Deep learning
-- Introduction

Junjie Cao @ DLUT

Spring 2019


@teenybiscuit
@teenybiscuit
@teenybiscuit

Face blindness: 'I can't recognise my loved-ones'

Prosopagnosia

By Max Evans
BBC News

6 March 2019

A stranger once waved at Boo James on a bus. She did not think any more of it - until it later emerged it was her mother.

She has a relatively rare condition called face blindness, which means she cannot recognise the faces of her family, friends, or even herself.

[...]

But how do people with prosopagnosia perceive faces? Those with the condition say it can be difficult to describe.

"I can see component parts of a face," Boo said. "I can see there's a nose, I can see there are eyes and a mouth and ears.

"But it's very difficult for my brain to hold them all together as the image of a face."

So far…

PASCAL VOC	= ~75%	20-class
ImageNet	= ~75%	1000-class, top 5

Human brains used intuition and understanding of how we think vision works to develop computer vision systems,

and it’s pretty good.

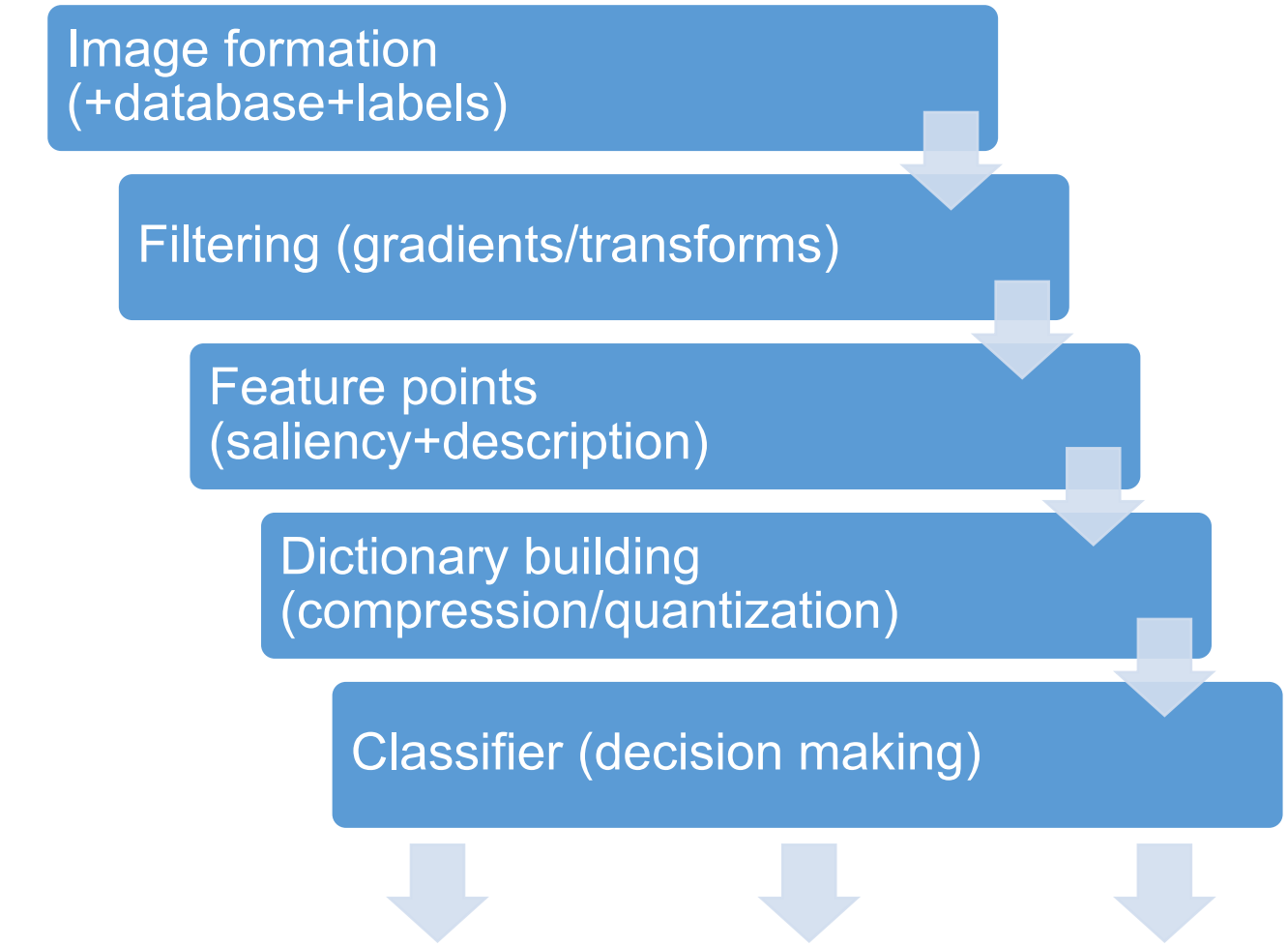


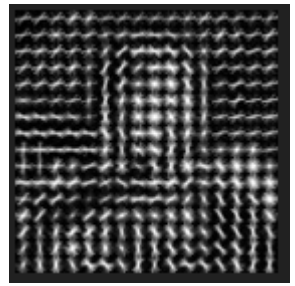
Image formation
(+database+labels)
Filtering (gradients/transforms)
Feature points
(saliency+description)
Dictionary building
(compression/quantization)
Classifier (decision making)
Captured+manual.
Hand designed.
Hand designed.
Hand designed.
Learned.
Recognition:
Classificatio
n
Object
Detection
Segmentatio
n

Well, what do we have?

Best performing visions systems have commonality:

Hand designed features

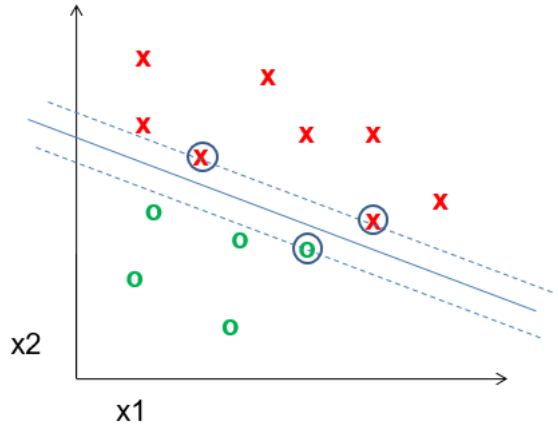
- Gradients + non-linear operations (exponentiation, clamping, binning)
- Features in combination (parts-based models)
- Multi-scale representations



Machine learning from databases

Linear classifiers (SVM)

- Some non-linear kernel tricks



But it’s still not that good…

PASCAL VOC = ~75% 20-class

ImageNet = ~75% 1000-class, top 5

ImageNet (human) = ~95%

Problems:

- Lossy features
- Lossy quantization
- ‘Imperfect’ classifier

But it's still not that good…

- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

How to solve?

- *Features: More principled modeling?*
  We *know* why the world looks (it's physics!);
  Let's build better physically-meaningful models.
- *Quantization: More data and more compute?*
  It's just an interpolation problem; let's represent the space with fewer data approximations.
- *Classifier: …*

The limits of learning?

Previous claim:

It is more important to have
more or better labeled data than to use
a different supervised learning technique.

"The **Unreasonable Effectiveness of Data**" - Norvig

No free lunch theorem

Hume (c.1739):

“Even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience.”

-> **Learning beyond our experience is impossible.**

No free lunch theorem for ML

Wolpert (1996):

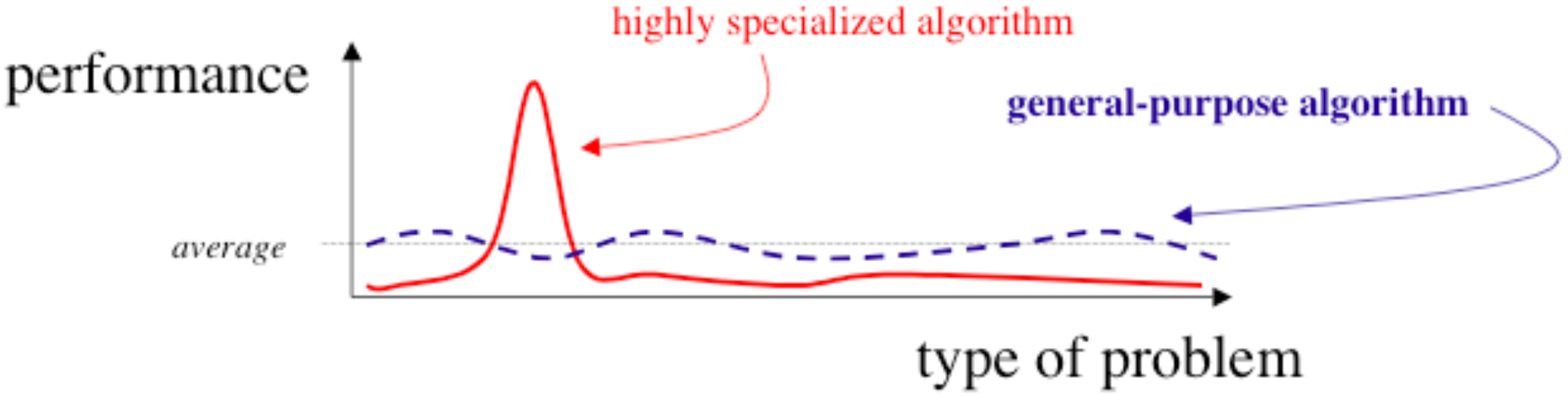
‘No free lunch’ for supervised learning:

“In a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no *a priori* distinctions between learning algorithms.”

-> **Averaged over all possible datasets, no learning algorithm is better than any other.**

OK, well, let's give up. Class over.

No, no, no!



We can build a classifier which better matches the **characteristics of the problem**!

But…didn’t we just do that?

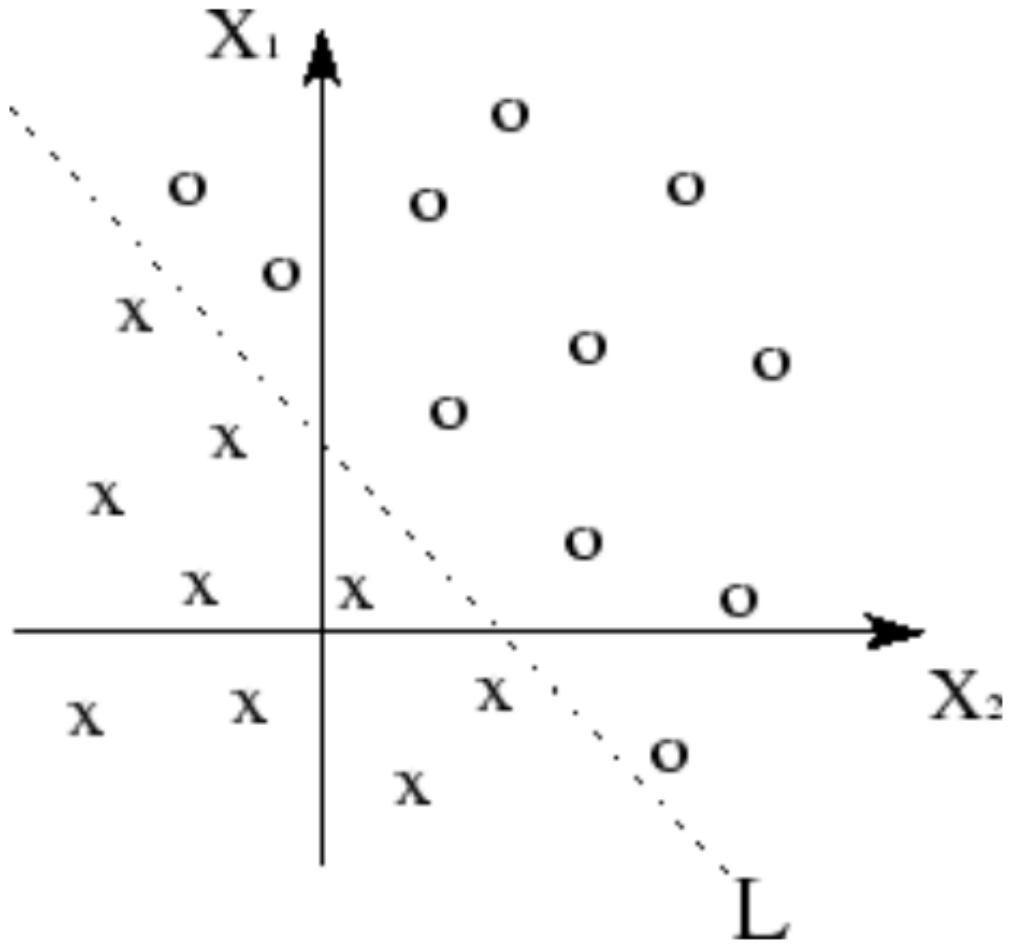
- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

We used intuition and understanding of how we think vision works, but it still has limitations.

Why?

Linear spaces - separability

- + kernel trick to transform space.



Linearly separable data
+ linear classifer = good.

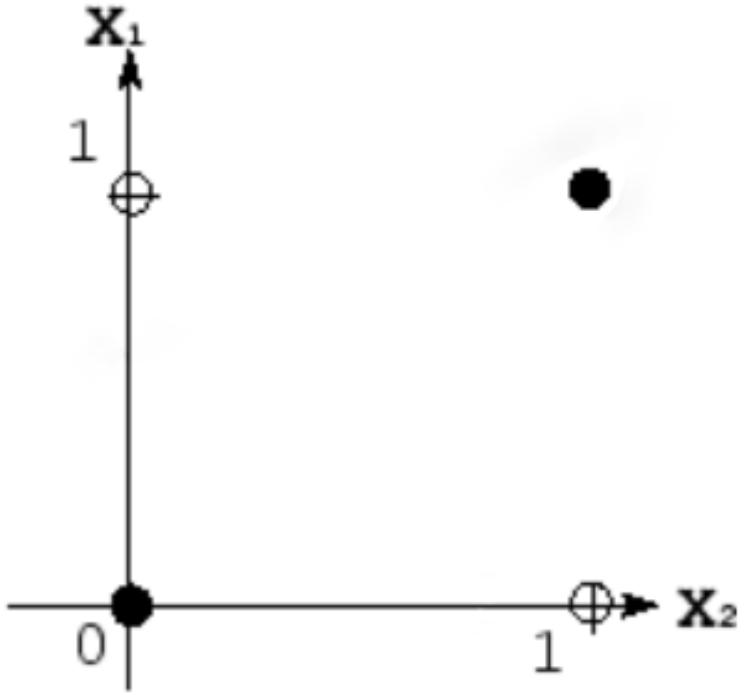
Non-linear spaces - separability

Take XOR – exclusive OR

E.G., human face has two eyes XOR sunglasses

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$

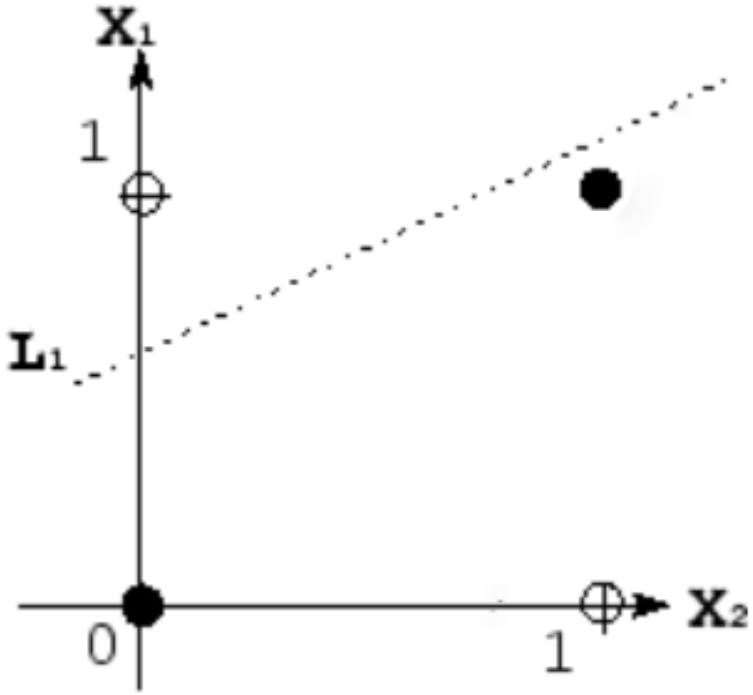


Non-linear spaces - separability

Linear functions are insufficient on their own.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$

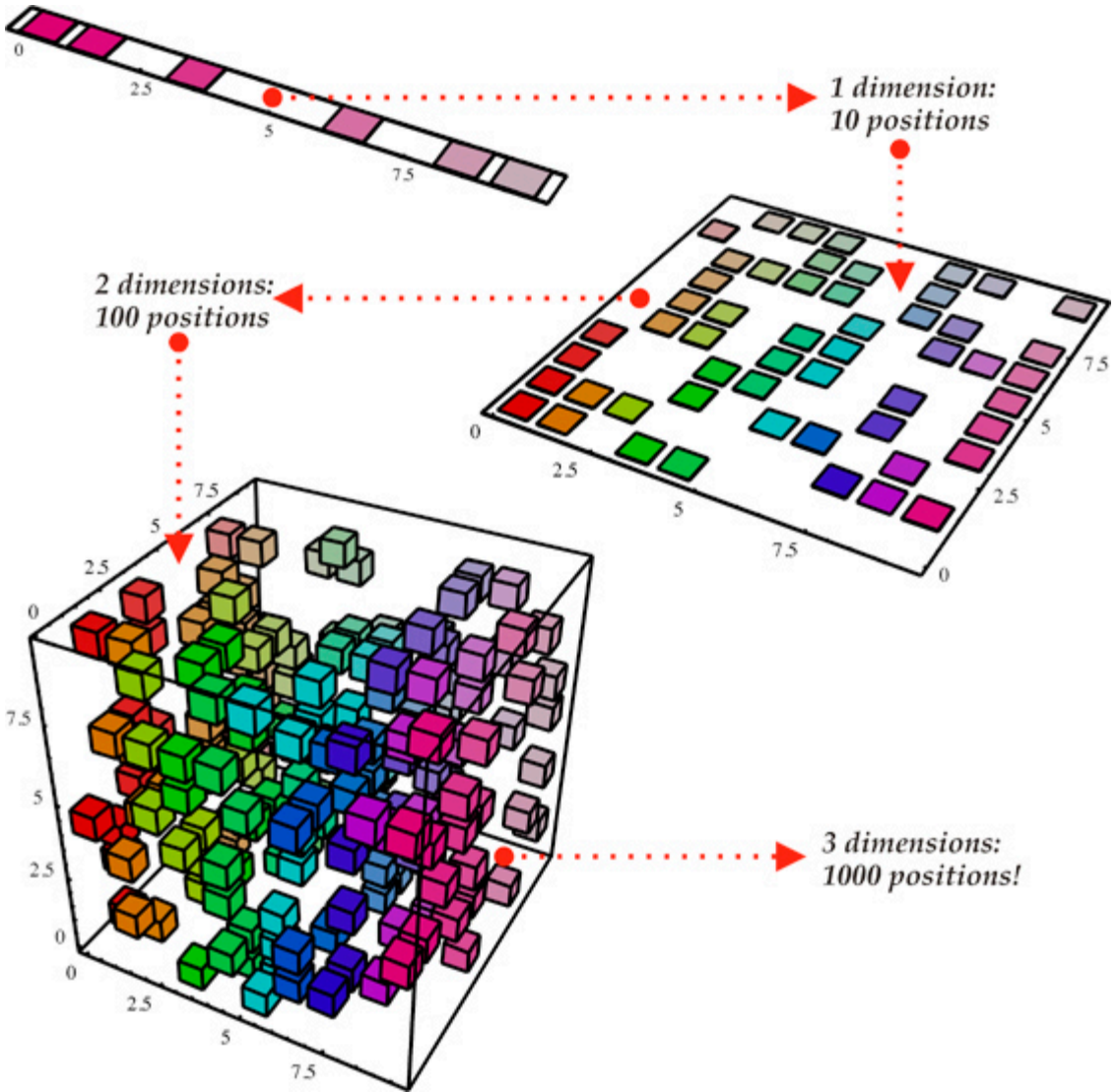


Kawaguchi

Curse of Dimensionality

Every feature that we add requires us to learn the useful regions in a much larger volume.

d binary variables = $O(2^d)$ combinations

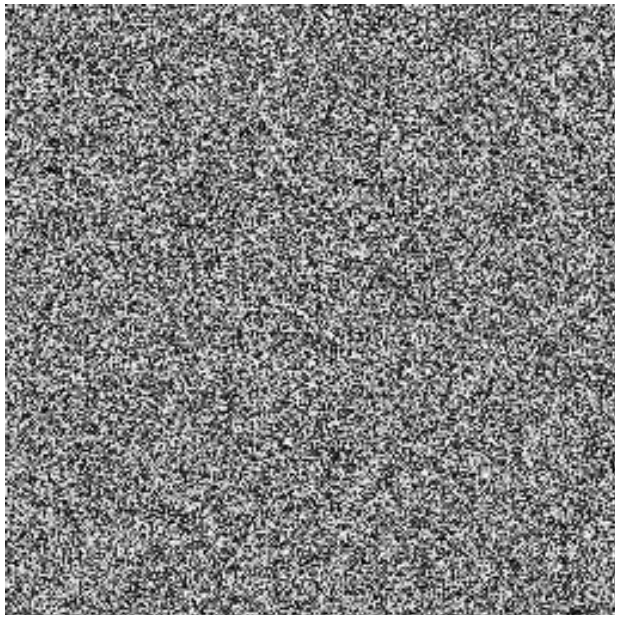


Curse of Dimensionality

Not all regions of this high-dimensional space are meaningful.

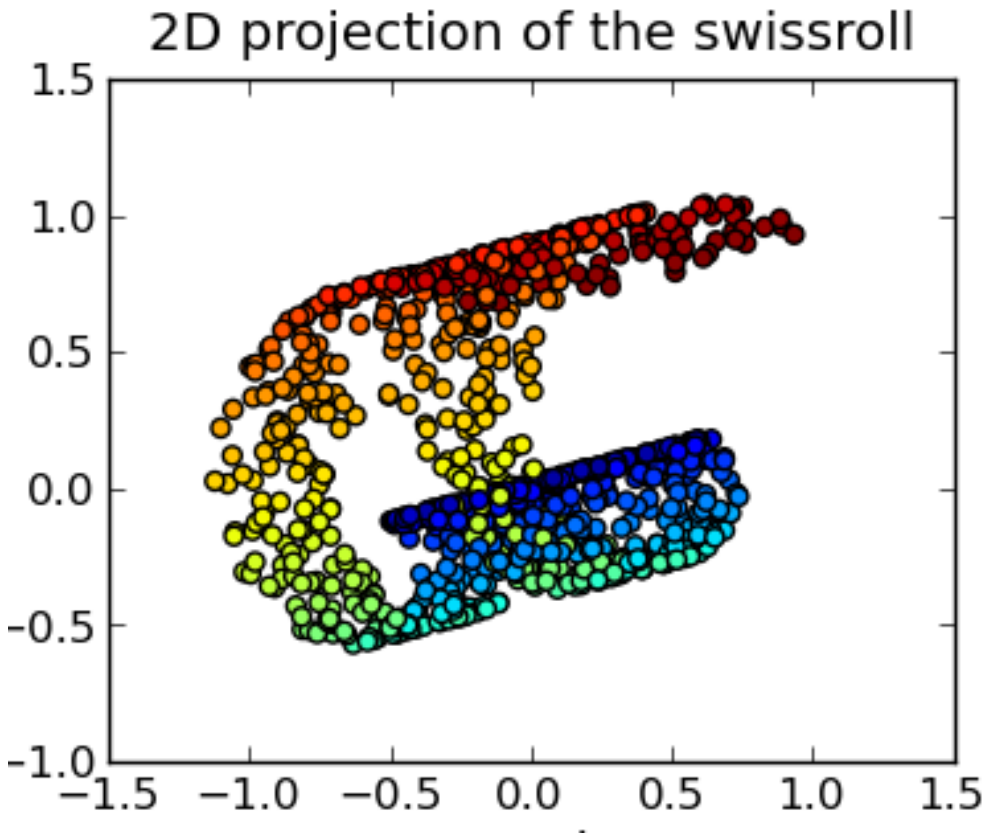
```
>> I = rand(256,256);
>> imshow(I);
```

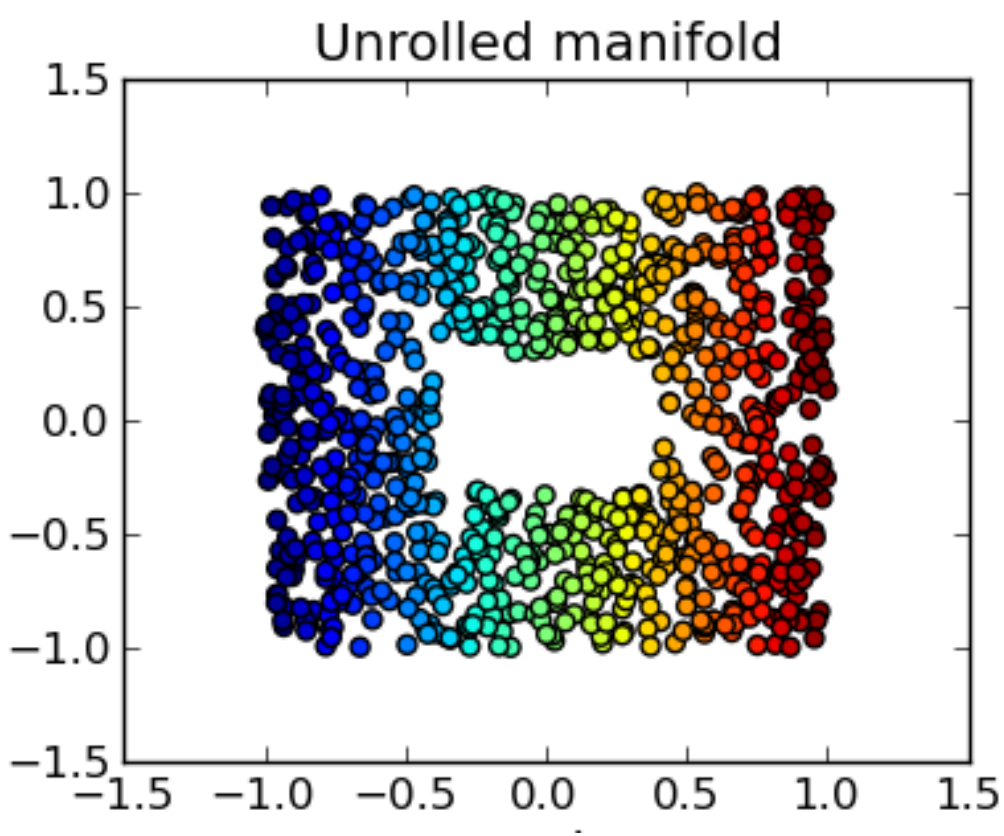
@ 8bit = 256 values ^ 65,536



Related: Manifold Learning

Learning locally low-dimensional Euclidean spaces connected and embedded within a high-dim. space.

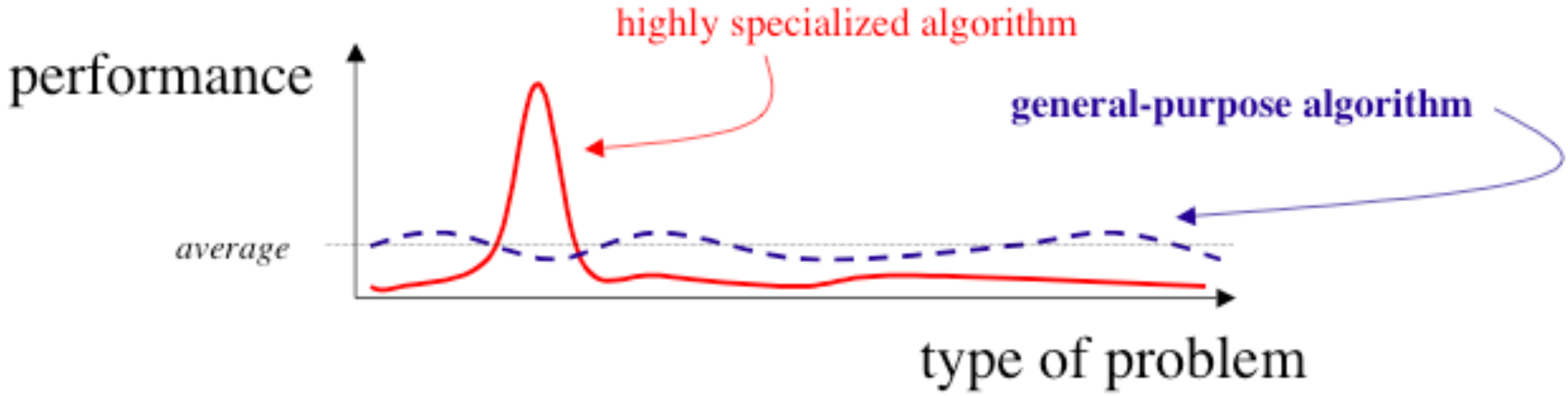




Wikipedia

More specialization?

- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%



Is there a way to make our system better suited to the problem?

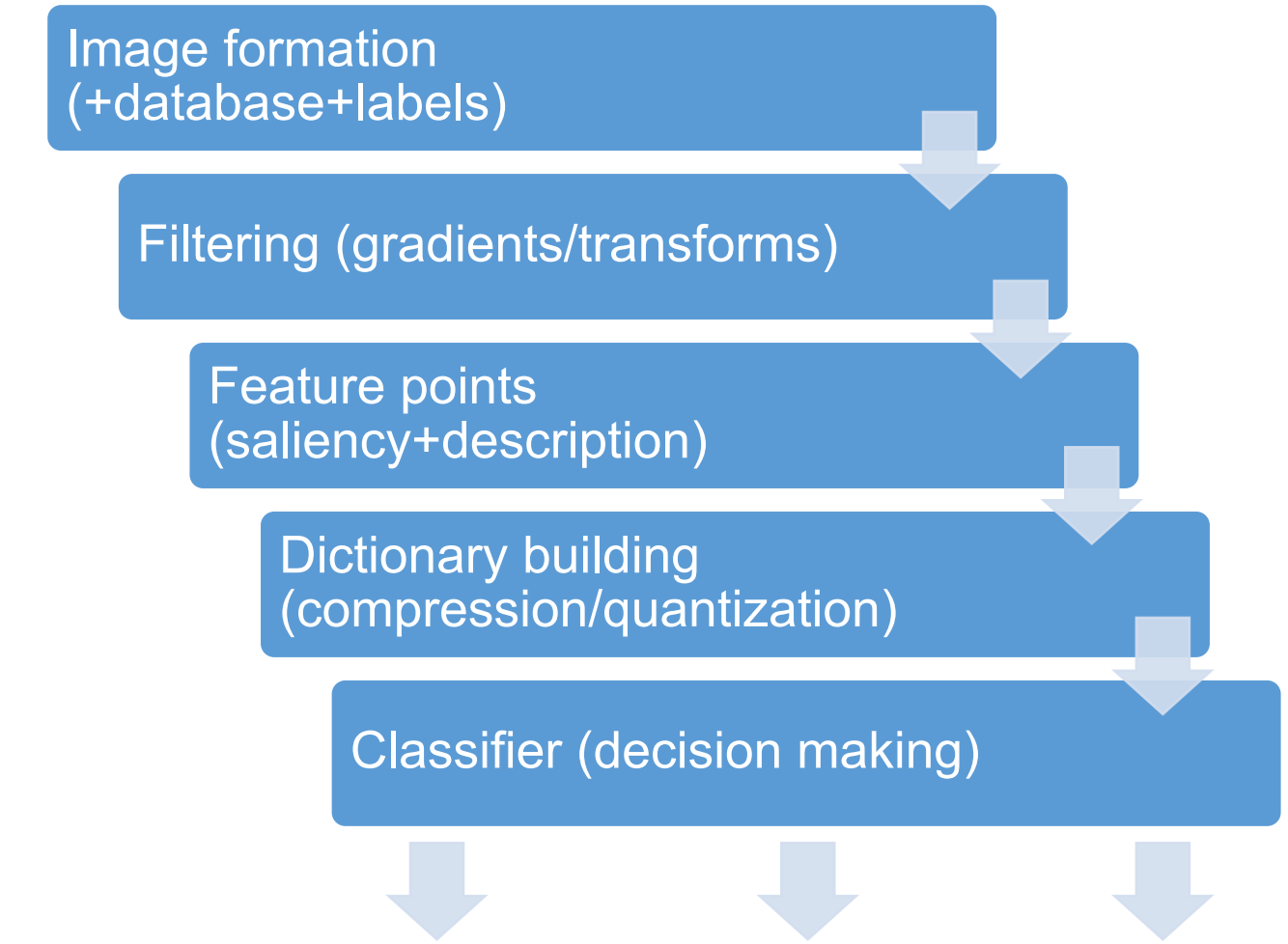
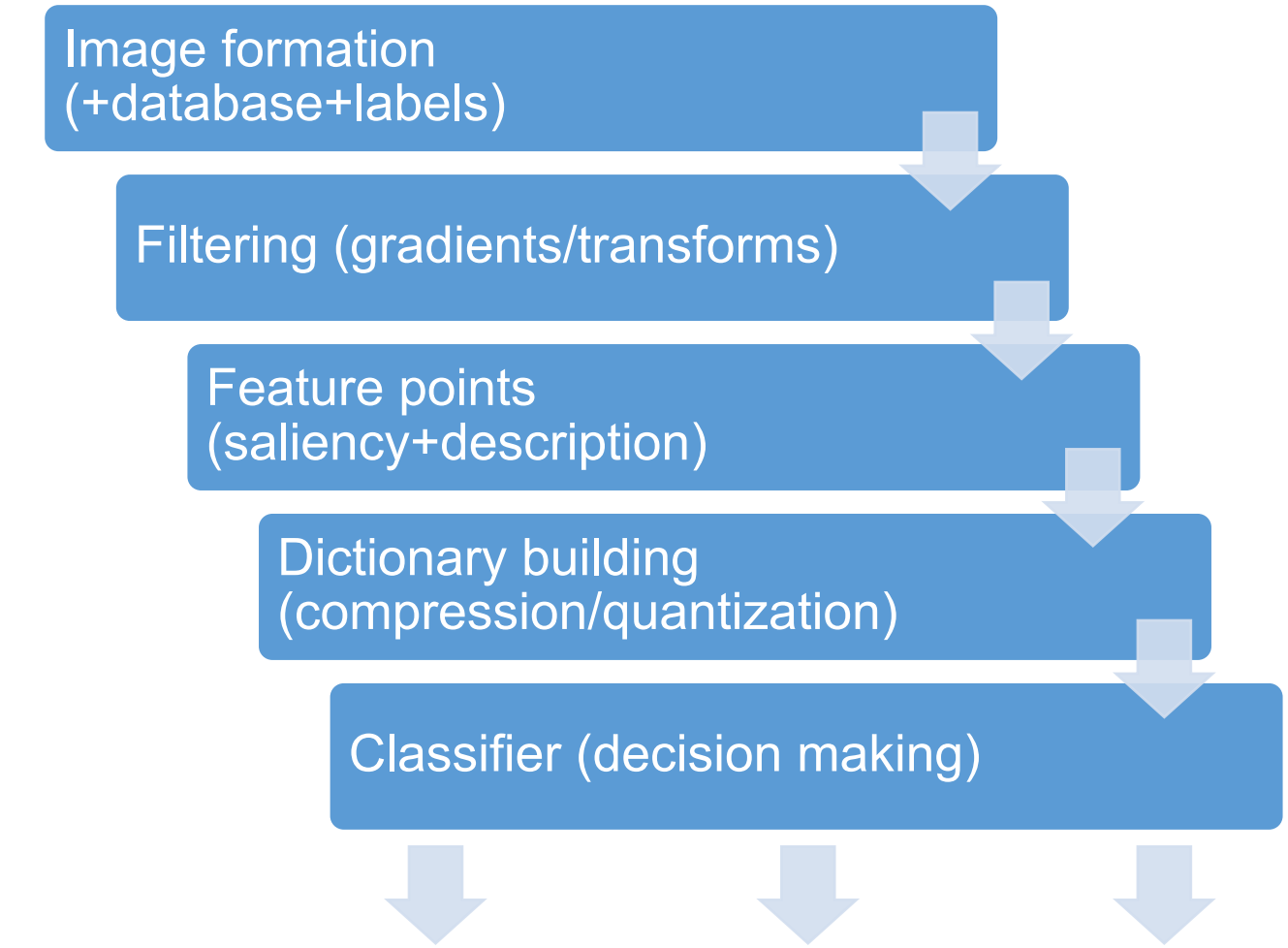


Image formation
(+database+labels)
Filtering (gradients/transforms)
Feature points
(saliency+description)
Dictionary building
(compression/quantization)
Classifier (decision making)
Captured+manual.
Hand designed.
Hand designed.
Hand designed.
Learned.
Recognition:
Classification
Object Detection
Segmentation

Wouldn’t it be great if we could…



Goals

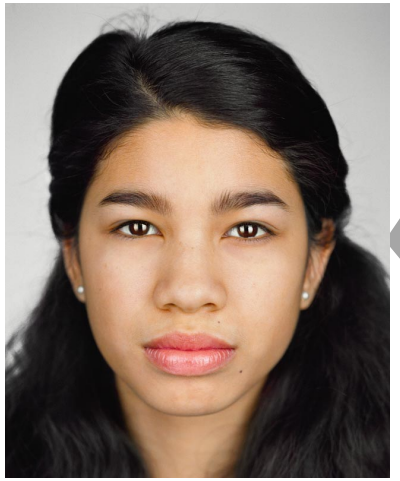
Build a classifier which is more powerful
at representing **complex functions**
and more suited to the learning problem.

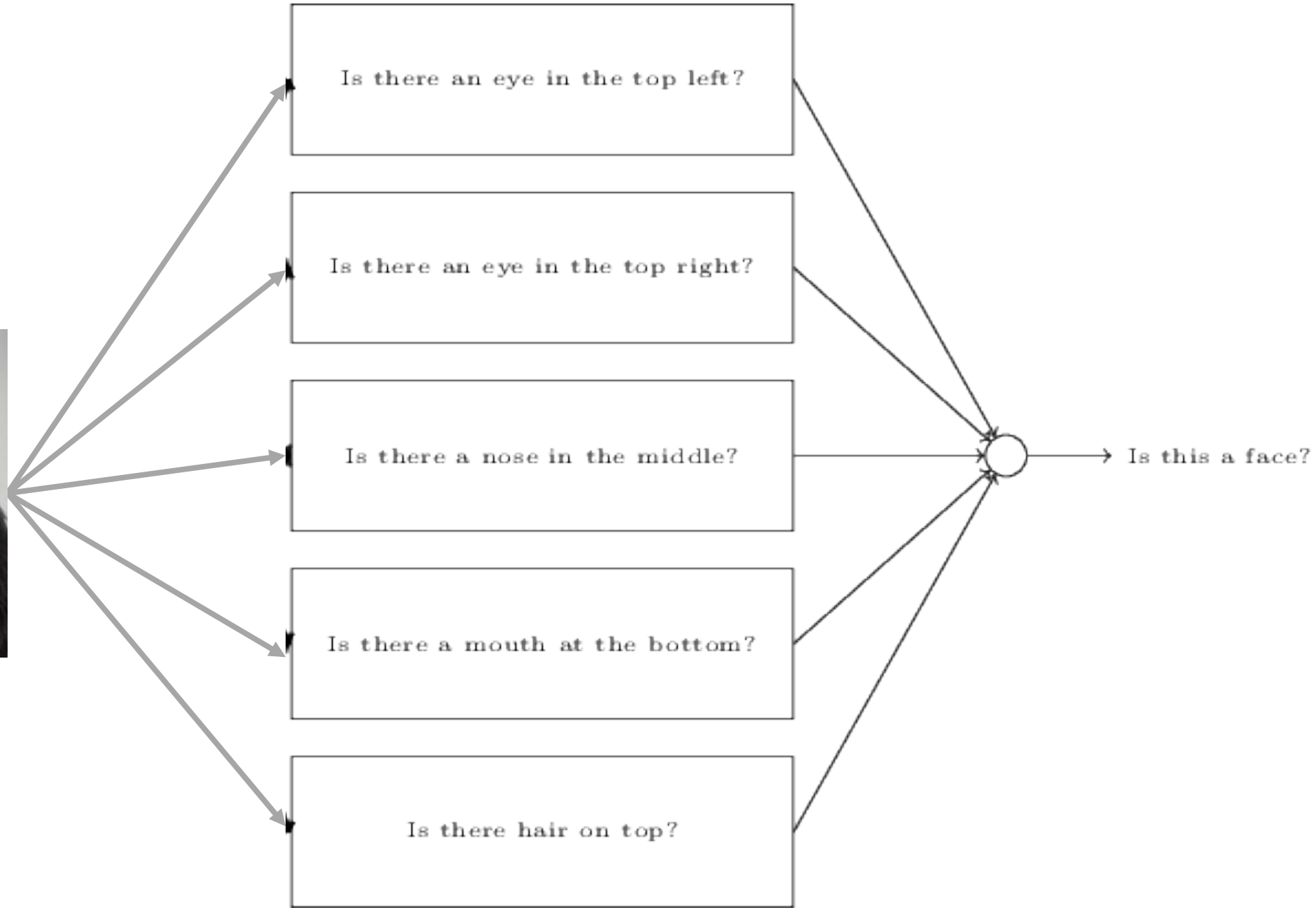
What does this mean?

1. Assume that the *underlying data generating function* relies on a **composition of factors in a hierarchy**.

Dependencies between regions in feature space
= factor composition

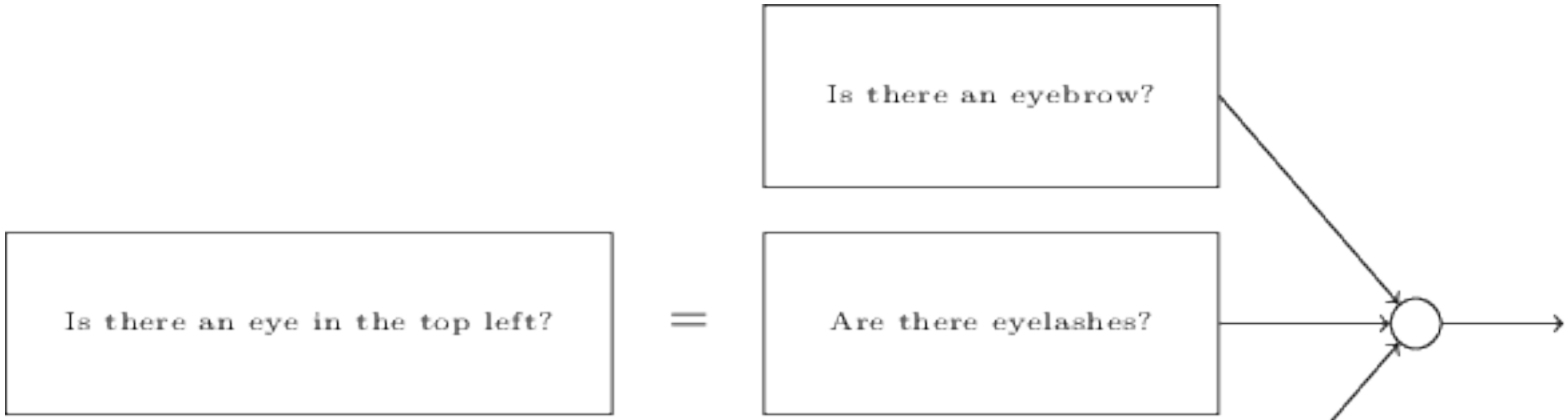
Example





Nielsen, National Geographic

Example



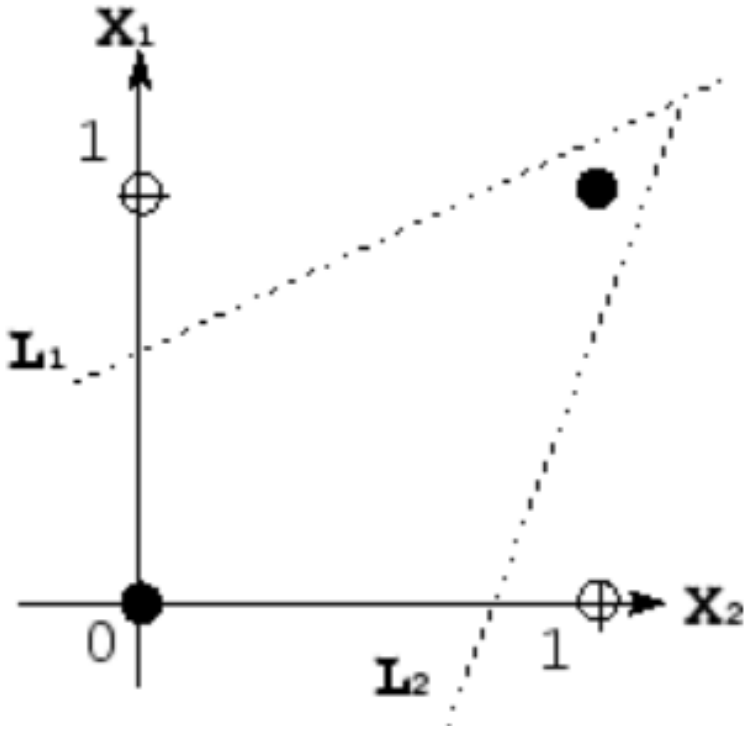
Nielsen, National Geographic

Non-linear spaces - separability

Composition of linear functions can represent more complex functions.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$



Kawaguchi

Goals

Build a classifier which is more powerful
at representing complex functions
and more suited to the learning problem.

What does this mean?

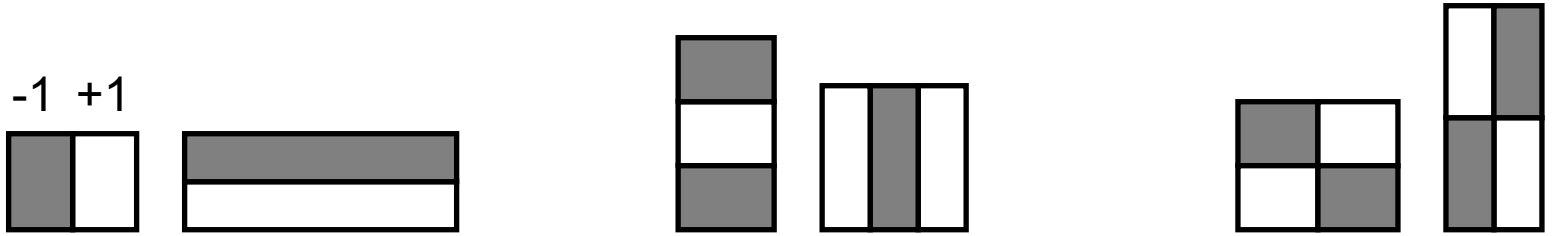
1. Assume that the *underlying data generating function* relies on a composition of factors in a hierarchy.

2. Learn a feature representation **specific to the dataset**.

10k/100k + data points + factor composition
= sophisticated representation.

Reminder: Viola Jones Face Dete

Combine *thousands* of ‘weak classifiers’

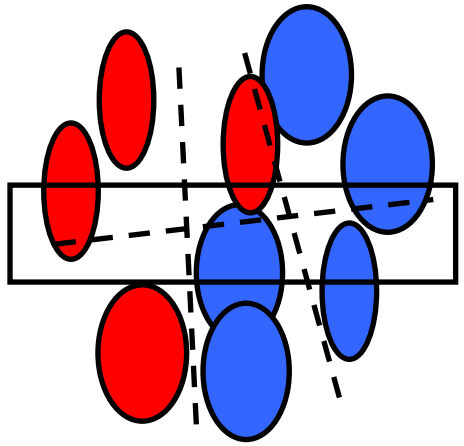


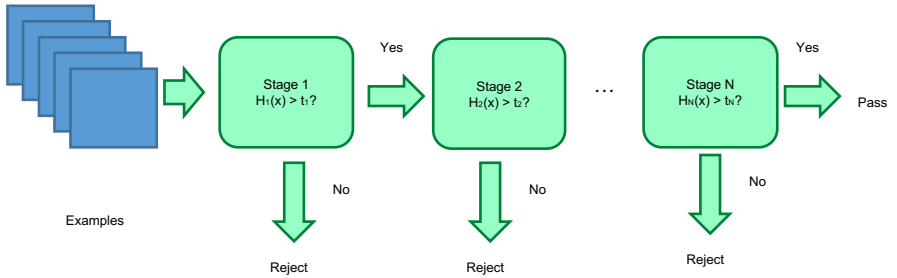
Two-rectangle features

Three-rectangle features

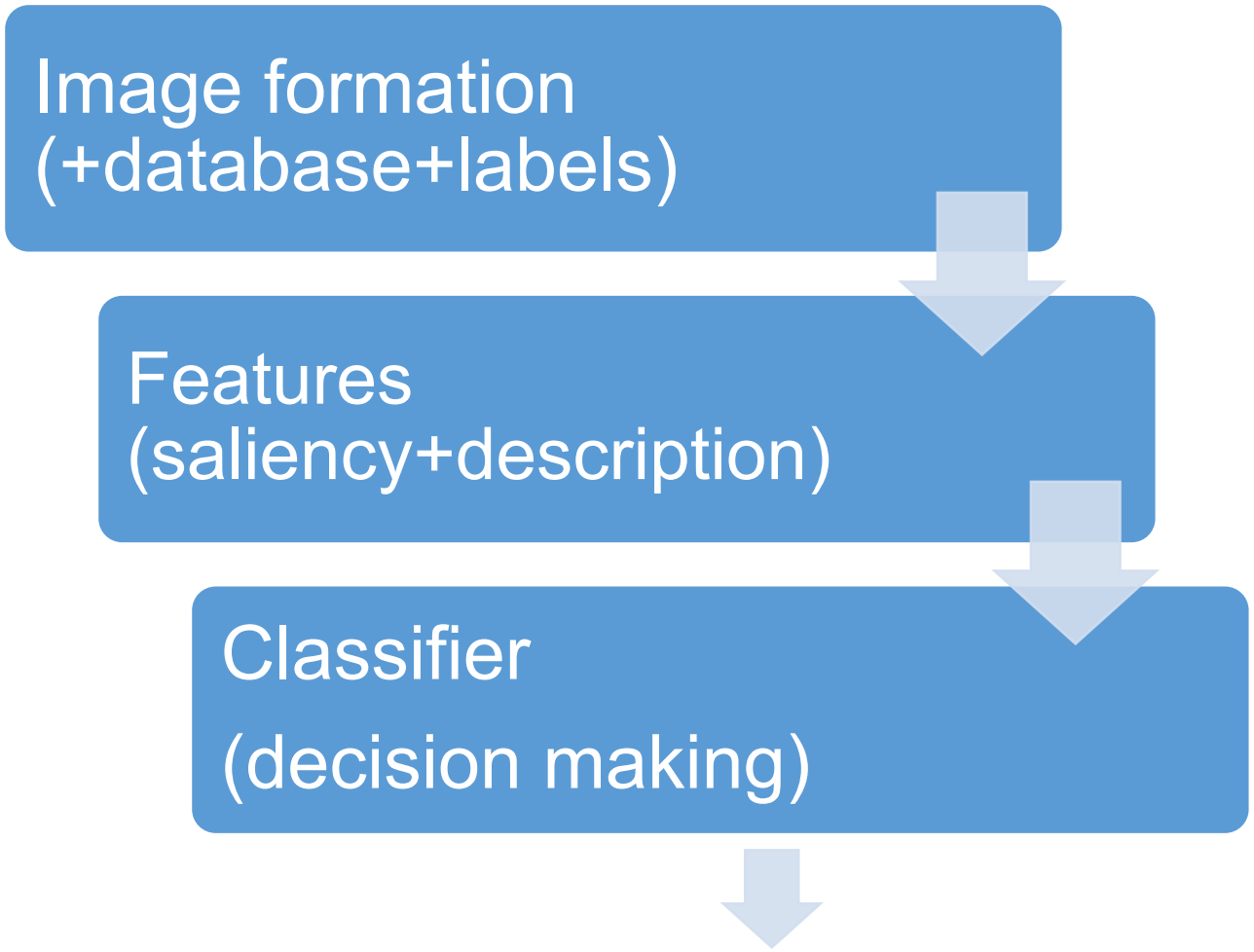
Etc.

Learn how to combine in cascade with boosting





Viola Jones



Captured+manual.

Specified space, but selected automatically.

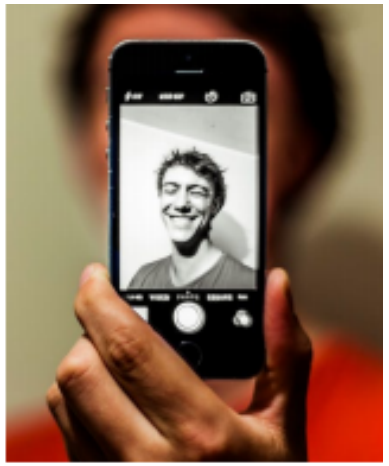
Learned combination.

Recognition:

Object Detection

Welcome







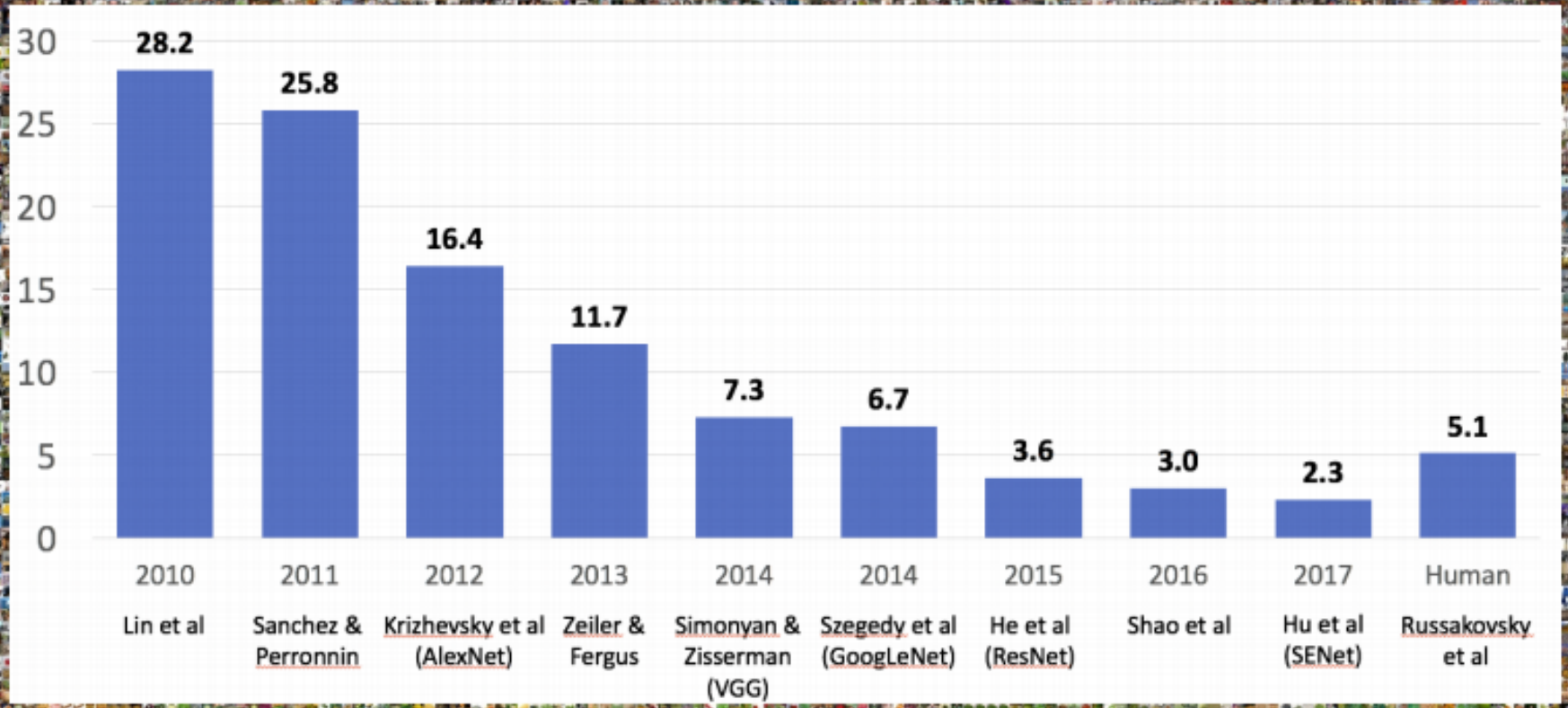


IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

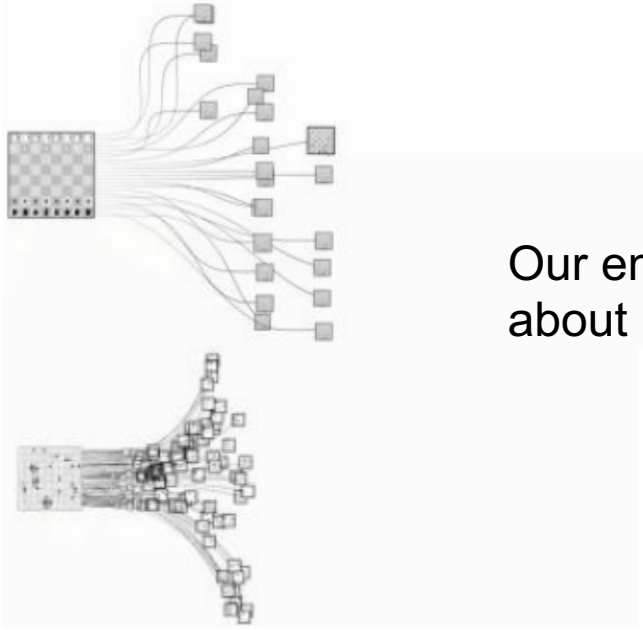
1,000 object classes

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

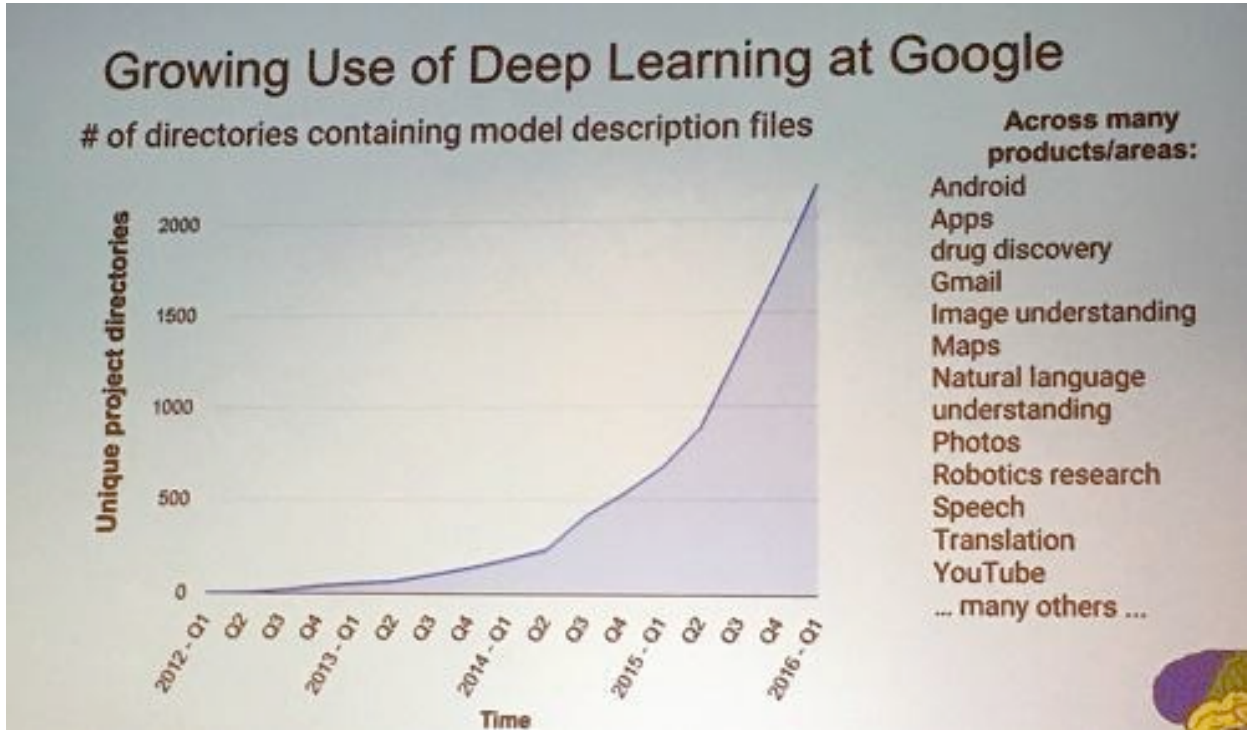
In "Nature" 27 January 2016:

- **"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."**

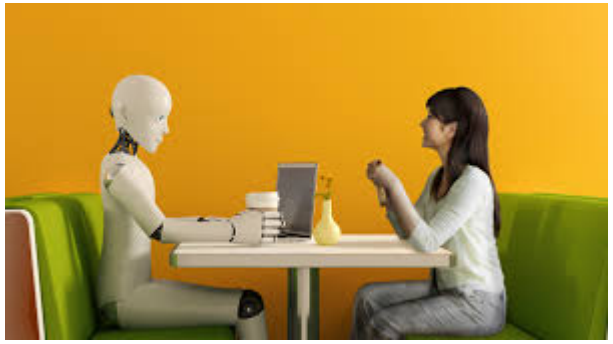
We (will) lose on many specific tasks!

- 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…



Nothing that has meaning is easy.
Easy doesn't enter into grown-up life.
-- Weatherman movie

- 单纯问题：难，但是目标明确，方向（方法）明确，有让人放心的答案，易衡量成败。如高考。

- 复杂问题：
  - **两难问题**：没有唯一正确的方向，必须取舍。如买房、子女教育，…
  - **棘手问题**：给别人做主；问题没有清晰定义；没有终结答案；解决方法没有对错，只有好坏（好坏的标准也是即个人化的）；前人经验没有大的帮助【**知识**】；没有试错练习的地方【**数据**】；没有即刻的**反馈**【**增强学习**】，常有意想不到的结果；…

人工智能潮起潮落

达特茅斯会议：7个问题，4个刚起步：

1. **第三个，神经网络；**
2. **第五个，自我改进；**
3. **第六个，抽象；**
4. **第七个，随机性和创造性**

Titans

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


Yoshua Bengio


Yann LeCun

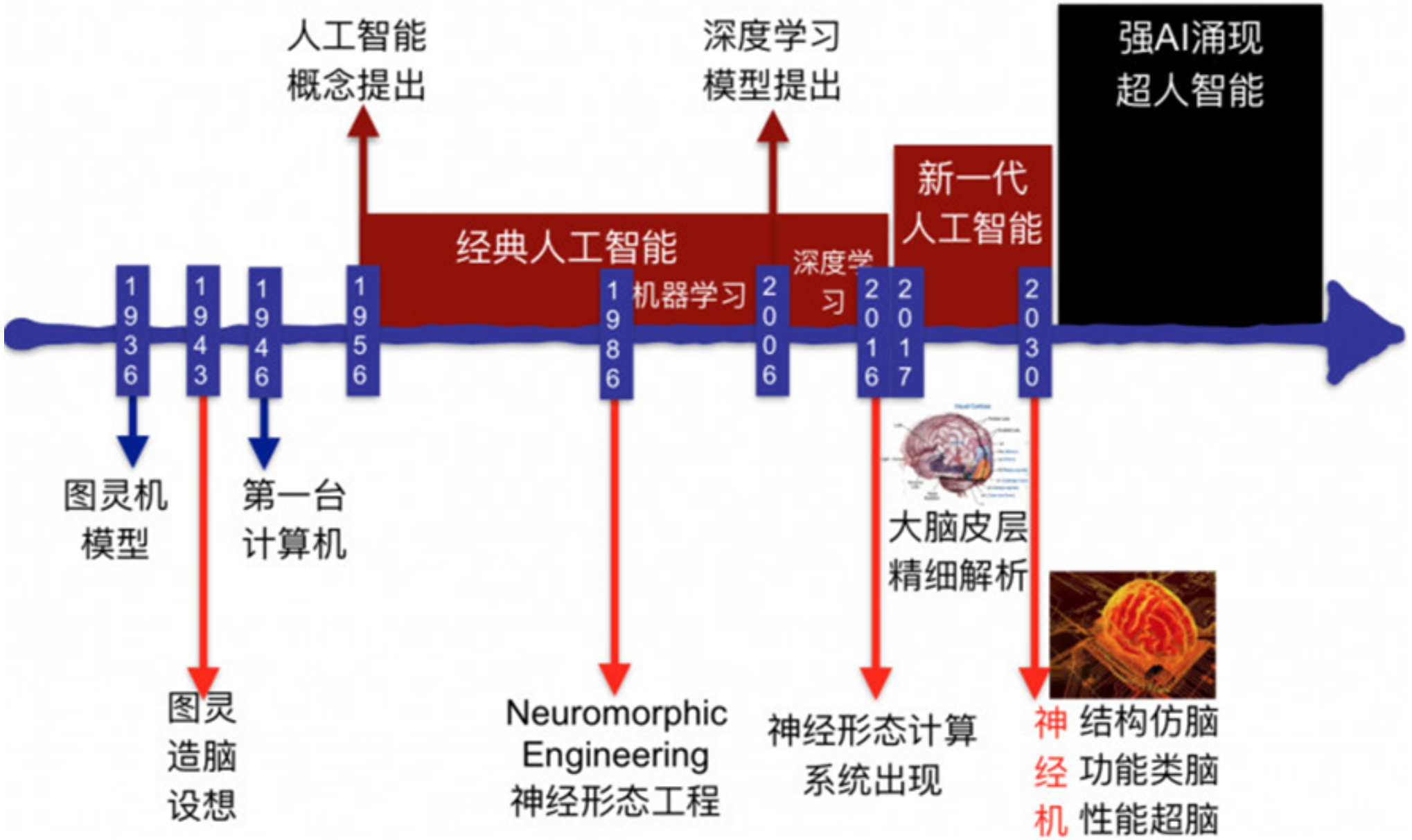

Geoffrey Hinton


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

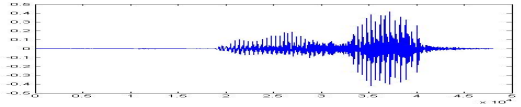
Platform:	Human brain	Neurogrid	BrainScaleS	TrueNorth	SpiNNaker
Technology:	Biology	Analogue, sub-threshold	Analogue, over threshold	Digital, fixed	Digital, programmable
Microchip:		Neurocore	HiCANN		18 ARM cores
Feature size:	10 μm[a]	180 nm	180 nm	28 nm	130 nm
# transistors:		23 M	15 M	5.4 B	100 M
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″ racks		600 PCBs in 6 × 19″ 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

$f($  $)=$ “How are you”

- Image Recognition

$f($  $)=$ “Cat”

- Playing Go

$f($  $)=$ “5-5” (next move)

- Dialogue System

$f($ “Hi” $)=$ “Hello”

(what the user said) (system response)

Framework

Image Recognition:

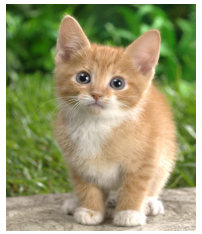
$f($  $)=$ “cat”

A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $)=$ “cat”

$f_2($  $)=$ “money”

$f_1($  $)=$ “dog”

$f_2($  $)=$ “snake”

Framework

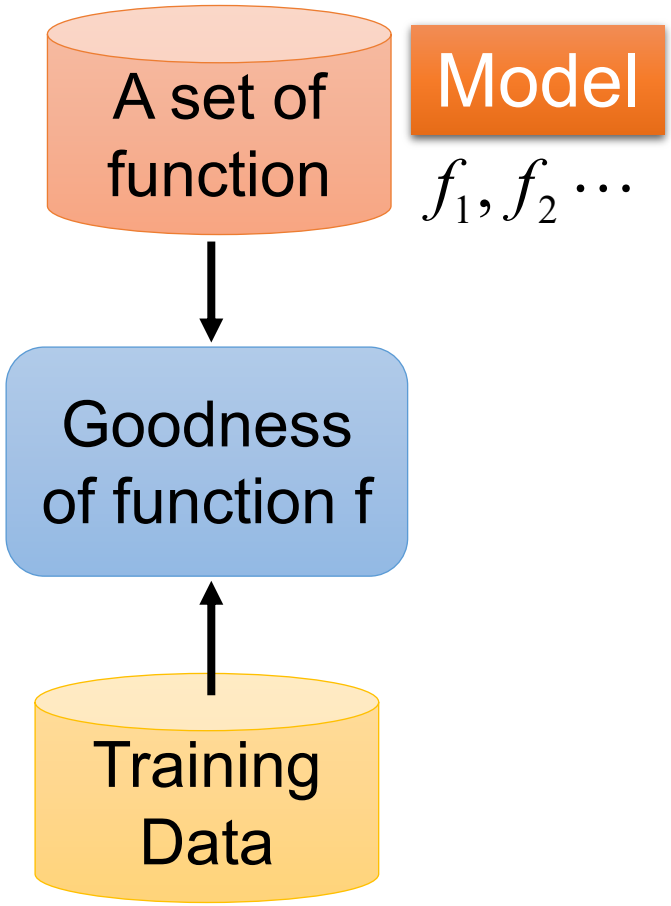


Image Recognition:

$f($  $) =$ "cat"

$f_1($  $) =$ "cat"

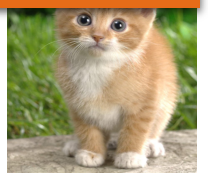
$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Supervised Learning

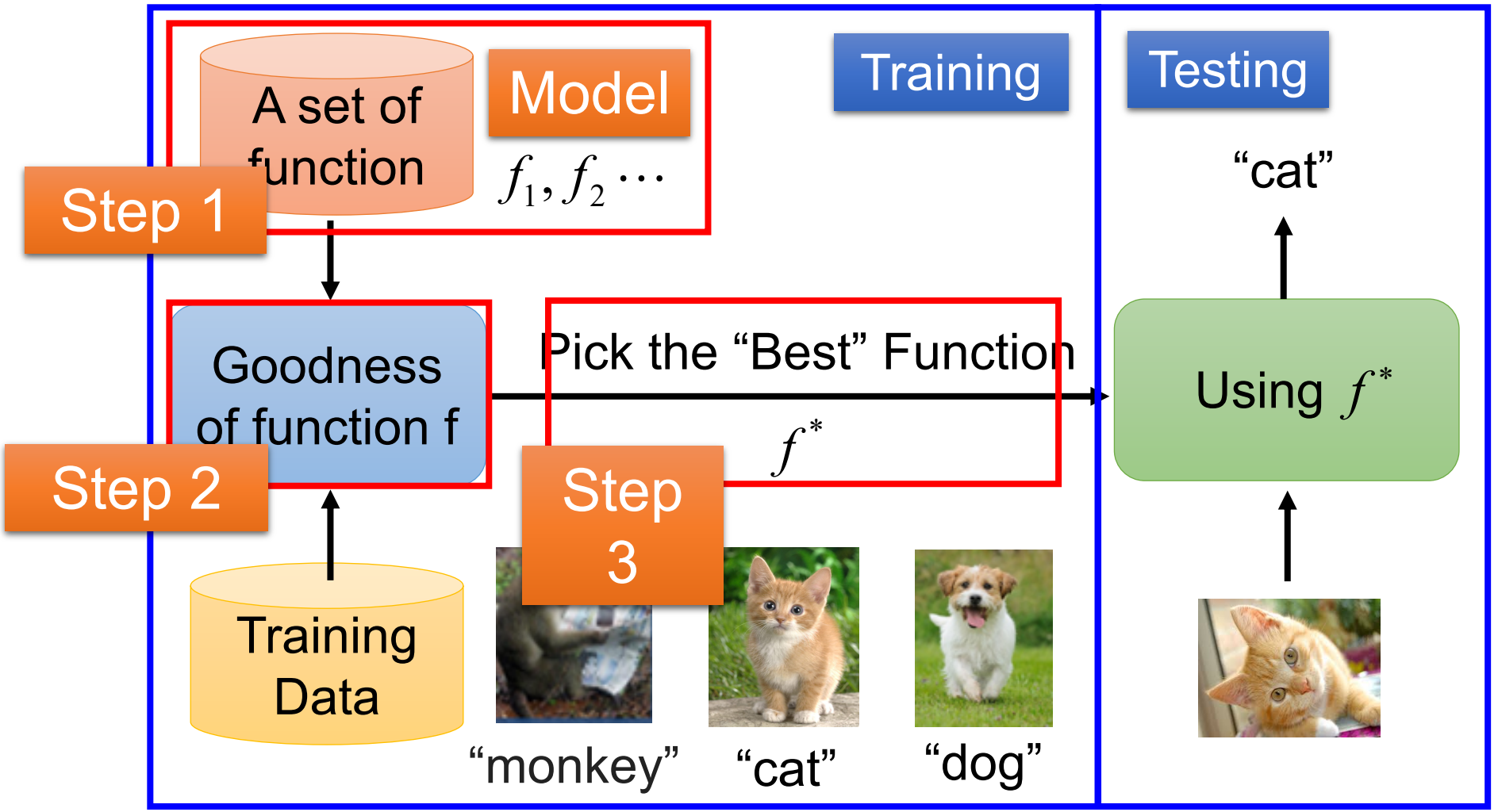
function input:   

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

Framework

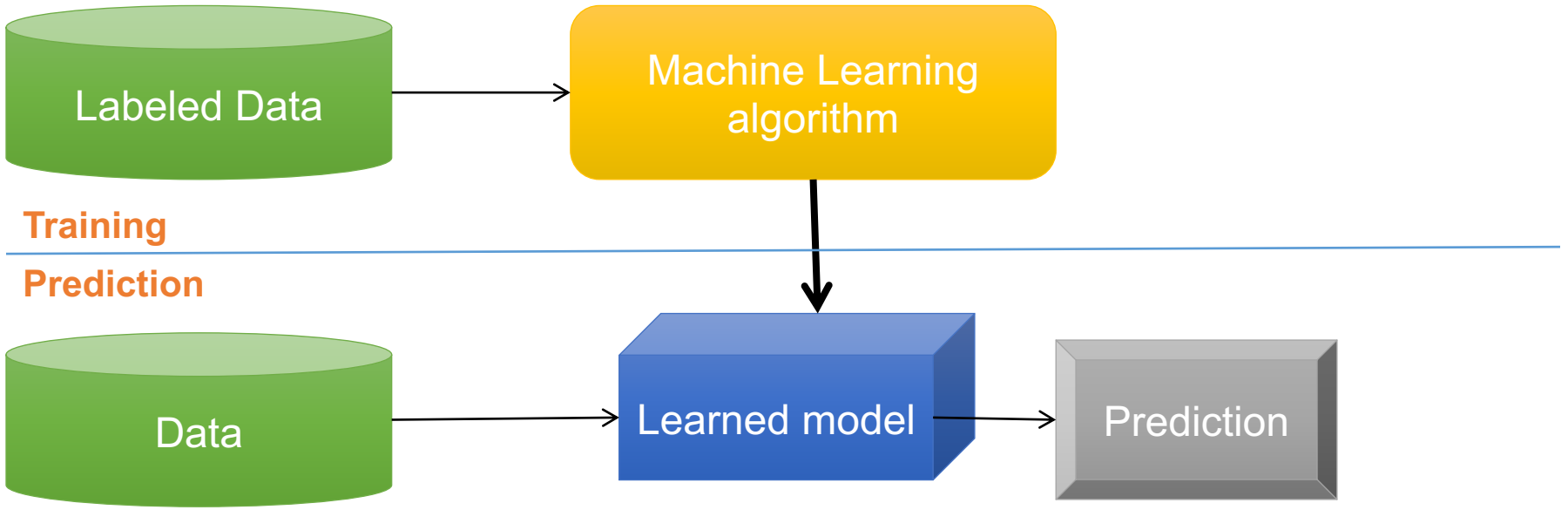
Image Recognition:

$f($  $) =$ "cat"



Machine Learning Basics

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



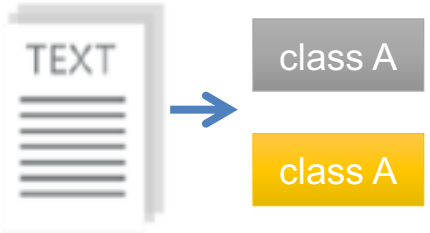
Methods that can learn from and make predictions on data

Types of Learning

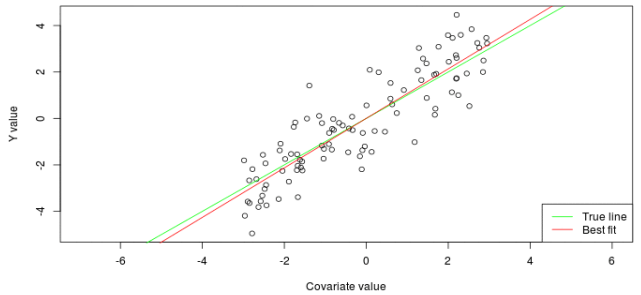
- **Supervised**: Learning with a **labeled training** set
  - Example: email *classification* with already labeled emails

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

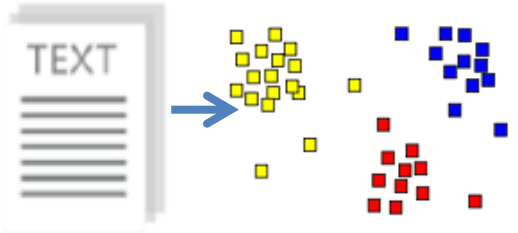
- **Reinforcement learning**: learn to **act** based on **feedback/reward**
  - Example: learn to play Go, reward: *win or lose*



Classification



Regression



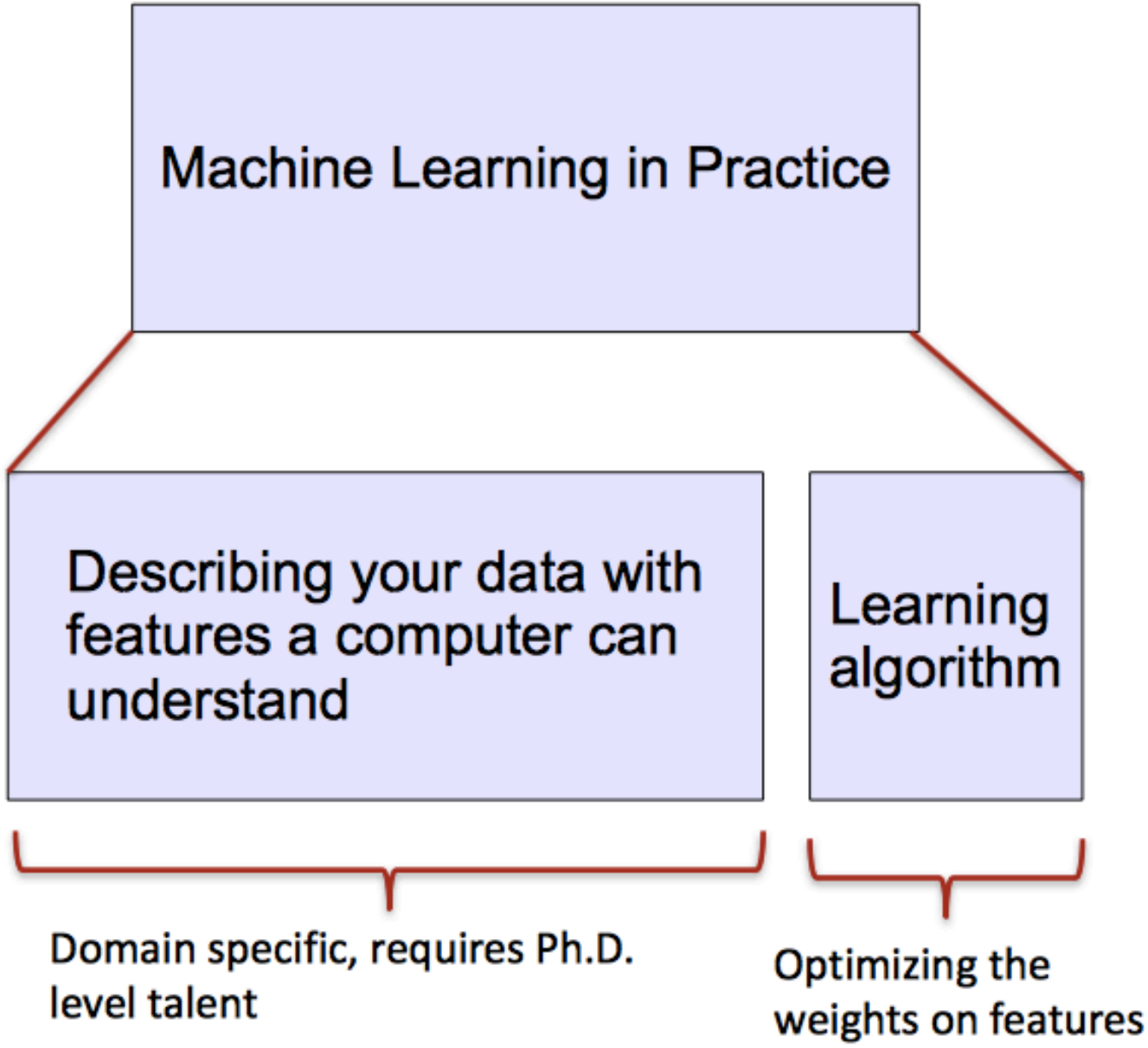
Clustering

Anomaly Detection
Sequence labeling
…

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

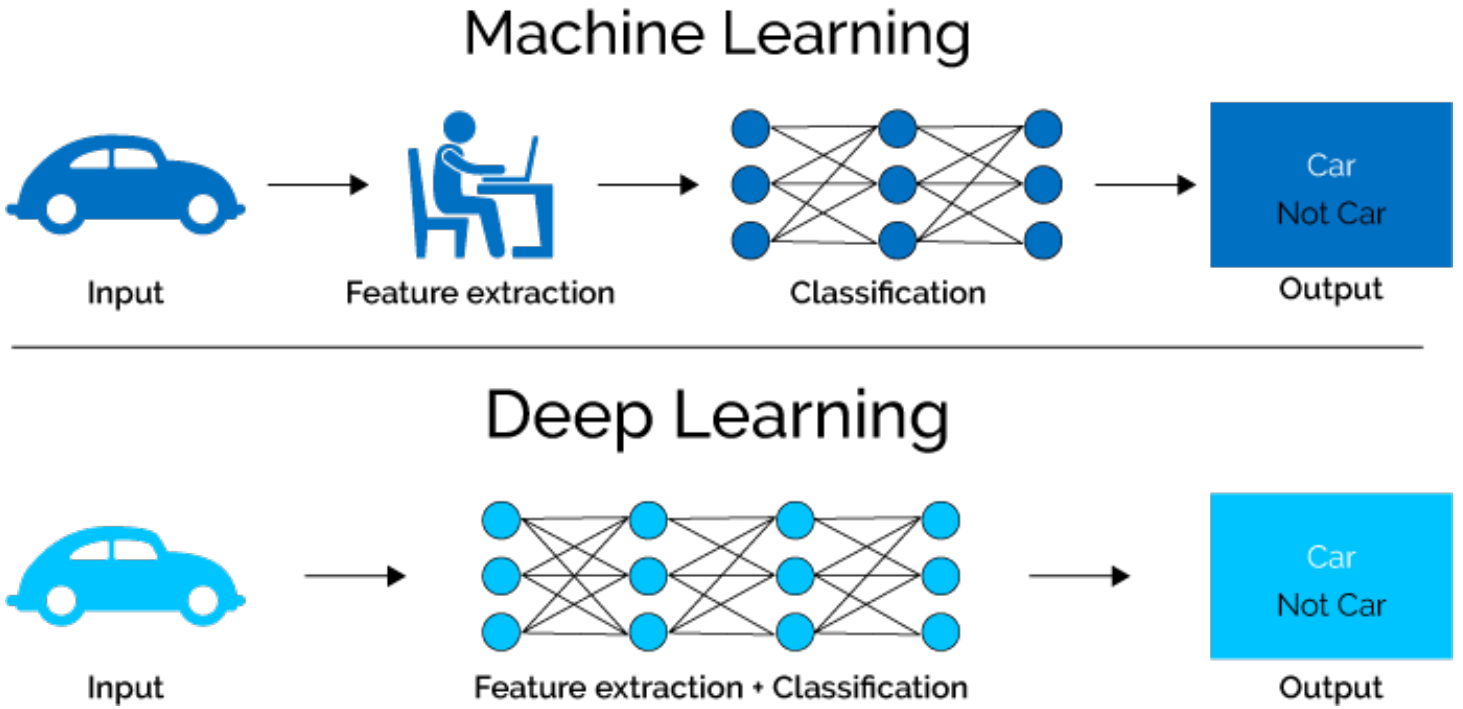
ML vs. Deep Learning

- **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.

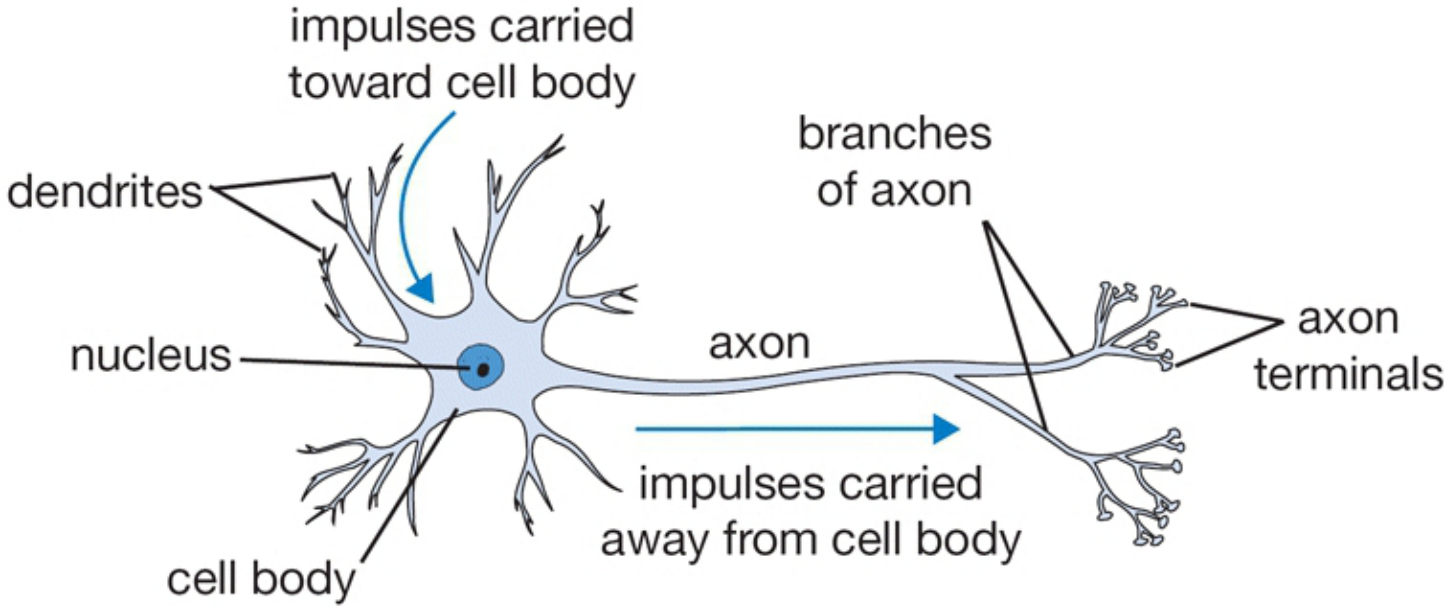


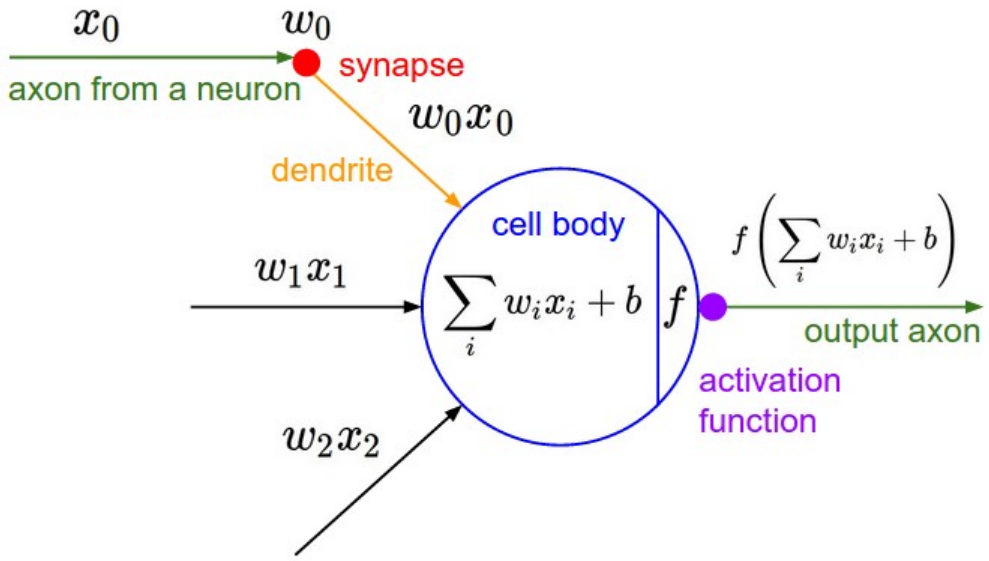
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

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

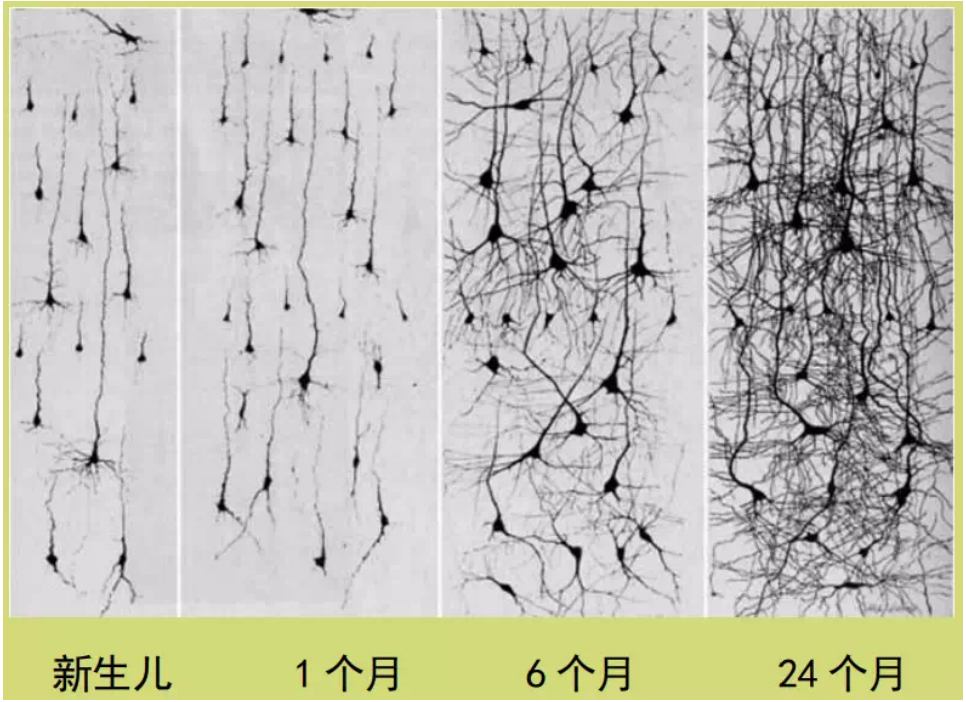


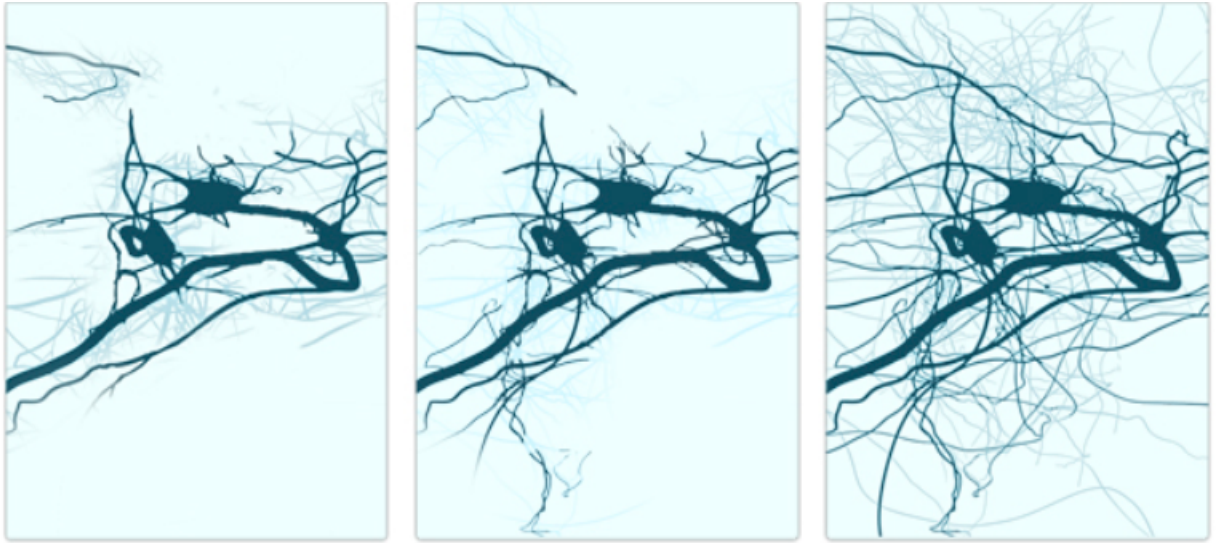


- 神经网络中的神经元的灵感来源于人脑，人体中大约有860亿个神经元neuron，大约有 10^14 - 10^15 突触（synapses）.
- 每个神经元由树突dendrites接收信号轴突axon发射信号. The axon eventually branches out and connects via synapses to dendrites of other neurons.
- **信号x, 权重w(突触强度synaptic strengths，可训练), 经过细胞体求合之后的信号大于阈值，神经元就被激活（fire）, 通过axon释放信号（spike）.**

神经可塑性

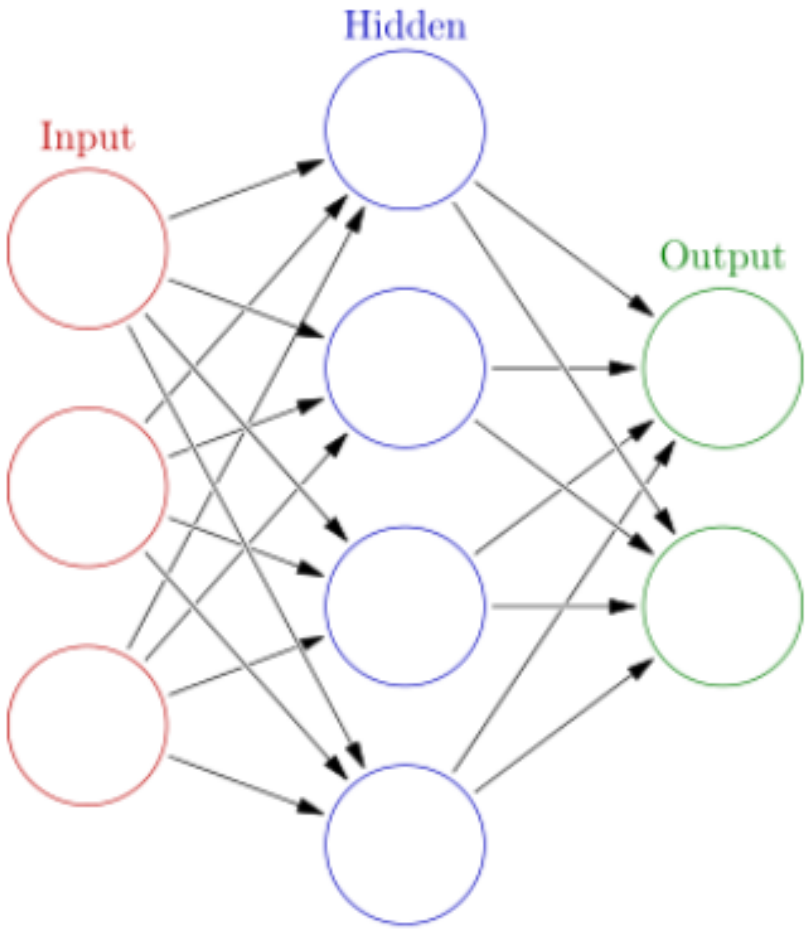
- **肌肉、血管的可塑性；**
- **“基因难以改变”**
- **连续的大脑训练可以增长神经网络连接；**
- 《关键20小时快速学会任何技能》
  - 日常的重复！=“刻意练习”（deliberate practice）





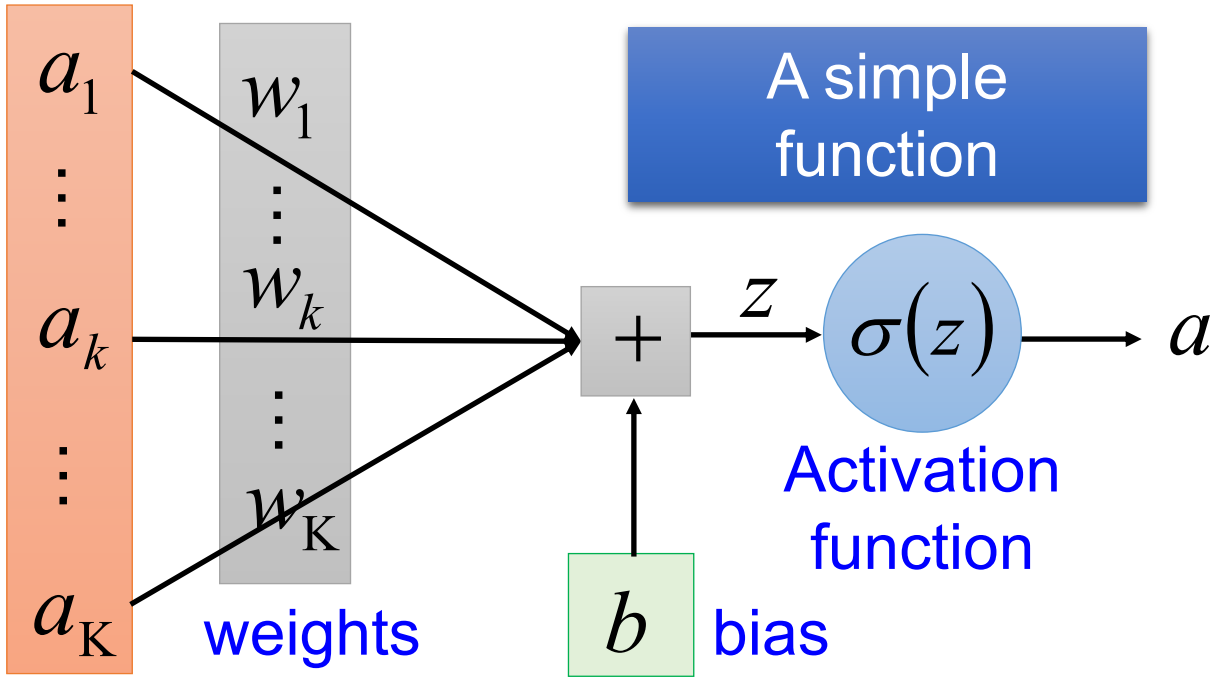
经过2个月刺激（训练）后的脑神经局部

Neural Network

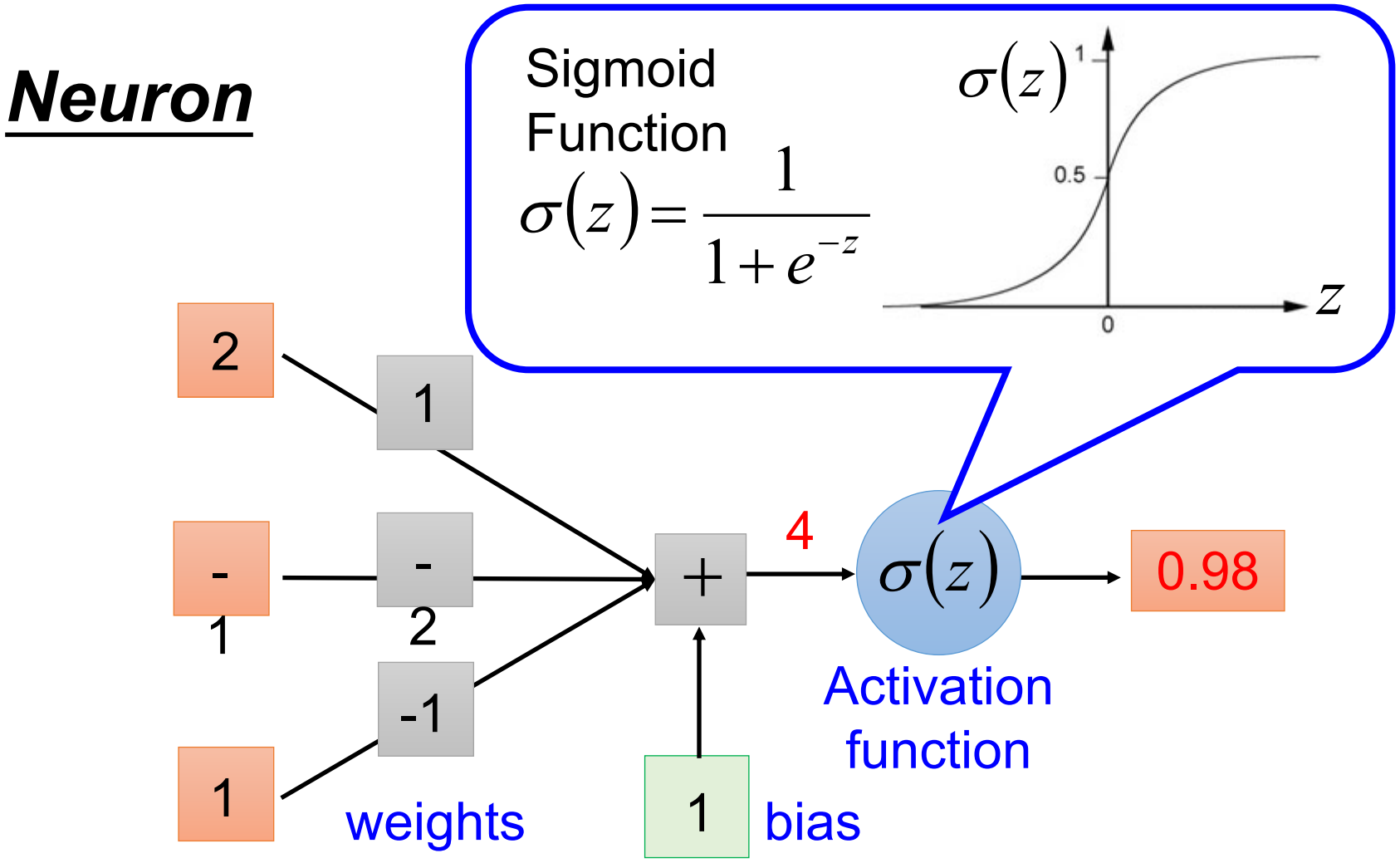


Neuron

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

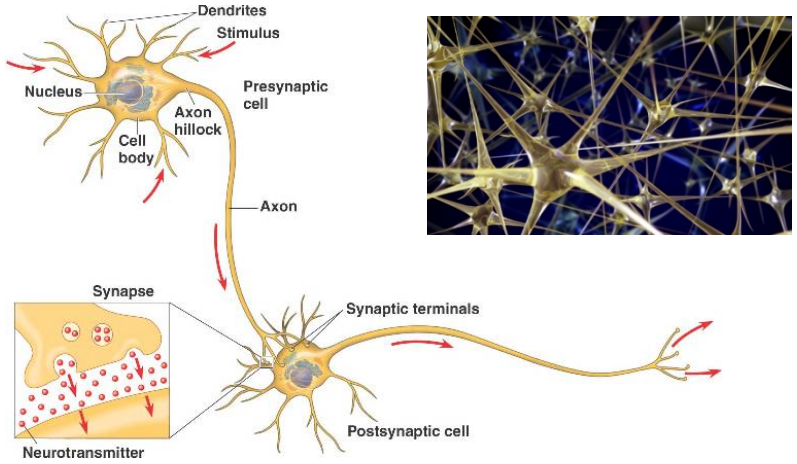


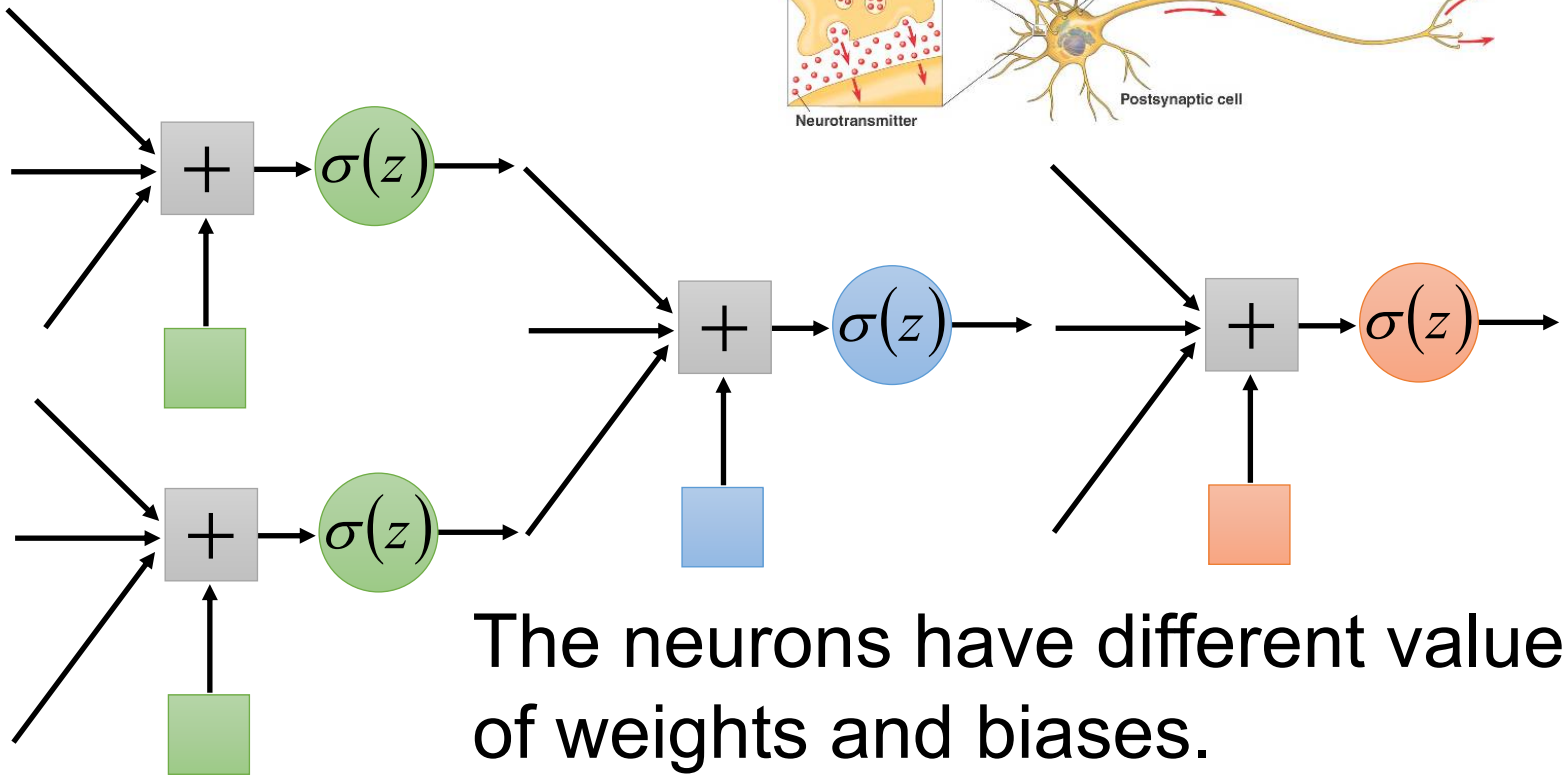
Neural Network



Neural Network

Different connections lead to different network structures

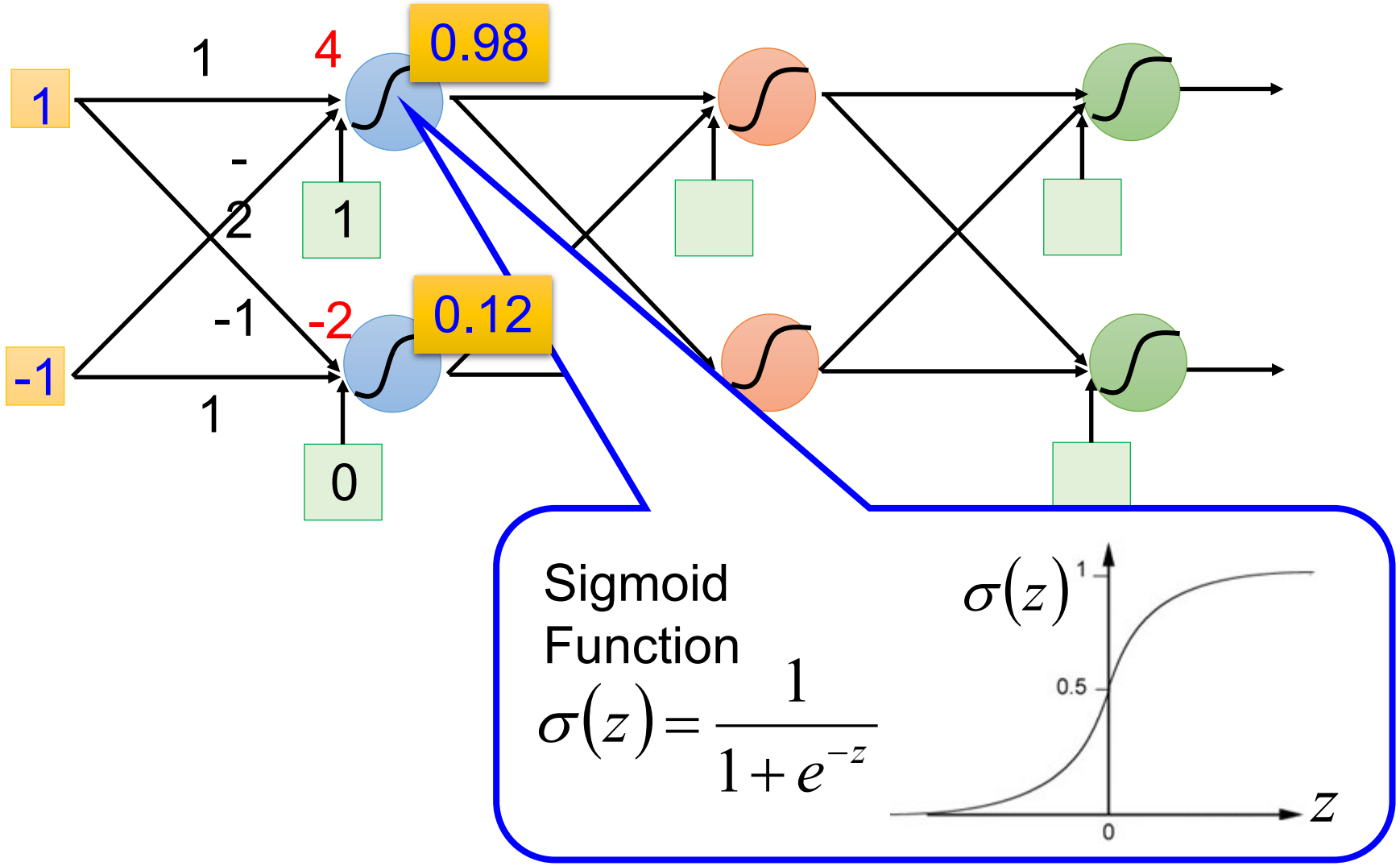




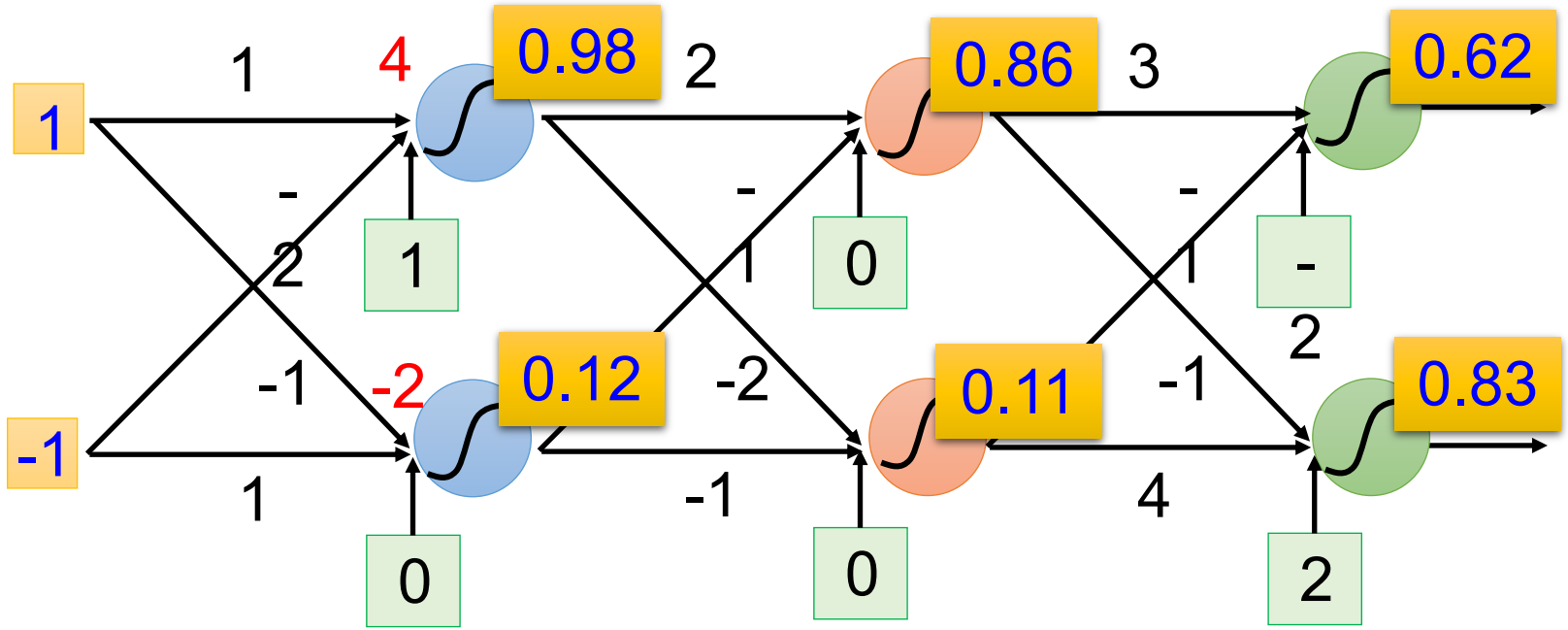
The neurons have different values of weights and biases.

Weights and biases are network parameters θ

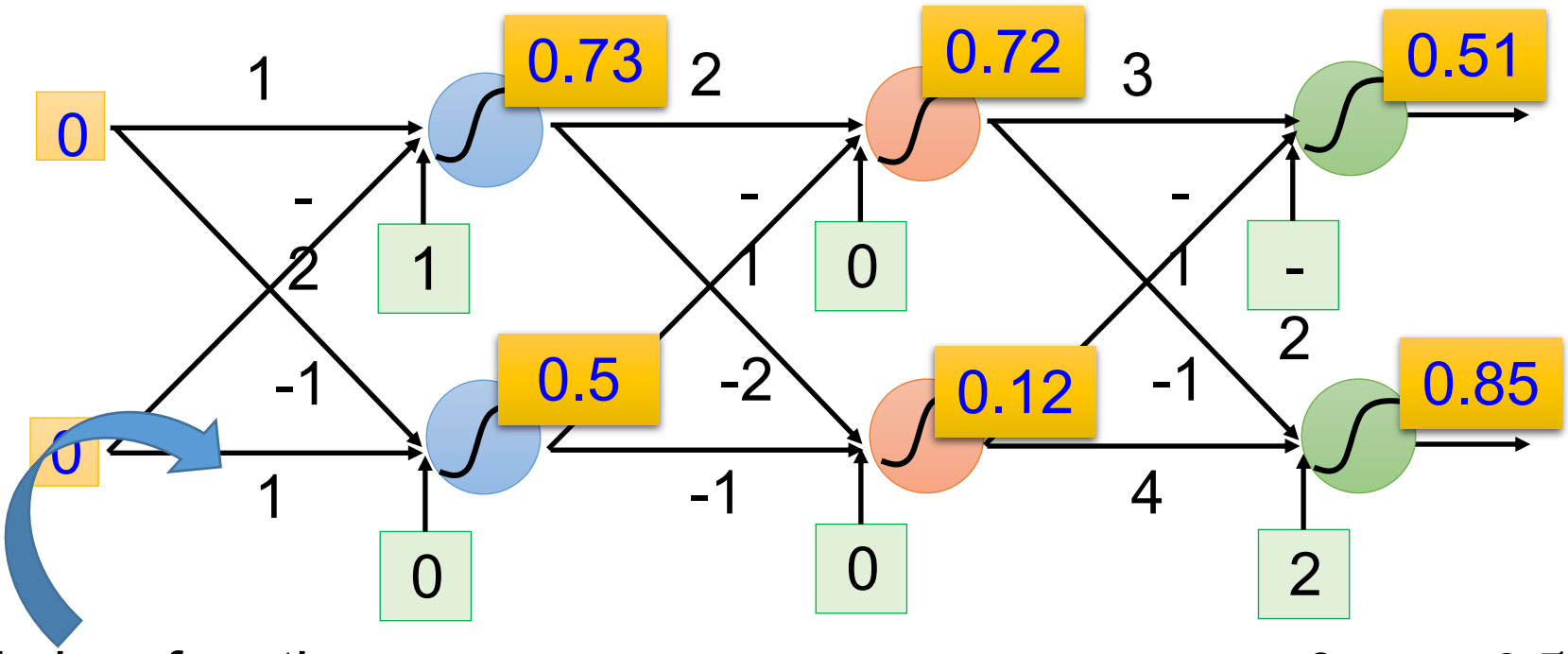
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

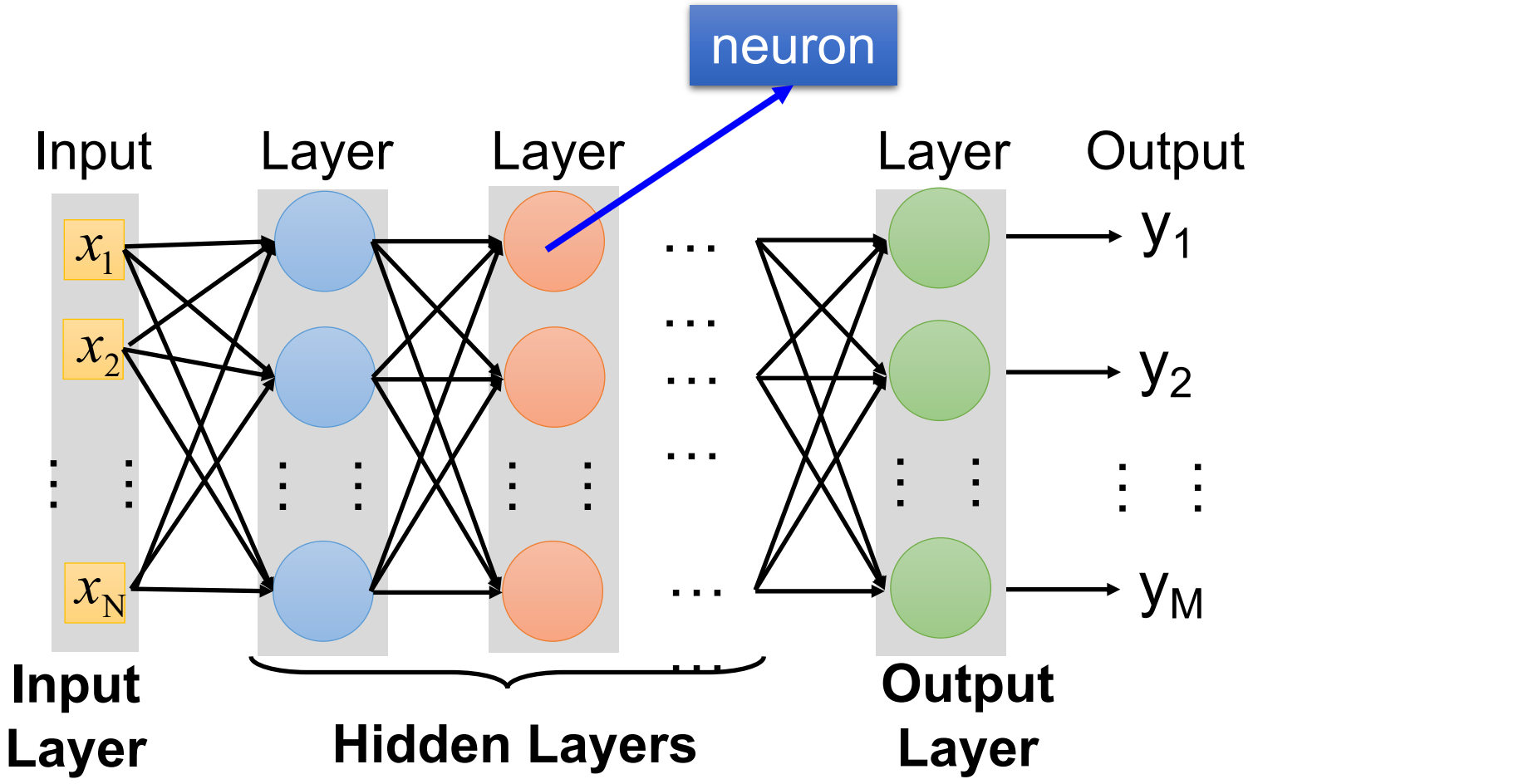
Input vector, output vector

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

Given parameters θ, define a function

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

Fully Connect Feedforward Network



Deep means many hidden layers

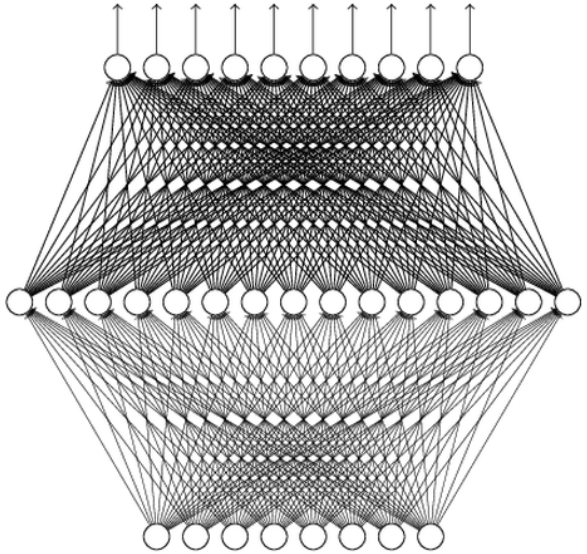
Why Deep? Universality Theorem

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?

If a single-layer network can learn any function…

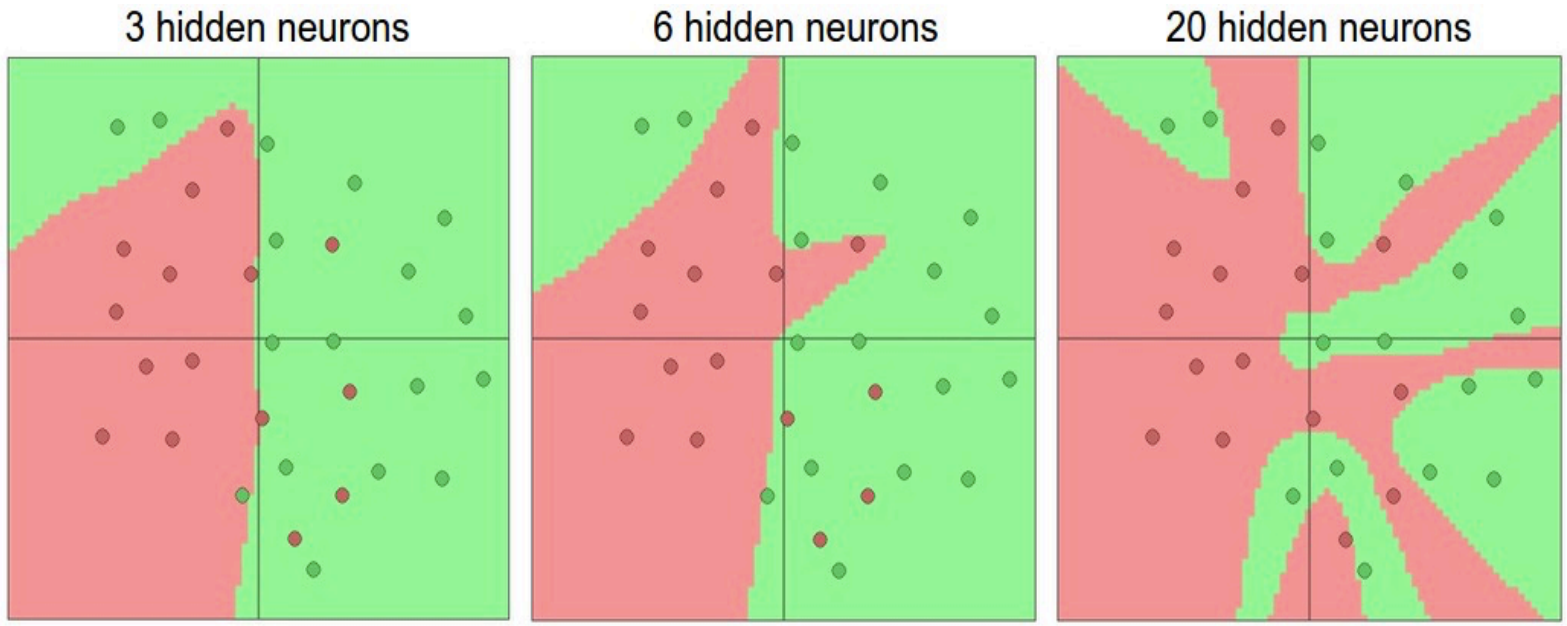
…given enough parameters…

…then ***why do we go deeper?***

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

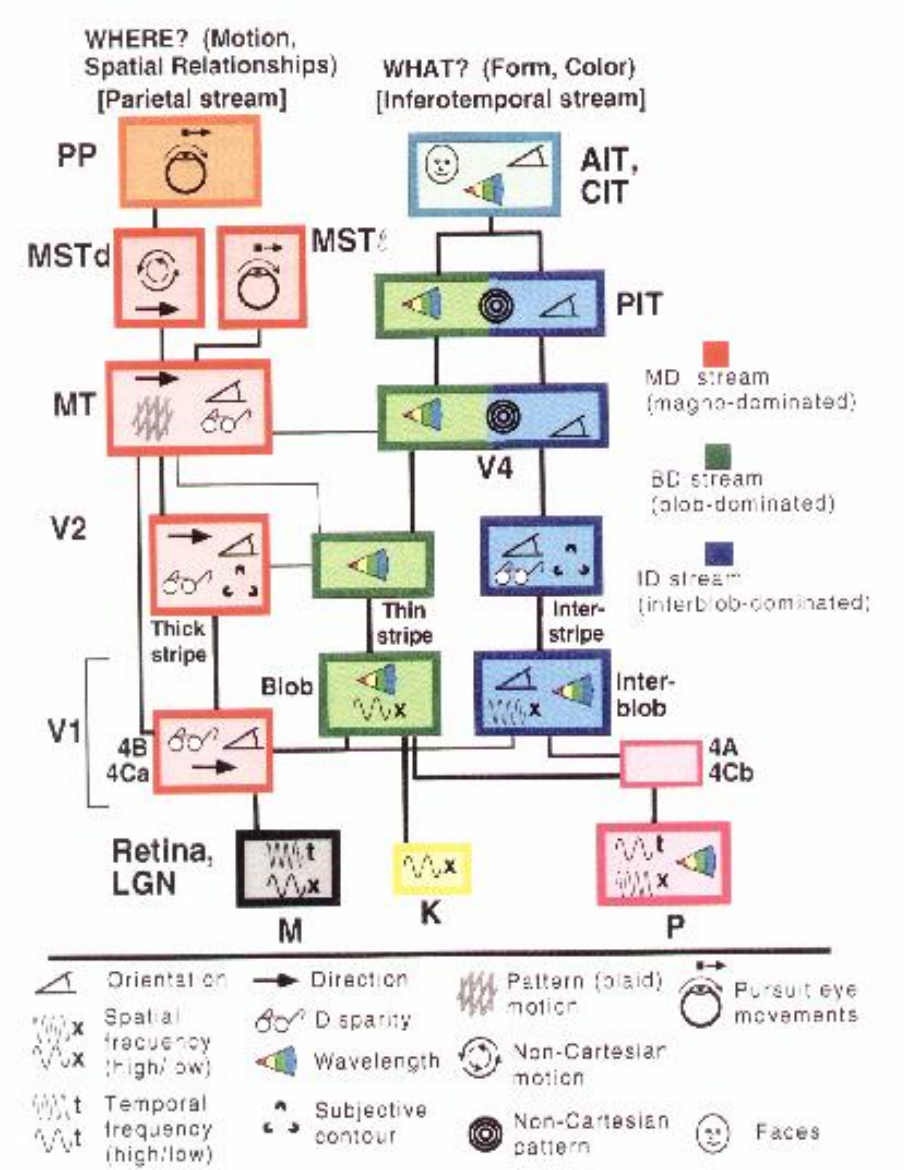
Full list: https://en.wikipedia.org/wiki/Activation_function

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

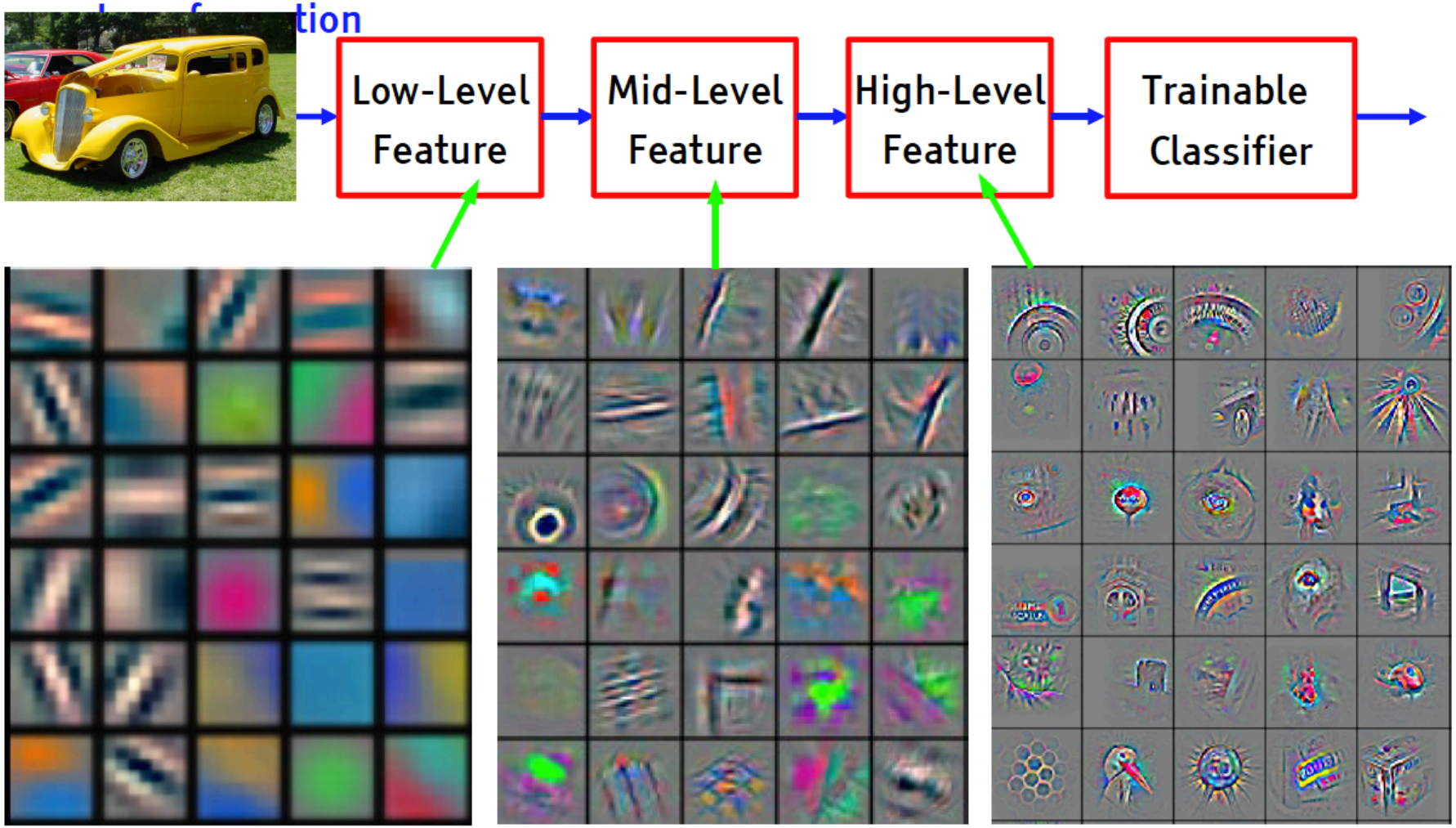


(van Essen and Gallant, 1994)

- 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.

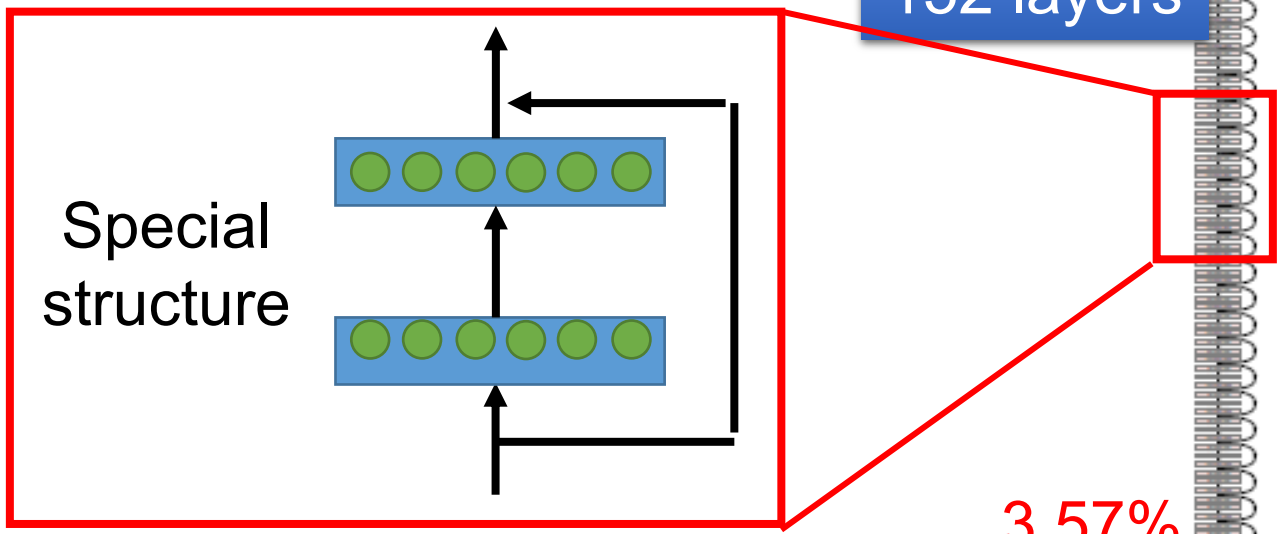
Deep Learning = Learning Hierarchical Representations

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



152 layers

101 layers

3.57%

16.4%

7.3%

6.7%

AlexNet
(2012)

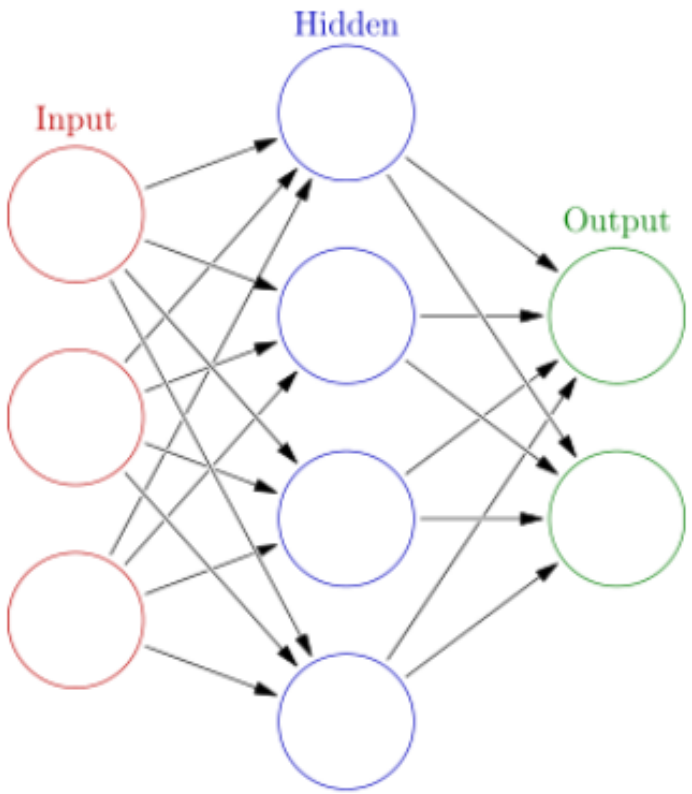
VGG
(2014)

GoogleNet
(2014)

Residual Net
(2015)

Taipei
101

Output Layer



Ordinary Layer

$z_1 \rightarrow \sigma \rightarrow y_1 = \sigma(z_1)$

$z_2 \rightarrow \sigma \rightarrow y_2 = \sigma(z_2)$

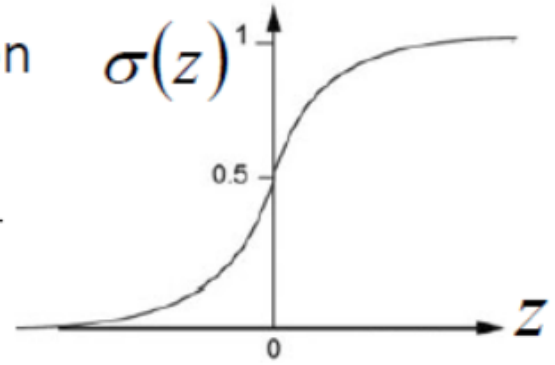
$z_3 \rightarrow \sigma \rightarrow y_3 = \sigma(z_3)$

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

May not be easy to interpret

Sigmoid Function

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



- Softmax layer as the output layer

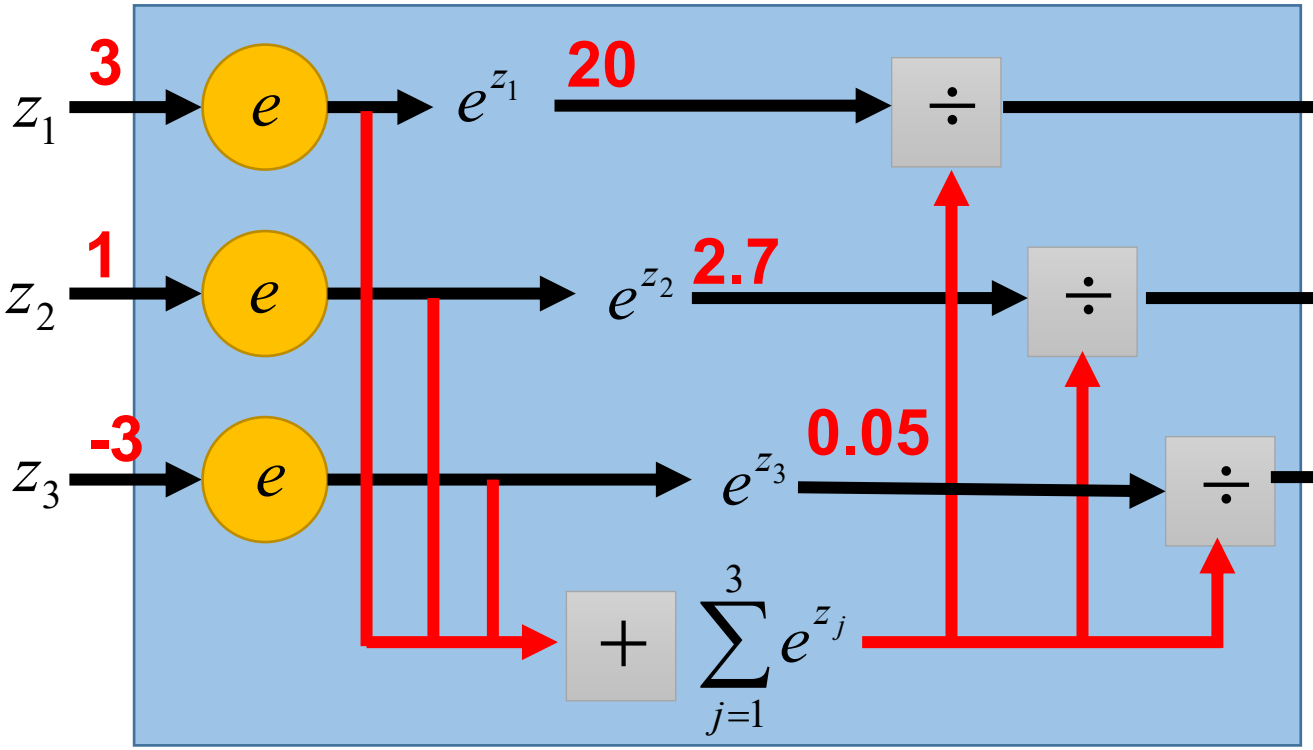
Output Layer

- Softmax layer as the output layer

Probability:

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

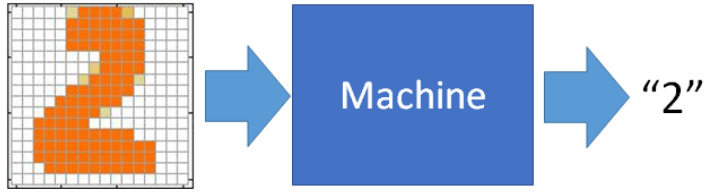
Softmax Layer



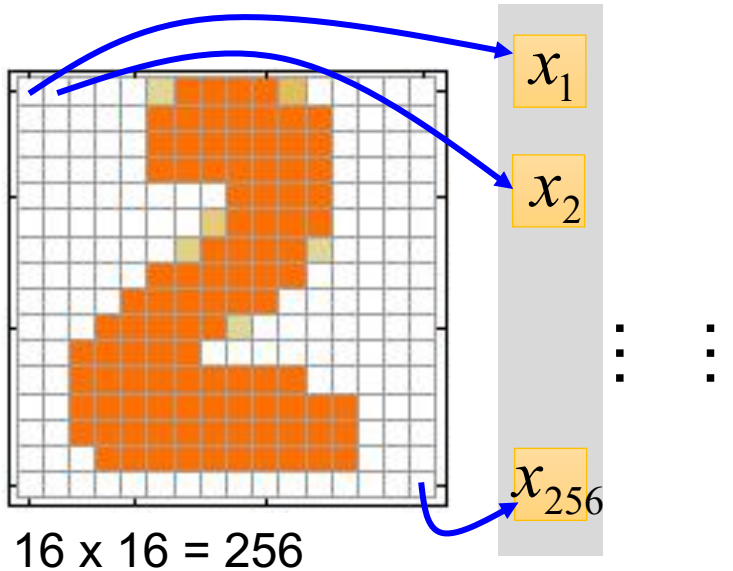
$$\sigma : \mathbb{R}^K \to \left\{ z \in \mathbb{R}^K \mid z_i > 0, \sum_{i=1}^{K} z_i = 1 \right\}$$

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

Example Application

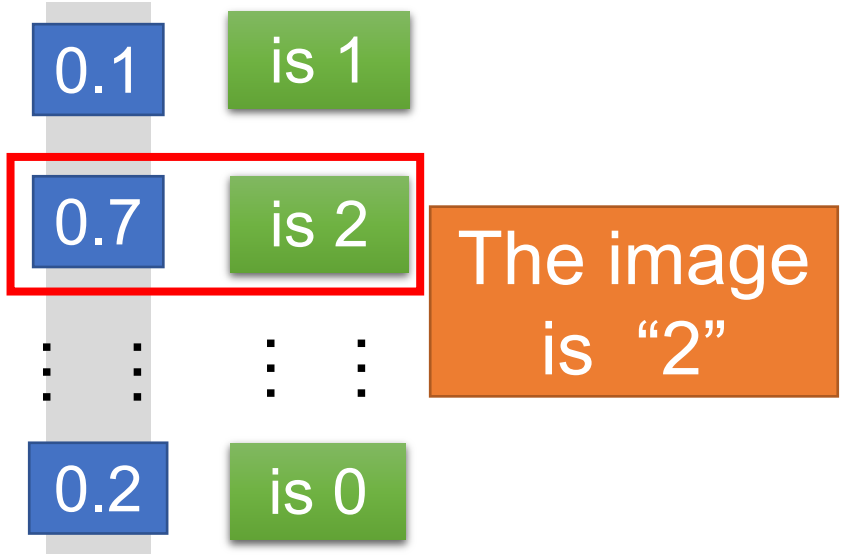


Input



Ink → 1
No ink → 0

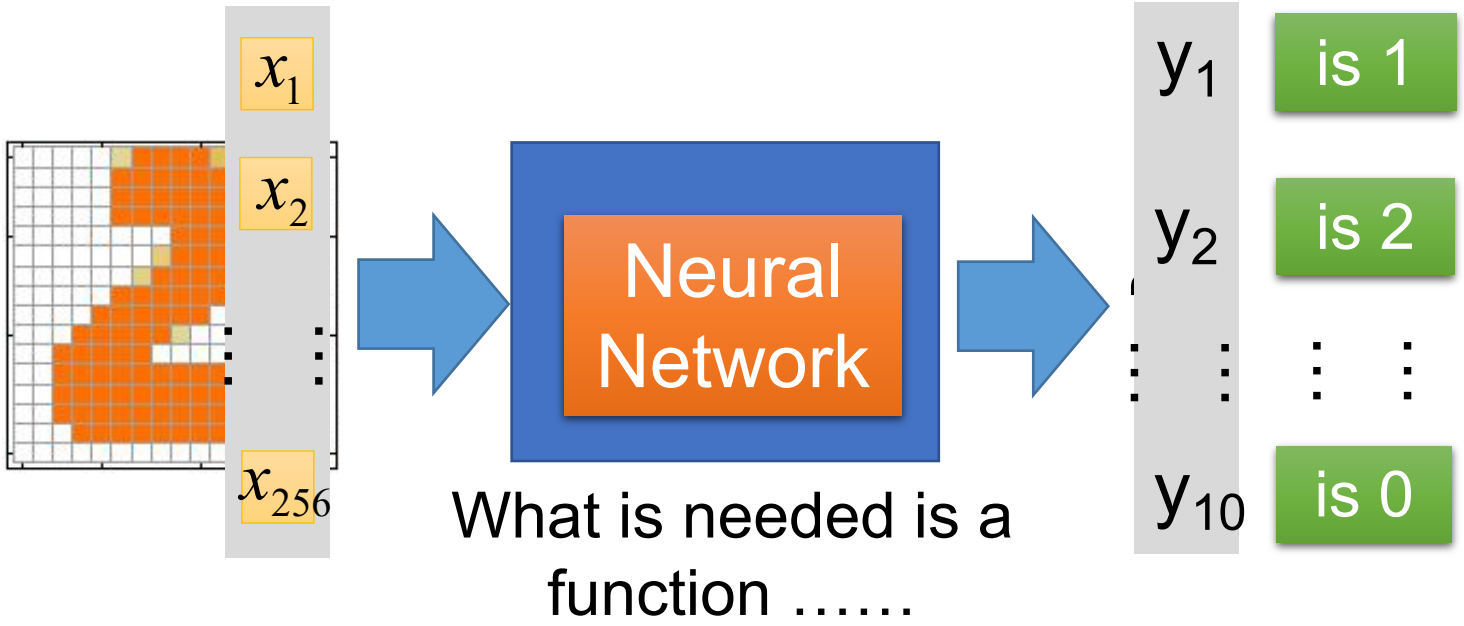
Output



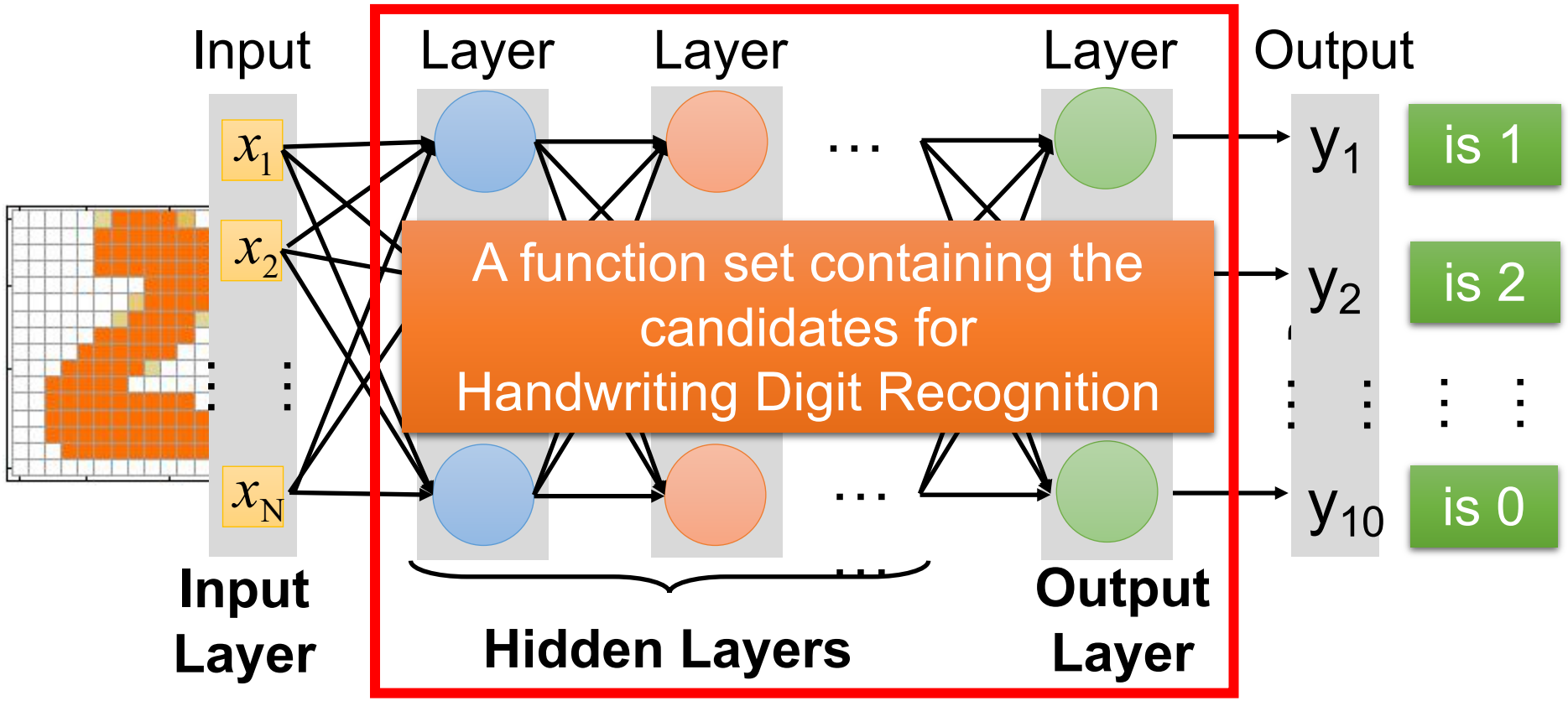
Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition



Example Application



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

FAQ

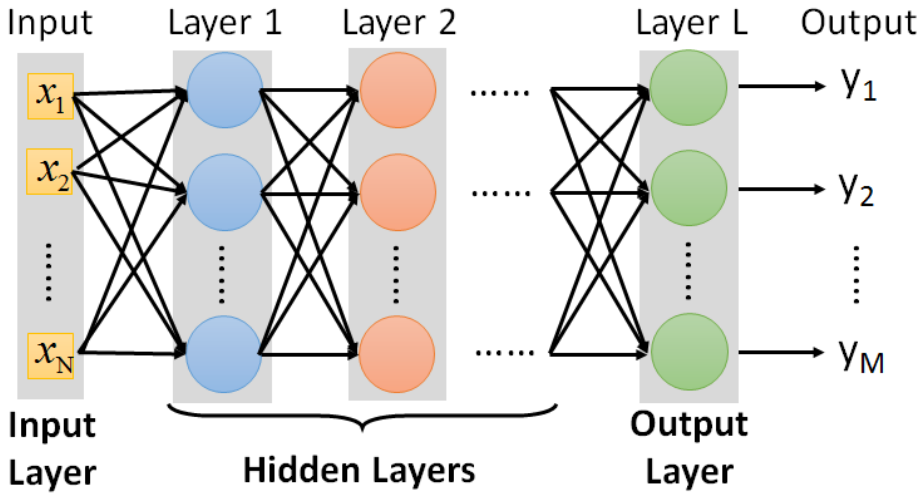
- Q: How many layers? How many neurons for each layer?

Trial and Error + Intuition

- Q: Can we design the network structure?

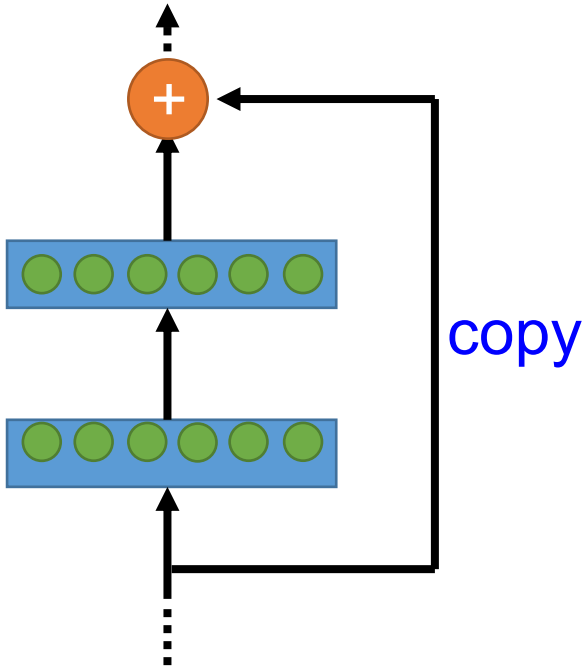
Convolutional Neural Network (CNN) in the next lecture

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



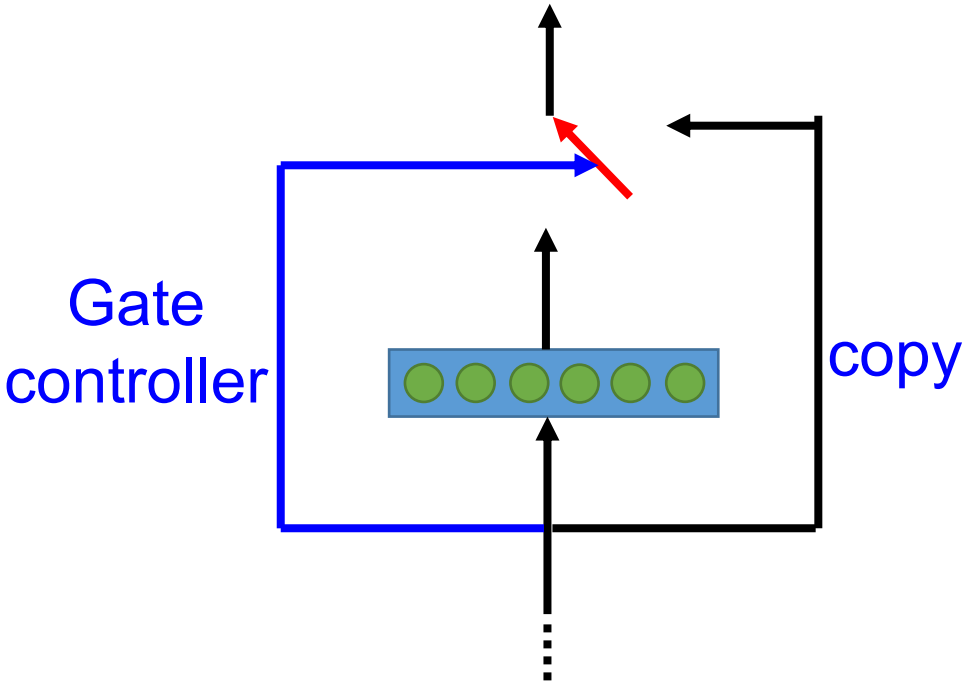
Highway Network

- **Residual Network**

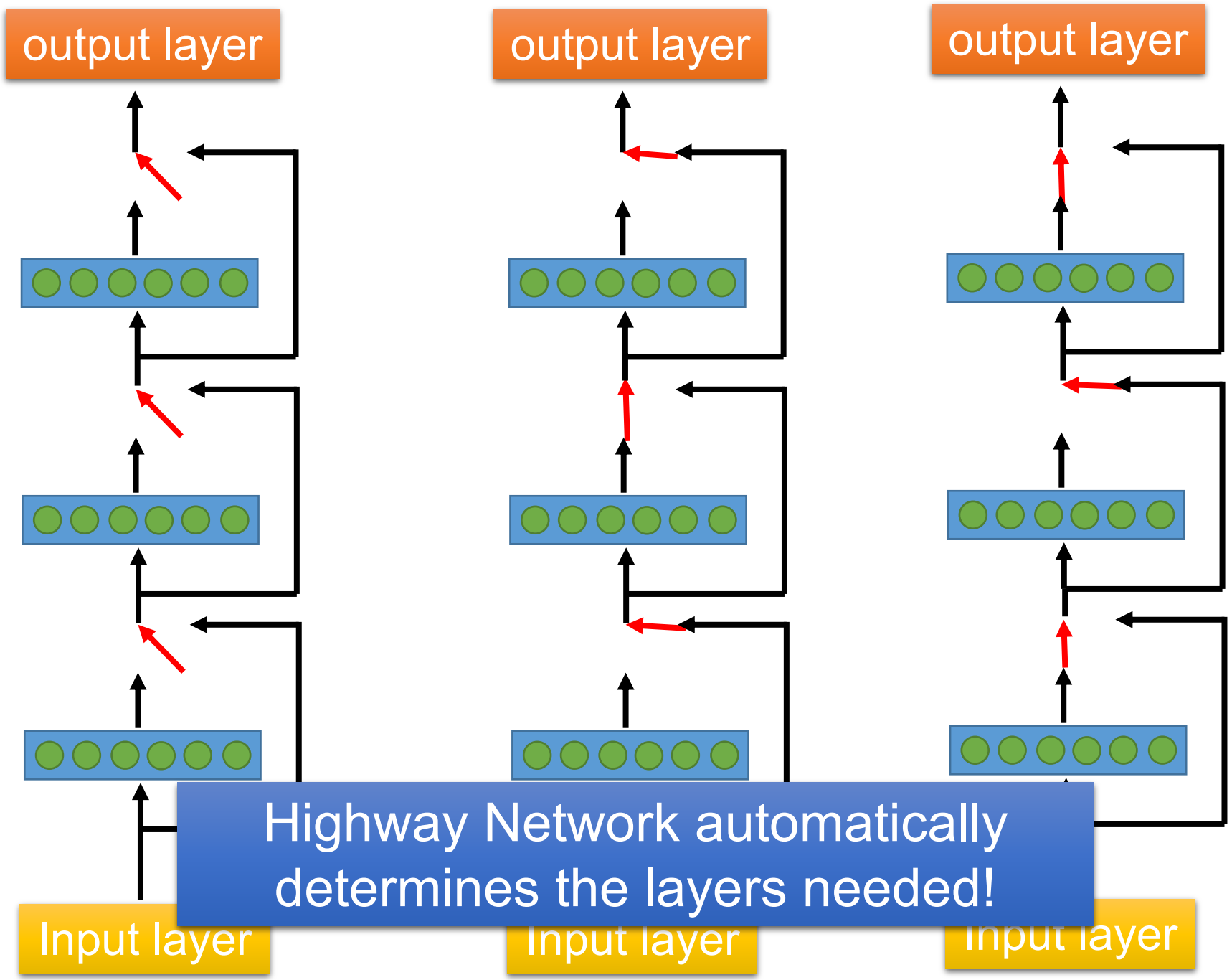


Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

- **Highway Network**

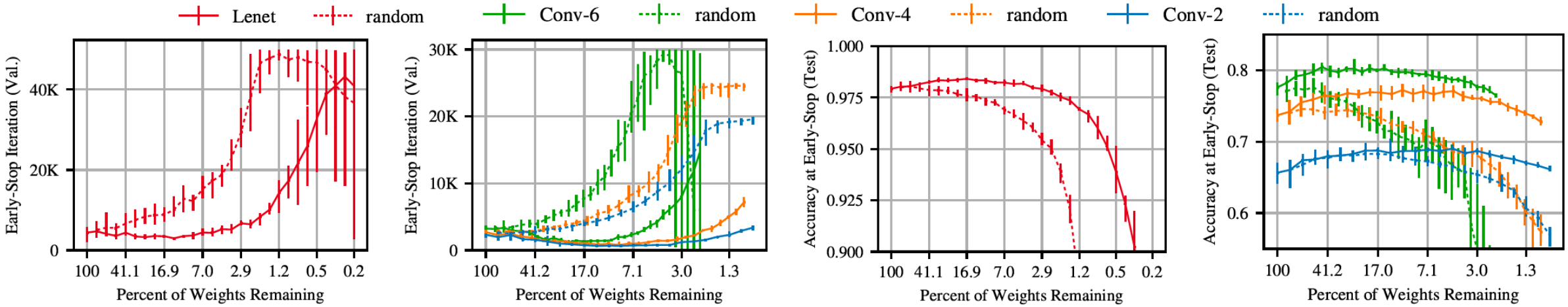


Training Very Deep Networks
https://arxiv.org/pdf/1507.06228v2.pdf


output layer
output layer
output layer
Highway Network automatically determines the layers needed!
Input layer
Input layer
Input layer

Pruning for NN & CNN

- Neural network pruning techniques can reduce the parameter counts of trained networks by **over 90%,** decreasing storage requirements and improving computational performance of inference **without compromising accuracy**.
- there consistently exist **smaller subnetworks** that **train from the start with the same initialization & learn at least as fast as their larger counterparts while reaching similar or better test accuracy**.



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks-ICLR 2019 best paper

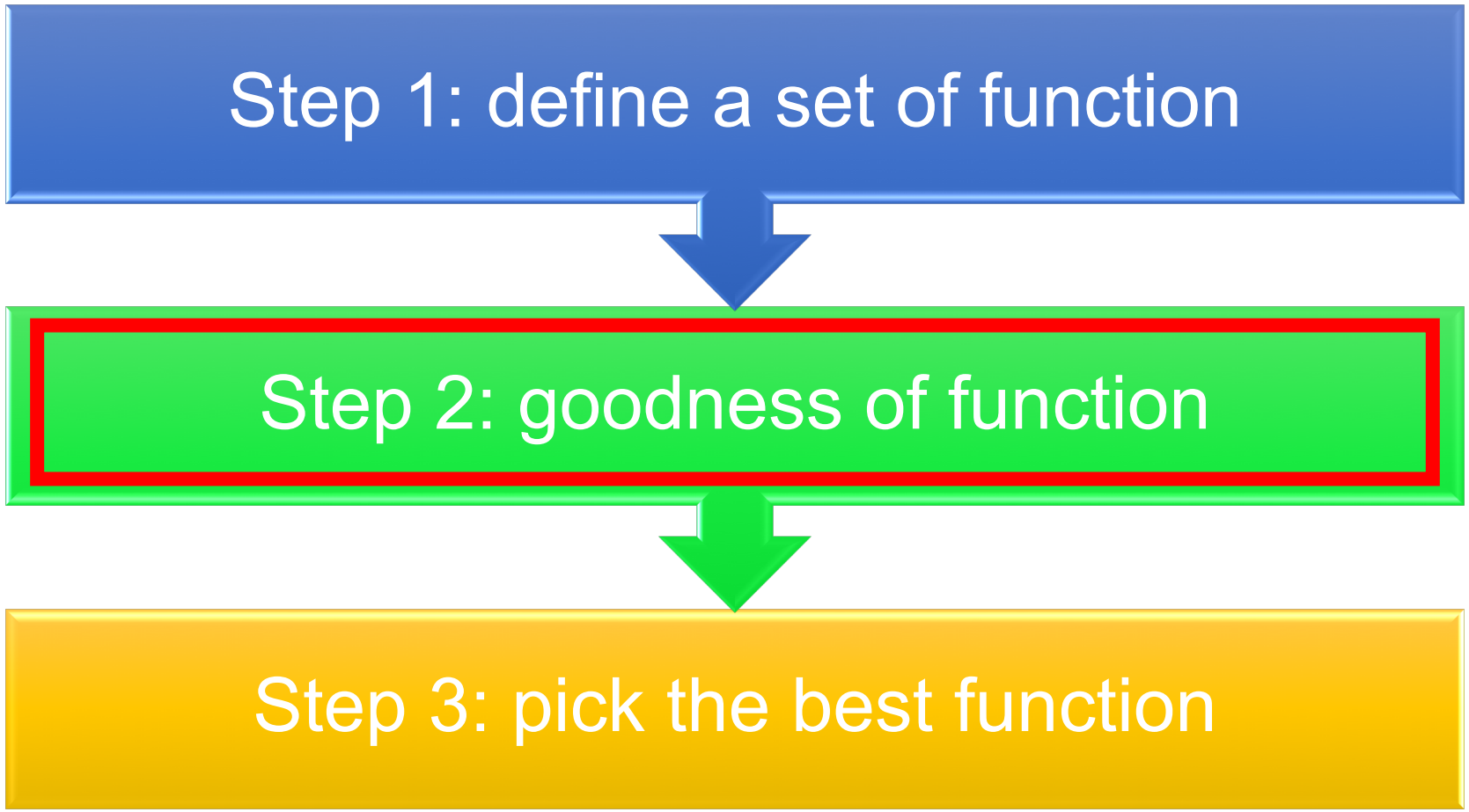
1. Randomly initialize a neural network $f(x; \theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$).
2. Train the network for j iterations, arriving at parameters θ_j.
3. Prune $p\%$ of the parameters in θ_j, creating a mask m.
4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

4. Reset the remaining parameters to their values in θ_0, creating the winning ticket $f(x; m \odot \theta_0)$.

- One-shot approach => iterative pruning
  - On deeper networks (Resnet-18 and VGG-19), iterative pruning is unable to find winning tickets unless we train the networks with learning rate warmup.
- Winning tickets reveal **combinations of sparse architectures and initializations**.
  - For vgg-19, networks **pruned by up to 80% and randomly reinitialized match the accuracy of the original network.**
  - However, after **further pruning, initialization matters**:

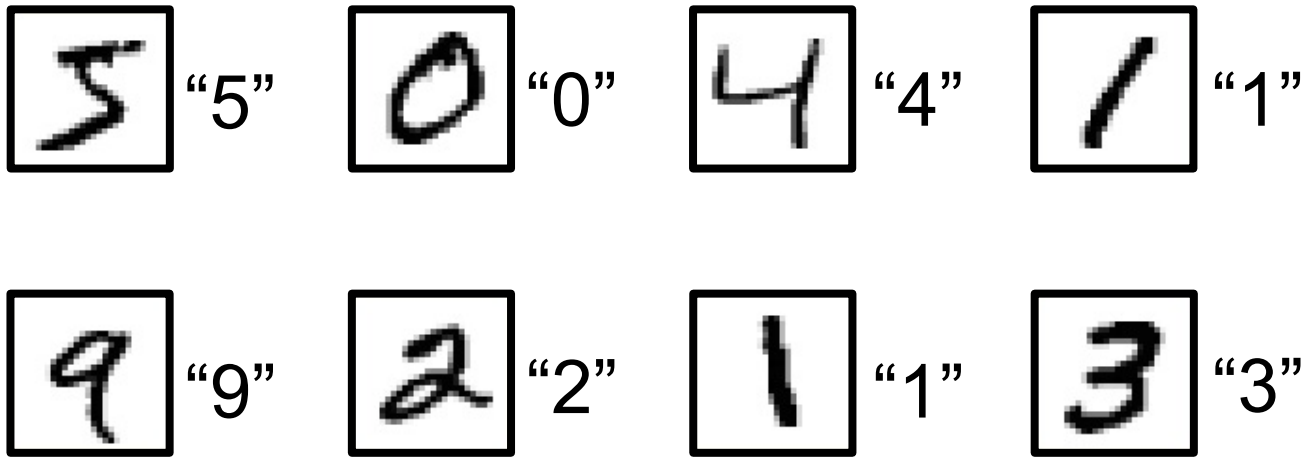
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks-ICLR 2019 best paper

Three Steps for Deep Learning



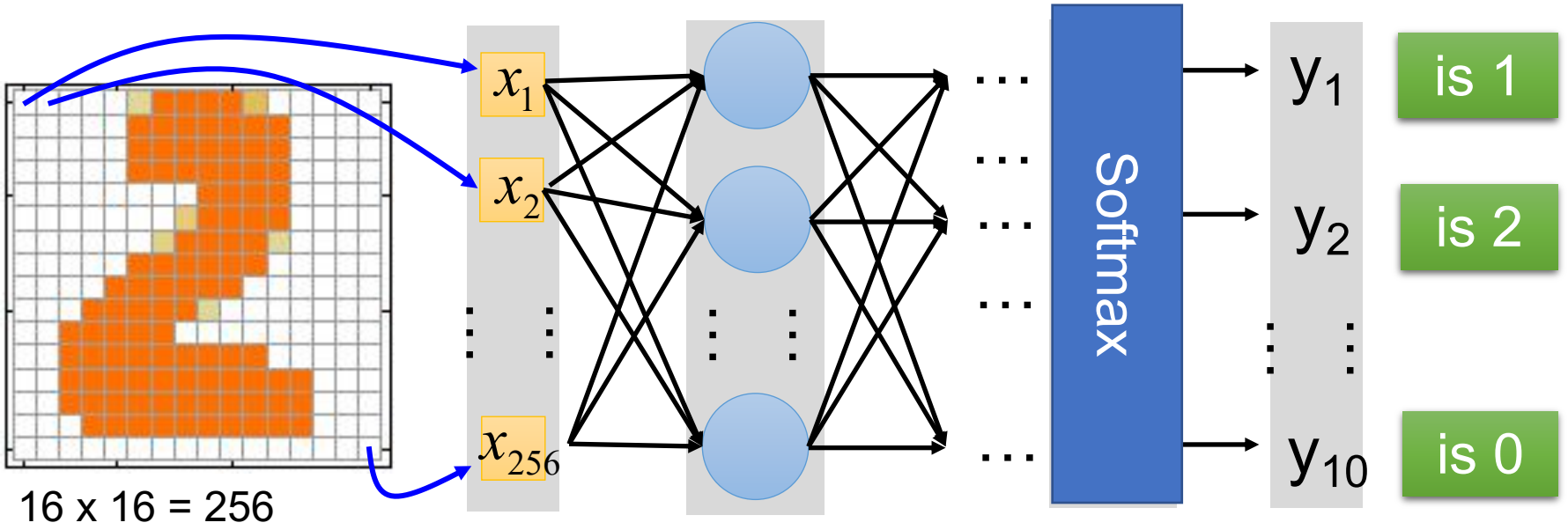
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

Learning Target



16 x 16 = 256

Ink → 1

No ink → 0

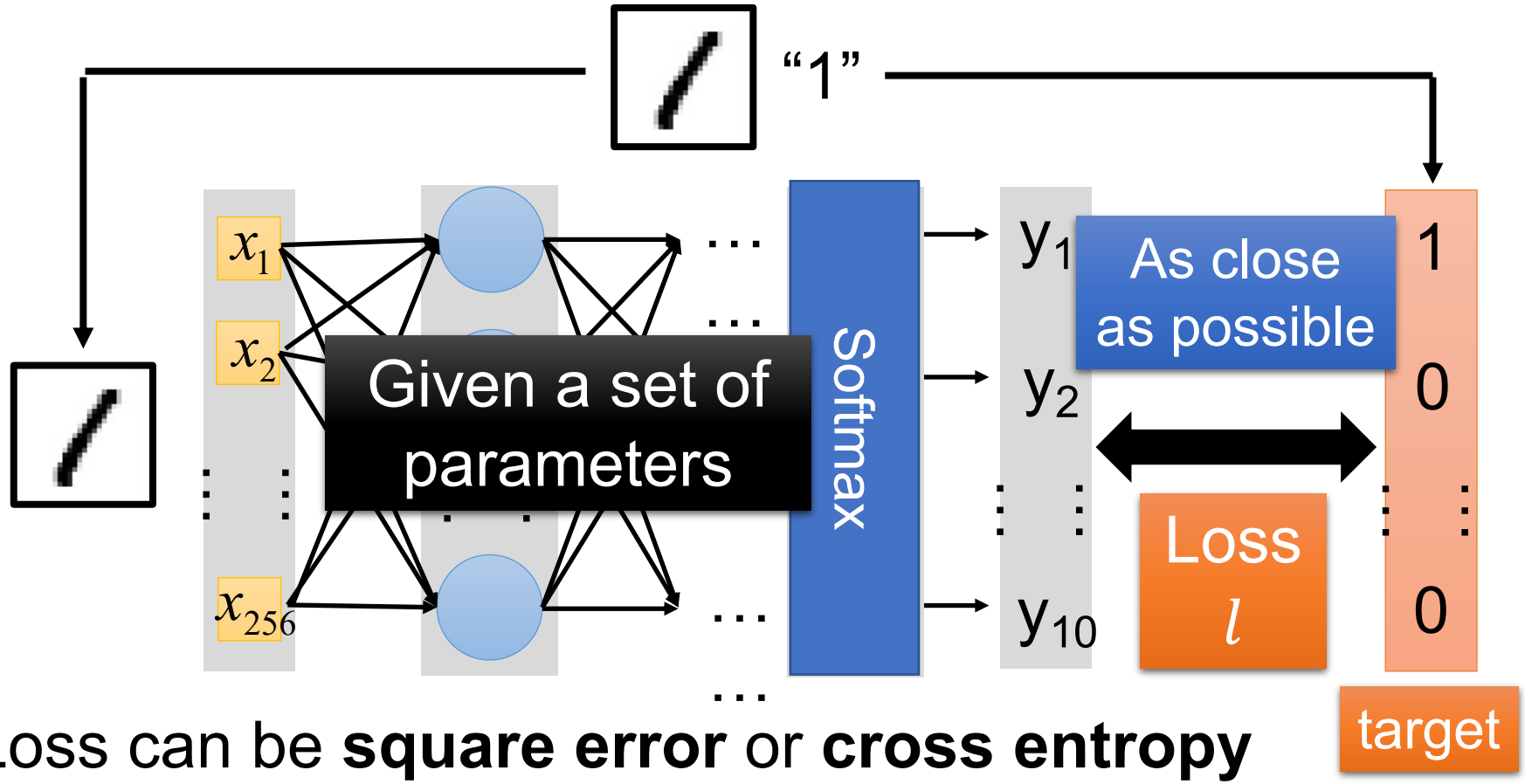
The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

Loss

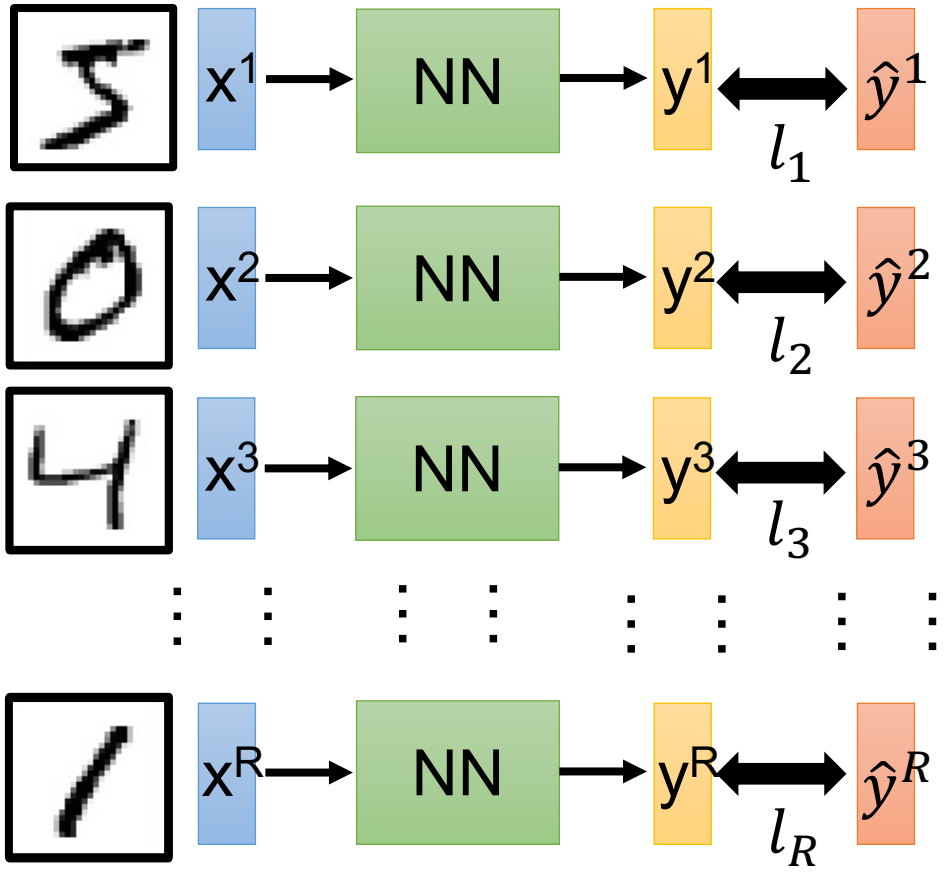
A good function should make the loss of all examples as small as possible.



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

Total Loss

For all training data …



Total Loss:

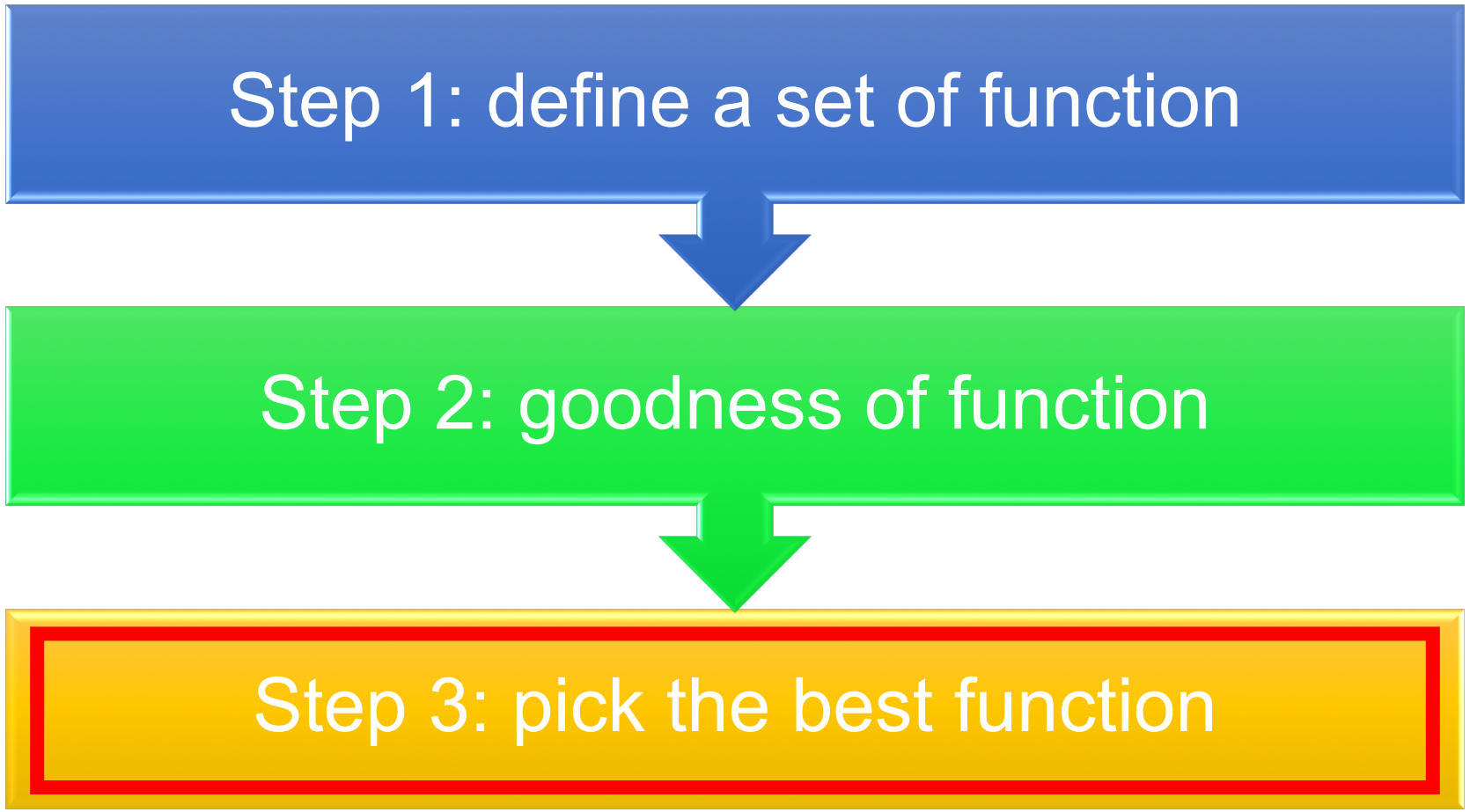
$$L = \sum_{r=1}^{R} l_r$$

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

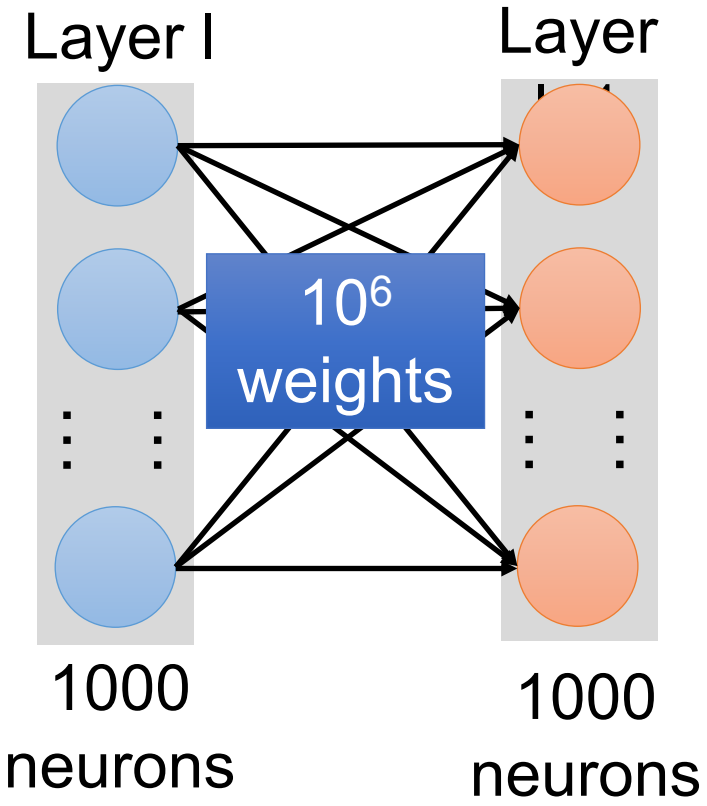
Find ***network parameters*** θ^* that minimize total loss L

Enumerate all possible values

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

Millions of parameters

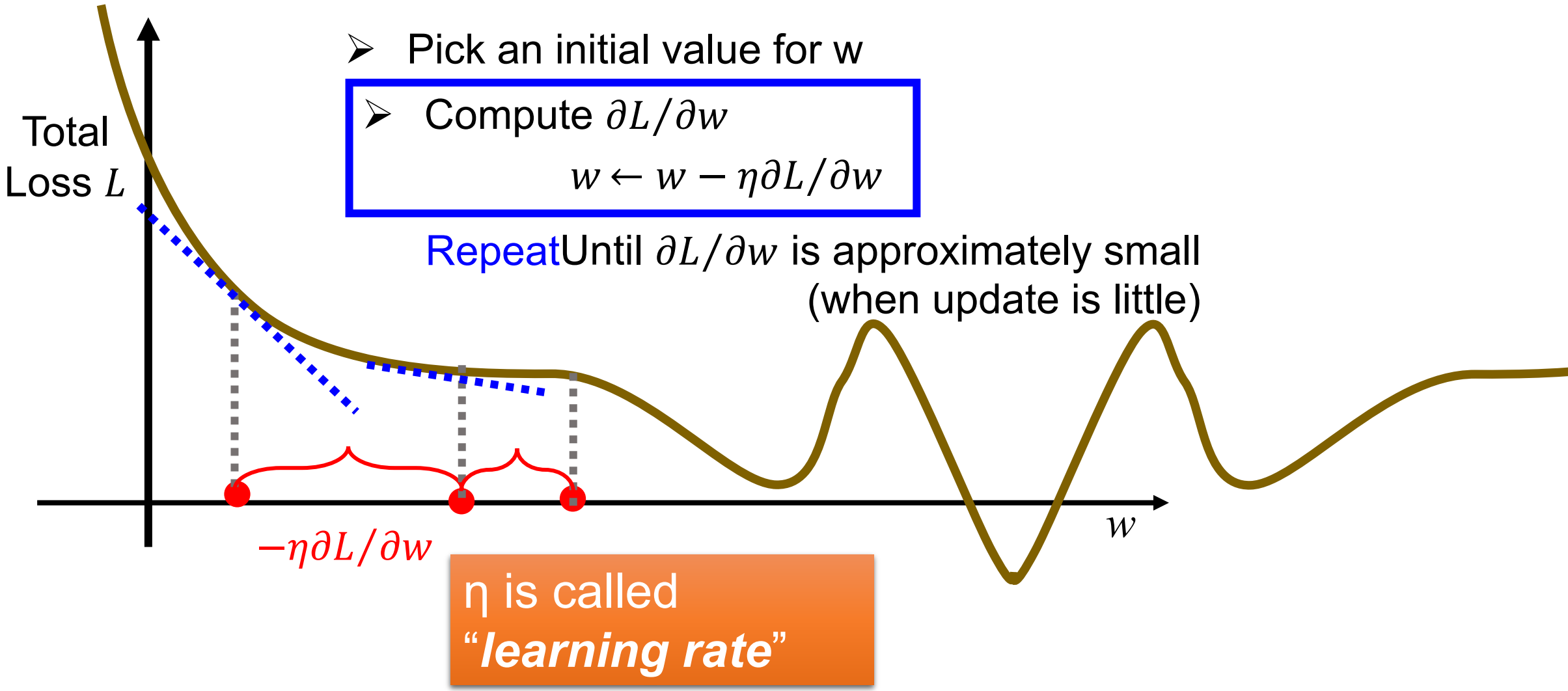
E.g. speech recognition: 8 layers and 1000 neurons each layer



Gradient Descent

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

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



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

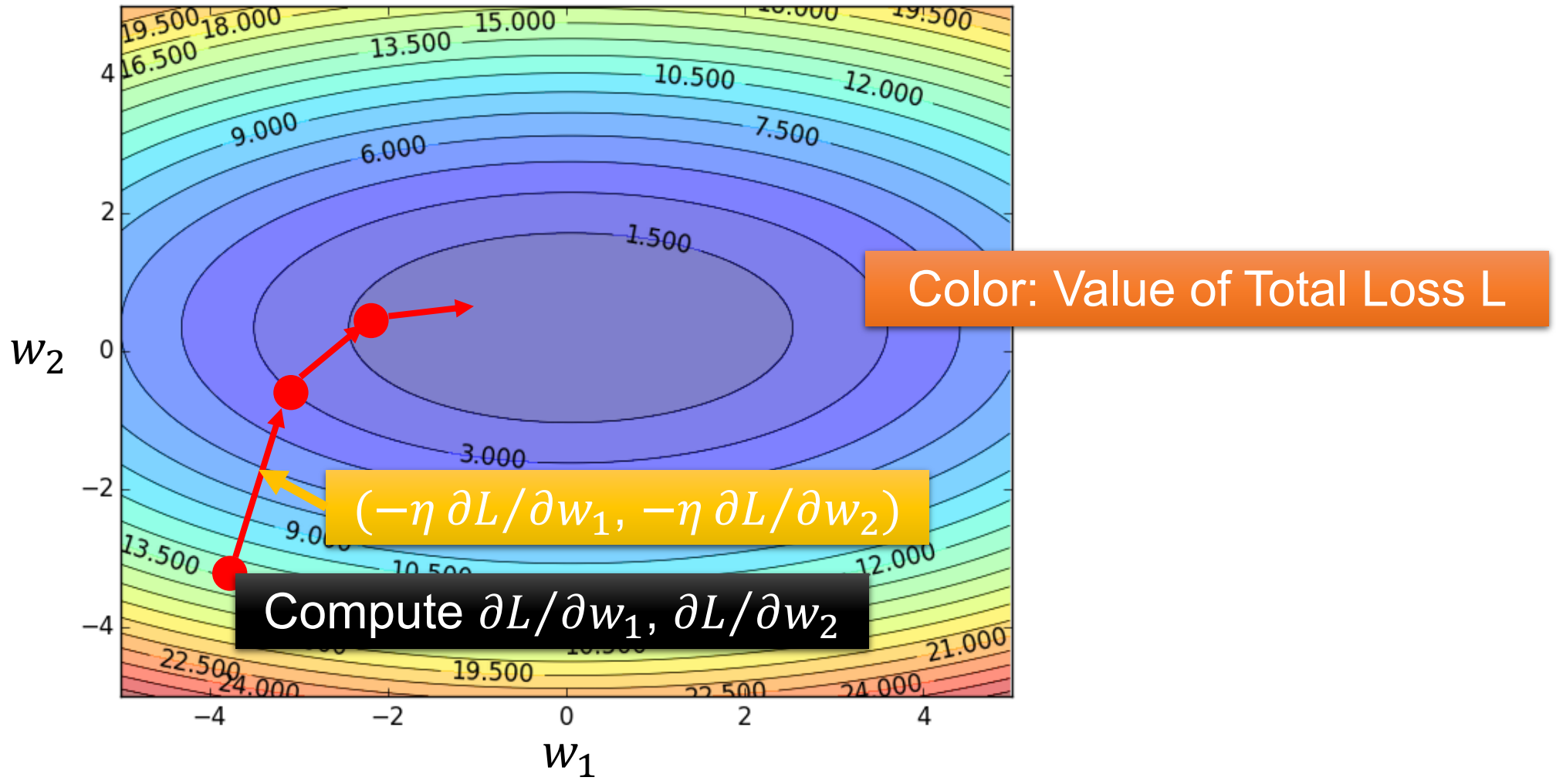
$$w \leftarrow w - \eta\partial L/\partial w$$

RepeatUntil $\partial L/\partial w$ is approximately small (when update is little)

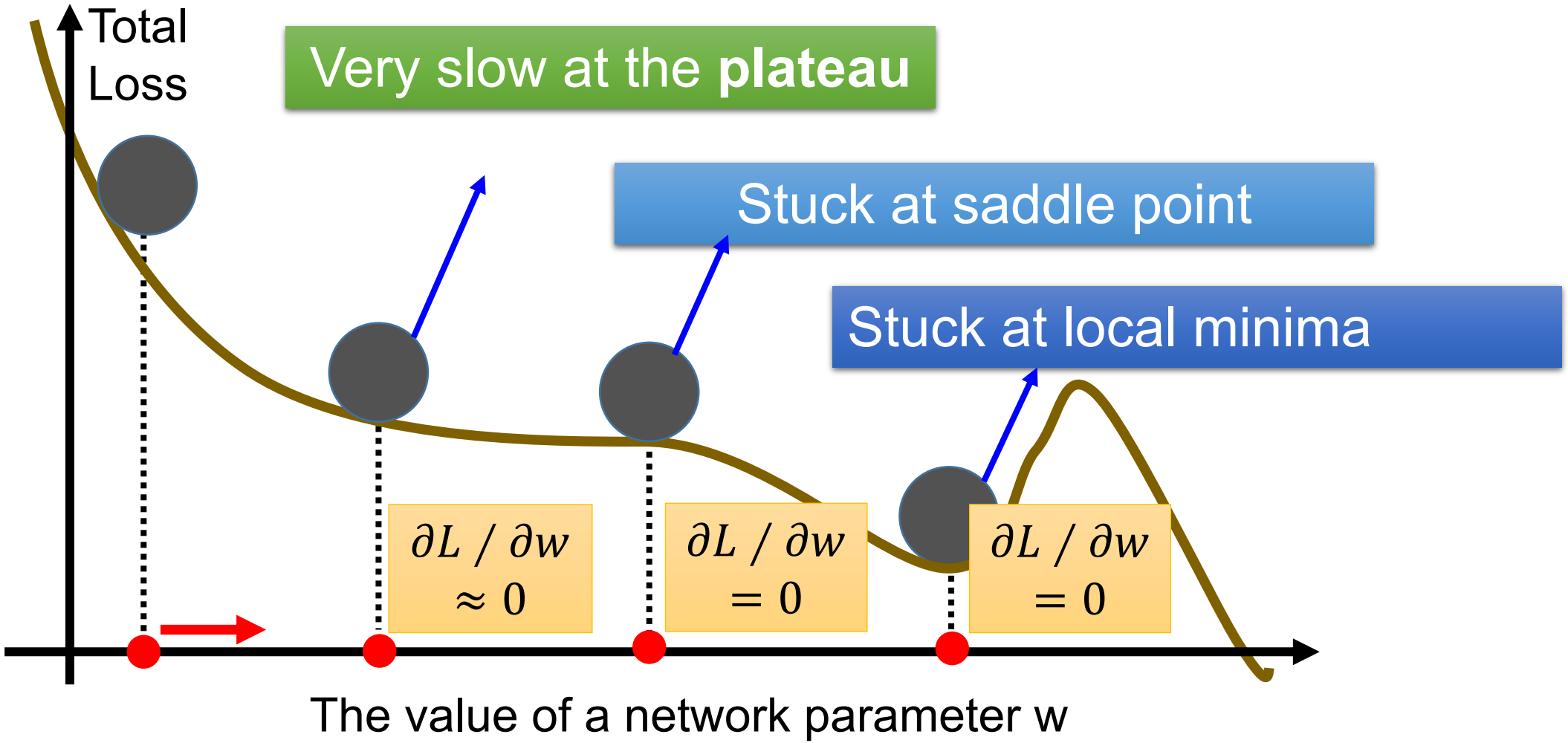
η is called "***learning rate***"

Gradient Descent

Hopfully, we would reach a minima …..

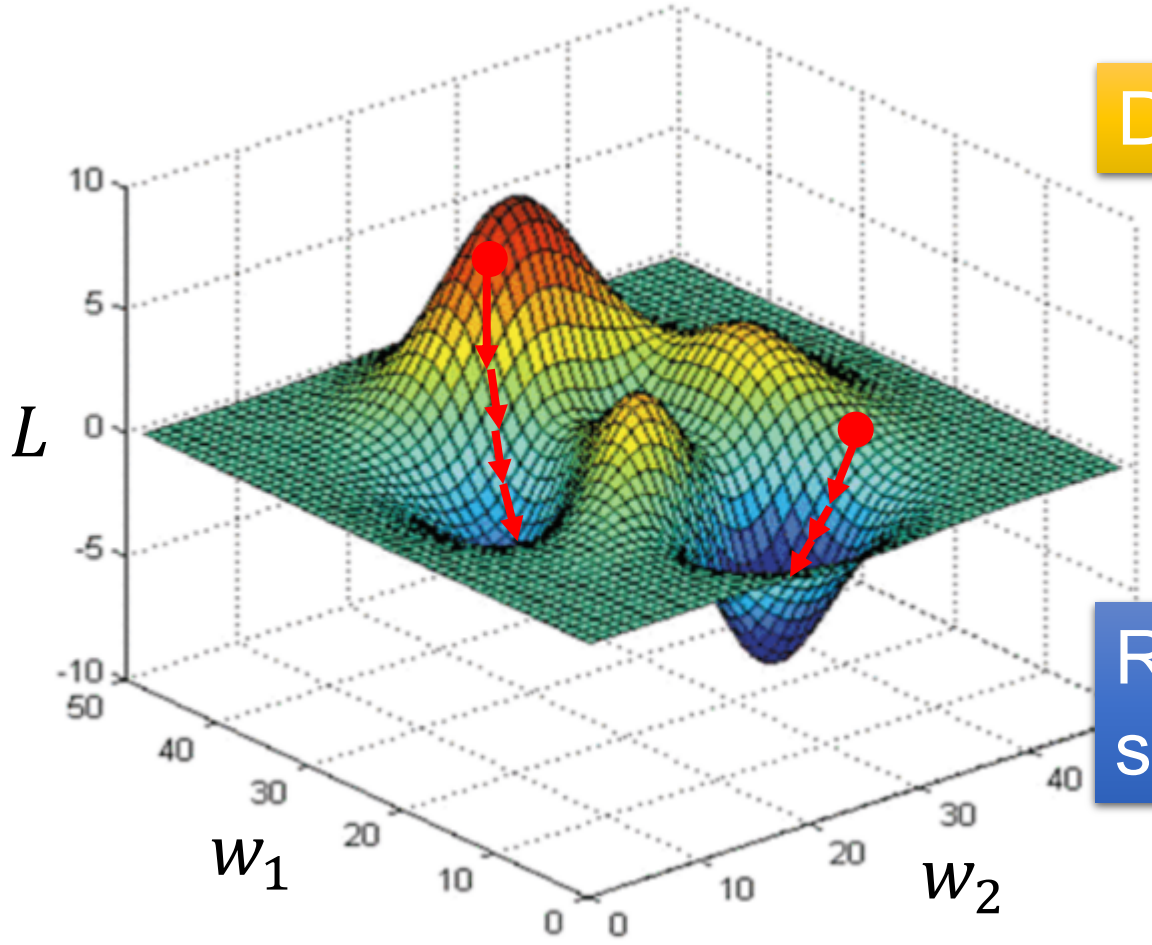


Local Minima



Local Minima

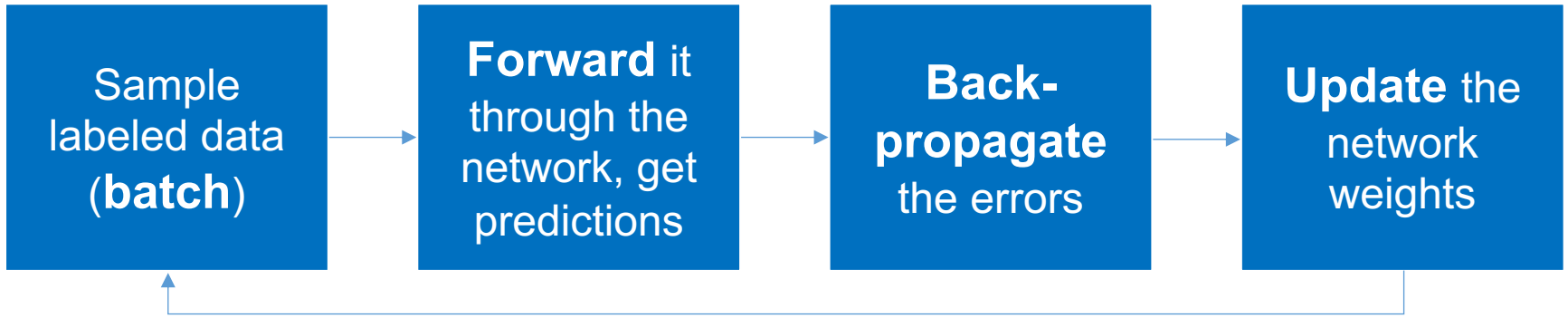
- Gradient descent never guarantee global minima

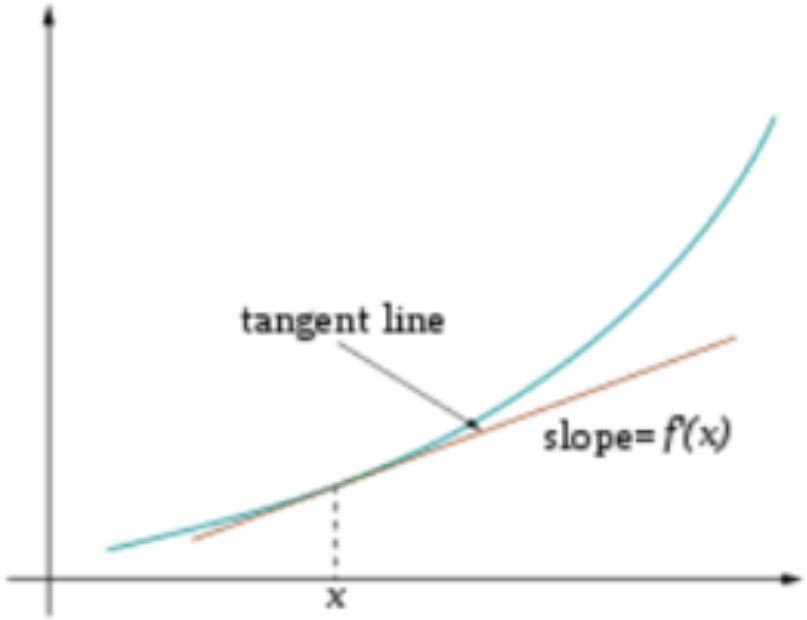


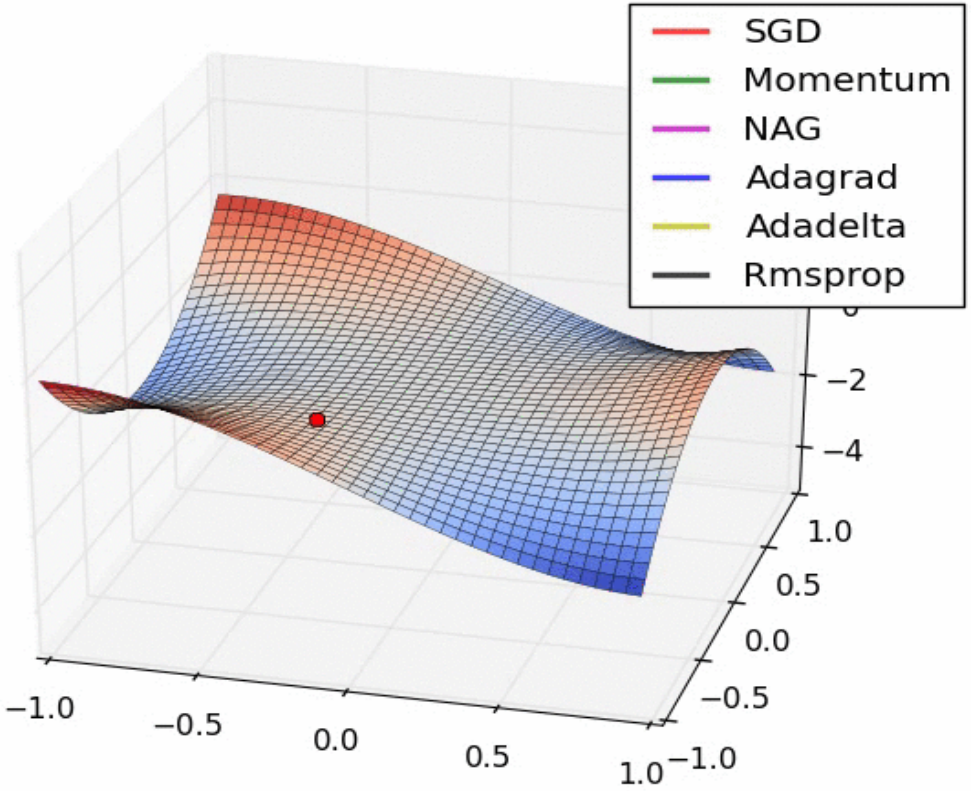
Different initial point

Reach different minima, so different results

Training


Sample labeled data (**batch**)
Forward it through the network, get predictions
Back-propagate the errors
Update the network weights


tangent line
slope= f'(x)
x

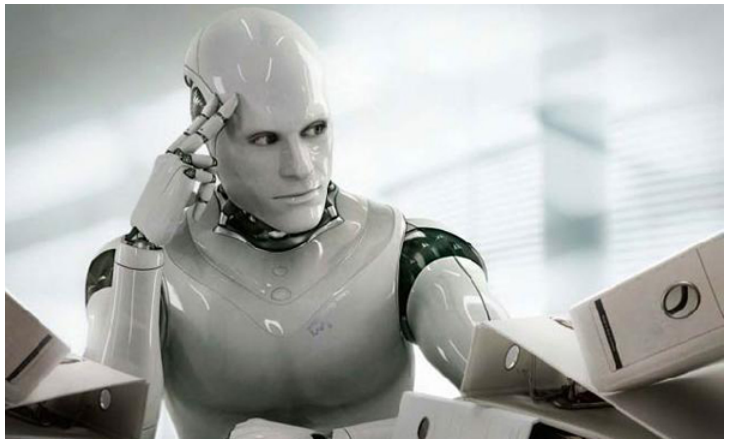

SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
−1.0
−0.5
0.0
0.5
1.0
−2
−4
1.0
0.5
0.0
−0.5
−1.0

Gradient Descent

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

➡ Even alpha go using this approach.

People image ……

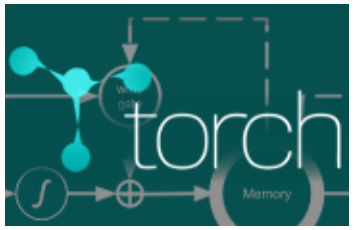


Actually …..

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



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

Three Steps for Deep Learning

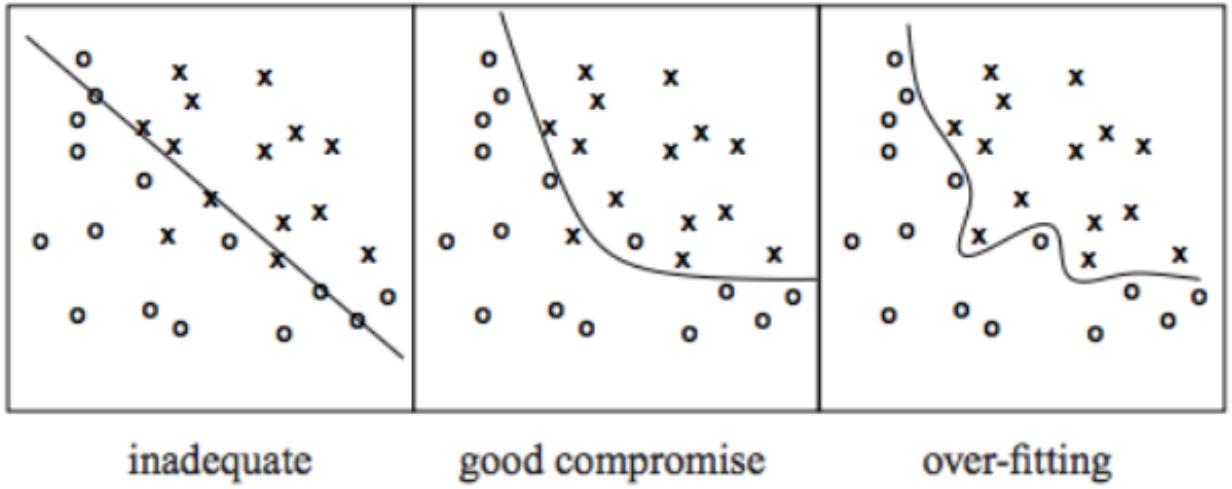
Deep Learning is so simple ……

Now If you want to find a function

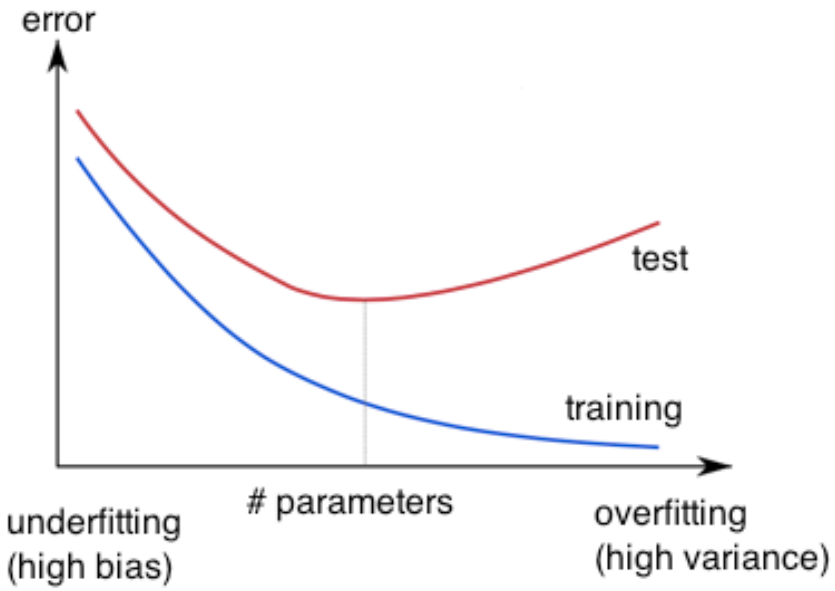
If you have lots of function input/output (?) as training data

➡ You can use deep learning

Overfitting



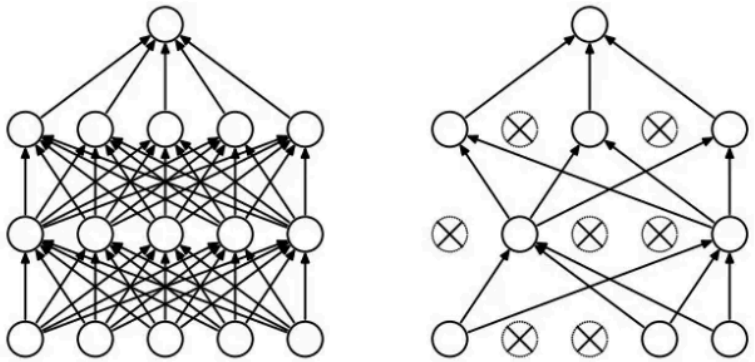
http://wiki.bethanycrane.com/overfitting-of-data



https://www.neuraldesigner.com/images/learning/selection_error.svg

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

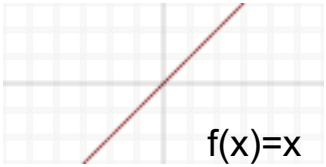
- Regularization term that penalizes big weights, added to the objective
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

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

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

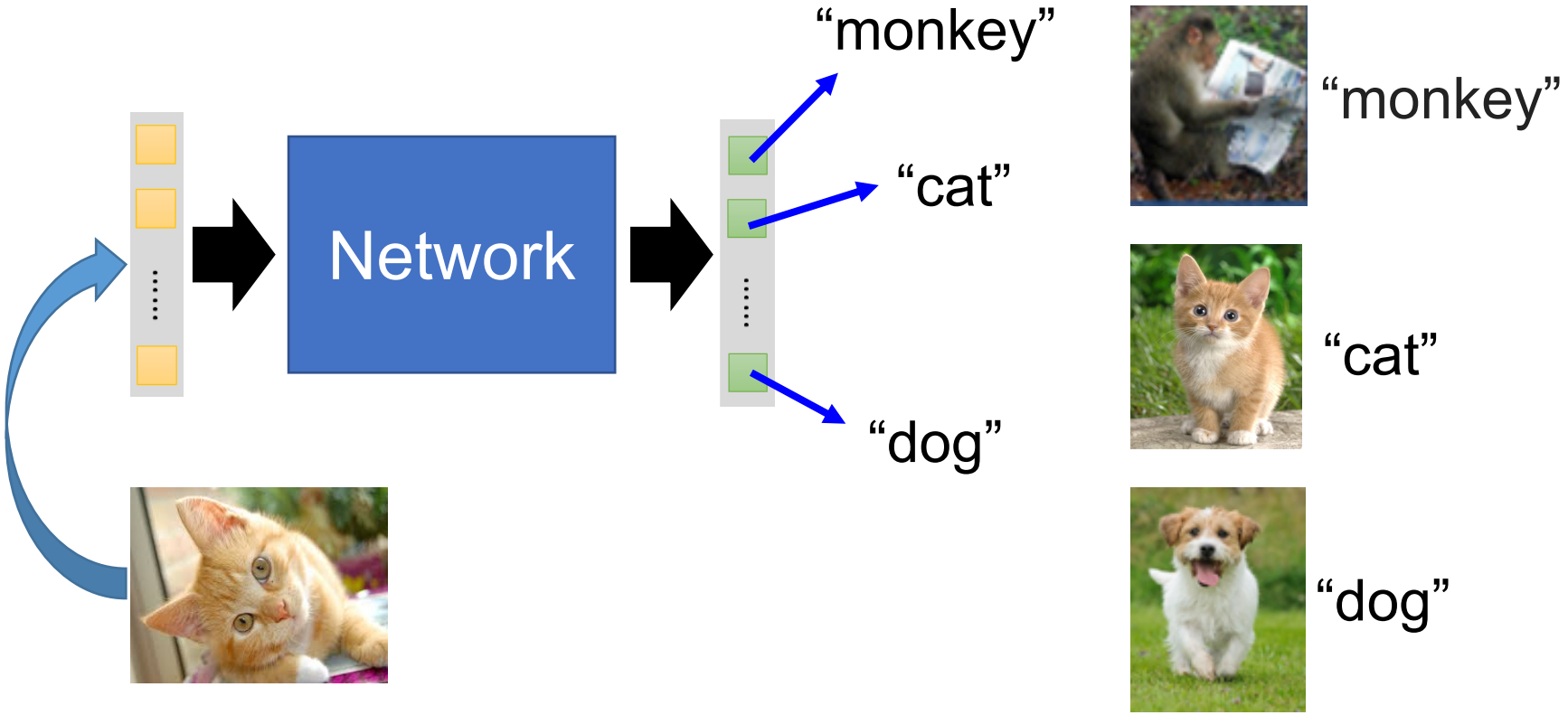
Loss functions and output

	Classification	Regression
Training examples	R^n x {class_1, ..., class_n} (one-hot encoding)	R^n x R^m
Output Layer	Soft-max [map R^n to a probability distribution] $P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$	Linear (Identity) or Sigmoid 
Cost (loss) function	Cross-entropy $J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\left[y_k^{(i)} \log \hat{y}_k^{(i)} + \left(1 - y_k^{(i)}\right) \log\left(1 - \hat{y}_k^{(i)}\right)\right]$	Mean Squared Error $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$ Mean Absolute Error $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left\|y^{(i)} - \hat{y}^{(i)}\right\|$

List of loss functions

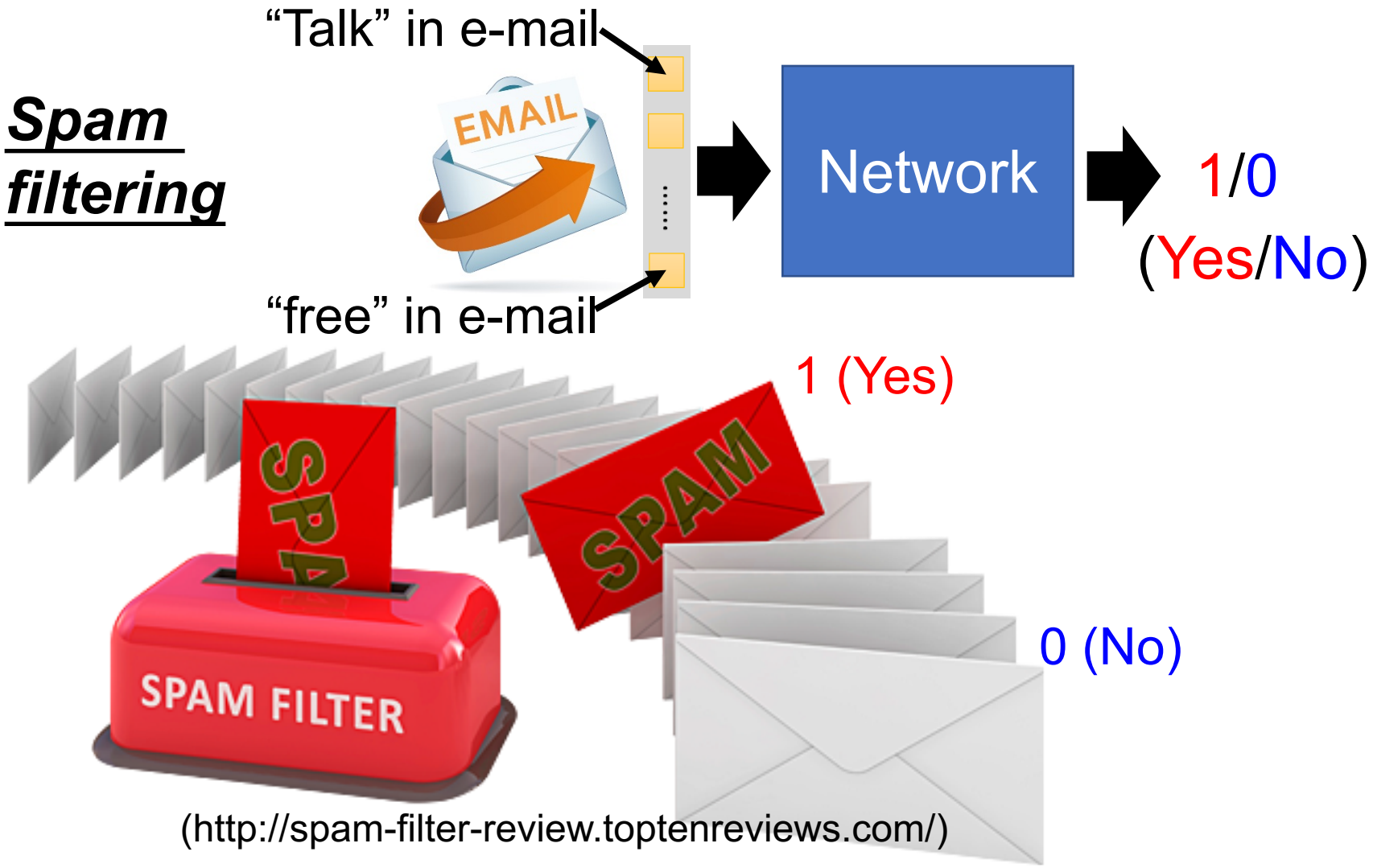
For example, you can do …….

- Image Recognition



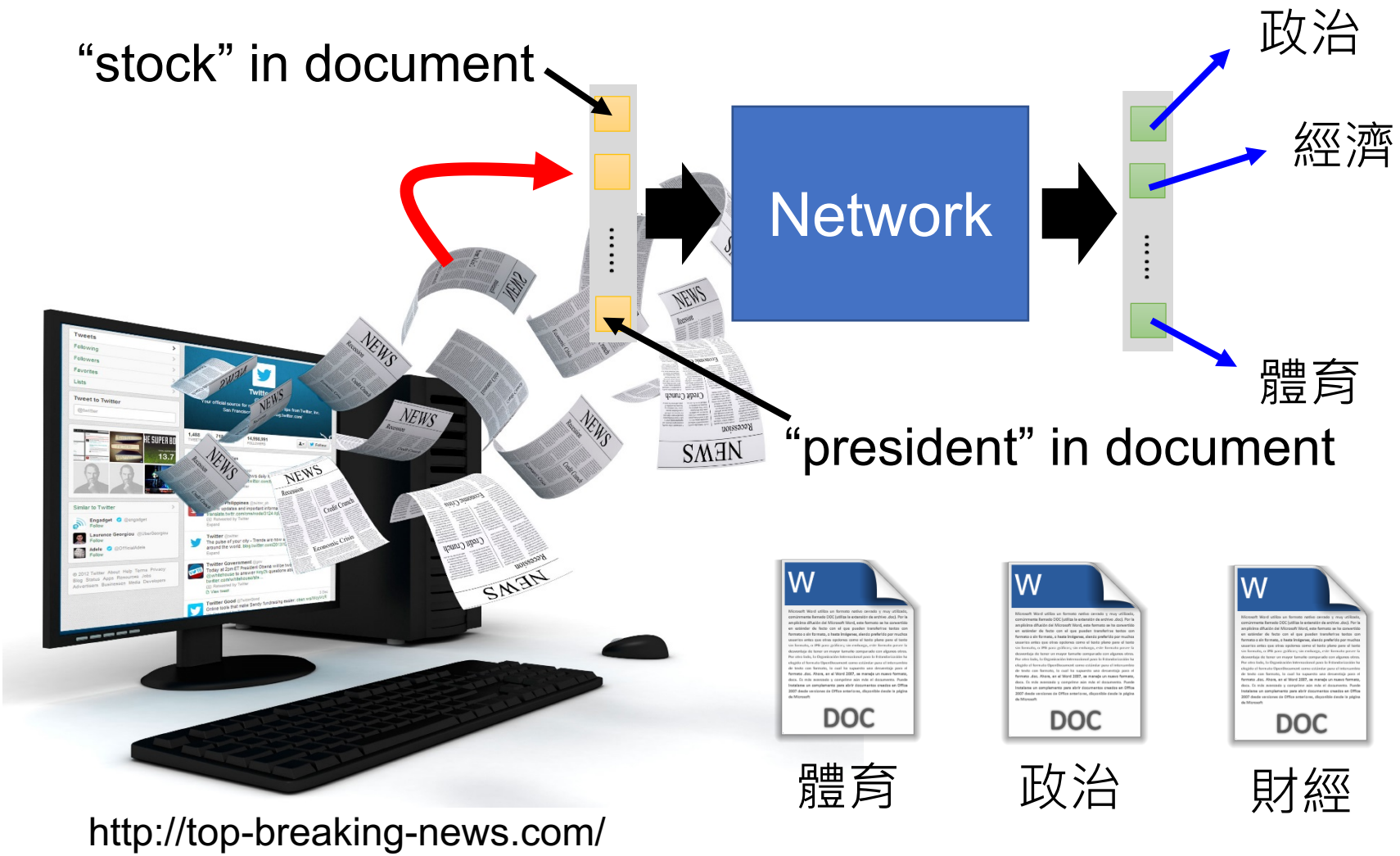
For example, you can do …….

Spam filtering

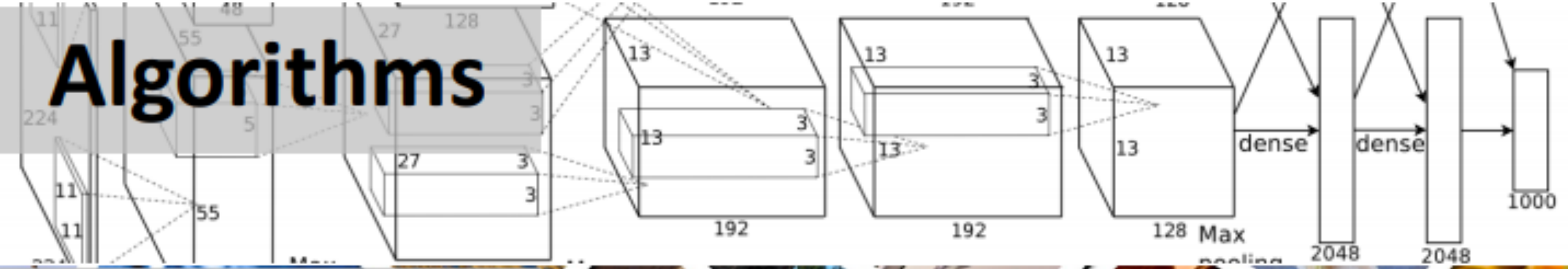


(http://spam-filter-review.toptenreviews.com/)

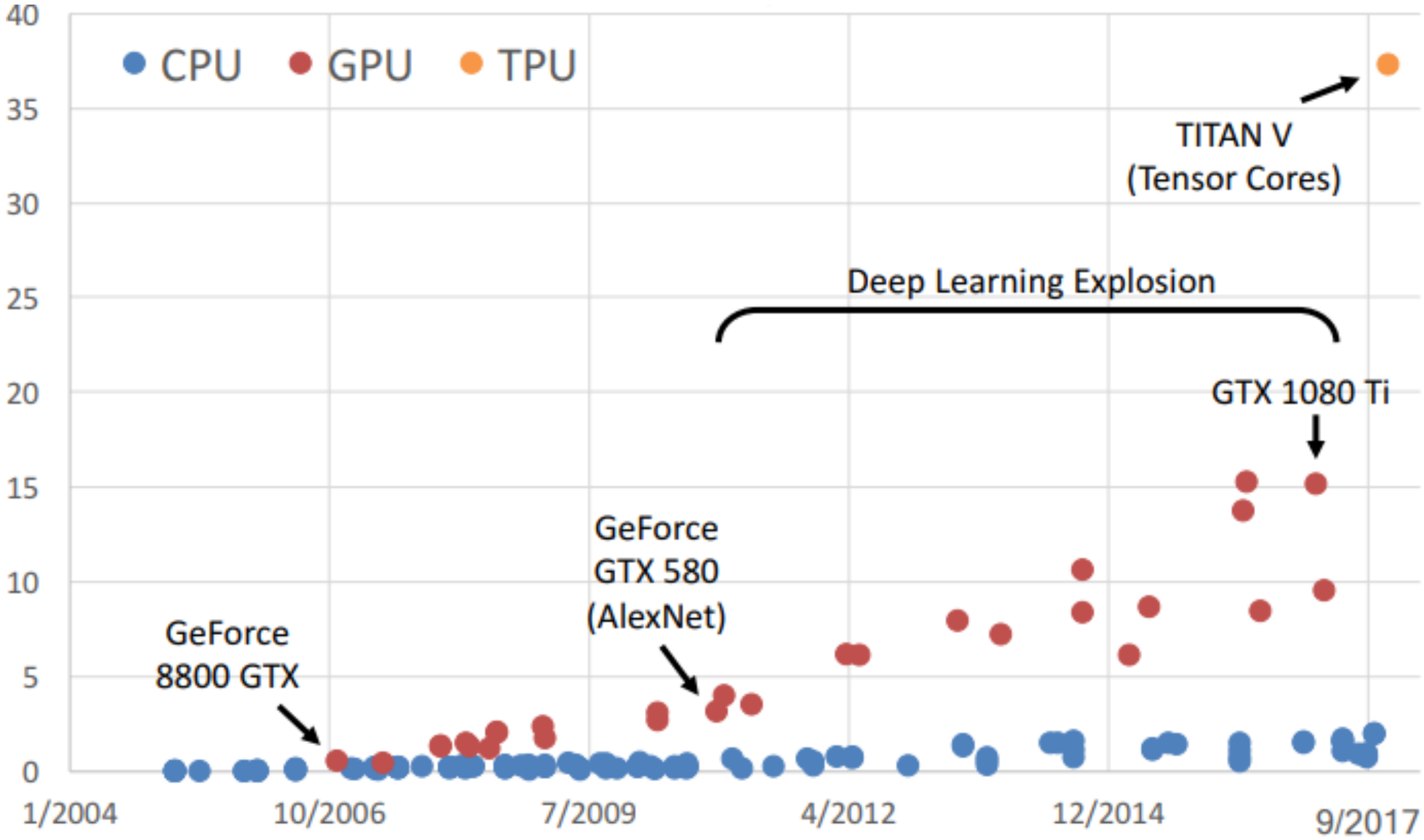
For example, you can do .....



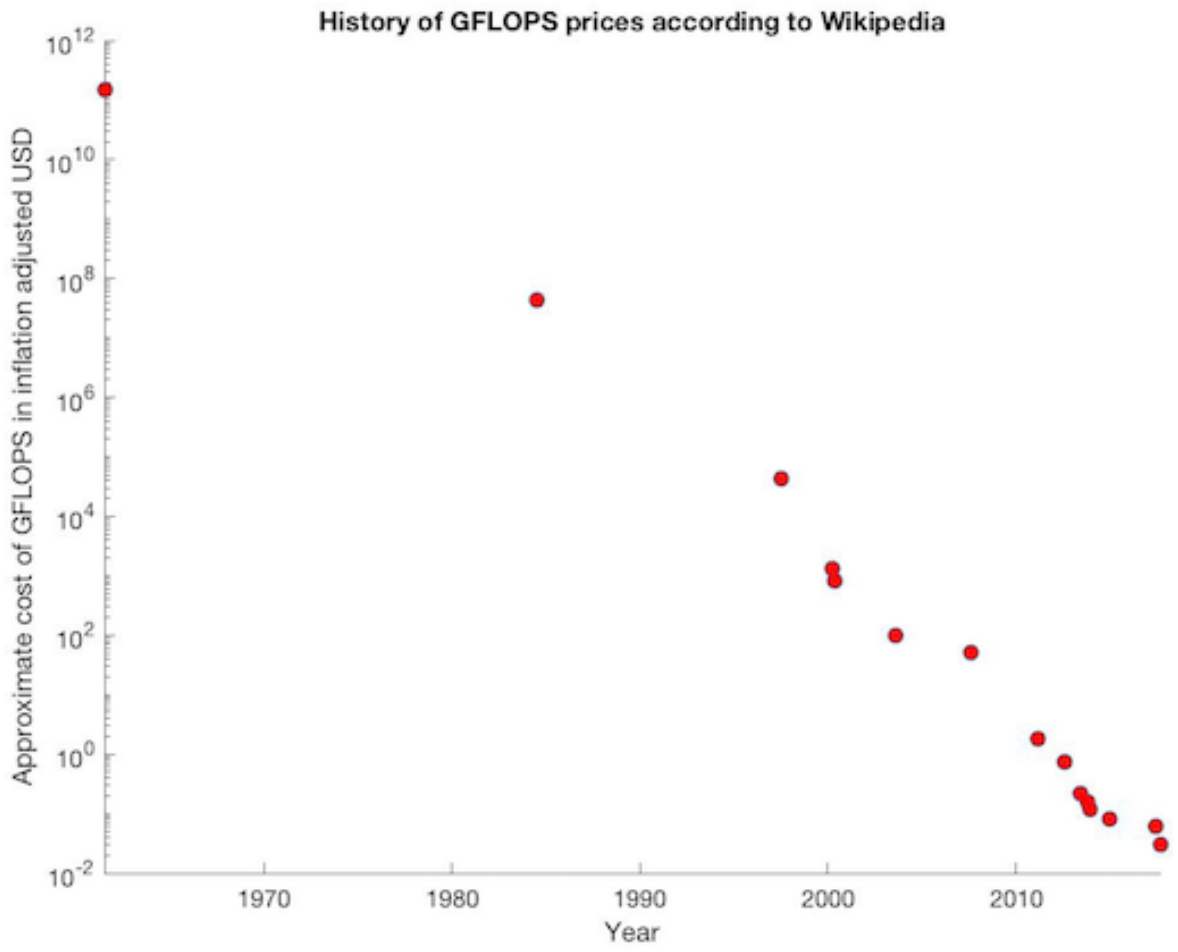
Ingredients for Deep Learning



GigaFLOPs (1*10^9) per Dollar



History of GFLOPS costs

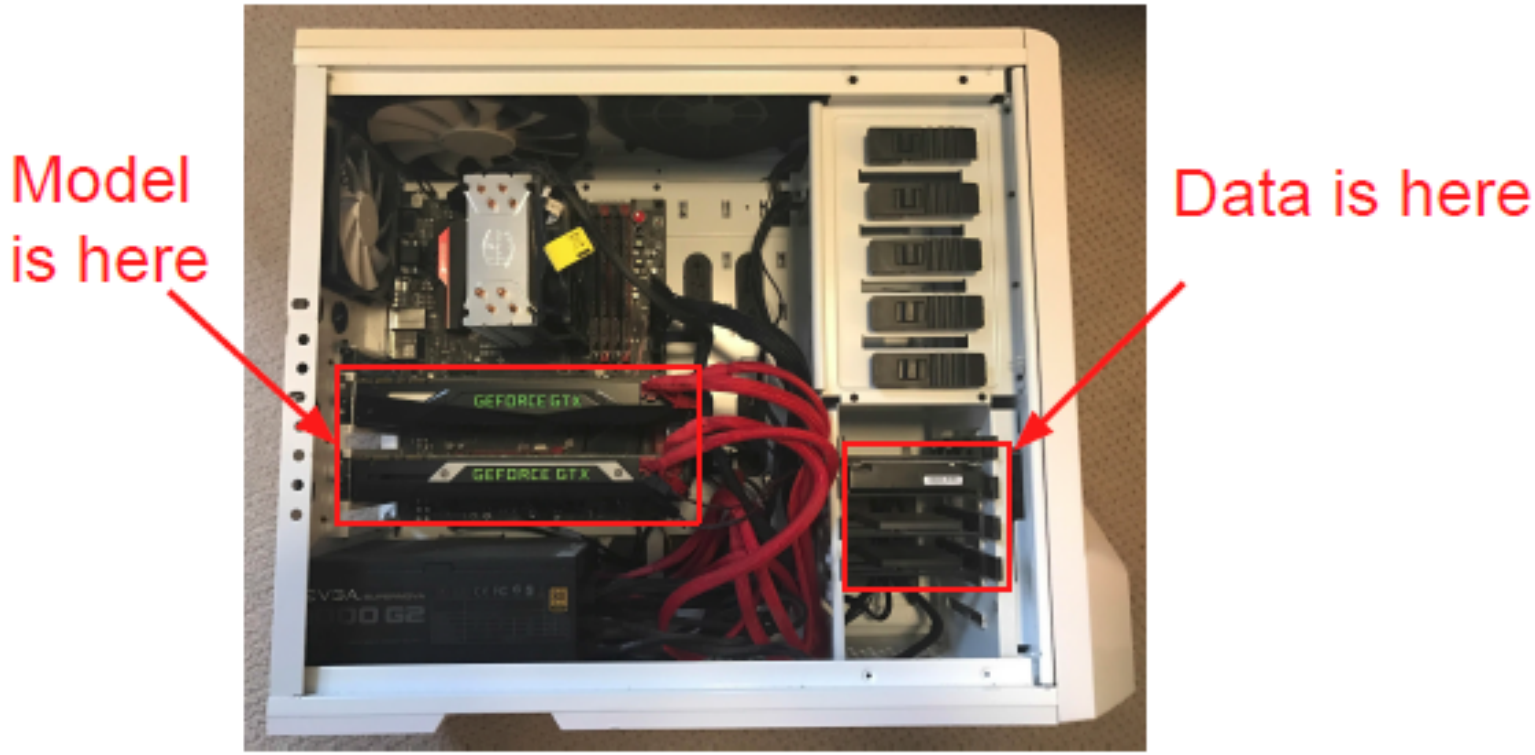


- 1961, US$145.5 billion
- 2017, $0.03

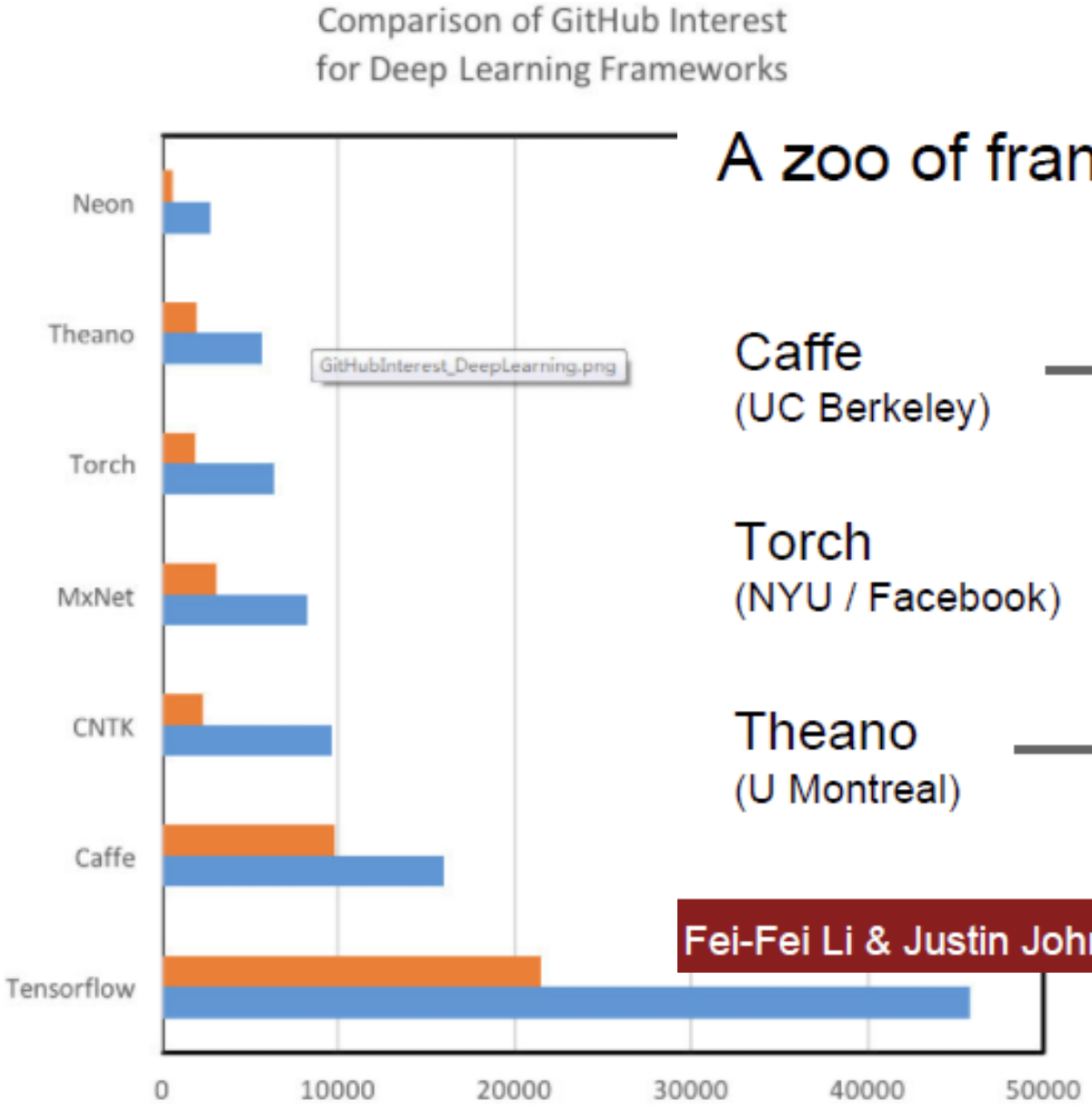
- The overall rate in the figure appears to be very roughly an **order of magnitude every five years**.

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



A zoo of frameworks!

Caffe (UC Berkeley) → Caffe2 (Facebook)

Torch (NYU / Facebook) → PyTorch (Facebook)

Theano (U Montreal) → TensorFlow (Google)

We'll focus on these

PaddlePaddle (Baidu)

Chainer

MXNet (Amazon)

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

CNTK (Microsoft)

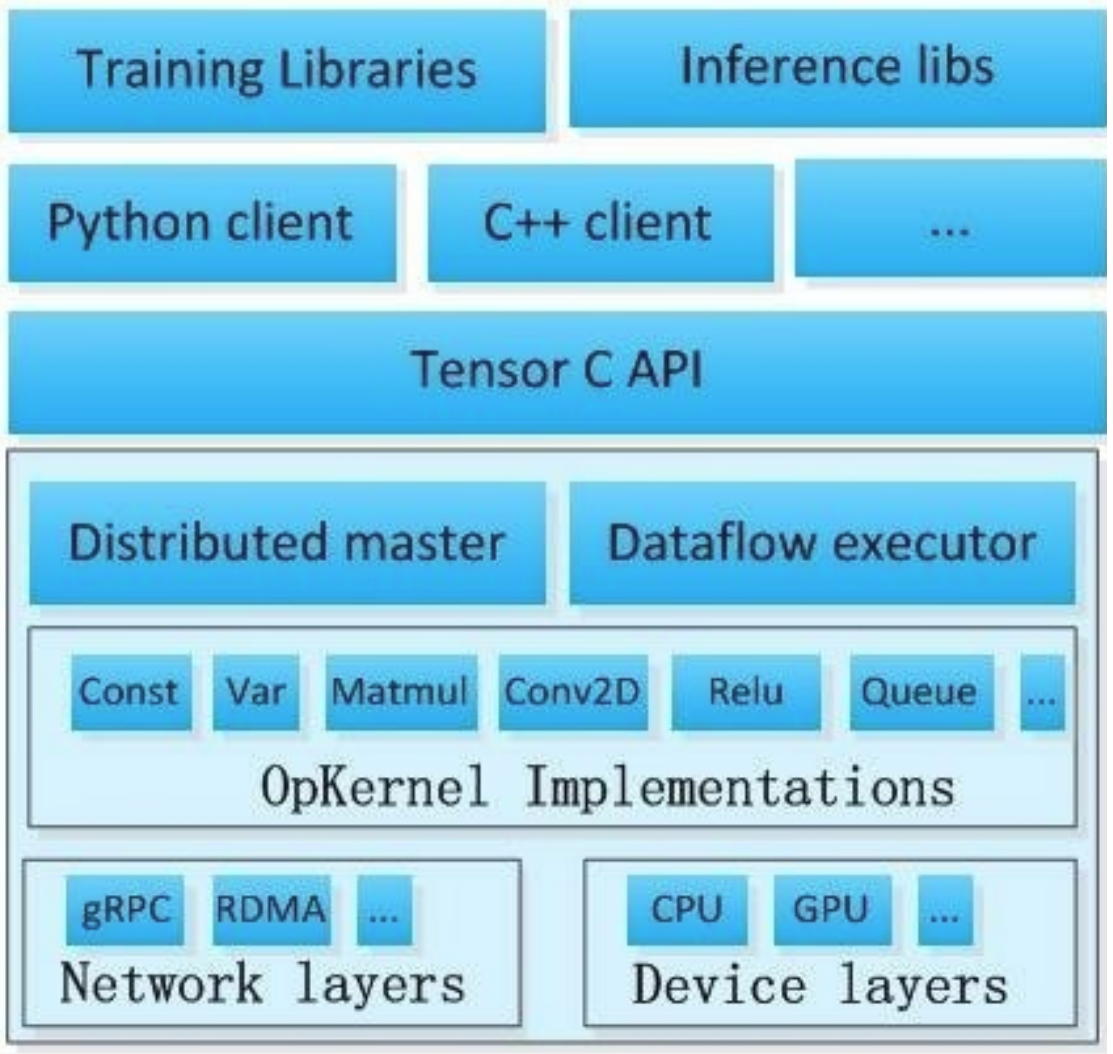
Deeplearning4j

And others...

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



Architecture of TensorFlow

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。

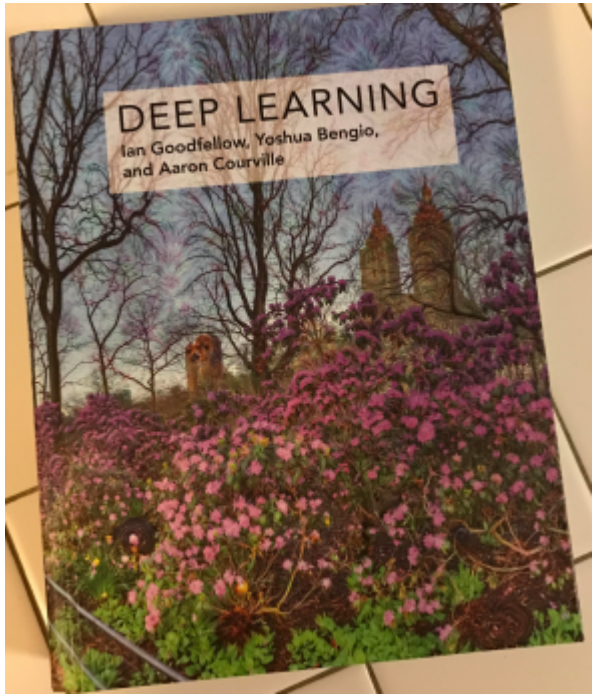
  -- https://www.jiqizhixin.com/articles/2017-06-13-4

	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	**	**

- 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**。

  - --CMU LTI博士研究生王赟

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