

1



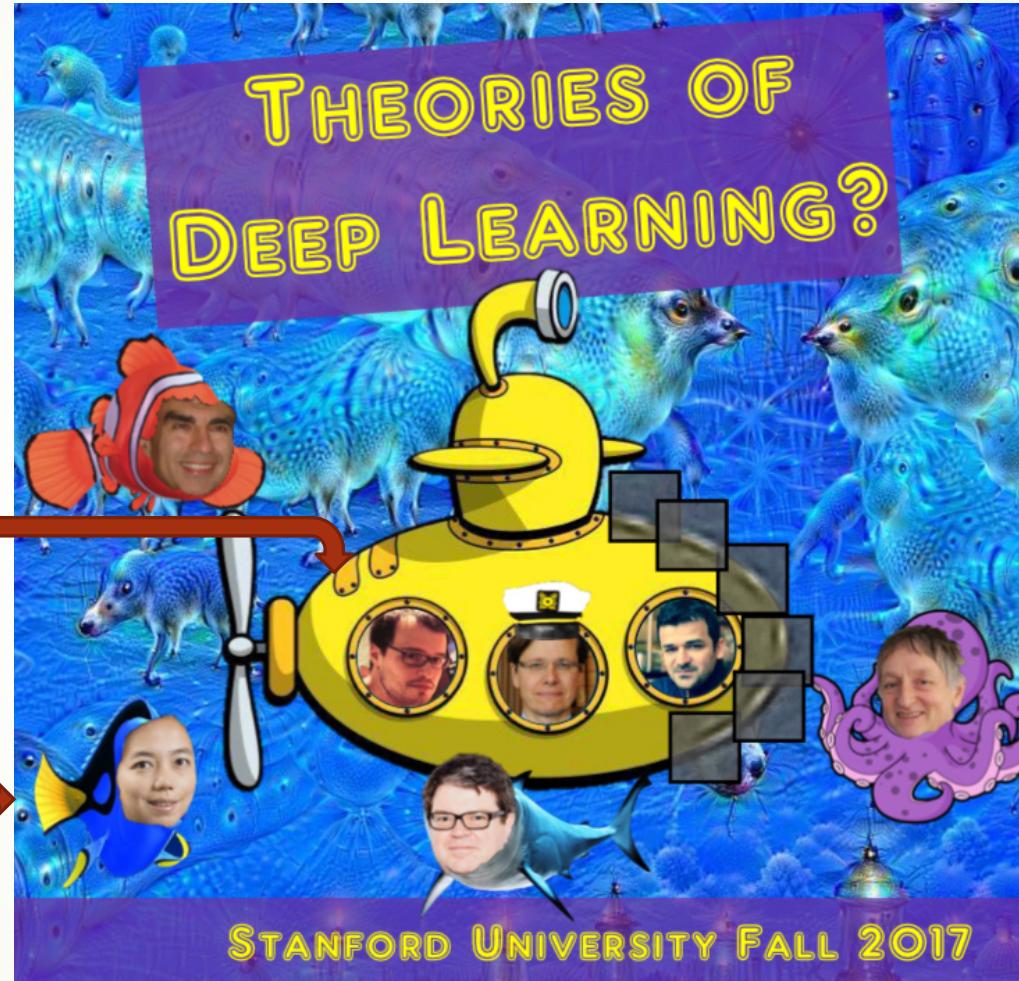
# Advanced Topics in Deep Learning

Yuan YAO  
HKUST

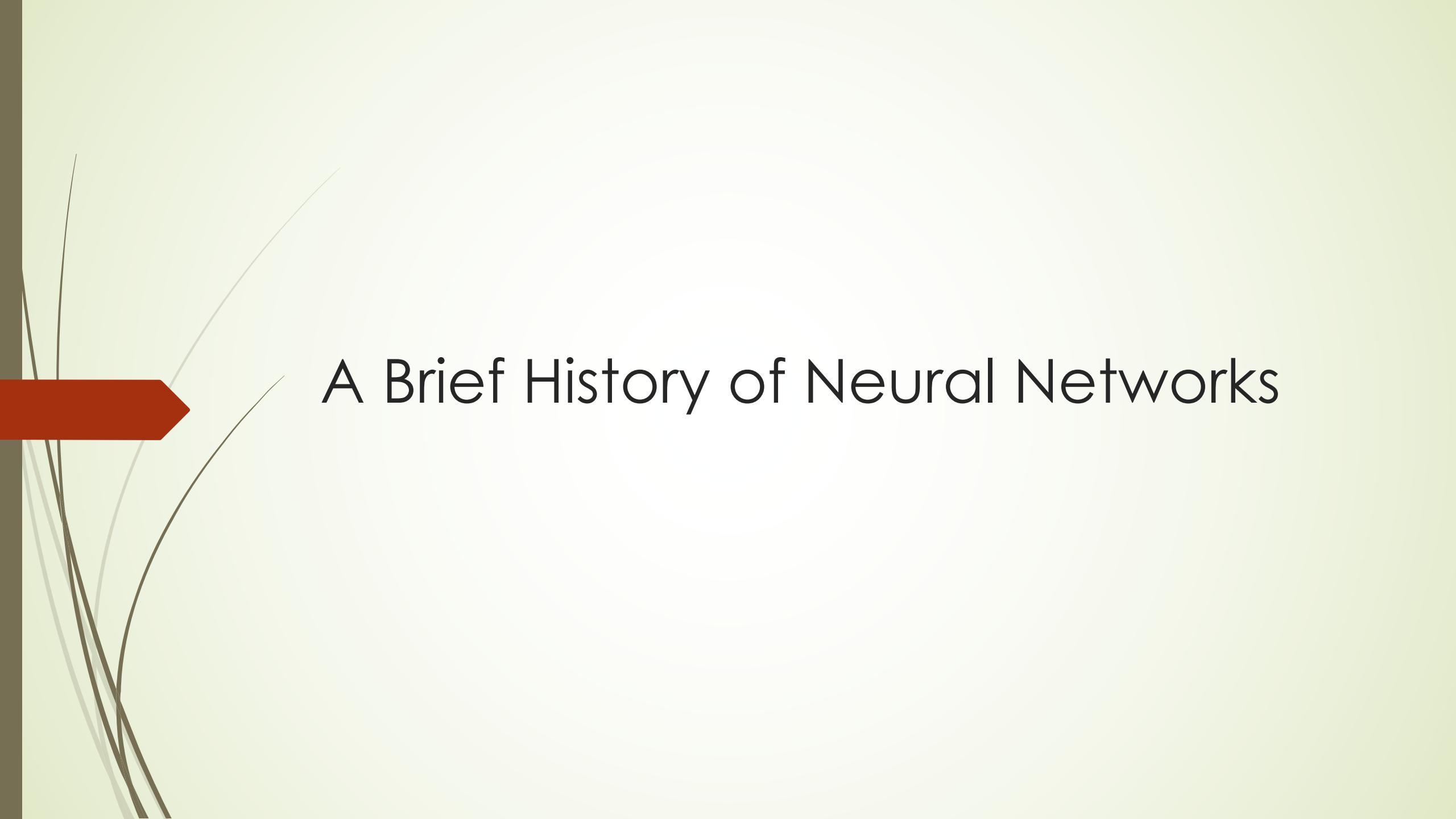
# Acknowledgement

<https://stats385.github.io/>

<http://cs231n.github.io/>



A following-up course at HKUST: <https://deeplearning-math.github.io/>

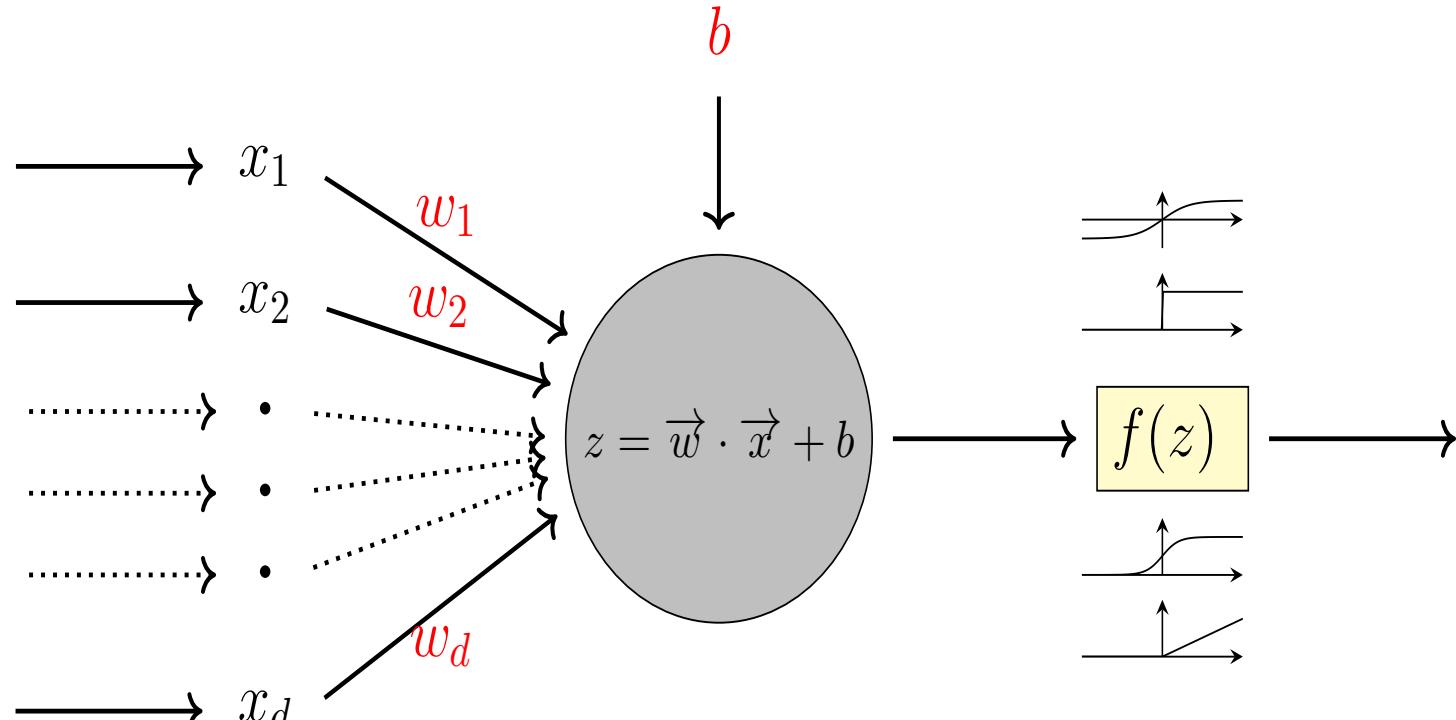


# A Brief History of Neural Networks

# Perceptron: single-layer



- Invented by Frank Rosenblatt (1957)



# Stochastic Approximation

- ▶ **Robbins-Monro, 1951, Ann. Math. Statist. 22(3):400-407**

$$M(x) = \mathbb{E}_\xi A(x; \xi) = b$$

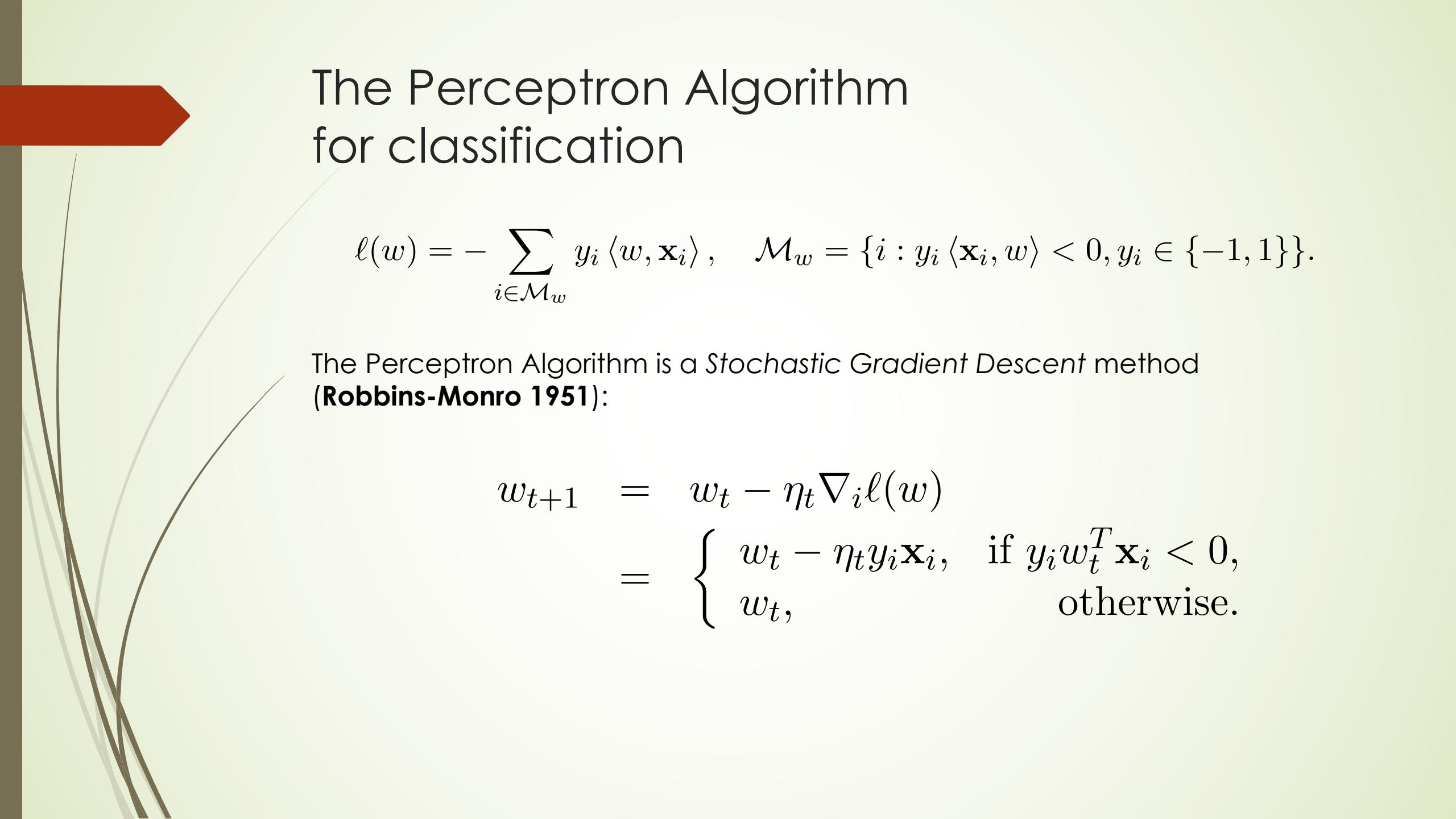
where  $M(x)$  is monotone, stochastic approximation:

$$x_{t+1} = x_t - \eta_t(A(x_t; \xi_t) - b) \quad (1)$$

- ▶ **Kiefer-Wolfowitz, 1952, Ann. Math. Statist. 23(3):462-466**

$$\min_x \mathbb{E}_\xi \ell(x; \xi)$$

$$x_{t+1} = x_t - \eta_t \nabla_x \ell(x_t; \xi_t) \quad (2)$$



# The Perceptron Algorithm for classification

$$\ell(w) = - \sum_{i \in \mathcal{M}_w} y_i \langle w, \mathbf{x}_i \rangle, \quad \mathcal{M}_w = \{i : y_i \langle \mathbf{x}_i, w \rangle < 0, y_i \in \{-1, 1\}\}.$$

The Perceptron Algorithm is a *Stochastic Gradient Descent* method  
**(Robbins-Monro 1951)**:

$$\begin{aligned} w_{t+1} &= w_t - \eta_t \nabla_i \ell(w) \\ &= \begin{cases} w_t - \eta_t y_i \mathbf{x}_i, & \text{if } y_i w_t^T \mathbf{x}_i < 0, \\ w_t, & \text{otherwise.} \end{cases} \end{aligned}$$

# Finiteness of Stopping Time and Margin

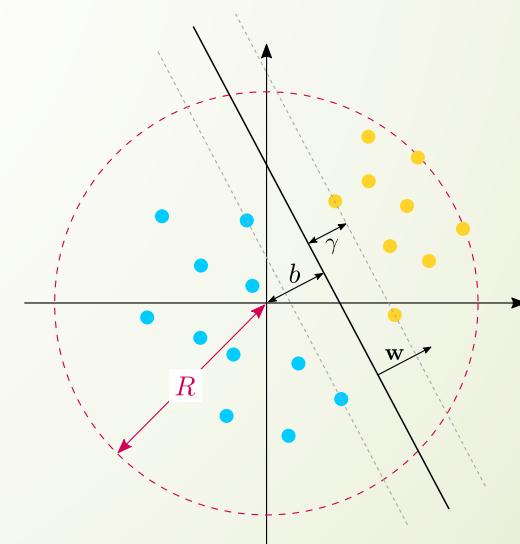
The perceptron convergence theorem was proved by [Block \(1962\)](#) and [Novikoff \(1962\)](#). The following version is based on that in [Cristianini and Shawe-Taylor \(2000\)](#).

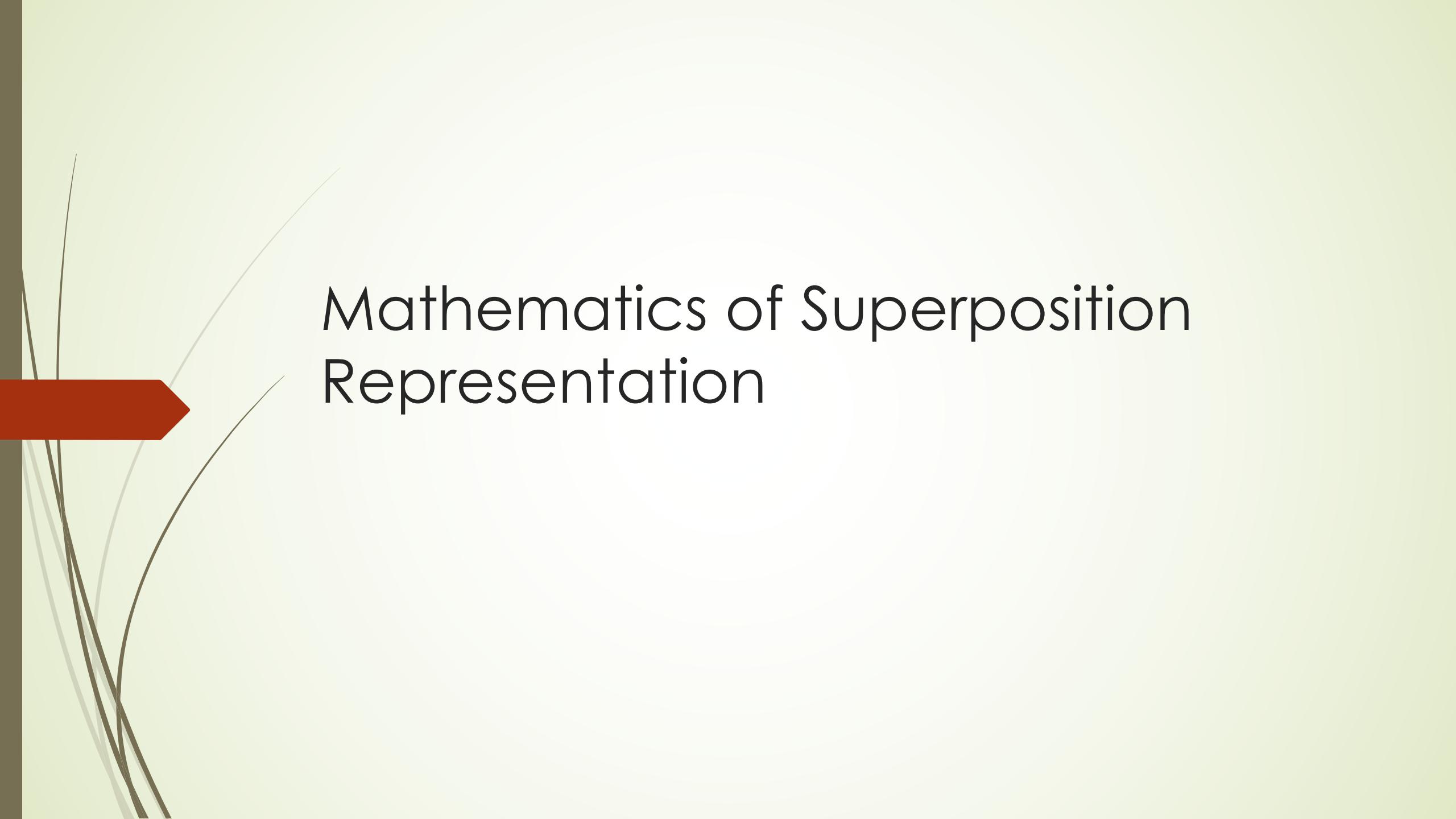
**Theorem 1** (Block, Novikoff). *Let the training set  $S = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_n, t_n)\}$  be contained in a sphere of radius  $R$  about the origin. Assume the dataset to be linearly separable, and let  $\mathbf{w}_{\text{opt}}$ ,  $\|\mathbf{w}_{\text{opt}}\| = 1$ , define the hyperplane separating the samples, having functional margin  $\gamma > 0$ . We initialise the normal vector as  $\mathbf{w}_0 = \mathbf{0}$ . The number of updates,  $k$ , of the perceptron algorithms is then bounded by*

$$k \leq \left( \frac{2R}{\gamma} \right)^2. \quad (10)$$

Input ball:  $R = \max_i \|\mathbf{x}_i\|$ .

Margin:  $\gamma := \min_i y_i f(\mathbf{x}_i)$





# Mathematics of Superposition Representation

# Hilbert's 13th Problem

Algebraic equations (under a suitable transformation) of degree up to 6 can be solved by functions of two variables. What about

$$x^7 + ax^3 + bx^2 + cx + 1 = 0?$$

Hilbert's conjecture:  $x(a, b, c)$  cannot be expressed by a superposition (sums and compositions) of bivariate functions.

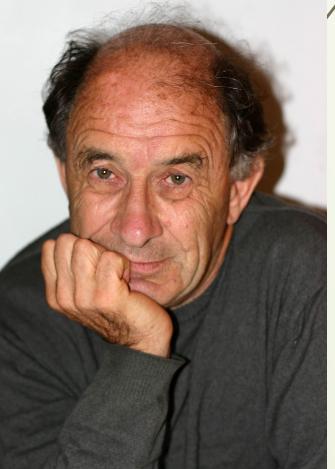
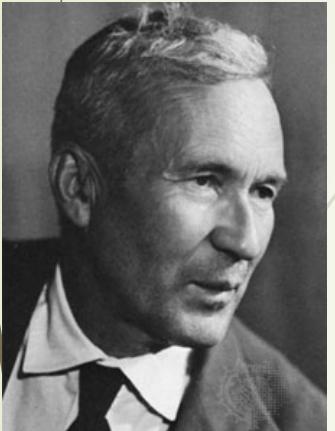


**Question:** can every continuous (analytic,  $C^\infty$ , etc) function of  $n$  variables be represented as a superposition of continuous (analytic,  $C^\infty$ , etc) functions of  $n - 1$  variables?

**Theorem (D. Hilbert)**

*There is an analytic function of three variables that cannot be expressed as a superposition of bivariate ones.*

# Kolmogorov's Superposition Theorem



Theorem (A. Kolmogorov, 1956; V. Arnold, 1957)

Given  $n \in \mathbb{Z}^+$ , every  $f_0 \in C([0, 1]^n)$  can be represented as

$$f_0(x_1, x_2, \dots, x_n) = \sum_{q=1}^{2n+1} g_q \left( \sum_{p=1}^n \phi_{pq}(x_p) \right),$$

where  $\phi_{pq} \in C[0, 1]$  are increasing functions independent of  $f_0$  and  $g_q \in C[0, 1]$  depend on  $f_0$ .

- Can choose  $g_q$  to be all the same  $g_q \equiv g$  (Lorentz, 1966).
- Can choose  $\phi_{pq}$  to be Hölder or Lipschitz continuous, but not  $C^1$  (Fridman, 1967).
- Can choose  $\phi_{pq} = \lambda_p \phi_q$  where  $\lambda_1, \dots, \lambda_n > 0$  and  $\sum_p \lambda_p = 1$  (Sprecher, 1972).

If  $f$  is a multivariate continuous function, then  $f$  can be written as a superposition of composite functions of mixtures of continuous functions of single variables:  
finite **composition** of continuous functions of a **single variable** and the **addition**.

# Kolmogorov's Exact Representation is not stable or smooth

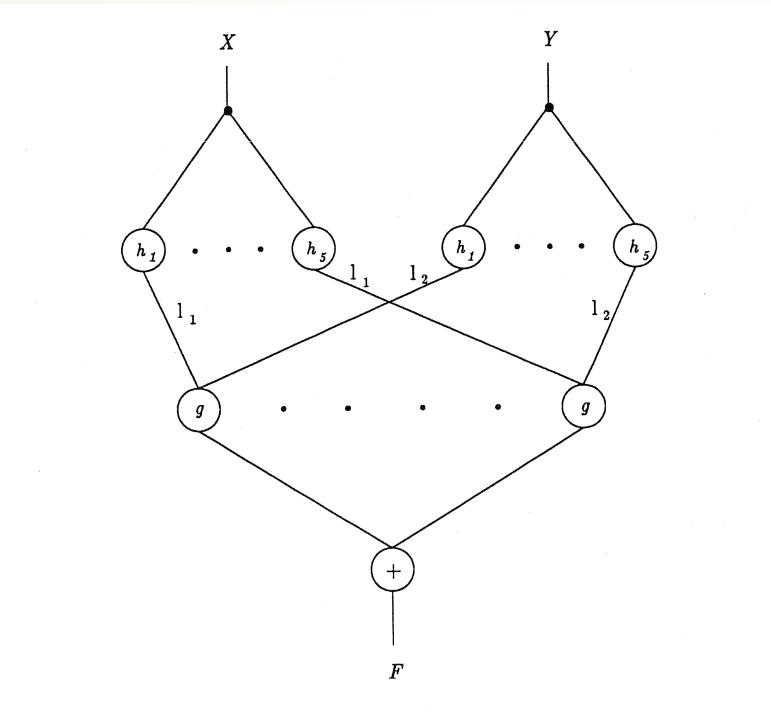


Figure 1: The network representation of an improved version of Kolmogorov's theorem, due to Kahane (1975). The figure shows the case of a bivariate function. The Kahane's representation formula is  $f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} g[\sum_{p=1}^n l_p h_q(x_p)]$  where  $h_q$  are strictly monotonic functions and  $l_p$  are strictly positive constants smaller than 1.

- ▶ [Girosi-Poggio'1989] Representation Properties of Networks:  
Kolmogorov's Theorem Is Irrelevant,  
<https://www.mitpressjournals.org/doi/pdf/10.1162/neco.1989.1.4.465>
- ▶ Lacking smoothness in  $h$  and  $g$   
[Vitushkin'1964] fails to guarantee the **generalization ability (stability)** against noise and perturbations
- ▶ The representation is **not universal** in the sense that  $g$  and  $h$  both depend on the function  $F$  to be represented.

# A Simplified illustration by David McAllester (TTI-Chicago)

## A Simpler, Similar Theorem

For (possibly discontinuous)  $f : [0, 1]^N \rightarrow \mathbb{R}$  there exists (possibly discontinuous)  $g, h_i : \mathbb{R} \rightarrow \mathbb{R}$ .



$$f(x_1, \dots, x_N) = g \left( \sum_i h_i(x_i) \right)$$

Proof: Select  $h_i$  to spread out the digits of its argument so that  $\sum_i h_i(x_i)$  contains all the digits of all the  $x_i$ .

# Universal Approximate Representation

[Cybenko'1989, Hornik et al. 1989, Poggio-Girosi'1989, ...]

For continuous  $f : [0, 1]^N \rightarrow \mathbb{R}$  and  $\varepsilon > 0$  there exists

$$F(x) = \alpha^\top \sigma(Wx + \beta)$$

$$= \sum_i \alpha_i \sigma \left( \sum_j W_{i,j} x_j + \beta_i \right)$$

such that for all  $x$  in  $[0, 1]^N$  we have  $|F(x) - f(x)| < \varepsilon$ .

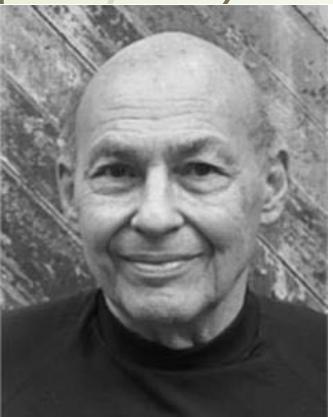
Complexity (regularity, smoothness) thereafter becomes the central pursuit in Approximation Theory.

# Locality or Sparsity of Computation

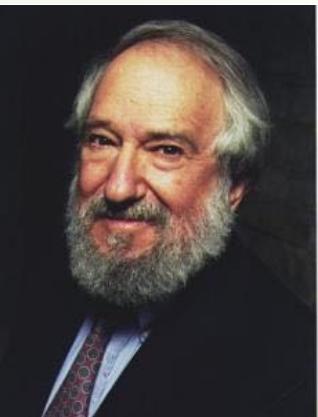
Minsky and Papert, 1969

Perceptron can't do **XOR** classification

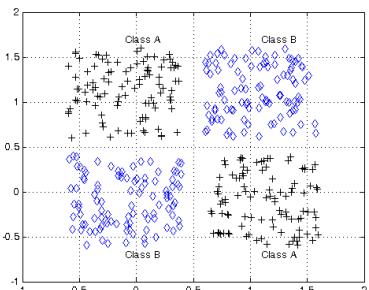
Perceptron needs infinite global  
information to compute **connectivity**



Marvin Minsky

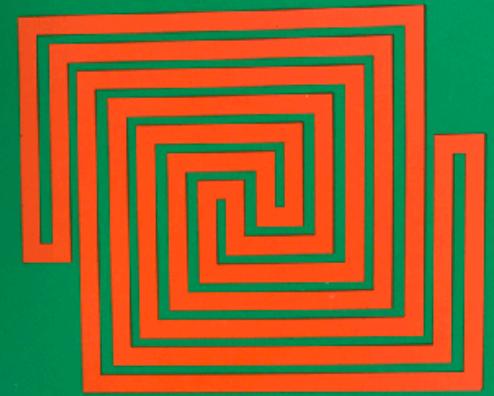


Seymour Papert



**Locality** or **Sparsity** is important:  
Locality in time?  
Locality in space?

Expanded Edition



Perceptrons



Marvin L. Minsky  
Seymour A. Papert

# Locality or Sparsity is a fundamental limitation

Definition (Minsky-Papert'1969)

The decision function that  $f(X) \in \{1, -1\}$  for  $X \subseteq \mathbb{R}^p$  has **order**  $k$ , if it can be represented by a superposition of functions whose supports are at most  $k$ , i.e. there exists a (possibly of infinite members) family of  $\{\phi_\alpha(X) : \text{supp}(\phi_\alpha) \leq k\}$  such that

$$f(X) = \sum_{\alpha} \phi_{\alpha}(X)$$

- Minsky-Papert model admits **infinitely many neurons** (wide network) in parallel processing, yet only **sparse** or **local inputs**. Note that it is not a Turing model.

# Examples of Finite Orders

- $f(X) = [X \text{ is nonempty}]$  has order 1, as  $\phi_a(X) = [a \in X]$  and  $f(X) = \sum_a \phi_a(X)$ .
- $f(X) = [X \text{ is convex}]$  has order 3, as

$$f(X) = - \sum_{a,b \in X} [\text{midpoint } ([a, b]) \text{ not in } X]$$

- The only topologically invariant predicates of finite order are functions of the Euler number  $E(X)$ , which for simplicial complex  $X \subseteq \mathbb{R}^2$  is defined as

$$\begin{aligned} E(X) &:= \#(\text{faces } (X)) - \#(\text{edges } (X)) + \#(\text{vertices } (X)) \\ &= \beta_0 - \beta_1 \end{aligned}$$

# Connectivity is of infinite order

- Which one of these two figures is connected?



Figure 5.1

Theorem (Minsky-Papert'1969)

*The decision function that  $f(X) = [X \text{ is connected}]$  for  $X \subseteq \mathbb{R}^p$  is **not of any finite order**, i.e. for any  $k < \infty$ , there does not exist a (possibly of infinite members) family of  $\{\phi_\alpha(X) : \text{supp}(\phi_\alpha) \leq k\}$  whose supports are at most  $k$ , such that*

$$f(X) = \left[ \sum_{\alpha} \phi_{\alpha}(X) \geq 0 \right] \quad (21)$$

# Multilayer Perceptrons (MLP) and Back-Propagation (BP) Algorithms



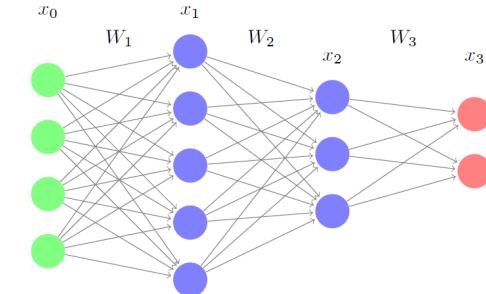
Rumelhart, Hinton, Williams (1986)

Learning representations by back-propagating errors, Nature, 323(9): 533-536

BP algorithms as **stochastic gradient descent** algorithms (**Robbins–Monro 1950; Kiefer–Wolfowitz 1951**) with Chain rules of Gradient maps for multi-layer perceptrons

MLP classifies **XOR**, yet **connectivity?** Condition number in **Blum-Shub-Smale** real computation models helps.

NATURE VOL. 323 9 OCTOBER 1986 LETTERS TO NATURE 533



## Learning representations by back-propagating errors

David E. Rumelhart\*, Geoffrey E. Hinton† & Ronald J. Williams\*

\* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA  
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, USA

We describe a new learning procedure, back-propagation, for networks of neuron-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output code to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-error-correcting rule.

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors<sup>1</sup>. Learning becomes more interesting but

more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input-output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate learned representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom, any number of intermediate layers, and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input,  $x_j$ , to unit  $j$  is a linear function of the outputs,  $y_i$ , of the units that are connected to  $j$  and of the weights,  $w_{ij}$ , on these connections

$$x_j = \sum_i y_i w_{ij} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output,  $y_j$ , which is a non-linear function of its total input

$$y_j = \frac{1}{1 + e^{-x_j}} \quad (2)$$

\* To whom correspondence should be addressed

# BP Algorithm: Network Forward

- Cascade of repeated [linear operation followed by coordinatewise nonlinearity]'s
- Nonlinearities: sigmoid, hyperbolic tangent, (recently) ReLU.

---

## Algorithm 1 Forward pass

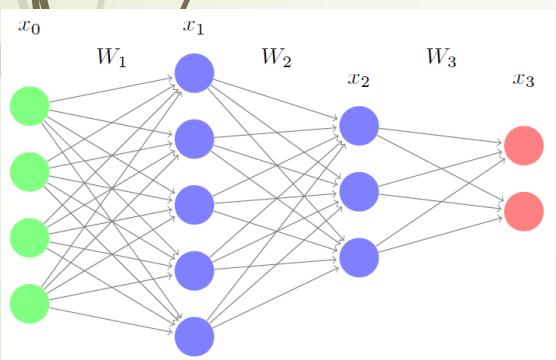
---

**Input:**  $x_0$

**Output:**  $x_L$

```
1: for  $\ell = 1$  to  $L$  do
2:    $x_\ell = f_\ell(W_\ell x_{\ell-1} + b_\ell)$ 
3: end for
```

---



# BP algorithm = Gradient Descent Method

- Training examples  $\{x_0^i\}_{i=1}^n$  and labels  $\{y^i\}_{i=1}^n$
- Output of the network  $\{x_L^i\}_{i=1}^m$
- Objective    Square loss, cross-entropy loss, etc.

$$J(\{W_l\}, \{b_l\}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \|y^i - x_L^i\|_2^2 \quad (1)$$

- Gradient descent

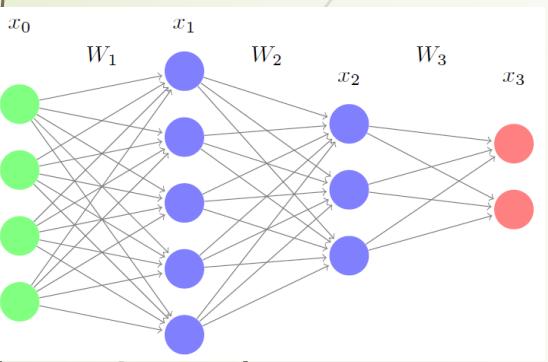
$$W_l = W_l - \eta \frac{\partial J}{\partial W_l}$$

$$b_l = b_l - \eta \frac{\partial J}{\partial b_l}$$

In practice: use Stochastic Gradient Descent (SGD)

# Derivation of BP: Lagrangian Multiplier

LeCun et al. 1988



Given  $n$  training examples  $(I_i, y_i) \equiv (\text{input}, \text{target})$  and  $L$  layers

- Constrained optimization

$$\begin{aligned} & \min_{W,x} && \sum_{i=1}^n \|x_i(L) - y_i\|_2 \\ & \text{subject to} && x_i(\ell) = f_\ell[W_\ell x_i(\ell-1)], \\ & && i = 1, \dots, n, \quad \ell = 1, \dots, L, \quad x_i(0) = I_i \end{aligned}$$

- Lagrangian formulation (Unconstrained)

$$\min_{W,x,B} \mathcal{L}(W, x, B)$$

$$\begin{aligned} \mathcal{L}(W, x, B) = \sum_{i=1}^n & \left\{ \|x_i(L) - y_i\|_2^2 + \right. \\ & \left. \sum_{\ell=1}^L B_i(\ell)^T \left( x_i(\ell) - f_\ell[W_\ell x_i(\ell-1)] \right) \right\} \end{aligned}$$

# back-propagation – derivation

- $\frac{\partial \mathcal{L}}{\partial B}$

## Forward pass

$$x_i(\ell) = f_\ell \left[ \underbrace{W_\ell x_i(\ell-1)}_{A_i(\ell)} \right] \quad \ell = 1, \dots, L, \quad i = 1, \dots, n$$

- $\frac{\partial \mathcal{L}}{\partial x}, z_\ell = [\nabla f_\ell] B(\ell)$

## Backward (adjoint) pass

$$z(L) = 2\nabla f_L \left[ A_i(L) \right] (y_i - x_i(L))$$

$$z_i(\ell) = \nabla f_\ell \left[ A_i(\ell) \right] W_{\ell+1}^T z_i(\ell+1) \quad \ell = 0, \dots, L-1$$

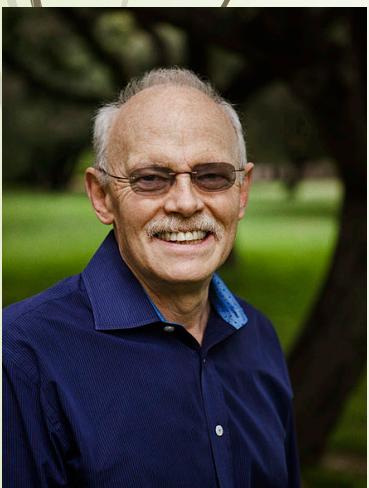
- $W \leftarrow W + \lambda \frac{\partial \mathcal{L}}{\partial W}$

## Weight update

$$W_\ell \leftarrow W_\ell + \lambda \sum_{i=1}^n z_i(\ell) x_i^T(\ell-1)$$

# Parallel Distributed Processing

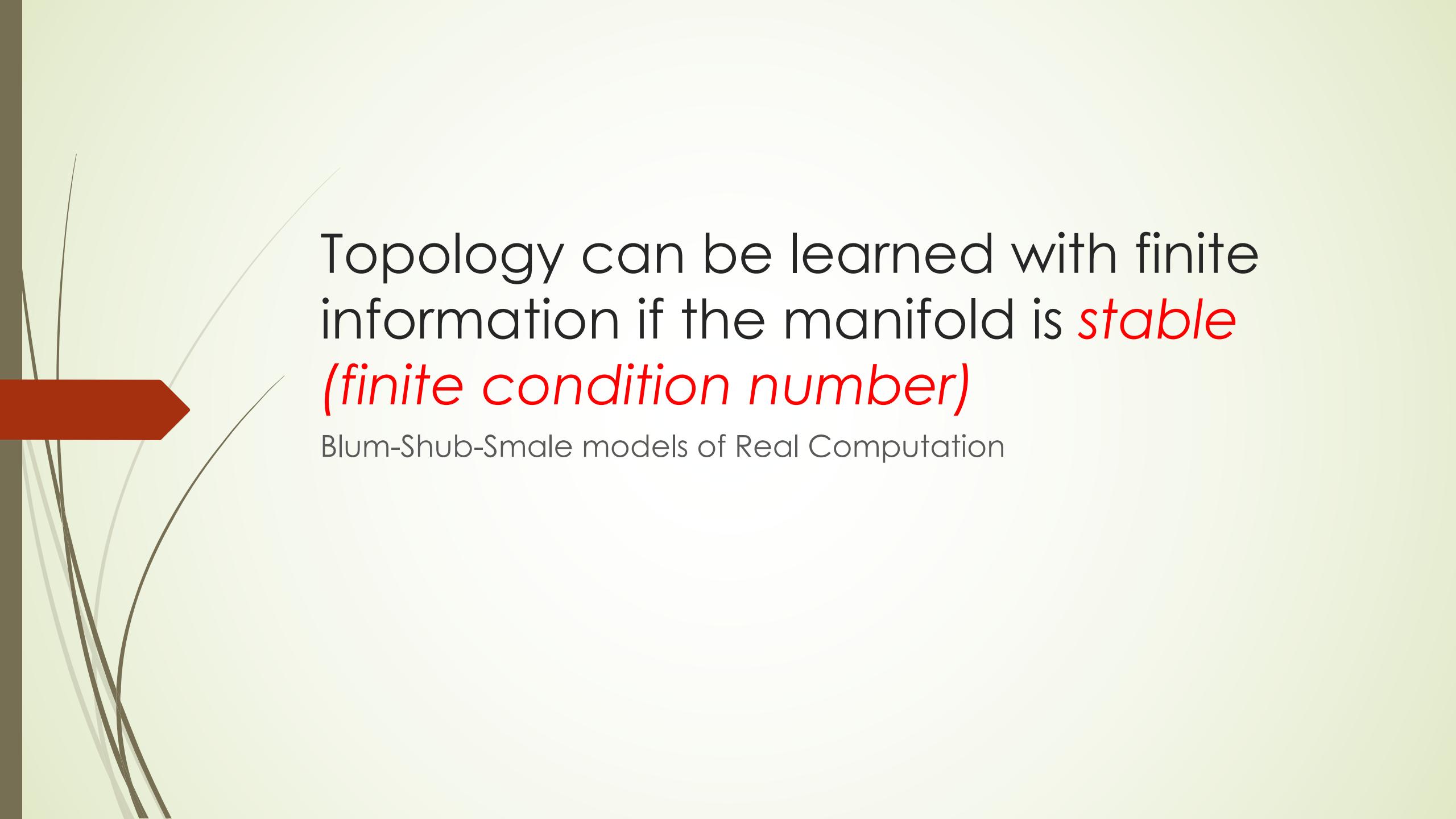
by Rumelhart and McClelland, 1986



Minsky and Papert set out to show which functions can and cannot be computed by this class of machines. They demonstrated, in particular, that such perceptrons are unable to calculate such mathematical functions as parity (whether an odd or even number of points are on in the retina) or the topological function of connectedness (whether all points that are on are connected to all other points that are on either directly or via other points that are also on) without making use of absurdly large numbers of predicates. The analysis is extremely elegant and demonstrates the importance of a mathematical approach to analyzing

of multilayer networks that compute parity). Similarly, it is not difficult to develop networks capable of solving the connectedness or inside/outside problem. Hinton and Sejnowski have analyzed a version of such a network (see Chapter 7).

Essentially, then, although Minsky and Papert were exactly correct in their analysis of the *one-layer perceptron*, the theorems don't apply to systems which are even a little more complex. In particular, it doesn't apply to multilayer systems nor to systems that allow feedback loops.



Topology can be learned with finite information if the manifold is *stable*  
*(finite condition number)*

Blum-Shub-Smale models of Real Computation

# A Model of Real Computation



- ▶ Starting from **Blum, Shub, Smale** (1989)
- ▶ It admits inputs and operations (addition, subtraction, multiplication, and (in the case of fields) division) of **real (complex) numbers** with *infinite precision*
- ▶ “The key importance of the **condition number**, which measures the closeness of a problem instance to the manifold of ill-posed instances, is clearly developed.” – **Richard Karp**



# The Condition Number of a Manifold

Throughout our discussion, we associate to  $\mathcal{M}$  a condition number  $(1/\tau)$  where  $\tau$  is defined as the largest number having the property: The open normal bundle about  $\mathcal{M}$  of radius  $r$  is embedded in  $\mathbb{R}^N$  for every  $r < \tau$ . Its image  $\text{Tub}_\tau$  is a tubular neighborhood of  $\mathcal{M}$  with its canonical projection map

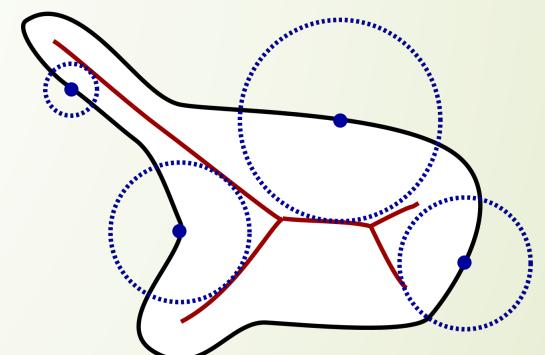
$$\pi_0 : \text{Tub}_\tau \rightarrow \mathcal{M}.$$

## Smallest Local Feature Size

$$G = \{x \in \mathbb{R}^N \text{ such that } \exists \text{ distinct } p, q \in \mathcal{M} \text{ where } d(x, \mathcal{M}) = \|x - p\| = \|x - q\|\},$$

where  $d(x, \mathcal{M}) = \inf_{y \in \mathcal{M}} \|x - y\|$  is the distance of  $x$  to  $\mathcal{M}$ . The closure of  $G$  is called the medial axis and for any point  $p \in \mathcal{M}$  the local feature size  $\sigma(p)$  is the distance of  $p$  to the medial axis. Then it is easy to check that

$$\tau = \inf_{p \in \mathcal{M}} \sigma(p).$$



# Find Homology with Finite Samples

[Niyogi, Smale, Weinberger (2008)]

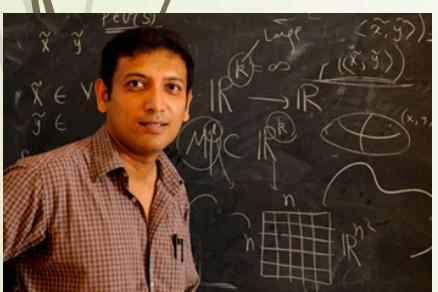
**Theorem 3.1** Let  $\mathcal{M}$  be a compact submanifold of  $\mathbb{R}^N$  with condition number  $\tau$ . Let  $\bar{x} = \{x_1, \dots, x_n\}$  be a set of  $n$  points drawn in i.i.d. fashion according to the uniform probability measure on  $\mathcal{M}$ . Let  $0 < \epsilon < \tau/2$ . Let  $U = \bigcup_{x \in \bar{x}} B_\epsilon(x)$  be a correspondingly random open subset of  $\mathbb{R}^N$ . Then for all

$$n > \beta_1 \left( \log(\beta_2) + \log\left(\frac{1}{\delta}\right) \right),$$

the homology of  $U$  equals the homology of  $\mathcal{M}$  with high confidence (probability  $> 1 - \delta$ ).

$$\beta_1 = \frac{\text{vol}(\mathcal{M})}{(\cos^k(\theta_1))\text{vol}(B_{\epsilon/4}^k)} \quad \text{and} \quad \beta_2 = \frac{\text{vol}(\mathcal{M})}{(\cos^k(\theta_2))\text{vol}(B_{\epsilon/8}^k)}.$$

Here  $k$  is the dimension of the manifold  $\mathcal{M}$  and  $\text{vol}(B_\epsilon^k)$  denotes the  $k$ -dimensional volume of the standard  $k$ -dimensional ball of radius  $\epsilon$ . Finally,  $\theta_1 = \arcsin(\epsilon/8\tau)$  and  $\theta_2 = \arcsin(\epsilon/16\tau)$ .



# Curse of Dimensionality and “Quantum Algorithms”

To construct a Rips-complex of dimension of  $n$  points:  $O(2^n)$  number of simplices is needed in the worst case =>  $O(\text{poly}(n))$  in Quantum Algorithms



 nature  
COMMUNICATIONS

ARTICLE

Received 17 Sep 2014 | Accepted 9 Nov 2015 | Published 25 Jan 2016 DOI: [10.1038/ncomms10138](https://doi.org/10.1038/ncomms10138) OPEN

Quantum algorithms for topological and geometric analysis of data

Seth Lloyd<sup>1</sup>, Silvano Garnerone<sup>2</sup> & Paolo Zanardi<sup>3</sup>

# A Proof of Concept Demonstration by 6-photon Quantum Computer [Huang et al. 2018, arXiv:1801.06316]

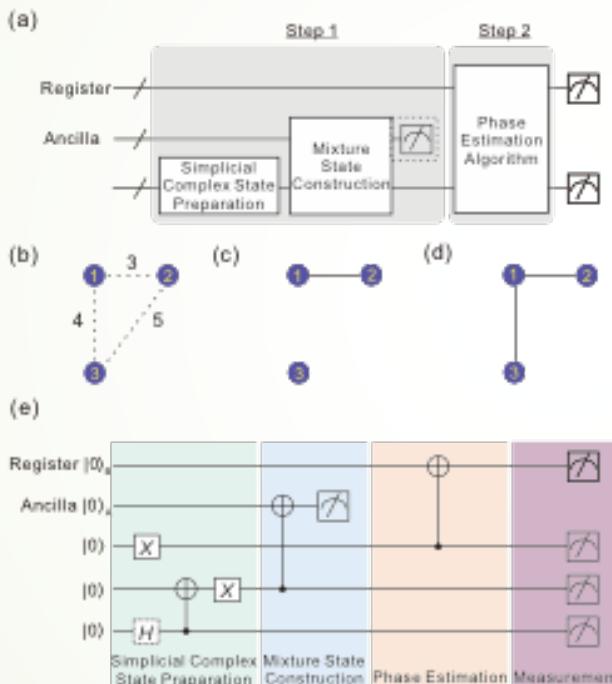


FIG. 2. Quantum circuit for quantum TDA. (a) Outline of the original quantum circuit. (b) A scatterplot including three data points. (c) Graph representation of the 1-simplices state  $|\varphi\rangle_1^{e_1} = |110\rangle$  for  $3 < \epsilon_1 < 4$ . The first and second data points are connected by an edge. (d) Graph representation of 1-simplices state  $|\varphi\rangle_1^{e_2} = (|110\rangle + |101\rangle)/\sqrt{2}$  for  $4 < \epsilon_2 < 5$ . The first data point is connected to the second and third points by two edges. (e) Optimized circuit with 5 qubits. The blocks with different colors represent the four basic stages.

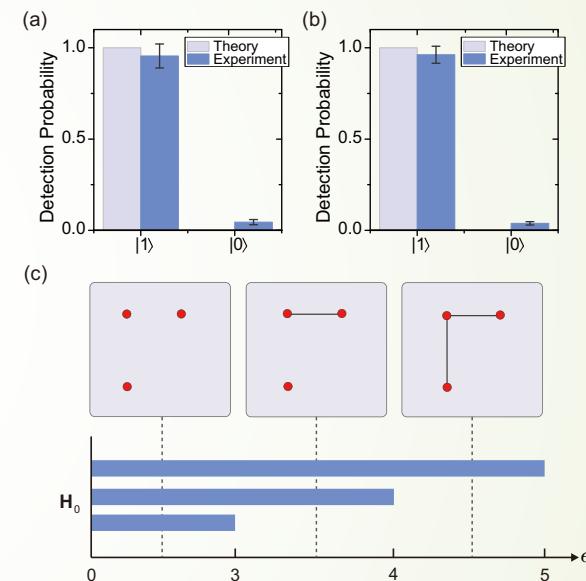


FIG. 4. Final experimental results. The output is determined by measuring the eigenvalue register in the Pauli-Z basis. Measured expectation values (blue bars) and theoretically predicted values (gray bars) are shown for two different 1-simplices state inputs: (a)  $|\varphi\rangle_1^{e_1} = |110\rangle$ , (b)  $|\varphi\rangle_1^{e_2} = (|110\rangle + |101\rangle)/\sqrt{2}$ . Error bars represent one standard deviation, deduced from propagated Poissonian counting statistics of the raw detection events. (c) The barcode for  $0 < \epsilon < 5$ . Since no  $k$ -dimensional holes for  $k \geq 1$  exist at these scales, only the 0-th Betti barcode is given here. For  $0 < \epsilon < 3$ , there is no connection between each point, so the 0-th Betti number is equal to the number of points. That is, there are three bars at  $0 < \epsilon < 3$ . At scales of  $3 < \epsilon_1 < 4$  and  $4 < \epsilon_2 < 5$ , the 0-th Betti number are 2 and 1.



# Convolutional Neural Networks

Fukushima, LeCun, etc.

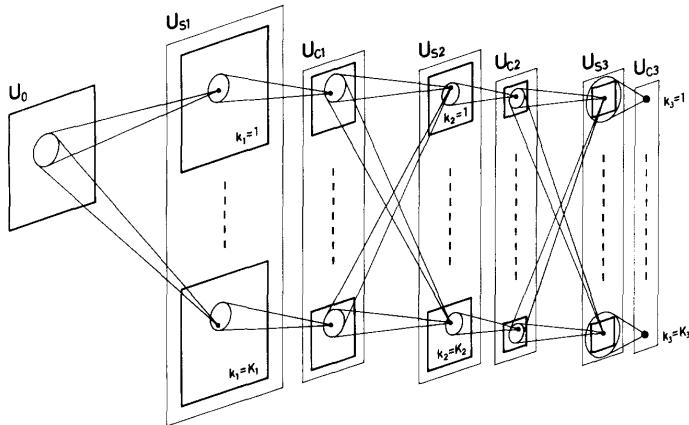
# Convolutional Neural Networks: shift invariances and locality

Biol. Cybernetics 36, 193–202 (1980)

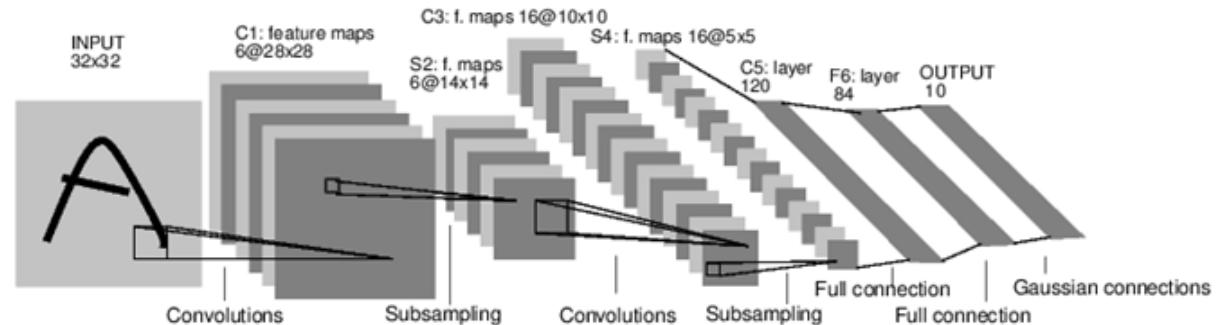
**Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position**

Kunihiko Fukushima

NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan



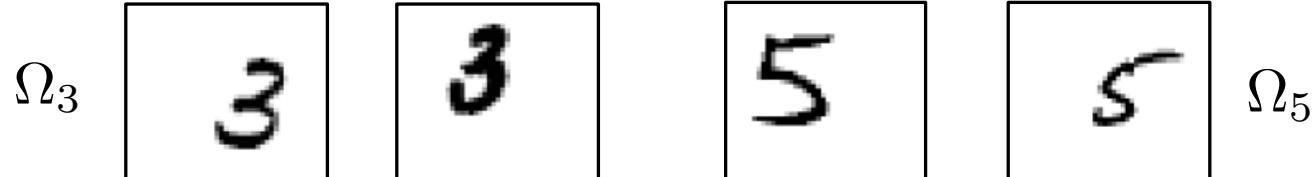
- Can be traced to *Neocognitron* of Kunihiko Fukushima (1979)
- Yann LeCun combined convolutional neural networks with back propagation (1989)
- Imposes **shift invariance** and **locality** on the weights
- Forward pass remains similar
- Backpropagation slightly changes – need to sum over the gradients from all spatial positions



# Translation and Deformations

- Digit classification:

$$x(u) \quad x'(u) = x(u - \tau(u))$$

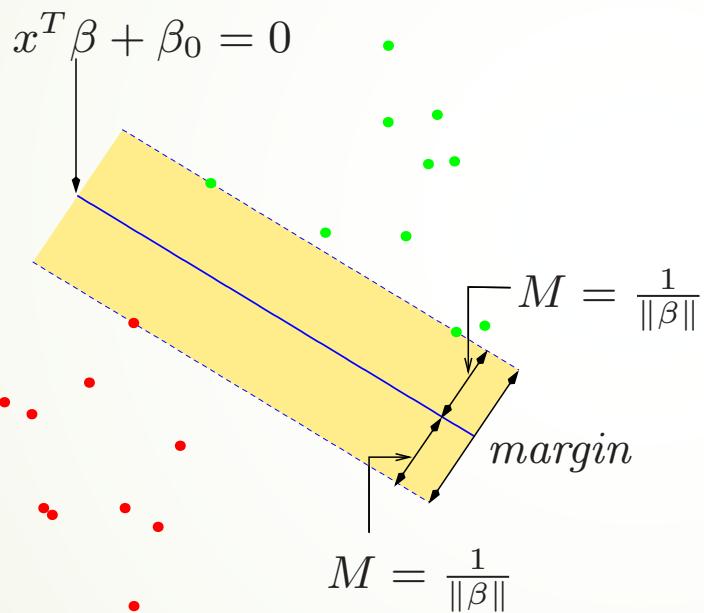


- Globally invariant to the translation group: small
- Locally invariant to small diffeomorphisms: huge group



*Video of Philipp Scott Johnson*

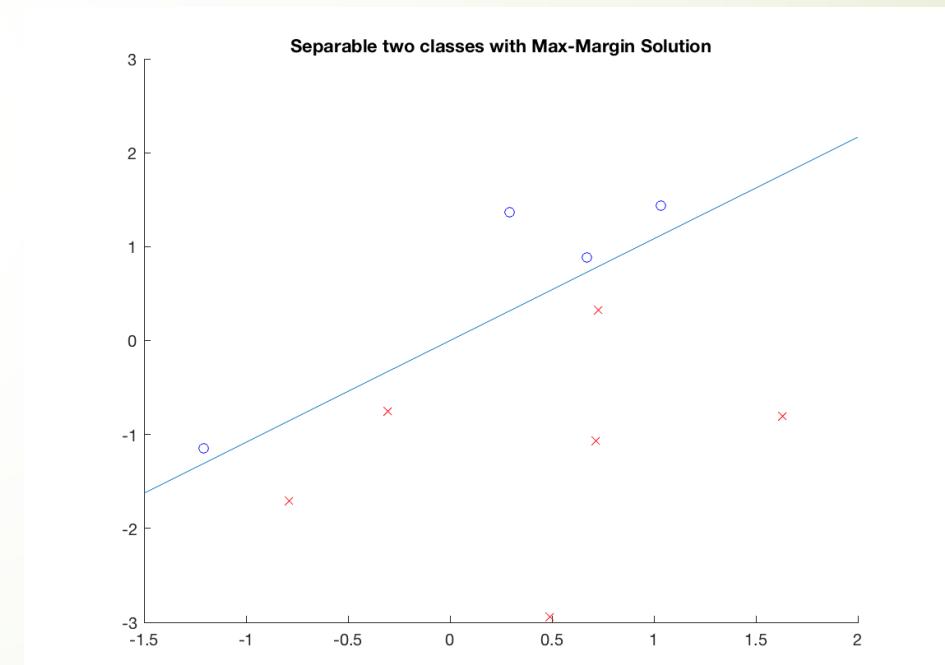
# Max-Margin Classifier (SVM)



Vladimir Vapnik, 1994

$$\text{minimize}_{\beta_0, \beta_1, \dots, \beta_p} \|\beta\|^2 := \sum_j \beta_j^2$$

subject to  $y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq 1$  for all  $i$



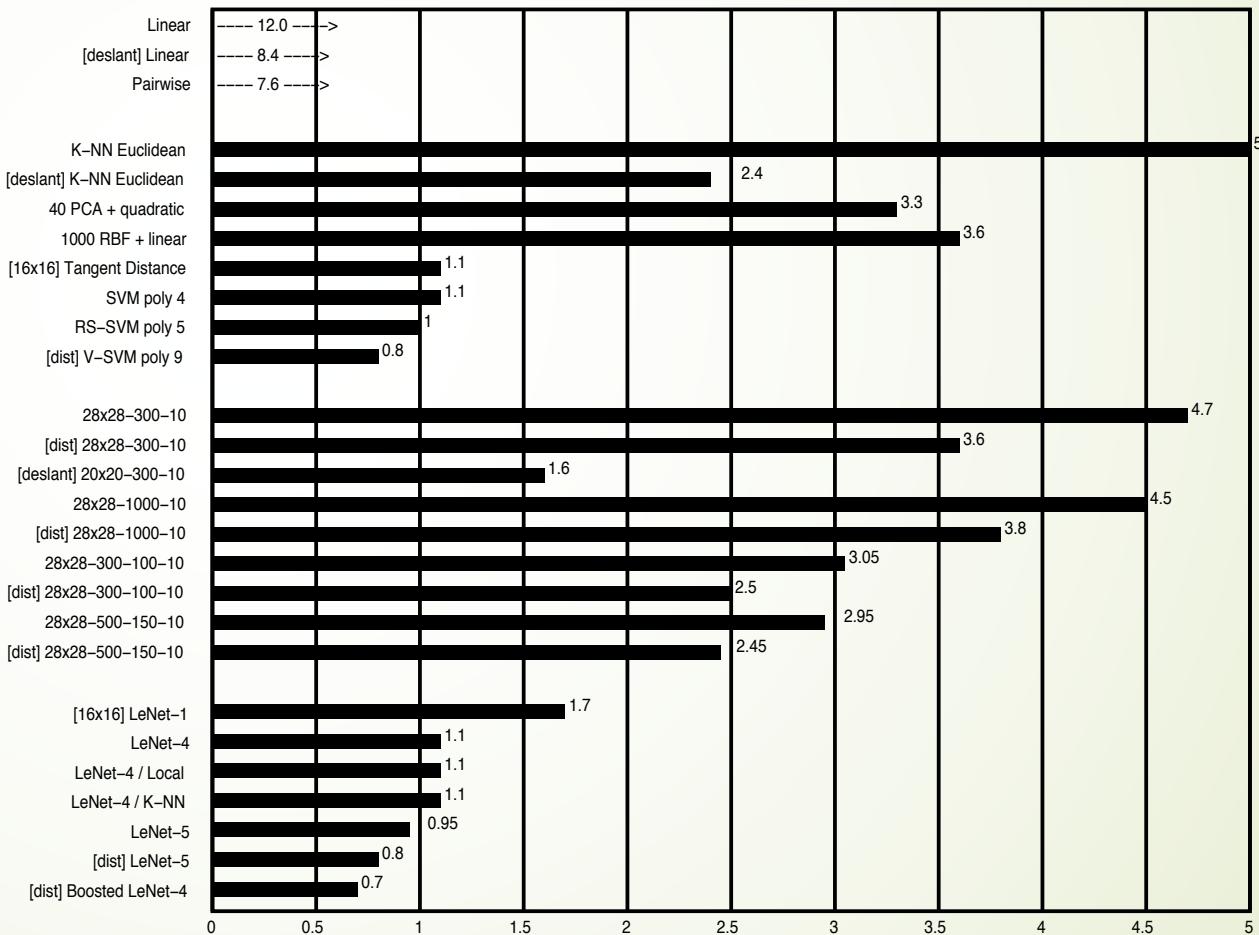
# MNIST Dataset Test Error

## LeCun et al. 1998



Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets)

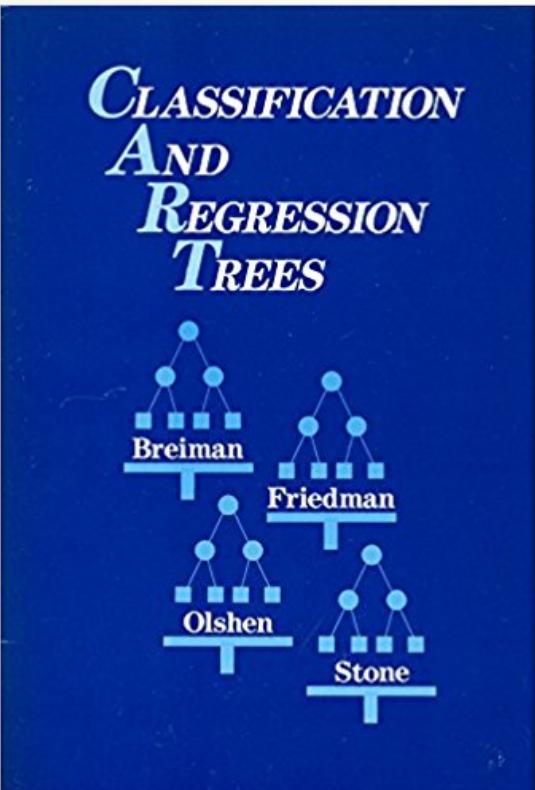
Dark era for NN: 1998-2012



# 2000-2010: The Era of SVM, Boosting, ... as nights of Neural Networks



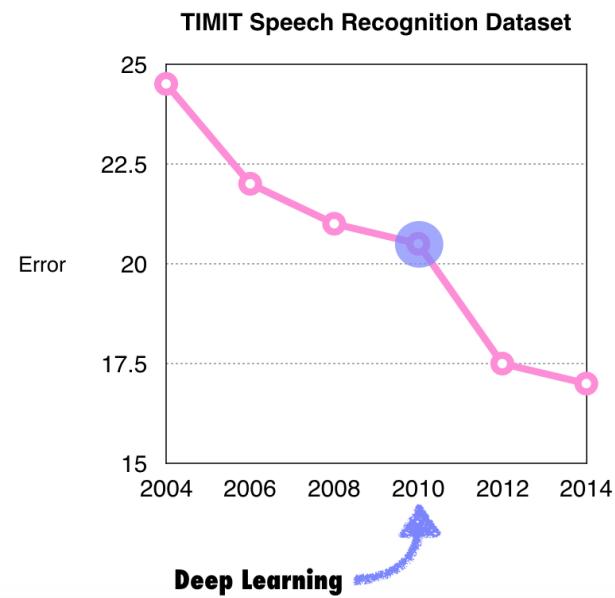
# Decision Trees and Boosting



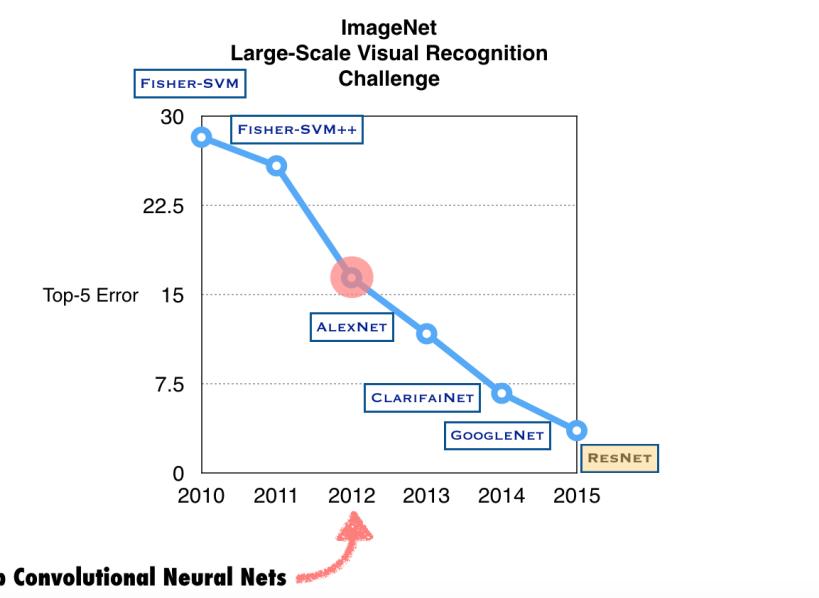
- ▶ Breiman, Friedman, Olshen, Stone, (1983): CART
- ▶ ``The Boosting problem'' (M. Kearns & L. Valiant):  
**Can a set of weak learners create a single strong learner?** (三个臭皮匠顶个诸葛亮?)
- ▶ Breiman (1996): Bagging
- ▶ Freund, Schapire (1997): **AdaBoost** ("the best off-the-shelf algorithm" by Breiman)
- ▶ Breiman (2001): **Random Forests**

# Around the year of 2012: return of NN as `deep learning'

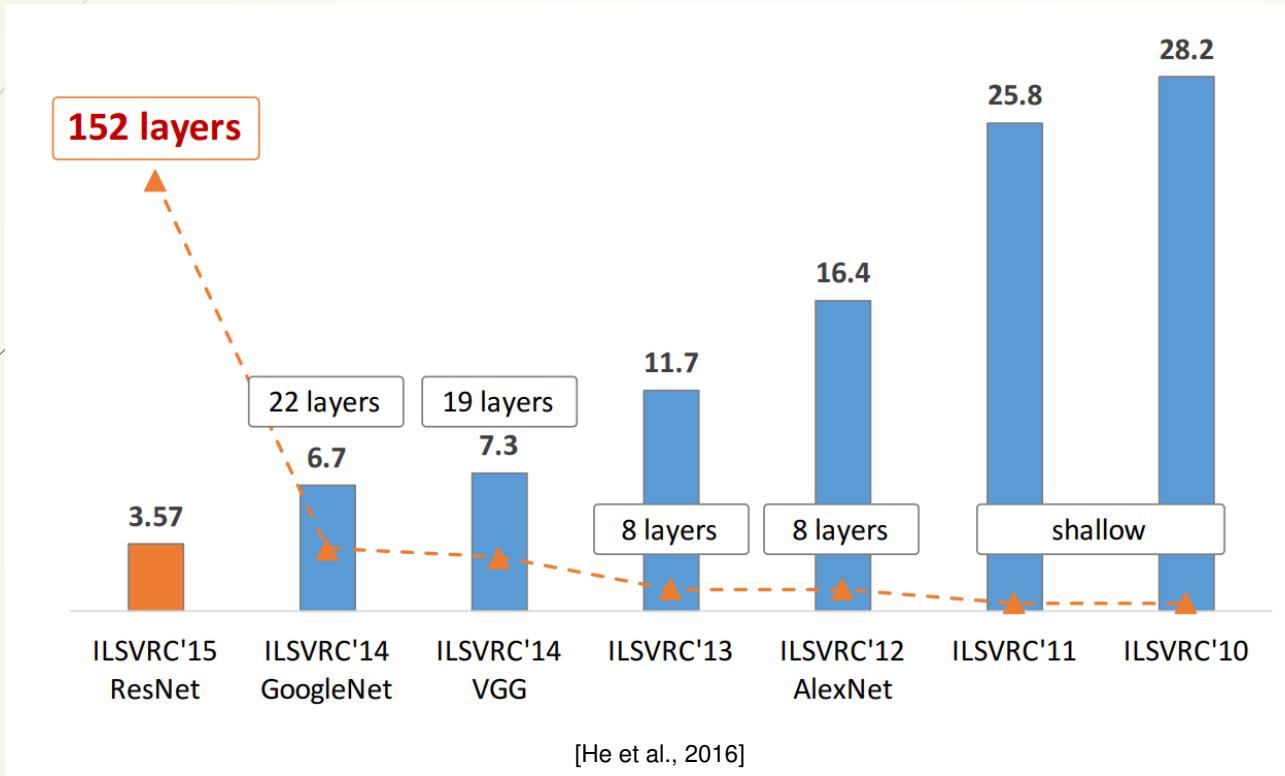
Speech Recognition: TIMIT



Computer Vision: ImageNet

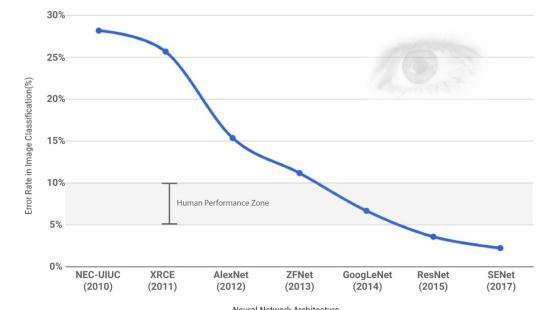


# Depth as function of year



## ILSVRC ImageNet Top 5 errors

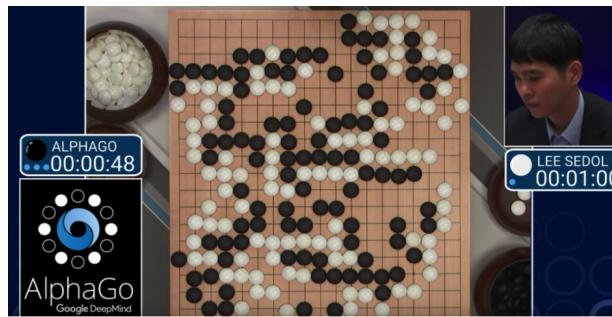
- ImageNet (subset):
  - 1.2 million training images
  - 100,000 test images
  - 1000 classes
- ImageNet large-scale visual recognition Challenge



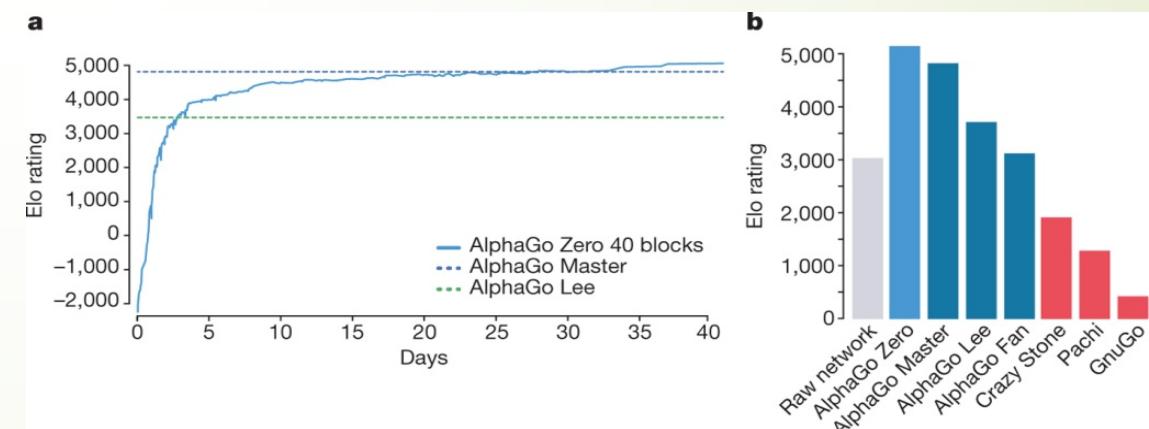
# Reaching Human Performance Level in Games



Deep Blue in 1997



AlphaGo "LEE" 2016



# Number of AI papers on arXiv, 2010-2019

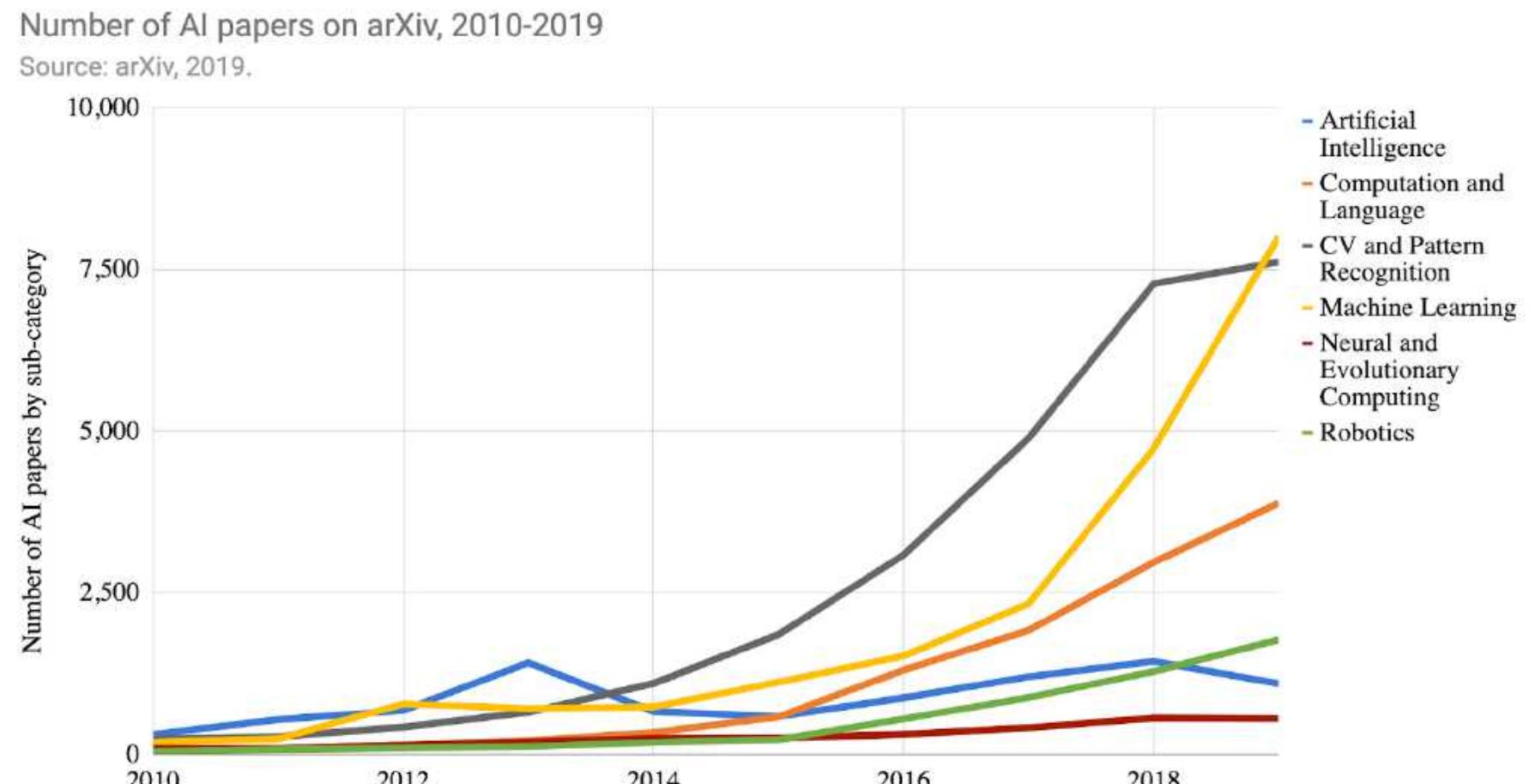
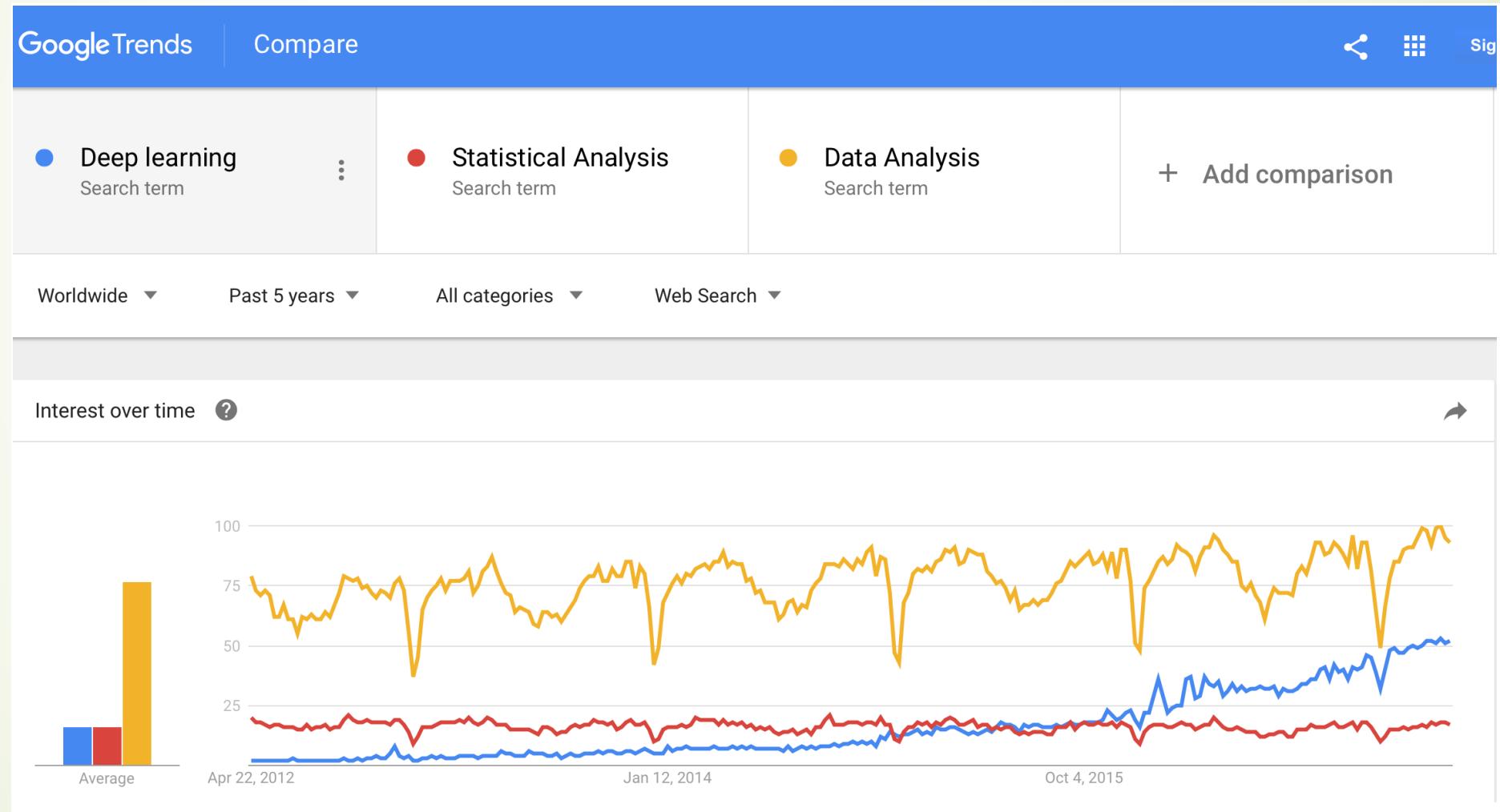


Fig. 1.6.

# Growth of Deep Learning

'Deep Learning' is coined by Hinton et al. in their Restricted Boltzman Machine paper, *Science* 2006, not yet popular until championing ImageNet competitions.





Yet ...

# Some Cold Water: Tesla Autopilot Misclassifies Truck as Billboard



**Problem:** Why? How can you trust a blackbox?

# Deep Learning may be fragile in generalization against noise!



$x$   
“panda”  
57.7% confidence

$+ .007 \times$

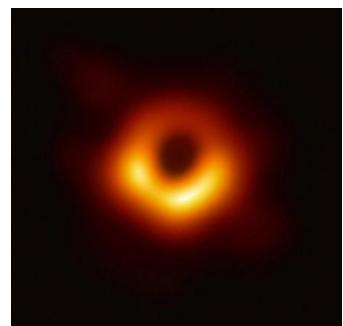


“nematode”  
8.2% confidence

$=$



“gibbon”  
99.3 % confidence

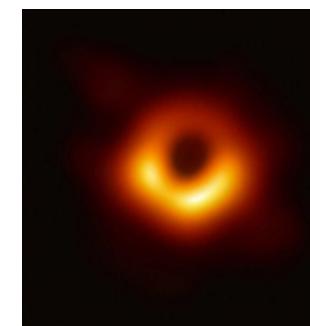


87.7% confidence

$+ .007 \times$



$=$



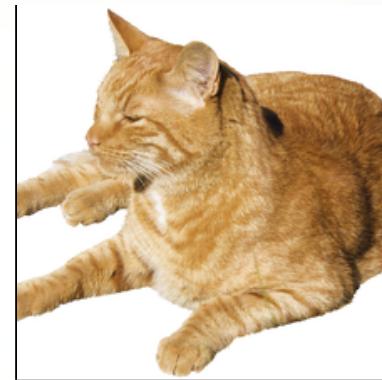
99.3% confidence

[Goodfellow et al., 2014]

# CNN learns **texture** features, not shapes



(a) Texture image  
81.4% **Indian elephant**  
10.3% indri  
8.2% black swan



(b) Content image  
71.1% **tabby cat**  
17.3% grey fox  
3.3% Siamese cat



(c) Texture-shape cue conflict  
63.9% **Indian elephant**  
26.4% indri  
9.6% black swan

Geirhos et al. ICLR 2019

<https://videoken.com/embed/W2HvLBMhCJQ?tocitem=46>

# Spurious Correlations rather than Causation

► Leon Bottou, ICLR 2019

Example: detection of the action “*giving a phone call*”



Not giving a phone call.

Giving a phone call ????

# Overfitting causes privacy leakage

- Model inversion attack leaks privacy



Figure: Recovered (Left), Original (Right)

# What's wrong with deep learning?

**Ali Rahimi** NIPS'17: Machine (deep) Learning has become **alchemy**.  
<https://www.youtube.com/watch?v=ORHFOnaEzPc>

**Yann LeCun** CVPR'15, invited talk: **What's wrong with deep learning?**  
One important piece: **missing some theory!**

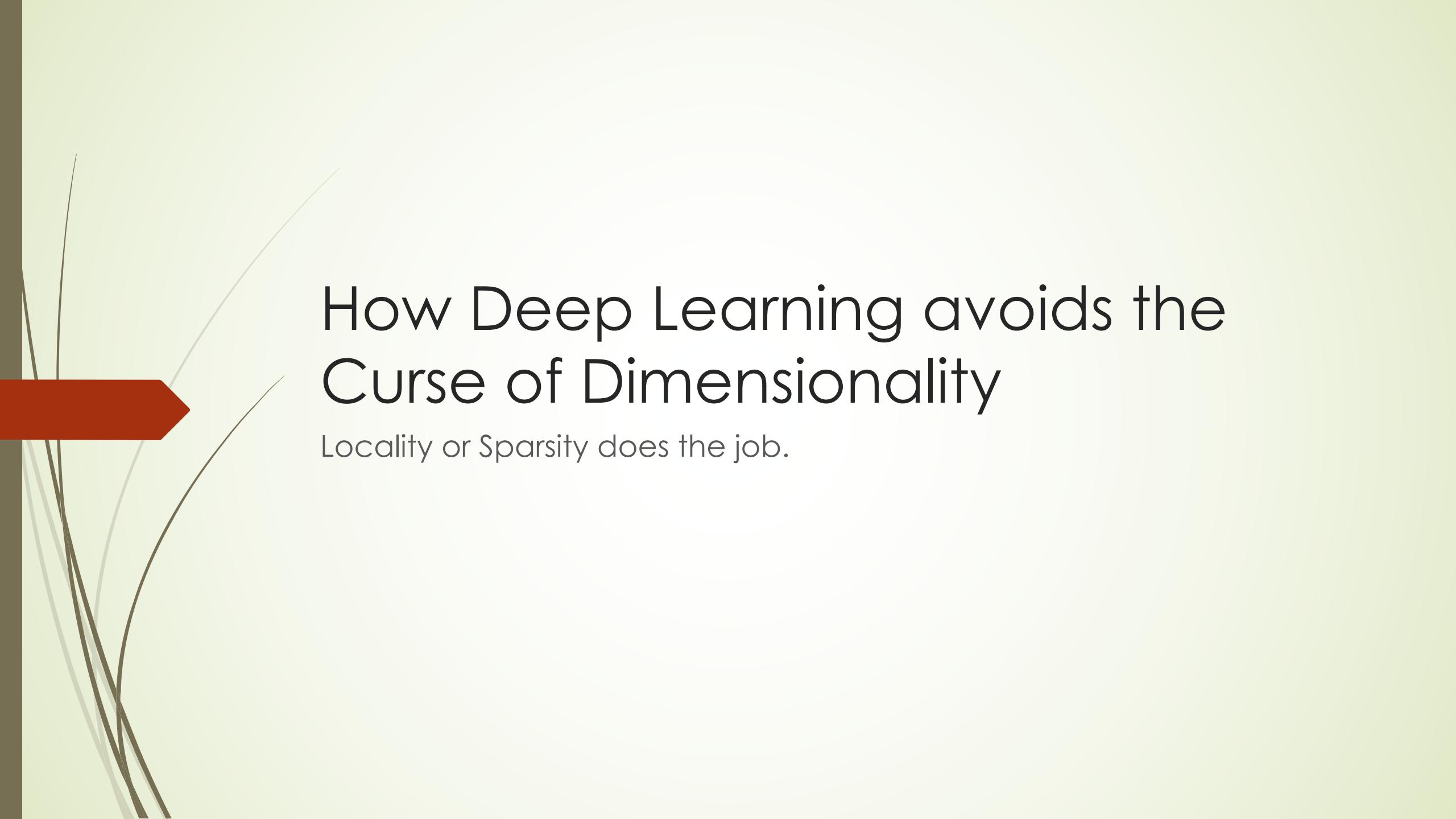
<http://techtalks.tv/talks/whats-wrong-with-deep-learning/61639/>



Being alchemy is certainly not a shame, not wanting to work on advancing to chemistry is a shame! -- **by Eric Xing**

# Some Theoretical Problems

- ▶ Approximation Theory and Harmonic Analysis : **What functions are represented well by deep neural networks, without suffering the curse of dimensionality and better than shallow networks?**
  - ▶ Sparse (local), hierarchical (multiscale), compositional functions avoid the curse of dimensionality
  - ▶ Group (translation, rotational, scaling, deformation) invariances achieved as depth grows
- ▶ Statistics learning: **How can deep learning generalize well without overfitting the noise?**
  - ▶ Over-parameterized models change nonseparable classification to separable, and maximize margin in gradient descent
- ▶ Optimization: **What is the landscape of the empirical risk and how to optimize it efficiently?**
  - ▶ Over-parameterized models make empirical risk landscapes simple (multilinear or 2-layer NN) with degenerate (flat) equilibria
  - ▶ SGD tends to find flat minima, and gradient-free algorithms like block-coordinate-descent
- ▶ **Causal feature learning, interpretability, robustness? ...**

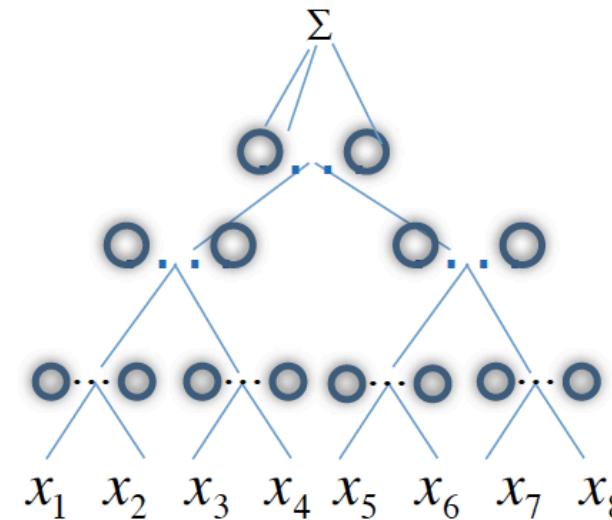
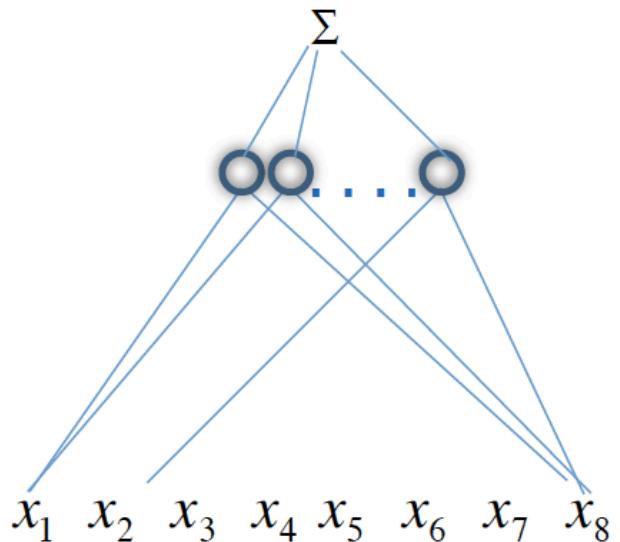


# How Deep Learning avoids the Curse of Dimensionality

Locality or Sparsity does the job.

# Deep and shallow networks: universality

**Theorem** Shallow, one-hidden layer networks with a nonlinear  $\phi(x)$  which is not a polynomial are universal. Arbitrarily deep networks with a nonlinear  $\phi(x)$  (including polynomials) are universal.



$$\phi(x) = \sum_{i=1}^r c_i |< w_i, x > + b_i|_+$$

Cybenko, Girosi, ...

Both deep and shallow models can approximate continuous functions, but suffering the curse of dimensionality... [Cybenko (1989), Hornik (1991), Barron (1993), Mhaskar (1994), Micchelli, Pinkus, Chui-Li ]

# Curse of Dimensionality

$$y = f(x_1, \dots, x_d)$$

## Curse of dimensionality

Both shallow and deep network can approximate a function of  $d$  variables equally well. The number of parameters in both cases depends exponentially on  $d$  as  $O(\varepsilon^{-d})$ .

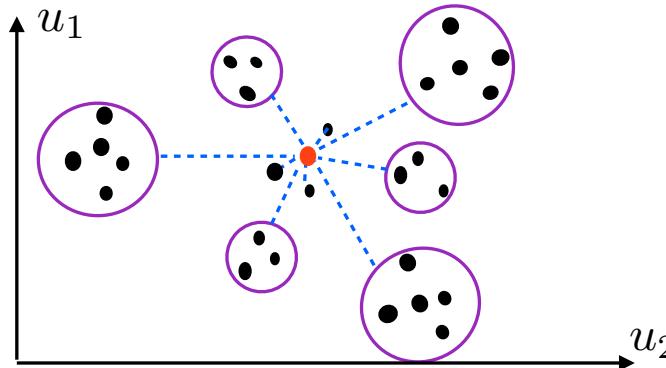


CENTER FOR  
Brains  
Minds +  
Machines

Mhaskar, Poggio, Liao, 2016

# A Blessing from Physical world? Multiscale “compositional” sparsity

- Variables  $x(u)$  indexed by a low-dimensional  $u$ : time/space... pixels in images, particles in physics, words in text...
- Multiscale interactions of  $d$  variables:

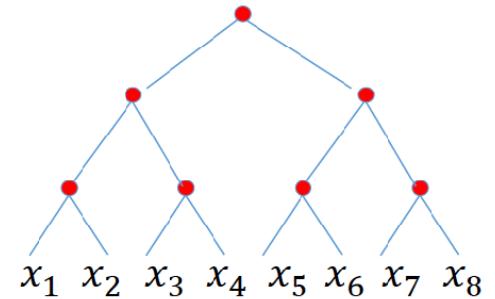


From  $d^2$  interactions to  $O(\log^2 d)$  multiscale interactions.  
(Or even of constant numbers.)

- Multiscale analysis: wavelets on groups of symmetries.  
hierarchical architecture.

# Hierarchically local compositionality

$$f(x_1, x_2, \dots, x_8) = g_3(g_{21}(g_{11}(x_1, x_2), g_{12}(x_3, x_4)), g_{22}(g_{11}(x_5, x_6), g_{12}(x_7, x_8)))$$



## Theorem (informal statement)

Suppose that a function of  $d$  variables is hierarchically, locally, compositional . Both shallow and deep network can approximate  $f$  equally well. The number of parameters of the shallow network depends exponentially on  $d$  as  $O(\varepsilon^{-d})$  with the dimension whereas for the deep network dance is  $O(d\varepsilon^{-2})$



CENTER FOR  
Brains  
Minds +  
Machines

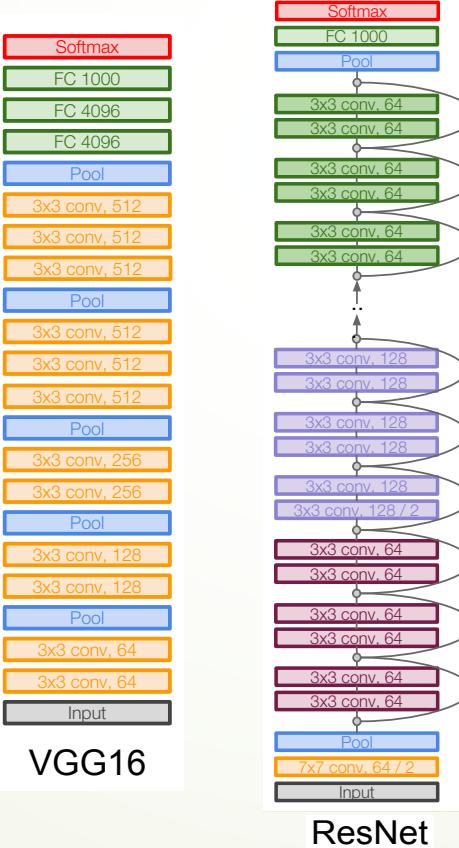
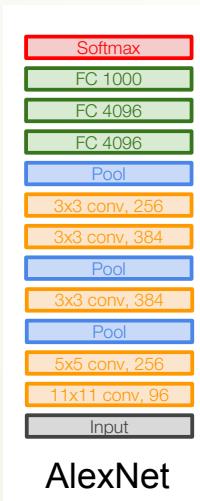
Mhaskar, Poggio, Liao, 2016



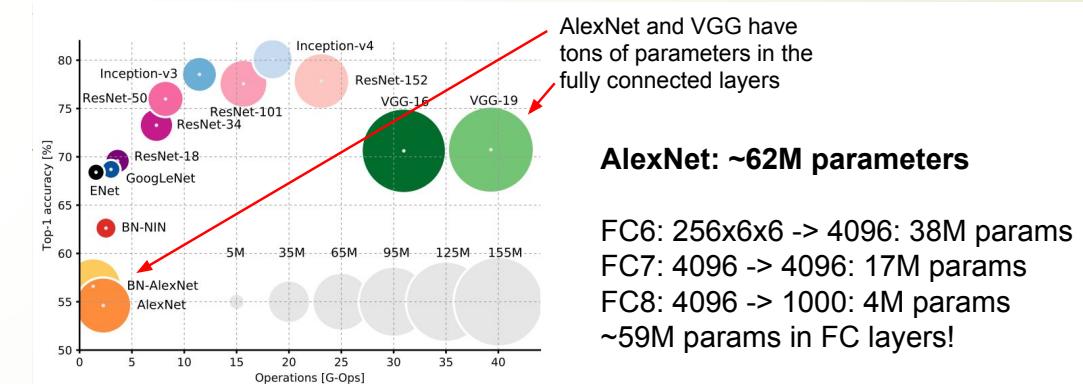
Convolutional Neural Networks (VGG, ResNet etc.) are of this type.

# Local filters of small receptive fields (sparsity) are the key to avoid the curse-of-dimensionality

Stacking local filters -> large receptive fields



Fully connected layers -> explosion of parameters



**AlexNet: ~62M parameters**

FC6: 256x6x6 -> 4096: 38M params  
FC7: 4096 -> 4096: 17M params  
FC8: 4096 -> 1000: 4M params  
~59M params in FC layers!

# Important Special Cases in Statistics

Minimax rates of estimations (Stone 1982): if a regression function  $f$  is Lipschitz on  $\mathbb{R}^d$   $\alpha$  with  $0 < \alpha < 1$ , then the optimal minimax rate of statistical regression estimators with  $N$  samples is  $N^{-\frac{2\alpha}{2\alpha+d}}$ .

Additive models (Stone 1985):  $f(x_1, \dots, x_d) = f_1(x_1) + \dots + f_d(x_d)$  with minimax rate  $N^{-\frac{2\alpha}{2\alpha+1}}$ .

Raskutti-Wainwright-Yu (IEEE TIT, 2011), Yuan-Zhou (AoS, 2016), ...

Interaction models (Stone 1994):  $f = \sum_{I \subseteq \{1, \dots, d\}, |I|=d^*} f_I(x_I)$  with minimax rate  $N^{-\frac{2\alpha}{2\alpha+d^*}}$ . Here  $d^* \in \{1, \dots, d\}$  and for  $I = \{i_1, \dots, i_{d^*}\} \subseteq \{1, \dots, d\}$  with  $|I| = d^*$ ,  $x_I = (x_{i_1}, \dots, x_{i_{d^*}})$ .



Single index models (Härdle and Stoker 1989):  $f = g(a \cdot x)$  for some  $a \in \mathbb{R}^d$  and  $g : \mathbb{R} \rightarrow \mathbb{R}$

Projection pursuit (Friedman and Stuetzle 1981):  $f(x_1, \dots, x_d) = \sum_{k=1}^K g_k(a_k \cdot x)$  with  $K \in \mathbb{N}$ ,  $a_k \in \mathbb{R}^d$  and univariate functions  $g_k$

Hierarchical interaction models (Kohler1 and Krzyzak 2016)

Simple case:  $f = g(f_1(x_{I_1}), f_2(x_{I_2}), \dots, f_{d^*}(x_{I_{d^*}}))$

Generalized hierarchical model:  $f = g(a_1 \cdot x, \dots, a_{d^*} \cdot x)$

Generalized hierarchical interaction model:  $f = \sum_k g_k(f_{1,k}, \dots, f_{d^*,k})$  with  $f_{i,k}(x)$  generalized hierarchical model

All models are wrong, but some are useful ... blessing-of-dimensionality?

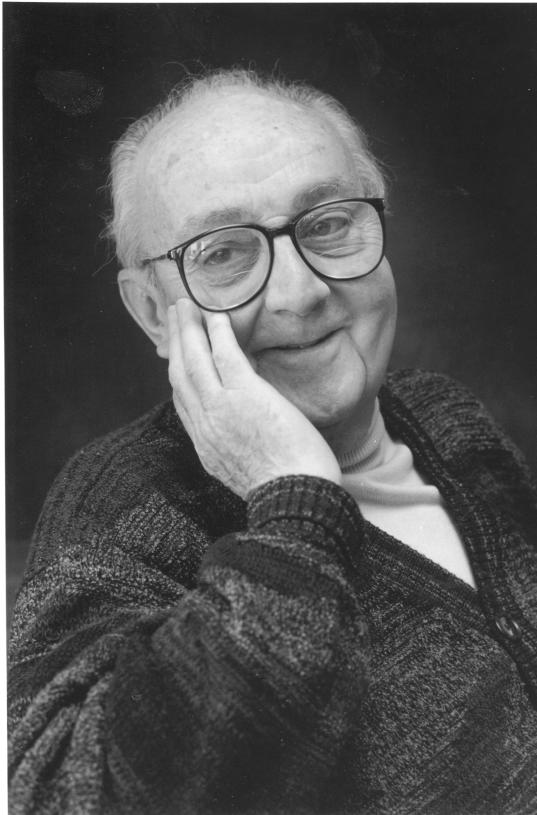


Figure 7: George Box: “Essentially, all models are wrong, but some are useful.”

# Some Historical Results

- ▶ A classical theorem [**Sipser, 1986; Hastad, 1987**] shows that deep circuits are more efficient in representing certain Boolean functions than shallow circuits. Hastad proved that highly-variable functions (in the sense of having high frequencies in their Fourier spectrum) in particular the parity function cannot even be decently approximated by small constant depth circuits
- ▶ **Chui-Li-Mhaskar (1994)** shows that multilayer networks can do localized approximation while single layer ones can not. Older examples exist: consider a function which is a linear combination of  $n$  tensor product Chui–Wang spline wavelets, where each wavelet is a tensor product cubic spline. It was shown by **Chui and Mhaskar** that is impossible to implement such a function using a shallow neural network with a sigmoidal activation function using  $O(n)$  neurons, but a deep network with the activation function  $(x_+)^2$  do so. In this case, as we mentioned, there is a formal proof of a gap between deep and shallow networks.
- ▶ The main result of [**Telgarsky, 2016, Colt**] says that there are functions with many oscillations that cannot be represented by shallow networks with linear complexity but can be represented with low complexity by deep networks.
- ▶ **Eldan and Shamir (2016)** show an example of a function expressible by a 3-layer feedforward neural network cannot be approximated by any 2-layer neural network to certain accuracy unless the width is exponential in the dimension.
- ▶ **Shaham-Cloningen-Coifman (2018)**: functions on manifolds and order of approximation by fully connected deep neural networks



# What are the group invariant properties of deep networks?

Translation, deformation, and general groups defined by classification level sets.

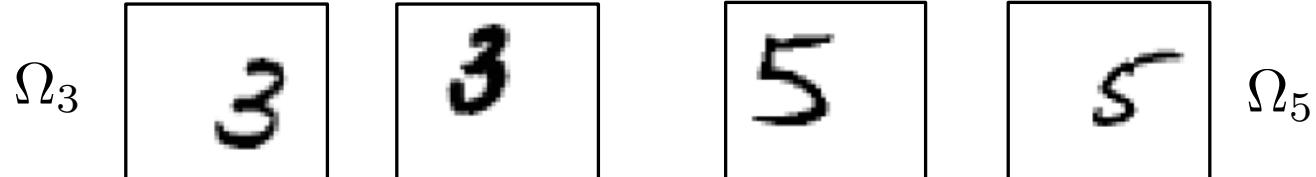


Group invariants starting from Felix Klein's Erlangen Program, 1872

# Translation and Deformations

- Digit classification:

$$x(u) \quad x'(u) = x(u - \tau(u))$$



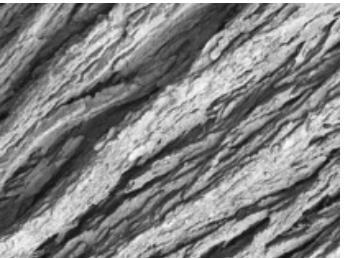
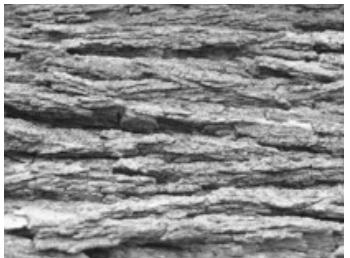
- Globally invariant to the translation group: small
- Locally invariant to small diffeomorphisms: huge group



*Video of Philipp Scott Johnson*

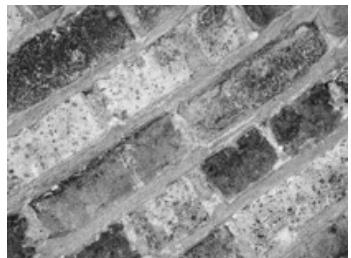
# Rotation and Scaling Variability

- Rotation and **deformations**



Group:  $SO(2) \times \text{Diff}(SO(2))$

- Scaling and **deformations**



Group:  $\mathbb{R} \times \text{Diff}(\mathbb{R})$

# High Dimensional Natural Image Classification

- High-dimensional  $x = (x(1), \dots, x(d)) \in \mathbb{R}^d$ :
- **Classification:** estimate a class label  $f(x)$   
given  $n$  sample values  $\{x_i, y_i = f(x_i)\}_{i \leq n}$

Image Classification     $d = 10^6$

Anchor



Joshua Tree



Beaver



Lotus



Water Lily

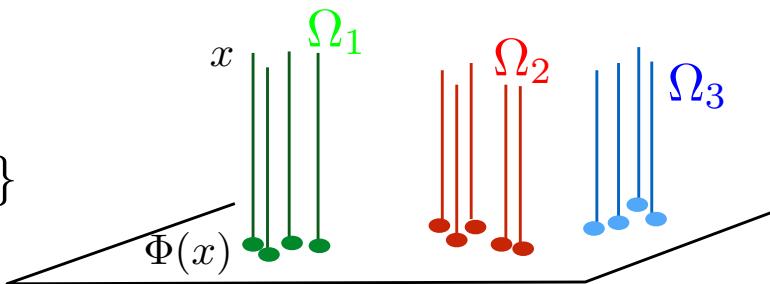


Huge variability  
inside classes

Find invariants

# Fisher's Linear Discriminant (1936) (Linear Dimensionality Reduction)

*Classes*  
*Level sets of  $f(x)$*   
 $\Omega_t = \{x : f(x) = t\}$



If level sets (classes) are parallel to a linear space  
then variables are eliminated by linear projections: *invariants*.

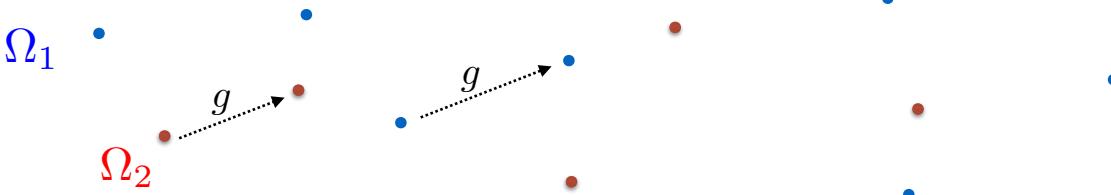
$$\Phi(x) = \alpha \hat{\Sigma}_W^{-1} (\hat{\mu}_1 - \hat{\mu}_0)$$

$$\hat{\mu}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \quad \hat{\Sigma}_W = \sum_k \sum_{i \in C_k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

# Nonlinear Level Set Group Symmetries



- Curse of dimensionality  $\Rightarrow$  not local but global geometry  
Level sets: classes, characterised by their global symmetries.



- A symmetry is an operator  $g$  which preserves level sets:

$$\forall x \ , \ f(g.x) = f(x) : \text{global}$$

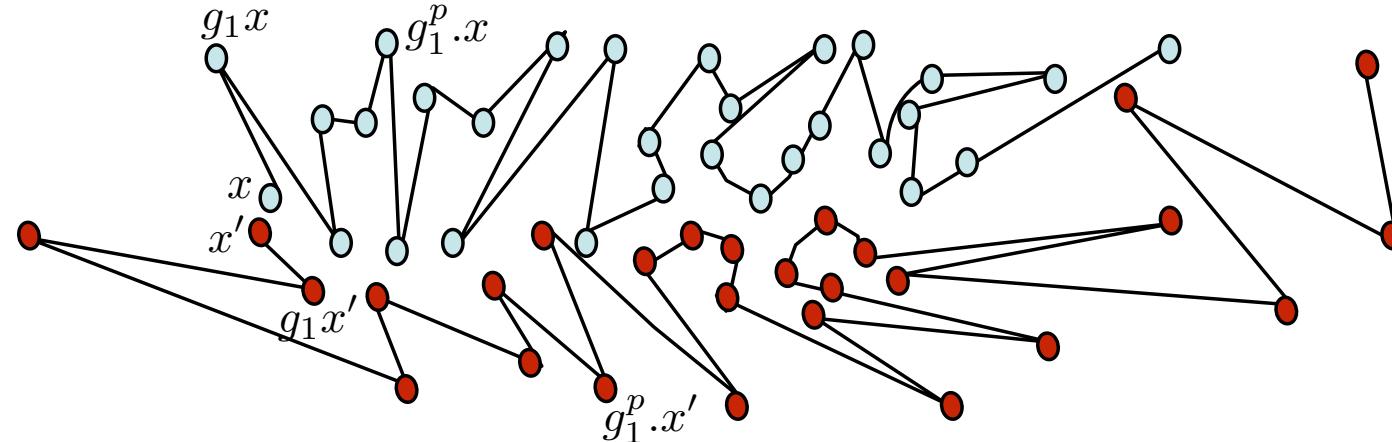
If  $g_1$  and  $g_2$  are symmetries then  $g_1.g_2$  is also a symmetry

$$f(g_1.g_2.x) = f(g_2.x) = f(x)$$

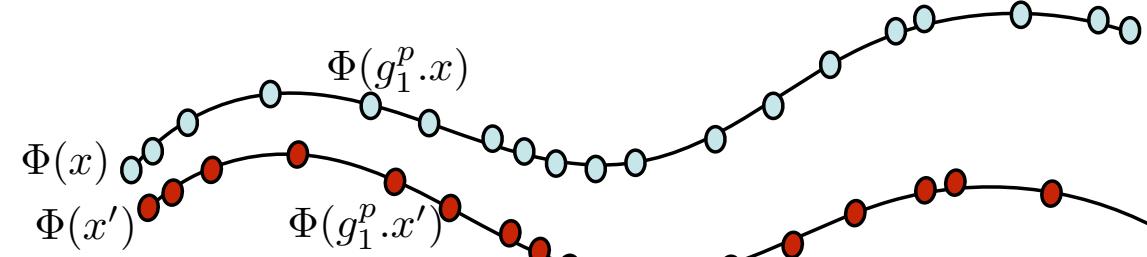
Level set symmetries lead to groups...

# Linearize Symmetries

- A change of variable  $\Phi(x)$  must linearize the orbits  $\{g.x\}_{g \in G}$



- Linearise symmetries with a change of variable  $\Phi(x)$



- Lipschitz:  $\forall x, g : \|\Phi(x) - \Phi(g.x)\| \leq C \|g\|$

# Wavelet Scattering Net

Stephane Mallat et al. 2012

- Architecture:

- Convolutional filters: band-limited complex wavelets
- Nonlinear activation: modulus (Lipschitz)
- Pooling: averaging (L1)

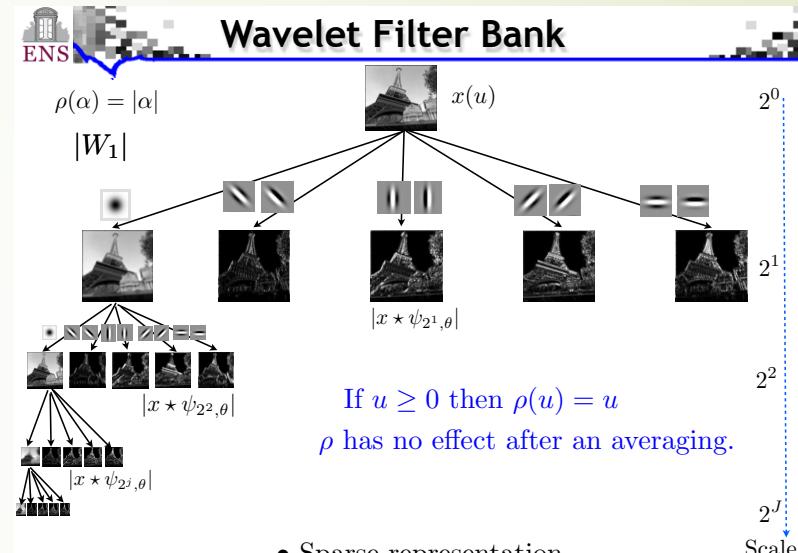
- Properties:

- A Multiscale Sparse Representation
- Norm Preservation (Parseval's identity):

$$\|Sx\| = \|x\|$$

- Contraction:

$$\|Sx - Sy\| \leq \|x - y\|$$



- Sparse representation

$$Sx = \begin{pmatrix} x * \phi(u) \\ |x * \psi_{\lambda_1}| * \phi(u) \\ ||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \phi(u) \\ |||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \psi_{\lambda_3}| * \phi(u) \\ \dots \end{pmatrix}_{u, \lambda_1, \lambda_2, \lambda_3, \dots}$$

# Invariants/Stability of Scattering Net

## ► Translation Invariance (generalized to **rotation** and **scaling**):

- The average  $|x \star \psi_{\lambda_1}| \star \phi(t)$  is invariant to small translations relatively to the support of  $\phi$ .
- Full translation invariance at the limit:

$$\lim_{\phi \rightarrow 1} |x \star \psi_{\lambda_1}| \star \phi(t) = \int |x \star \psi_{\lambda_1}(u)| du = \|x \star \psi_{\lambda_1}\|_1$$

## ► Stable Small Deformations:

*stable to deformations*  $x_\tau(t) = x(t - \tau(t))$

$$\|Sx - Sx_\tau\| \leq C \sup_t |\nabla \tau(t)| \|x\|$$

# Wiatowski-Bolcskei'15



- ▶ Scattering Net by Mallat et al. so far
  - ▶ Wavelet Linear filter
  - ▶ Nonlinear activation by modulus
  - ▶ Average pooling
- ▶ Generalization by [Wiatowski-Bolcskei'15](#)
  - ▶ Filters as frames
  - ▶ Lipschitz continuous Nonlinearities
  - ▶ General Pooling: Max/Average/Nonlinear, etc.
  - ▶ As depth grows, the multiplicative pooling factors leads to full invariances.

**Filters:** Semi-discrete frame  $\Psi_n := \{\chi_n\} \cup \{g_{\lambda_n}\}_{\lambda_n \in \Lambda_n}$

$$A_n \|f\|_2^2 \leq \|f * \chi_n\|_2^2 + \sum_{\lambda_n \in \Lambda_n} \|f * g_{\lambda_n}\|^2 \leq B_n \|f\|_2^2, \quad \forall f \in L^2(\mathbb{R}^d)$$

**Pooling:** In continuous-time according to

$$f \mapsto S_n^{d/2} P_n(f)(S_n \cdot),$$

where  $S_n \geq 1$  is the **pooling factor** and  $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$  is  $R_n$ -Lipschitz-continuous

*Assume that the filters, non-linearities, and poolings satisfy*

$$B_n \leq \min\{1, L_n^{-2} R_n^{-2}\}, \quad \forall n \in \mathbb{N}.$$

*Let the pooling factors be  $S_n \geq 1$ ,  $n \in \mathbb{N}$ . Then,*

$$|||\Phi^n(T_t f) - \Phi^n(f)||| = \mathcal{O}\left(\frac{\|t\|}{S_1 \dots S_n}\right),$$

*for all  $f \in L^2(\mathbb{R}^d)$ ,  $t \in \mathbb{R}^d$ ,  $n \in \mathbb{N}$ .*

# Summary

- ▶ All these works partially explains the success of CNNs
  - ▶ Contraction within level set symmetries toward invariance when depth grows (invariants)
  - ▶ Separation kept between different levels (discriminant)
- ▶ Other questions?
  - ▶ Can one adaptively learn some networks with the same invariant properties as the scattering net?
  - ▶ How deep networks generalize well without overfitting?
  - ▶ What's the landscape of empirical risks and how to efficiently optimize?

Thank you!

