

Deep learning

-- Introduction

Junjie Cao @ DLUT

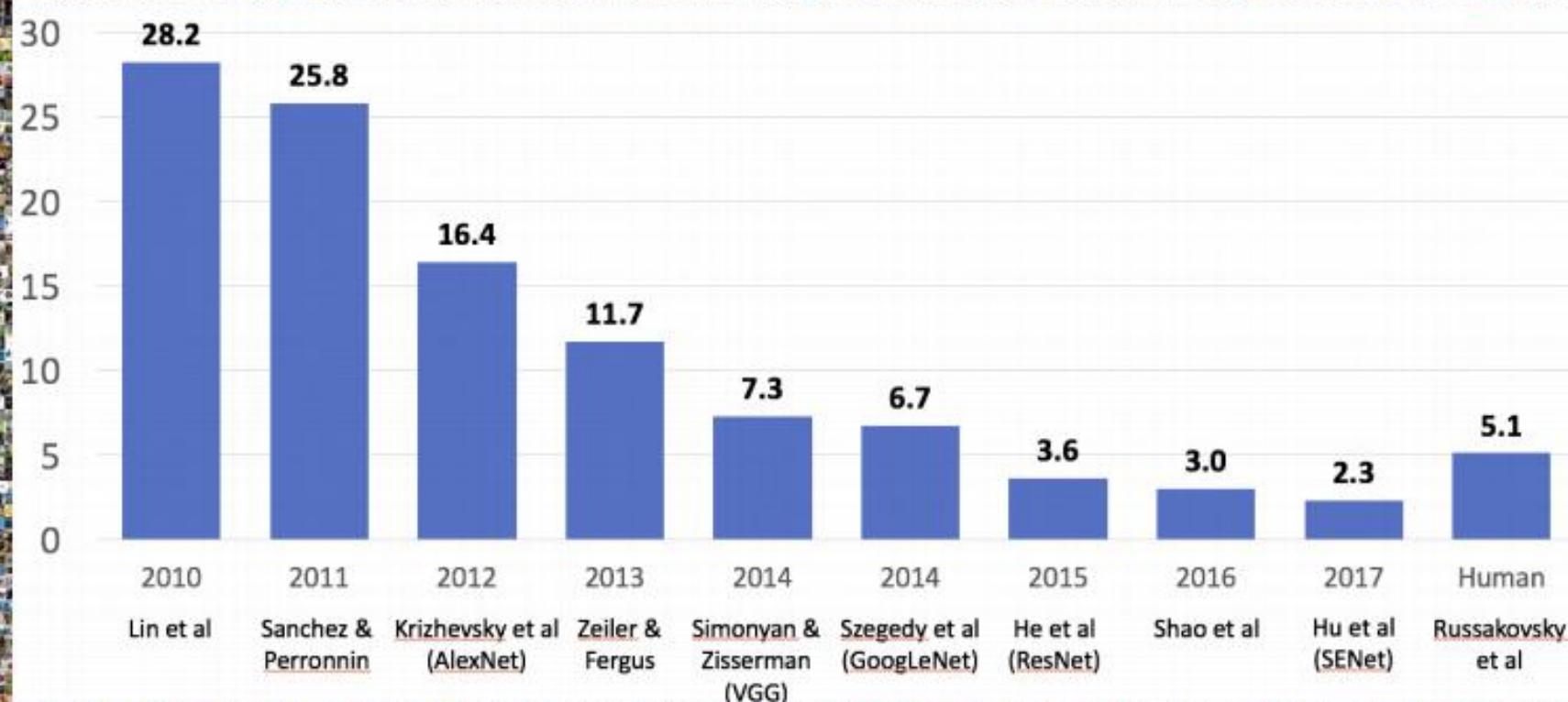
Spring 2018

Welcome



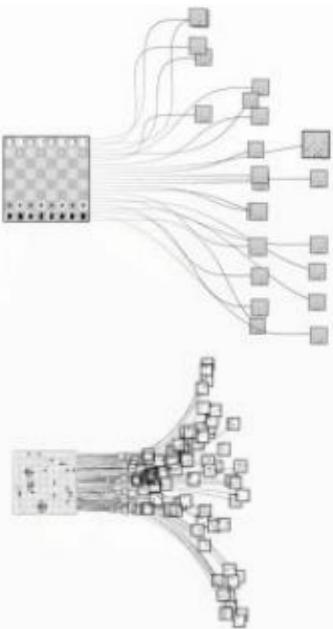


The Image Classification Challenge:
1,000 object classes
1,431,167 images



But, we human actually lose!

- A demo that shows we, human, **lose, on the classification task, we are proud of, we have been trained for millions of years!**



Chess: 10^{47}
Deep Blue, Feb 10, 1996

Our entire universe contains only
about 10^{80} atoms

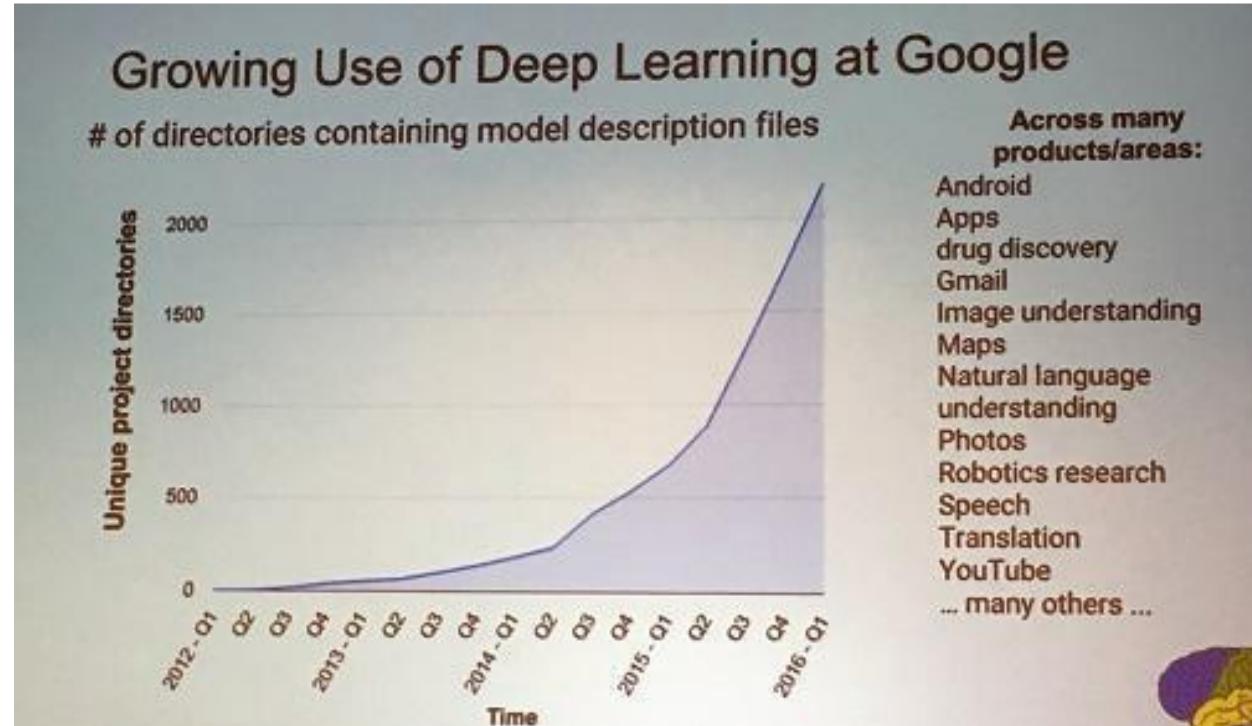
Go: 10^{170}
AlphaGo, March, 2016

In "Nature" 27 January 2016:

- “AlphaGo was not preprogrammed to play Go: rather, it learned using a **general-purpose algorithm** that allowed it to interpret the game’s patterns.”
- “...AlphaGo program applied **deep learning** in neural networks (**convolutional NN**) — brain-inspired programs in which connections between layers of simulated neurons are strengthened through examples and experience.”

We (will) lose on many specific tasks!

- Speech recognition
- Translation
- Self-driving
- ...



Deep learning trends at Google. Source: SIGMOD/Jeff Dean

- BUT, they are **not AI yet...**
- Don't worry until it dates with your girl/boy friend...





人工智能潮起潮落



Titans

- LeCun, Y., Bengio, Y. and Hinton, G. E. (2015), Deep Learning. Nature, [\[pdf\]](#)



Yoshua Bengio



Yann LeCun



Geoffrey
Hinton



Ian Goodfellow

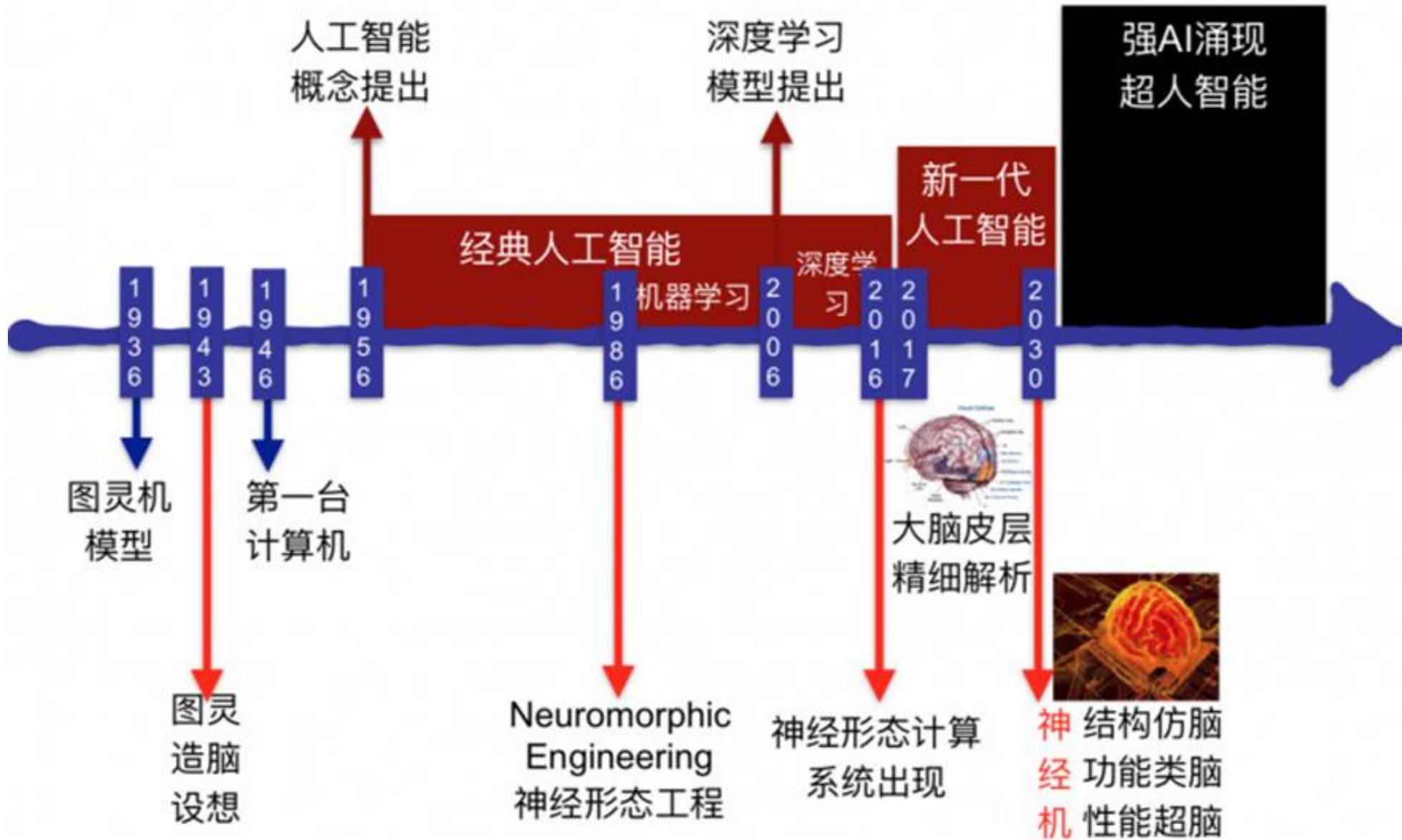


Andrew Ng



Fei-Fei Li

人类能制造出超级大脑吗？



全球四套神经计算系统 (2016) 和人脑的比较

Platform:	Human brain	Neurogrid	BrainScaleS	TrueNorth	SpiNNaker
Technology:	Biology	Analogue, sub-threshold	Analogue, over threshold	Digital, fixed	Digital, programmable
Microchip:		Neurocore	HiCANN		18 ARM cores
Feature size:	10 μm^2	180 nm	180 nm	28 nm	130 nm
# transistors:	23 M	15 M	5.4 B	100 M	
die size:	1.7 cm ²	0.5 cm ²	4.3 cm ²	1 cm ²	
# neurons:	65 k	512	1 M	16 k	
# synapses:	~100 M	100 k	256 M	16 M	
power:	150 mW	1.3 W	72 mW	1 W	
Board/unit:	PCB	20 cm wafer	PCB	PCB	
# chips:	16	352	16	48	
# neurons:	1 M	200 k	16 M	768 k	
# synapses:	4 B	40 M	4B	768 M	
power:	3 W	500 W	1 W	80 W	
Reference system:	1.4 kg		20 wafers in 7 × 19 ^a racks		600 PCBs in 6 × 19 ^a racks
# neurons:	100 B		4 M		460 M
# synapses:	10 ¹⁵		1 B		460 B
power:	20 W		10 kW		50 kW
Energy/connection:	10 fJ	100 pJ	100 pJ	25 pJ	10 nJ
Speed versus biology:	1×	1×	10 000×	1×	1×
Interconnect:	3D direct signalling	Tree-multicast	Hierarchical	2D mesh-unicast	2D mesh-multicast
Neuron model:	Diverse, fixed	Adaptive quadratic IF	Adaptive exponential IF	LIF	Programmable ^b
Synapse model:	Diverse	Shared dendrite	4-bit digital	Binary, 4 modulators	Programmable ^c
Run-time plasticity:	Yes!	No	STDP	No	Programmable ^d

Steve Furber. Large-scale neuromorphic computing systems.
Journal of Neural Engineering. (September 2016)

OLD CROW



MODERN CROW



Update Yourself - It saves a lot of extra effort

Machine Learning ≈ Looking for a Function

- Speech Recognition

$$f\left(\text{[Speech Waveform Image]} \right) = \text{“How are you”}$$

- Image Recognition

$$f\left(\text{[Orange Kitten Image]} \right) = \text{“Cat”}$$

- Playing Go

$$f\left(\text{[Go Board Image]} \right) = \text{“5-5” (next move)}$$

- Dialogue System

$$f\left(\text{“Hi”} \right) = \text{“Hello”}$$

(what the user said) (system response)

Framework

A set of
function

Model

$f_1, f_2 \Lambda$

Image Recognition:

$$f(\text{cat image}) = \text{"cat"}$$

$$f_1(\text{cat image}) = \text{"cat"}$$



$$f_2(\text{cat image}) = \text{"money"}$$



$$f_1(\text{dog image}) = \text{"dog"}$$

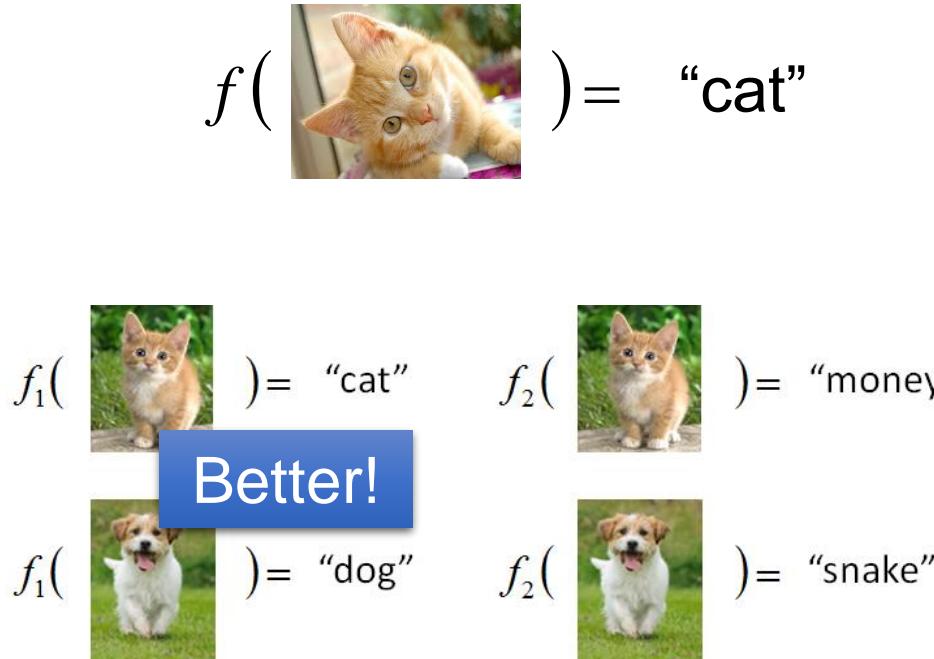
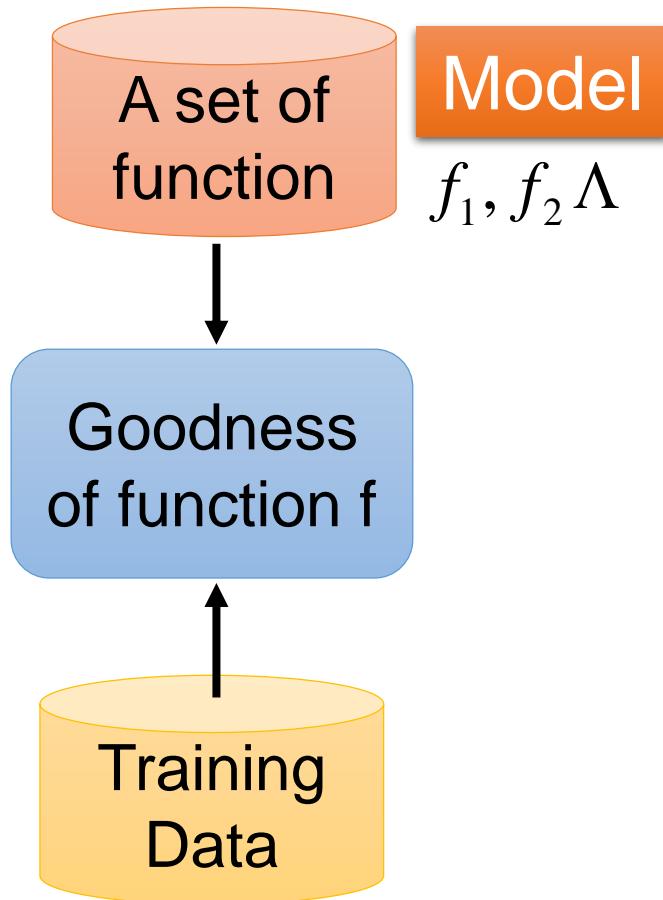


$$f_2(\text{dog image}) = \text{"snak e"}$$



Framework

Image Recognition:

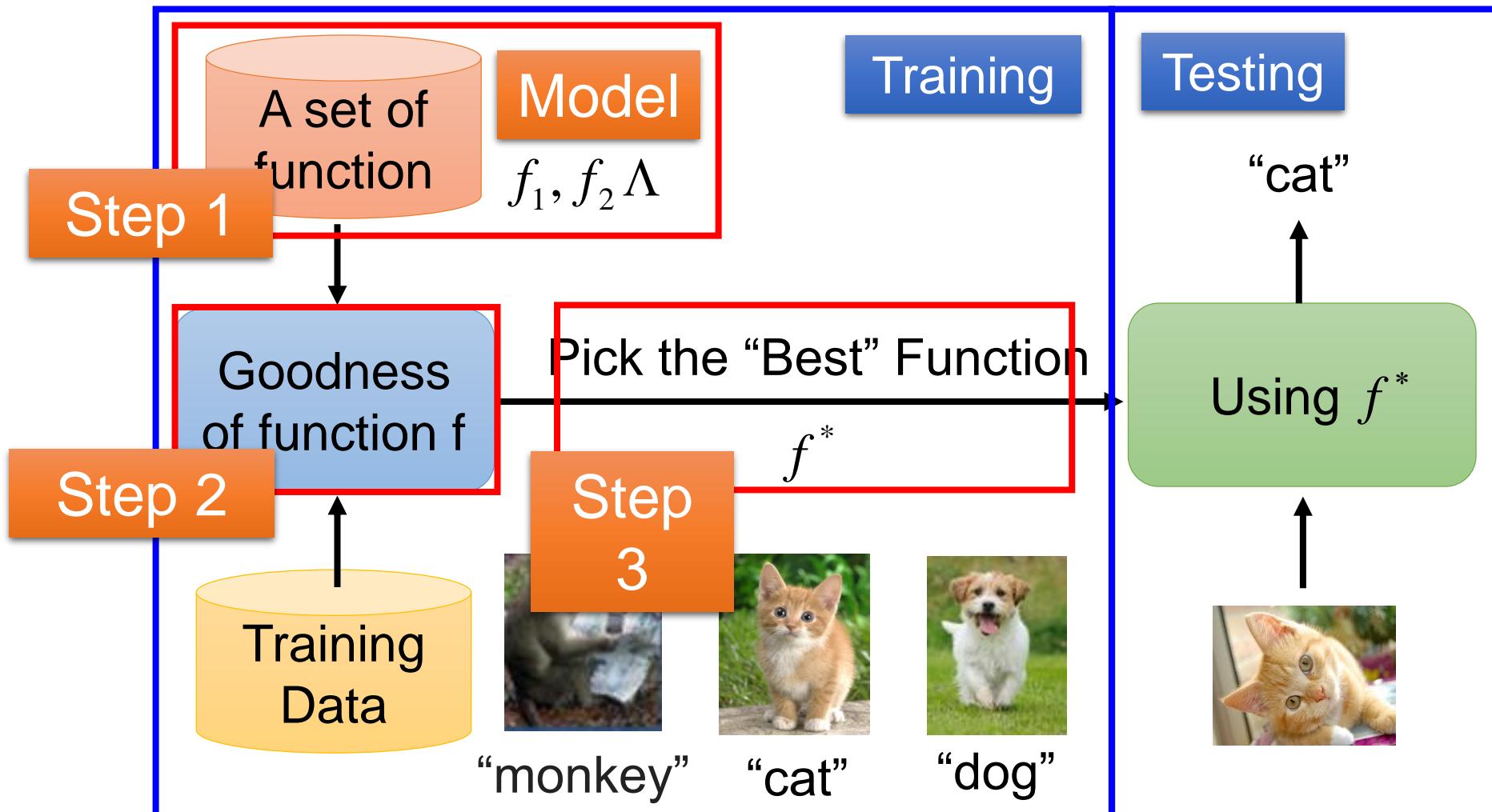


Framework

Image Recognition:

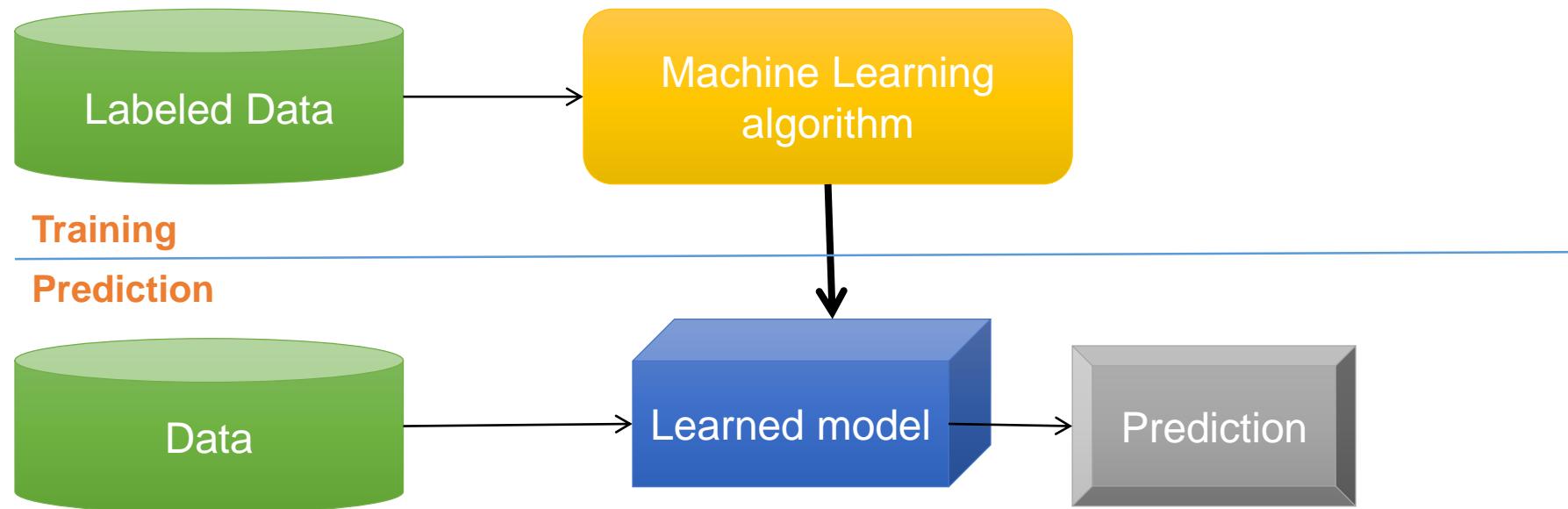


$$f(\text{cat}) = \text{"cat"}$$



Machine Learning Basics

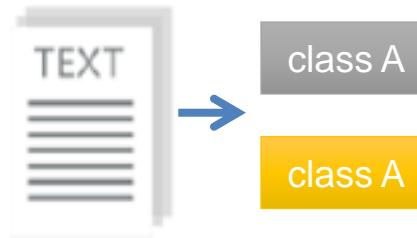
Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**



Methods that can learn from and make predictions on data

Types of Learning

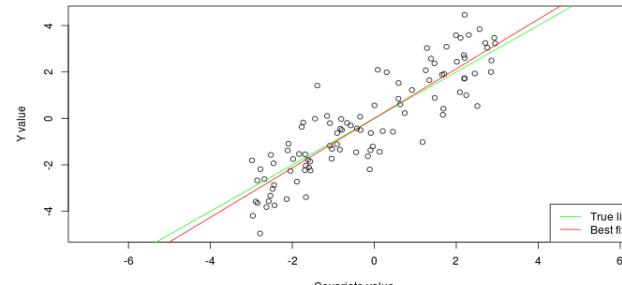
- **Supervised:** Learning with a **labeled training** set
 - Example: email **classification** with already labeled emails
- **Unsupervised:** Discover **patterns** in **unlabeled** data
 - Example: **cluster** similar documents based on text
- **Reinforcement learning:** learn to **act** based on **feedback/reward**
 - Example: learn to play Go, reward: **win or lose**



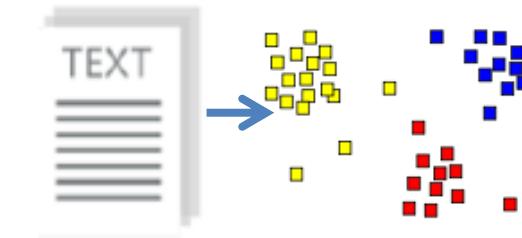
Classification

Anomaly Detection
Sequence labeling

...



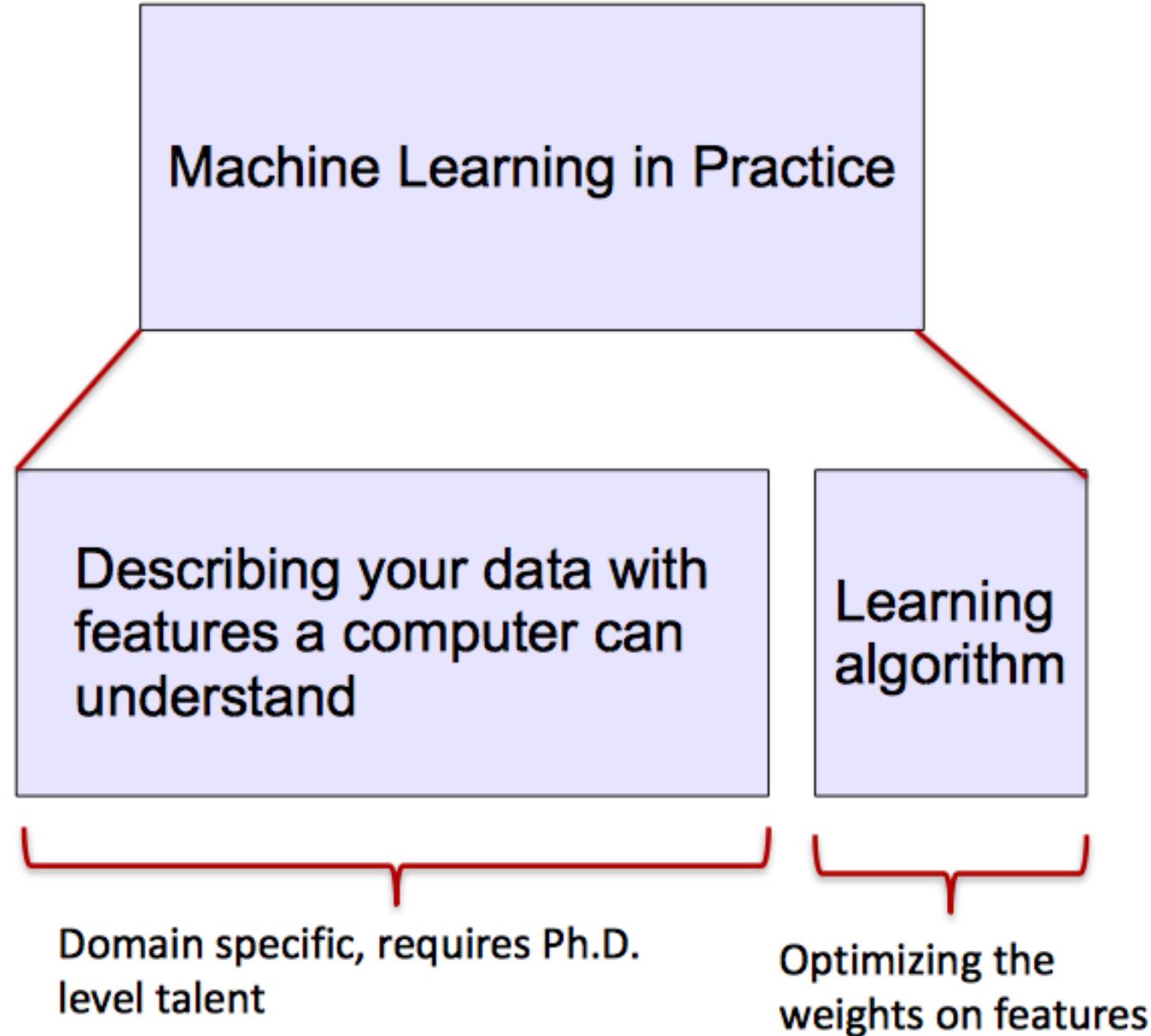
Regression



Clustering

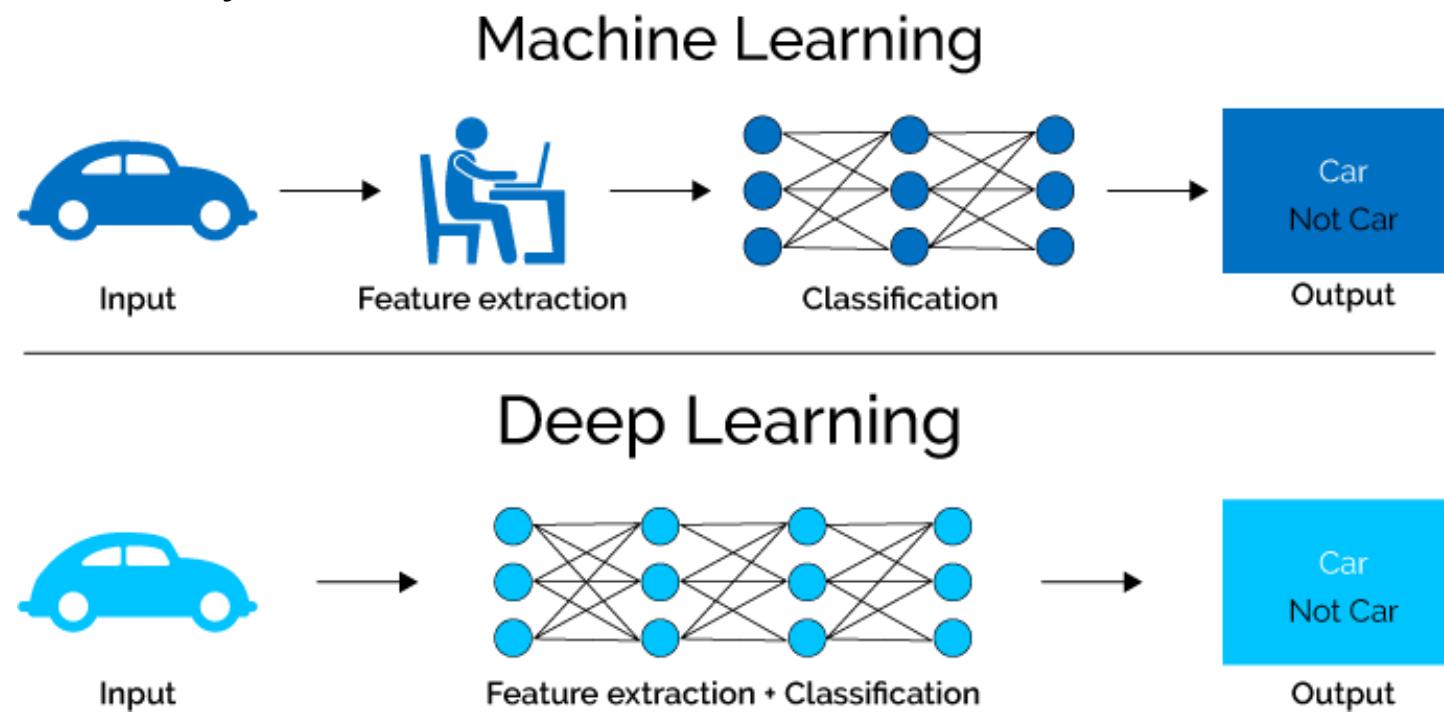
ML vs. Deep Learning

- Most machine learning methods work well because of **human-designed representations and input features**
- ML becomes just **optimizing weights** to best make a final prediction

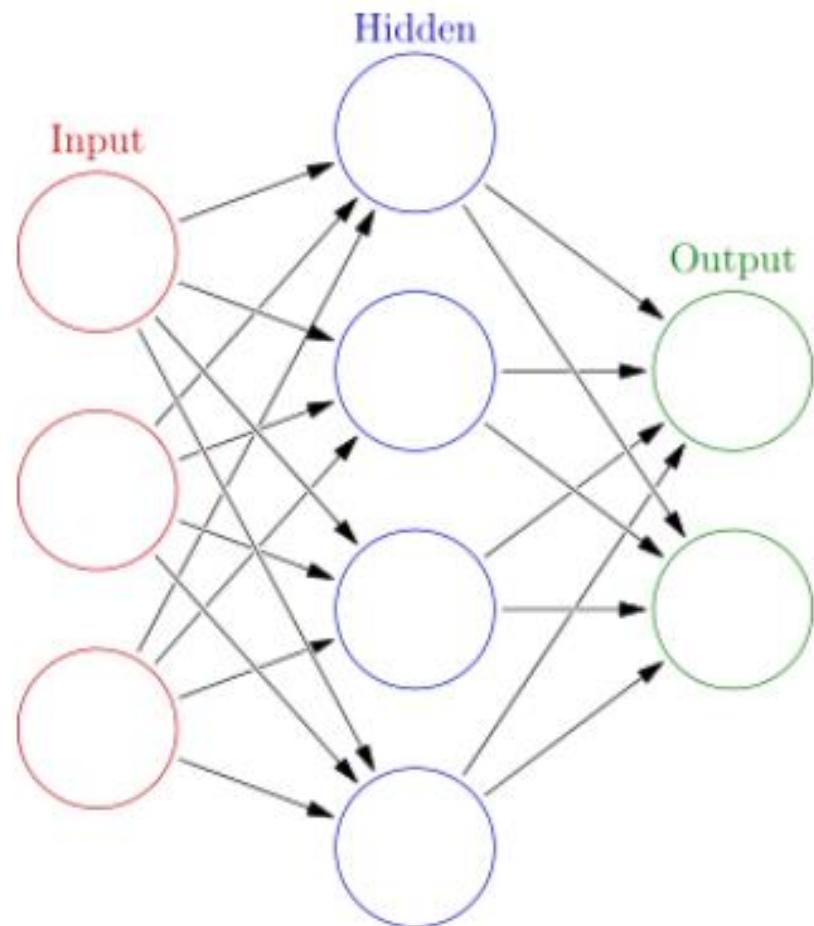


What is Deep Learning (DL) ?

- A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.
- Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**
- If you provide the system **tons of information**, it begins to understand it and respond in useful ways.

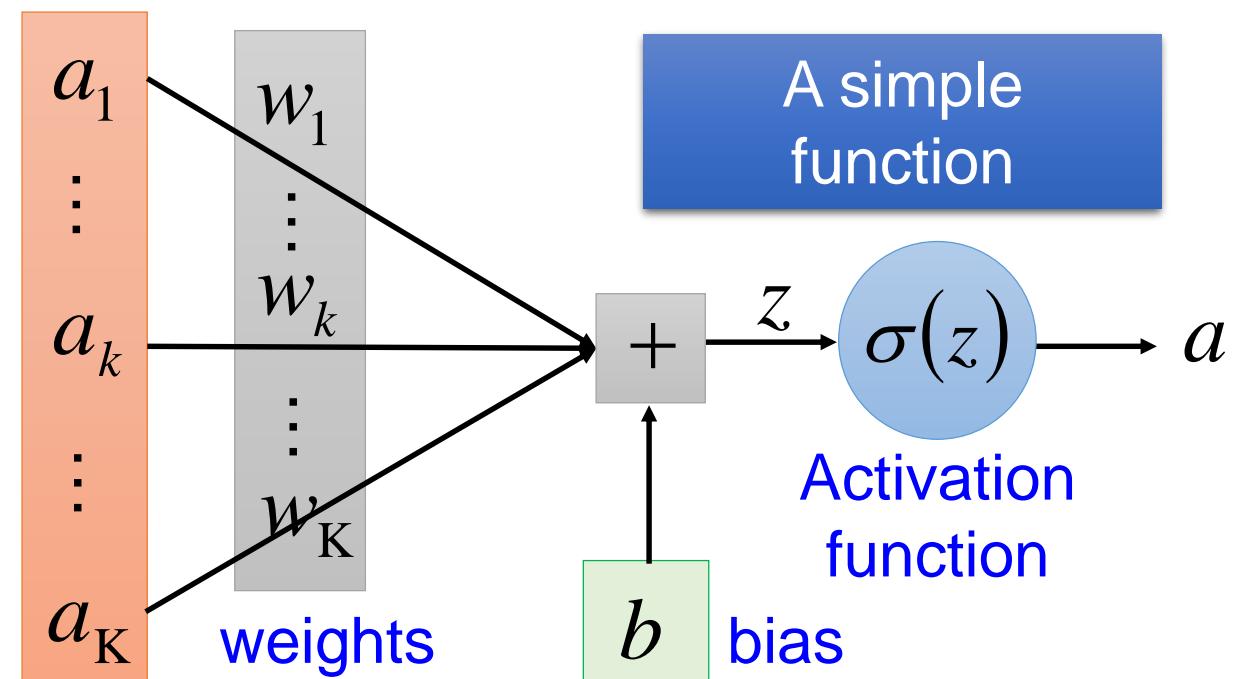


Neural Network



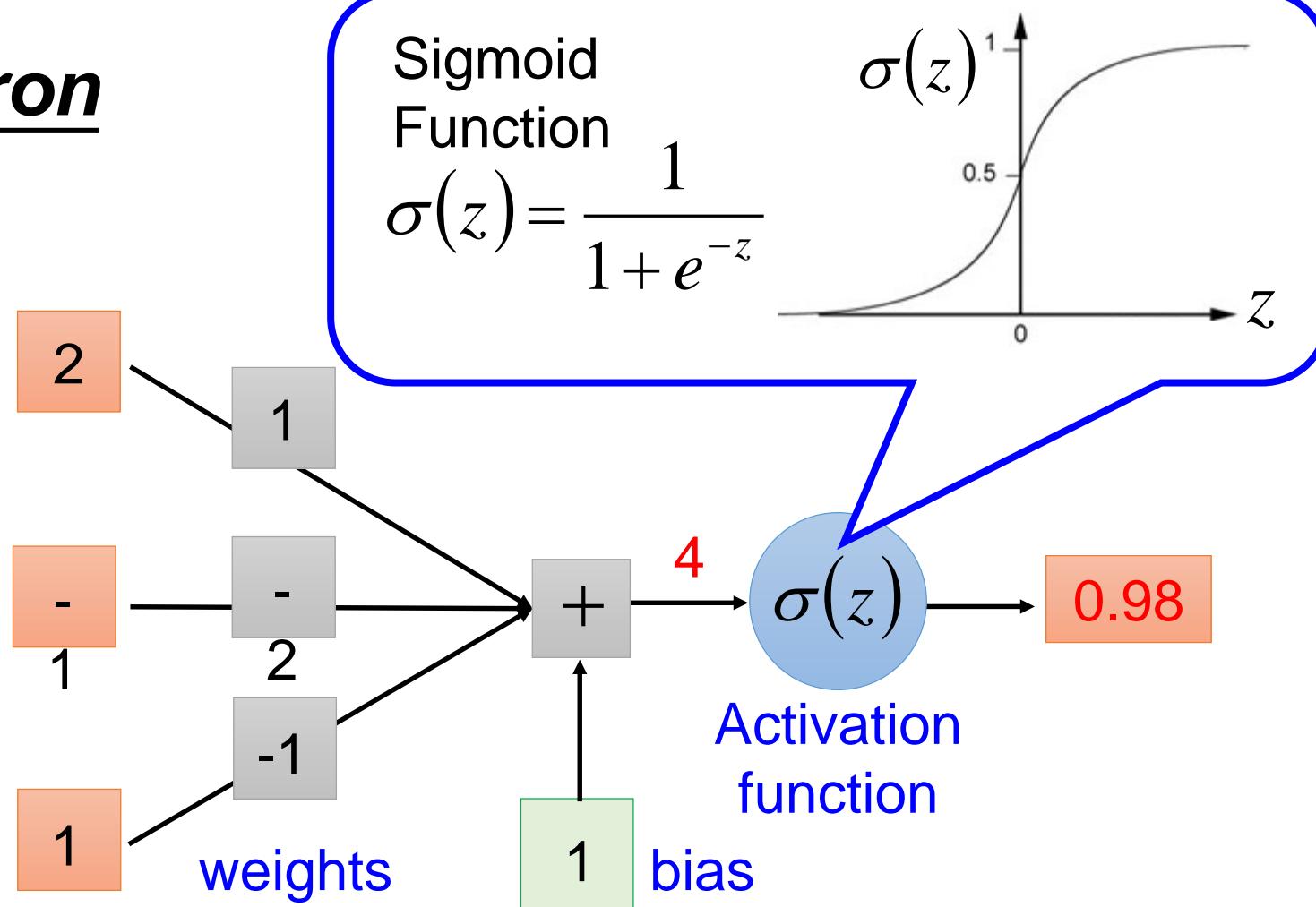
Neuron

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



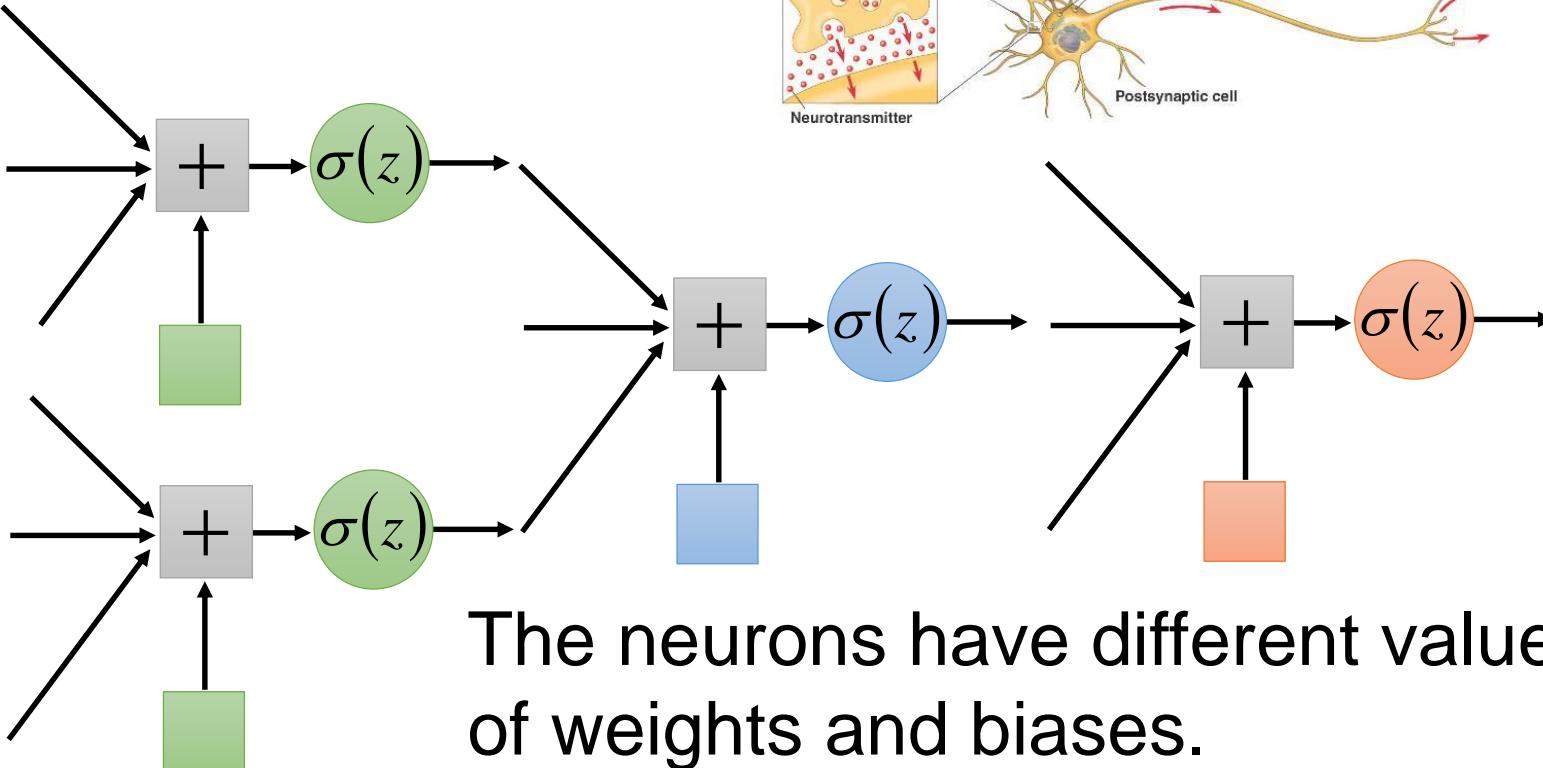
Neural Network

Neuron

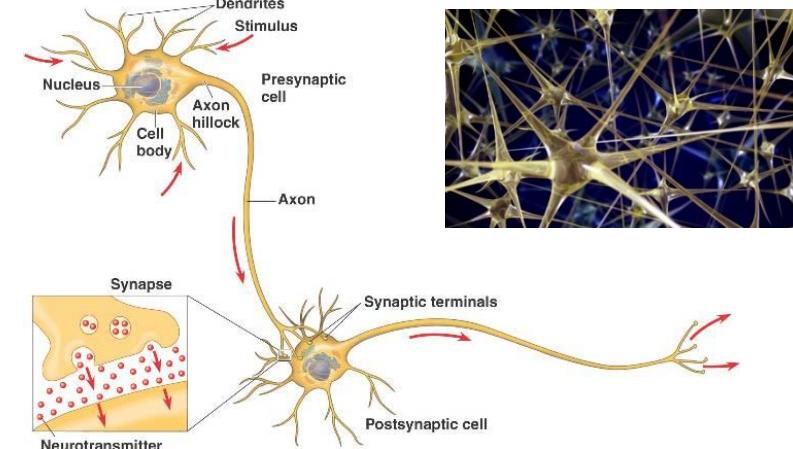


Neural Network

Different connections lead to different network structures



Weights and biases are network parameters θ

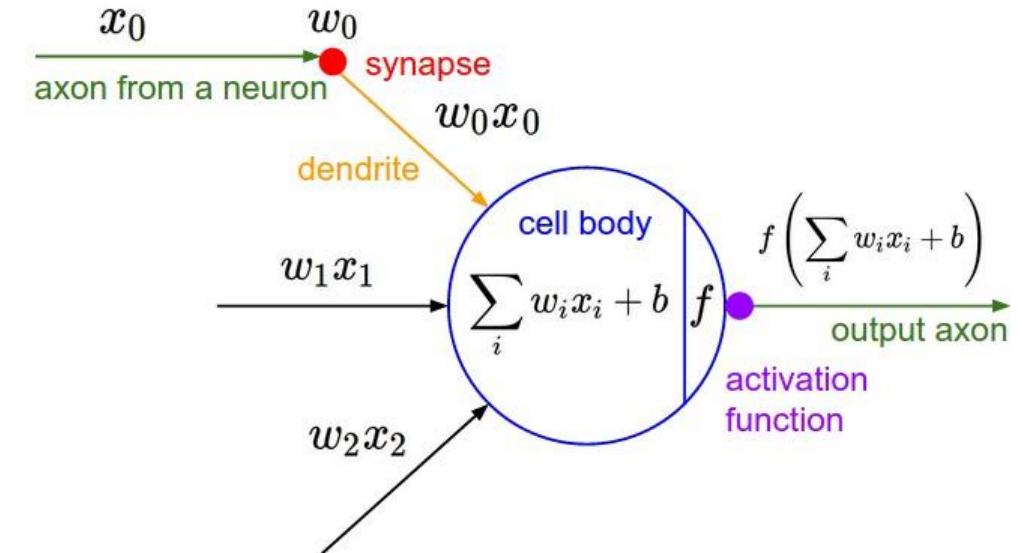
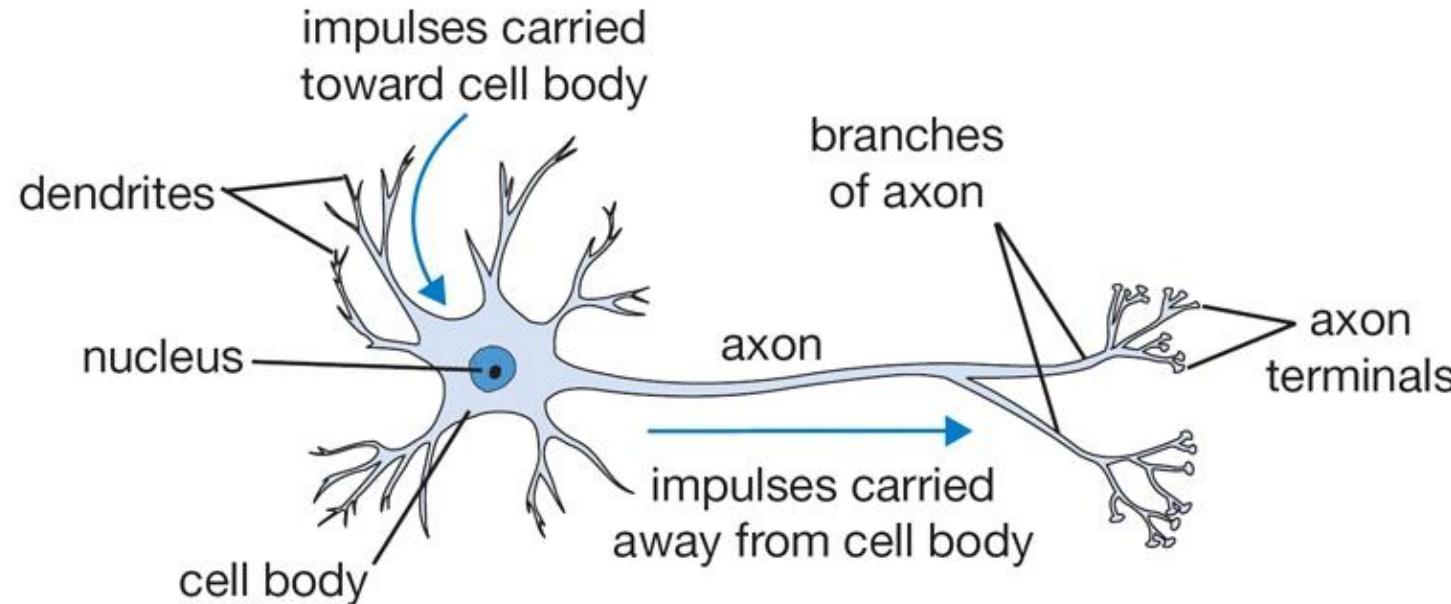


Nervous system



- 神经网络中的神经元的灵感来源于人脑，人体中大约有860亿个神经元neuron，大约有 $10^{14} - 10^{15}$ 突触（synapses）。
- 每个神经元由树突dendrites接收信号轴突axon发射信号. The axon eventually branches out and connects via synapses to dendrites of other neurons.

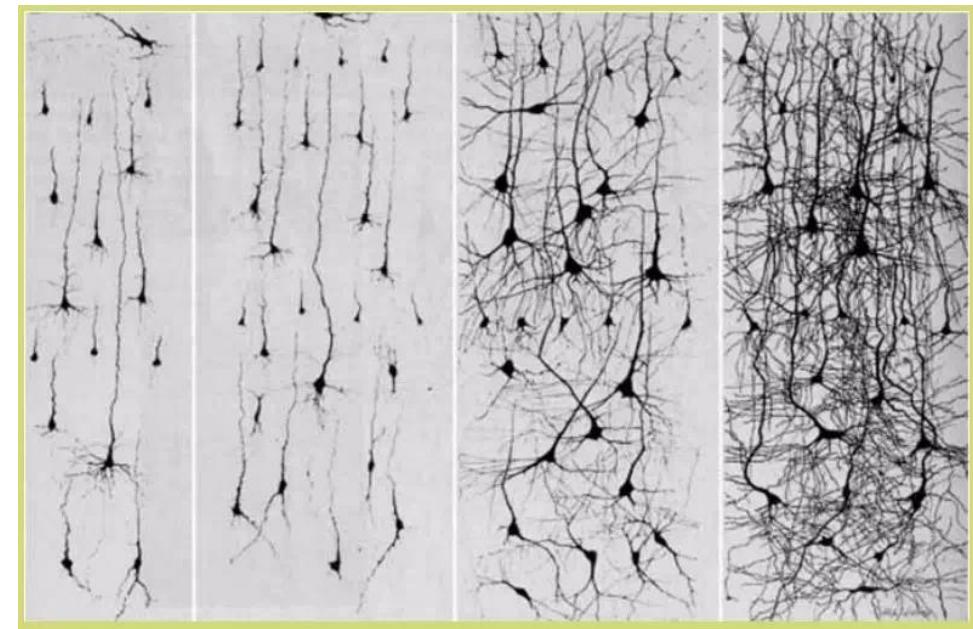
Nervous system



- 神经网络中的神经元的灵感来源于人脑，人体中大约有860亿个神经元neuron，大约有 $10^{14} - 10^{15}$ 突触（synapses）。
- 每个神经元由树突dendrites接收信号轴突axon发射信号. The axon eventually branches out and connects via synapses to dendrites of other neurons.
- 信号x, 权重w(突触强度synaptic strengths, 可训练), 经过细胞体求合之后的信号大于阈值, 神经元就被激活（fire）, 通过axon释放信号（spike）。

神经可塑性

- 肌肉、血管的可塑性；
- “基因难以改变”
- 连续的大脑训练可以增长神经网络连接；
- 《关键20小时快速学会任何技能》
 - 日常的重复！ = “刻意练习”
(deliberate practice)

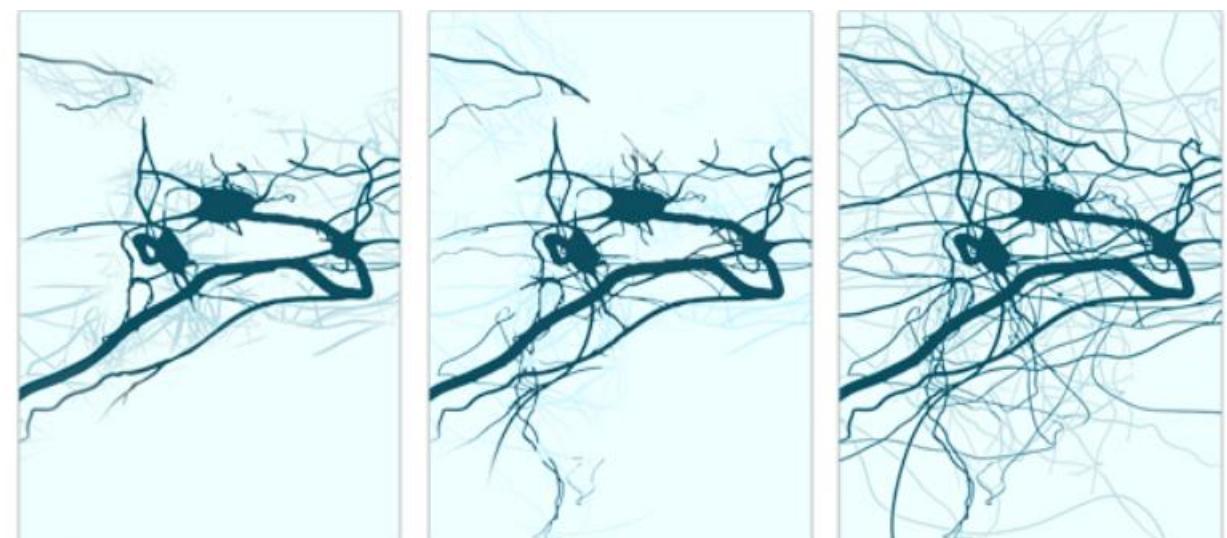


新生儿

1个月

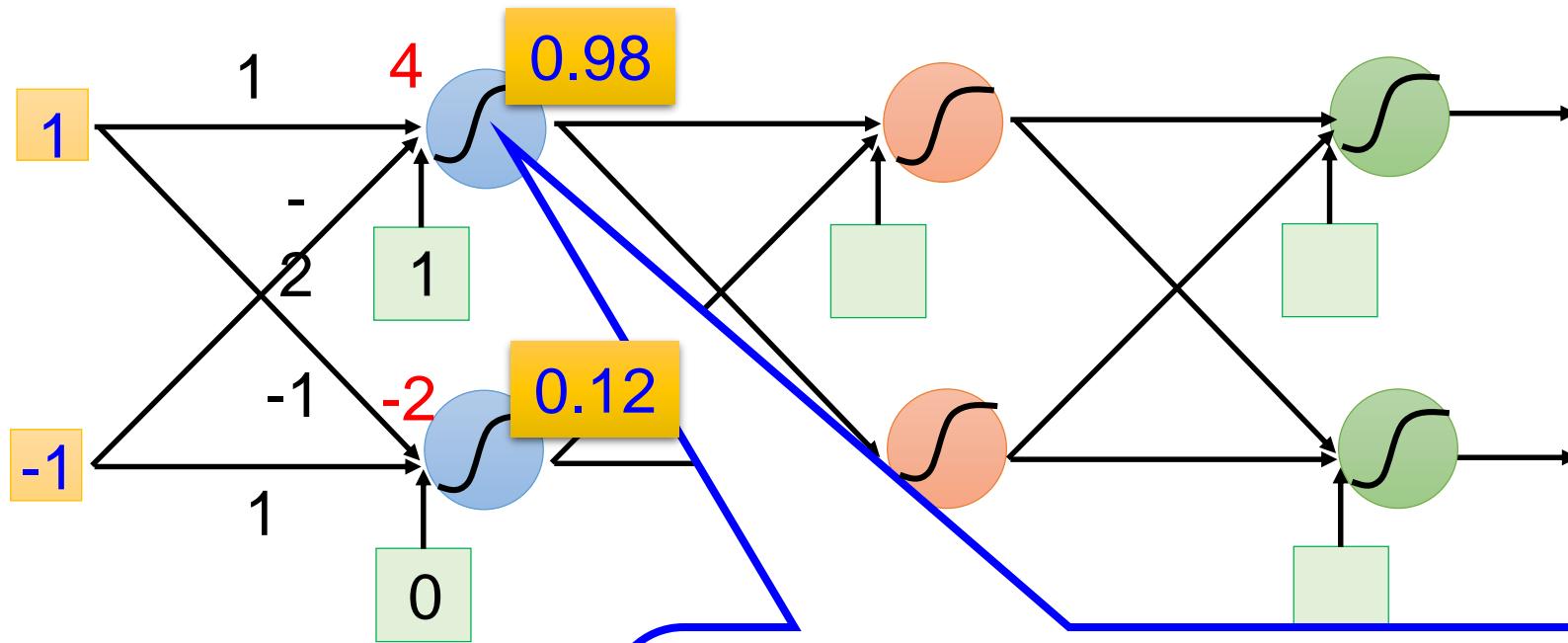
6个月

24个月



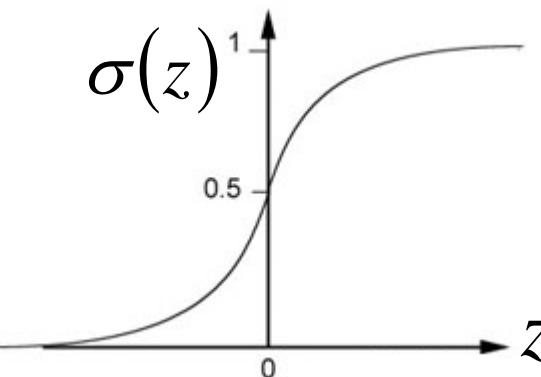
经过2个月刺激（训练）后的脑神经局部

Fully Connect Feedforward Network

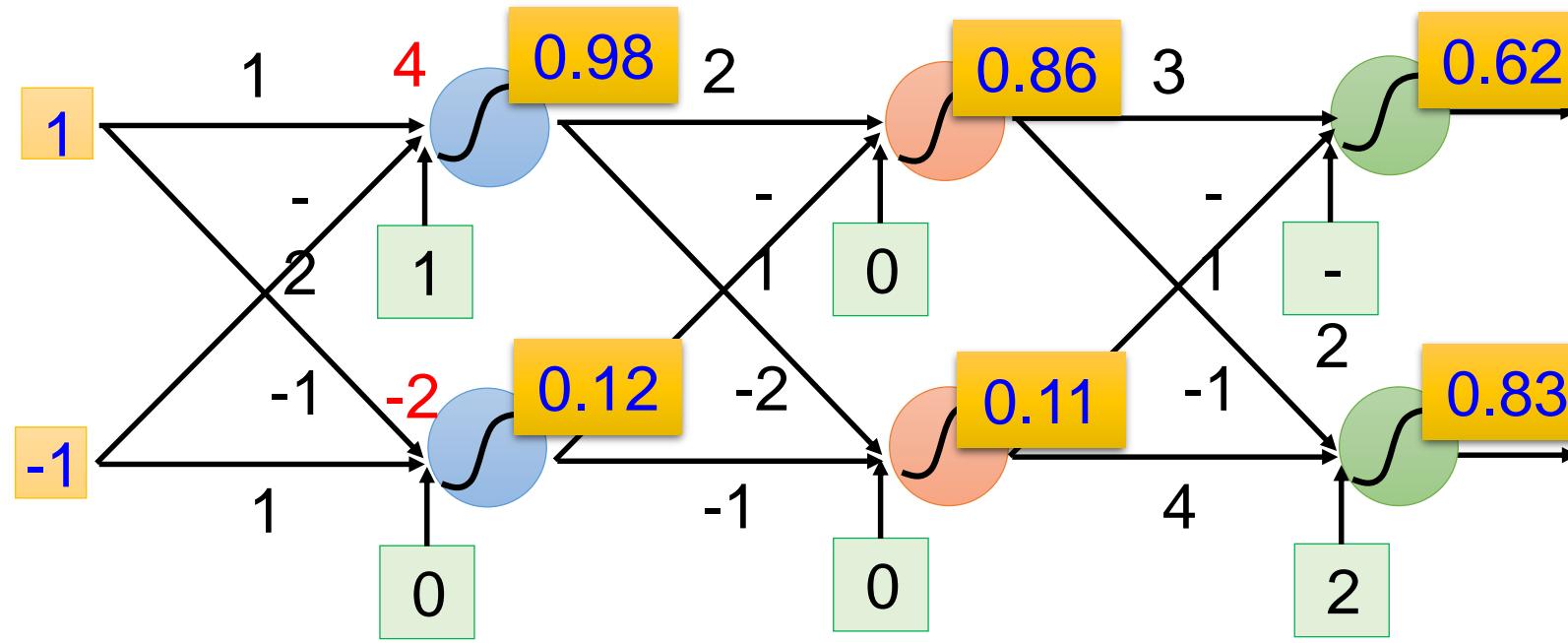


Sigmoid Function

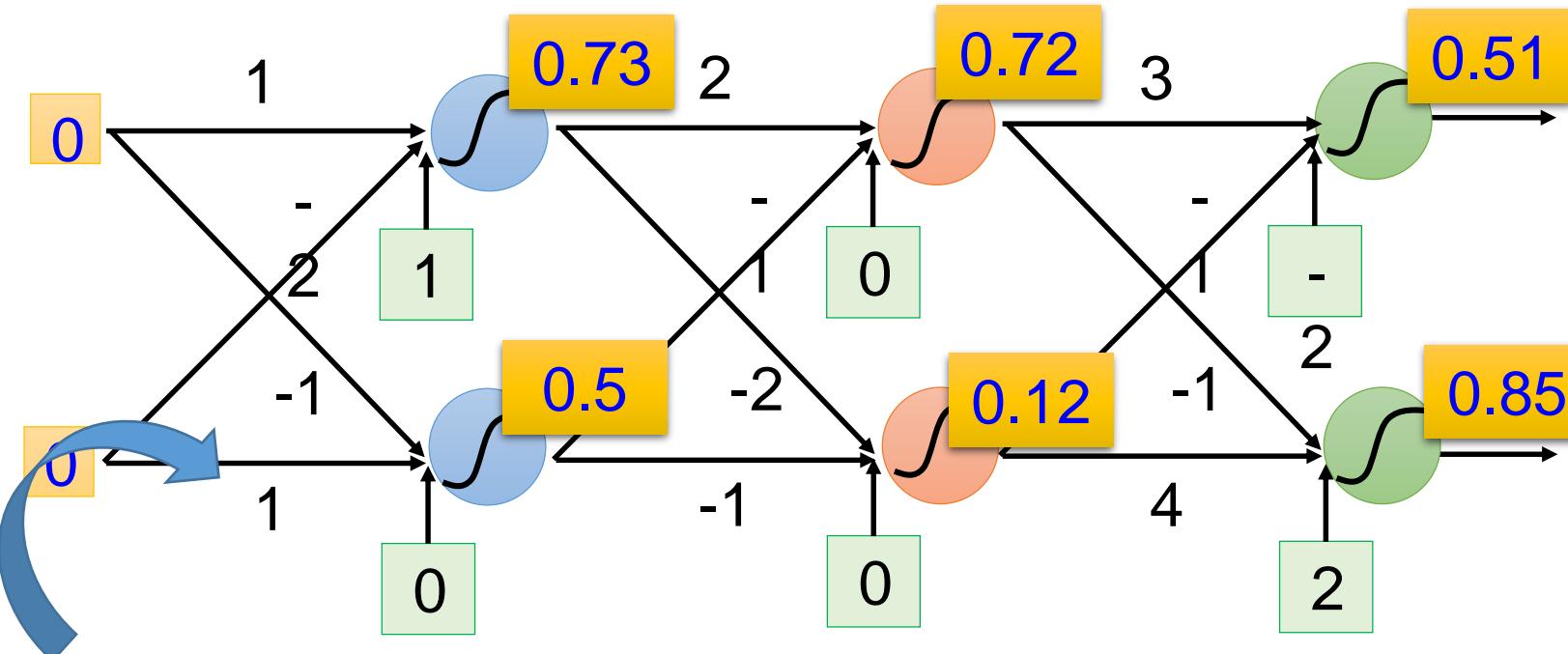
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Fully Connect Feedforward Network



Fully Connect Feedforward Network



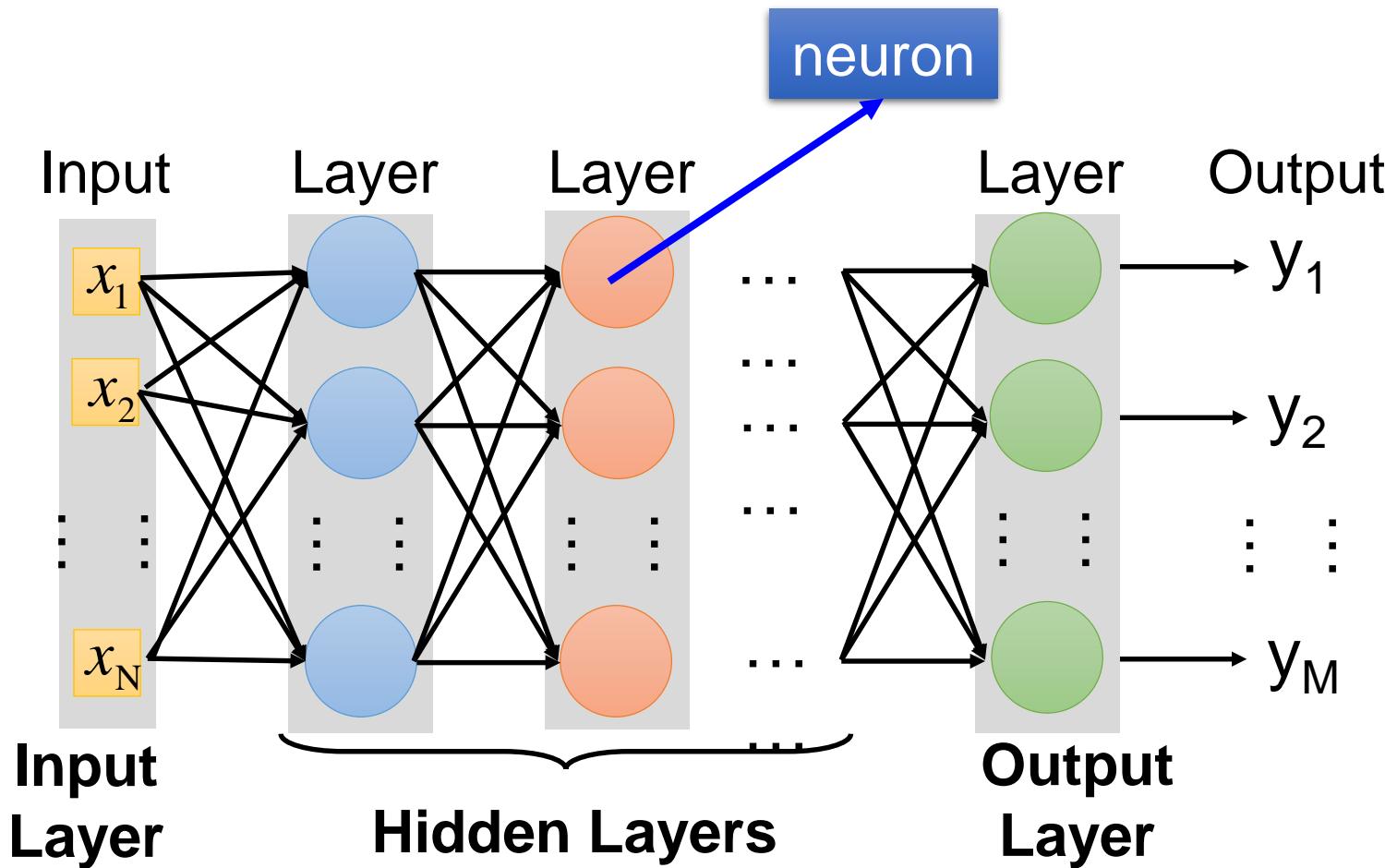
This is a function.
Input vector, output vector

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

Given parameters θ , define a function

Given network structure, define a function set

Fully Connect Feedforward Network



Deep means many hidden layers

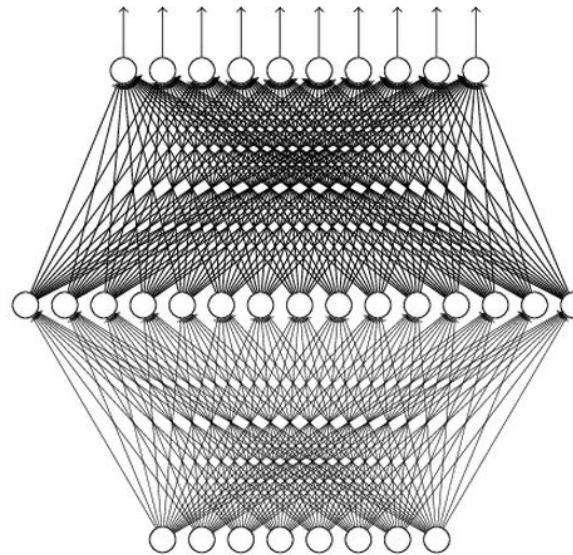
Why Deep? Universality Theorem

Any continuous function f

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

Can be realized by a network
with one hidden layer

(given **enough** hidden
neurons)



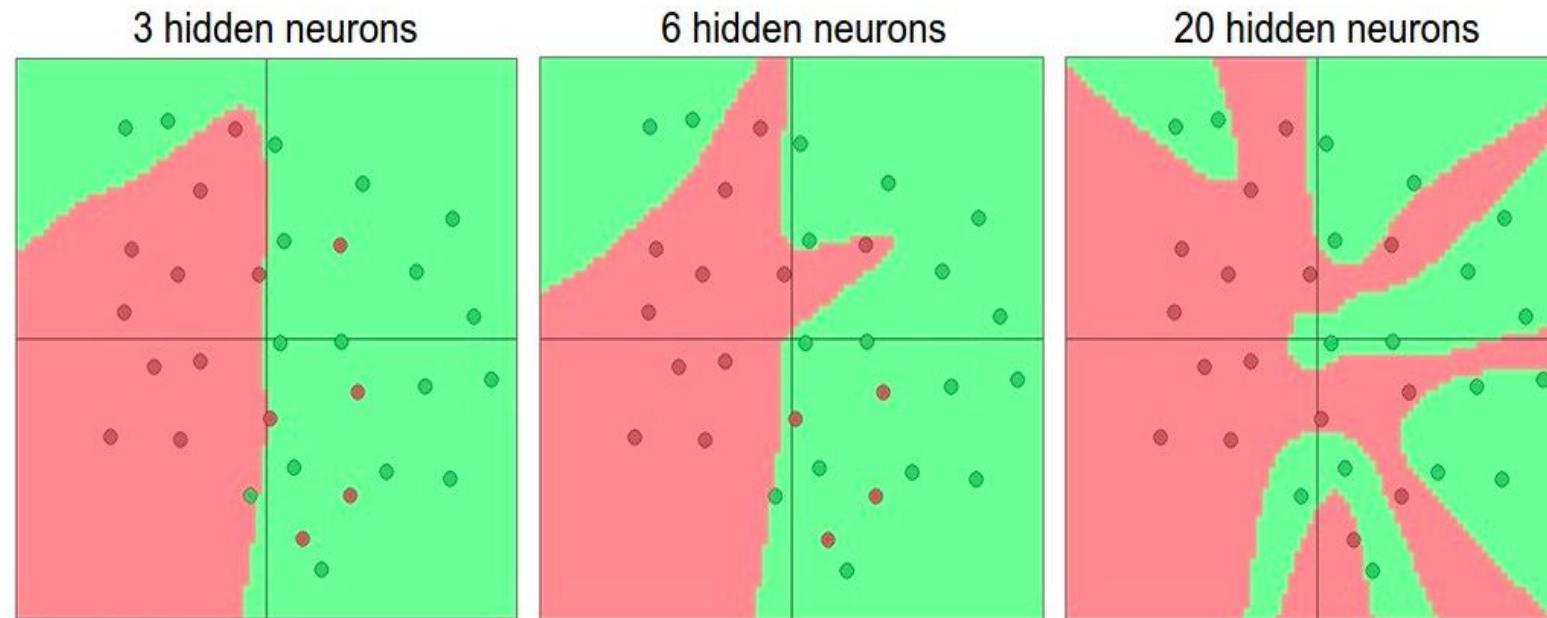
Reference for the reason:
<http://neuralnetworksanddeeplearning.com/chap4.html>

Why “Deep” neural network not “Fat” neural network?

Activation functions

Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function

$$W_1 W_2 x = Wx$$



http://cs231n.github.io/assets/nn1/layer_sizes.jpeg

More layers and neurons can approximate more complex functions

Full list: https://en.wikipedia.org/wiki/Activation_function

Motivations for Deep Architectures: less parameters, less data

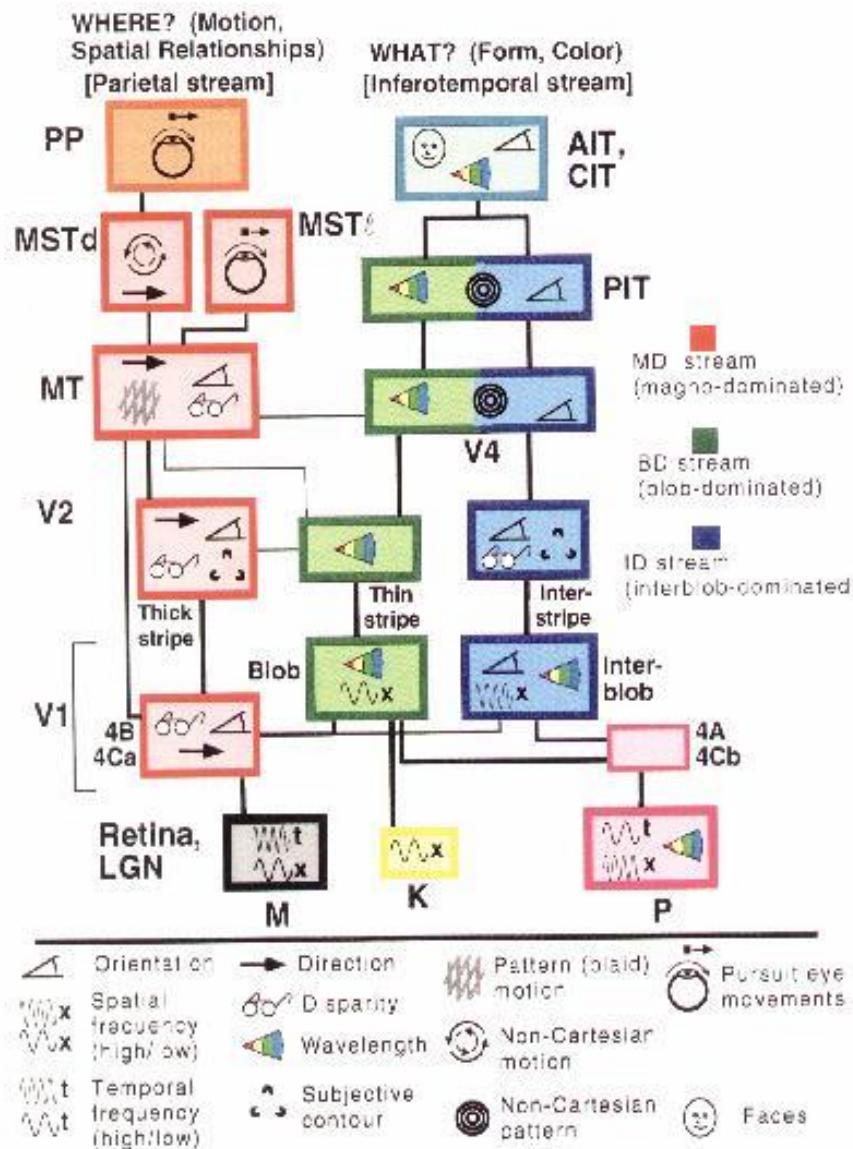
- **Insufficient depth can hurt**

- With shallow architecture (SVM, NB, KNN, etc.), the required number of nodes in the graph (i.e. computations, and also number of parameters, when we try to learn the function) may grow **very large**.
- Many functions that can be represented efficiently with a deep architecture cannot be represented **efficiently** with a shallow one.

- **Cognitive processes seem deep**

- Humans **organize** their ideas and concepts hierarchically.
 - Humans first learn simpler concepts and then compose them to represent more abstract ones.
- Engineers **break-up solutions** into multiple levels of abstraction and processing

The Mammalian Visual Cortex is Hierarchical

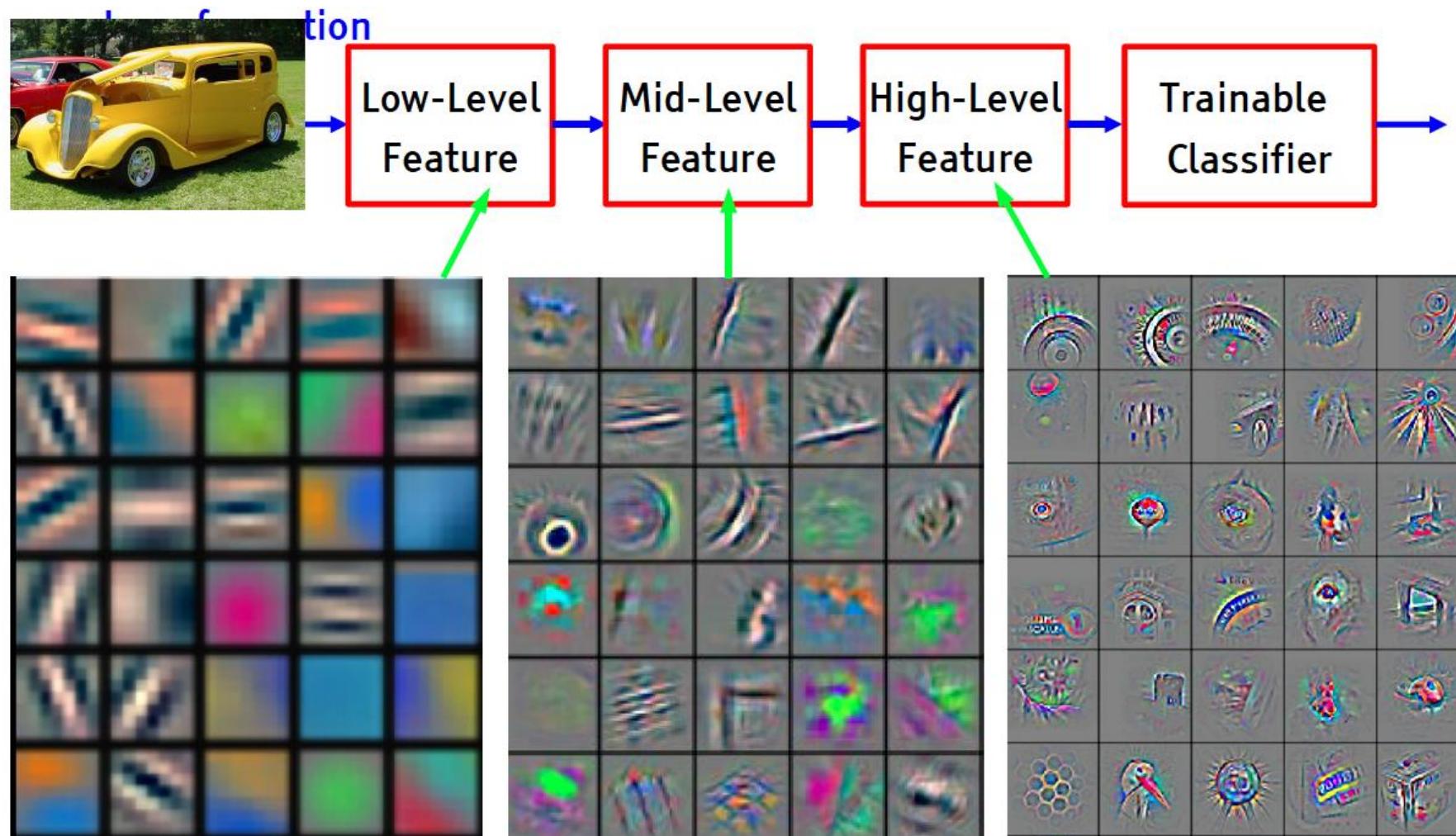


- It is good to be inspired by nature, but not too much.
- We need to understand which details are important, and which details are merely the result of evolution.
- Each module in Deep Learning transforms its input representation into a higher-level one, in a way similar to human cortex.

(van Essen and Gallant, 1994)

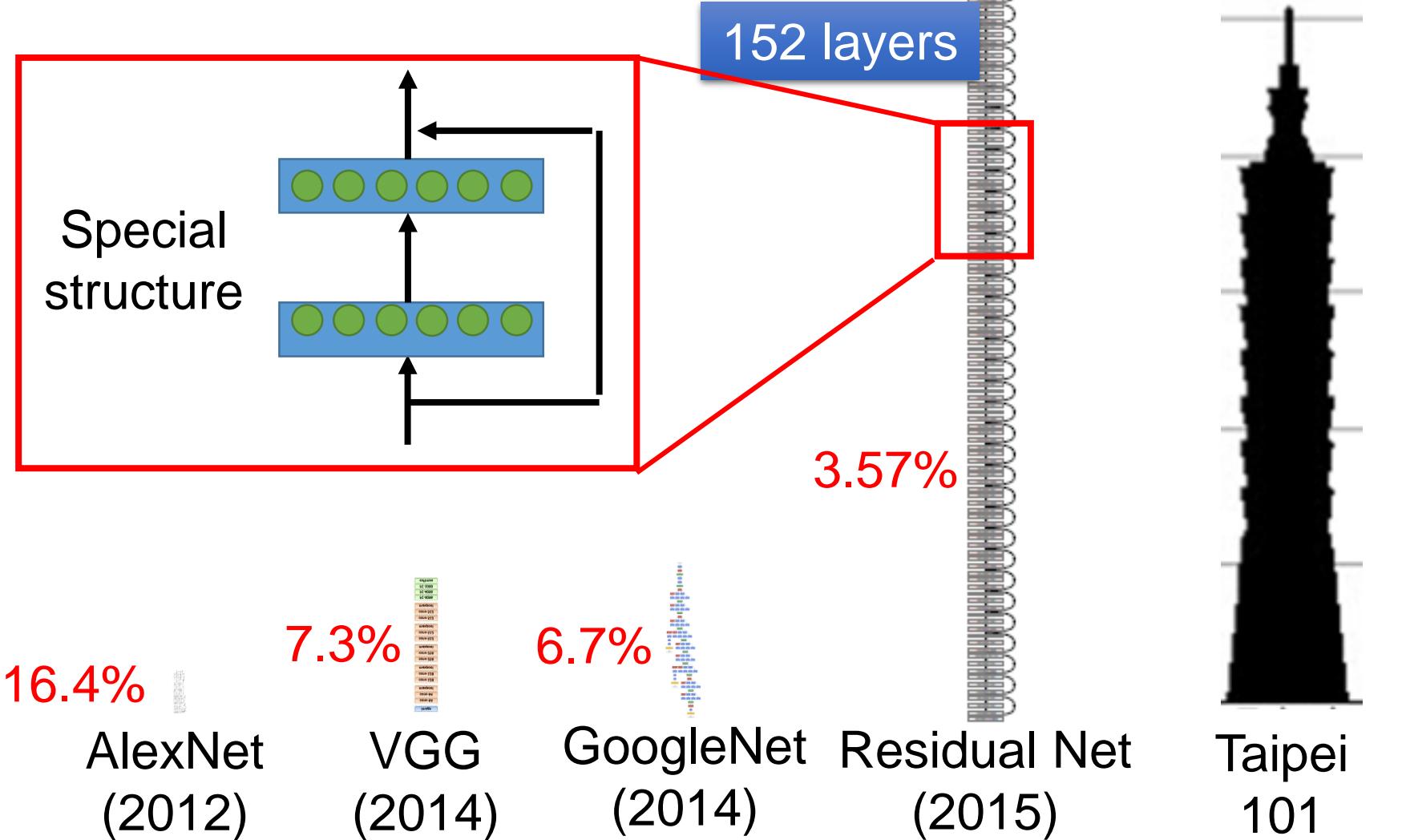
Deep Learning = Learning Hierarchical Representations

Deep learning (a.k.a. representation learning) seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stage of feature learning process.



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

Deep = Many hidden layers



Learning Representations: a challenge for ML, CV, AI, Neuroscience, Cognitive Science...

- How do we learn representations of the perceptual world?

- ▶ How can a perceptual system build itself by looking at the world?
- ▶ How much prior structure is necessary

- ML/AI: how do we learn features or feature hierarchies?

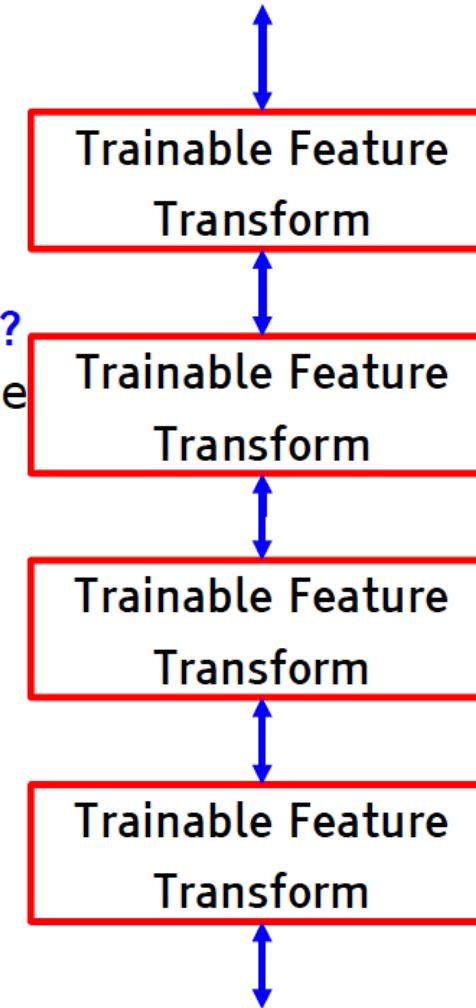
- ▶ What is the fundamental principle? What is the learning algorithm? What is the architecture?

- Neuroscience: how does the cortex learn perception?

- ▶ Does the cortex “run” a single, general learning algorithm? (or a small number of them)

- CogSci: how does the mind learn abstract concepts on top of less abstract ones?

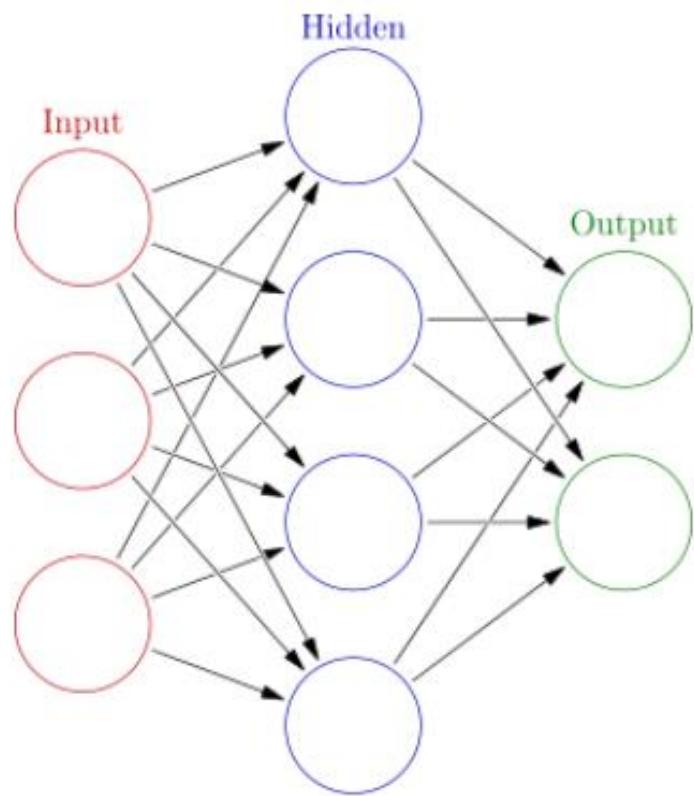
- Deep Learning addresses the problem of learning hierarchical representations with a single algorithm



Output Layer

- Softmax layer as the output layer

Output Layer

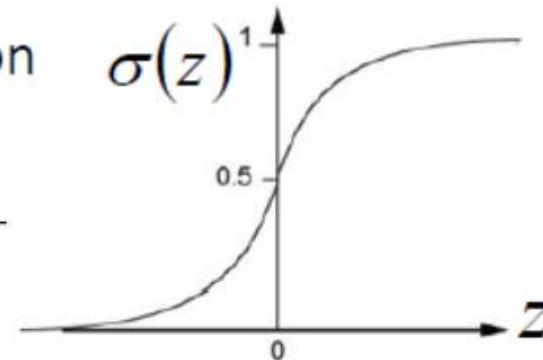


Ordinary Layer

$$\begin{aligned} z_1 &\rightarrow \sigma & y_1 = \sigma(z_1) \\ z_2 &\rightarrow \sigma & y_2 = \sigma(z_2) \\ z_3 &\rightarrow \sigma & y_3 = \sigma(z_3) \end{aligned}$$

Sigmoid Function

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



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

May not be easy to interpret

- Softmax layer as the output layer

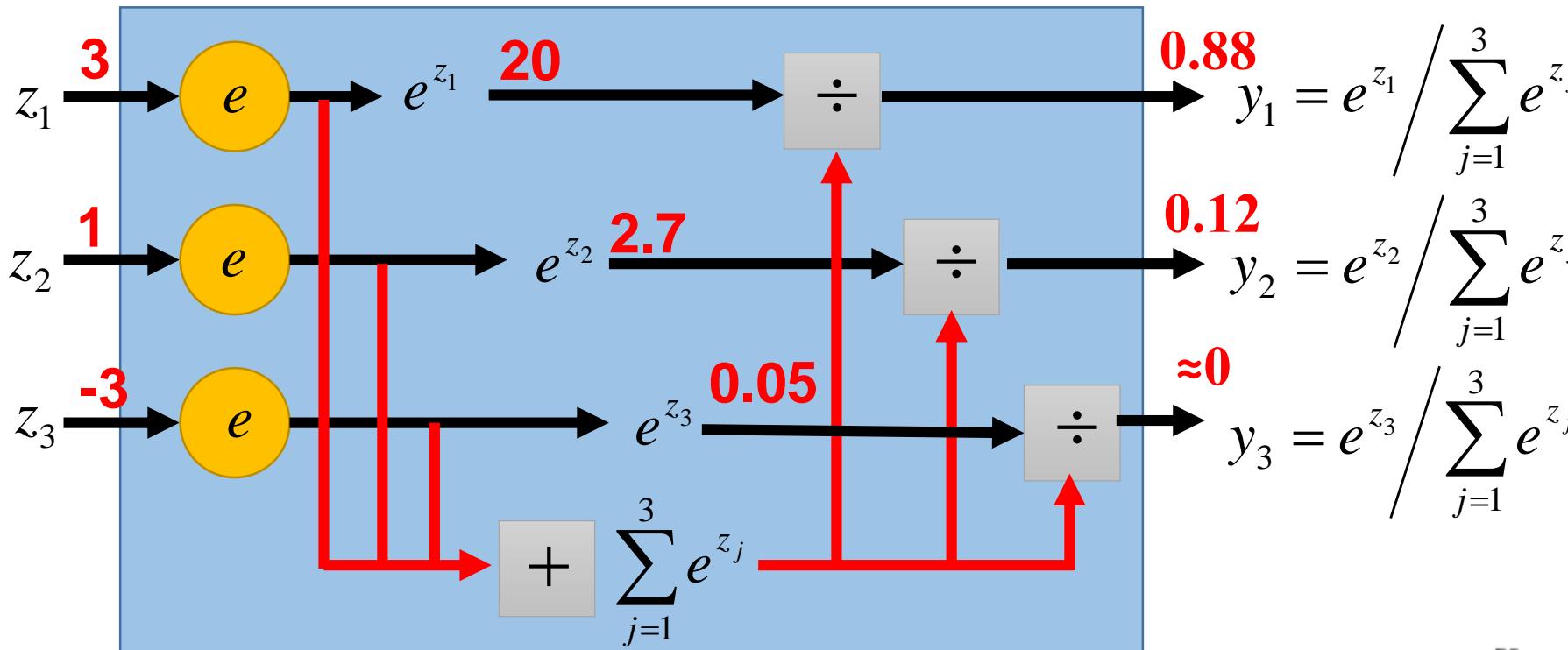
Output Layer

- Softmax layer as the output layer

Probability:

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

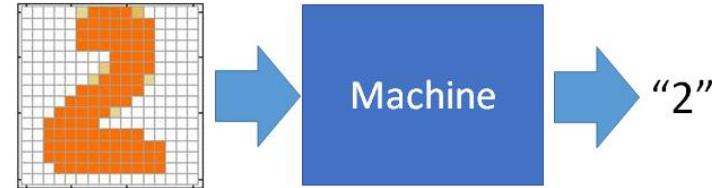
Softmax Layer



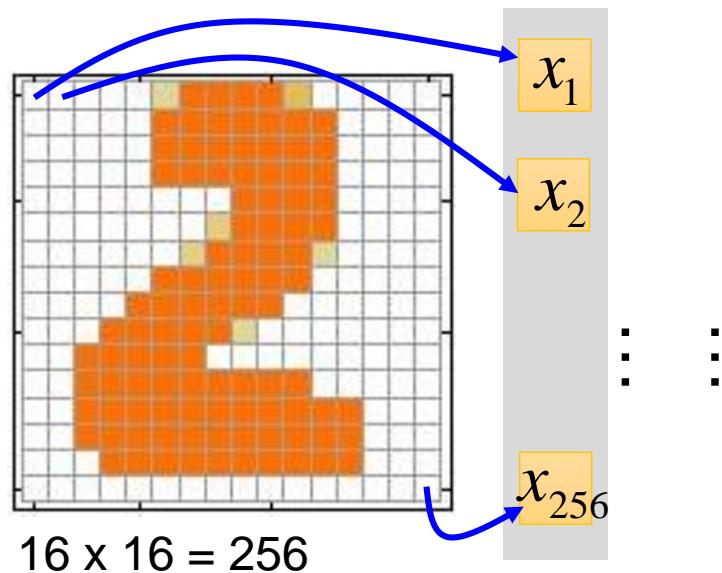
$$\sigma : \mathbb{R}^K \rightarrow \left\{ z \in \mathbb{R}^K \mid z_i > 0, \sum_{i=1}^K z_i = 1 \right\}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K$$

Example Application

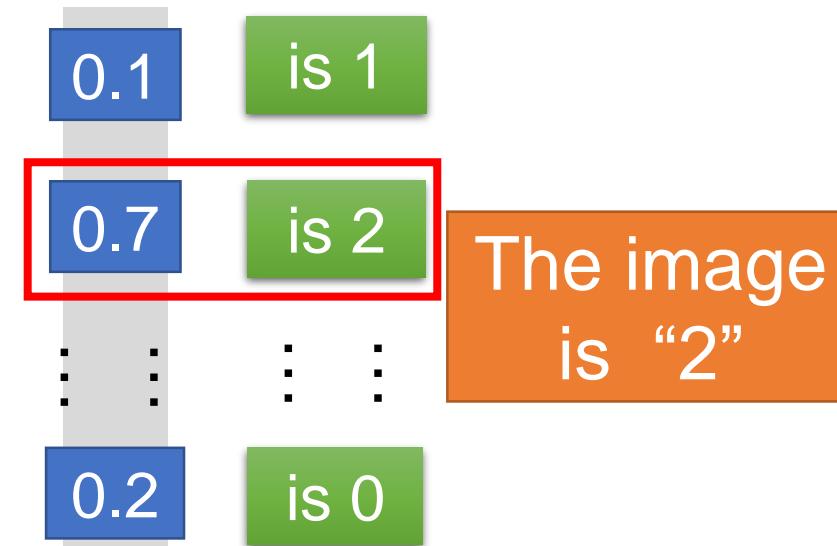


Input



Ink \rightarrow 1
No ink \rightarrow 0

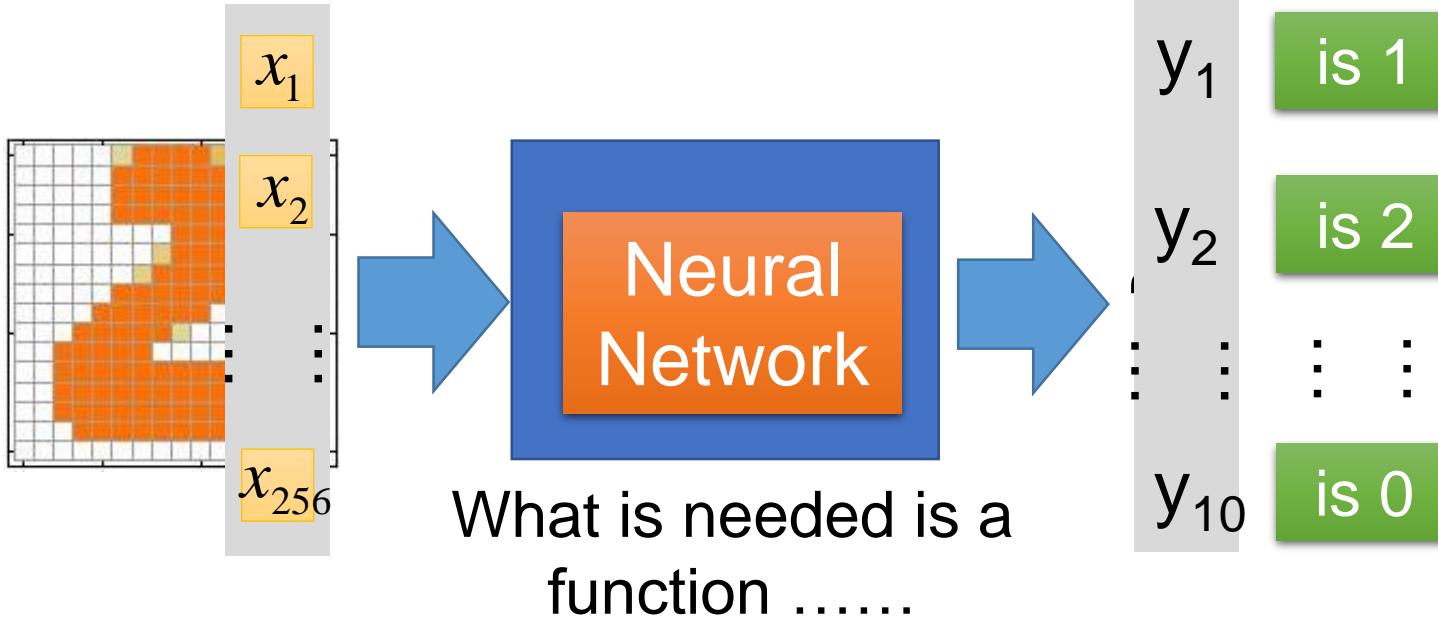
Output



Each dimension represents the confidence of a digit.

Example Application

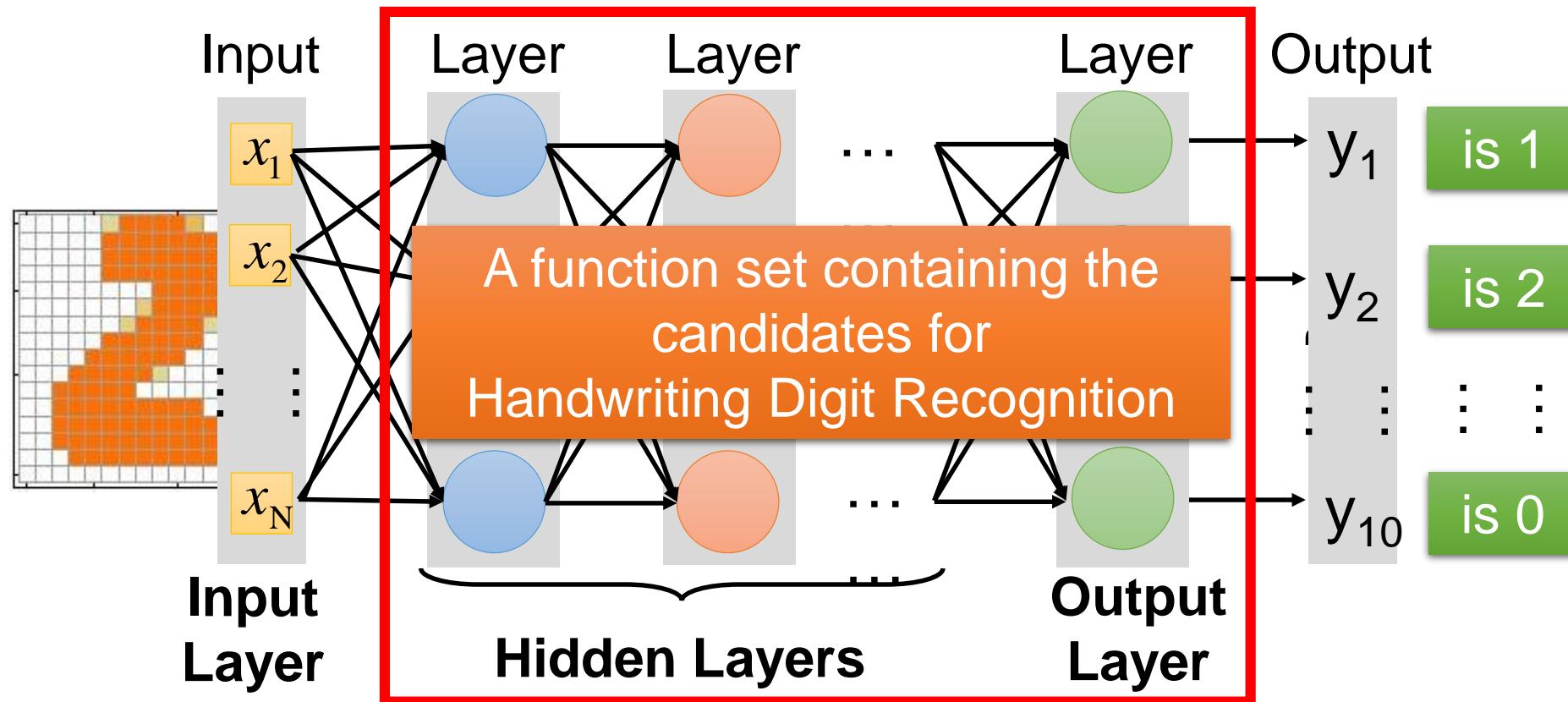
- Handwriting Digit Recognition



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

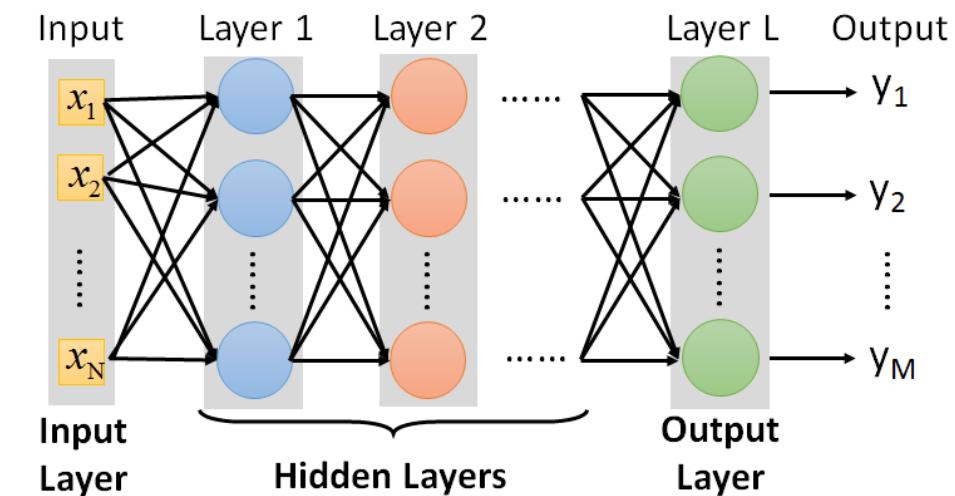
+

Intuition

- Q: Can we design the network structure?

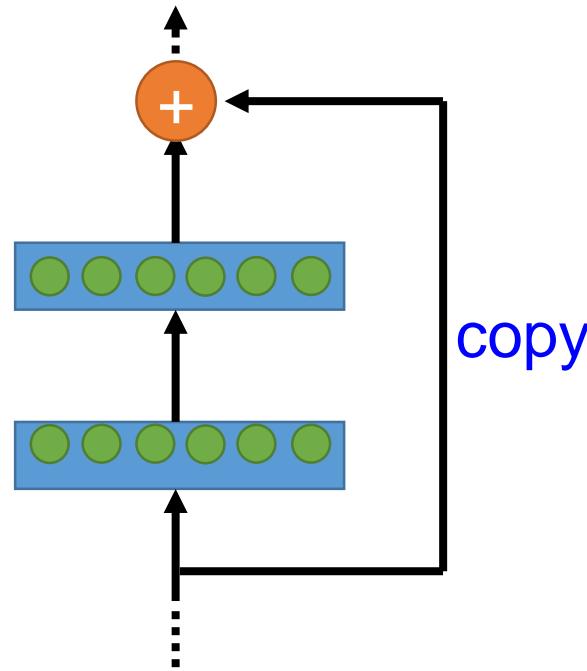
Convolutional Neural Network
(CNN) in the next lecture

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

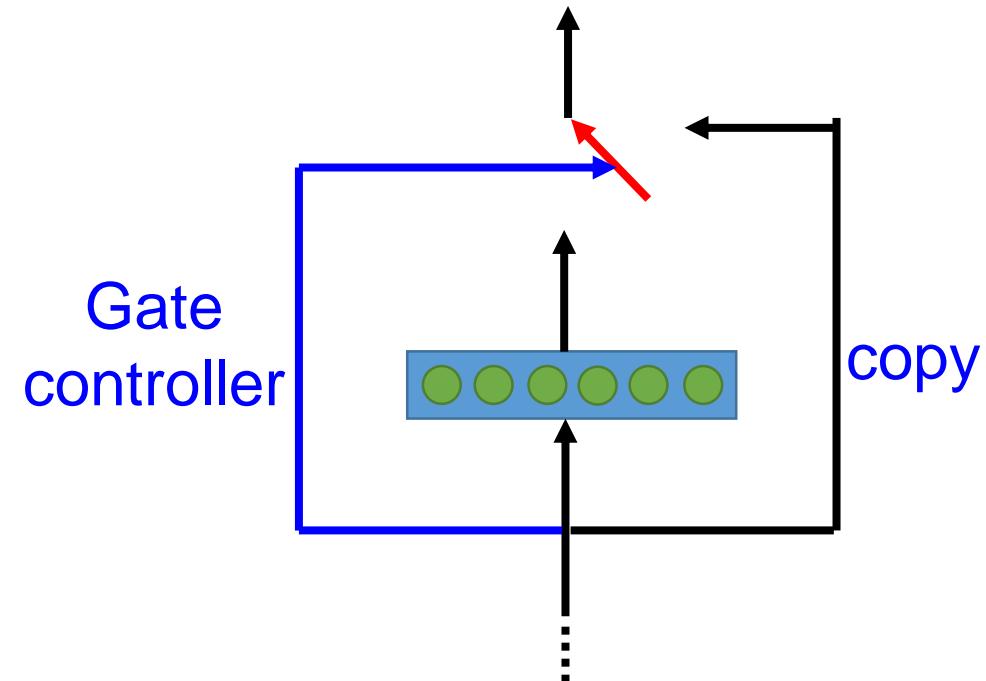


Highway Network

- Residual Network



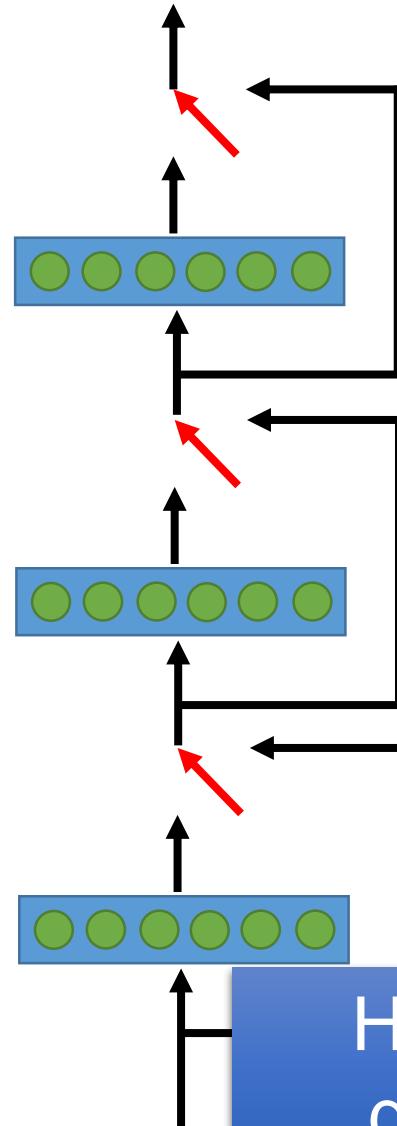
- Highway Network



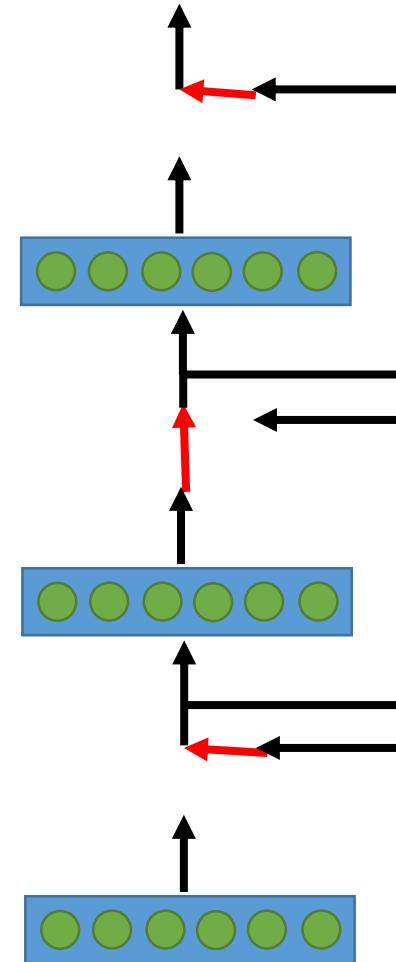
Deep Residual Learning for Image Recognition
<http://arxiv.org/abs/1512.03385>

Training Very Deep Networks
<https://arxiv.org/pdf/1507.06228v2.pdf>

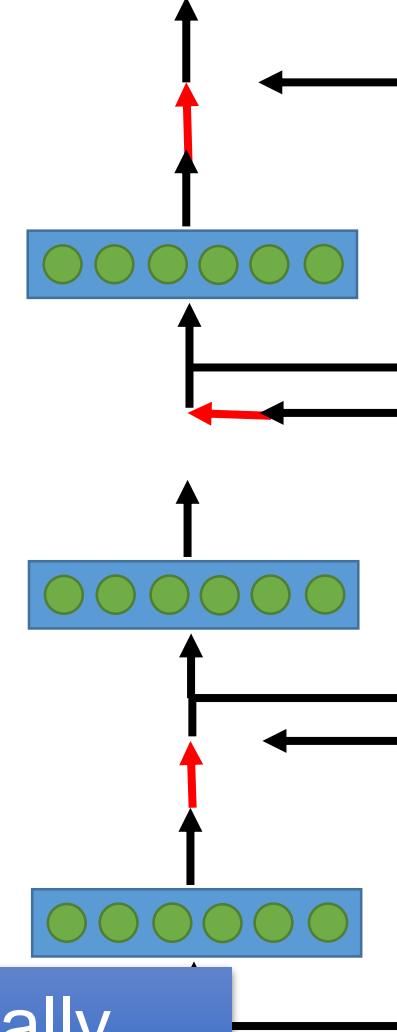
output layer



output layer



output layer



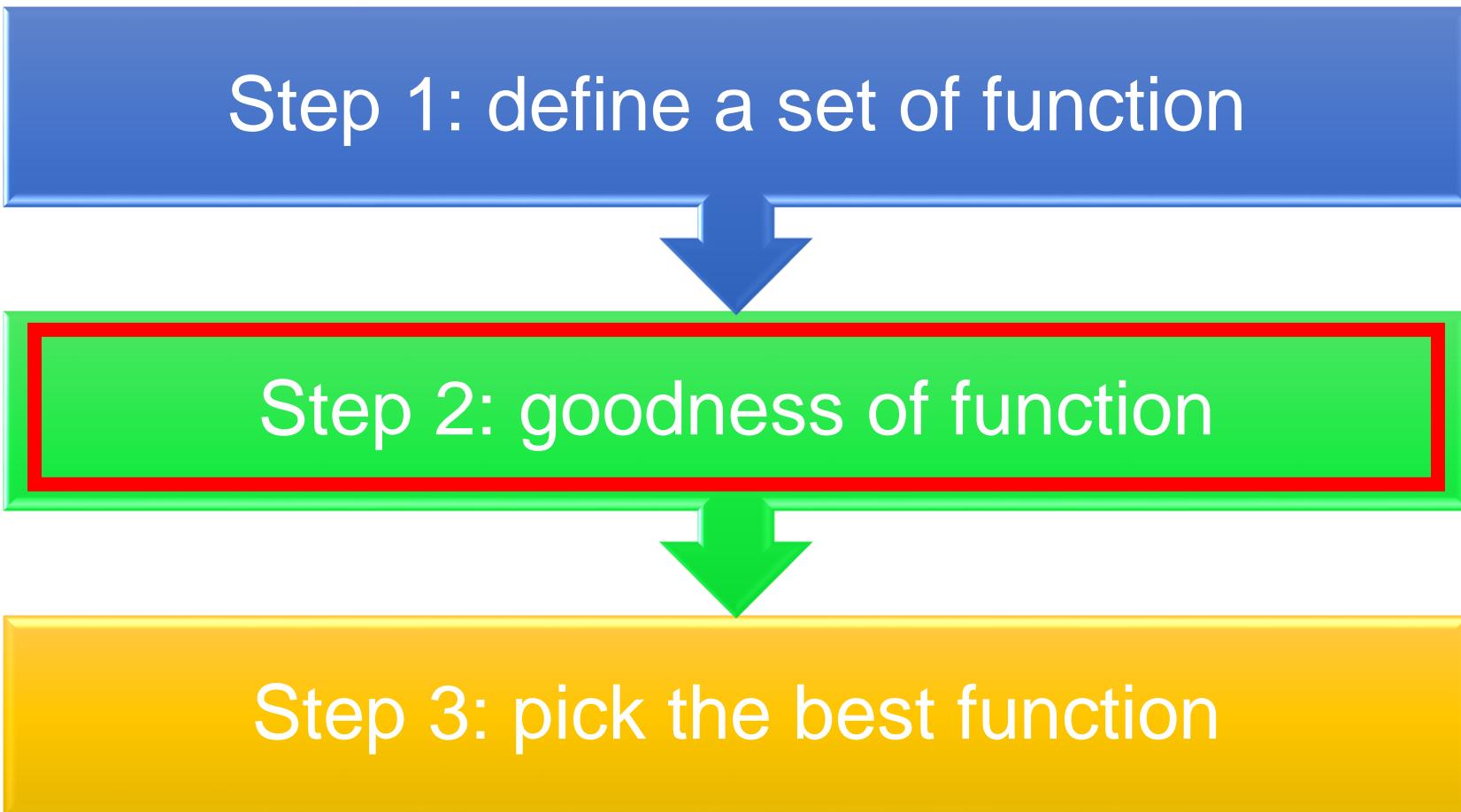
Input layer

Input layer

Input layer

Highway Network automatically
determines the layers needed!

Three Steps for Deep Learning



Training Data

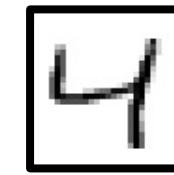
- Preparing training data: images and their labels



“5”



“0”



“4”



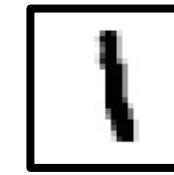
“1”



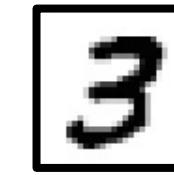
“9”



“2”



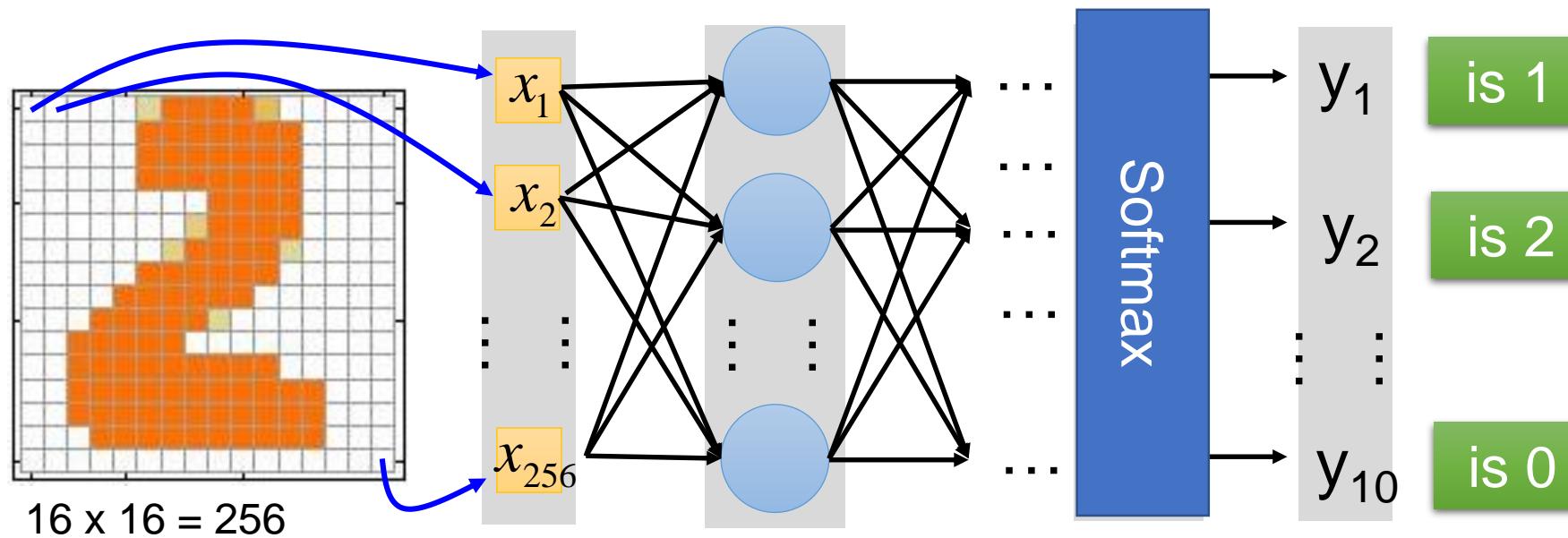
“1”



“3”

The learning target is defined
on the training data.

Learning Target



Ink \rightarrow 1
No ink \rightarrow 0

The learning target is

Input:

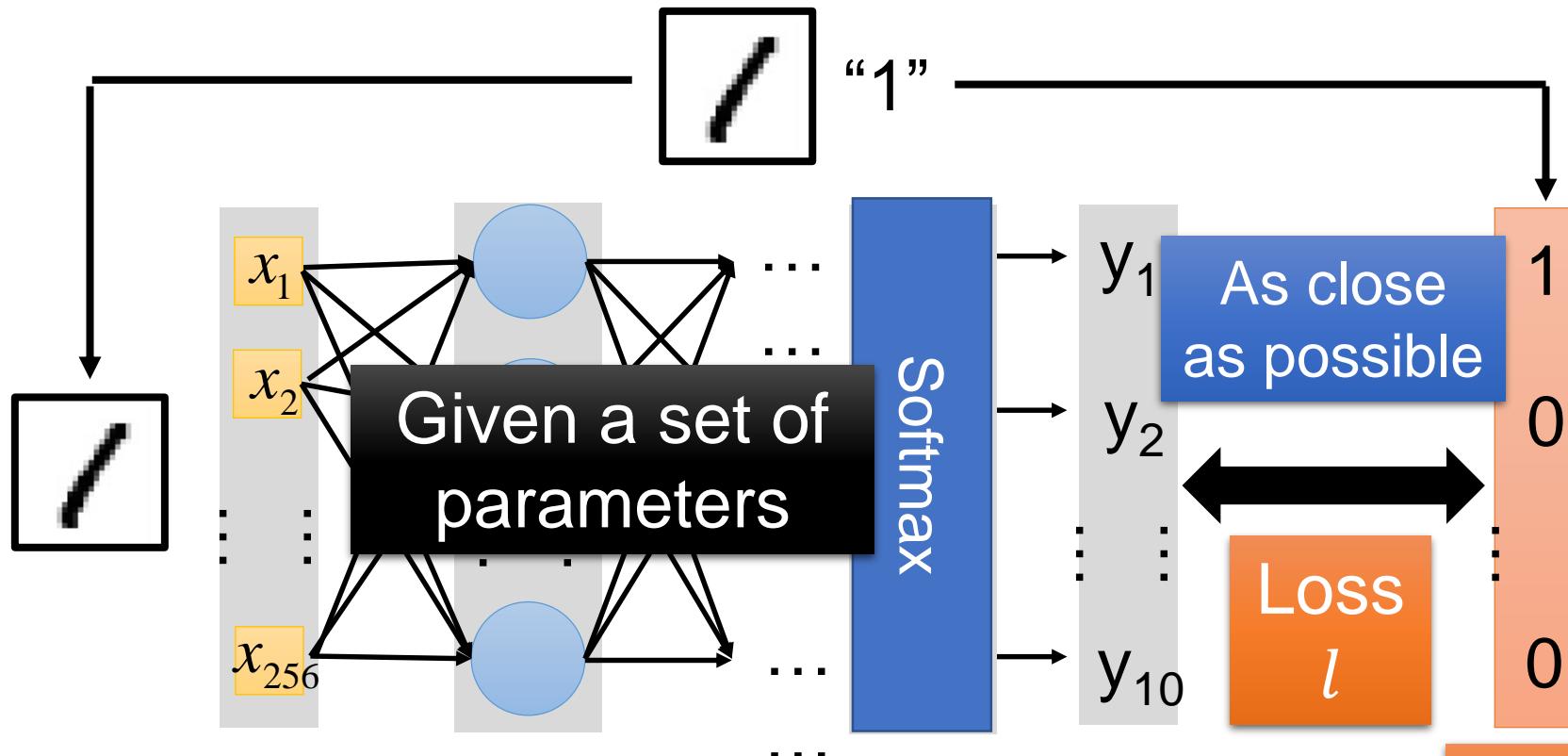
y_1 has the maximum value

Input:

y_2 has the maximum value

LOSS

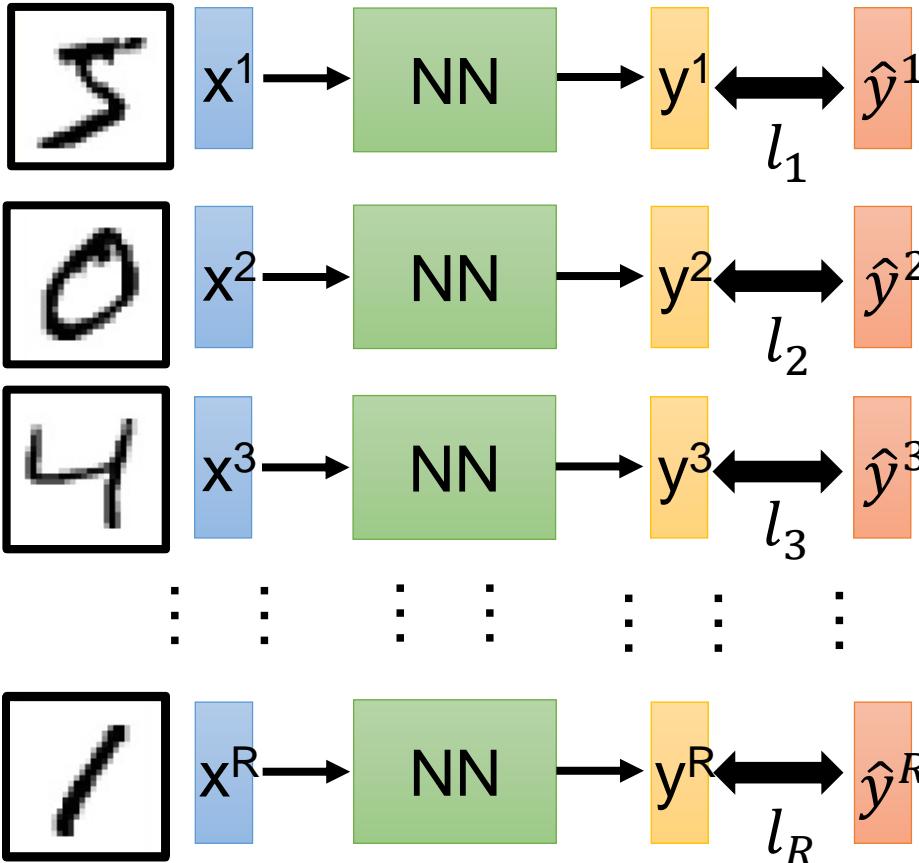
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data ...



Total Loss:

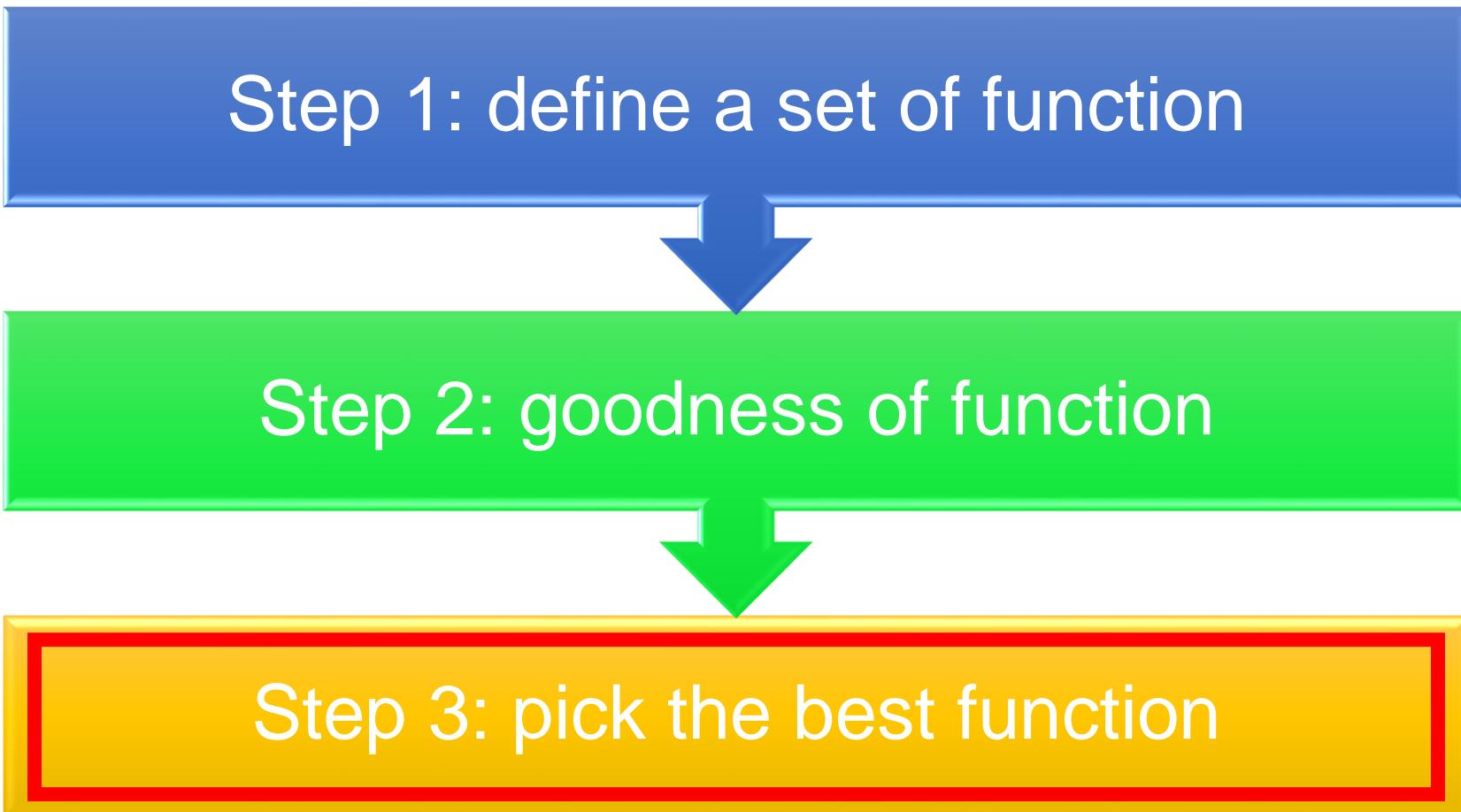
$$L = \sum_{r=1}^R l_r$$

As small as possible

Find a function in function set that minimizes total loss L

Find the network parameters θ^* that minimize total loss L

Three Steps for Deep Learning



How to pick the best function

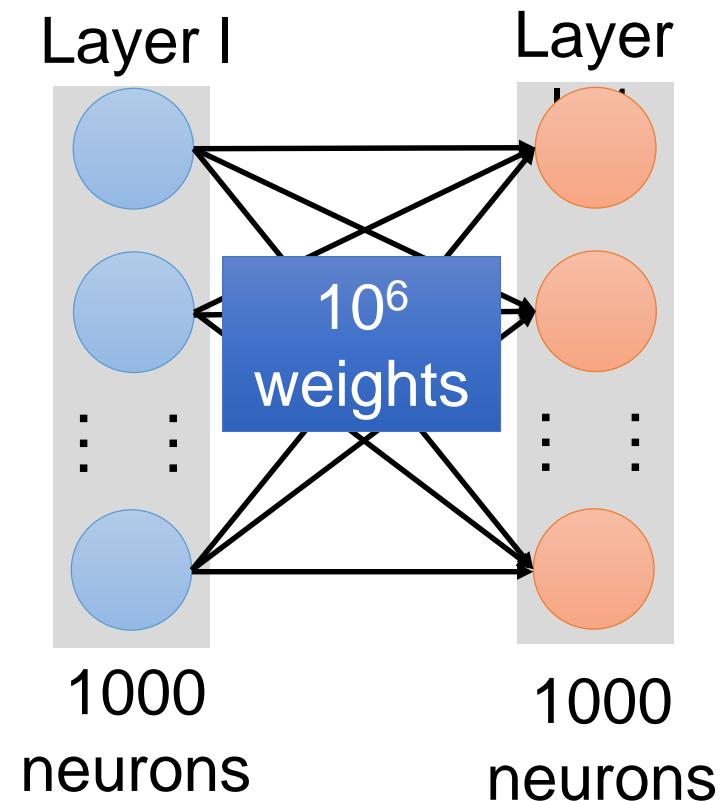
Find network parameters θ^* that minimize total loss L

Enumerate all possible values

Network parameters $\theta =$
 $\{w_1, w_2, w_3, \dots, b_1, b_2, b_3, \dots\}$

Millions of parameters

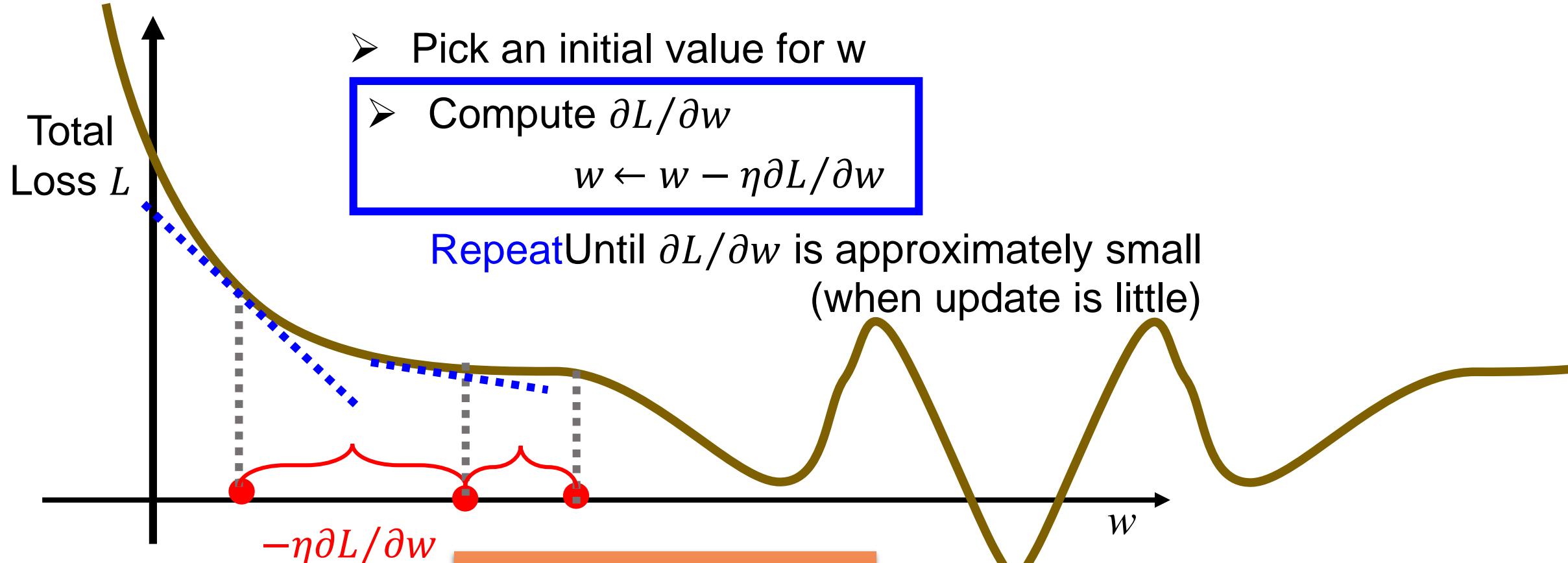
E.g. speech recognition: 8 layers and
1000 neurons each layer



Gradient Descent

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

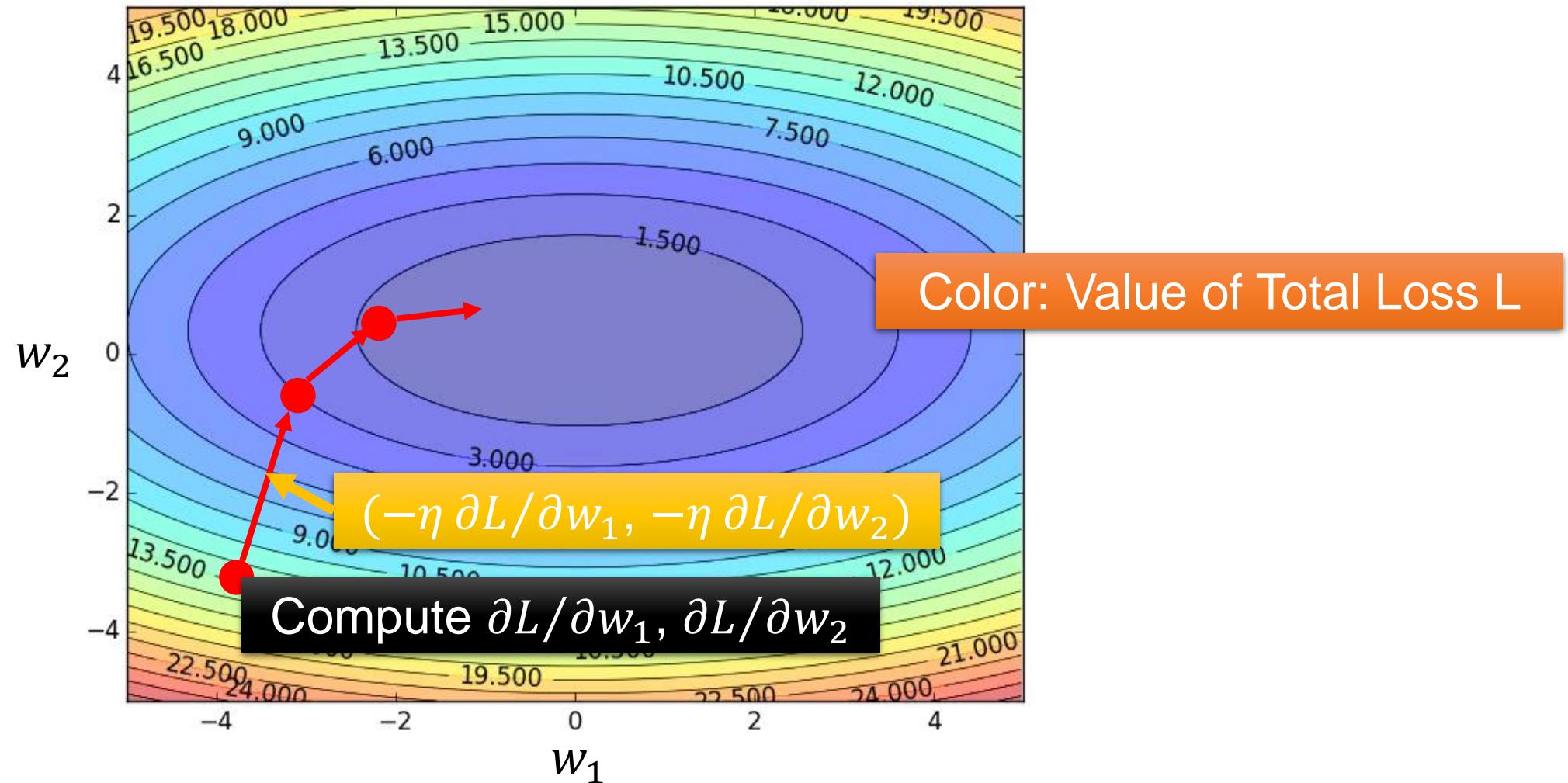
Find network parameters θ^* that minimize total loss L



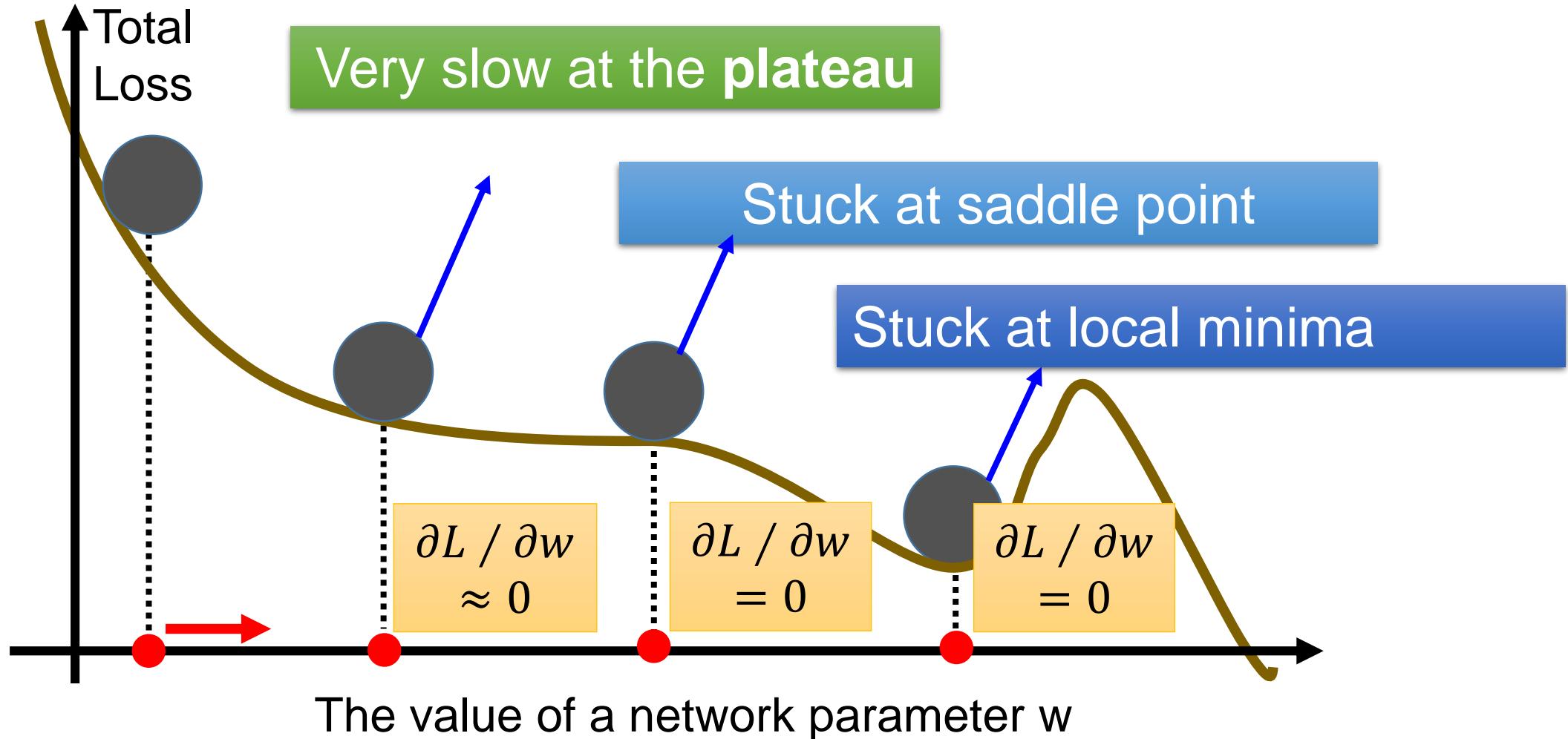
η is called
“*learning rate*”

Gradient Descent

Hopfully, we would reach a minima

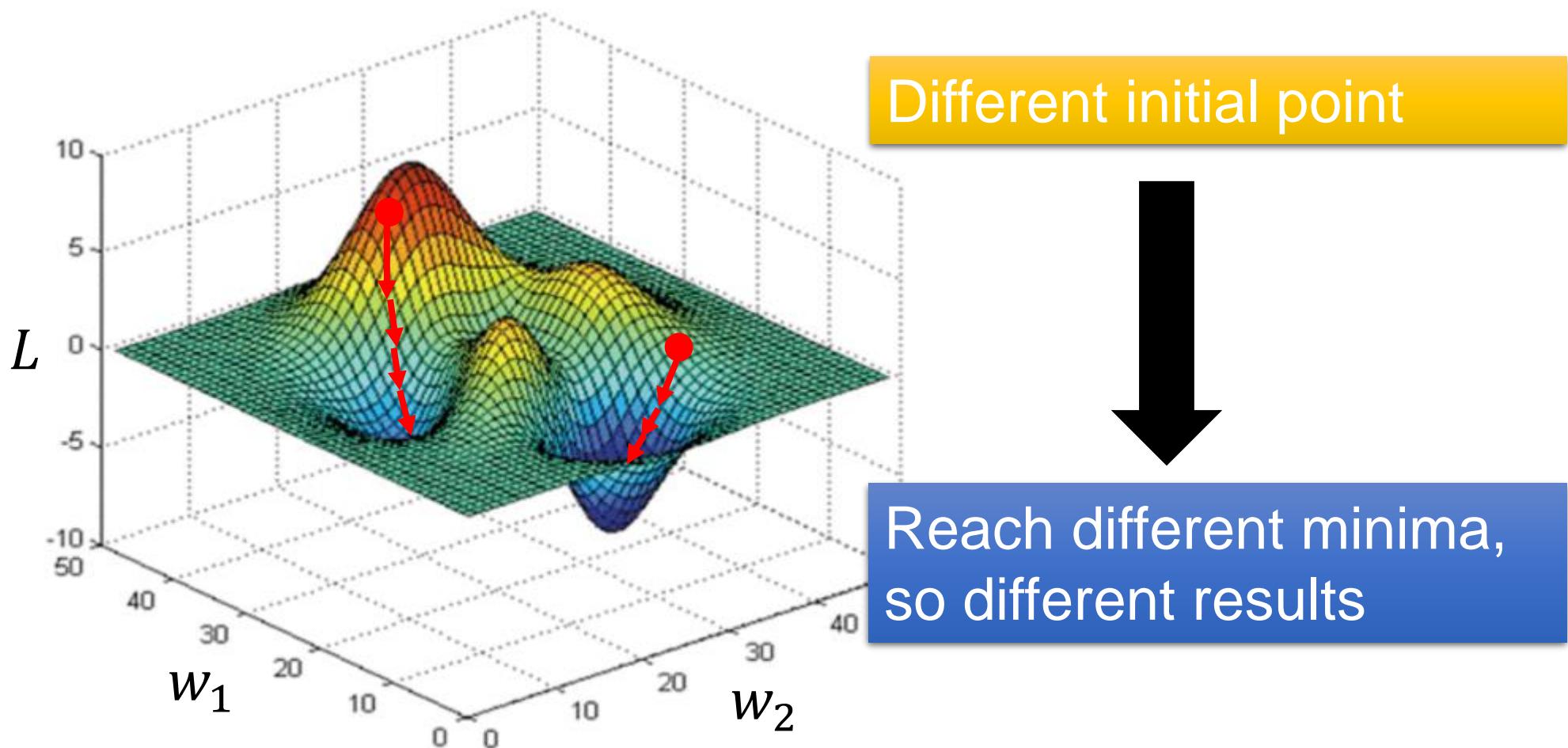


Local Minima

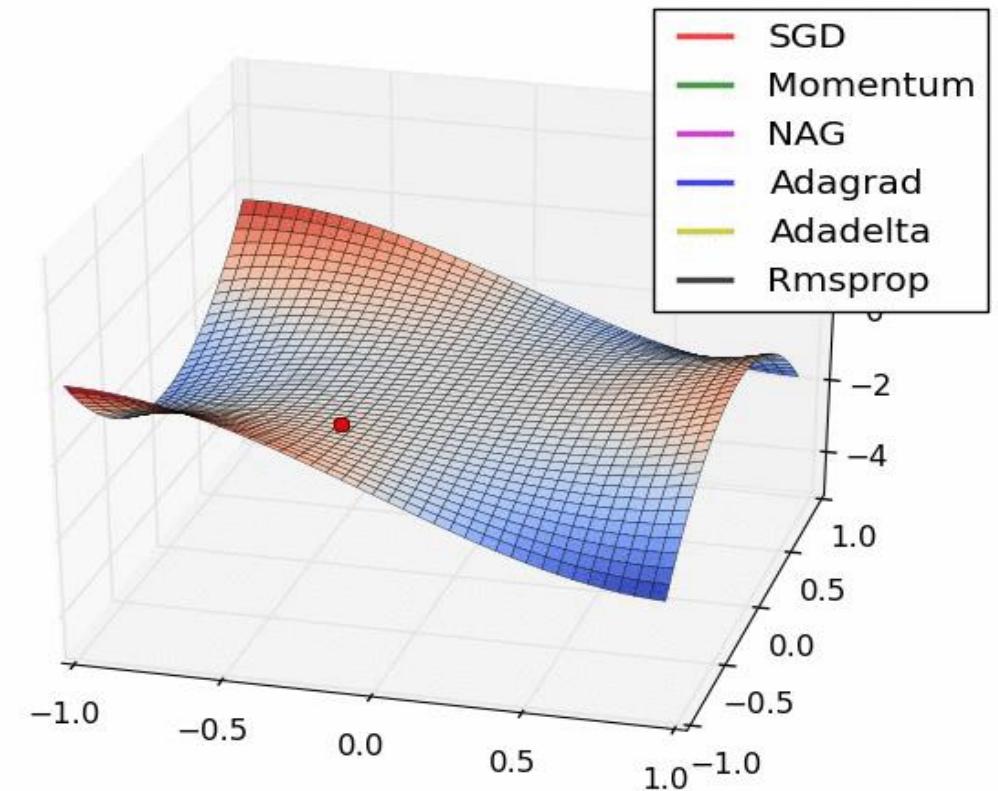
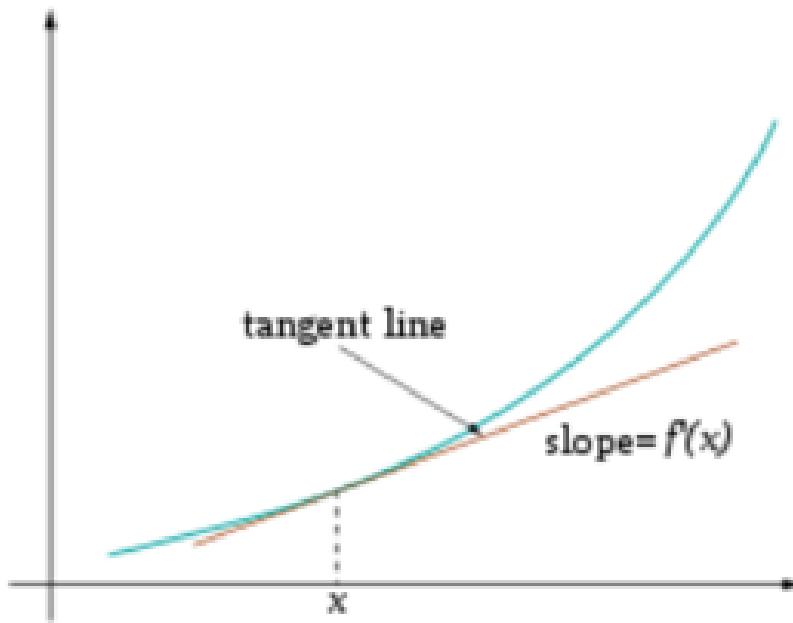
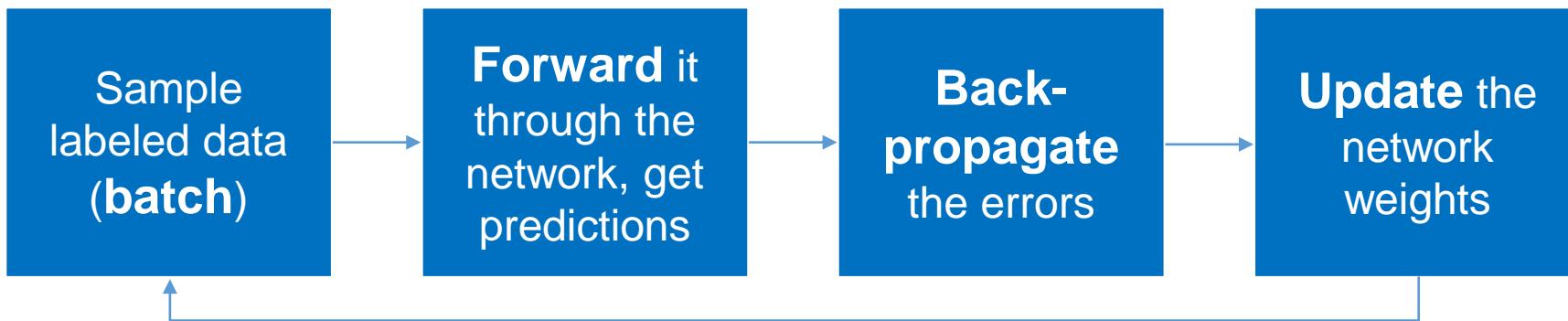


Local Minima

- Gradient descent never guarantee global minima



Training

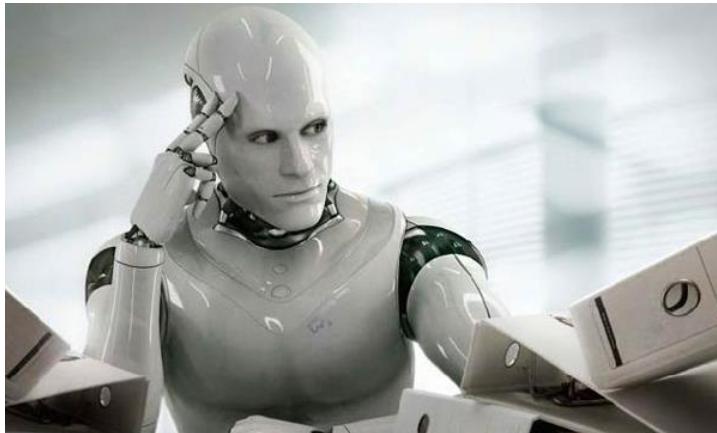


Gradient Descent

This is the “learning” of machines in deep learning

→ Even alpha go using this approach.

People image



Actually



I hope you are not too disappointed :p

Backpropagation

- Recursively apply **chain rule** though each node
- Backpropagation: an efficient way to compute $\partial L / \partial w$ in neural network



Caffe



theano



libdnn
台大周伯威
同學開發

Ref: <https://www.youtube.com/watch?v=ibJpTrp5mcE>

Three Steps for Deep Learning



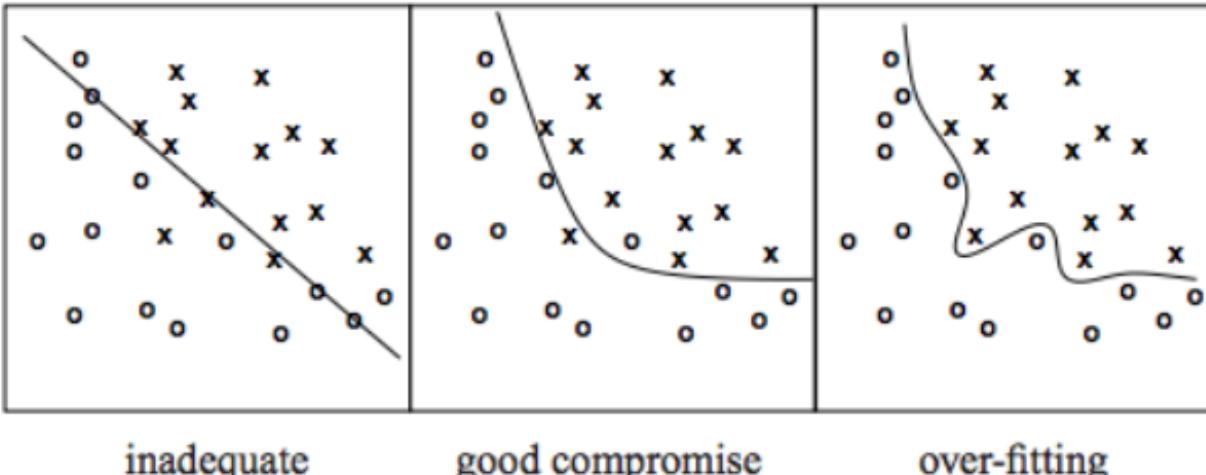
Deep Learning is so simple

Now If you want to find a function

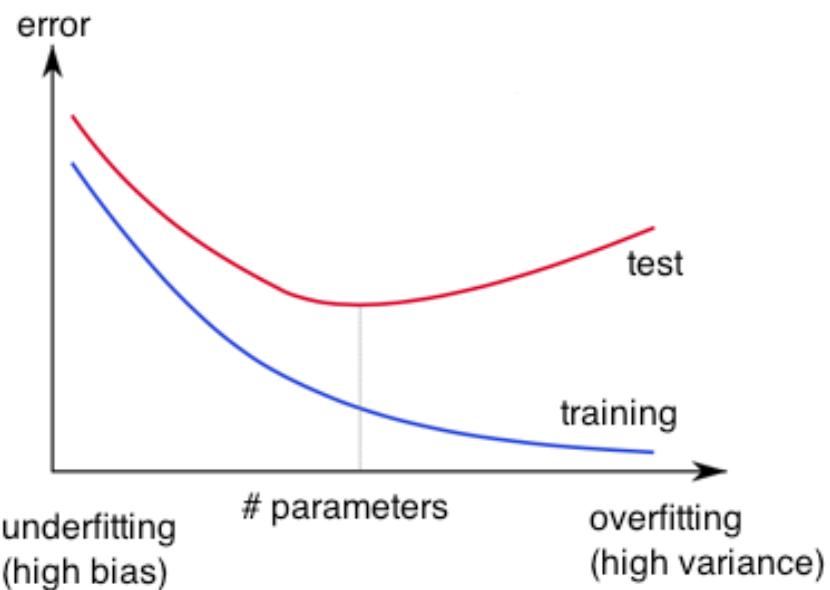
If you have lots of function input/output (?) as
training data

→ You can use deep learning

Overfitting



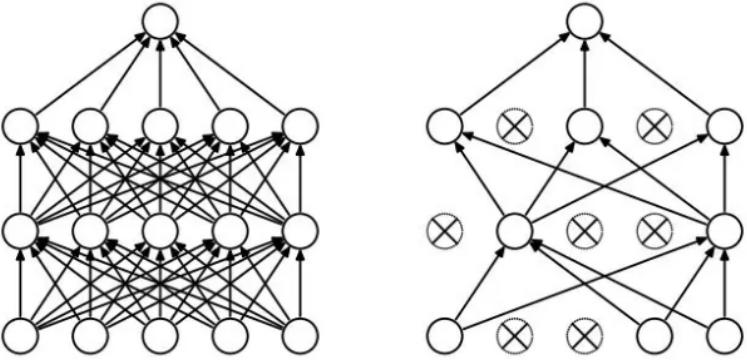
over-fitting



<http://wiki.bethanycrane.com/overfitting-of-data>

Learned hypothesis may **fit** the training data very well, even outliers (**noise**) but fail to **generalize** to new examples (**test data**)

Regularization



Dropout

- Randomly drop units (along with their connections) during training
- Each unit retained with fixed probability p , independent of other units
- **Hyper-parameter** p to be chosen (tuned)

Srivastava, Nitish, et al. [*"Dropout: a simple way to prevent neural networks from overfitting."*](#) Journal of machine learning research (2014)

L2 = weight decay

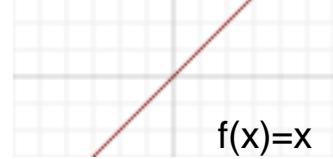
- Regularization term that penalizes big weights, added to the objective
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

$$J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$$

Early-stopping

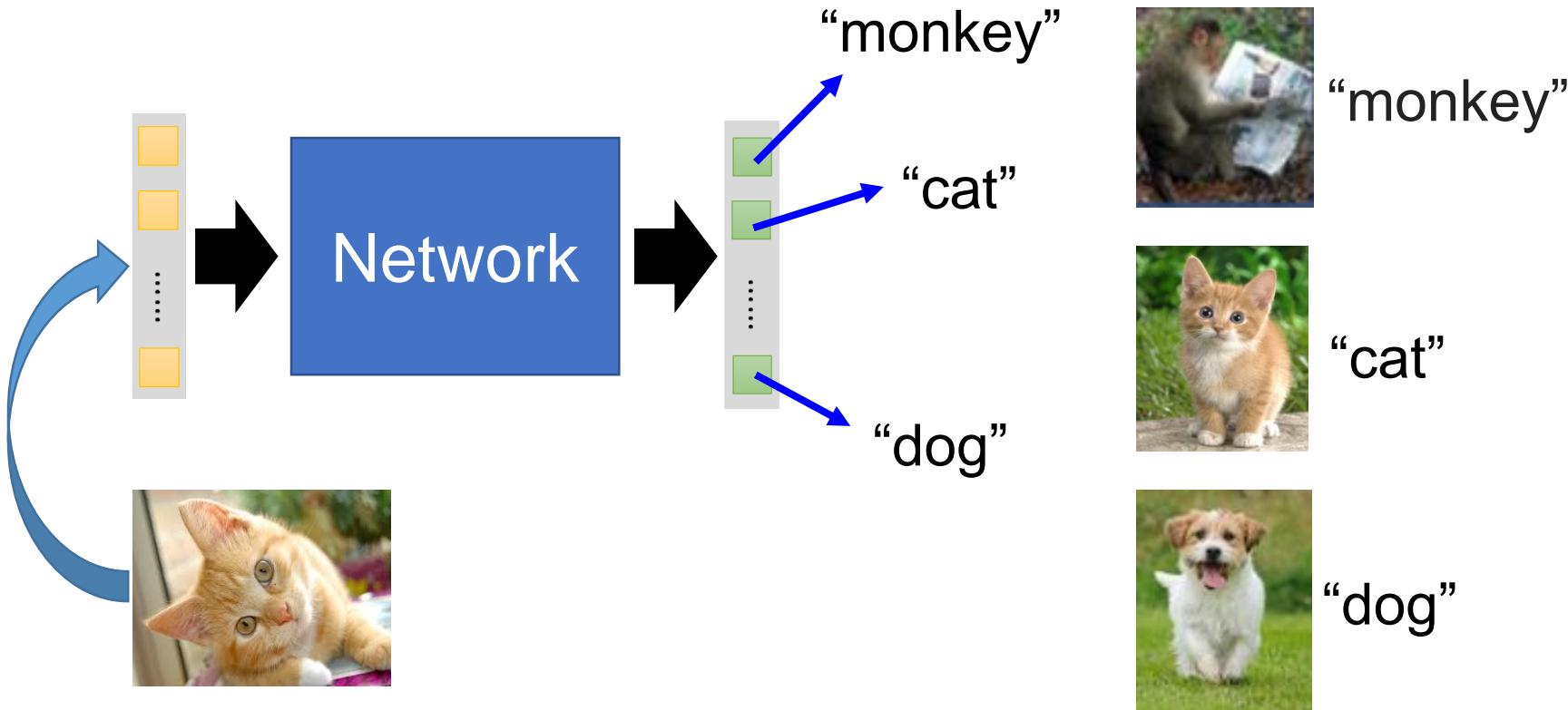
- Use **validation** error to decide when to stop training
- Stop when monitored quantity has not improved after n subsequent epochs
- n is called patience

Loss functions and output

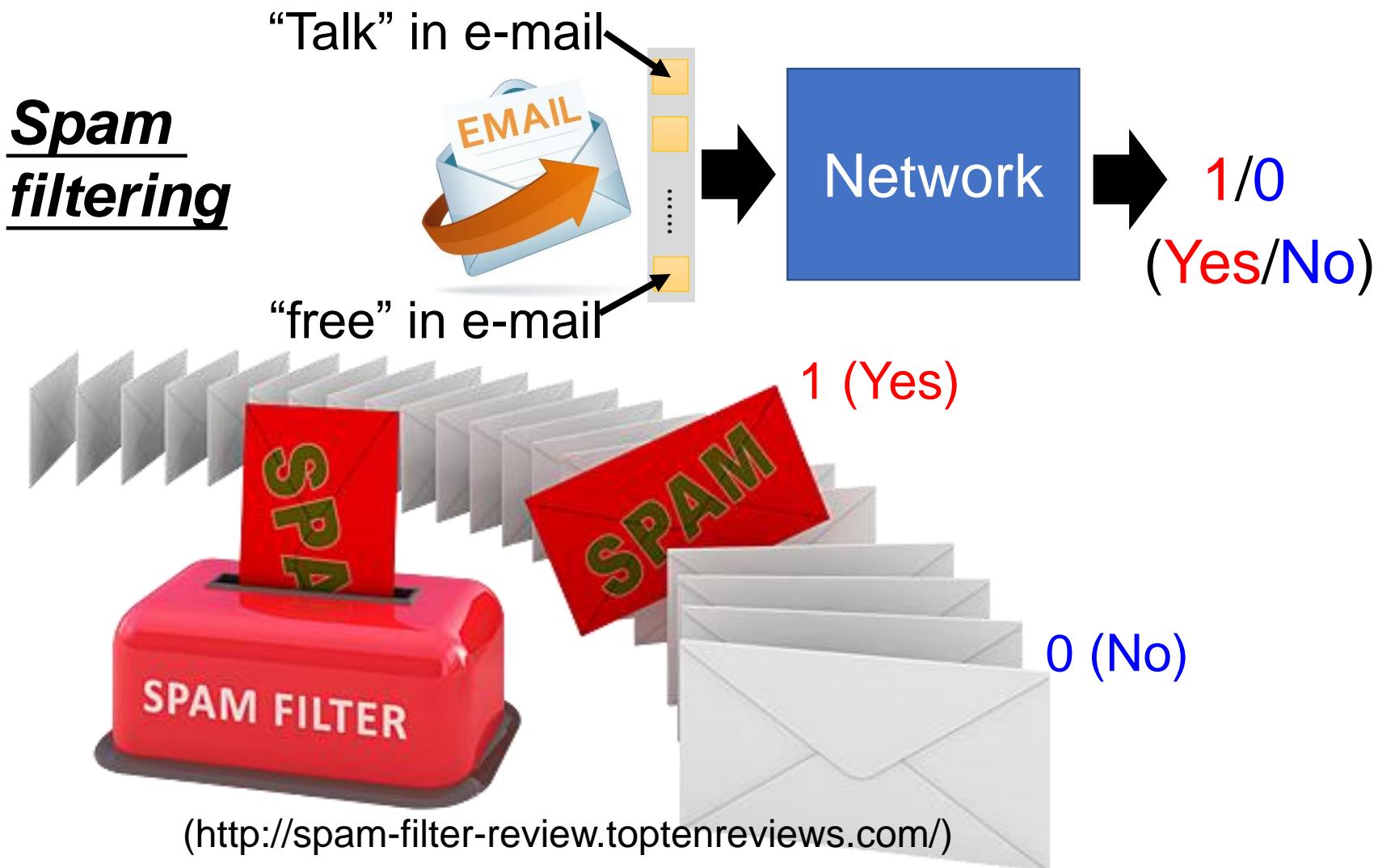
	Classification	Regression
Training examples	$R^n \times \{\text{class_1, ..., class_n}\}$ (one-hot encoding)	$R^n \times R^m$
Output Layer	Soft-max [map R^n to a probability distribution] $P(y = j \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}}$	Linear (Identity) or Sigmoid  A graph showing a red diagonal line passing through the origin, representing the identity function f(x) = x.
Cost (loss) function	Cross-entropy	Mean Squared Error $J(\theta) = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$
	$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \left[y_k^{(i)} \log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) \log (1 - \hat{y}_k^{(i)}) \right]$	Mean Absolute Error $J(\theta) = \frac{1}{n} \sum_{i=1}^n y^{(i)} - \hat{y}^{(i)} $
List of loss functions		

For example, you can do

- Image Recognition



For example, you can do

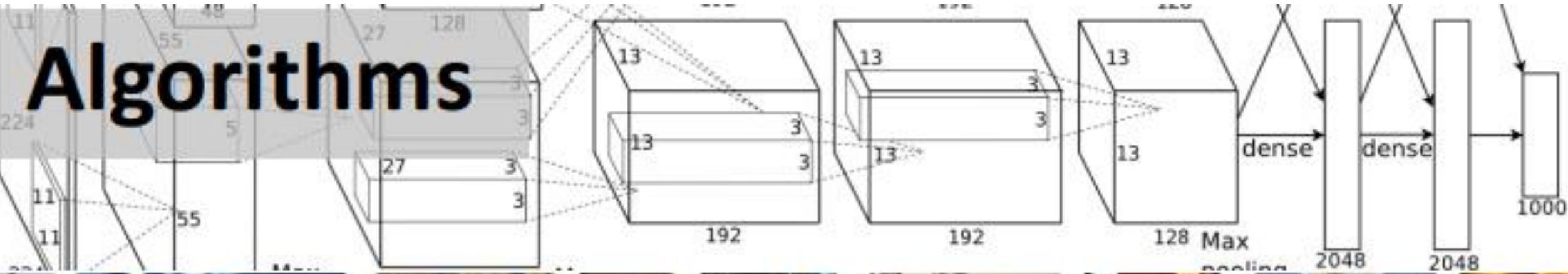


For example, you can do



Ingredients for Deep Learning

Algorithms



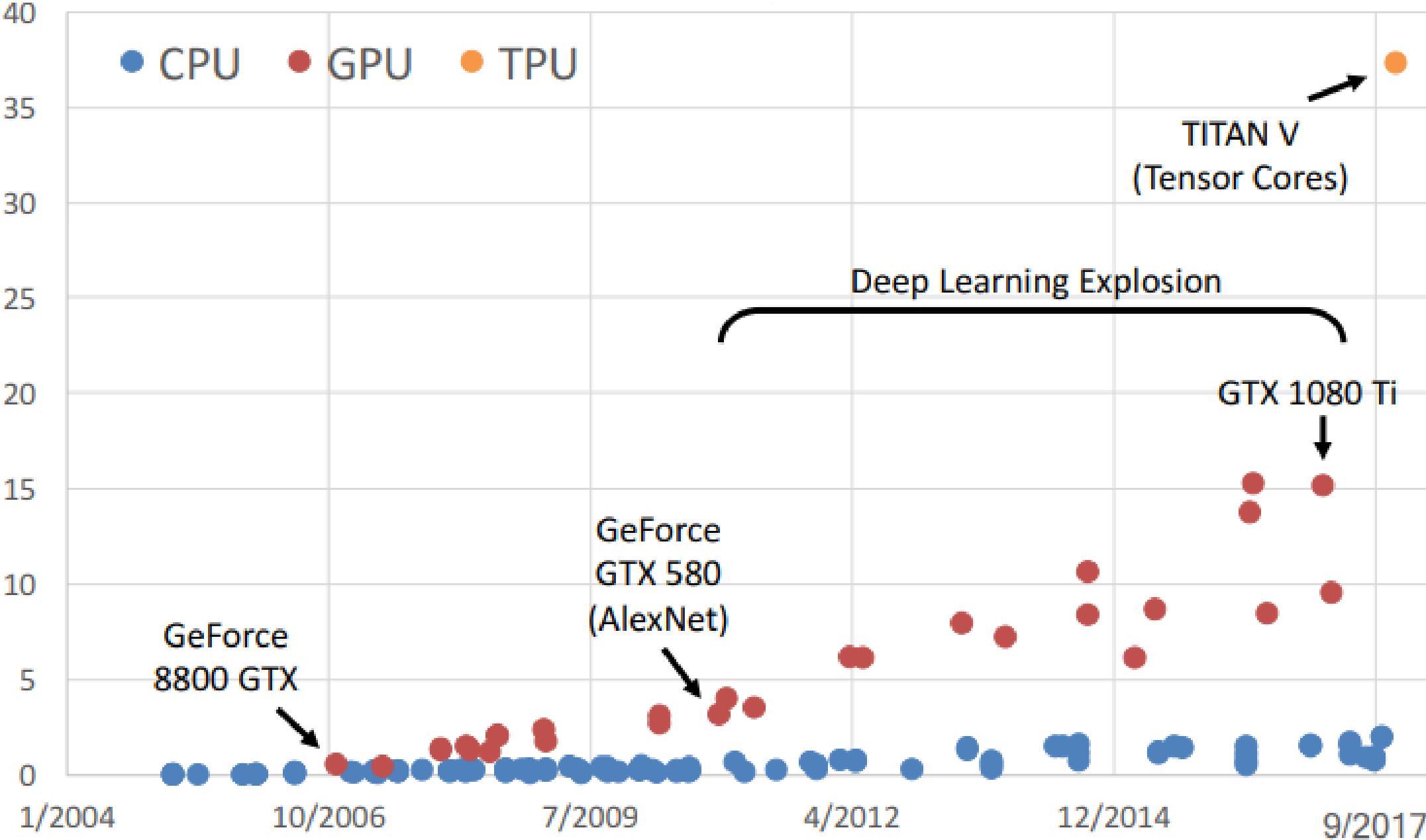
Data



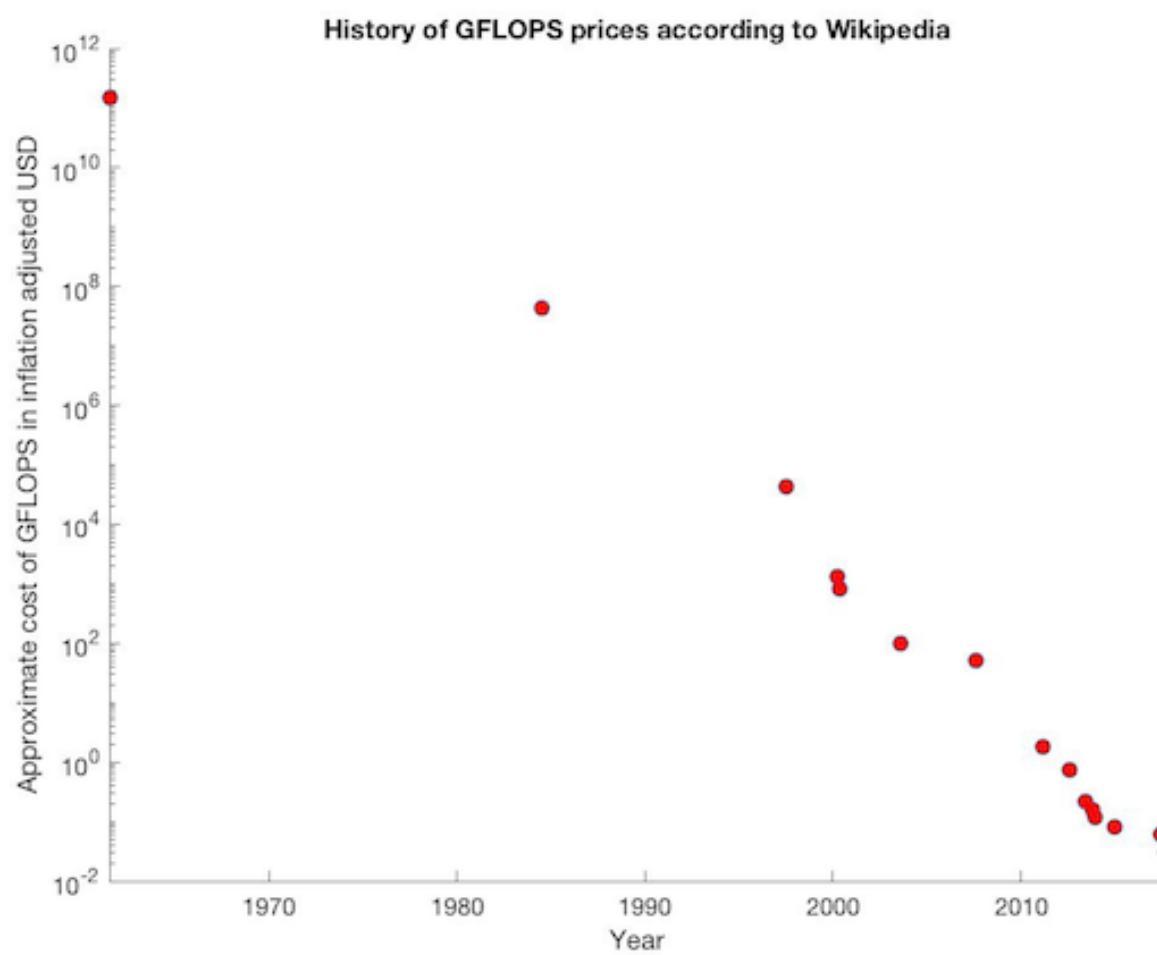
Computation



GigaFLOPs ($1*10^9$) per Dollar



History of GFLOPS costs



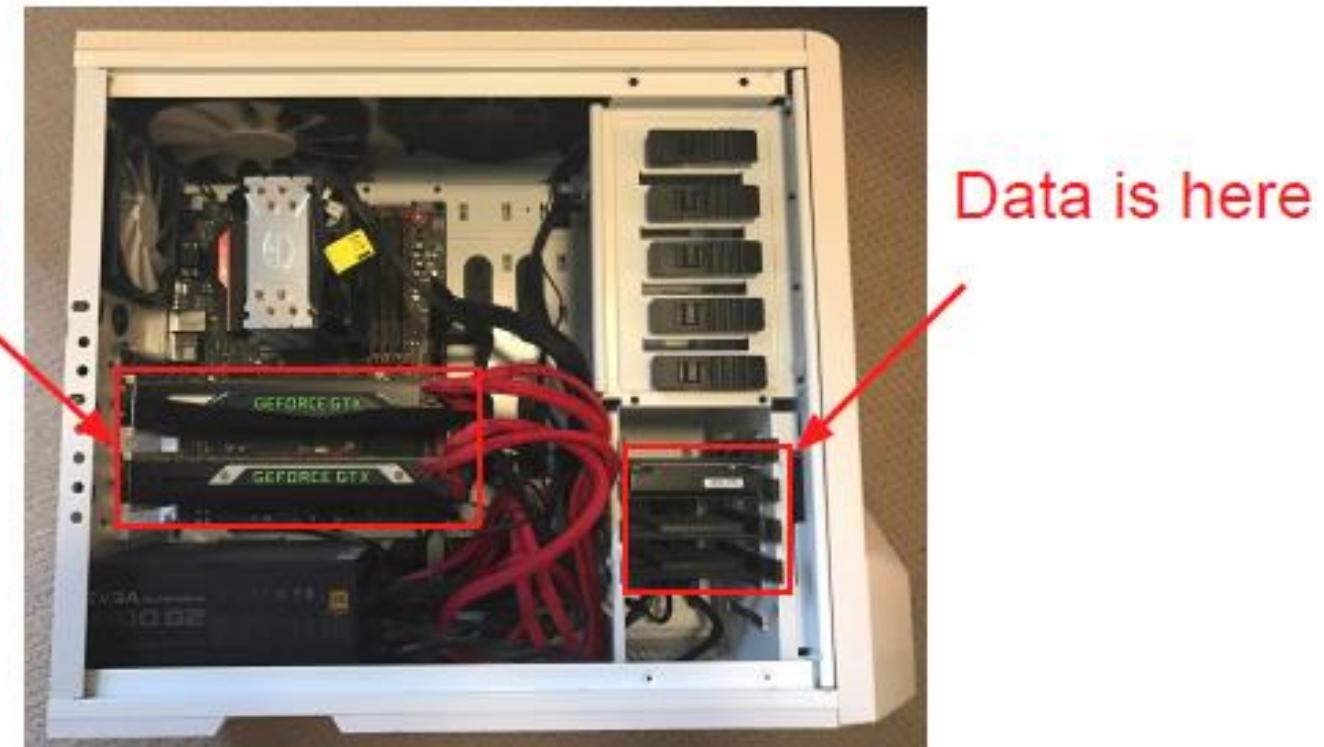
- 1961, US\$145.5 billion
- 2017, \$0.03
- The overall rate in the figure appears to be very roughly an **order of magnitude every five years**.

CPU / GPU Communication

- If you aren't careful, training can bottleneck on reading data & transferring to GPU!

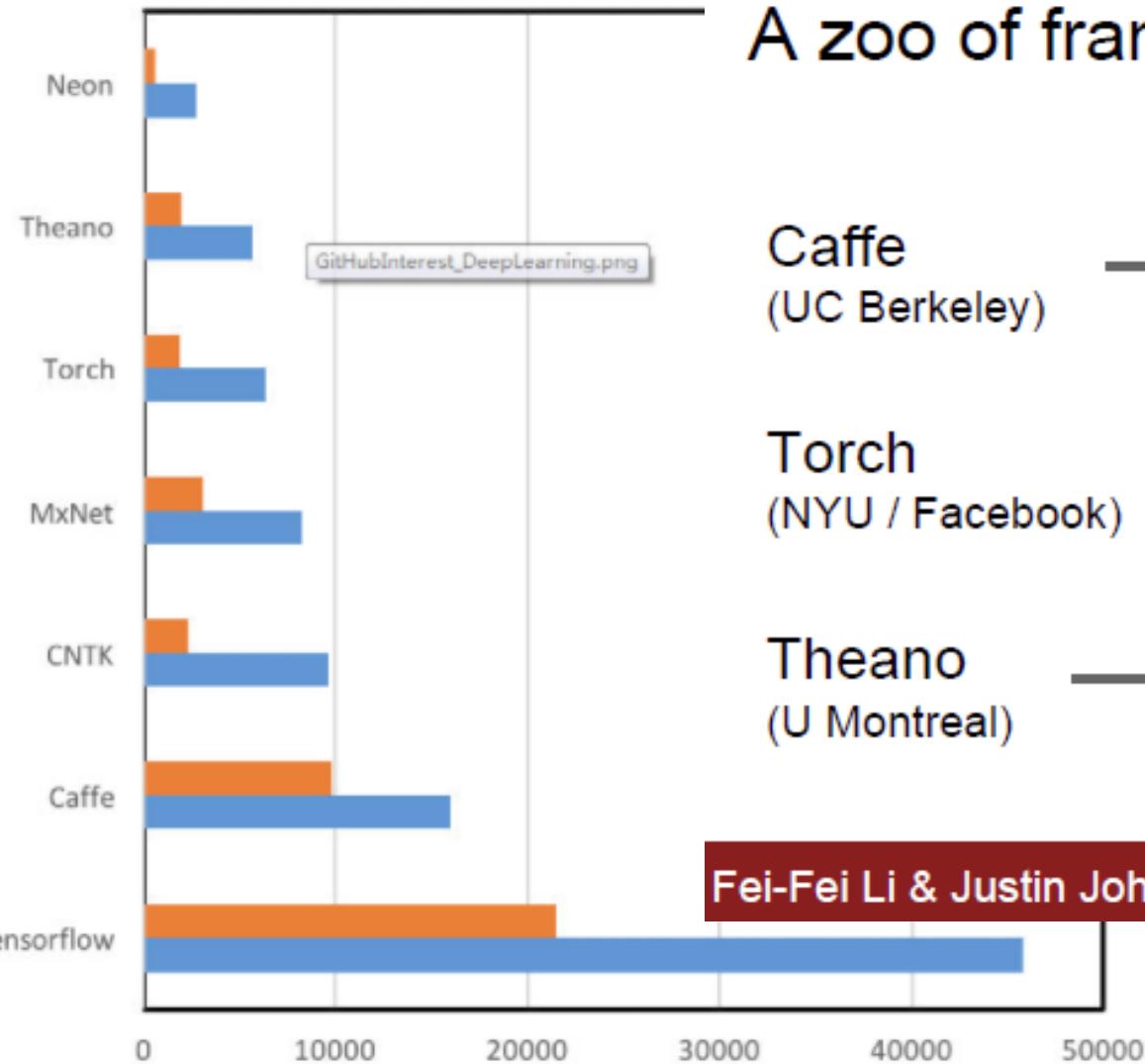
- **Solutions:**

- Read all data into RAM
- Use SSD instead of HDD
- Use multiple CPU threads to prefetch data



Seven important frameworks

Comparison of GitHub Interest
for Deep Learning Frameworks



A zoo of frameworks!

Caffe
(UC Berkeley)

Caffe2
(Facebook)

Torch
(NYU / Facebook)

PyTorch
(Facebook)

Theano
(U Montreal)

TensorFlow
(Google)

PaddlePaddle
(Baidu)

MXNet
(Amazon)
Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

Chainer

CNTK
(Microsoft)

Deeplearning4j

And others...

We'll focus on these

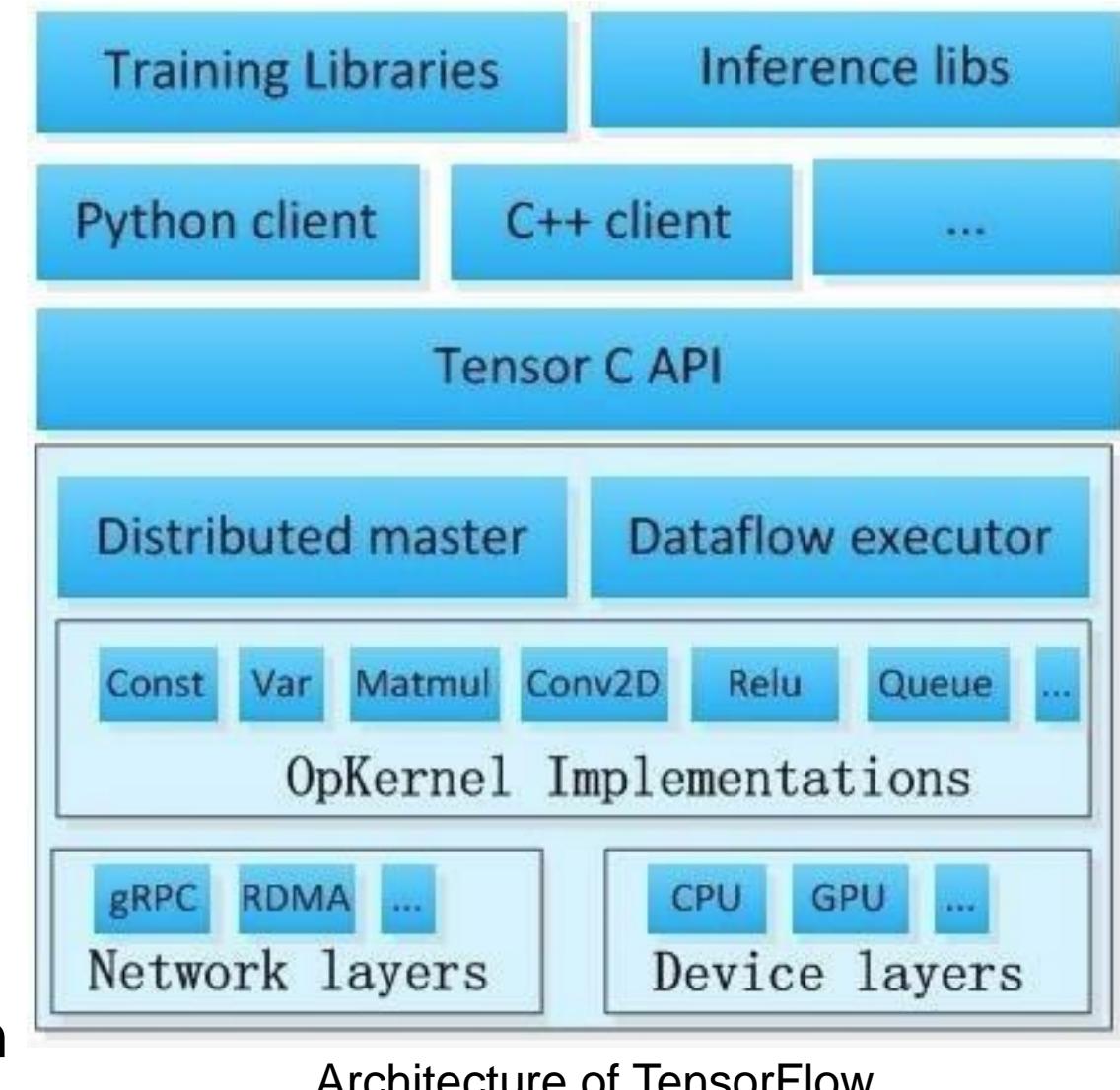
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 8 - 23 April 26, 2018

Prerequisites

- Proficiency in Python, high-level familiarity in C/C++
 - All class assignments will be in Python, but some of the deep learning libraries we may look at later in the class are written in C++.
 - If you have a lot of programming experience but in a different language (e.g. C/C++/Matlab/Javascript) you will probably be fine.

-- From cs231n



How to choose?

- 写代码意味着总是发现并修复 bug
- Tensorflow 静态图形库中寻找问题非常笨拙
- Pytorch 一类的动态图像，可以在堆栈跟踪中看到哪一行代码导致了错误，看看某个层会产生什么。通过采用断点并逐句检查代码，你发现上述 bug 的速度就可以提高 100 倍。

-- <https://www.jiqizhixin.com/articles/2017-06-13-4>

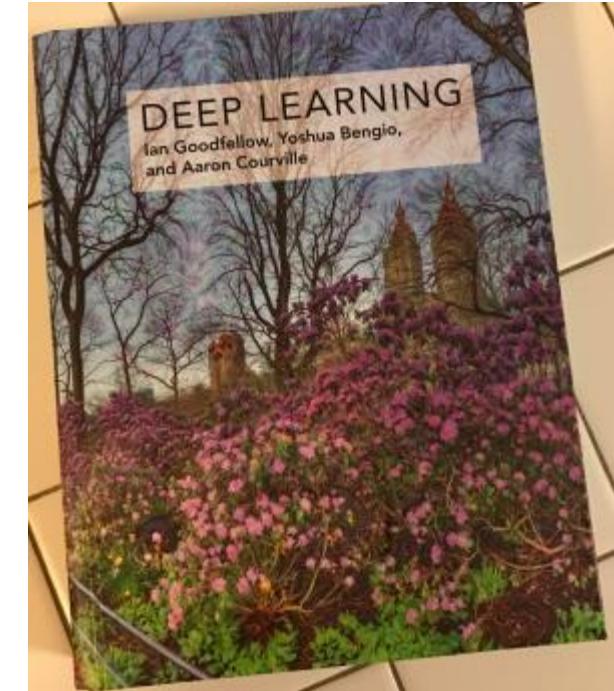
- 其实这么多年我看着各种库的起起落落，还有一种感慨是研究者不能始终抱着一个大腿，要与时俱进。但是时代的潮流在哪里也不是随时都能看出来的，也没法时刻保持自己在前沿，但好在掌握了一个库之后再换另一个库并不是很费劲。

	A	Pytorch	Keras
1	版本管理	***	*
2	调试	***	*
3	开发经验	**	***
4	上市时间	**	***
5	社区支持	***	**
6	商业支持	*	**
7	长期发展	**	**

• --CMU LTI博士研究生王贊

Resources

1. Deep learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, 2016, [free online](#)
2. Deep Learning @ NIPS'2015 Tutorial, Geoff Hinton, Yoshua Bengio & Yann LeCun
3. Understanding Deep Learning in One Day-2017-HYLee
4. CS231n: Convolutional Neural Networks for Visual Recognition, Spring 2018, Feifei Li
5. deeplearning.ai by andrew ng @ coursa or bilibili (video course)
6. Basic tutorials provided by Pytorch, etc



Thanks