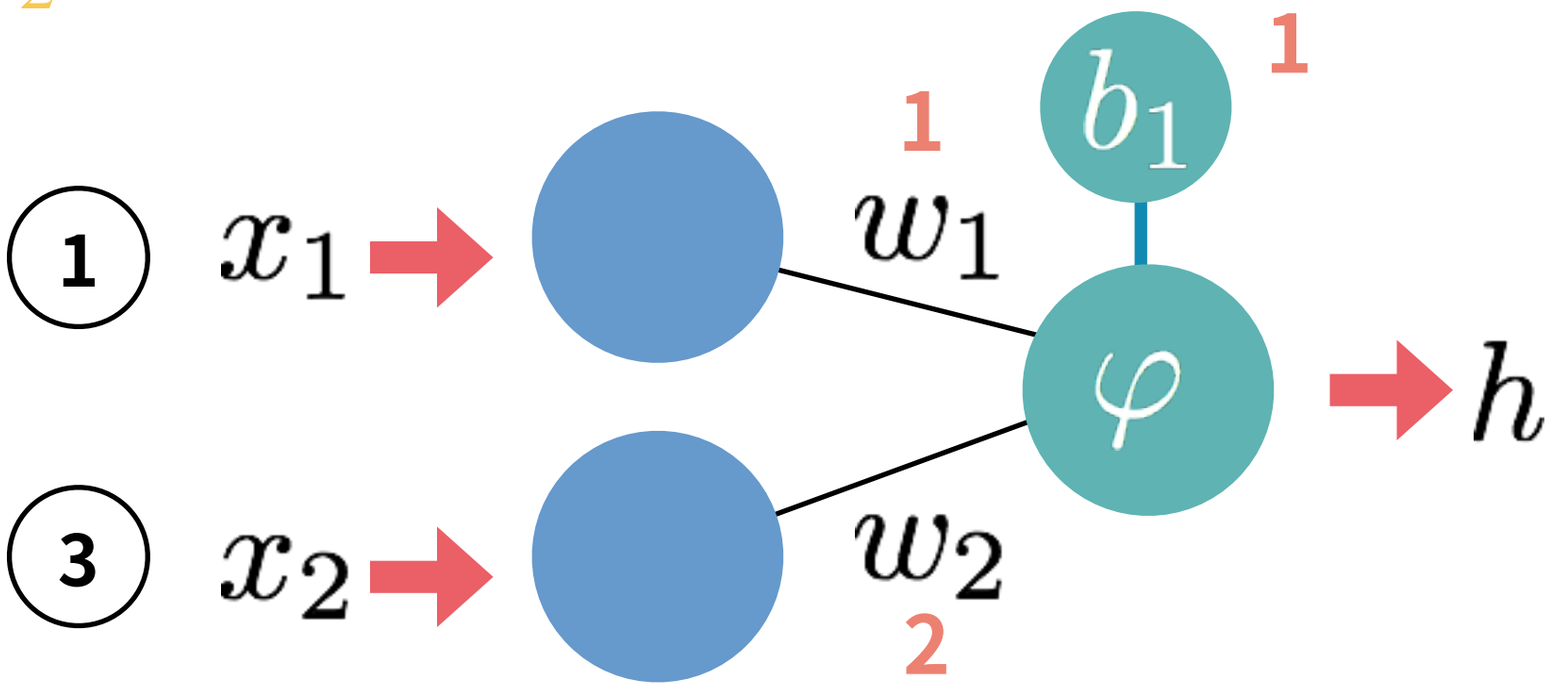
“Learned” neural network



$$\varphi(\underset{1}{w_1}x_1 + \underset{2}{w_2}x_2 + \underset{1}{b_1}) = h$$

Suppose we have the input

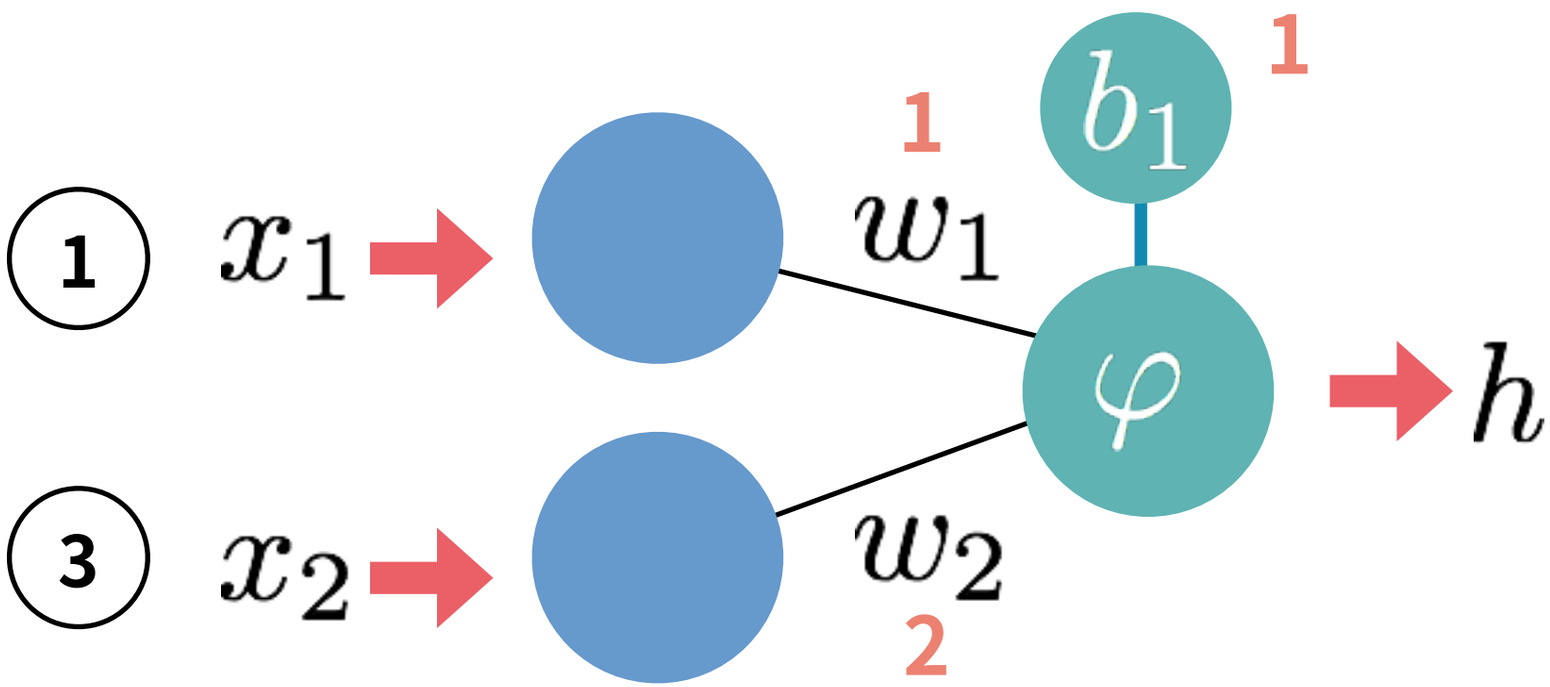
$(x_1, x_2) = (1, 3)$



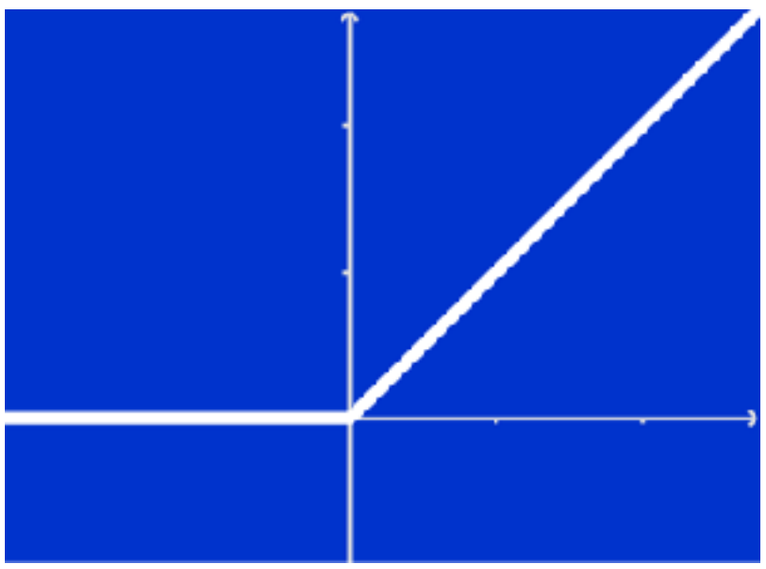
$$\varphi(\underset{1}{w_1}\underset{(1)}{x_1} + \underset{2}{w_2}\underset{(3)}{x_2} + \underset{1}{b_1}) = h$$

8

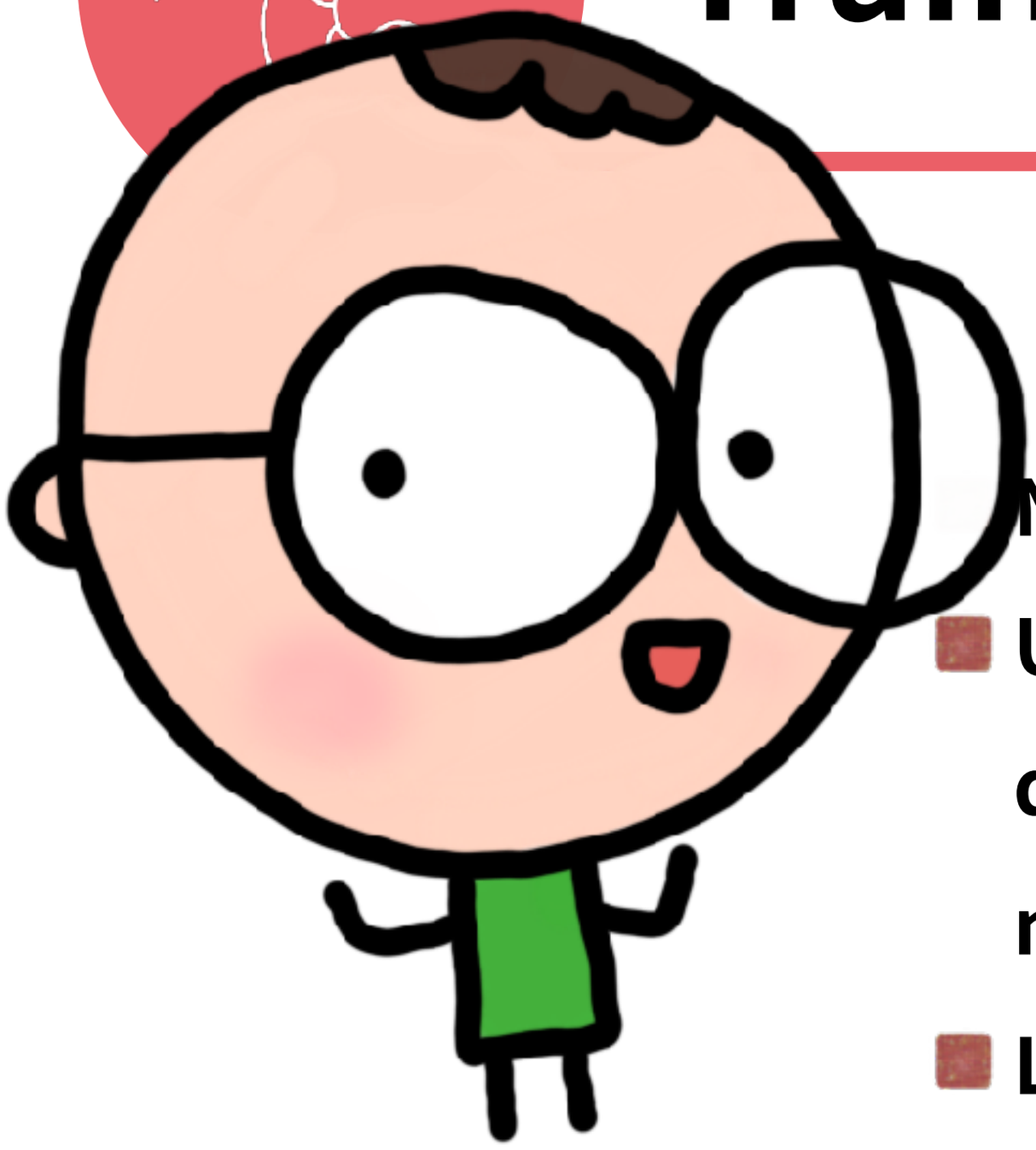
Using ReLU as activation functions



$$\varphi(8) = 8$$



Training Methods



- NN needs to be trained
- Using “previous exam questions” to train our neural network
- Learning method is called backpropagation

The Function Space of our NN

When a neural network structure is determined and activation functions are also determined, it can be adjusted by weights, biases. We call the set of these parameters θ, each of which defines a function, and we treat it as a set.

$$\{F_\theta\}$$

We are looking for θ^*

Makes F_{θ^*} the closest to the objective function

What does “the closest” mean

That means the value of “loss function” is minimum

Suppose we have the training data

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$$

The most popular loss function

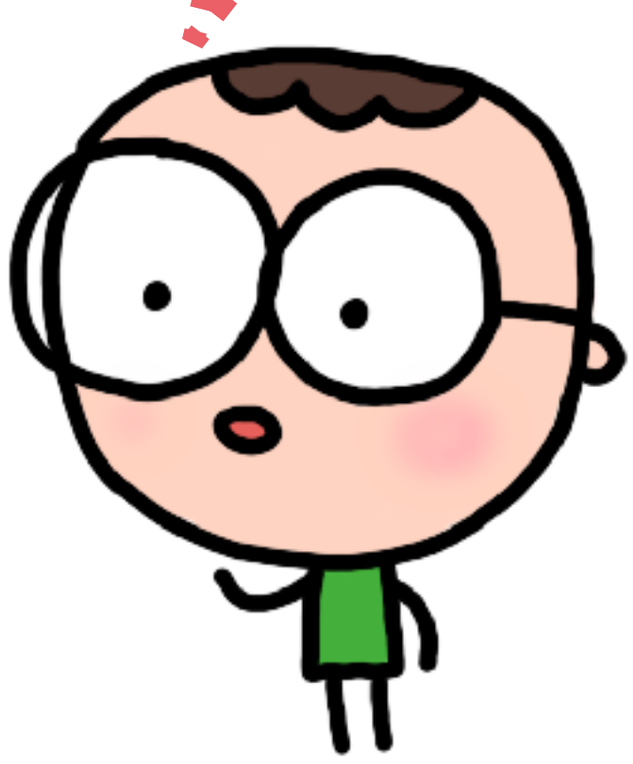
We the value is as small as possible

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{k} \|y_i - F_\theta(x_i)\|^2$$

What's this!?

x	網路輸出	真正的	誤差
1	2	3	1
2	5	2	-3
3	6	8	2
4	7	2	-5
5	6	6	0
6	4	8	4
7	8	9	1

How to calculate total error?



x	網路輸出	真正的	誤差
1	2	3	1
2	5	2	-3
3	6	8	2
4	7	2	-5
5	6	6	0
6	4	8	4
7	8	9	1

x	網路輸出	真正的	誤差
1	2	3	1
2	5	2	-3
3	6	8	2
4	7	2	-5
5	6	6	0
6	4	8	4
7	8	9	1

1

9

4

25

0

16

1

Squared Sum

56 in total

Basically, do such adjustment

learning rate

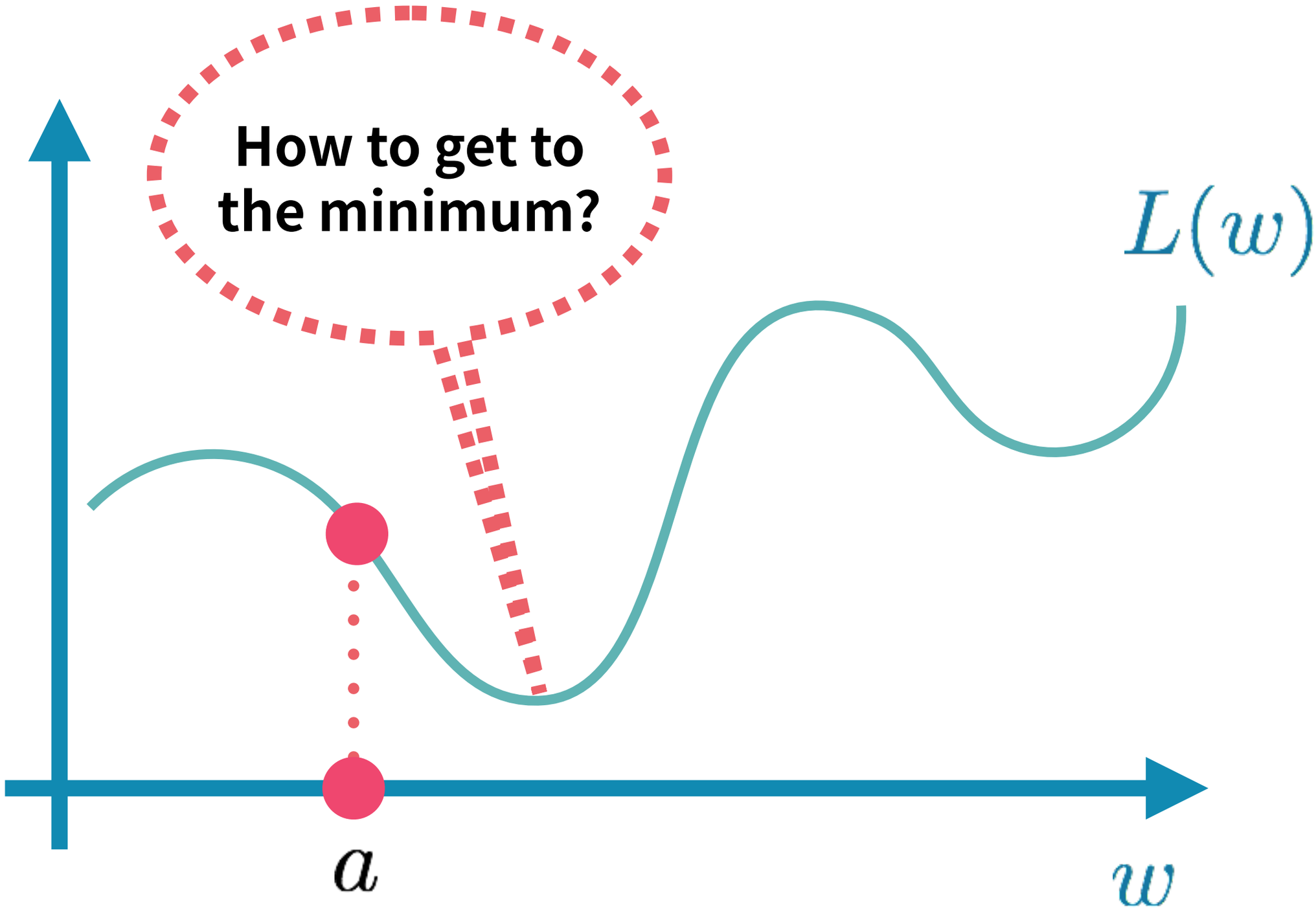
$$\eta \frac{\partial L}{\partial w_{ij}}$$

What does this terrible thing mean?

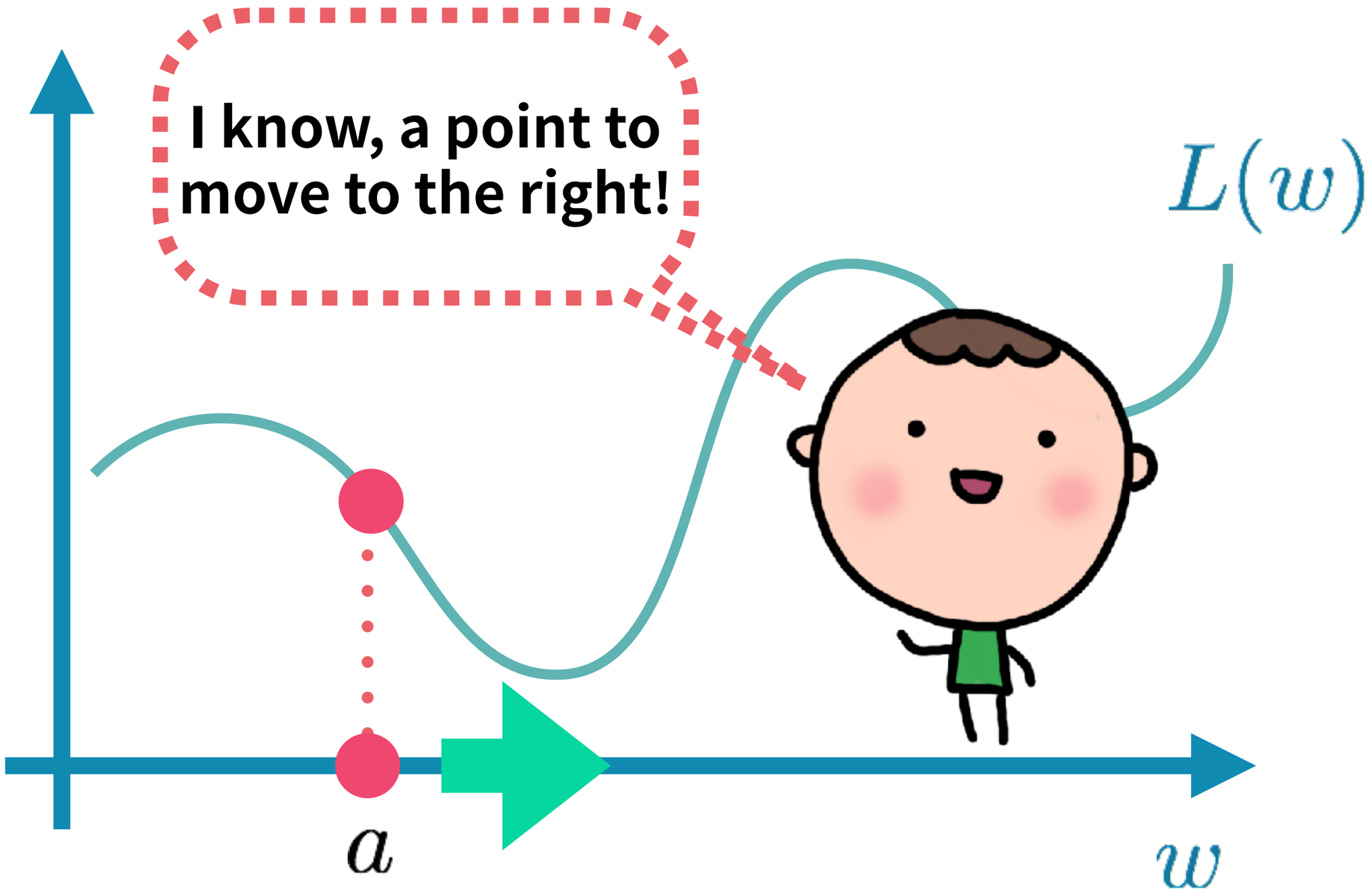
Remember that L is a function of w_1, w_2, b_1, …

The smaller the better

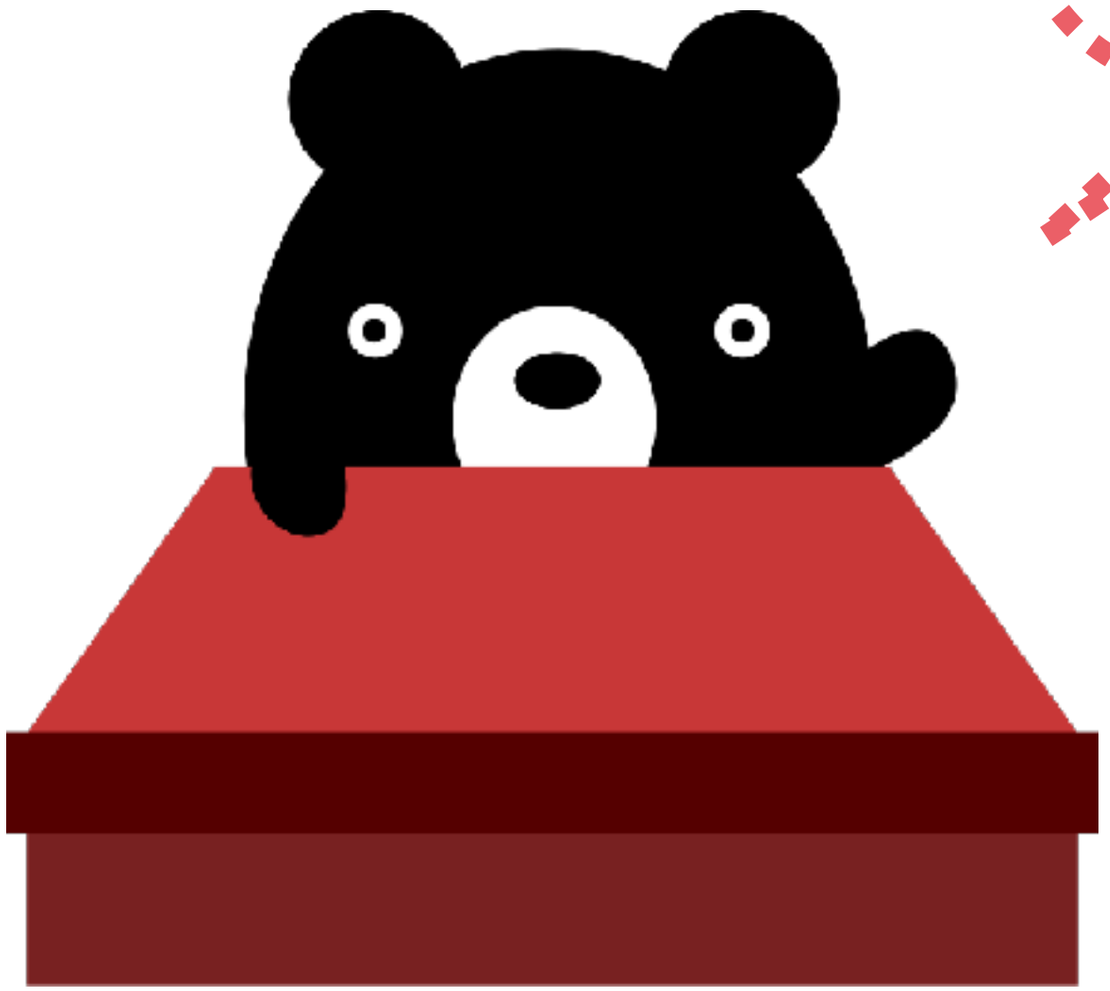
For simplification, we first think of L has only one variable w



Current value of w

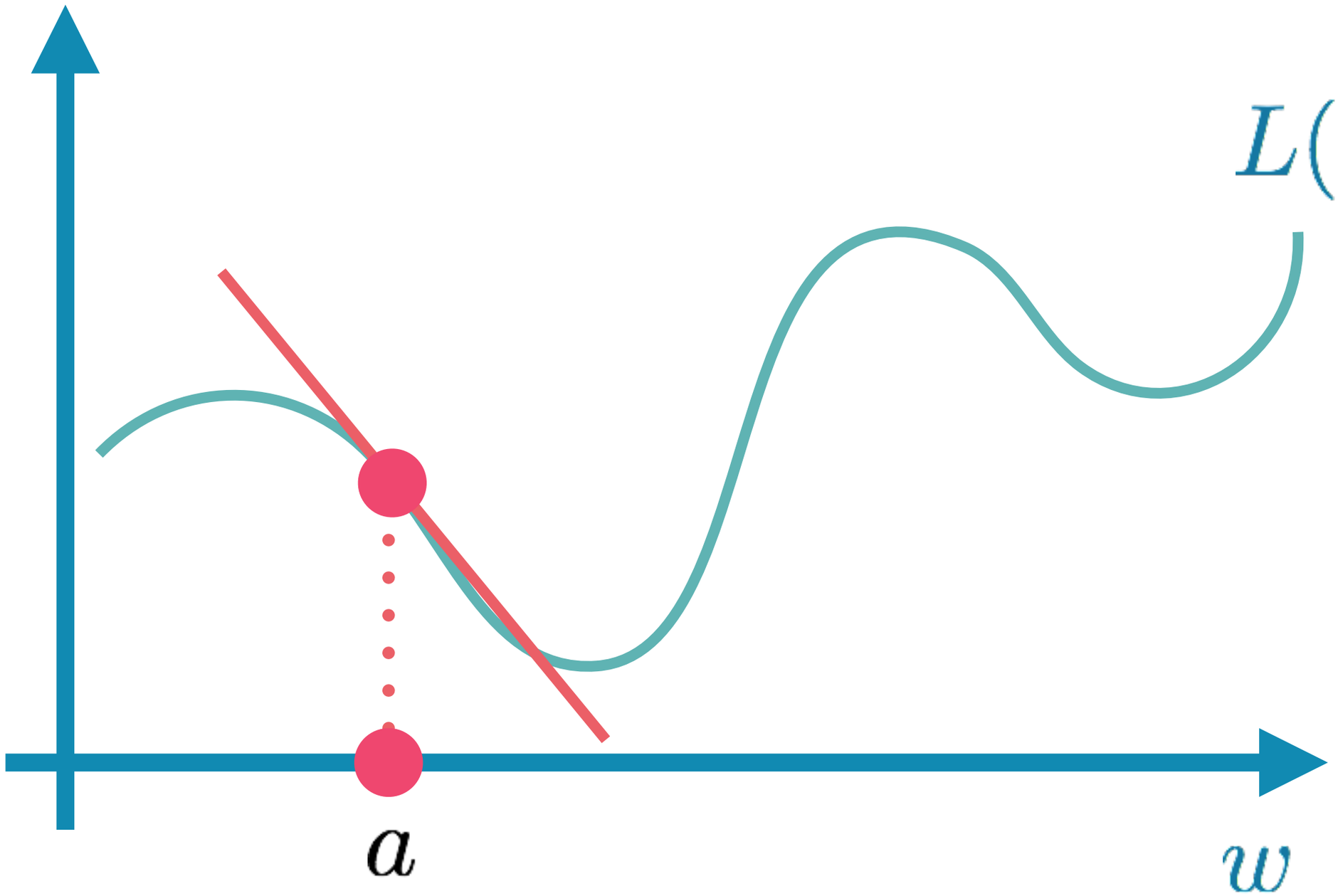


Current value of w

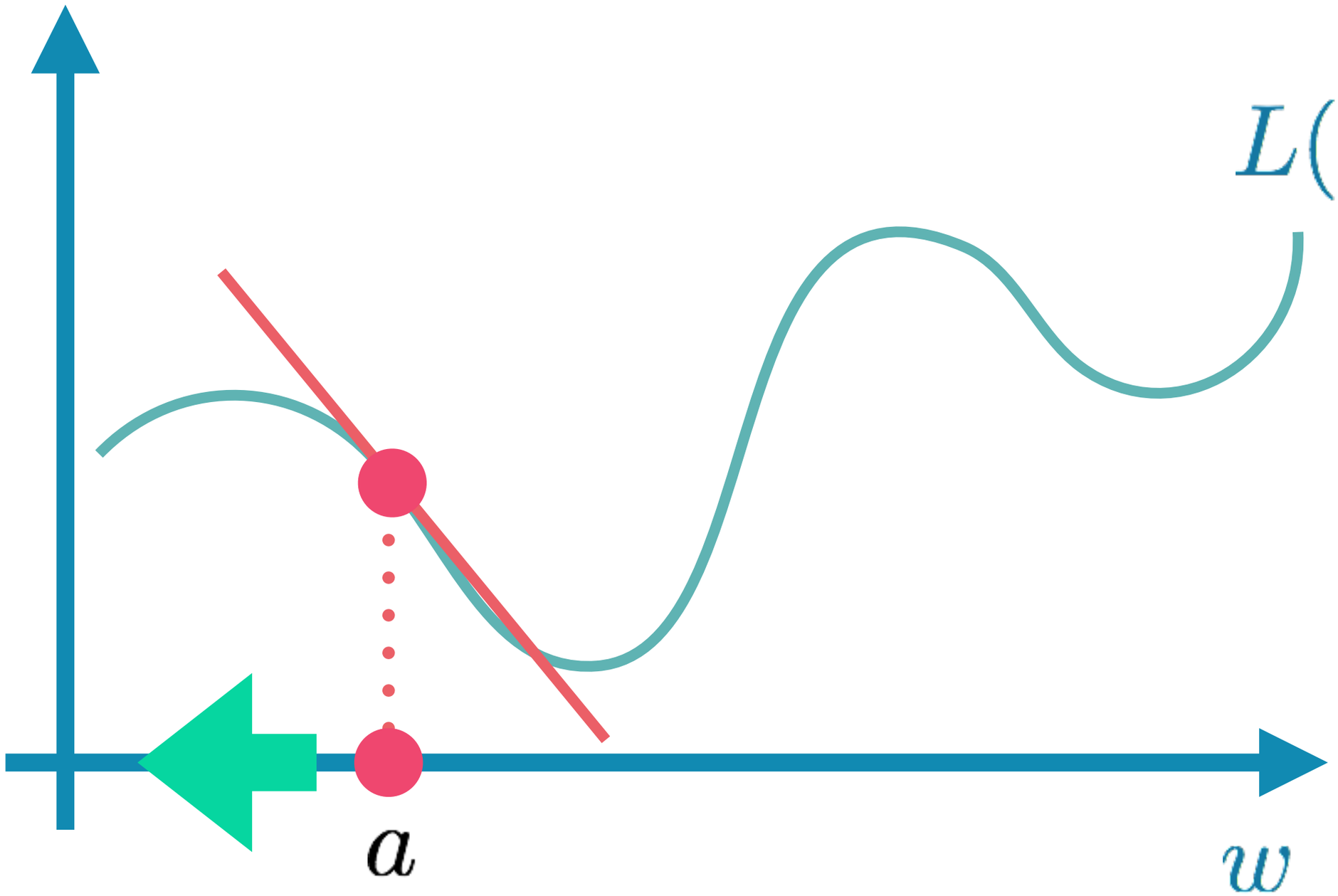


How does the computer “see” it?

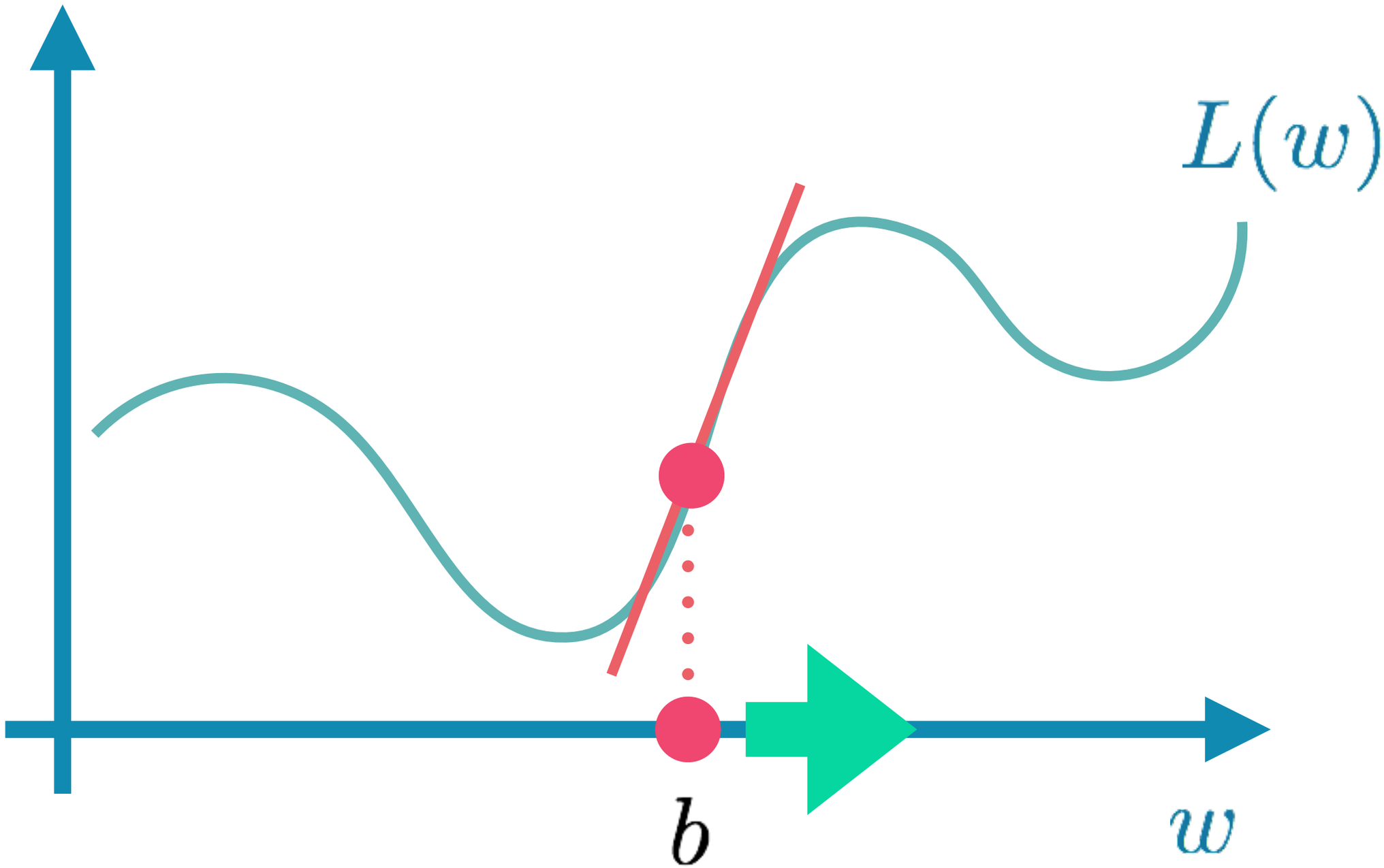
Tangent is the key!



Tangent slope < 0

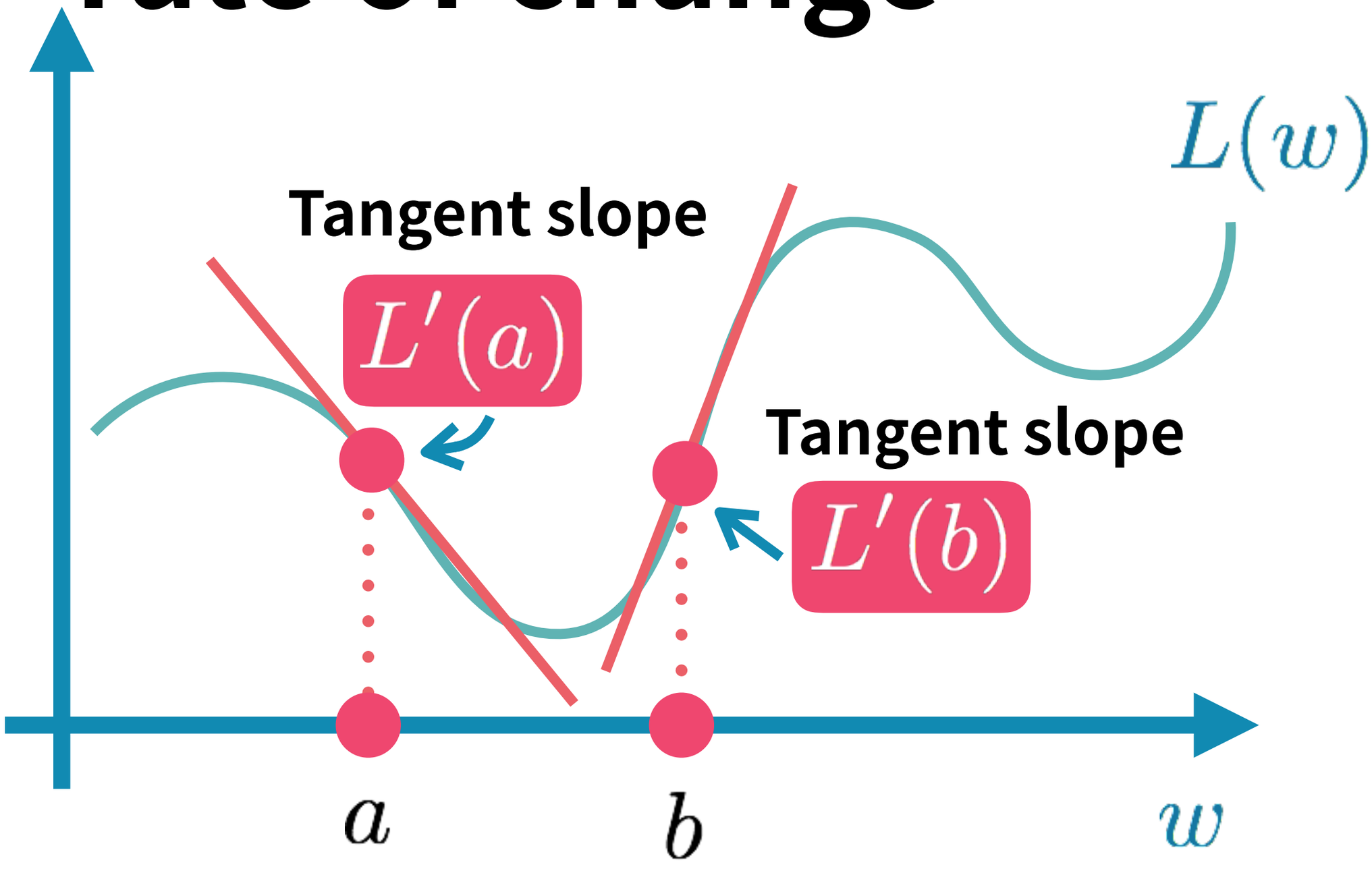


Tangent slope > 0

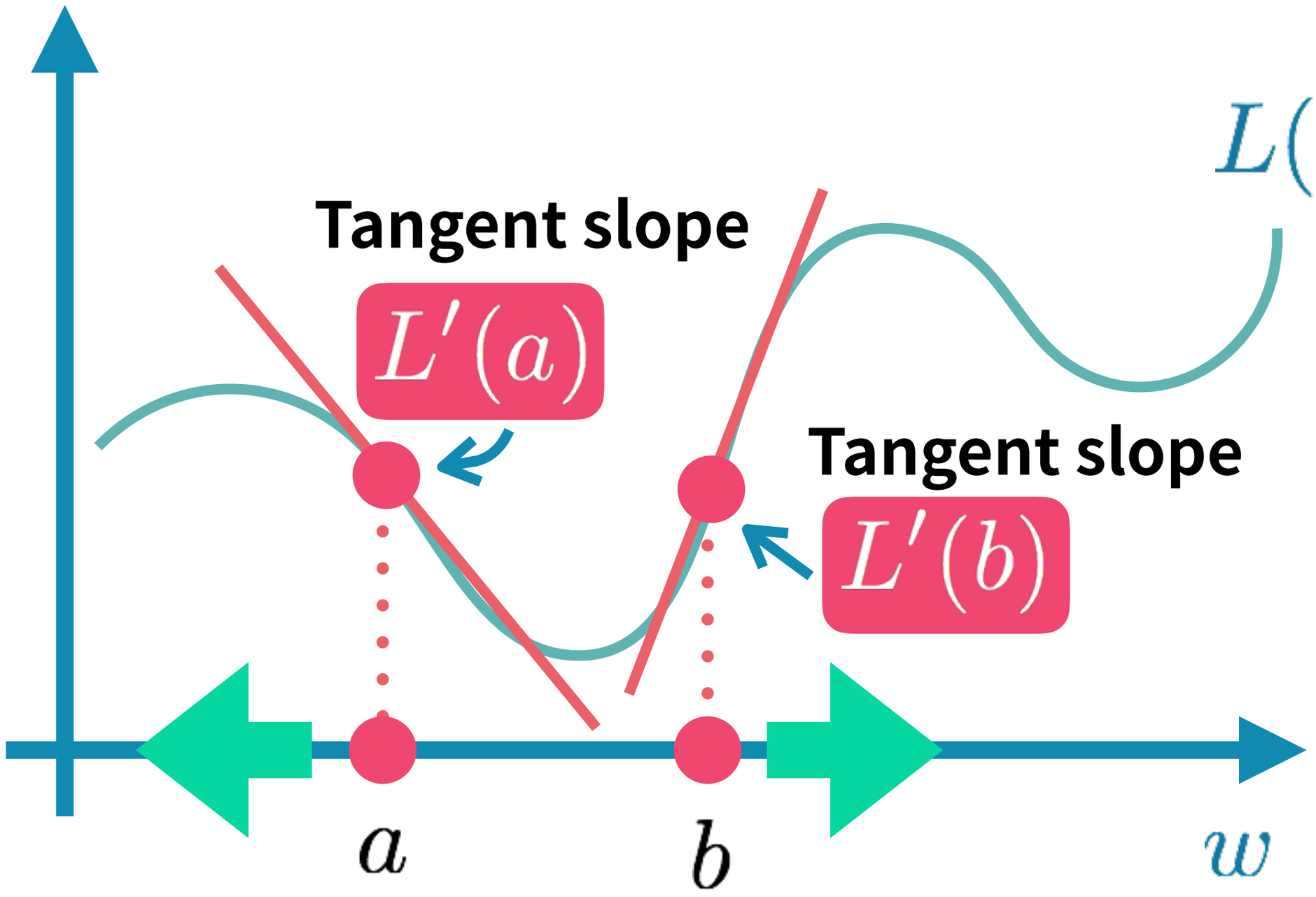


The slope of the tangent points to the (local) maximum direction!

Tangent slope is the rate of change



Pointing (local) maximum



Key

Moving toward (local) minima

In order to make the value of the loss function L smaller, we should adjust the weight w:

$$w - \frac{dL}{dw}$$

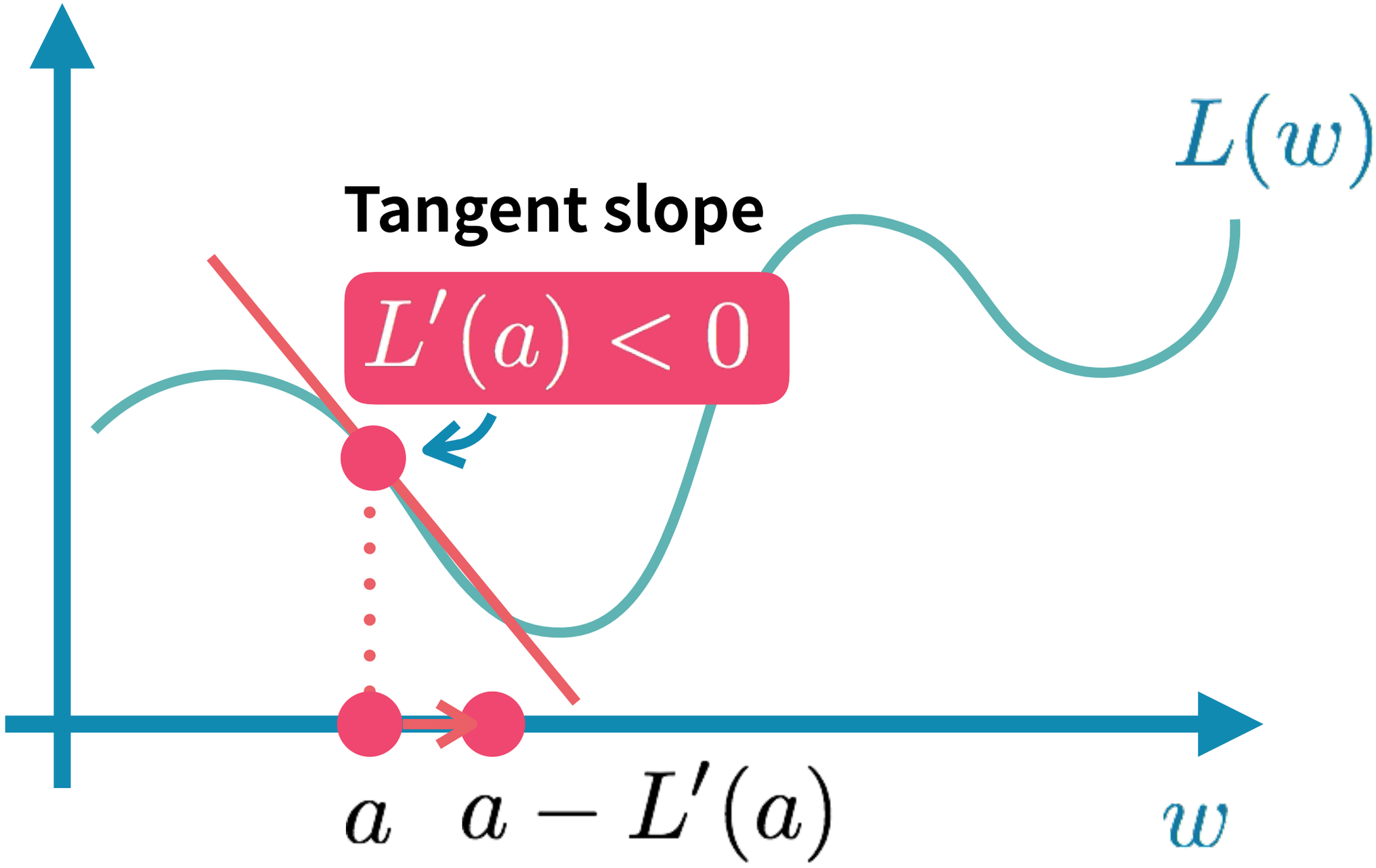
Key

Moving toward (local) minima

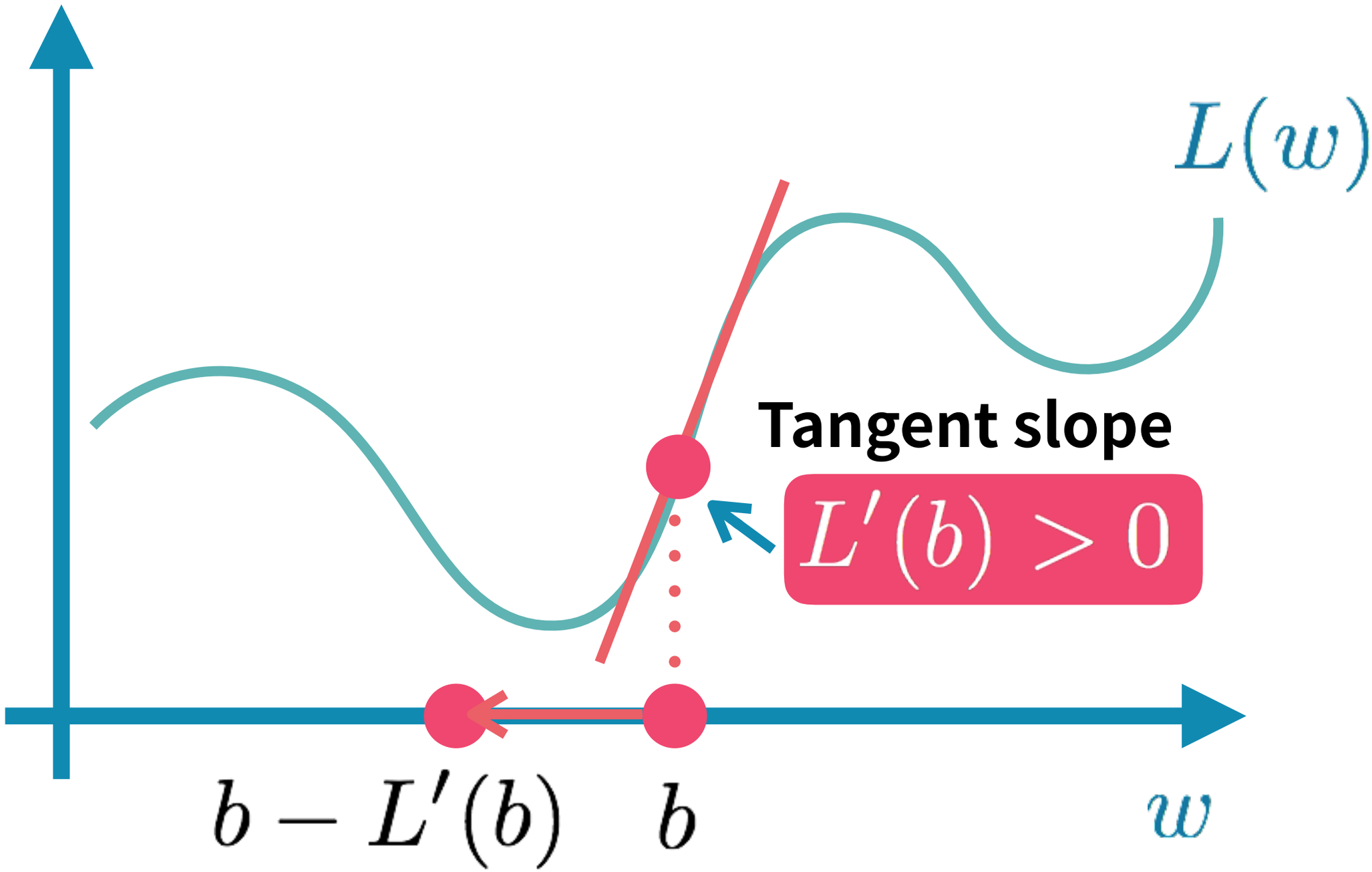
Suppose current w=a, we should update our w as:

$$a - L'(a)$$

Adjusting the Weight



Sometimes it will run over!



Key

Moving toward (local) minima

In order not to make too much adjustment at one time, we will multiply a small number called Leraning Rate:

$$w - \eta \frac{dL}{dw}$$

However, there are more than one parameter...

Example

$$L(w_1, w_2, b_1) = (b_1 + 2w_1 - w_2 - 3)^2$$

Suppose that

$$w_1 = 1,\ w_2 = -1,\ b_1 = 2$$

And we are moving toward to local minimum.

Pretend to have only one parameter!

say, it is w_1

例子

$$L(w_1, w_2, b_1) = (b_1 + 2w_1 - w_2 - 3)^2$$

$$w_1 = 1,\ w_2 = -1,\ b_1 = 2$$

All parameters are fixed, except w_1...

$$L_{w_1}(w_1) = L(w_1, -1, 2) = 4w_1^2$$

One-variable function!

Example

$$L(w_1, w_2, b_1) = (b_1 + 2w_1 - w_2 - 3)^2$$

$$w_1 = 1,\ w_2 = -1,\ b_1 = 2$$

Similarly, we can also define

$$L_{w_2}(w_2) = L(1, w_2, 2) = (-w_2 + 1)^2$$

$$L_{b_1}(b_1) = L(1, -1, b_1) = b_1^2$$

Example

So we can adjust w_1, w_2, b_1, and make L smaller and smaller…

w1 $w_1 - \eta \frac{dL_{w_1}}{dw_1}$

b1 $b_1 - \eta \frac{dL_{b_1}}{db_1}$

w2 $w_2 - \eta \frac{dL_{w_2}}{dw_2}$

Example

Writing together seems more learned!

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} - \eta \begin{bmatrix} \frac{dL_{w_1}}{dw_1} \\ \frac{dL_{w_2}}{dw_2} \\ \frac{dL_{b_1}}{db_1} \end{bmatrix}$$

Updated

We call this the gradient of L

Definition

Partial Derivative

What we have done is just so called partial derivatives.

$$\frac{\partial L}{\partial w_1} = \frac{dL_{w_1}}{dw_1}$$

Definition

Partial Derivative

Similarly,

$$\frac{\partial L}{\partial w_2} = \frac{dL_{w_2}}{dw_2}$$

$$\frac{\partial L}{\partial b_1} = \frac{dL_{b_1}}{db_1}$$

Notation

gradient

Recall the gradient of L is:

$$\begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial b_1} \end{bmatrix} \quad \begin{bmatrix} \frac{dL_{w_1}}{dw_1} \\ \frac{dL_{w_2}}{dw_2} \\ \frac{dL_{b_1}}{b_1} \end{bmatrix}$$

Notation

gradient

We should have cool notation for the gradient:

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

Notation

gradient

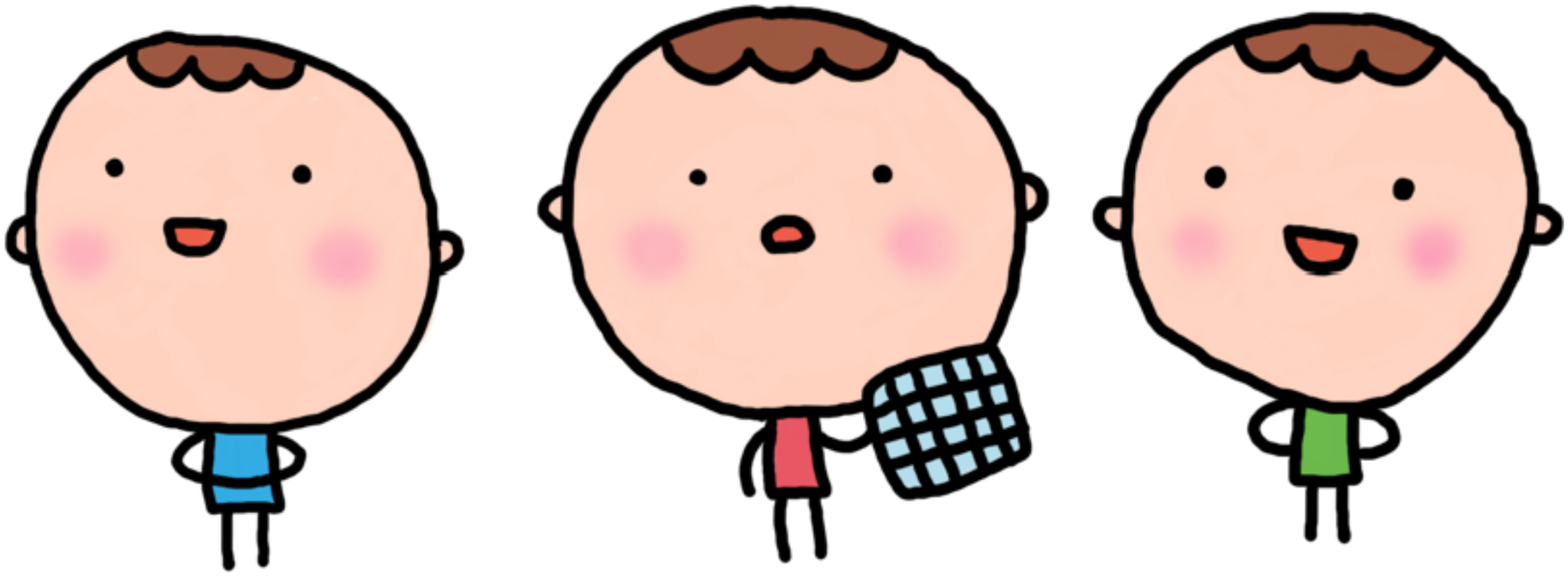
So we can update all parameters by the formula:

$$\begin{bmatrix} w_1 \\ w_2 \\ b_1 \end{bmatrix} - \eta \nabla L$$

This “learning method” has a fancy name

Gradient Descent

Deep Learning
Big 3



Standard NN **CNN** **RNN**

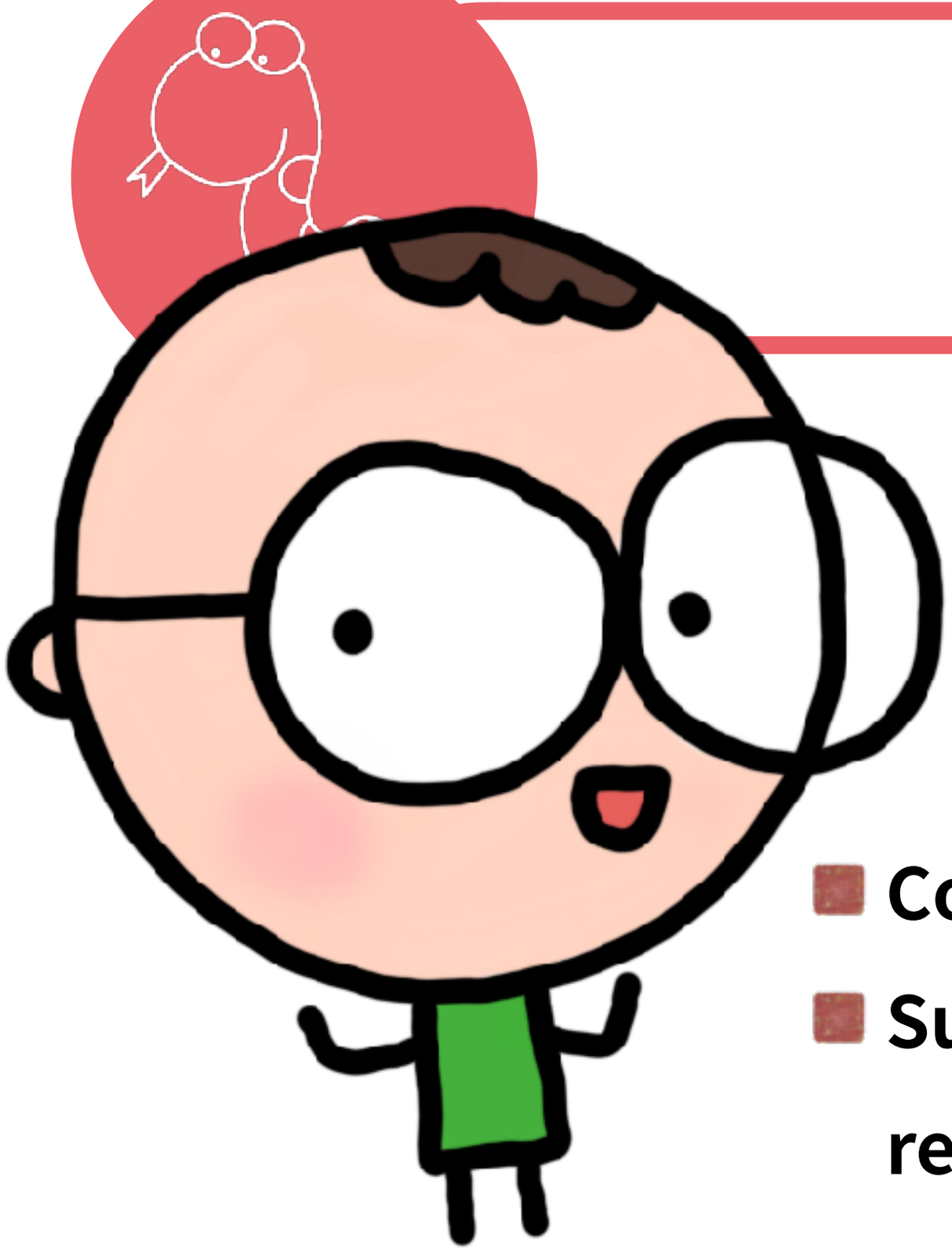
No Kidding!

eder @edersantana · 1天

When you gotta hustle selling neural nets in China. Already profitable AI startup!

6 111 200

CNN



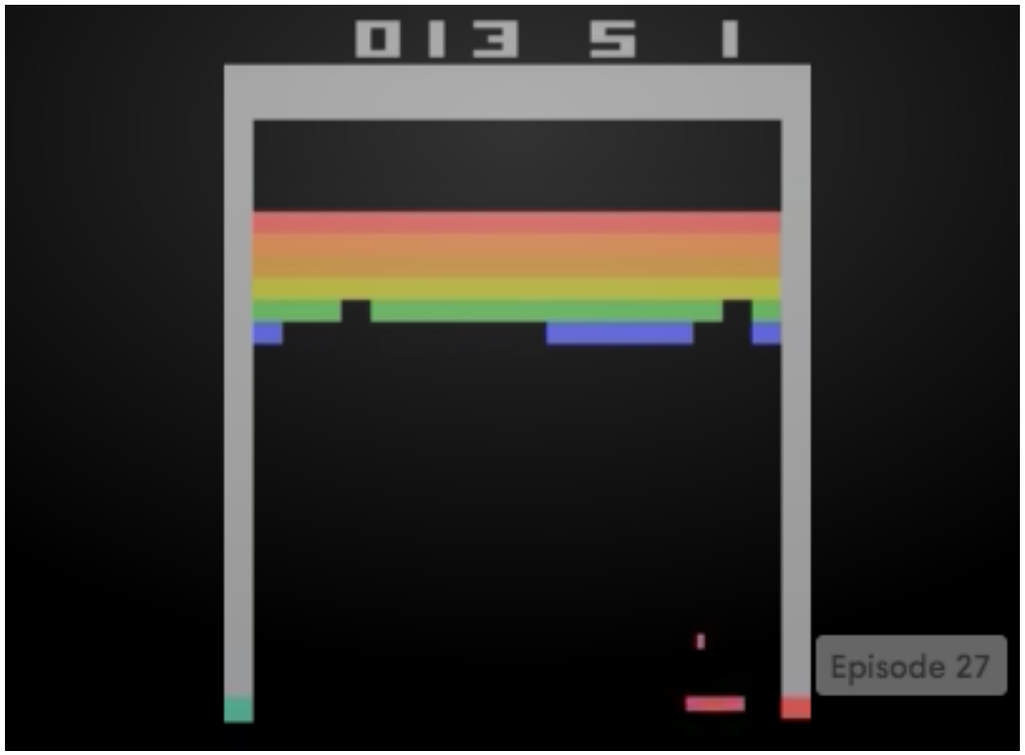
- Convolutional Neural Network
- Super star for image recognition

image recognition

$f($  $)$ = Formosan black bear

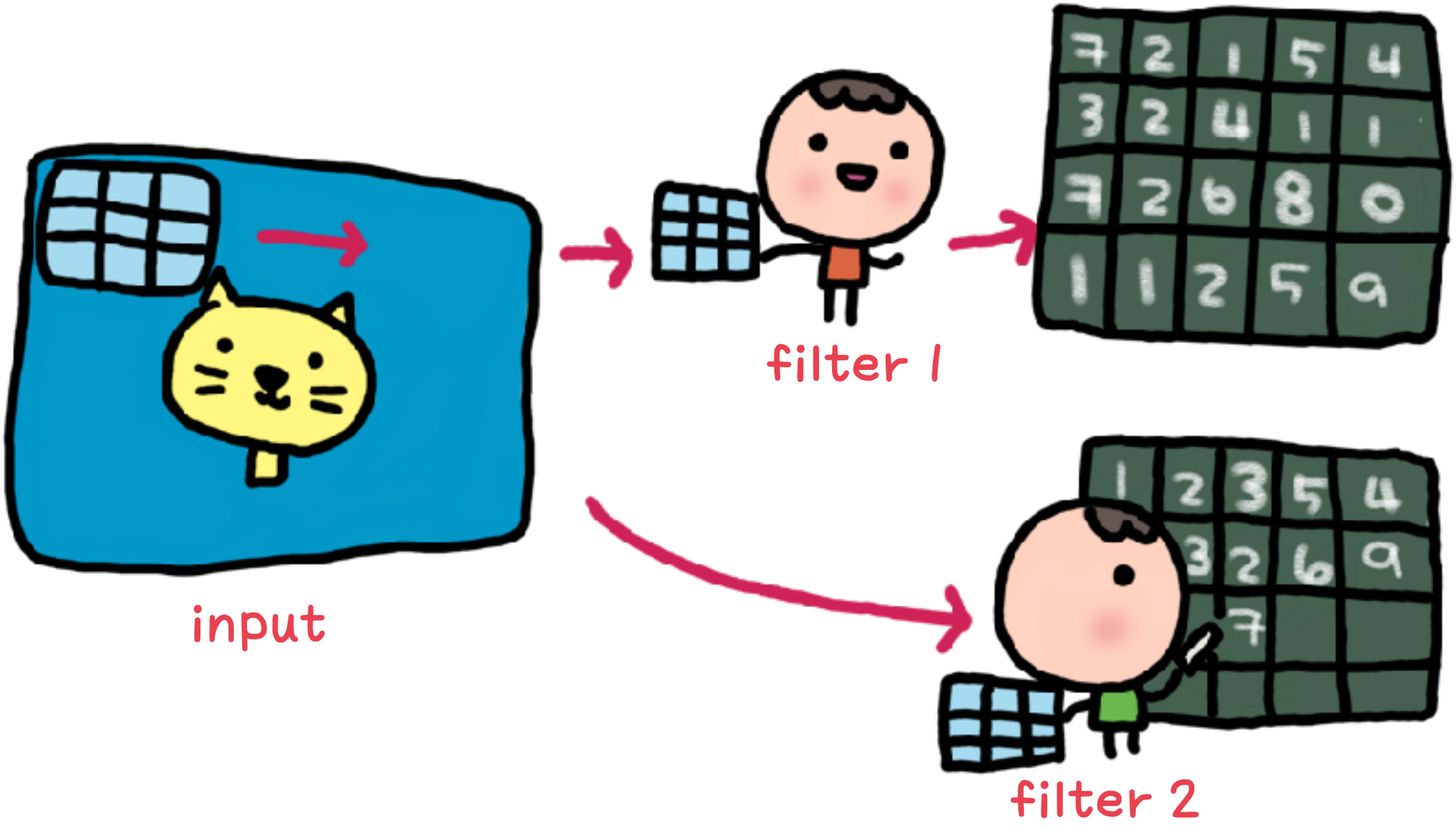
Playing Video Games

reinforcement learning



π (Current states) = “the best” action

Convolutional Neural Network (CNN)



1

Convolutional Layer

We need some “filters”

Say, 3x3 filters

Dot product

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

filter

This is learned.

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35					

Think this is an image.

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

filter

Same matrix as before!

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	3	3	1	5
2	3	4	4	4	3	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

Move to the right

35	**27**				

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	**40**

All the way to the end

Neurons look like this

The points on the picture are one input layer neuron

2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

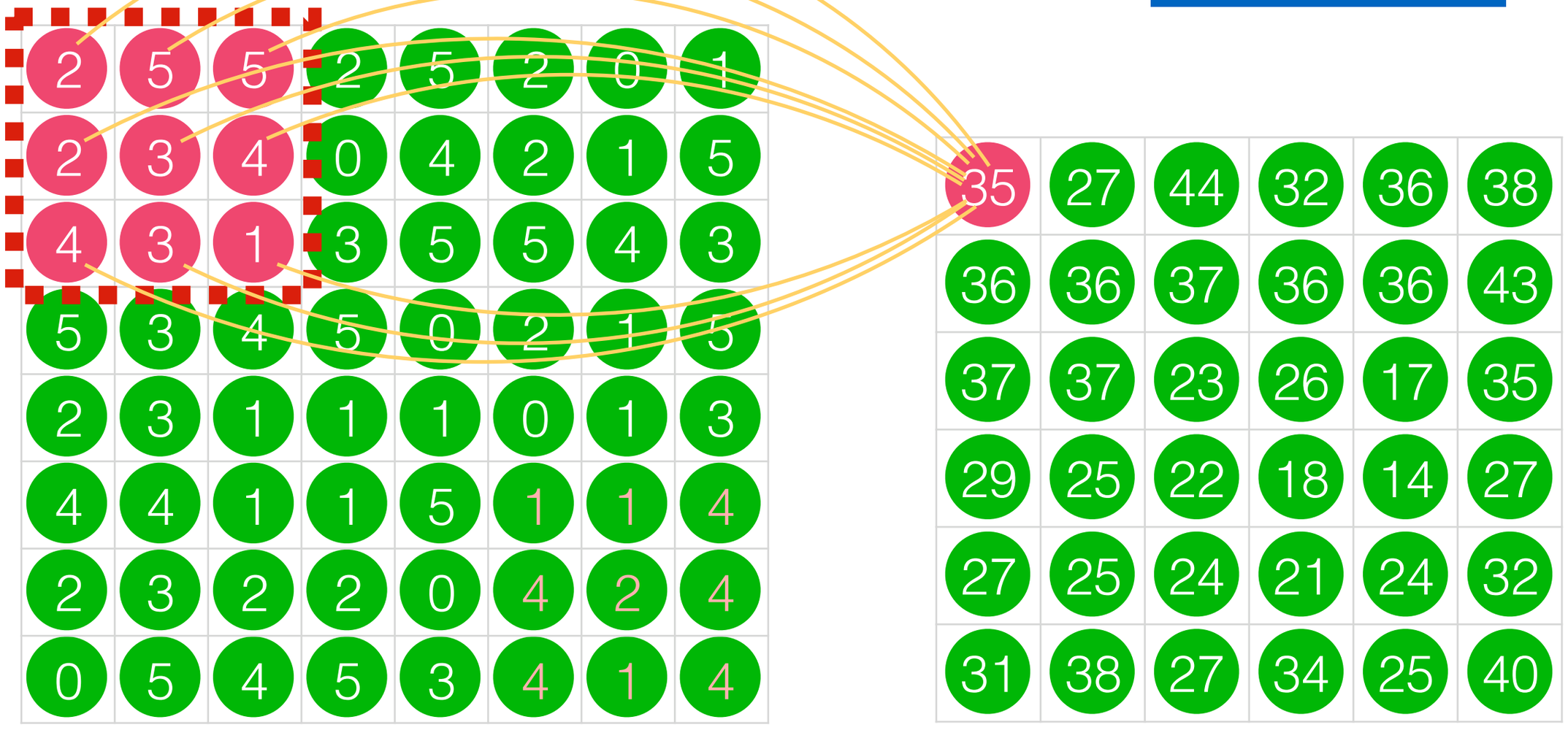
2	5	5	2	5	2	0	1
2	3	4	0	4	2	1	5
4	3	1	3	5	5	4	3
5	3	4	5	0	2	1	5
2	3	1	1	1	0	1	3
4	4	1	1	5	1	1	4
2	3	2	2	0	4	2	4
0	5	4	5	3	4	1	4

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

Conv layers also consist lots of neural

filter

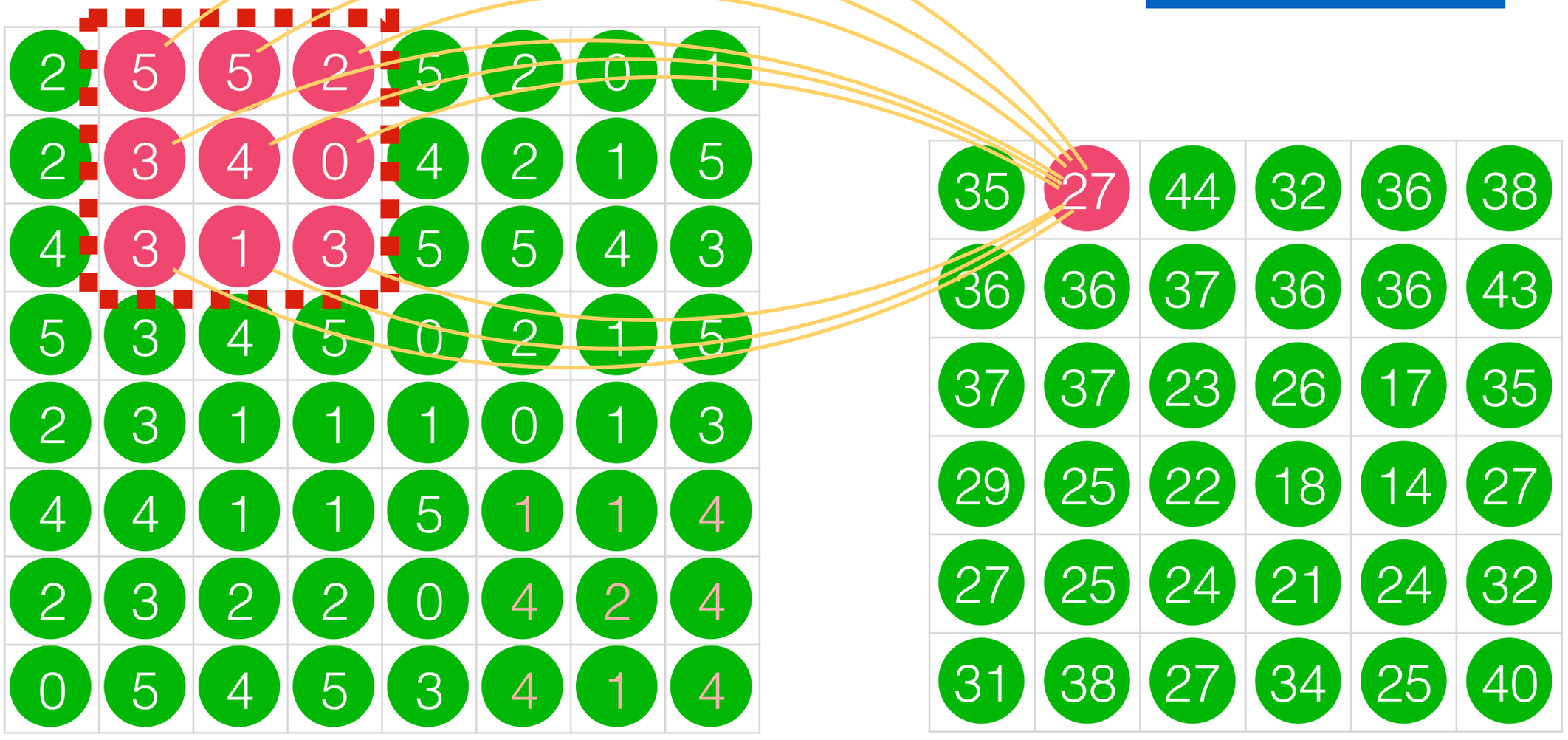
$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$



The two layers are not completely connected

filter

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$



the weights are shared
(same as previous ones)

Finally, we get a 6x6 matrix,
and actually we usually
make it a 8x8 matrix.

And we have a lot of filters!

35	27	44	32	36	38
36	36	37	36	36	43
37	37	23	26	17	35
29	25	22	18	14	27
27	25	24	21	24	32
31	38	27	34	25	40

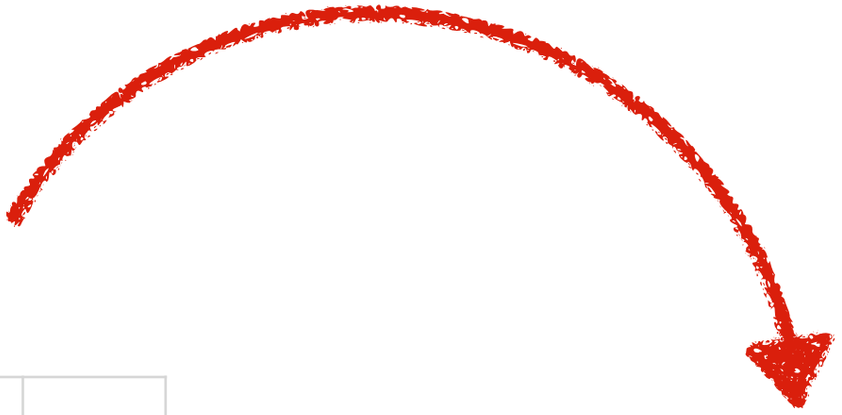
2

Max-Pooling Layer

Basically it is “voting”

We have to decide the size of area to have one representative. Say, 2x2.

35	27	**44**	32	36	38
36	**36**	37	36	36	**43**
37	37	23	**26**	17	**35**
29	25	22	18	14	27
27	25	24	21	24	32
31	**38**	27	**34**	25	**40**



36	**44**	**43**
37	**26**	**35**
38	**34**	**40**

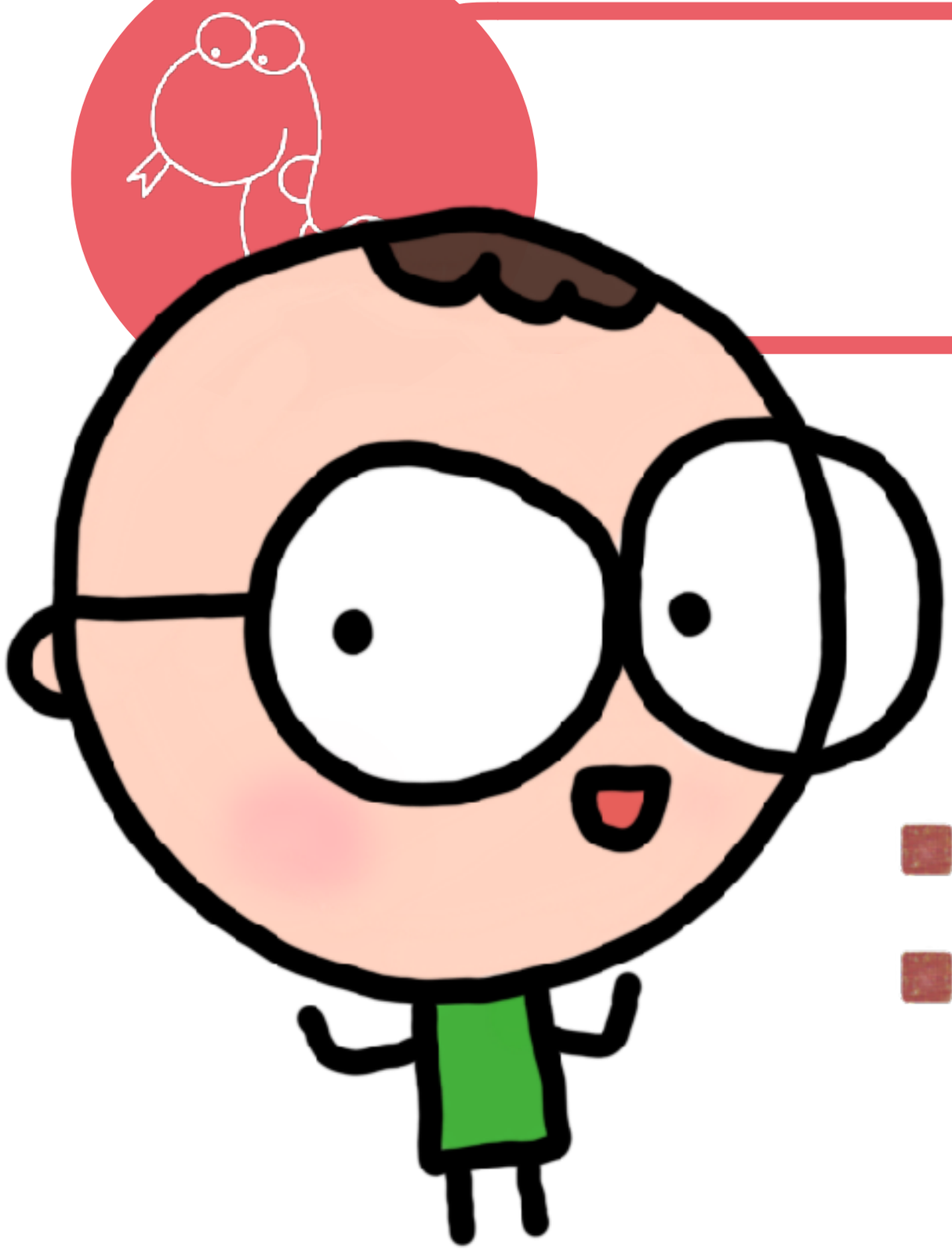
“Vote” for the largest number!!

We can repeat and repeat…

convolution, max-pooling, convolution, max-pooling…

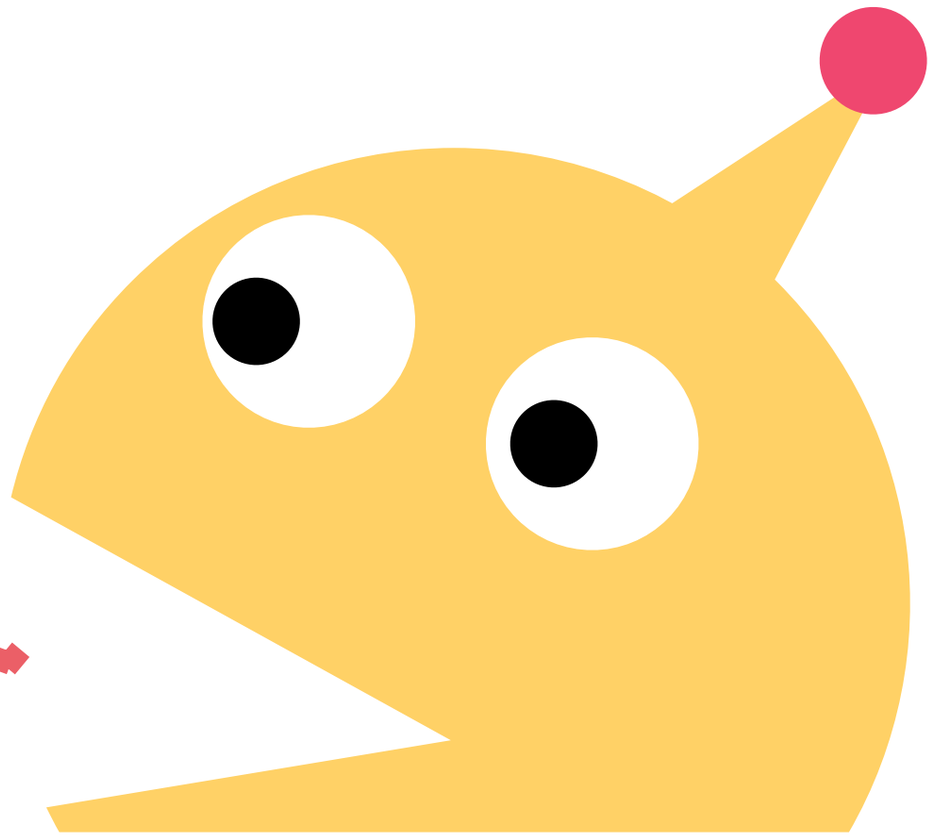
After we finished, we sent outputs to “normal” neural networks.

RNN

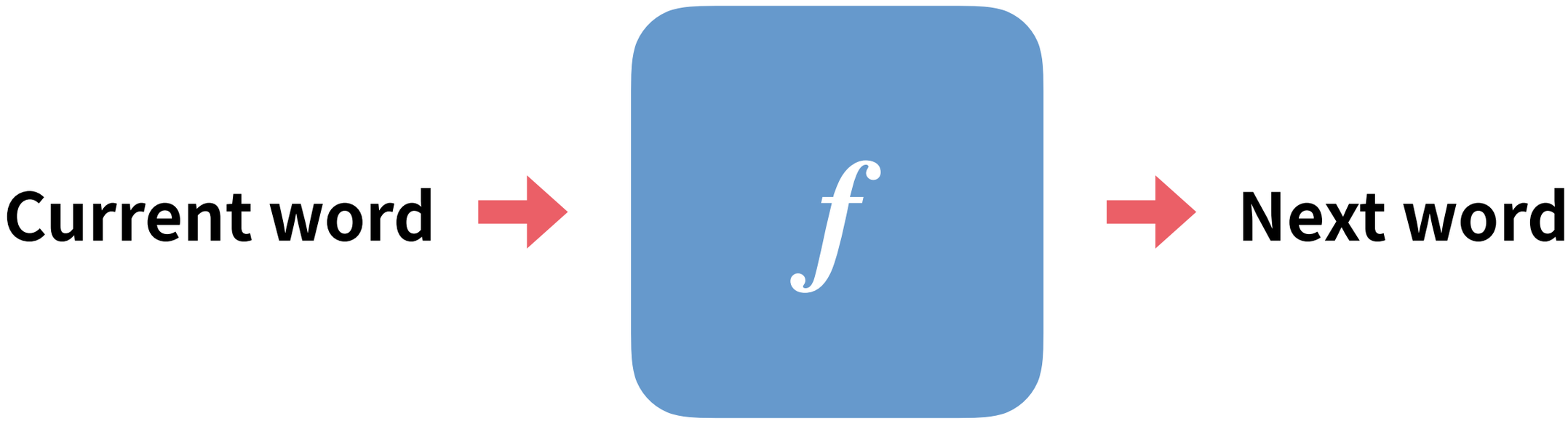


- Recurrent Neural Network
- Neural Netowork with memory

Chatbot

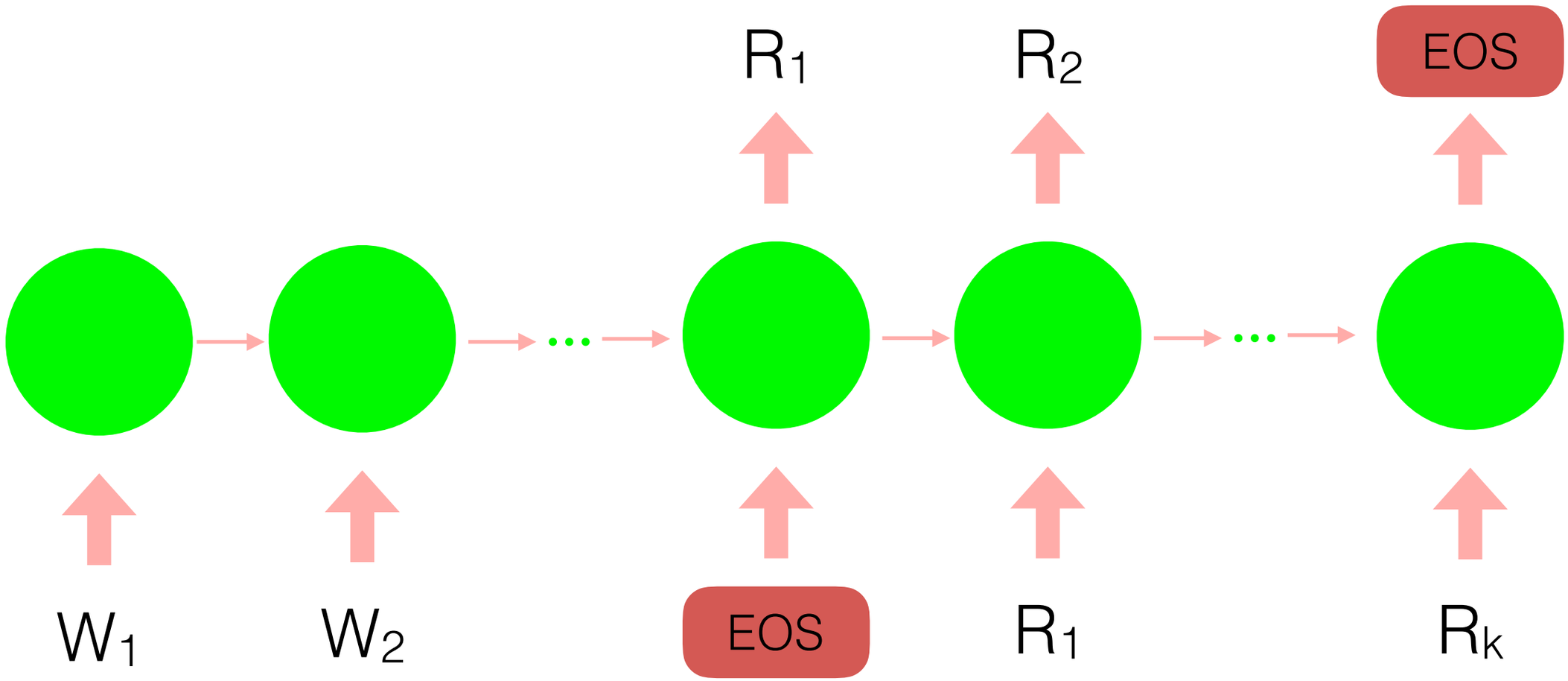


$$f(\text{Current word}) = \text{Next word}$$

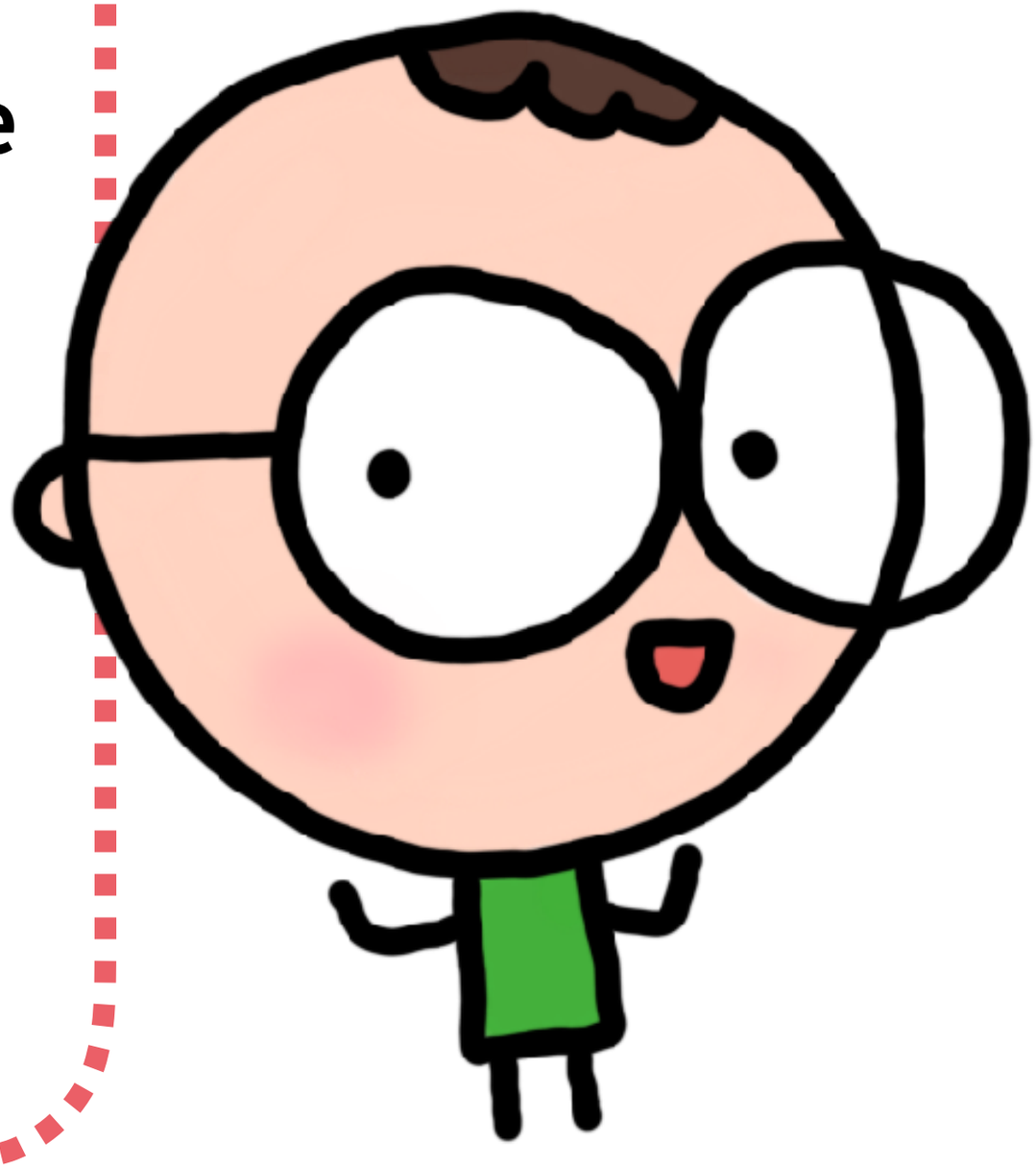

Current word
f
Next word

Application

Chatbot



In fact, the input does not have to be text, but it is also possible that the video (one by one) is output! The output can still be text, and the most common is probably to let the computer say what happened in the video.



Applications

- Translater
- Video Captioning Generator
- Context Generator
- Drawing

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

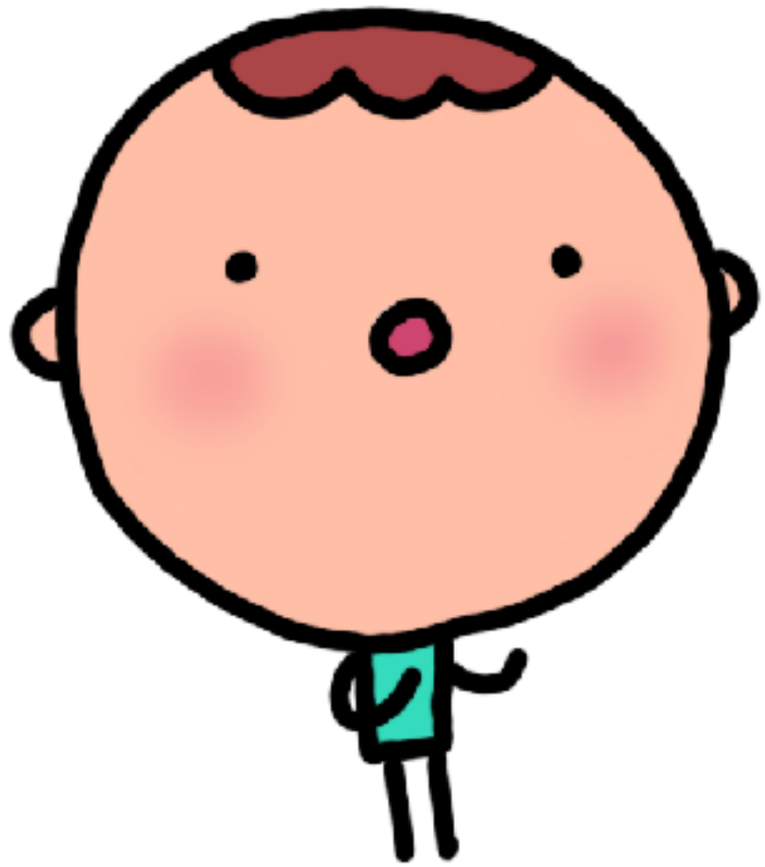
Proof. See discussion of sheaves of sets. □

Andrej Karpathy use RNN to generate a book in "Stacks" (a deep topic in Algebraic Geometry)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

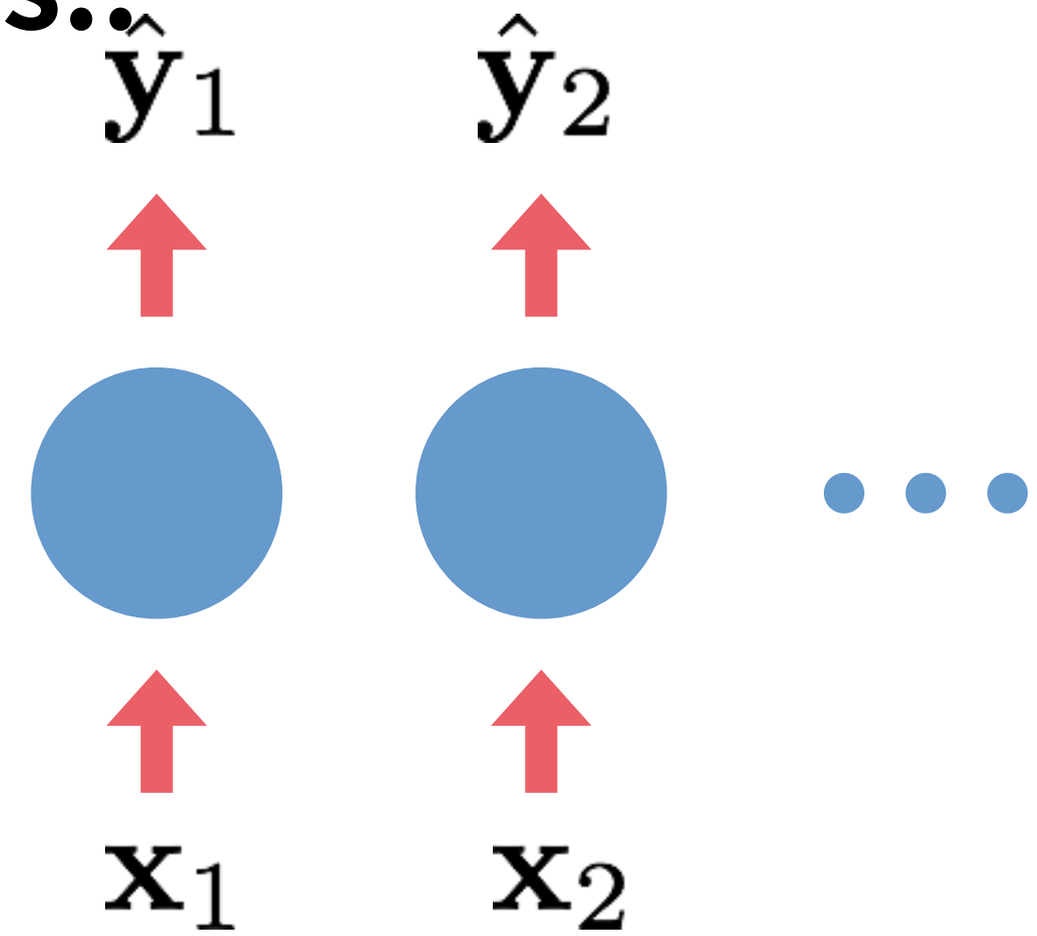
PANDARUS:
Alas, I think he shall be come
approached and the day
When little srain would be attain'd into
being never fed,
And who is but a chain and subjects of
his death,
I should not sleep.

Second Senator:
They are away this miseries, produced
upon my soul,
Breaking and strongly should be
buried, when I perish
The earth and thoughts of many states.

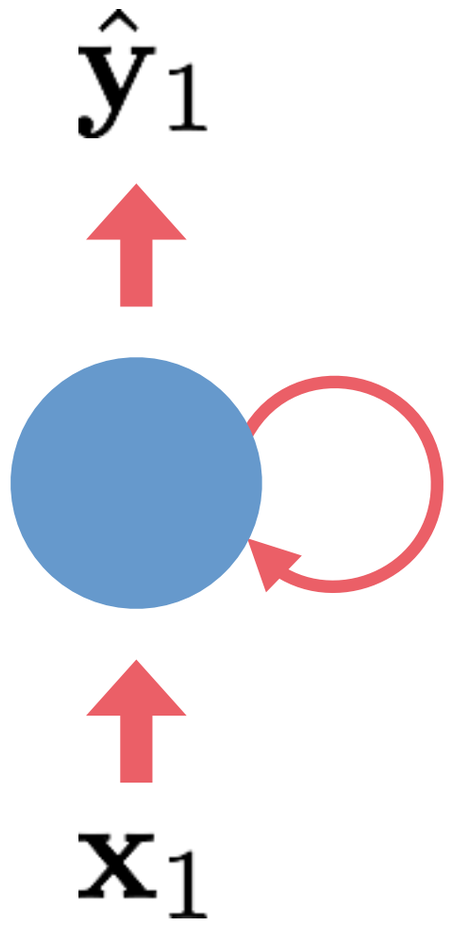


"Shakespeare"

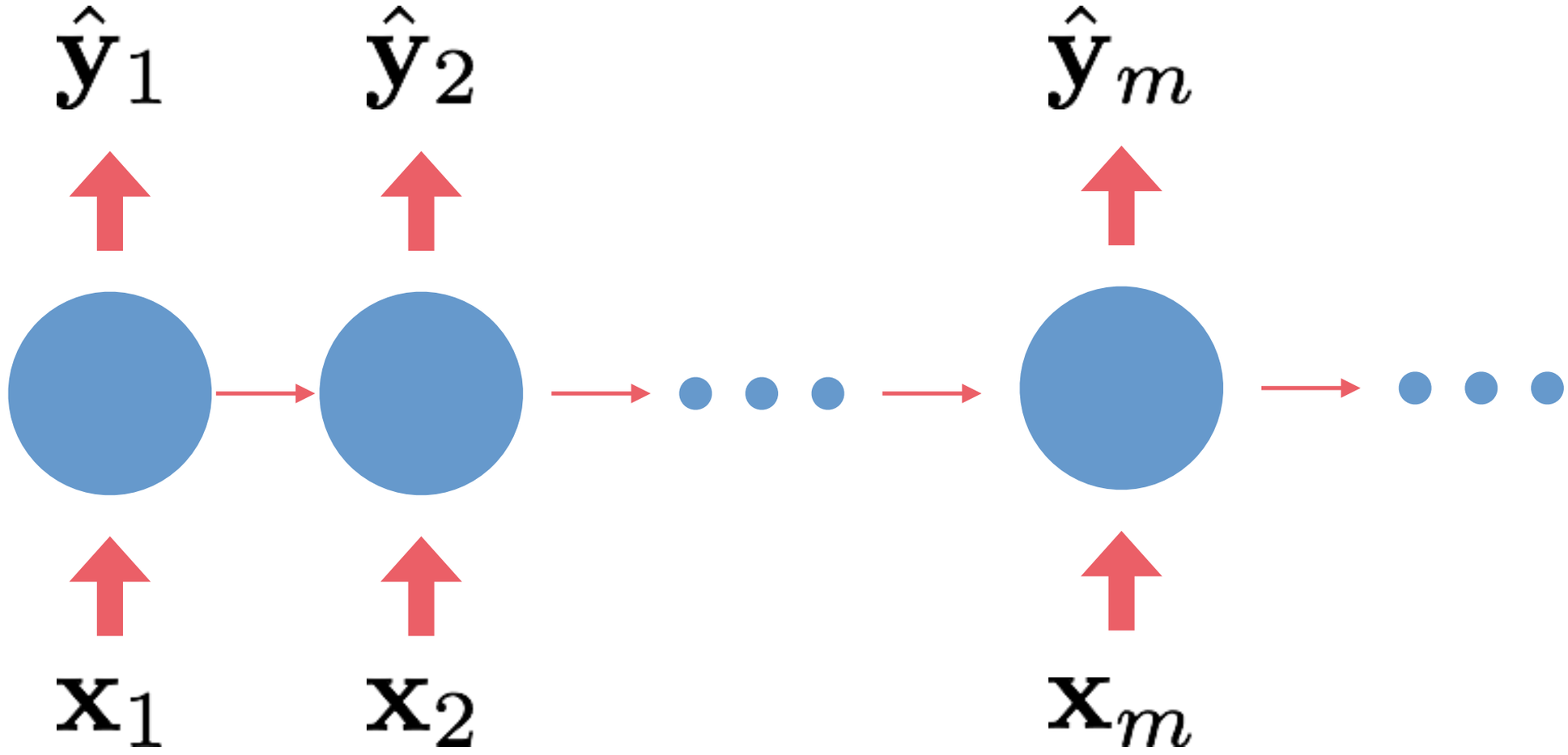
Usually, outputs of a Neural Network are not affected by the input orders..



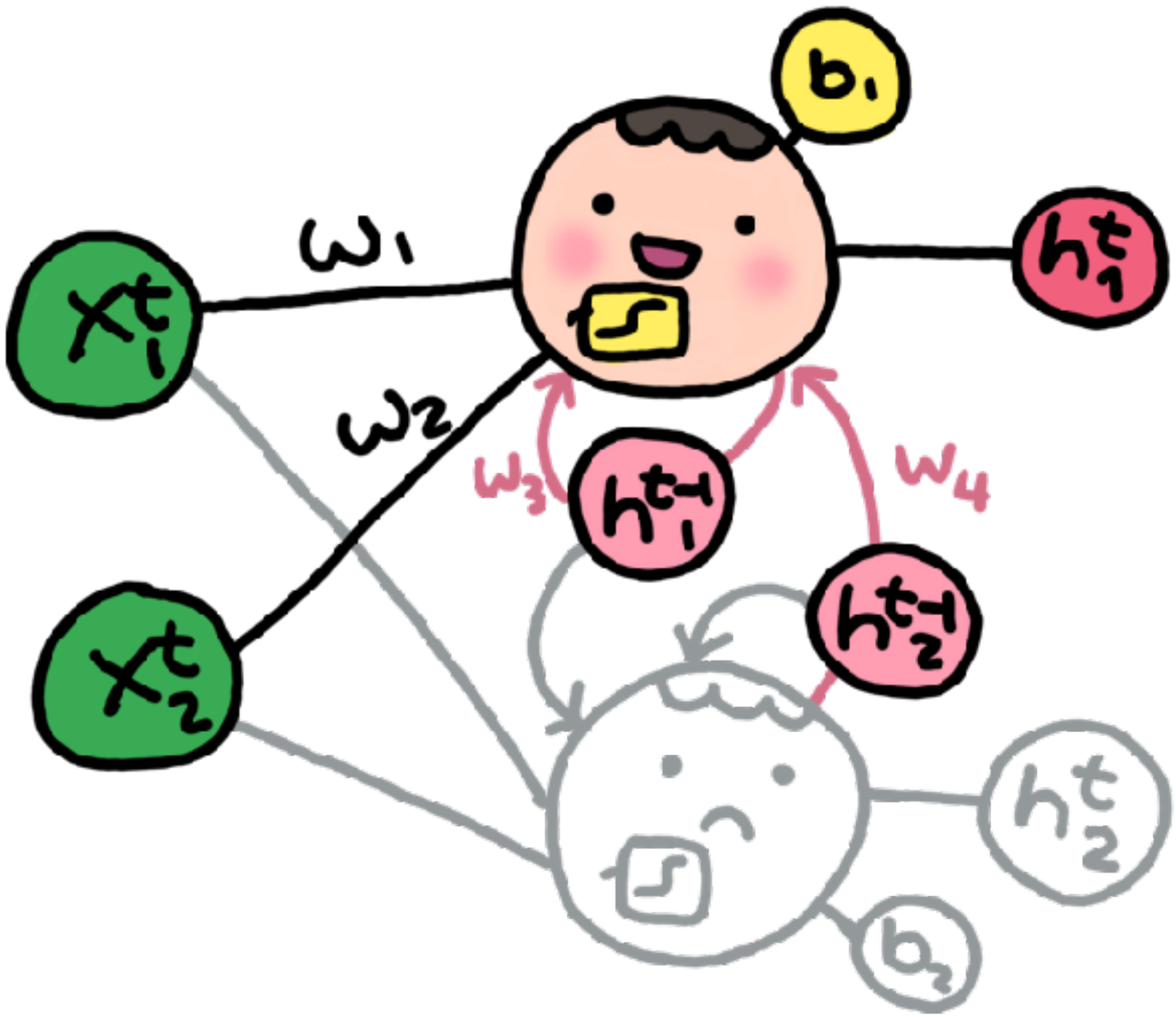
However, RNN cells will use previous outputs as part of inputs…



The “unfold” presentation.



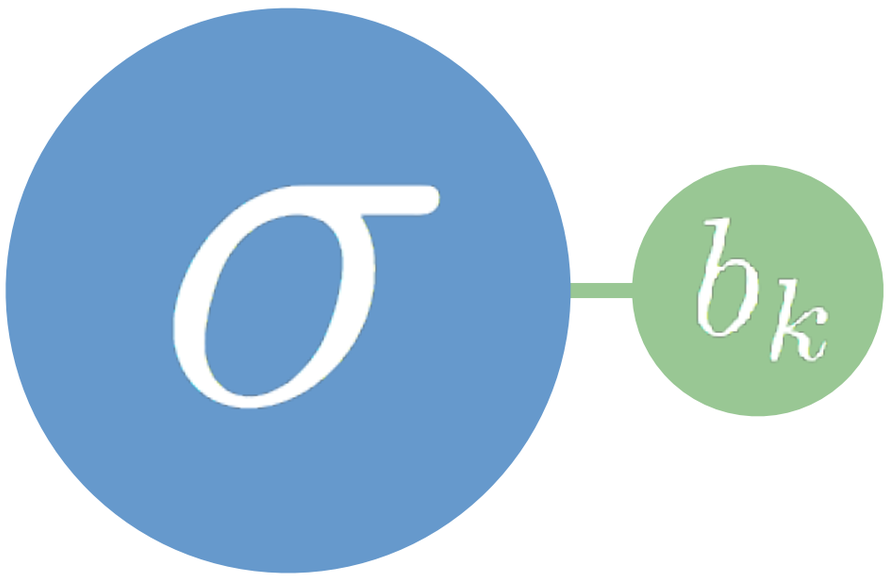
Recurrent Neural Network (RNN)



$$h^t_1 = \sigma(w_1 x^t_1 + w_2 x^t_2 + w_3 h^{t-1}_3 + w_4 h^{t-1}_4 + b_1)$$

Remark

To make it easier for everyone to understand, we will use simpler representations. Please note that the inputs are vectors and will have respected weight; while the outputs are scalars.

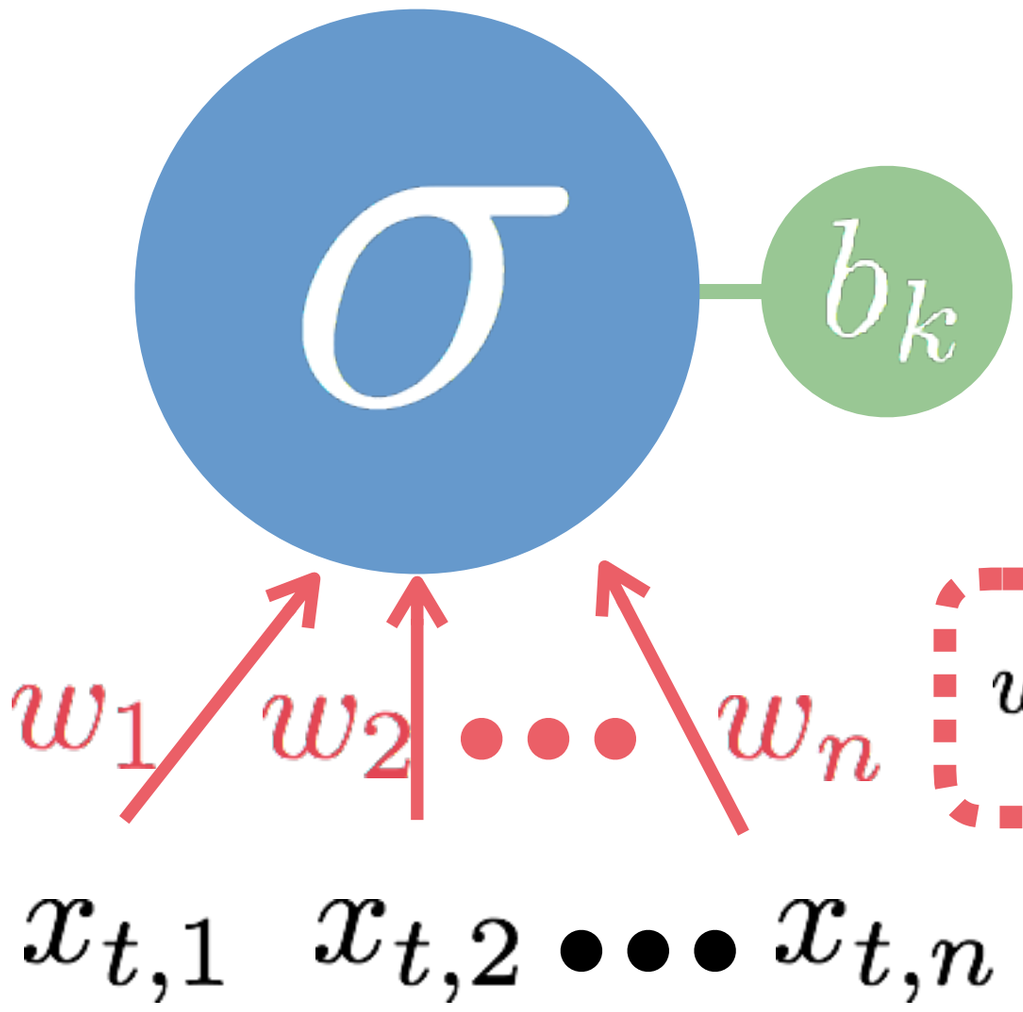


$$\mathbf{x}_t = \begin{bmatrix} x_{t,1} \\ x_{t,2} \\ \vdots \\ x_{t,n} \end{bmatrix}$$

Looks like this

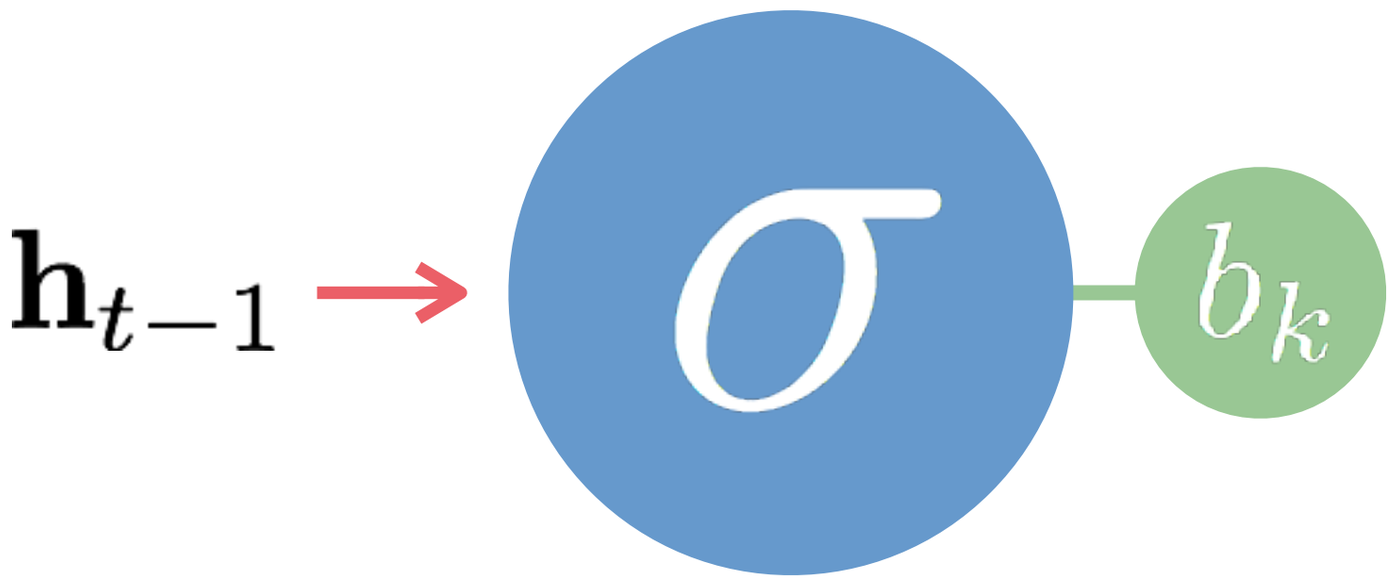
Remark

Actually look like this



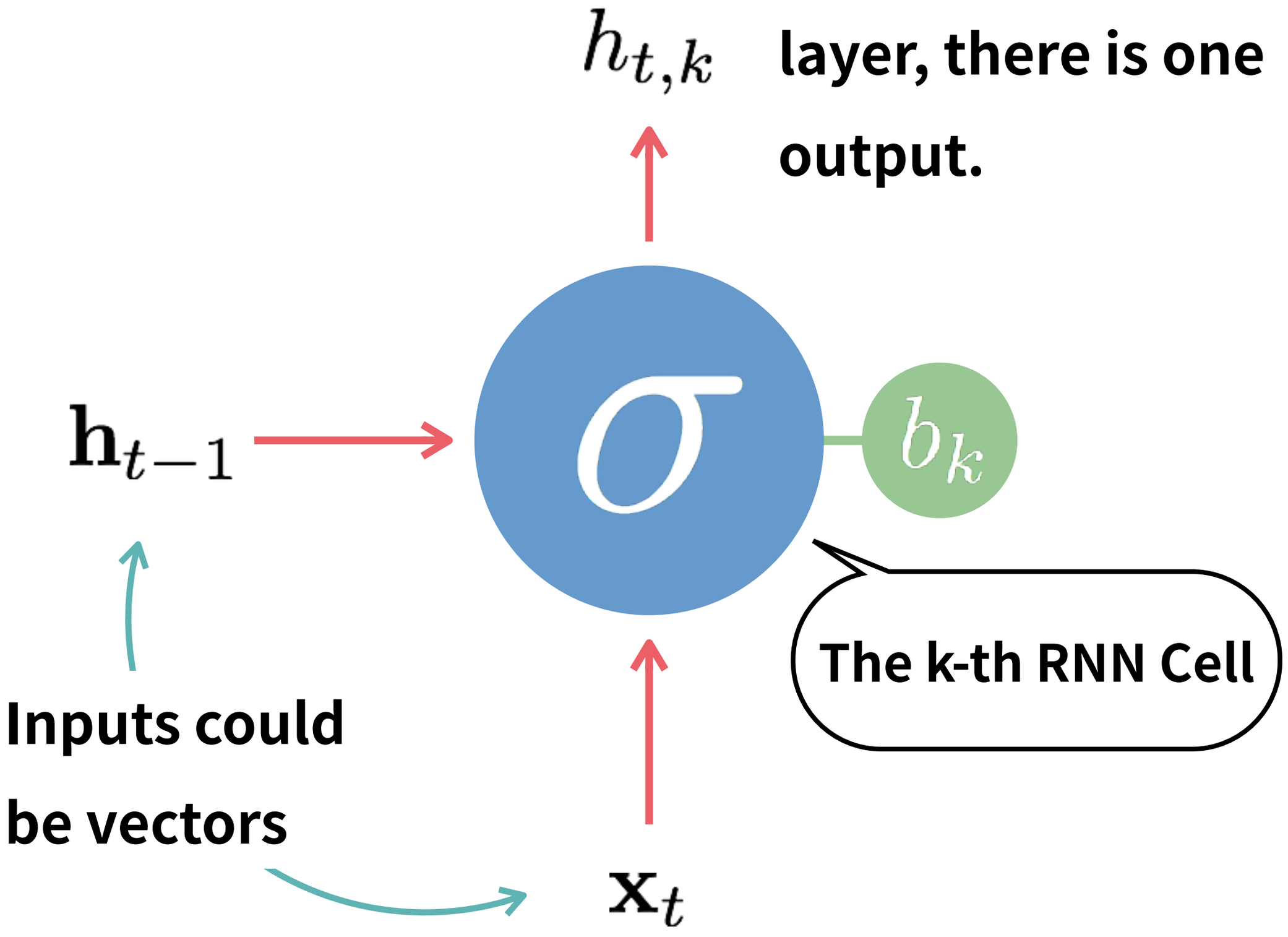
$$w_1 x_{t,1} + w_2 x_{t,2} + \cdots + w_n x_{t,n}$$

Remark

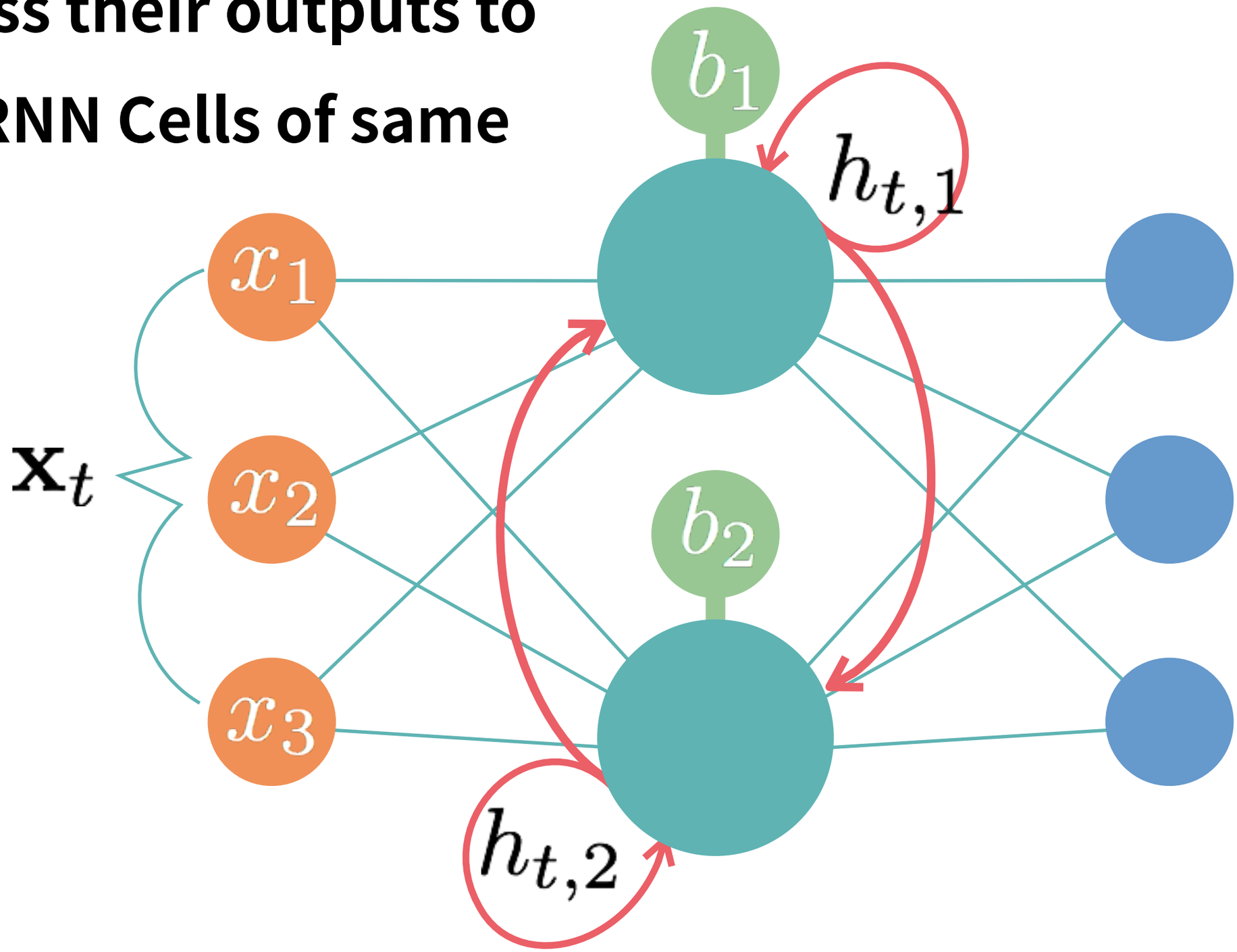


h's work similarly

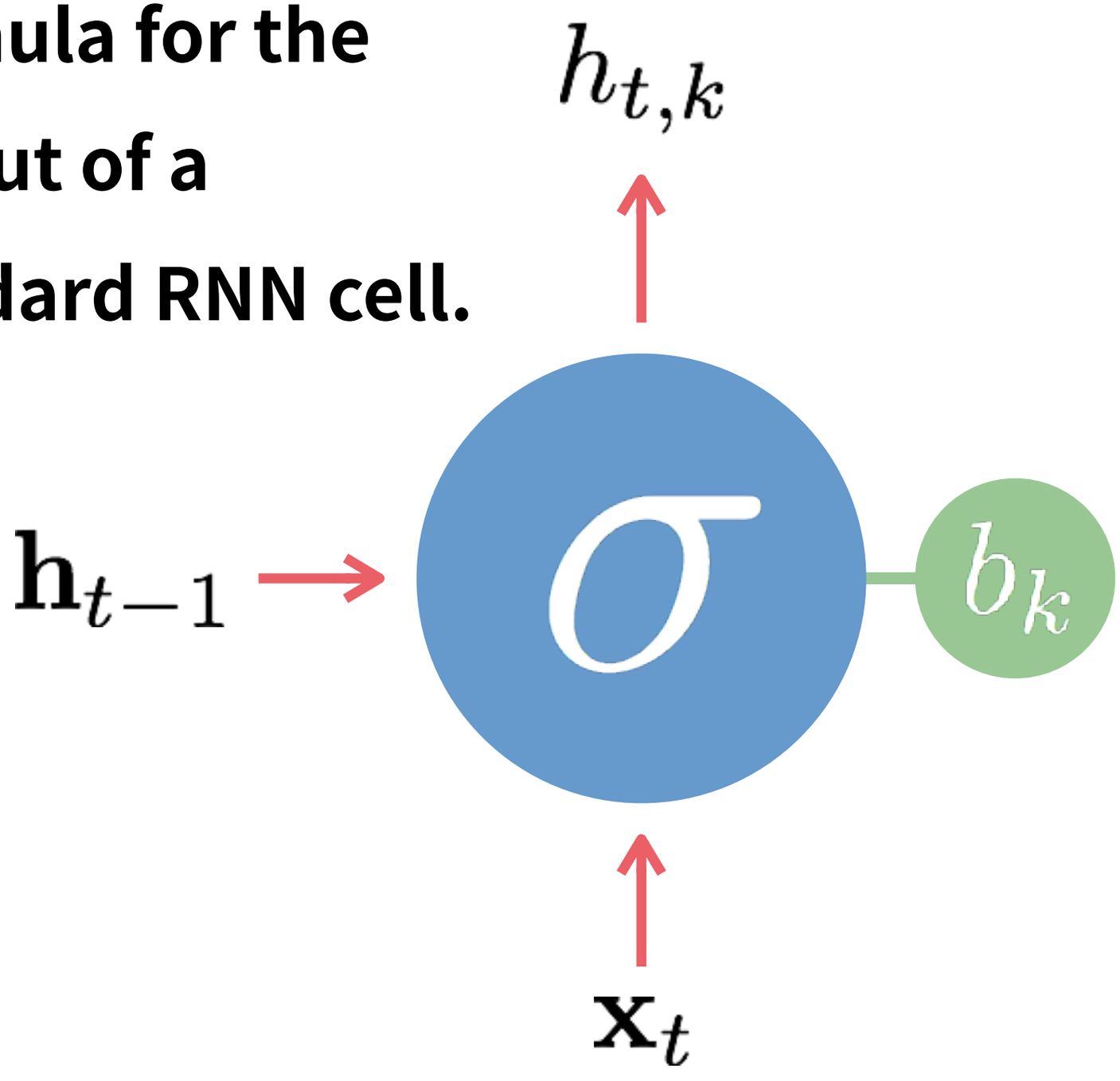
Each cell of some layer, there is one output.



The connections of a RNN layer. Note that RNN cells will pass their outputs to other RNN Cells of same layer.



Formula for the output of a standard RNN cell.



$$h_{t,k} = \sigma(\mathbf{w}_h^T \mathbf{h} + \mathbf{w}_x^T \mathbf{x} + b_k)$$

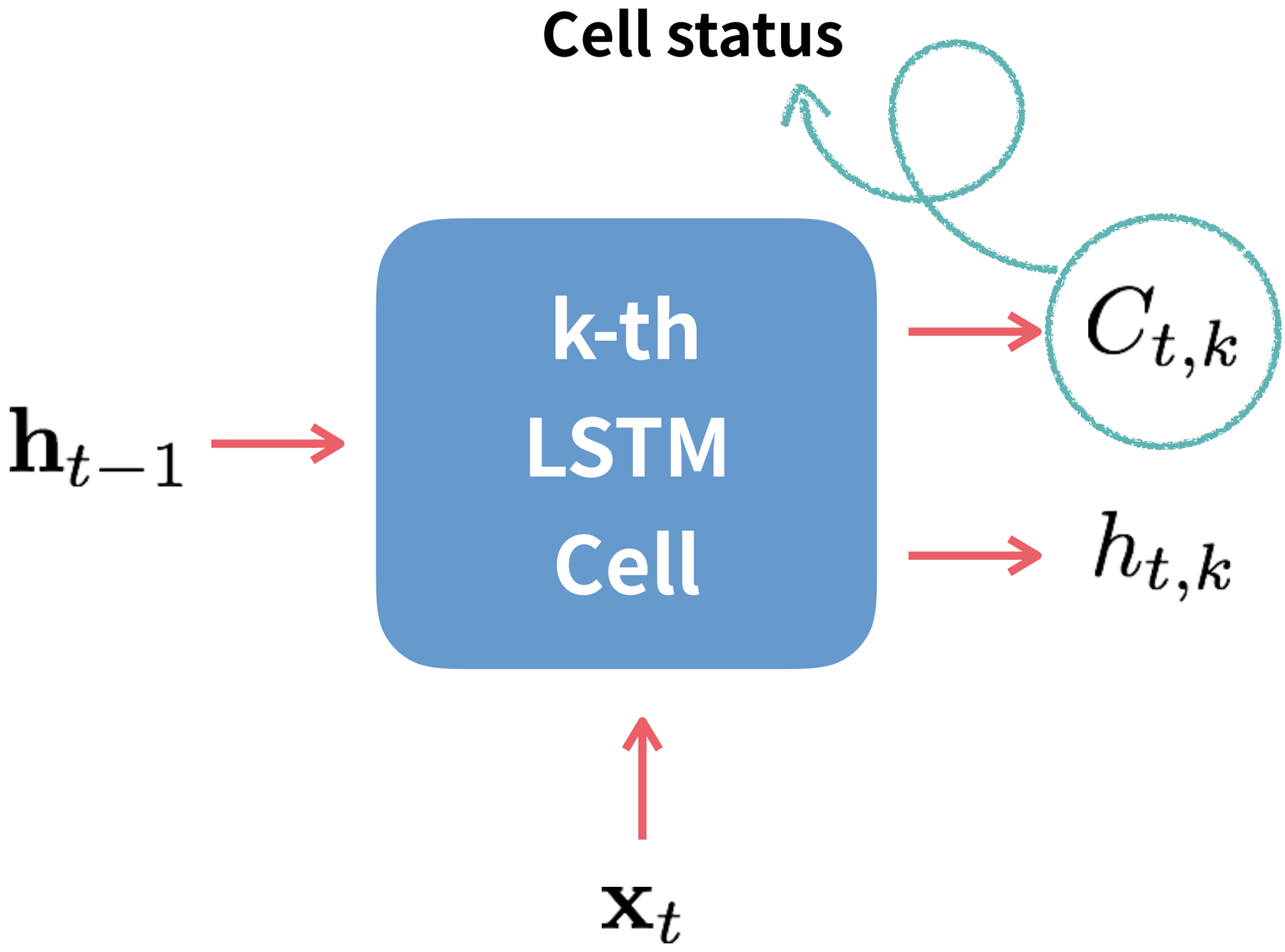
When we say “RNN,” most people actually think of…

Long Short Term Memory

Gated Recurrent Unit

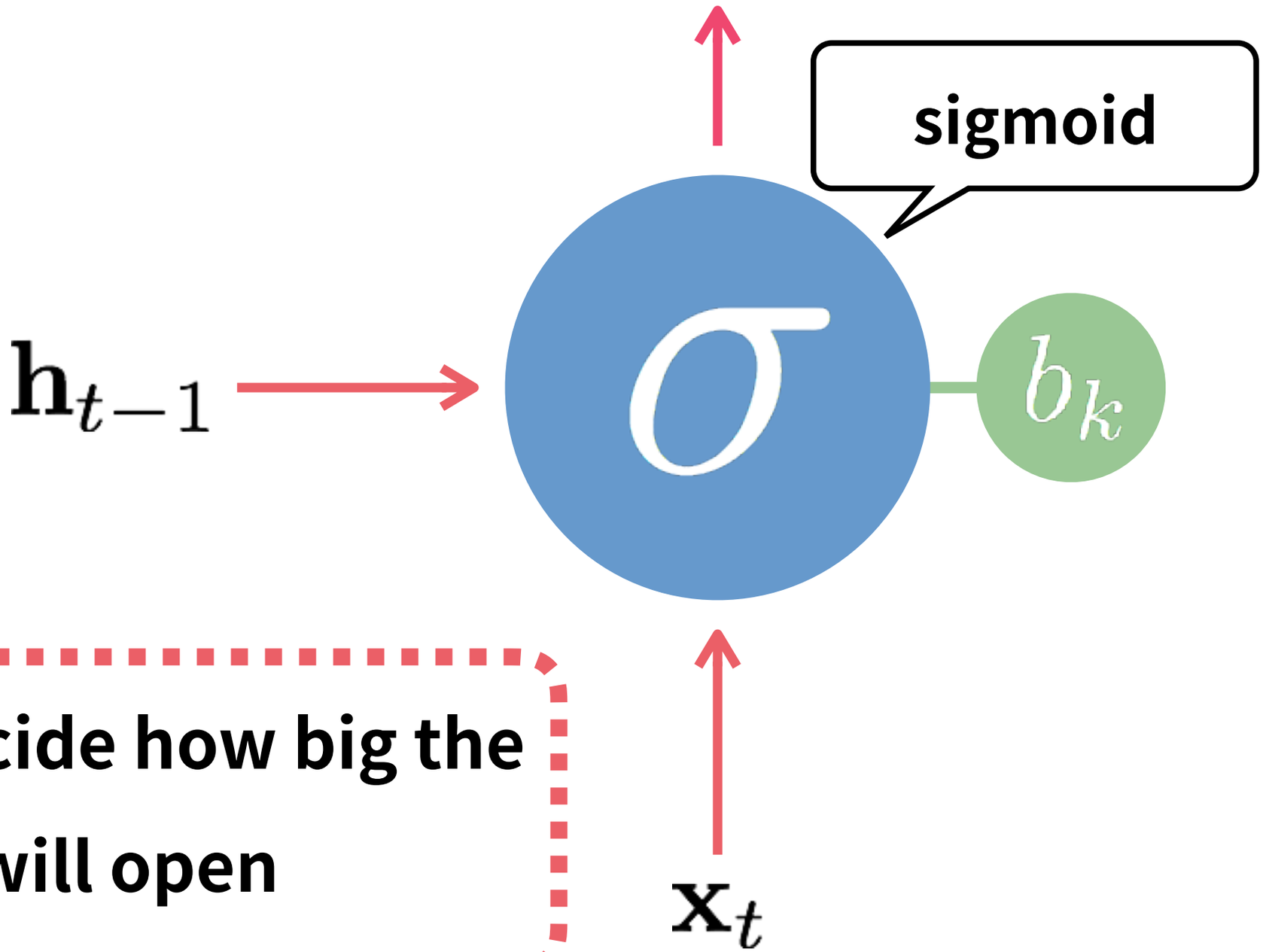
Long Short Term Memory

The Ace of RNNs


Cell status
k-th
LSTM
Cell
$\mathbf{h}_{t-1}$
$C_{t,k}$
$h_{t,k}$
$\mathbf{x}_t$

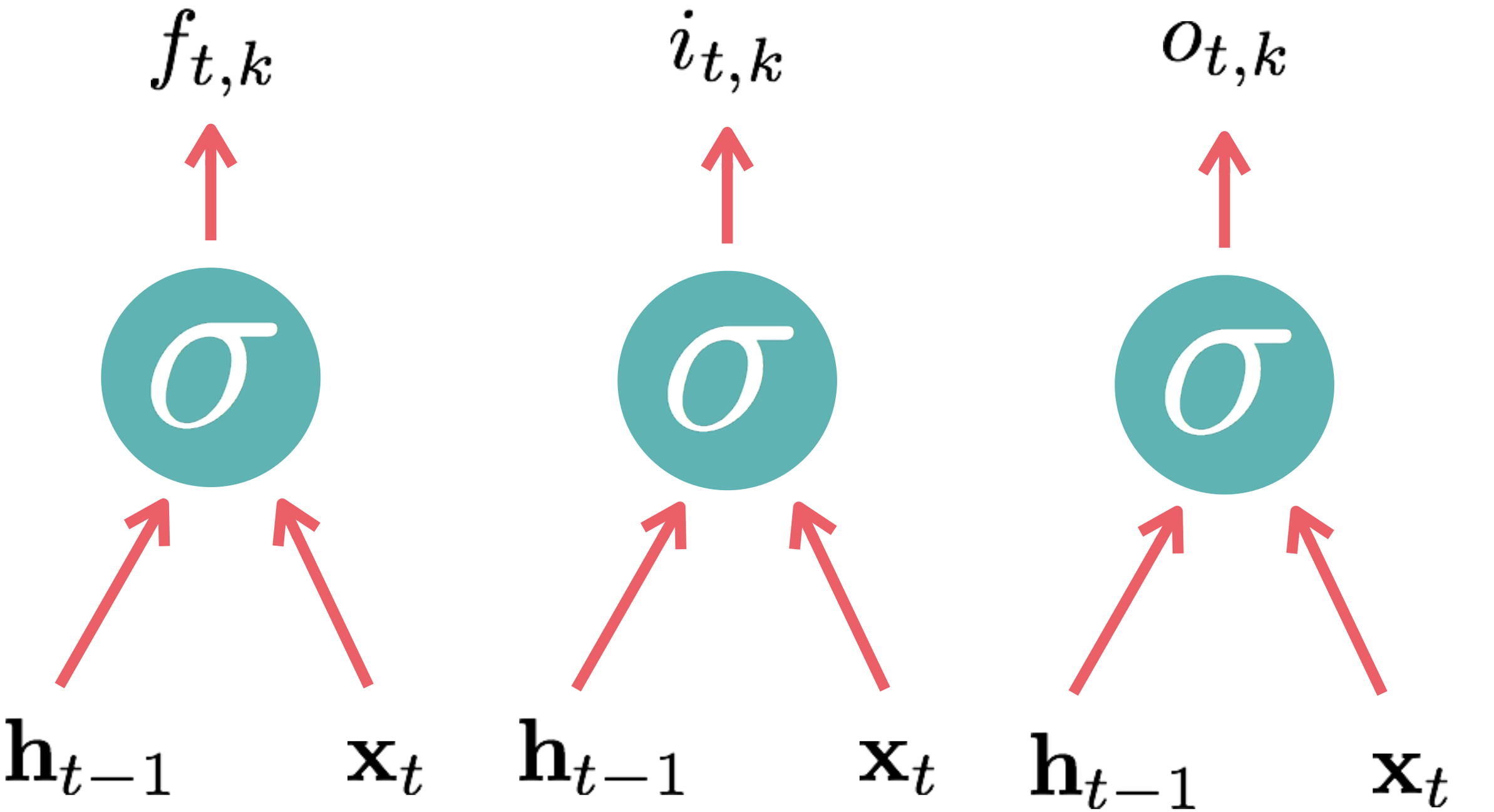
Gate

The output is a number between 0 and 1



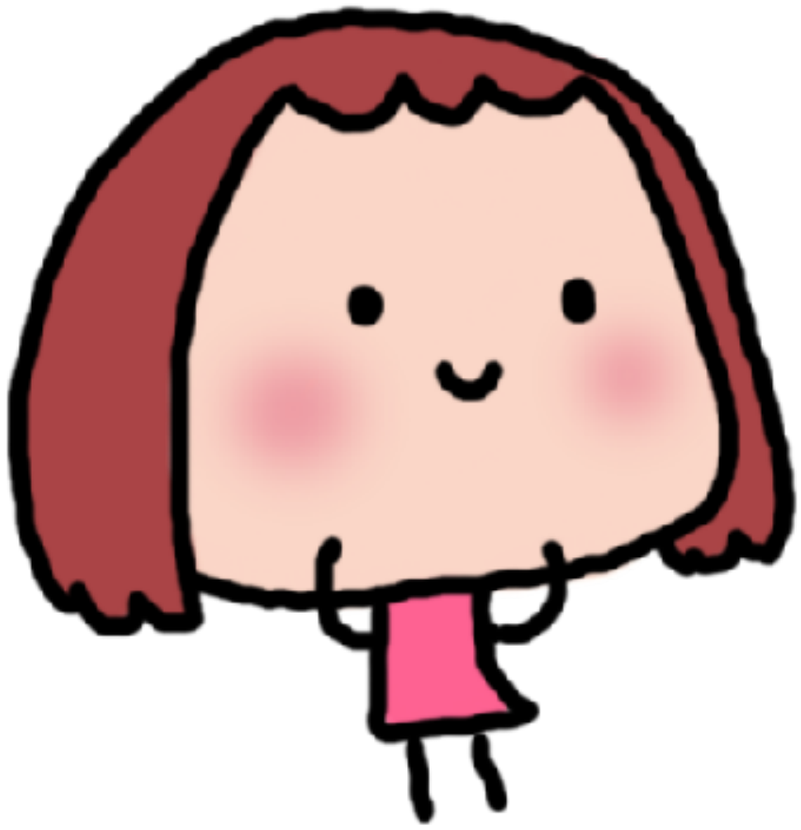
Just decide how big the “gate” will open

LSTMs has three types of Gates



forget gate **input gate** **output gate**

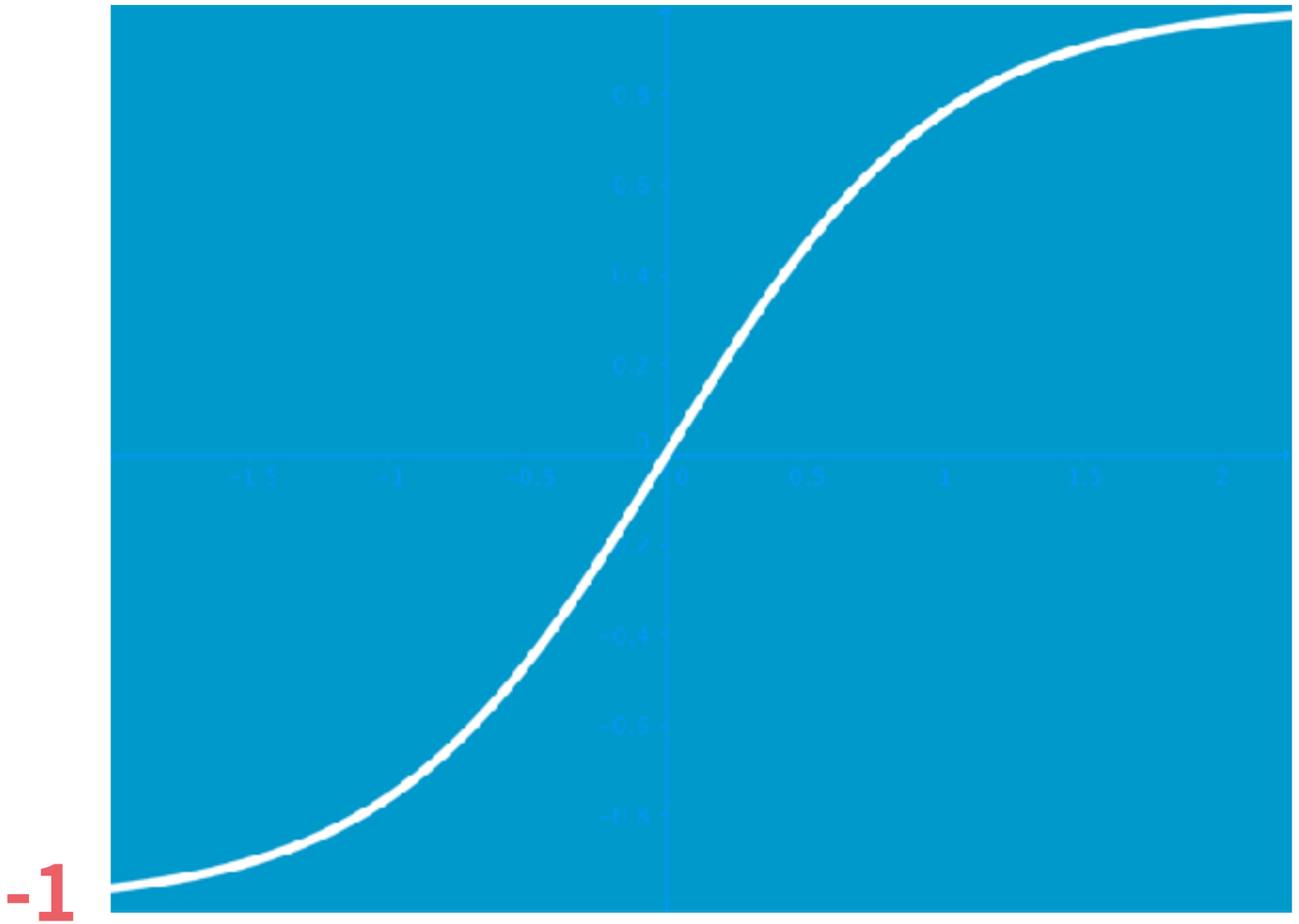
Recall



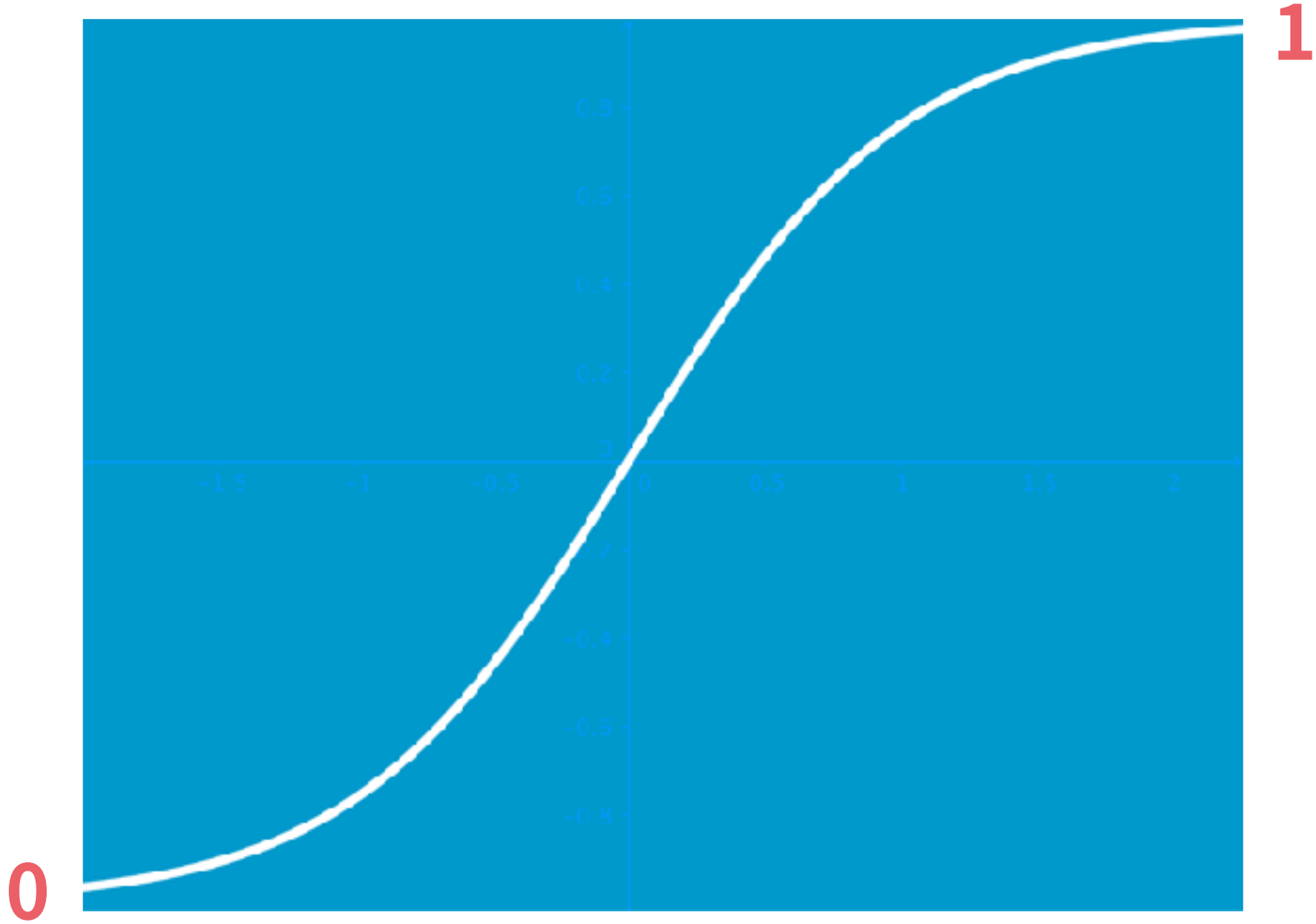
tanh

sigmoid

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

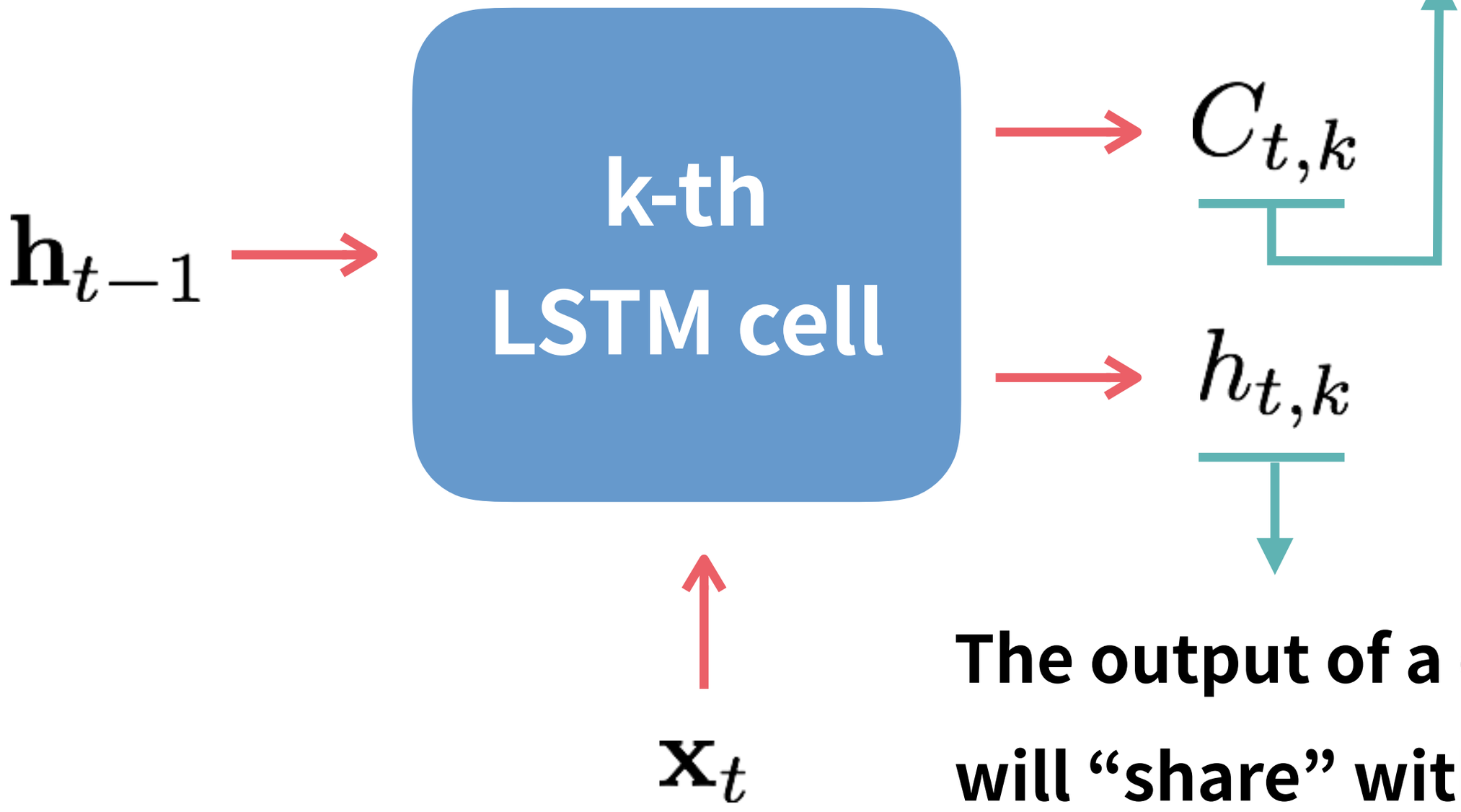


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

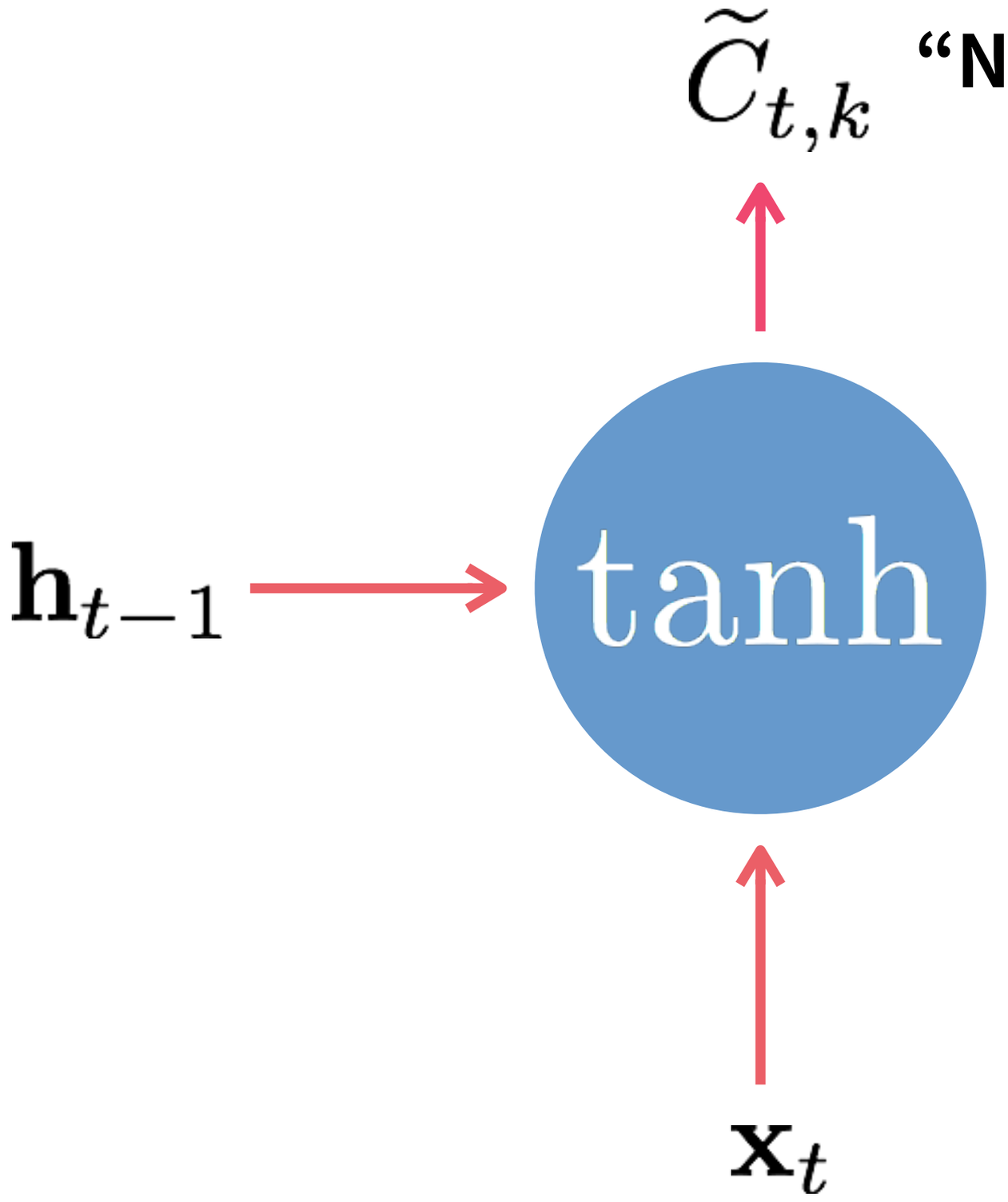


LSTM Again

The cell state is for the cell only



The output of a cell will “share” with other cells


$\tilde{C}_{t,k}$
"New" cell status
$\mathbf{h}_{t-1}$
tanh
$\mathbf{x}_t$

$$C_{t,k} = f_{t,k} C_{t,k-1} + i_{k,t} \tilde{C}_{t,k}$$

$$h_{t,k} = o_{t,k} \tanh(C_{t,k})$$

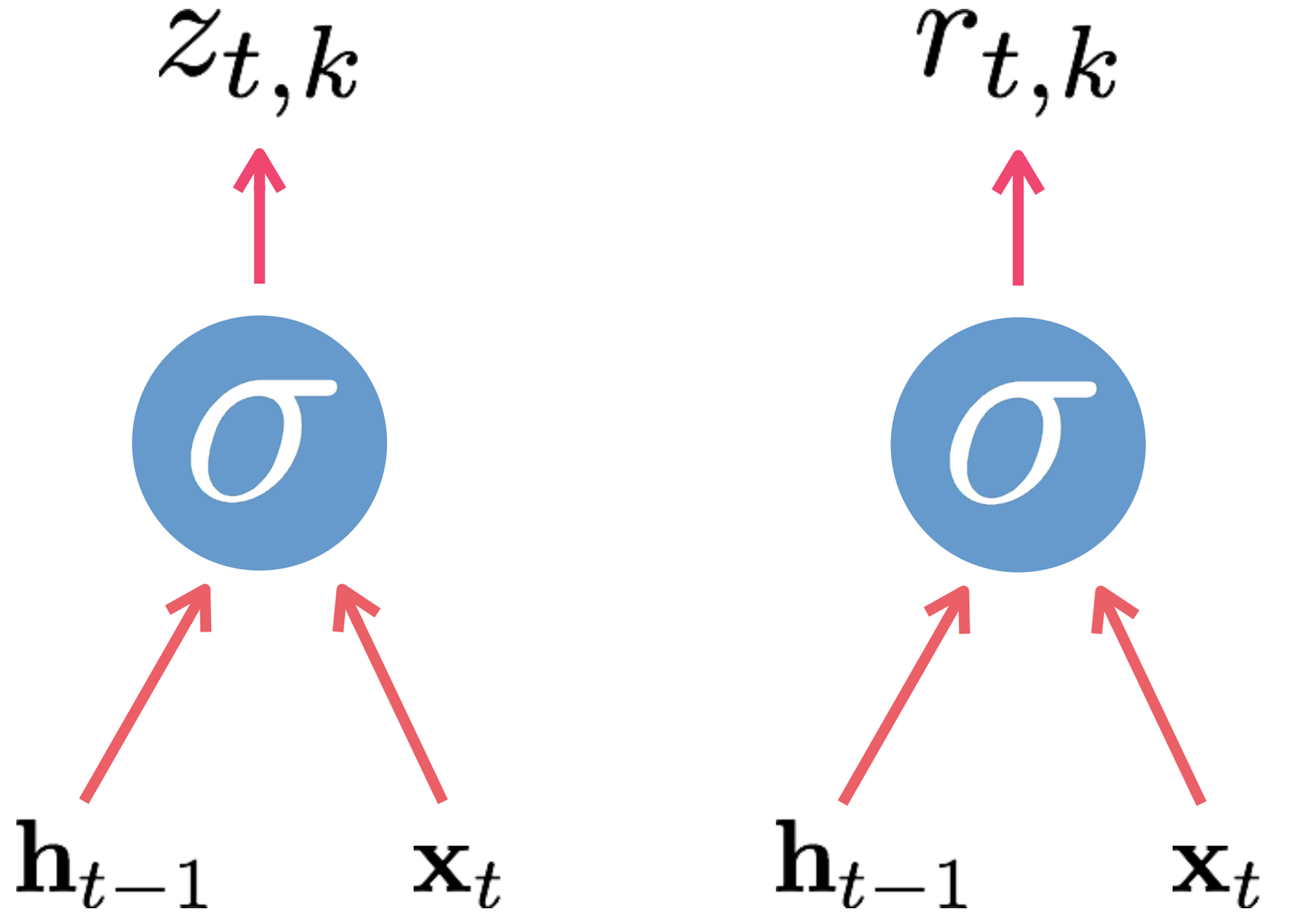
Do we really need to make things so complicated?

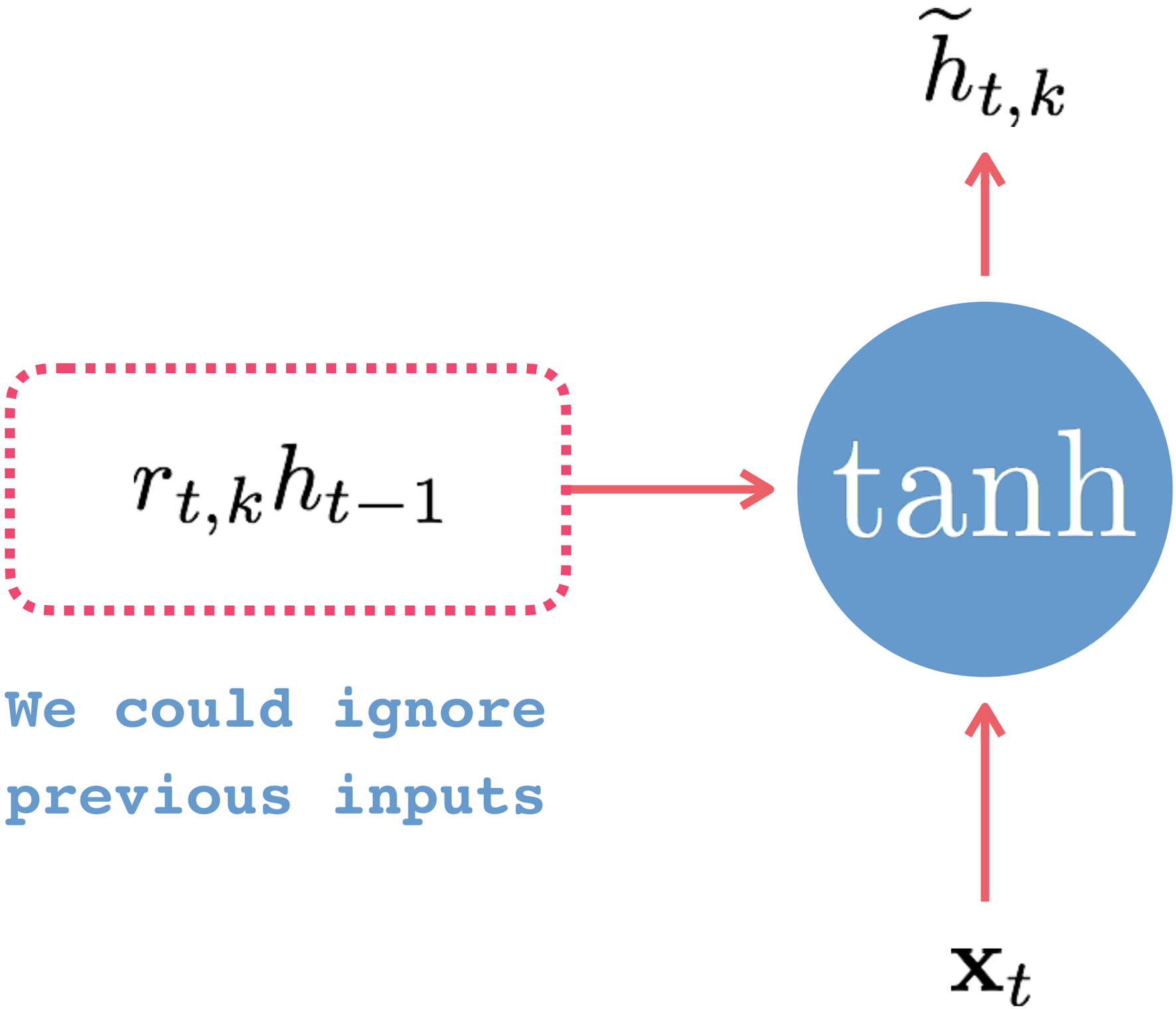
Gated Recurrent Unit

Simplified version of LSTM

Only 2 Gates

Although it has “gated” in the name


$z_{t,k}$
$r_{t,k}$
σ
σ
$\mathbf{h}_{t-1}$
$\mathbf{x}_t$
$\mathbf{h}_{t-1}$
$\mathbf{x}_t$
update gate
reset gate



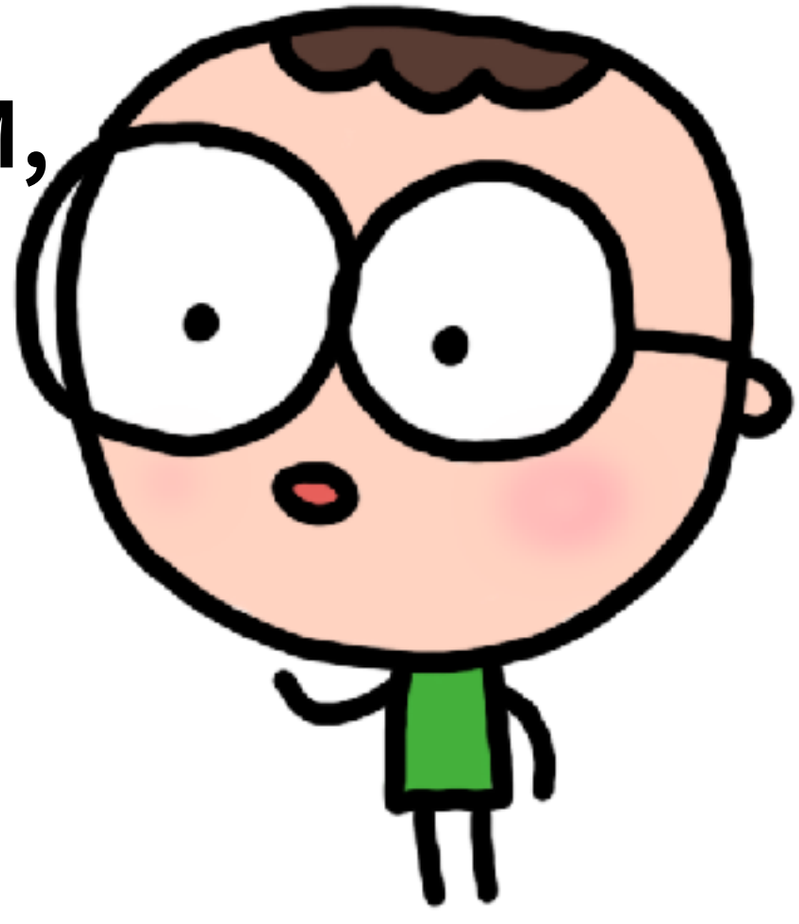
We could ignore previous inputs

$$h_{t,k} = z_{t,k} h_{t-1,k} + (1 - z_{t,k}) \tilde{h}_{t,k}$$

Key

Names of RNNs

Now talking about RNN, in fact, including the original RNN, LSTM, GRU and other variants.



In particular, the original RNN is called Vanilla RNN, and is SimpleRNN in Keras.

3

Design thinking, creative problem solving

Design Thinking Process (Standford d.school)

Very close to the engineer’s approach

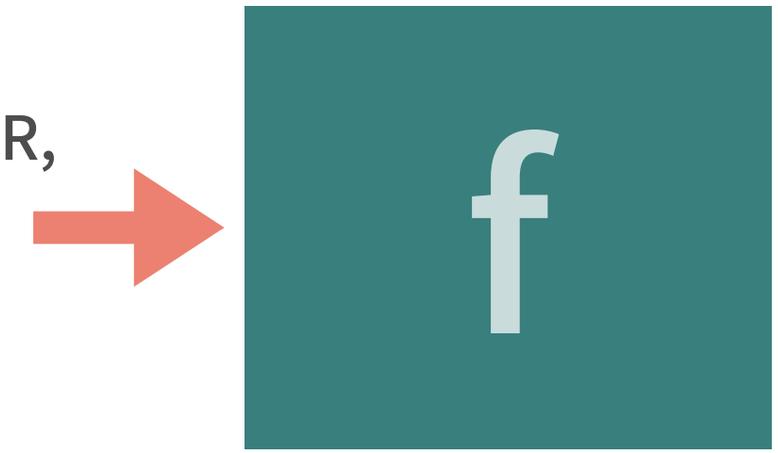
“Our model is rarely successful for the first time.”

—Yi-Shin Chen, National Tsing Hua University

MLB Player Home Runs Prediction

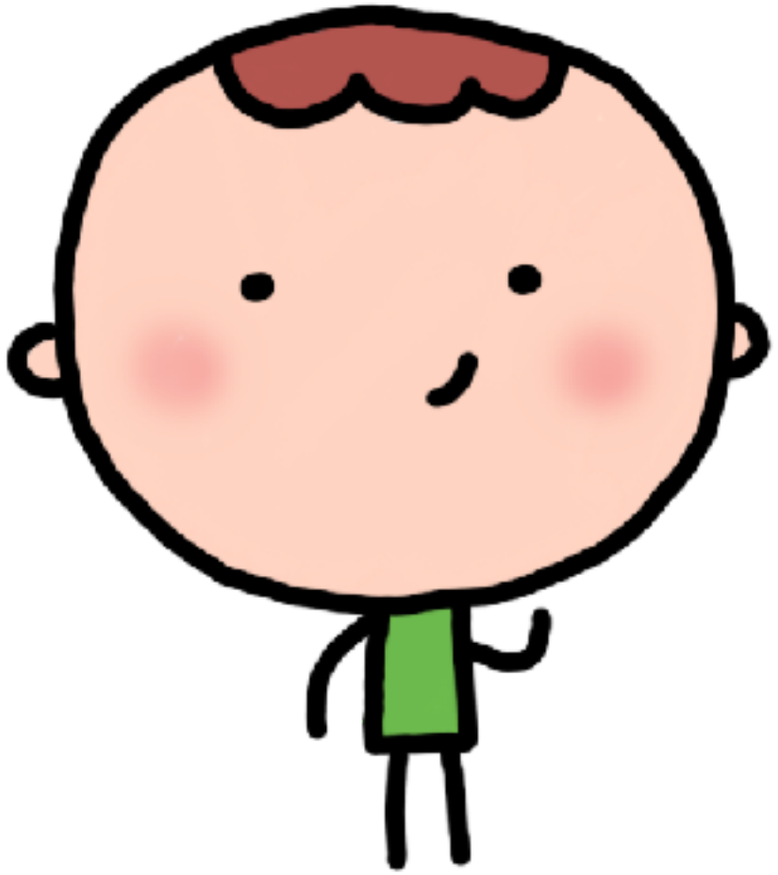
Year t-1

[Age, G, PA, AB, R, H, 2B, 3B, HR,
RBI, SB, BB, SO, OPS+, TB]
15 features

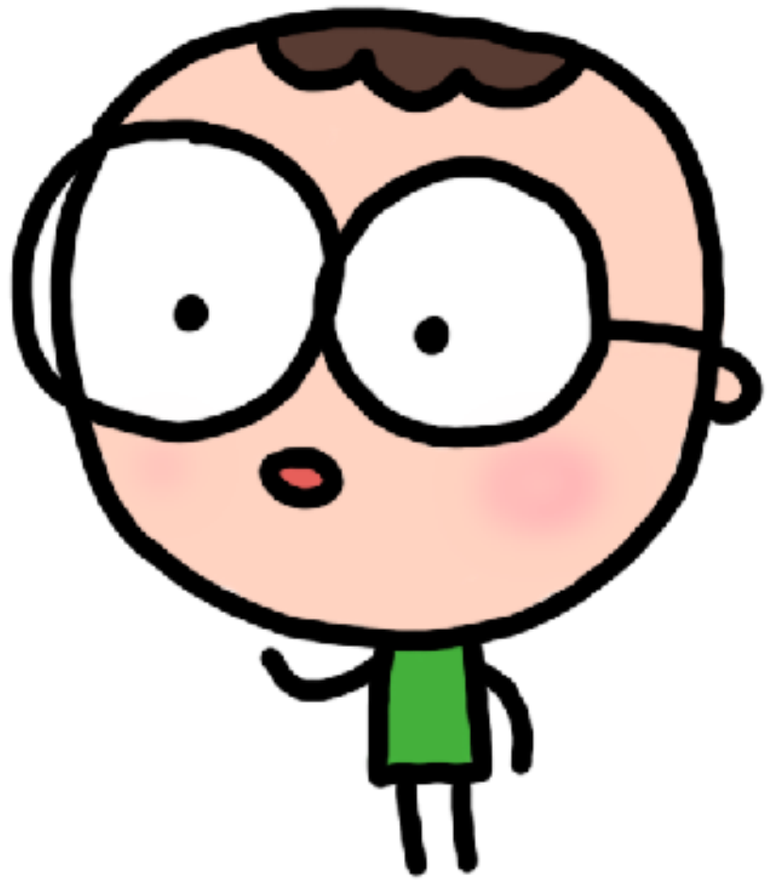


Number of homers in the year of t

- **Use LSTM. Input the data in a period of 10 years and predict the home runs in the next year.**
- **Only one LSTM layer!**



Don't guess the exact number, guess which interval!

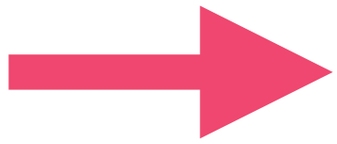


divided into 5 subintervals:

0-9, 10-19, 20-29, 30-39, 40+

One-Hot encoding

10-19



$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

1 0-9

2 10-19

3 20-29

4 30-39

5 40+

2017 forecast result

Mike Trout (LAA)
Predicted 30-39
Actual 33

Kris Bryant (CHC)
Predicted 30-39
Actual 29

Mookie Betts (BOS)
Predicted 20-29
Actual 24

Daniel Murphy (WSH)
Predicted 20-29
Actual 23

Jose Altuve (HOU)
Predicted 20-29
Actual 24

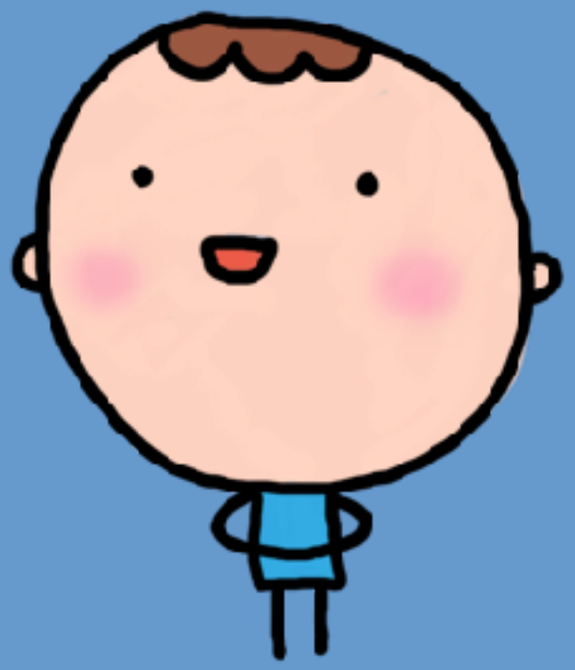
Corey Seager (LAD)
Predicted 20-29
Actual 22

Fancy ways to ask questions!

Standard NN

CNN

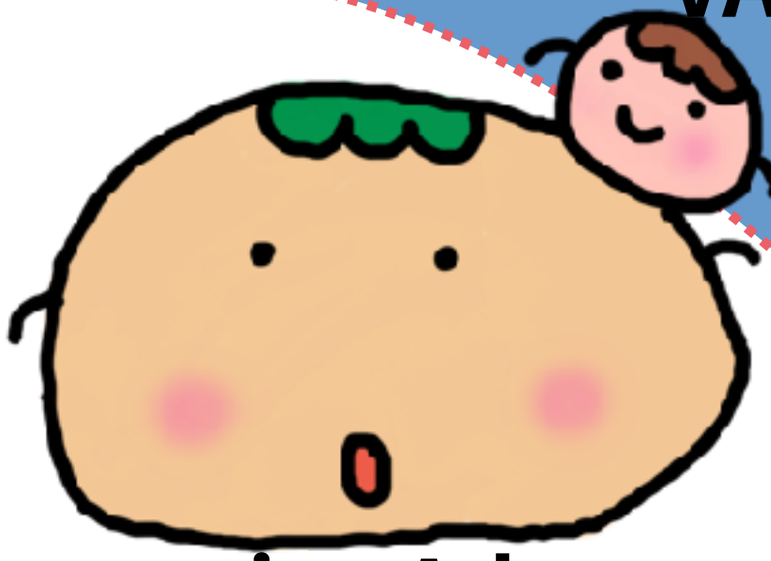
RNN



VAE

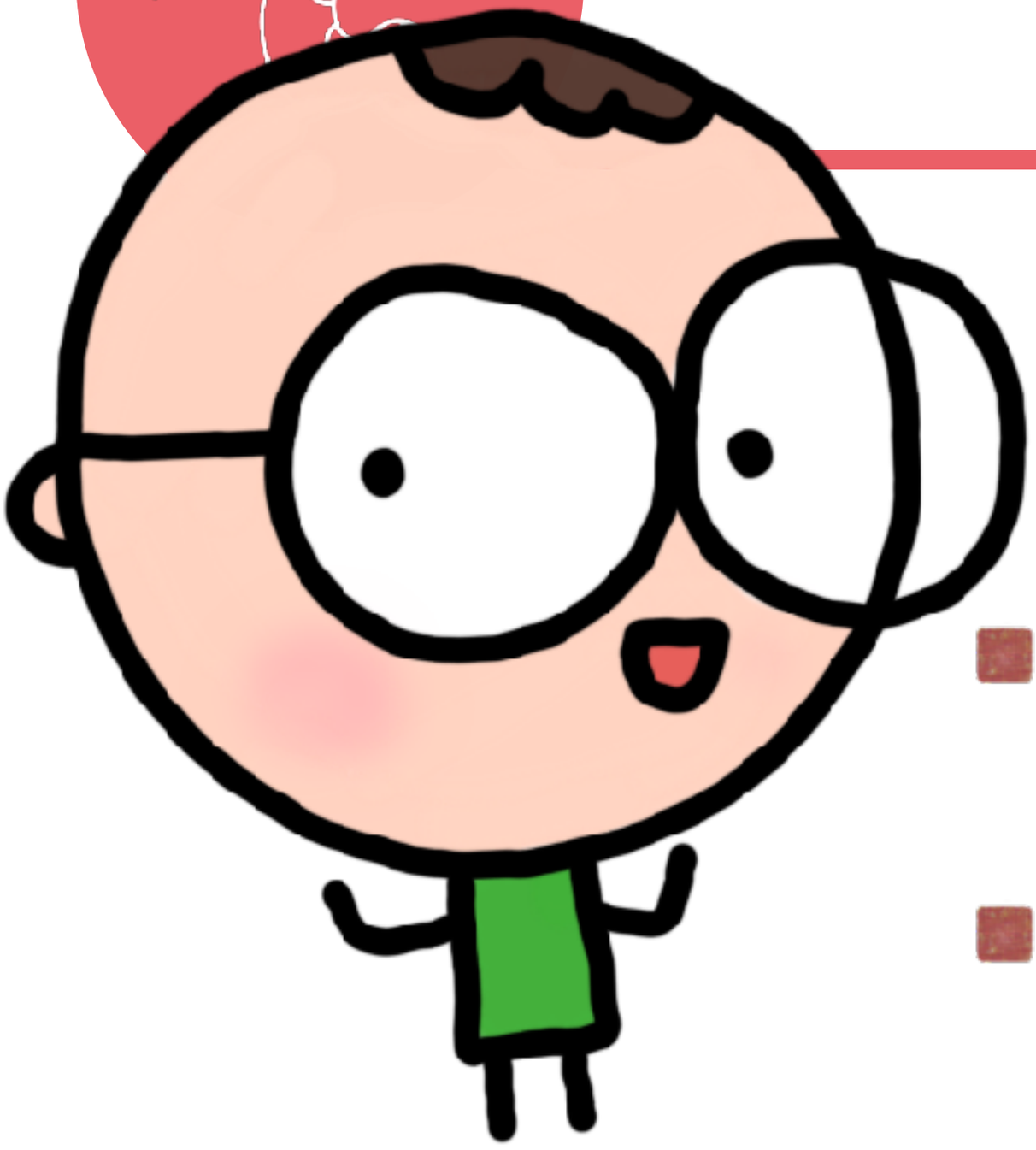
Capsule

Reinforcement Learning (RL)



Generative Adversarial Network (GAN)

GAN



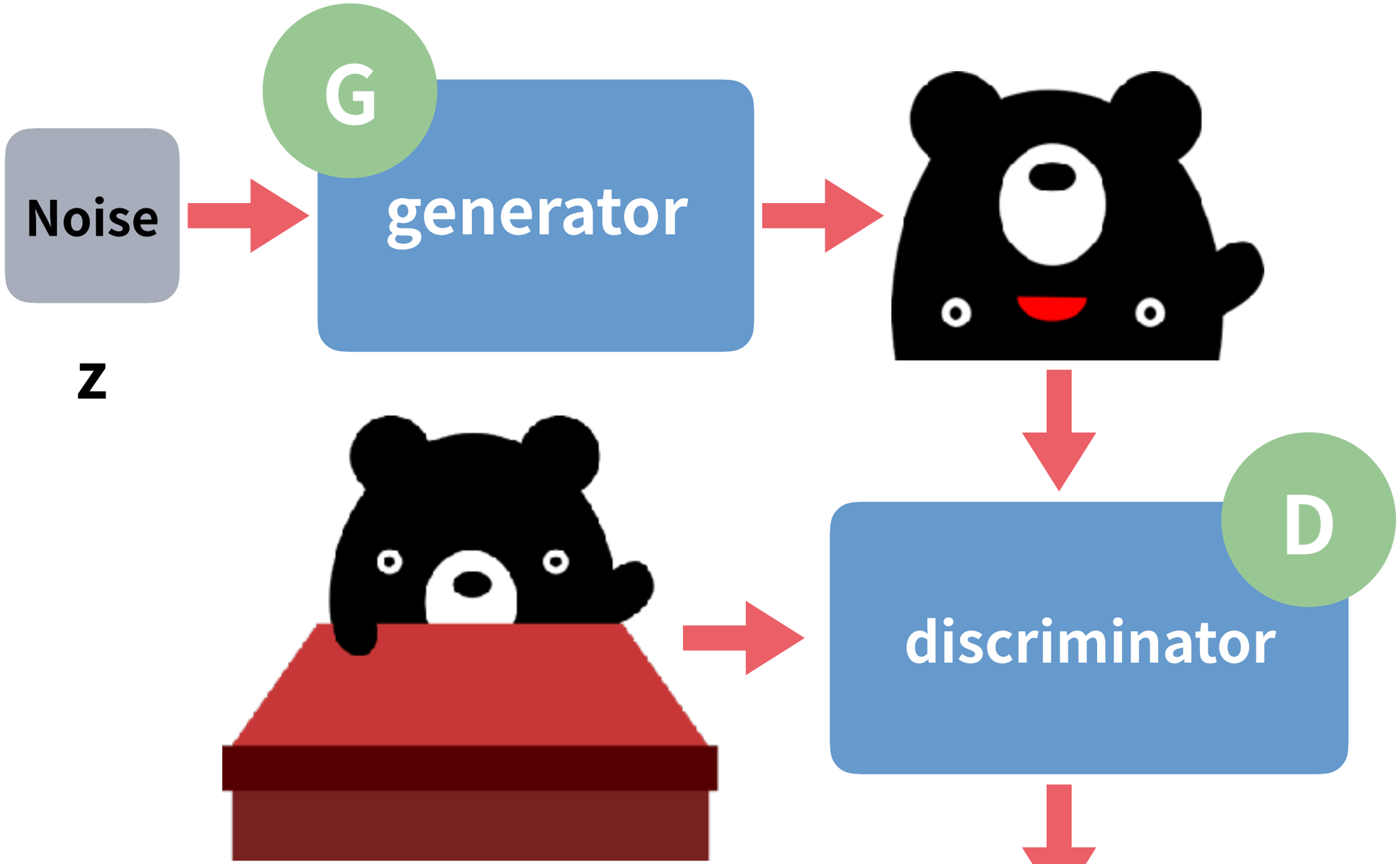
- Generative Adversarial Network
- Yann LeCun said GNN is the most promising model

> There are many interesting recent development in deep learning...
>
> The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).

—Yan LeCun (楊立昆), 2016

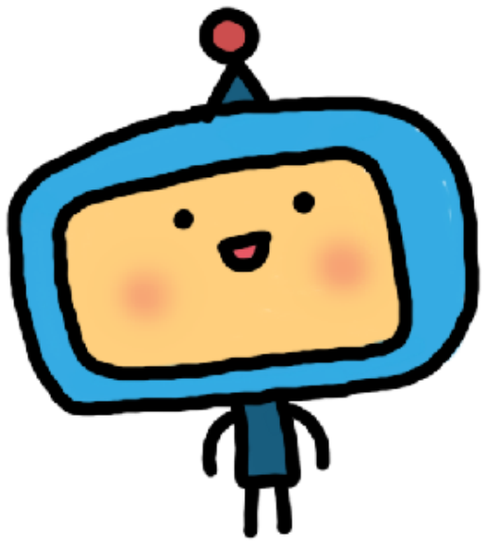
The GAN Zoo

https://github.com/hindupuravinash/the-gan-zoo



A GAN consists two neural networks,
a Generator and a Discirminator.

D, G PK!



Want

$D(G(\mathbf{z}))$

Close to 1

Generator

G

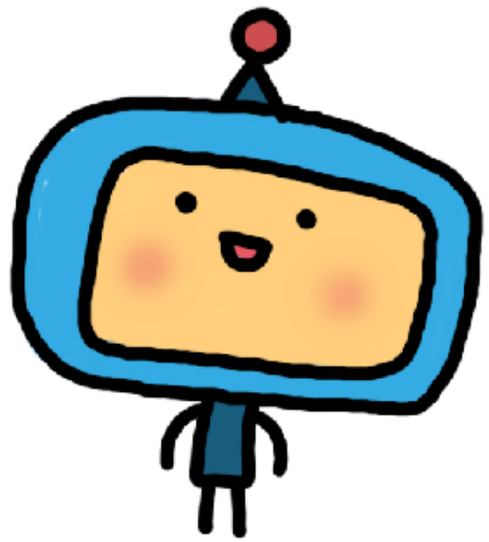
Want

$D(G(\mathbf{z}))$

Close to 0

$D(\mathbf{x})$

Close to 1



Discriminator

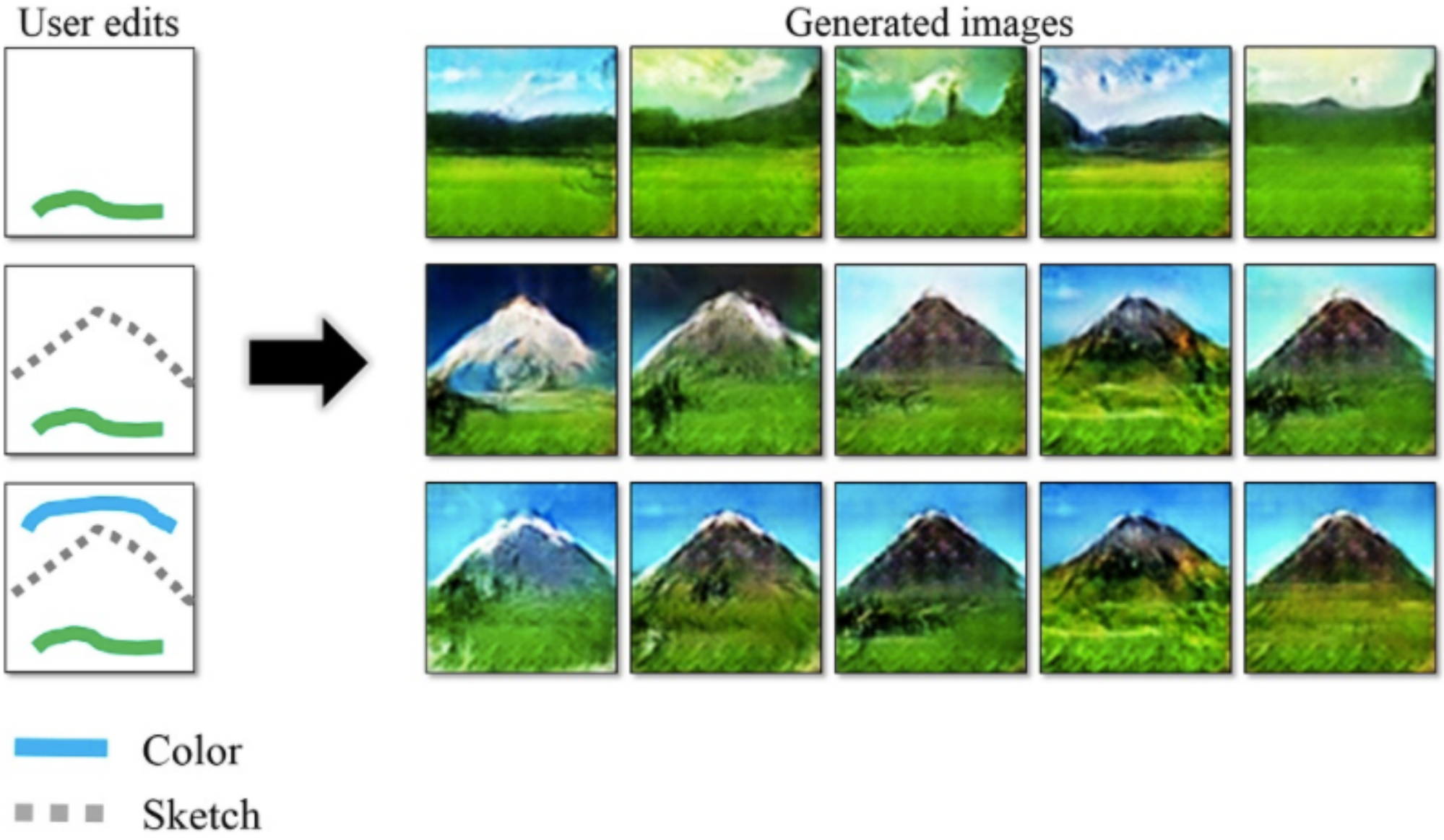
D

iGAN

Jun-Yan Zhu et al. (ECCV 2016)

https://arxiv.org/abs/1609.03552

"Generative Visual Manipulation on the Natural Image Manifold"



Every one can draw!

https://youtu.be/9c4z6YsBGQ0

Progressive GAN

Karras et al. NVIDIA team, (ICLR 2018)

https://arxiv.org/abs/1710.10196

"Progressive Growing of GANs for Improved Quality, Stability, and Variation"

Karras-Aila-Laine-Lehtinen

Progressive Growing of GANs for Improved Quality, Stability, and Variation

- By a NVIDIA team
- Theano, Python 2, single GPU (Hey, it's NVIDIA)

These are fake (1024x1024)

Pix2Pix

Isola, Zhu, et al., (CVPR 2017)

https://arxiv.org/abs/1611.07004

"Image-to-Image Translation with Conditional Adversarial Networks"

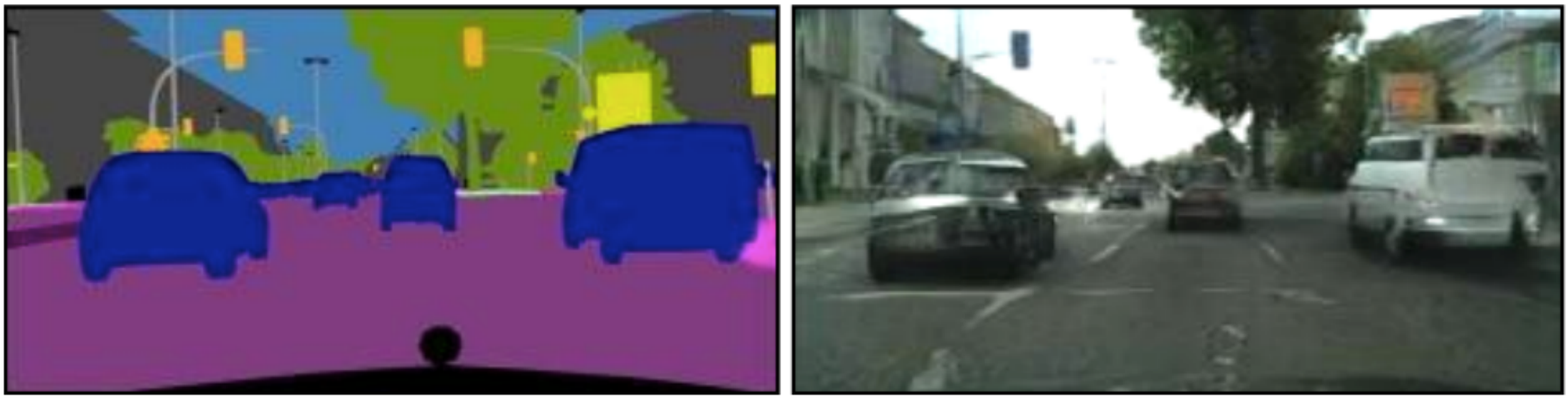
input

output

Pix2pix transfers satellite images to maps

*** From the original paper of Isola, Zhu et al. (2017)**

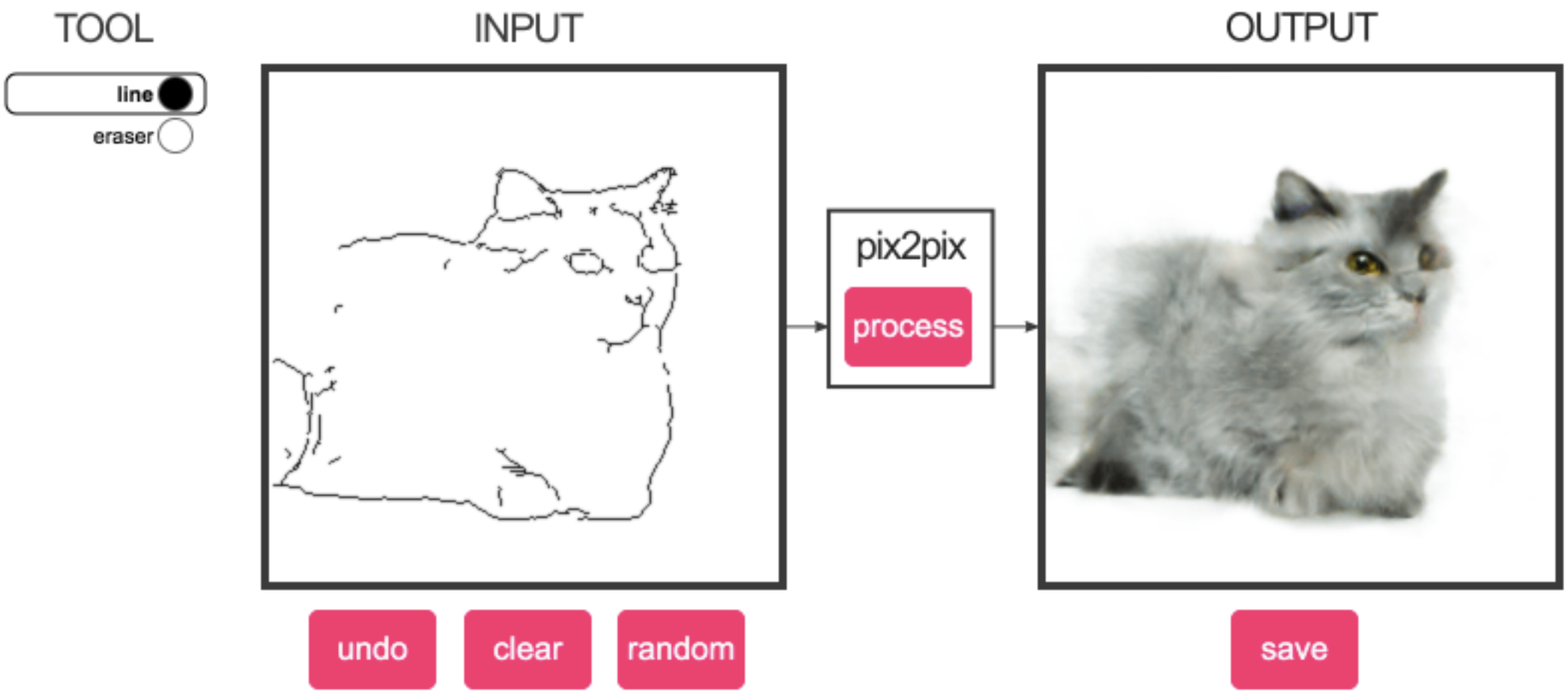
Labels to Street Scene



input output

Pix2pix generates street views

*** From the original paper of Isola, Zhu et al. (2017)**



Pix2pix On-line Version

https://affinelayer.com/pixsrv/

* by Christopher Hesse

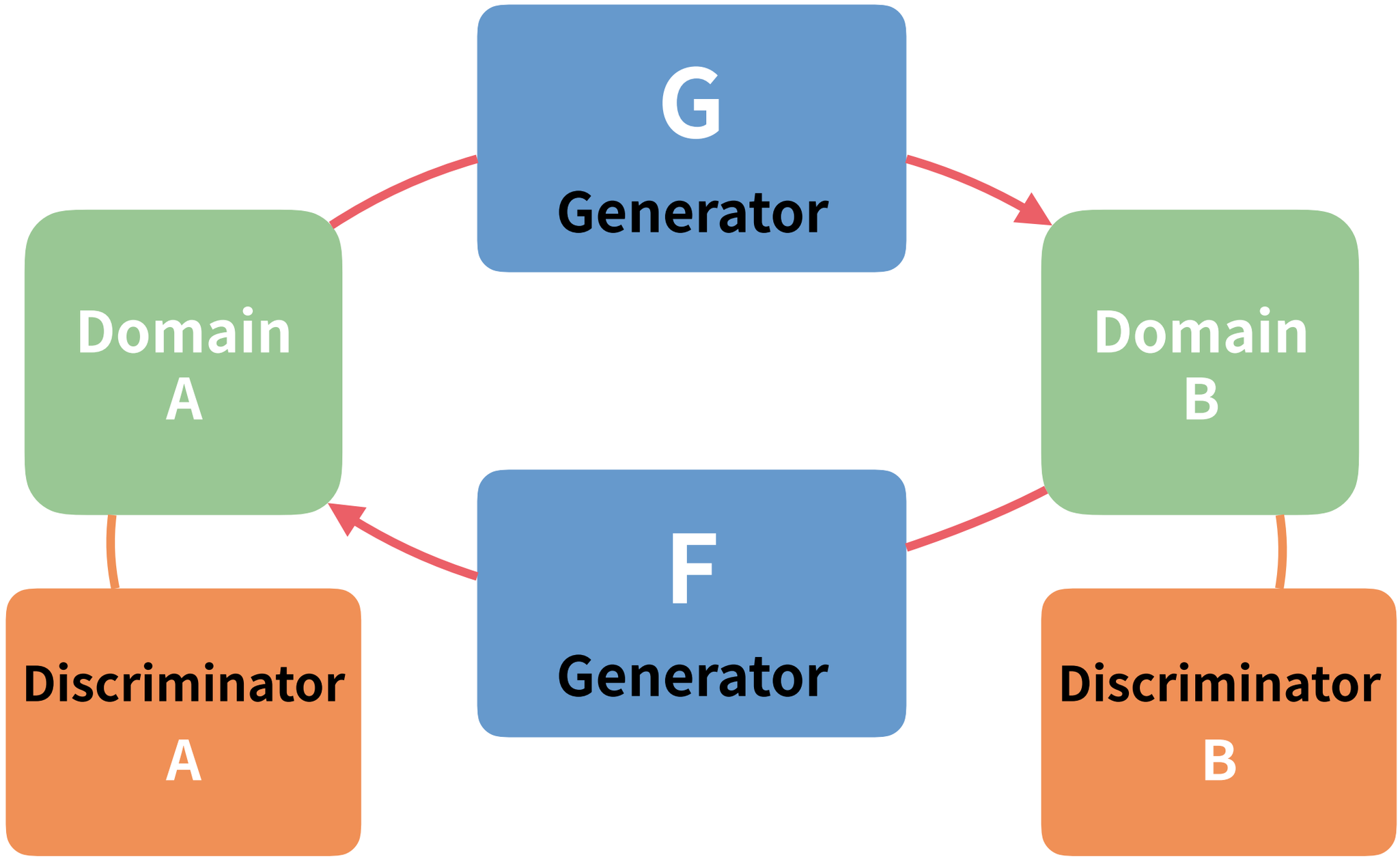
CycleGAN

Jun-Yan Zhu et al. (ICCV 2017)

https://arxiv.org/abs/1703.10593

"Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks"

CycleGAN


G
Generator
Domain
A
Domain
B
F
Generator
Discriminator
A
Discriminator
B

Don’t need to pair our data sets!!

And open infinitely many possibilities…

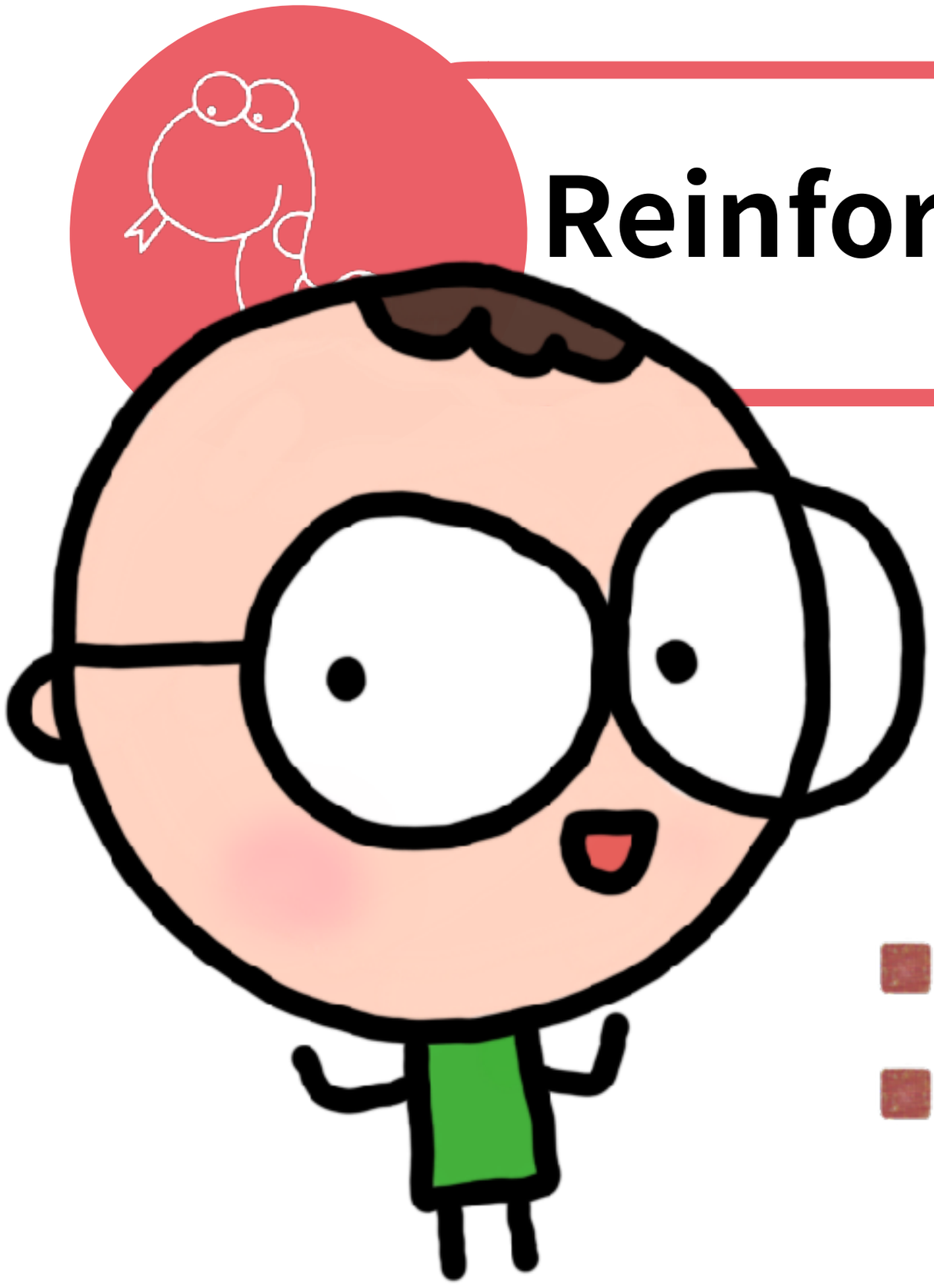
Horses to Zebras!!

https://youtu.be/9reHvktowLY

One example from authors of CycleGAN

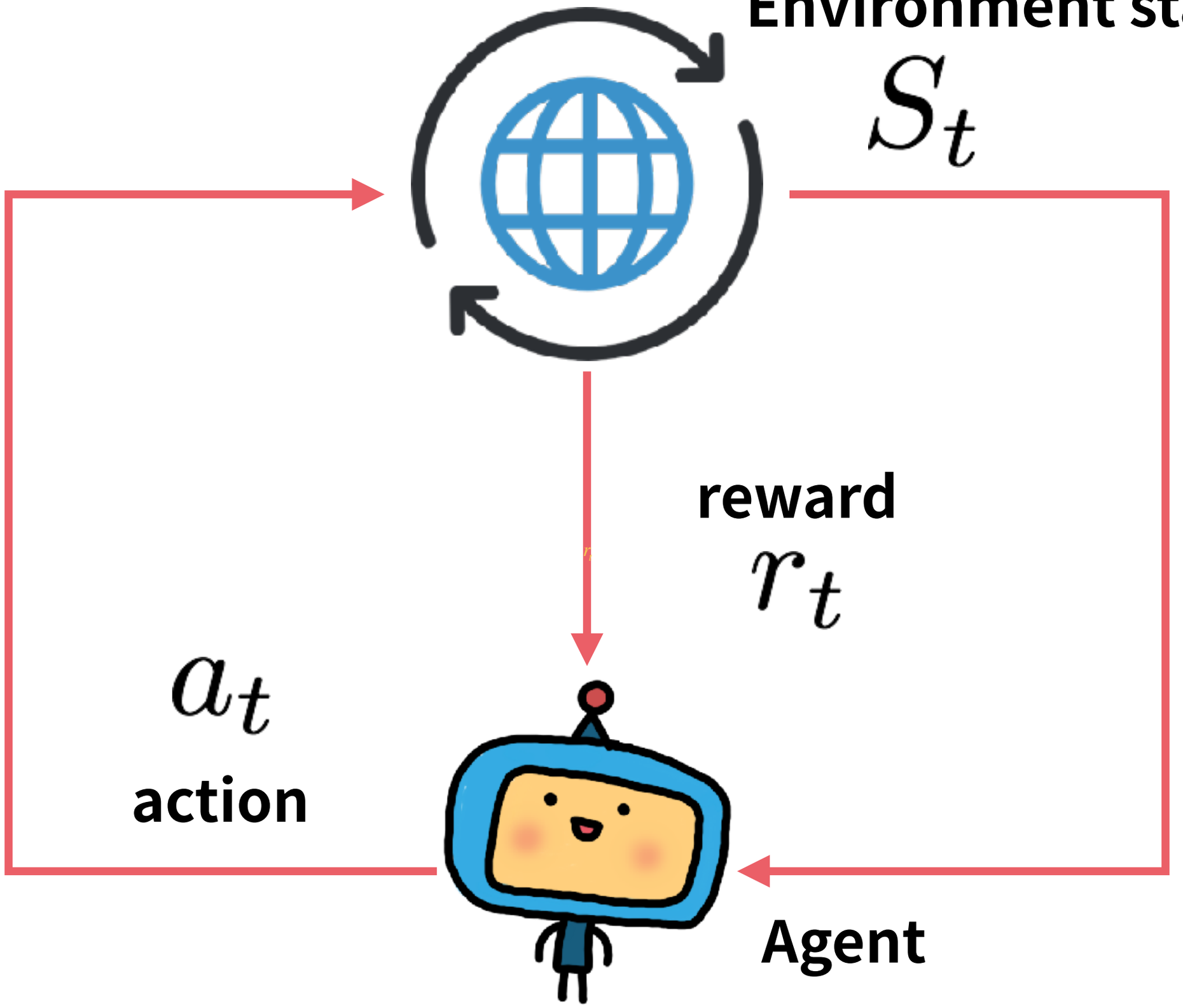
Face-off (Tzer-Jen Wei)

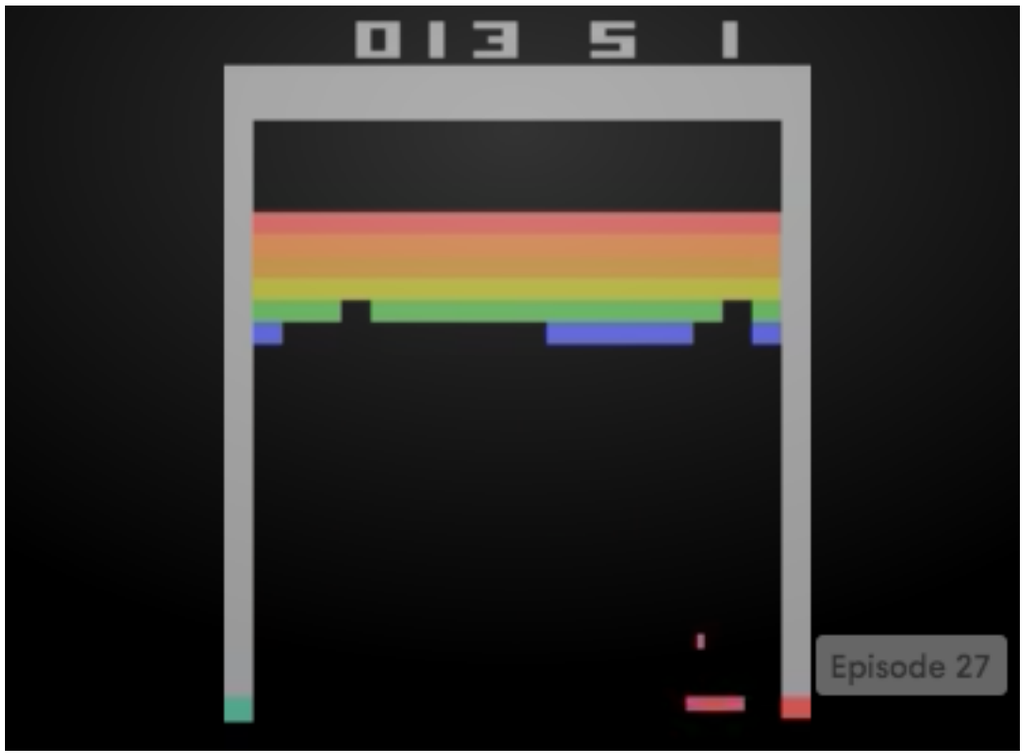
https://youtu.be/Fea4kZq0oFQ



Reinforcement Learning

- Reinforcement Learning
- AlphaGo key technique

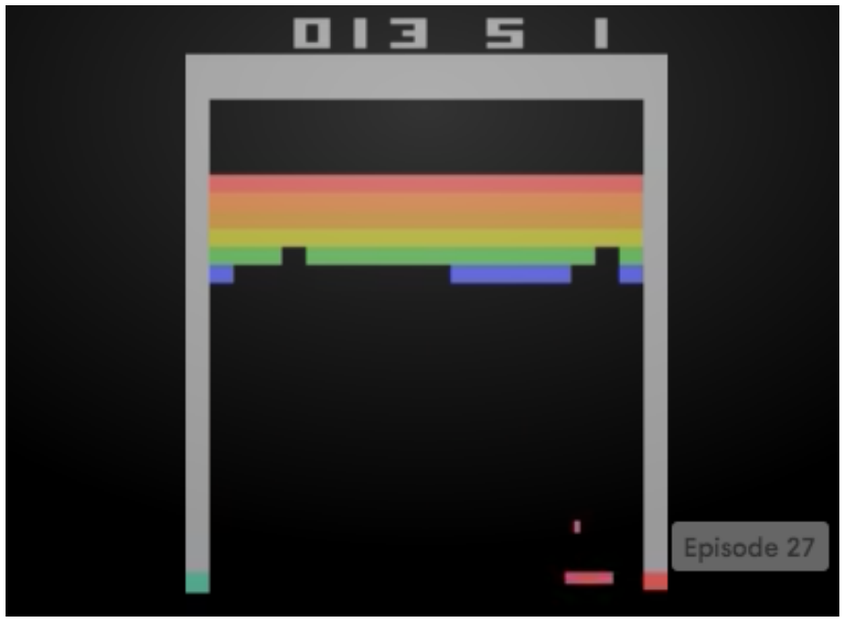

Environment state
S_t
reward
r_t
a_t
action
Agent
(computer)



Let’s play!

1 Policy Based

policy function



π

Left

or

Right

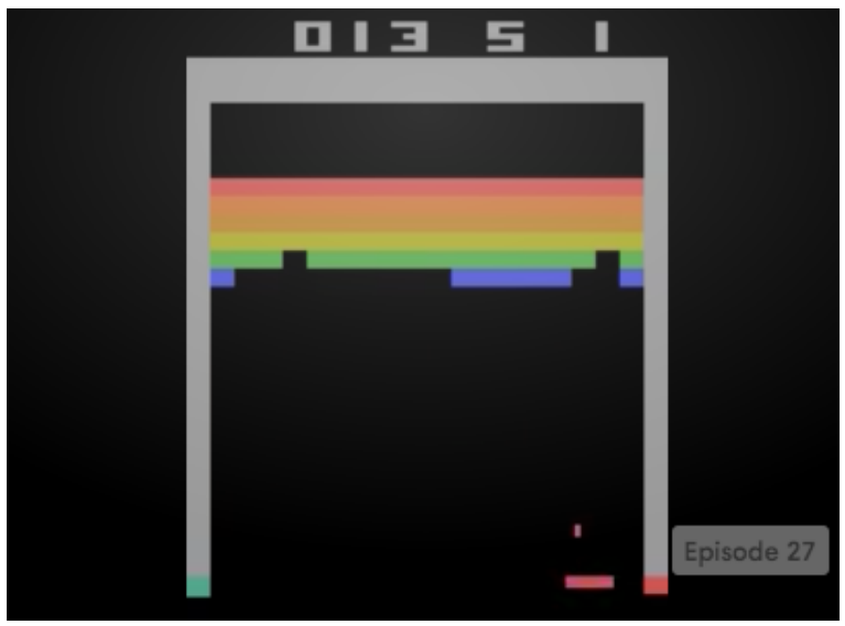
State S_t

Action a_t

Usually it is not easy to learn directly...

2 Value Based

Value function



+

Action

Grade (Usually estimate reward)

Netflix AlphaGo Film (Highly Recommended)

Self-driving cars

Automated trading system

- Fixed one ETF each time
- Start with $20,000
- Work for one year (Sell all ETF at the end)
- Using reinforcement learning

* ETF data provided by the Global Intelligence

Data from past month (20 days, in a 20x6 matrix)

5 possible actions

1. Buy 20 units
2. Buy 10 units
3. No trading
4. Sell 10 units
5. Sell 20 units

Trading Results

	CDQN	Hold & buy
ETF1	17.71%	10.89%
ETF2	16.53%	12.6%
ETF3	16.3%	0.35%
ETF4	14.4%	13.25%
ETF5	14.3%	12.7%
ETF6	13.91%	13.37%
ETF7	13.17%	10.52%
ETF8	12.35%	17.07%
ETF9	11.68%	10.81%
ETF10	11.09%	8.14%

	CDQN	Hold & buy
ETF11	10.76%	5.26%
ETF12	10.19%	13.17%
ETF13	7.8%	1.42%
ETF14	6.23%	3.56%
ETF15	5.73%	4.61%
ETF16	3.78%	-12.76%
ETF17	2.85%	5.83%
ETF18	1.59%	-4.45%
ETF19	1.07%	-18.09%
ETF20	-0.59%	-0.75%