Deep Learning toward Deeper Understanding

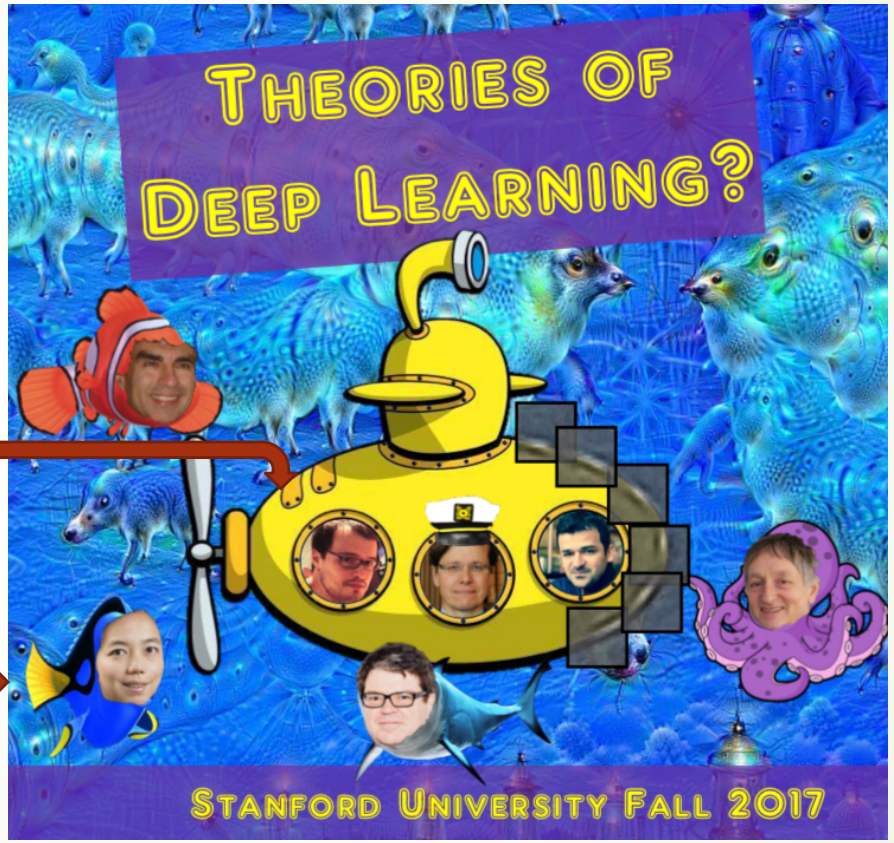
Yuan YAO

HKUST

Acknowledgement

https://stats385.github.io/

http://cs231n.github.io/

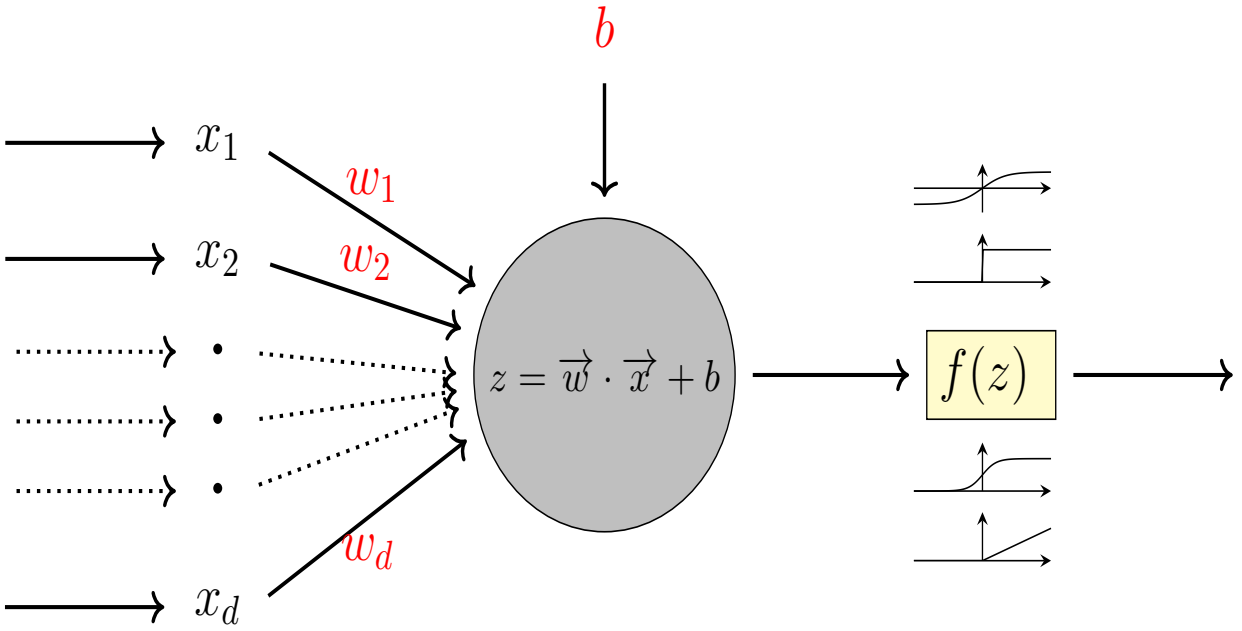


A following-up course at HKUST: https://deeplearning-math.github.io/

A Brief History of Neural Networks

Perceptron: single-layer

- Invented by Frank Rosenblatt (1957)



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^∞, etc) function of n variables be represented as a superposition of continuous (analytic, C^∞, 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 reprensented as

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

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 ϕ_{pq} to be Hölder or Lipschitz continuous, but not C^1 (Fridman, 1967).
- Can choose $\phi_{pq} = \lambda_p \phi_q$ where $\lambda_1, \cdots, \lambda_n > 0$ 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**.

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

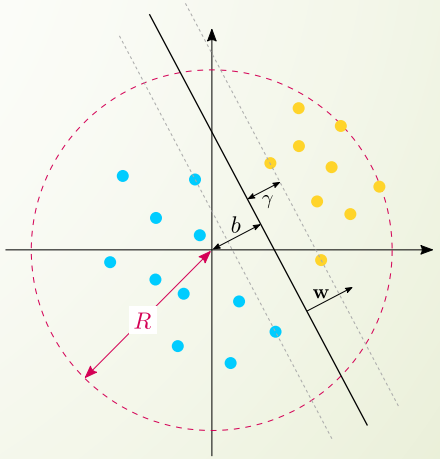
The perceptron convergence theorem was proved by Block (1962) and Novikoff (1962). The following version is based on that in Cristianini and Shawe-Taylor (2000).

Theorem 1 (Block, Novikoff). *Let the training set* $S = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ *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*

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

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

Margin: $\gamma := \min_i y_i f(x_i)$



Locality or Sparsity of Computation

Minsky and Papert, 1969

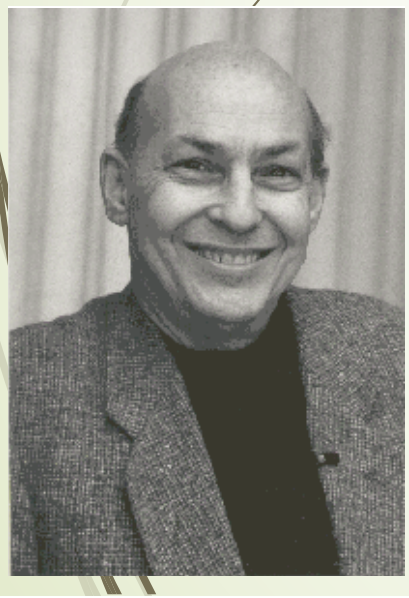
Perceptron can't do **XOR** classification

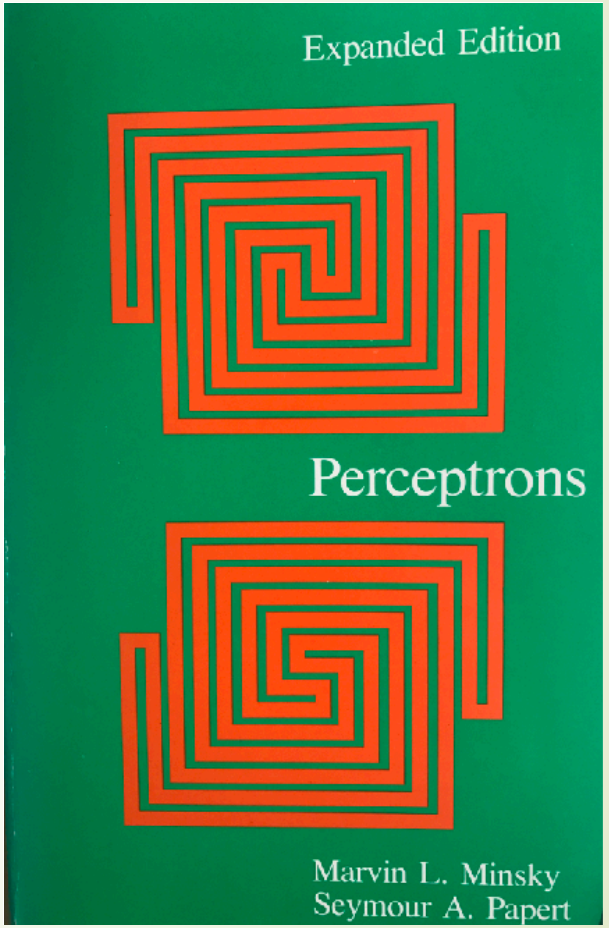
Perceptron needs infinite global information to compute **connectivity**

Locality or **Sparsity** is important:

Locality in time?

Locality in space?



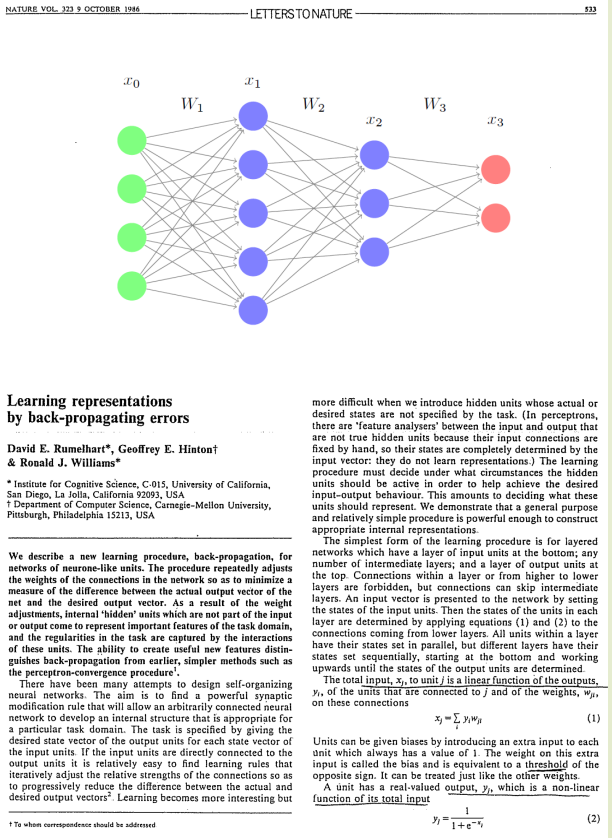


Multilayer Perceptrons (MLP) and Back-Propagation (BP) Algorithms

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

MLP classifies **XOR**, but the global hurdle on topology (connectivity) computation still exists: 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, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-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 come 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-convergence procedure[1].

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[2]. 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 internal 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_{ji}, on these connections

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

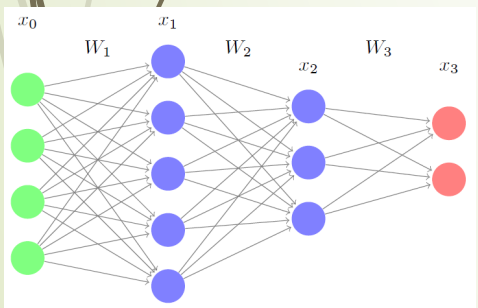
Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

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: Forward Pass



- 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: $\quad x_\ell = f_\ell(W_\ell x_{\ell-1} + b_\ell)$
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 $(I_i, y_i) \equiv$ (input,target) and L layers

- Constrained optimization

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

- Lagrangian formulation (Unconstrained)

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

$$\mathcal{L}(W, x, B) = \textstyle\sum_{i=1}^{n} \left\{ \|x_i(L) - y_i\|_2^2 + \right.$$

$$\left. \textstyle\sum_{\ell=1}^{L} B_i(\ell)^T \left(x_i(\ell) - f_\ell\Big[W_\ell x_i\,(\ell-1)\Big] \right) \right\}$$

http://yann.lecun.com/exdb/publis/pdf/lecun-88.pdf

back-propagation – derivation

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

Forward pass

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

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

Backward (adjoint) pass

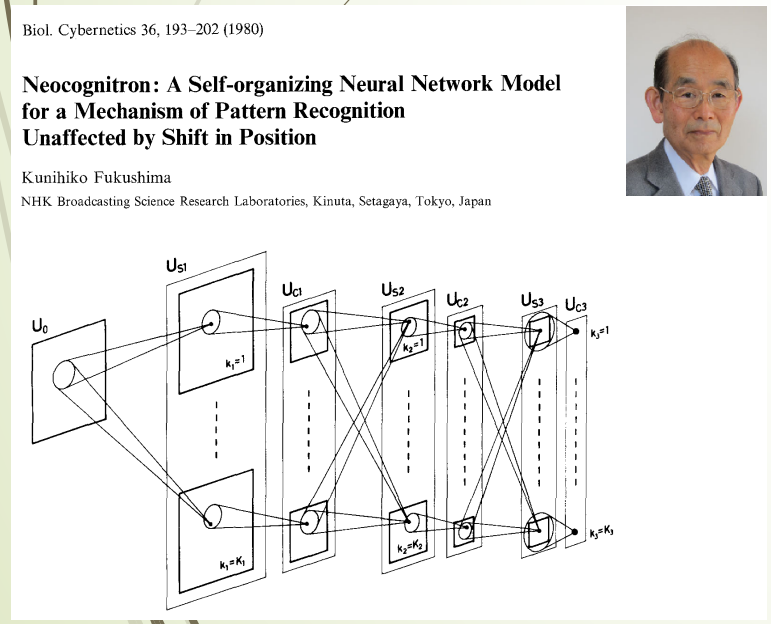
$$z(L) = 2\nabla f_L\Big[A_i(L)\Big]\,(y_i - x_i(L))$$
$$z_i(\ell) = \nabla f_\ell\Big[A_i(\ell)\Big] W_{\ell+1}^T z_i(\ell+1) \quad \ell = 0,\ldots,L-1$$

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

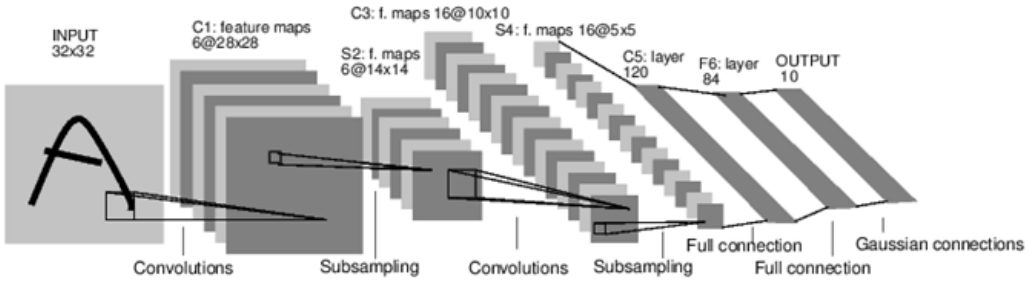
Weight update

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

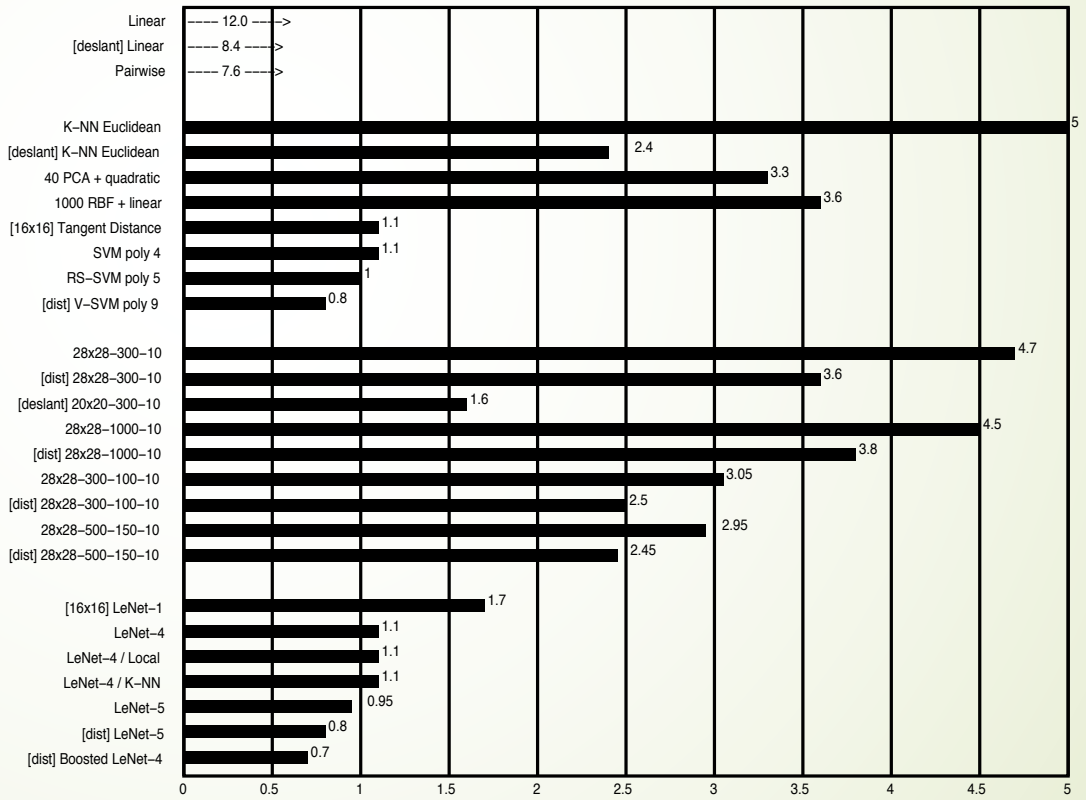
Convolutional Neural Networks: shift invariances and locality



- 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



MNIST Dataset Test Error
LeCun et al. 1998

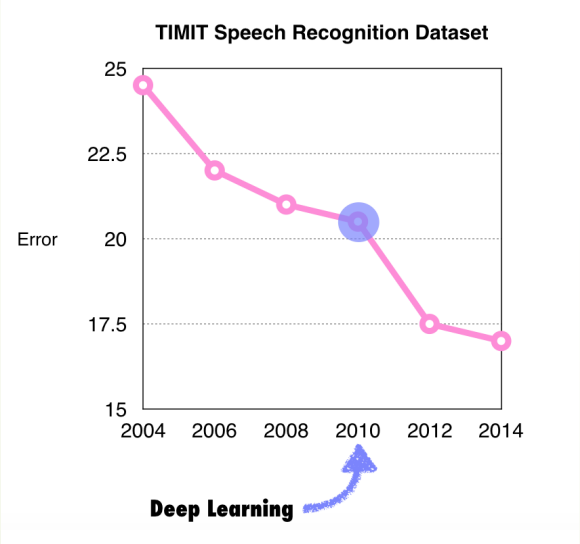


Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets)

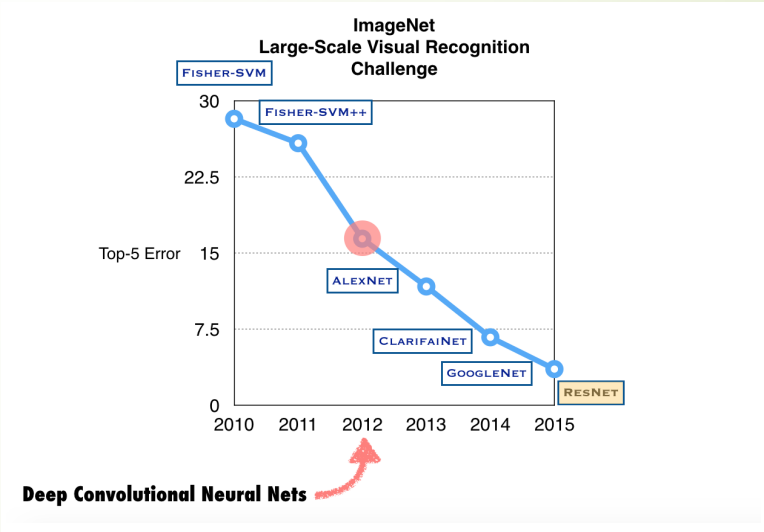
Dark era for NN: 1998-2012

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

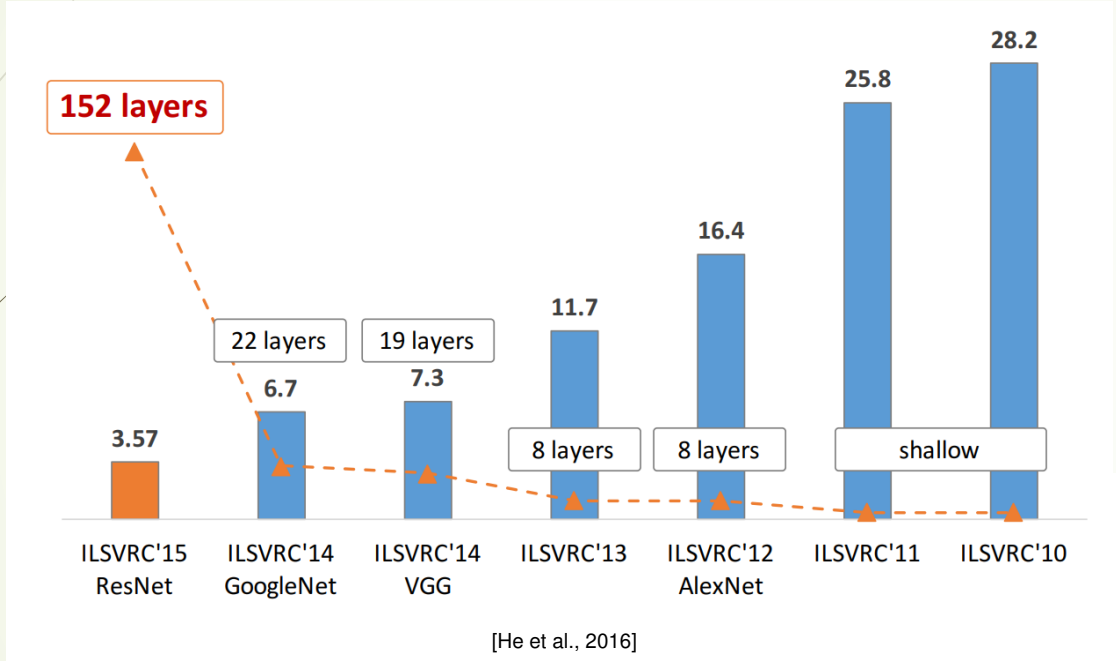
Speech Recognition: TIMIT



Computer Vision: ImageNet



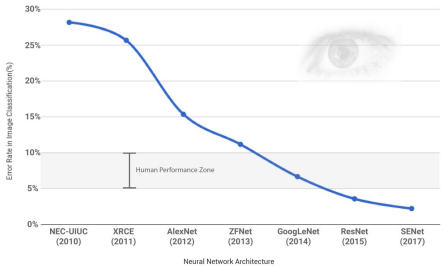
Depth as function of year



[He et al., 2016]

ILSVRC ImageNet Top 5 errors

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

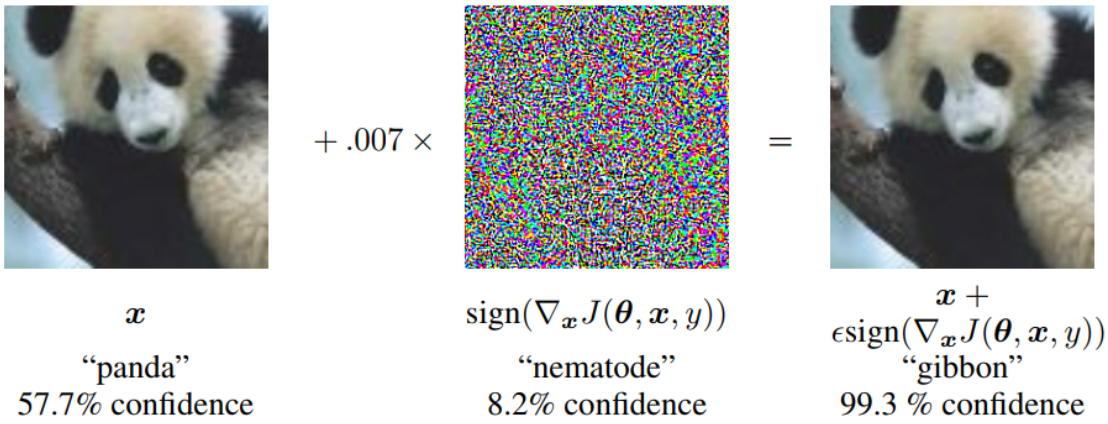


source: https://www.linkedin.com/pulse/must-read-path-breaking-papers-image-classification-muktabh-mayank

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!



[Goodfellow et al., 2014]

- Small but malicious perturbations can result in severe misclassification
- Malicious examples generalize across different architectures
- What is source of instability?
- Can we robustify network?

What's wrong with deep learning?

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

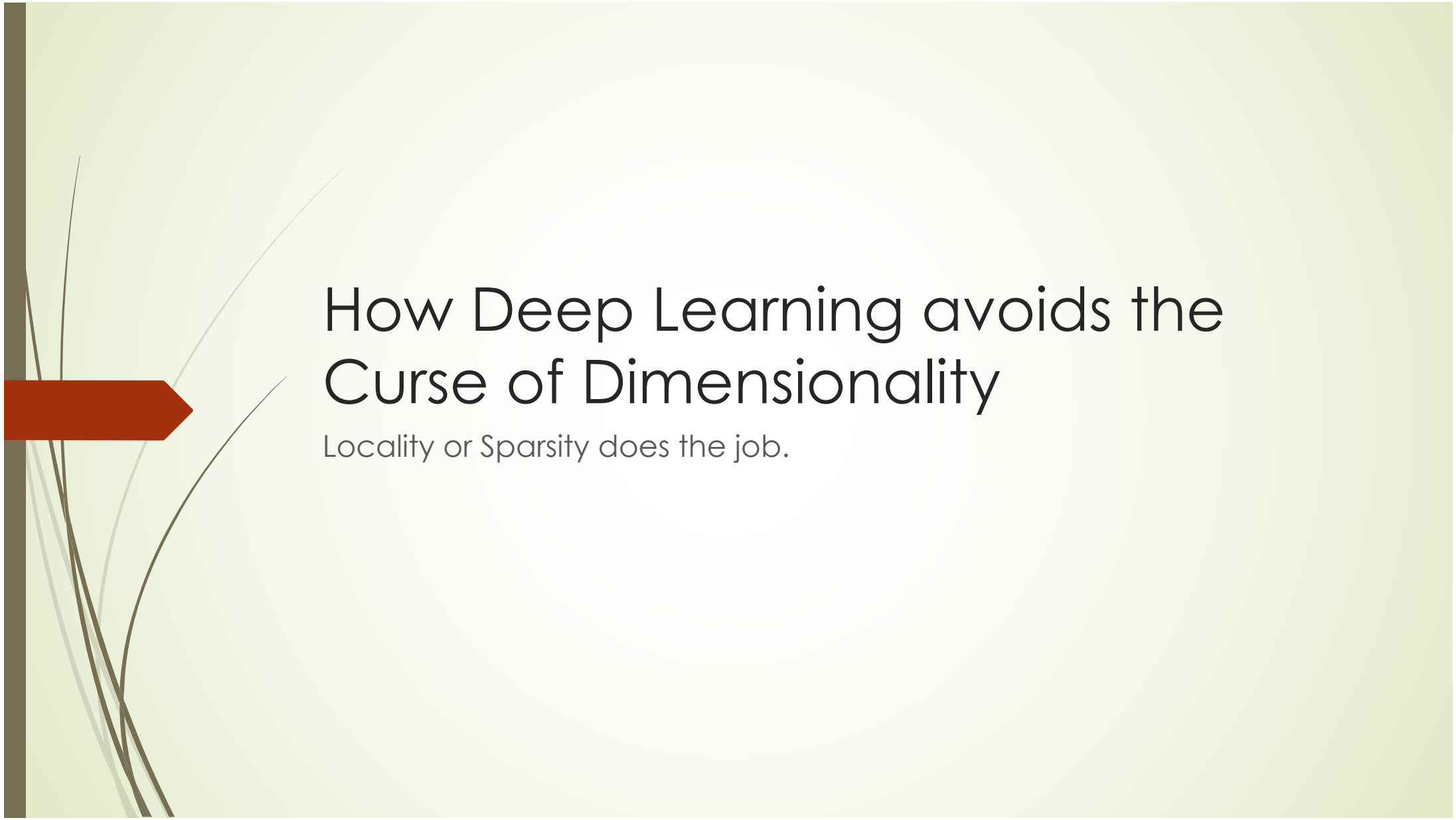
Yann LeCun CVPR'15, invited talk: **What's wrong with deep learning?**
One important piece: **missing some theory**!

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**

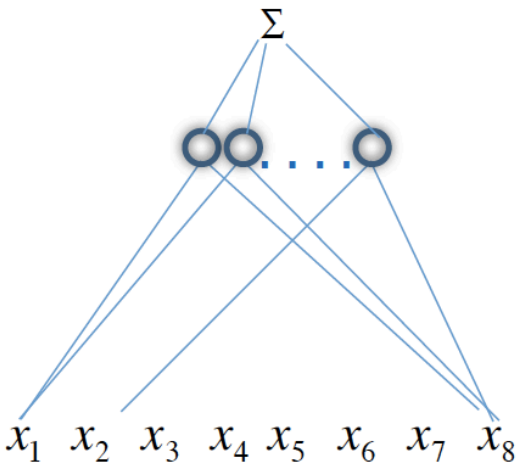
Some Theoretical Problems

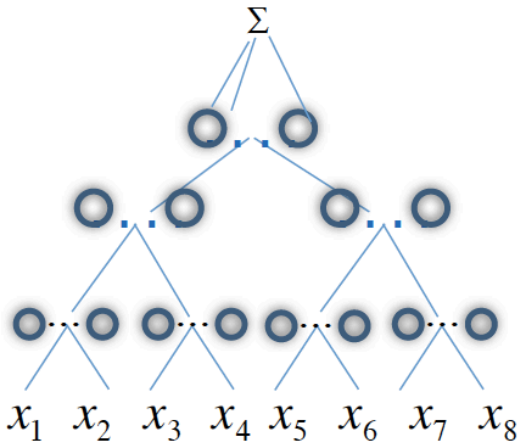
- *Approximation Theory and Harmonic Analysis :* **What functions are represented well by deep neural networks, without suffering the curse of dimensionality and better than shallow networks?**
  - Sparse (local), hierarchical (multiscale), compositional functions avoid the curse dimensionality
  - Group (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


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.*





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

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

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

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 $O(\varepsilon^{-d})$.

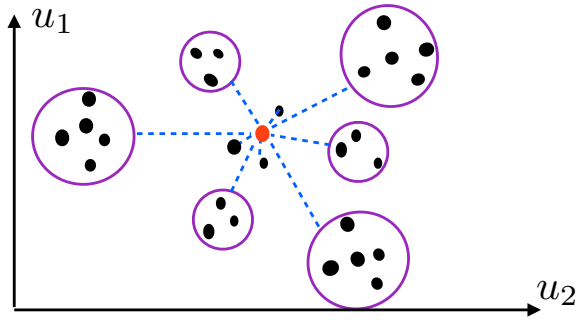
CENTER FOR Brains Minds + Machines

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...

- Mutliscale interactions of d variables:



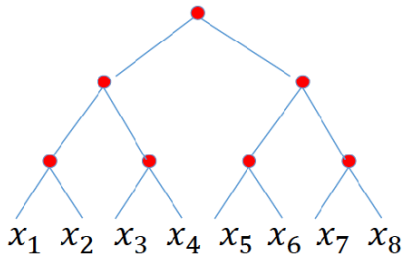
From d^2 interactions to $O(\log^2 d)$ multiscale interactions.

(Or even of constant numbers.)

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

Hierarchically local compositionality

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



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 $O(d\varepsilon^{-2})$

Mhaskar, Poggio, Liao, 2016

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

Important Special Cases in Statistics

Minimax rates of estimations (Stone 1982): if a regression function f is Lipschitz on $\mathbb{R}^d$ α 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): $f(x_1, \ldots, x_d) = f_1(x_1) + \ldots + f_d(x_d)$ with minimax rate $N^{-\frac{2\alpha}{2\alpha+1}}$.
Raskutti-Wainwright-Yu (IEEE TIT, 2011), Yuan-Zhou (AoS, 2016), ...

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

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

Projection pursuit (Friedman and Stuetzle 1981): $f(x_1, \ldots, x_d) = \sum_{k=1}^{K} g_k(a_k \cdot x)$ 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: $f = g(f_1(x_{I_1}), f_2(x_{I_2}), \ldots, f_{d^*}(x_{I_{d^*}}))$

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

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

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

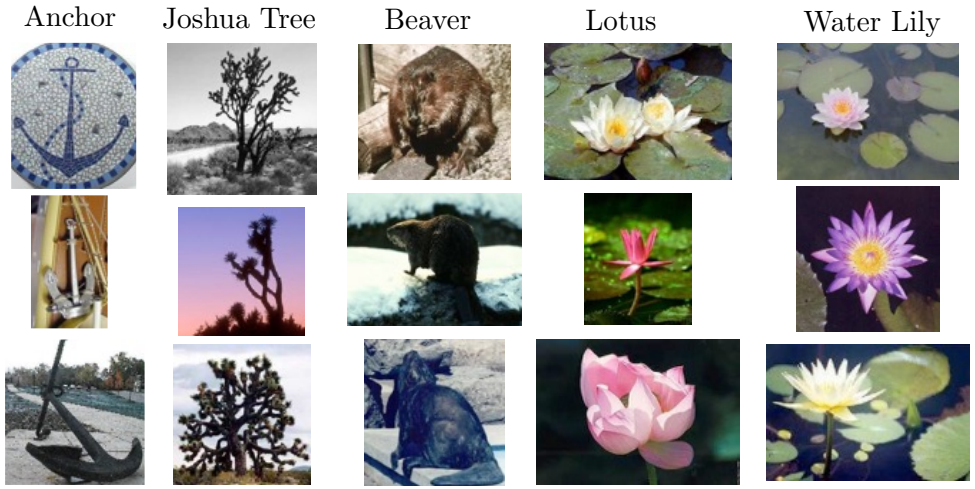
What are the geometric properties of deep networks?

Contraction within-class level sets toward invariants when depth grows, while keeping the separation between classes

High Dimensional Natural Image Classification

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

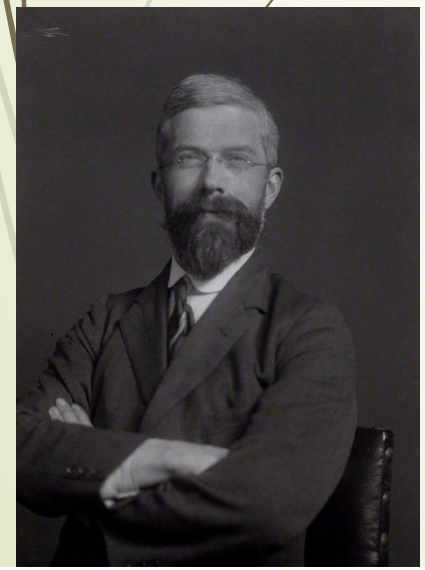
Image Classification $d = 10^6$



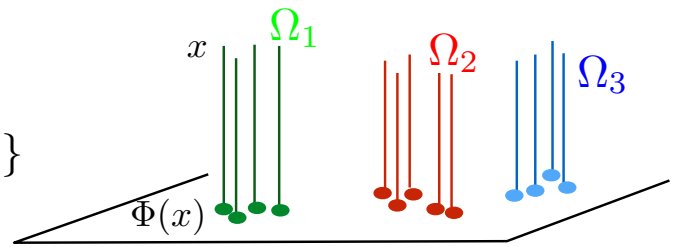
Huge variability inside classes

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.*

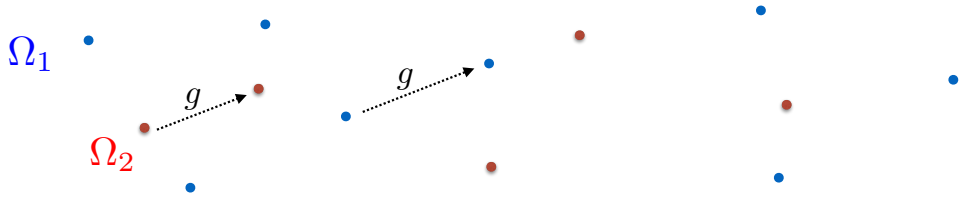
$$\Phi(x) = \alpha \hat{\Sigma}_W^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$$

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

Nonlinear Level Set Group Symmetries

Level Set Geometry: 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:

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

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

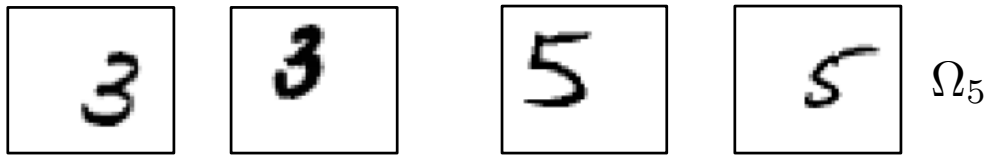
$$f(g_1.g_2.x) = f(g_2.x) = f(x)$$

Level set symmetries lead to groups…

Translation and Deformations

- Digit classification:

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

Ω_3  Ω_5

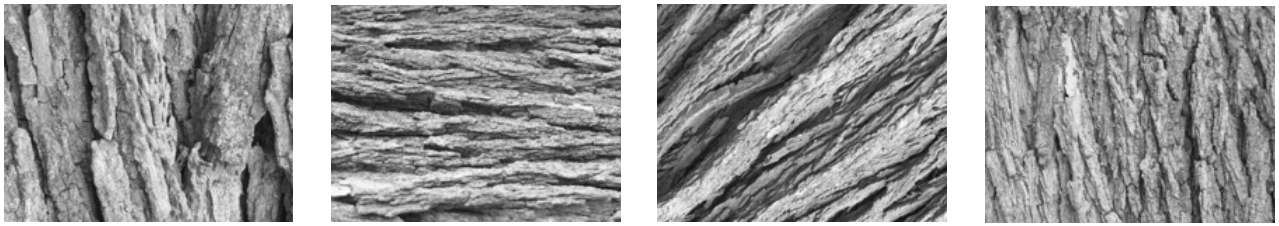
- Globally invariant to the translation group: small
- Locally invariant to small diffeomorphisms: huge group

Video of Philipp Scott Johnson

https://www.youtube.com/watch?v=nUDIoN-_Hxs

Rotation and Scaling Variability

- Rotation and deformations



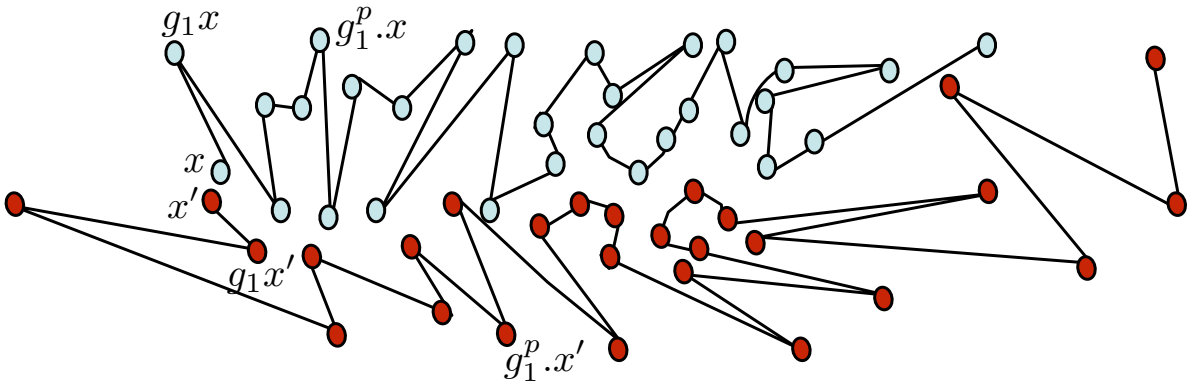
Group: $SO(2) \times \mathrm{Diff}(SO(2))$

- Scaling and deformations

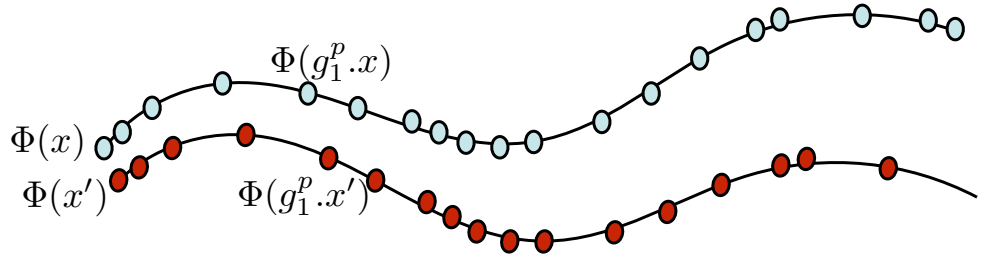
Group: $\mathbb{R} \times \mathrm{Diff}(\mathbb{R})$

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

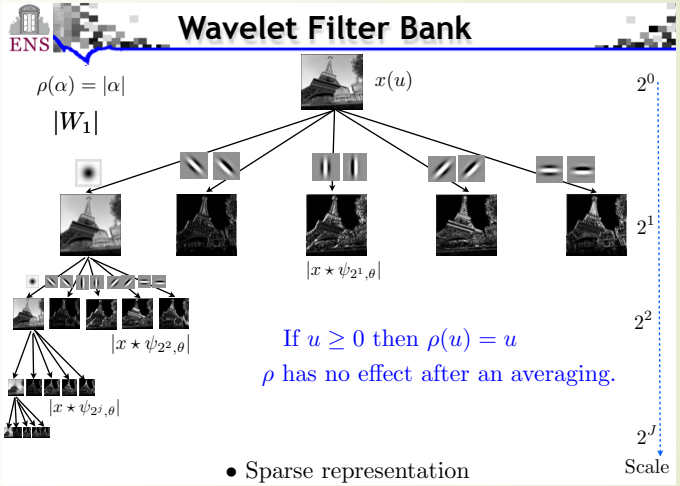
Stephane Mallat et al. 2012

- Architechture:
  - 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:

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



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

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 ϕ.
  - Full translation invariance at the limit:

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

- 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 $P_n : L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ 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?
  - How deep networks generalize well without overfitting?
  - What's the landscape of empirical risks and how to efficiently optimize?