Statistical Machine Learning

Yuan YAO

HKUST

Course Infomation

- Course web:
  - https://yao-lab.github.io/course/statml/2022/
- Time and Venure:
  - Lecture: **MonWed, 10:30-11:50am**
    - **Zoom** Meetings from CANVAS
    - **Or** Rm 2503, Lift 25-26 (87)
- Instructor:
  - Yuan Yao <yuany@ust.hk> (https://yao-lab.github.io/)
- Teaching Assistant:
  - ???

Course Content

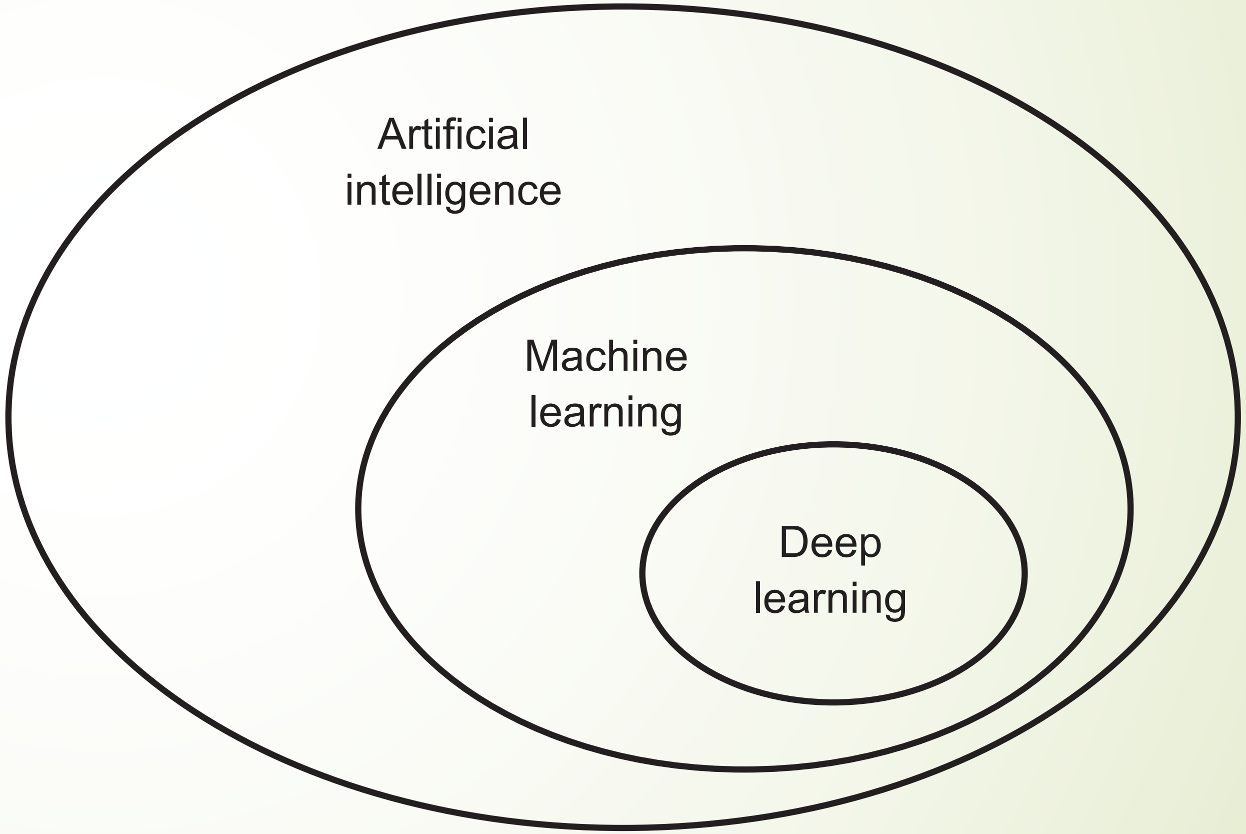
- Supervised Learning:
  - working knowledge about linear regression, classification, logistic regression, decision trees (CART), boosting, random forests, support vector machines, neural networks, etc.
- Unsupervised and Self-supervised Learning:
  - PCA, Generative Models, Generative Adversarial Networks
  - Self-supervision, e.g. masked language models etc.
- Reinforcement Learning:
  - Markov Decision Process and online learning, etc.

- **No exams. Project-based evaluation.**

A Brief History of AI, Machine Learning, and Deep Learning

Artificial Intelligence, Machine Learning, and Deep Learning

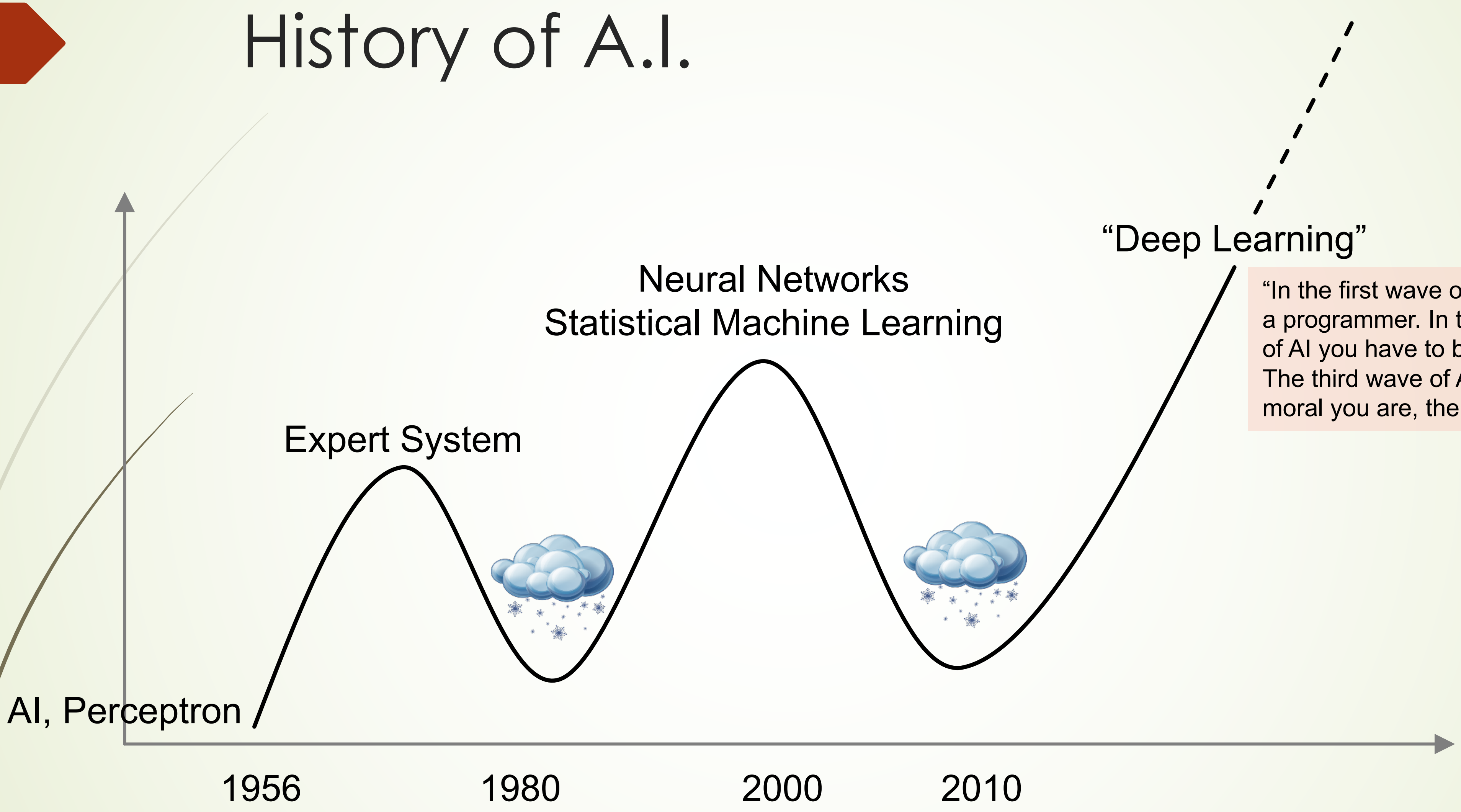
- AI is born in 1950s, when a handful of pioneers from the nascent field of computer science started asking **whether computers could be made to "think"**—a question whose ramifications we're still exploring today.



A brief history of AI

- **1943:** McCulloch & Pits proposed a boolean circuit model of neurons
- **1949:** Donald Hebb proposed **Hebbian learning rule**.
- **1950:** Alan Turing published **"Computing Machinery and Intelligence"** with **Turing test**.
- **1956:** John McCarthy at the Dartmouth Conference coined terminology "**Artificial Intelligence**"
- **1957**: Rosenblatt invented **Perceptron**
- **1960s:** golden years till **1969 Minsky-Papert's** critical book **Perceptron**
- **1970s**: the first AI winter
- **1980s**: boom of AI with **Expert System**
- **1990s:** the second AI winter, rise of **statistical machine learning**
- **1997: IBM Deep Blue** beats world chess champion Kasparov
- **2012:** return of neural networks as **deep learning** (speech, ImageNet in computer vision, NLP, …)
- **2016-2017: Google AlphaGo "Lee" and Zero**
- **2020: Google AlphaFold**
- …

History of A.I.

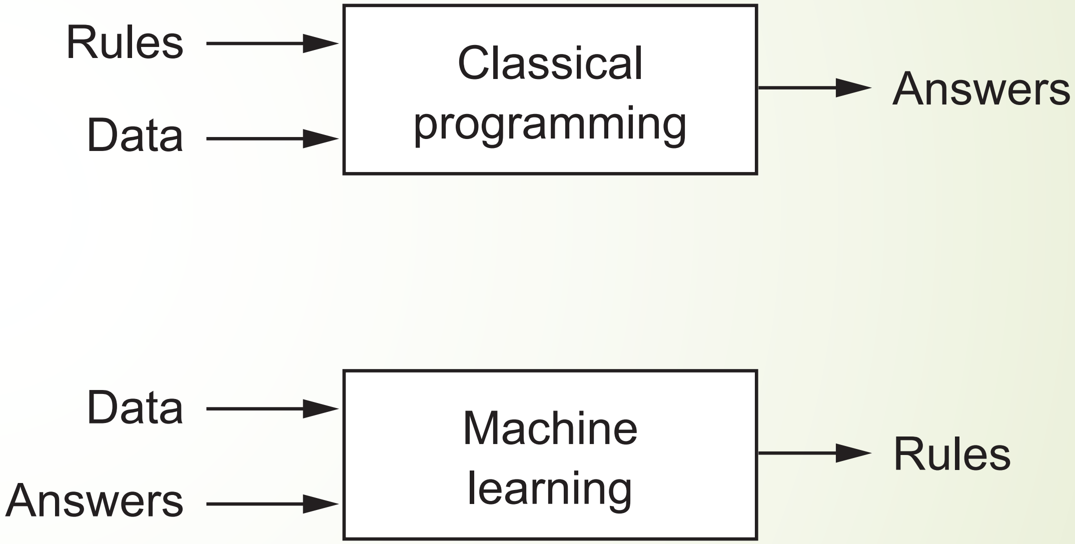


“In the first wave of AI you had to be a programmer. In the second wave of AI you have to be a data scientist. The third wave of AI—the more moral you are, the better.”

Source: Dr. Lee Kai-Fu • Sinovation Ventures and 2018 Conference on Neural Information Processing Systems

Statistical Machine Learning is a new paradigm of computer programming

- During 1950s-1980s, two competitive ideas of realizing AI exist
  - Rule based inference, or called **Expert System**
  - Statistics based inference, or called **Machine Learning**
- 1990s- Machine Learning becomes dominant



Rules

Data

Classical programming

Answers

Data

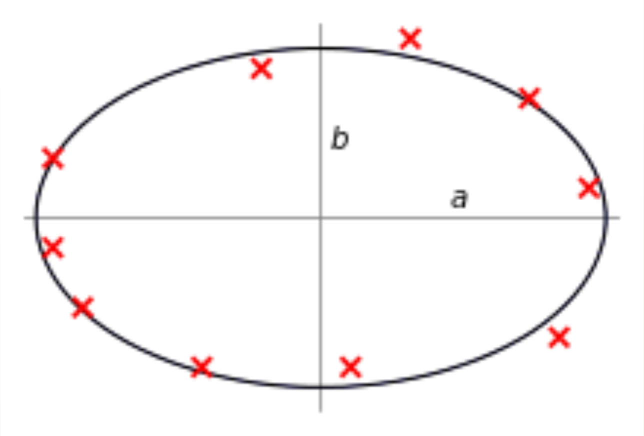
Answers

Machine learning

Rules

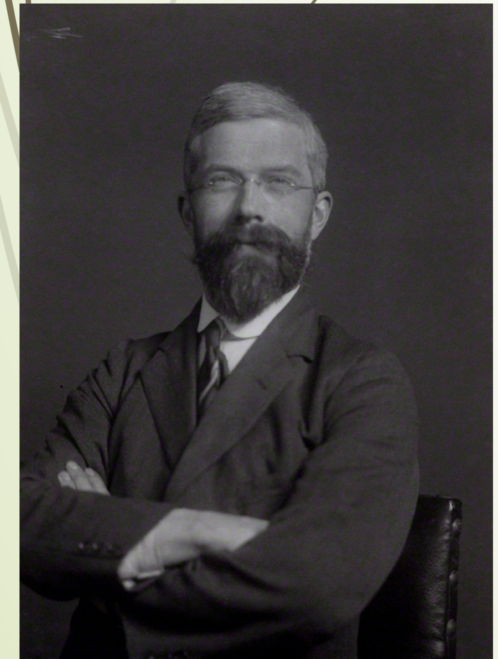
The 1st machine learning method: Least Squares

- Invention:
  - **Carl Friederich Gauss** (~1795/1809/1810),
  - Adrien-Marie Legendre (1805)
  - Robert Adrain (1808)
- Application:
  - Prediction of the location of asteroid Ceres after it emerged from behind the sun (Franz Xaver von Zach 1801)
  - Orbits of planets, Newton Laws
  - Statistics,
  - …



Fisher’s Maximum Likelihood Principle (1912-1922)

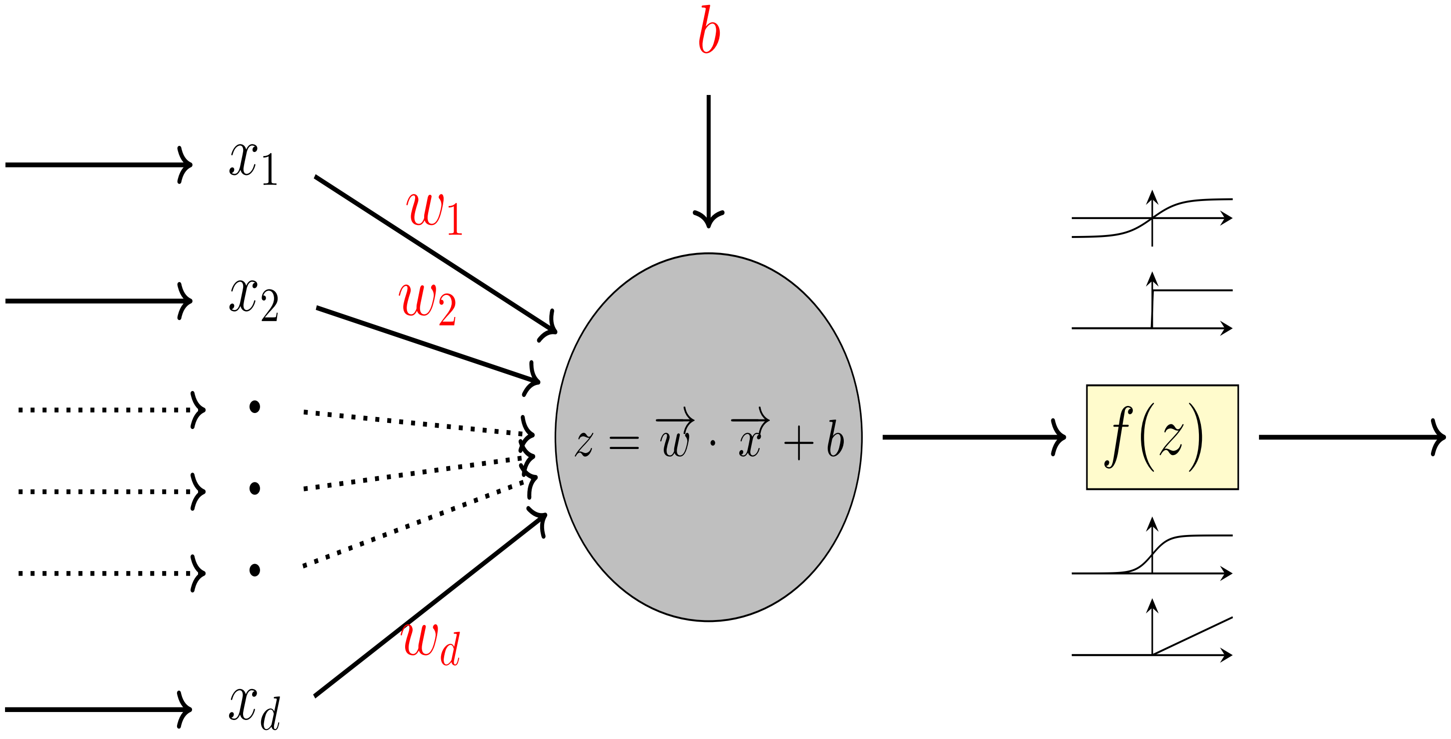
- **The least square method is the maximum likelihood estimate** (most probable values of the unknown parameters) when the noise is Gaussian.



- Fisher, R. A. (1912) **On an absolute criterion for fitting frequency curves**. *Messenger of Mathematics* 41:155-160.
- Fisher, R. A. (1922). **On the mathematical foundations of theoretical statistics**. *Philos. Trans. Roy. Soc. London Ser. A* 222:309-368.
- Aldrich, John (1997). **R. A. Fisher and the Making of Maximum Likelihood 1912 -- 1922**. *Statistical Science*, 12(3):162-176.

The 1st neural network: Perceptron

- Invented by Frank Rosenblatt (1957)



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**, *Ann. Math. Statist*. 22(3): 400-407):

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

Finiteness of Stopping Time and Margin

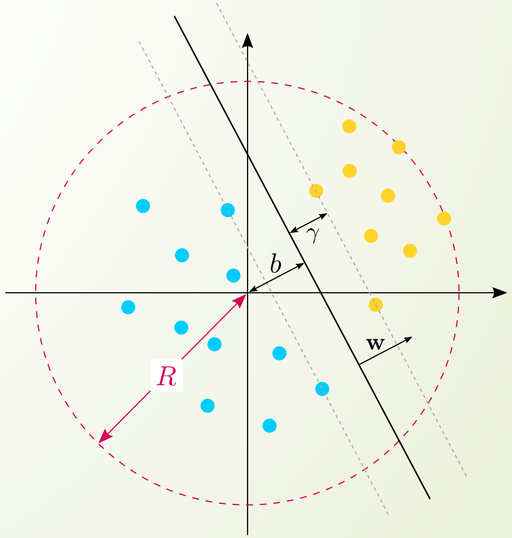
The perceptron convergence theorem was proved by Block (1962) and Novikoff (1962). The following version is based on that in Cristianini and Shawe-Taylor (2000).

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



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

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

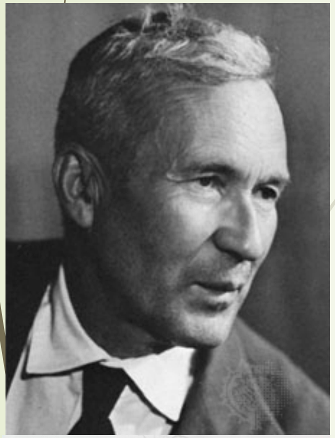
Hilbert's conjecture: $x(a, b, c)$ cannot be expressed by a superposition (sums and compositions) of bivariate functions.

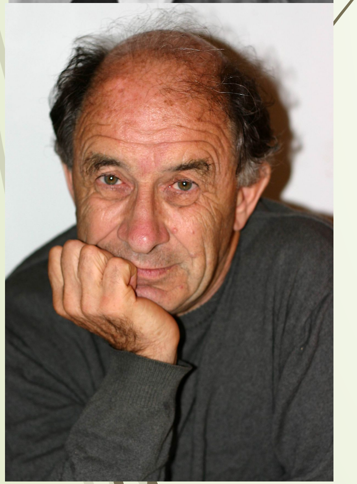
Question: can every continuous (analytic, C^∞, etc) function of n variables be represented as a superposition of continuous (analytic, C^∞, etc) functions of $n-1$ variables?

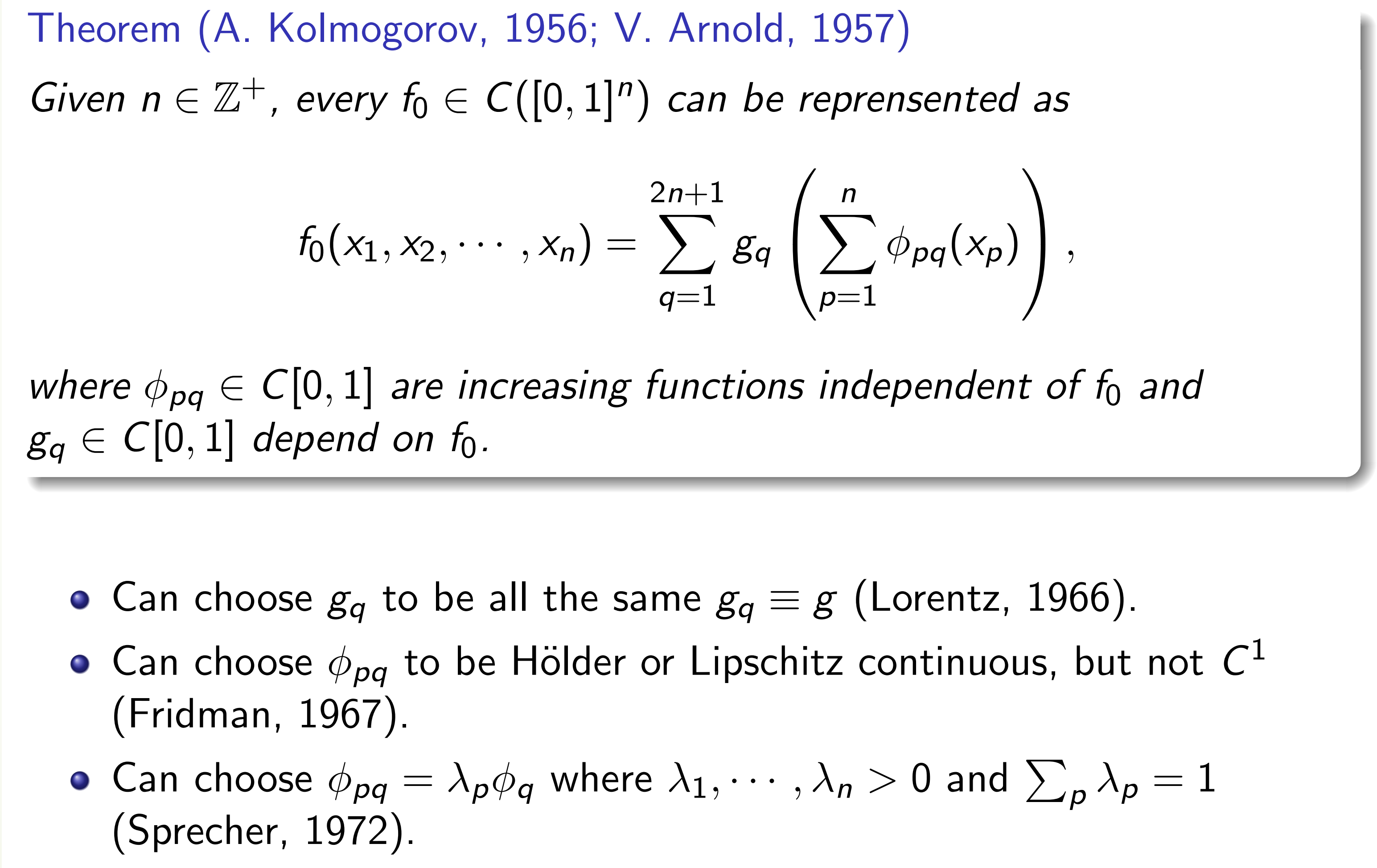
Theorem (D. Hilbert)

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**.

Kolmogorov’s Exact Representation is not stable or smooth

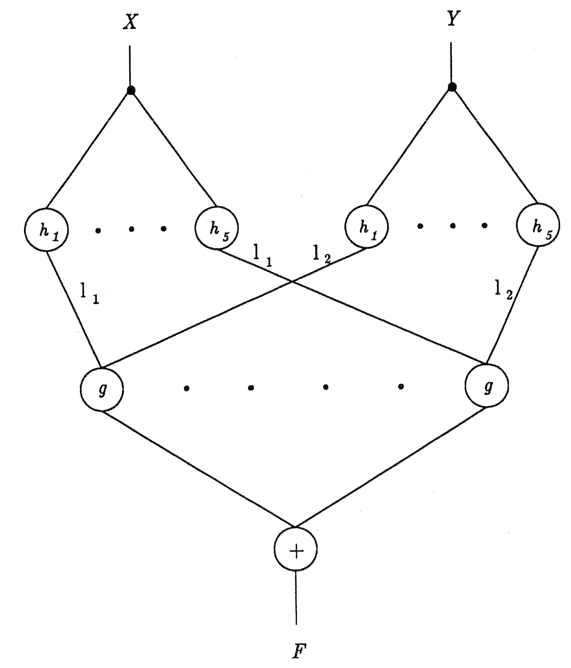


Figure 1: The network representation of an improved version of Kolmogorov’s theorem, due to Kahane (1975). The figure shows the case of a bivariate function. The Kahane’s representation formula is $f(x_1,\dots,x_n) = \sum_{q=1}^{2n+1} g[\sum_{p=1}^{n} l_p h_q(x_p)]$ where h_q are strictly monotonic functions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

Universal Approximate Representation

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

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

$$\begin{aligned} F(x) &= \alpha^\top \sigma(Wx + \beta) \\ &= \sum_i \alpha_i \sigma \left(\sum_j W_{i,j}\, x_j + \beta_i \right) \end{aligned}$$

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

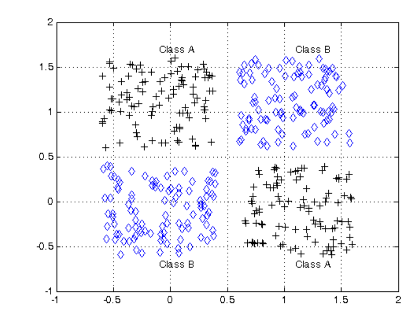
Complexity (regularity, smoothness) thereafter becomes the central pursuit in Approximation Theory.

Locality or Sparsity of Computation

Minsky and Papert, 1969

Perceptron can't do **XOR** classification

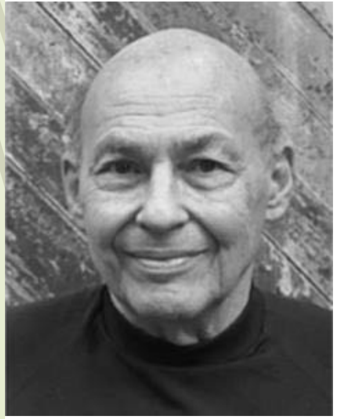
Perceptron needs infinite global information to compute **connectivity**



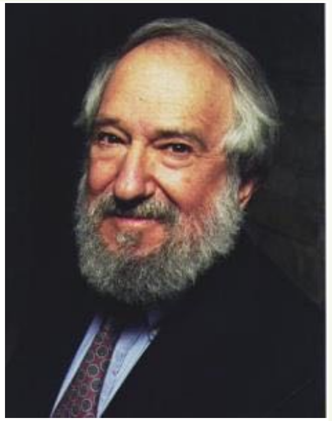
Locality or **Sparsity** is important:

Locality in time?

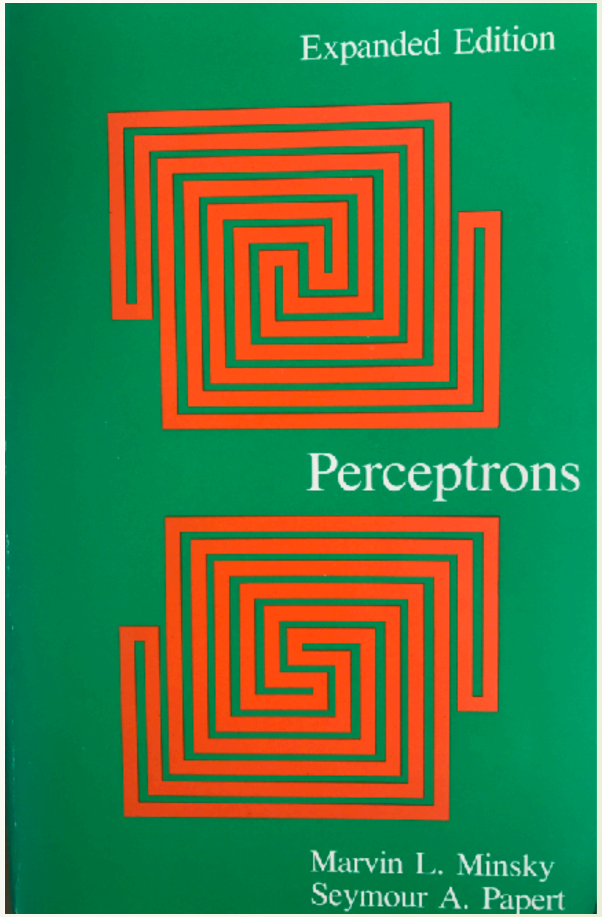
Locality in space?



Marvin Minsky



Seymour Papert



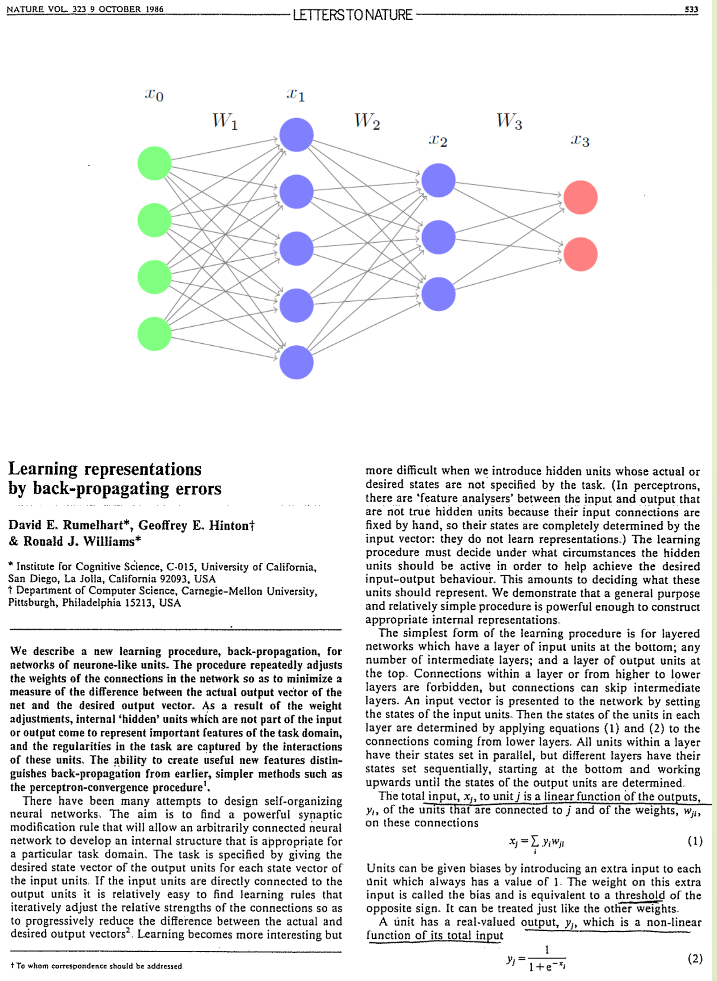
Multilayer Perceptrons (MLP) and Back-Propagation (BP) Algorithms

D.E. Rumelhart, G. Hinton, R.J. 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

Deep network may classify **XOR**. Yet **topology**?

We address complexity and geometric invariant properties first.


x_0
W_1
x_1
W_2
x_2
W_3
x_3

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

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

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.

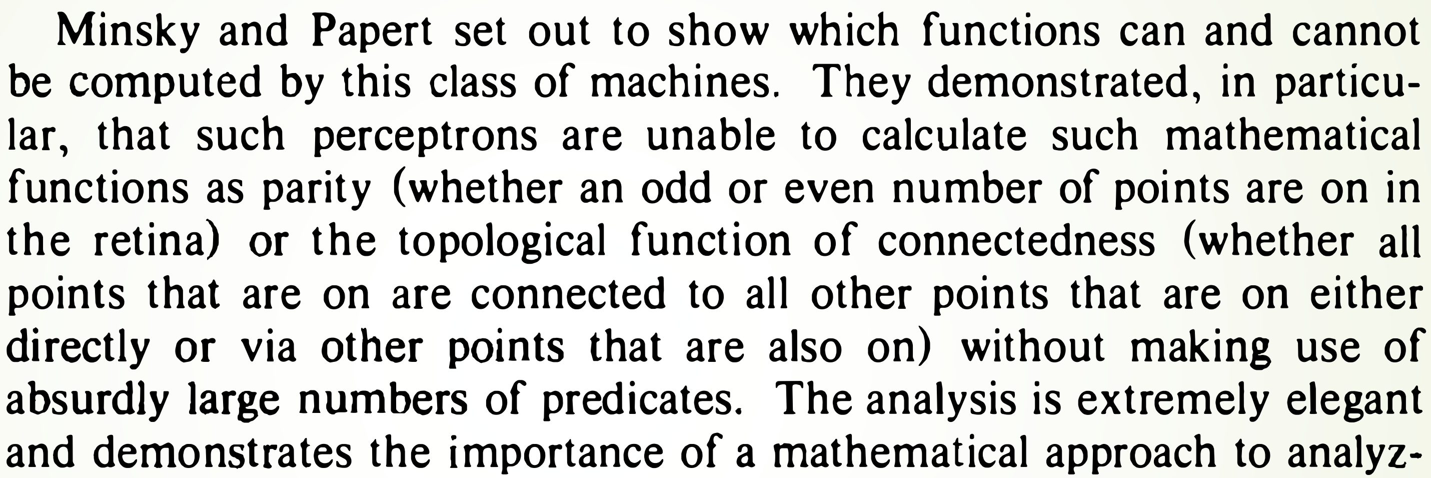
A unit has a real-valued output, y_j, which is a non-linear function of its total input

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

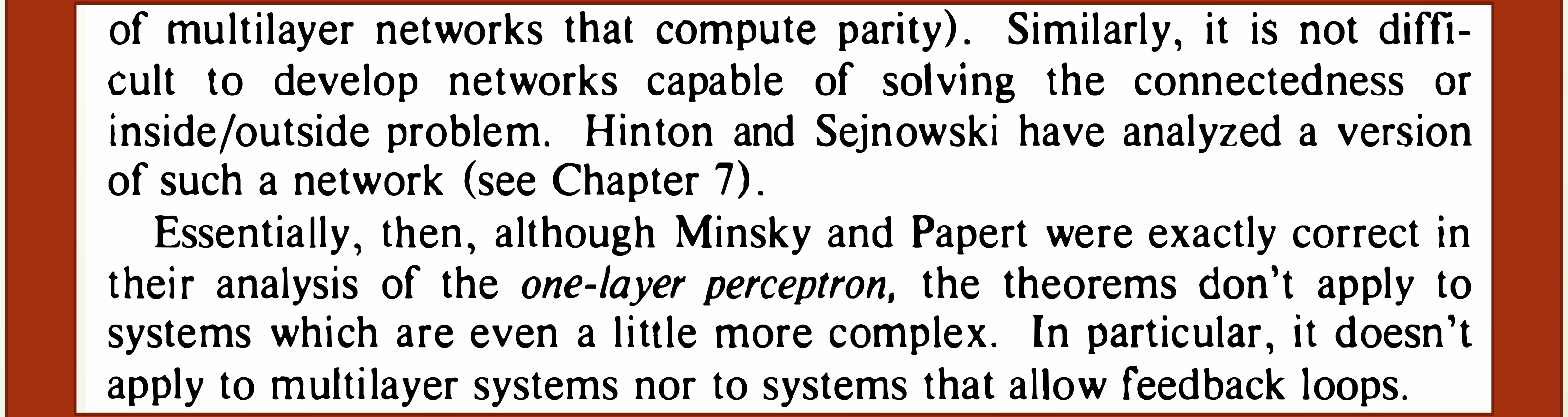
† To whom correspondence should be addressed.

Parallel Distributed Processing

by Rumelhart and McClelland, 1986

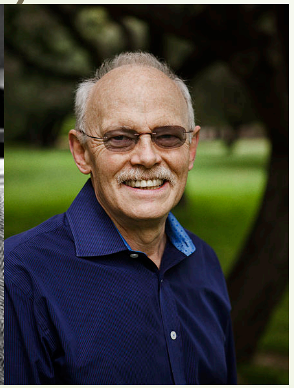


Minsky and Papert set out to show which functions can and cannot be computed by this class of machines. They demonstrated, in particular, that such perceptrons are unable to calculate such mathematical functions as parity (whether an odd or even number of points are on in the retina) or the topological 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 analyz-



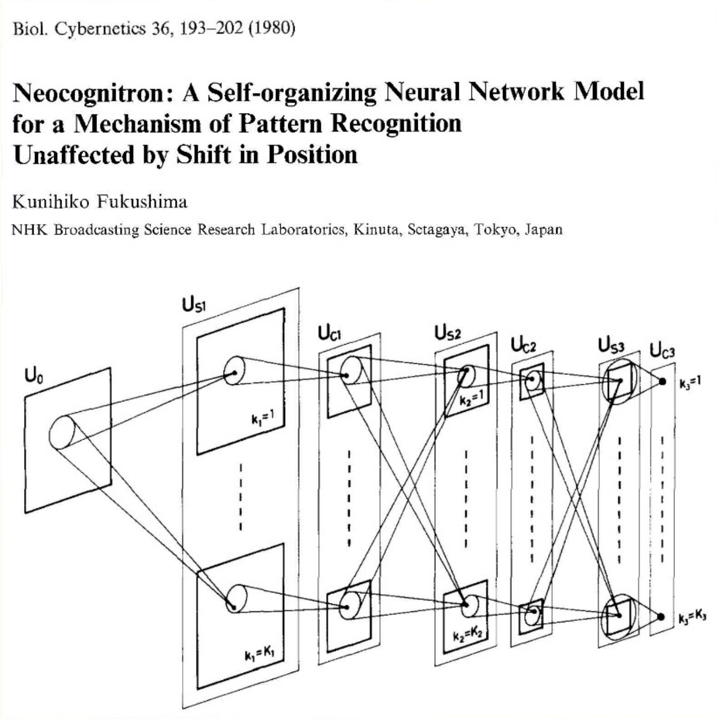
of multilayer networks that compute parity). Similarly, it is not difficult to develop networks capable of solving the connectedness or inside/outside problem. Hinton and Sejnowski have analyzed a version of such a network (see Chapter 7).

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.



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

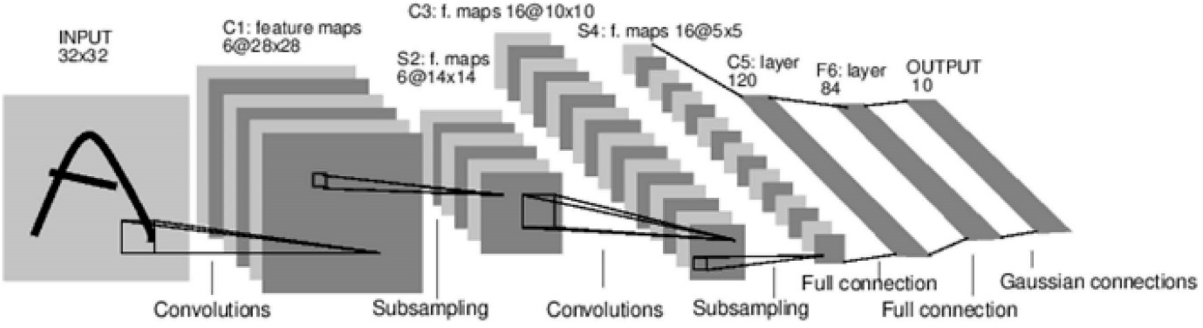


Biol. Cybernetics 36, 193–202 (1980)

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

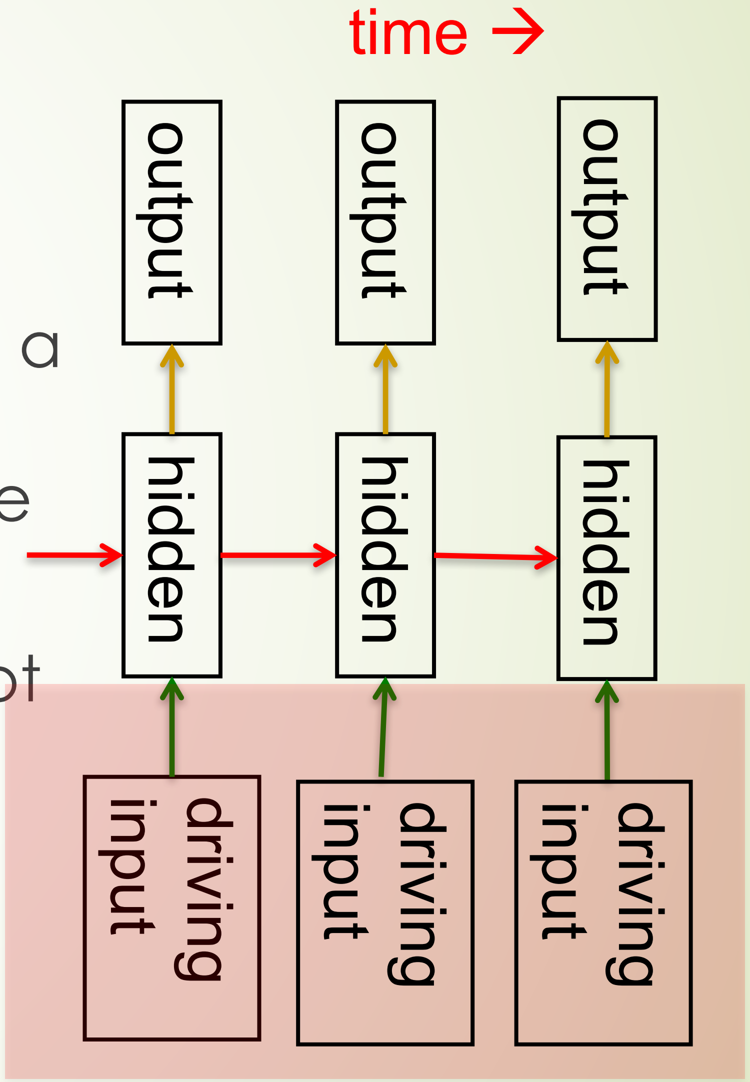


Time series: Linear Dynamical Systems (1940s-)

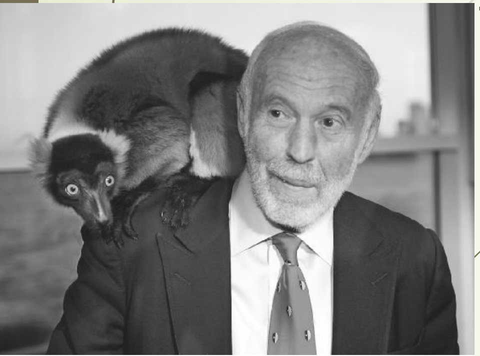
- The hidden state has linear dynamics with Gaussian noise and produces the observations using a linear model with Gaussian noise.
- Kalman Filter: A linearly transformed Gaussian is a Gaussian. So the distribution over the hidden state given the data so far is Gaussian. It can be computed using "Kalman filtering".
- To predict the next output (so that we can shoot down the missile) we need to infer the hidden state.

$$h_t = W_{hh} h_{t-1} + W_{hx} x_t + \epsilon_t^h$$

$$y_t = W_{yh} h_t + W_{yx} x_t + \epsilon_t^y$$

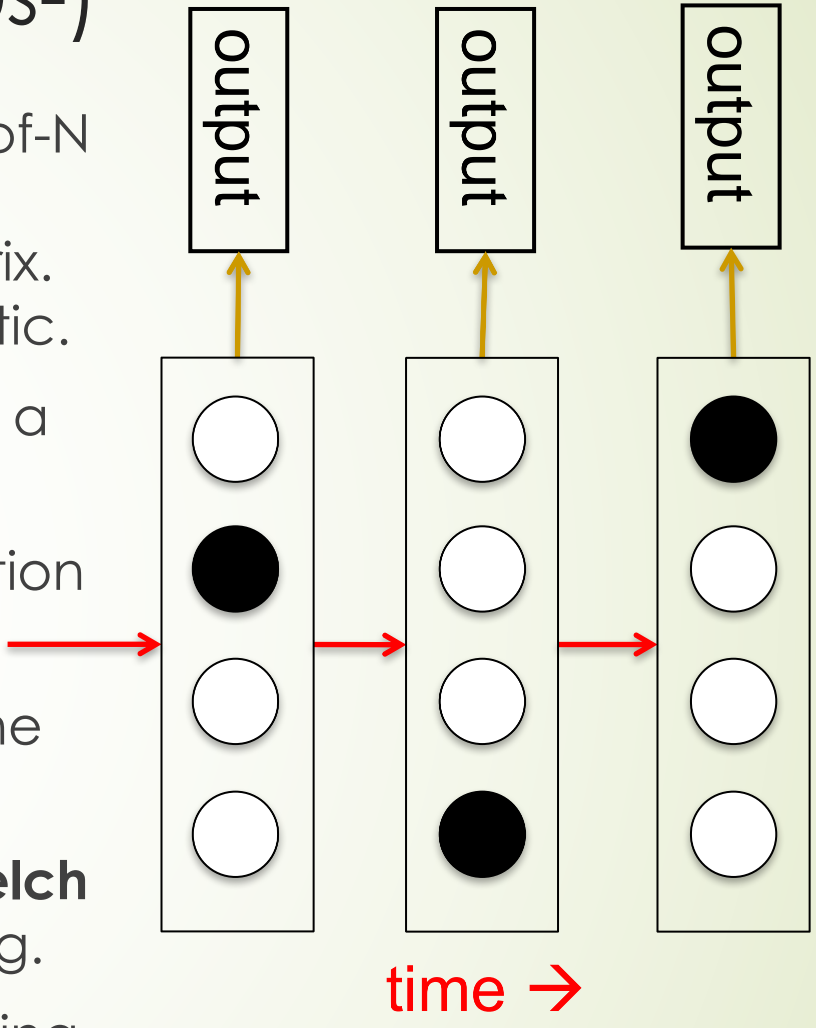


Hidden Markov Models (1970s-)



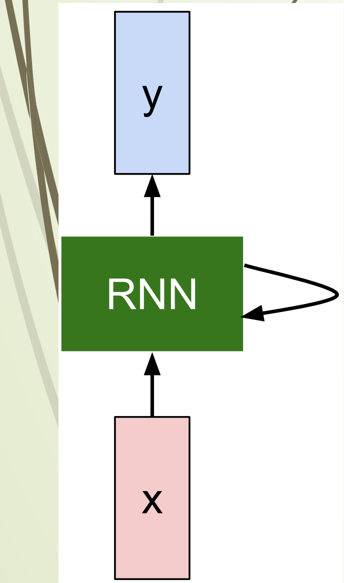
- Hidden Markov Models have a discrete one-of-N hidden state. Transitions between states are stochastic and controlled by a transition matrix. The outputs produced by a state are stochastic.
  - We cannot be sure which state produced a given output. So the state is "hidden".
  - It is easy to represent a probability distribution across N states with N numbers.
- To predict the next output we need to infer the probability distribution over hidden states.
  - HMMs have efficient algorithms (**Baum-Welch or EM Algorithm**) for inference and learning.
  - **Jim Simons** hires Lenny Baum as the founding member of Renaissance Technologies in 1979

Lenny Baum became a devoted Go player despite his deteriorating eyesight.



Recurrent Neural Networks (1986-)

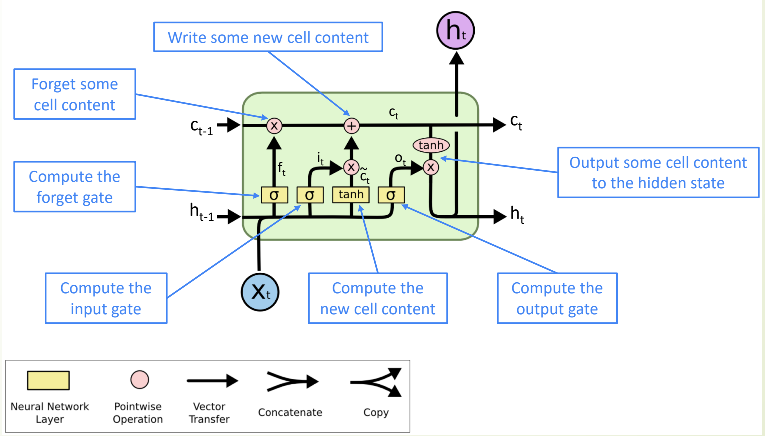
- **The issue of a hidden Markov model (HMM)**:
  - At each time step it must select one of its hidden states. So with N hidden states it can only remember log(N) bits about what it generated so far.
- RNNs are very powerful, because they combine two properties:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- Rumelhart et al. enables training by **BP** algorithm
  - With enough neurons and time, RNNs can compute anything that can be computed by your computer.



$$h_t = \sigma_h(W_{hh}h_{t-1} + W_{hx}x_t)$$
$$y_t = \sigma_y(W_{yh}h_t)$$

Long-Short-Term-Memory (LSTM)

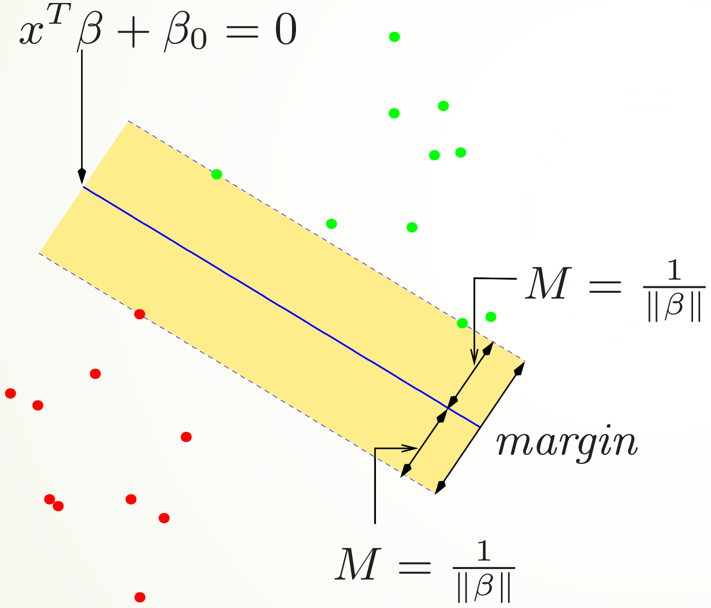
- Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". *Neural Computation*. **9** (8): 1735–1780. (https://www.bioinf.jku.at/publications/older/2604.pdf)
- Introduction of short path to learn deep networks without vanishing gradient problem.

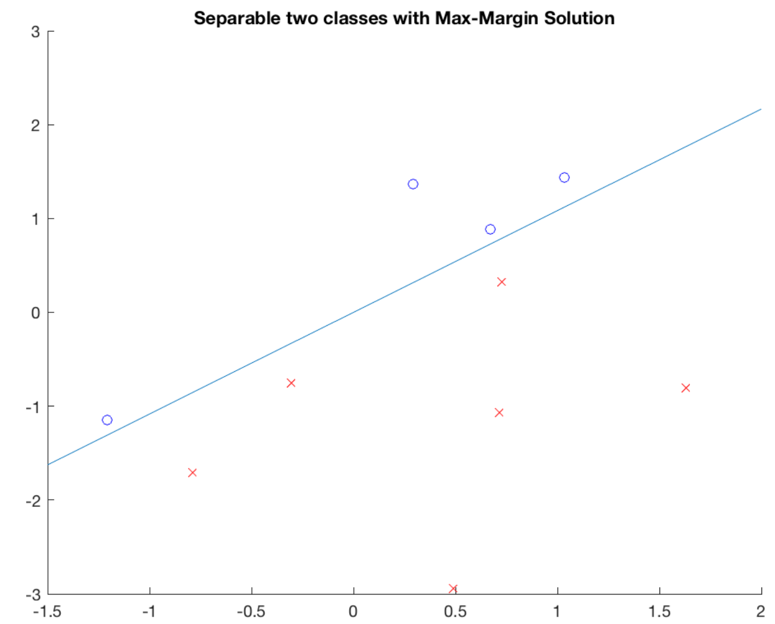


Max-Margin Classifier (SVM)

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

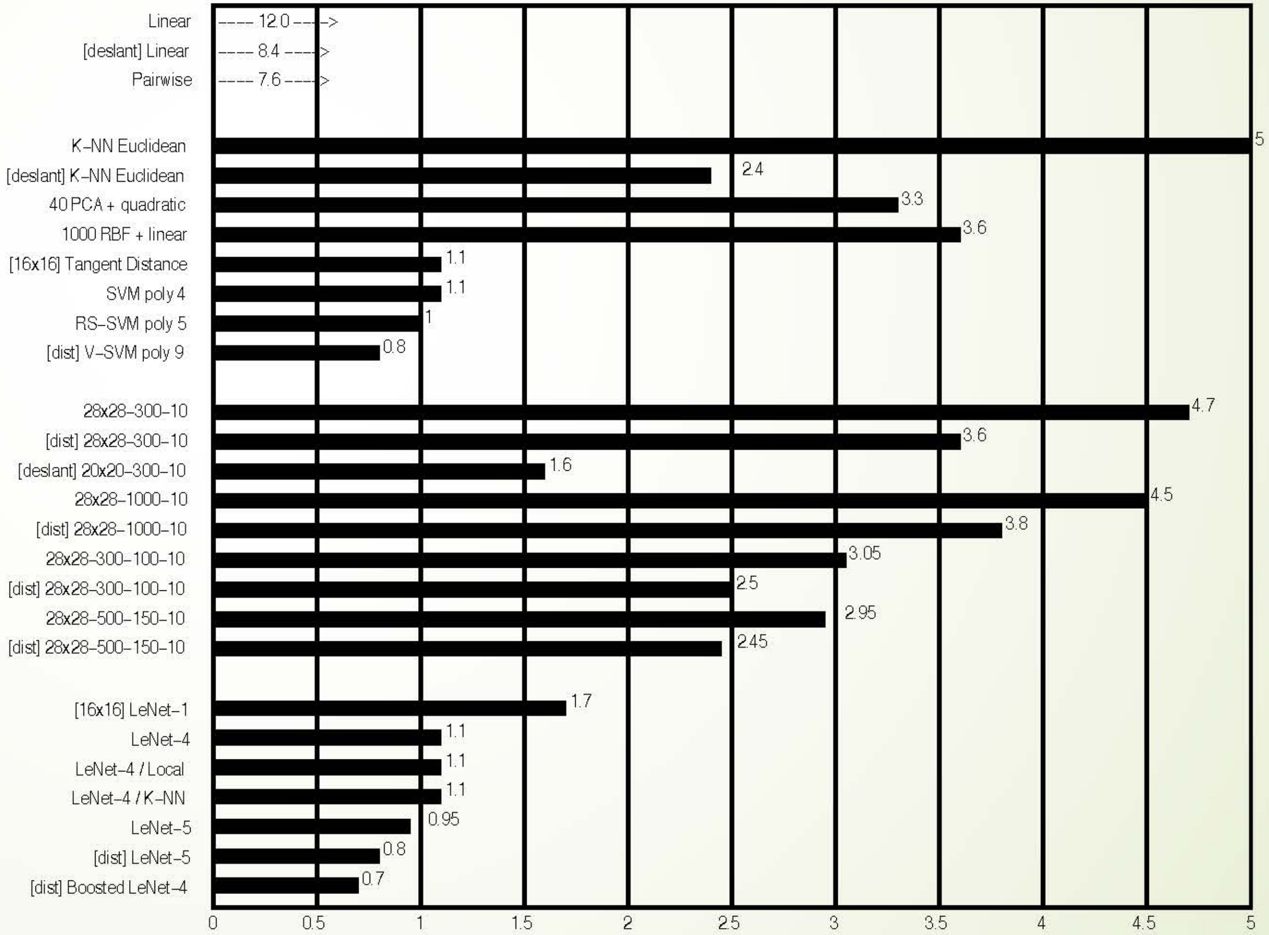
$$\text{subject to } y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1 \text{ for all } i$$





Vladmir Vapnik, 1994

MNIST Dataset Test Error LeCun et al. 1998

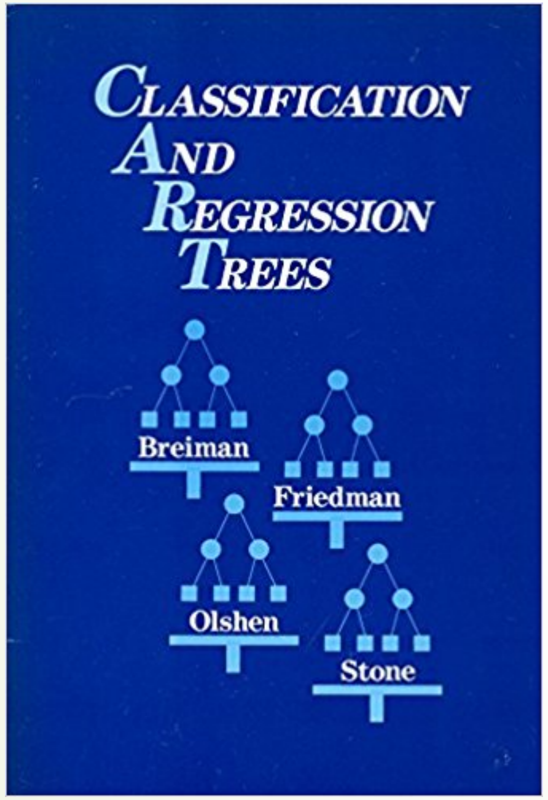


Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets)

Dark era for NN: 1998-2012

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

Decision Trees and Boosting



CLASSIFICATION AND REGRESSION TREES

Breiman

Friedman

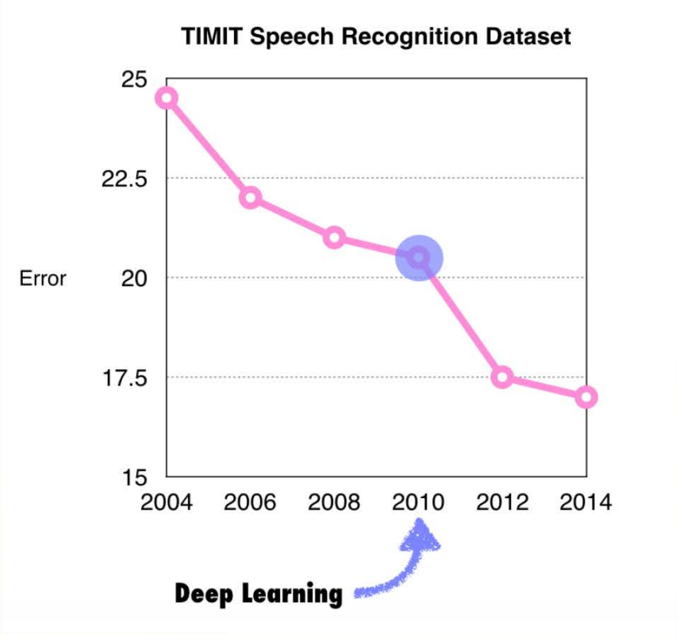
Olshen

Stone

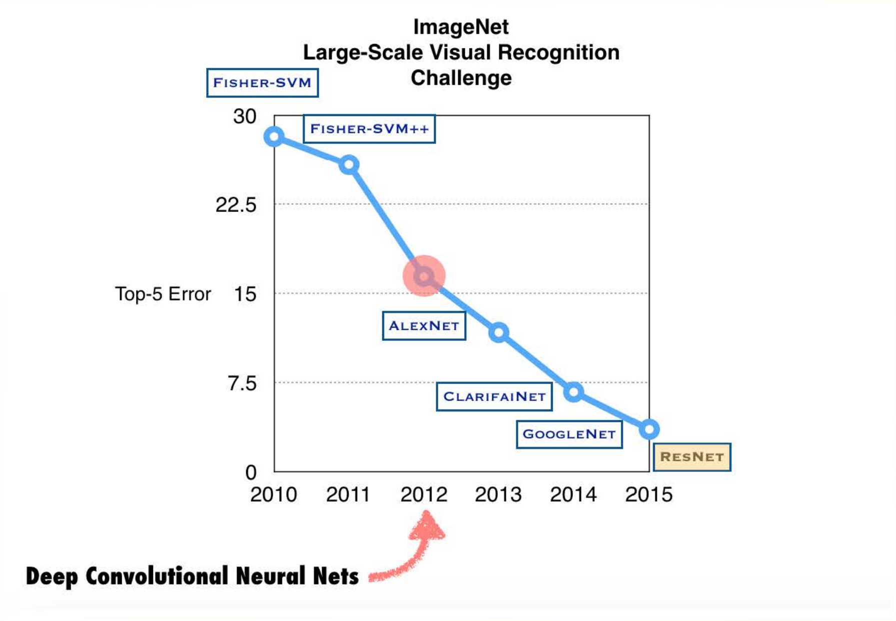
- Breiman, Friedman, Olshen, Stone, (1983): CART
- ``The Boosting problem‘’ (M. Kearns & L. Valiant): **Can a set of weak learners create a single strong learner**? (三个臭皮匠顶个诸葛亮？)
- Breiman (1996): Bagging
- Freund, Schapire (1997): **AdaBoost** (“the best off-the-shelf algorithm” by Breiman)
- Breiman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

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

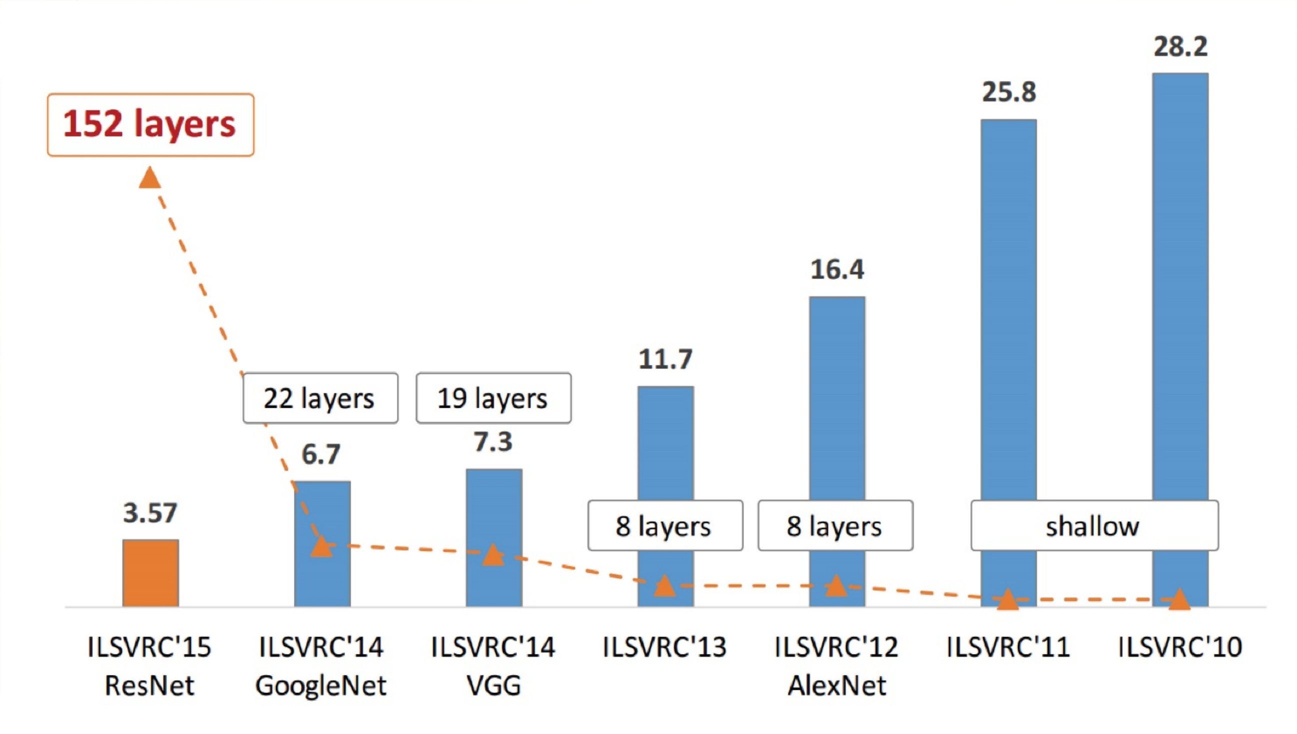
Speech Recognition: TIMIT



Computer Vision: ImageNet



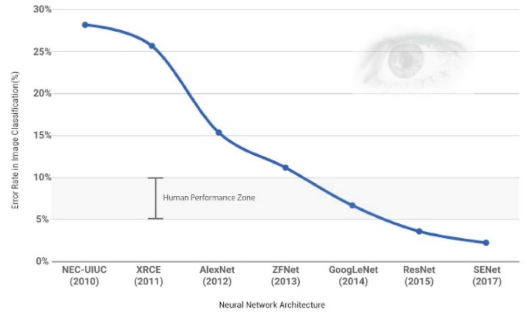
Depth as function of year



[He et al., 2016]

ILSVRC ImageNet Top 5 errors

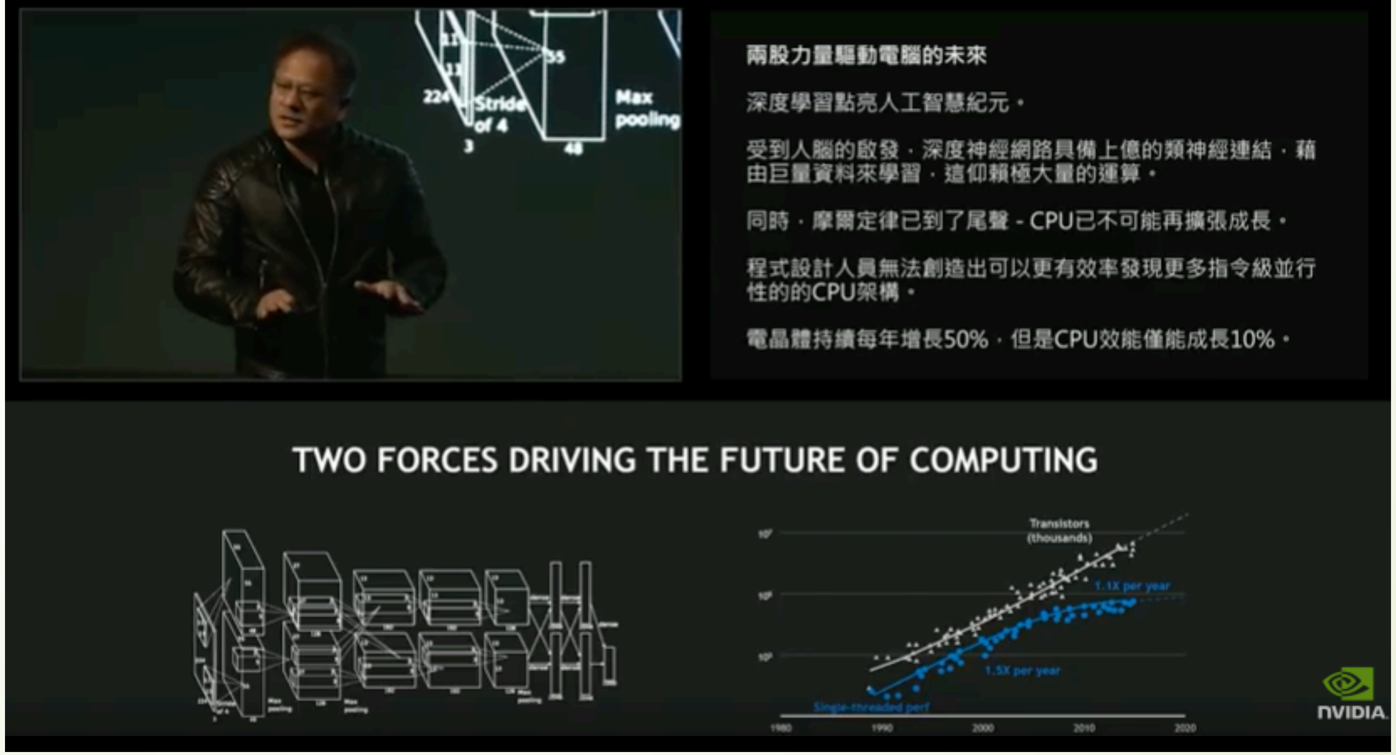
- ImageNet (subset):
  - 1.2 million training images
  - 100,000 test images
  - 1000 classes
- ImageNet large-scale visual recognition Challenge



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

GPU + Big labeled data

"We're at the beginning of a new day…
This is the beginning of the AI revolution."
— Jensen Huang, GTC Taiwan 2017



兩股力量驅動電腦的未來

深度學習點亮人工智慧紀元。

受到人腦的啟發，深度神經網路具備上億的類神經連結，藉由巨量資料來學習，這仰賴極大量的運算。

同時，摩爾定律已到了尾聲 - CPU已不可能再擴張成長。

程式設計人員無法創造出可以更有效率發現更多指令級並行性的的CPU架構。

電晶體持續每年增長50%，但是CPU效能僅能成長10%。

TWO FORCES DRIVING THE FUTURE OF COMPUTING

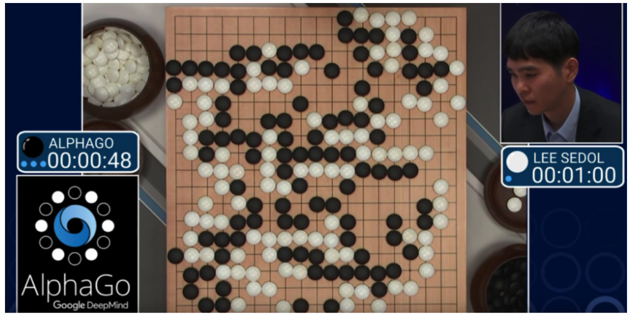
Reaching Human Performance Level in Games

May 11th, 1997

Computer won world champion of chess

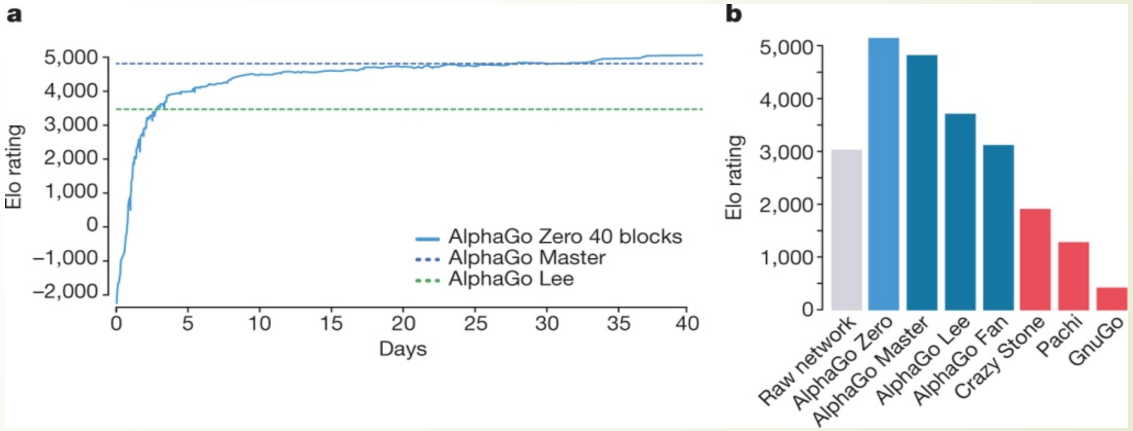
(Deep Blue) (Garry Kasparov)

Deep Blue in 1997



nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO

AlphaGo “LEE” 2016



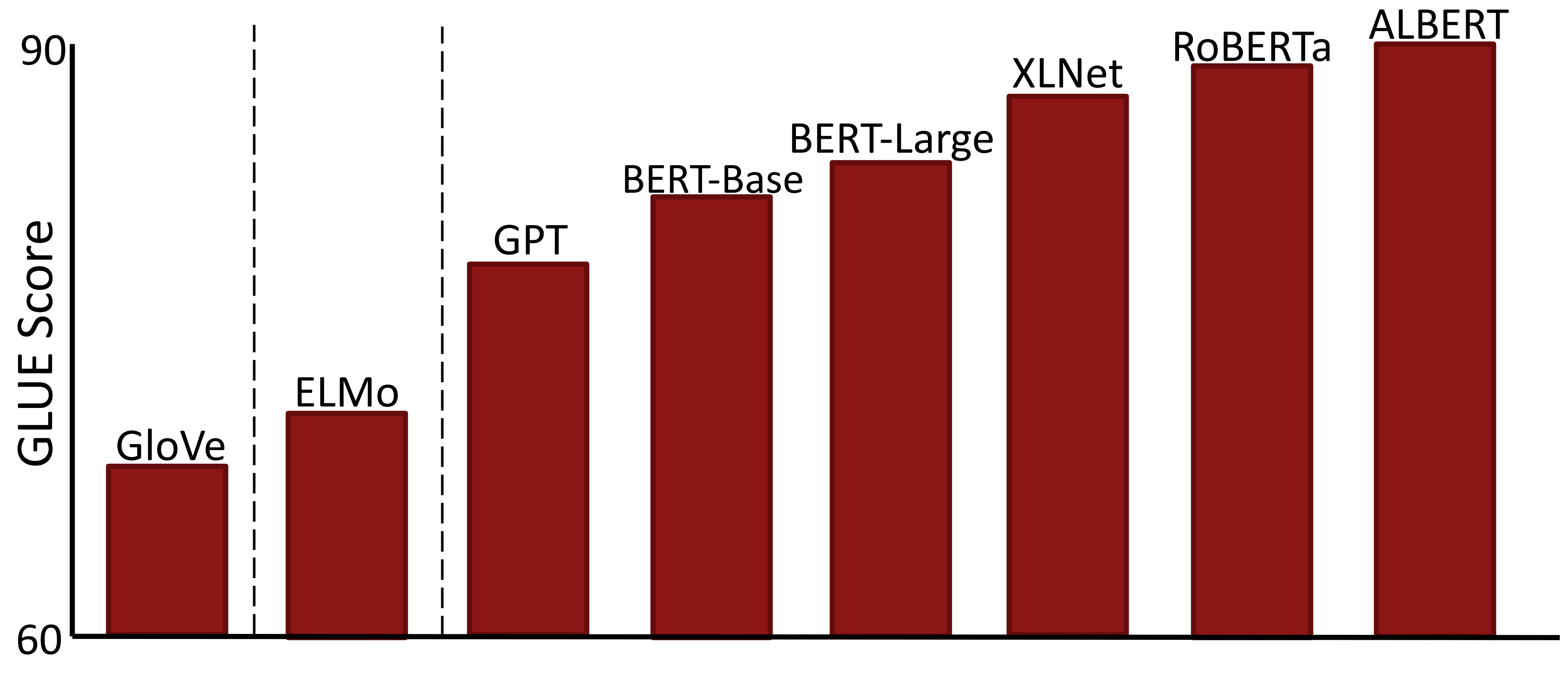
AlphaGo ”ZERO” D Silver *et al. Nature* **550,** 354–359 (2017) doi:10.1038/nature24270

Natural Language Processing (NLP) and Machine Translation

- In **2013-2015**, **LSTM**s started achieving state-of-the-art results
  - Successful tasks include: handwriting recognition, speech
  - recognition, machine translation, parsing, image captioning
  - LSTM became the dominant approach
- In **2019**, other approaches (e.g. **Transformers**) have become more dominant for certain tasks.
  - For example in **WMT** (a MT conference + competition):
  - In WMT 2016, the summary report contains ”RNN” 44 times
  - In WMT 2018, the report contains “RNN” 9 times and “Transformer” 63 times

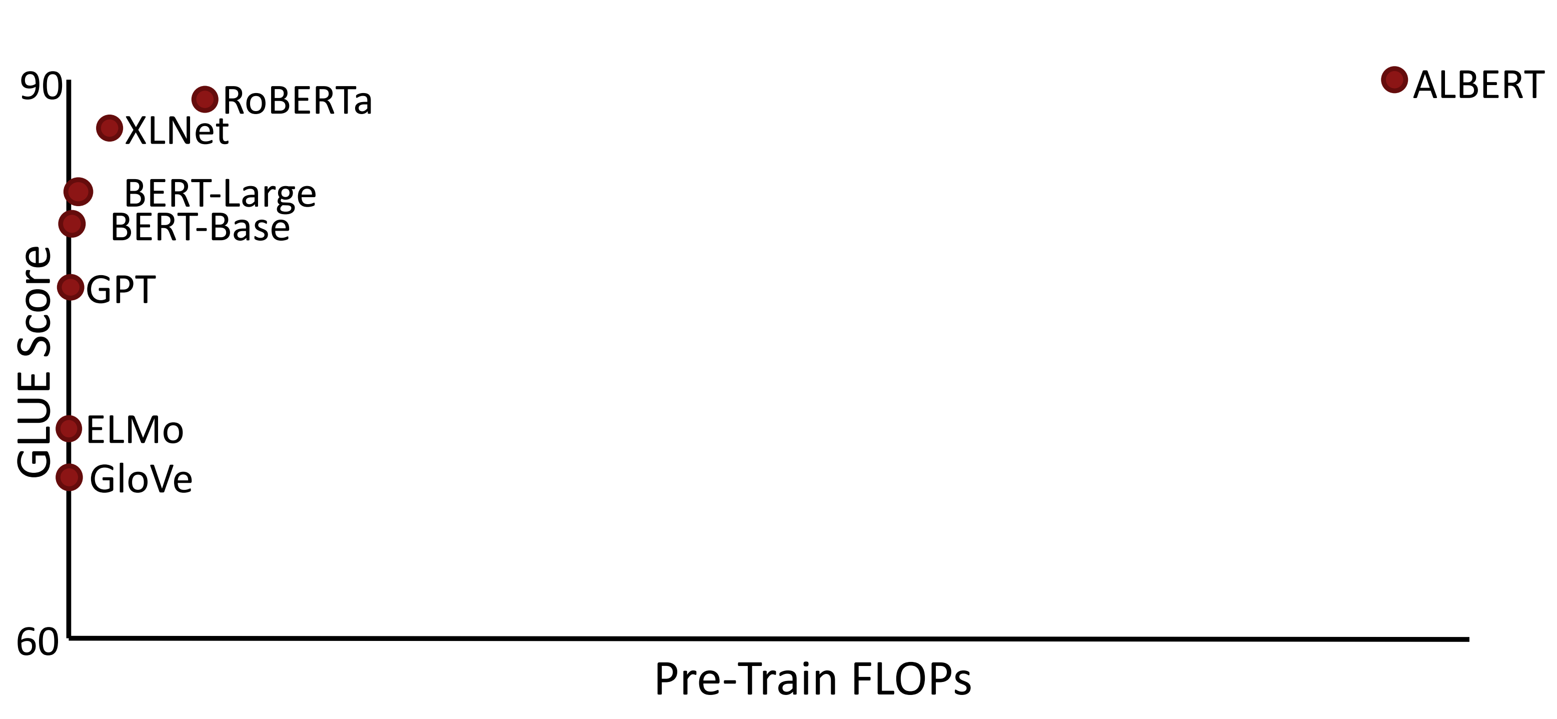
  - **Source:** "Findings of the 2016 Conference on Machine Translation (WMT16)", Bojar et al. 2016, http://www.statmt.org/wmt16/pdf/W16-2301.pdf
  - **Source:** "Findings of the 2018 Conference on Machine Translation (WMT18)", Bojar et al. 2018, http://www.statmt.org/wmt18/pdf/WMT028.pdf

Rapid Progress for NLP Pretraining (GLUE Benchmark)



Over 3x reduction in error in 2 years, “superhuman” performance

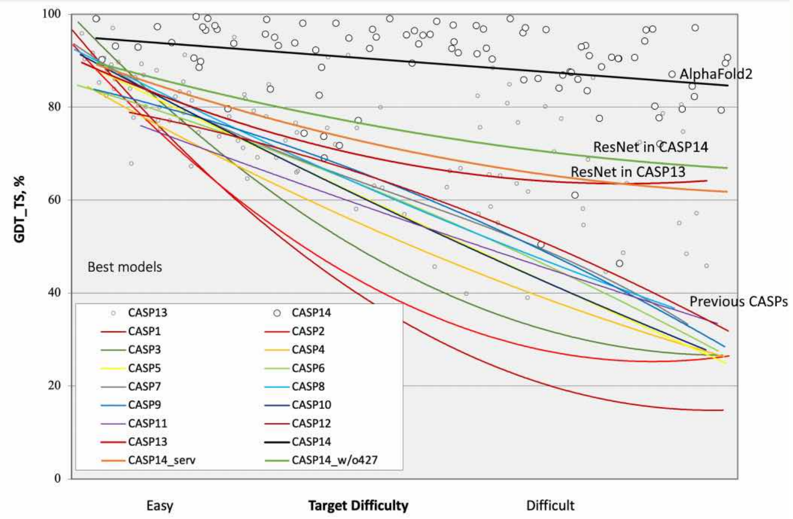
More compute, more better?

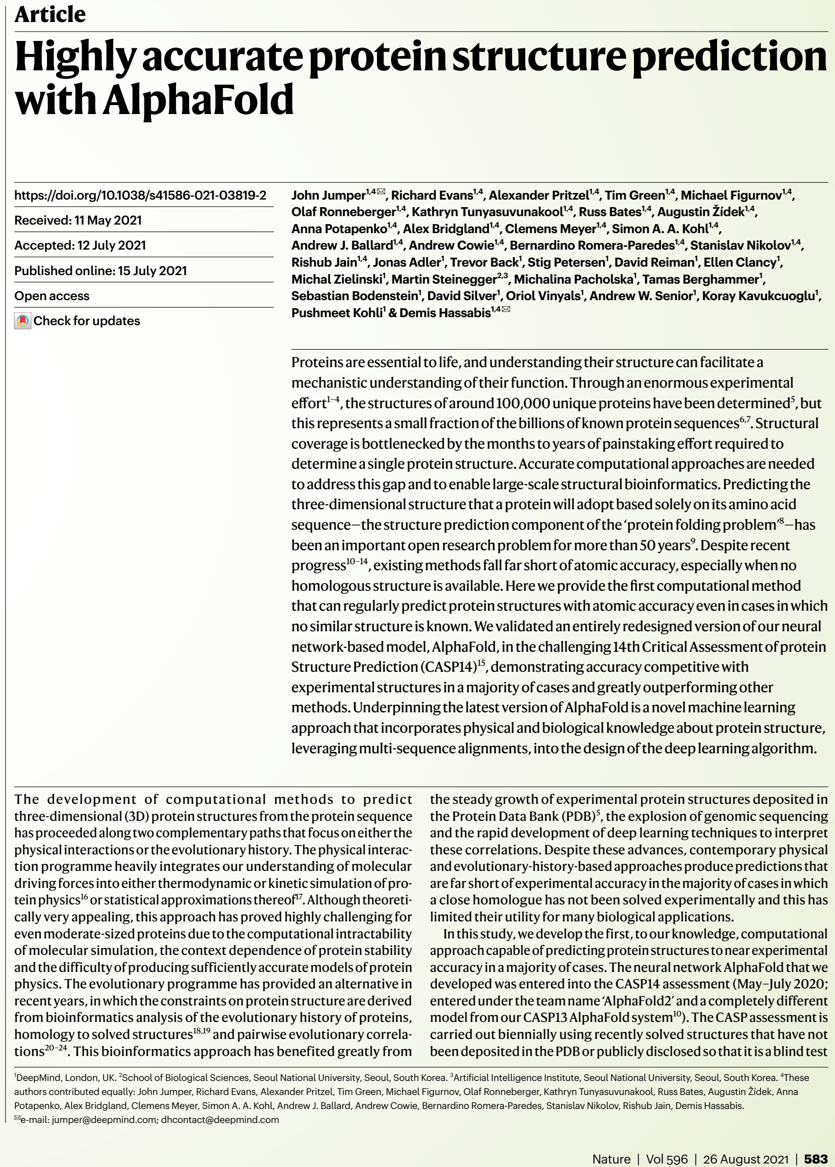


ALBERT uses 10x more compute than RoBERTa

Protein Folding Structure Prediction

AlphaFold





Highly accurate protein structure prediction with AlphaFold

https://doi.org/10.1038/s41586-021-03819-2
Received: 11 May 2021
Accepted: 12 July 2021
Published online: 15 July 2021
Open access
Check for updates

John Jumper[1,4]✉, Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kathryn Tunyasuvunakool[1,4], Russ Bates[1,4], Augustin Žídek[1,4], Anna Potapenko[1,4], Alex Bridgland[1,4], Clemens Meyer[1,4], Simon A. A. Kohl[1,4], Andrew J. Ballard[1,4], Andrew Cowie[1,4], Bernardino Romera-Paredes[1,4], Stanislav Nikolov[1,4], Rishub Jain[1,4], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2,3], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1] & Demis Hassabis[1,4]✉

Proteins are essential to life, and understanding their structure can facilitate a mechanistic understanding of their function. Through an enormous experimental effort[1–4], the structures of around 100,000 unique proteins have been determined[5], but this represents a small fraction of the billions of known protein sequences[6,7]. Structural coverage is bottlenecked by the months to years of painstaking effort required to determine a single protein structure. Accurate computational approaches are needed to address this gap and to enable large-scale structural bioinformatics. Predicting the three-dimensional structure that a protein will adopt based solely on its amino acid sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite recent progress[10–14], existing methods fall far short of atomic accuracy, especially when no homologous structure is available. Here we provide the first computational method that can regularly predict protein structures with atomic accuracy even in cases in which no similar structure is known. We validated an entirely redesigned version of our neural network-based model, AlphaFold, in the challenging 14th Critical Assessment of protein Structure Prediction (CASP14)[15], demonstrating accuracy competitive with experimental structures in a majority of cases and greatly outperforming other methods. Underpinning the latest version of AlphaFold is a novel machine learning approach that incorporates physical and biological knowledge about protein structure, leveraging multi-sequence alignments, into the design of the deep learning algorithm.

The development of computational methods to predict three-dimensional (3D) protein structures from the protein sequence has proceeded along two complementary paths that focus on either the physical interactions or the evolutionary history. The physical interaction programme heavily integrates our understanding of molecular driving forces into either thermodynamic or kinetic simulation of protein physics[16] or statistical approximations thereof[17]. Although theoretically very appealing, this approach has proved highly challenging for even moderate-sized proteins due to the computational intractability of molecular simulation, the context dependence of protein stability and the difficulty of producing sufficiently accurate models of protein physics. The evolutionary programme has provided an alternative in recent years, in which the constraints on protein structure are derived from bioinformatics analysis of the evolutionary history of proteins, homology to solved structures[18,19] and pairwise evolutionary correlations[20–24]. This bioinformatics approach has benefited greatly from the steady growth of experimental protein structures deposited in the Protein Data Bank (PDB)[5], the explosion of genomic sequencing and the rapid development of deep learning techniques to interpret these correlations. Despite these advances, contemporary physical and evolutionary-history-based approaches produce predictions that are far short of experimental accuracy in the majority of cases in which a close homologue has not been solved experimentally and this has limited their utility for many biological applications.

In this study, we develop the first, to our knowledge, computational approach capable of predicting protein structures to near experimental accuracy in a majority of cases. The neural network AlphaFold that we developed was entered into the CASP14 assessment (May–July 2020; entered under the team name 'AlphaFold2' and a completely different model from our CASP13 AlphaFold system[10]). The CASP assessment is carried out biennially using recently solved structures that have not been deposited in the PDB or publicly disclosed so that it is a blind test

[1] DeepMind, London, UK. [2] School of Biological Sciences, Seoul National University, Seoul, South Korea. [3] Artificial Intelligence Institute, Seoul National University, Seoul, South Korea. [4] These authors contributed equally: John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Demis Hassabis.
✉e-mail: jumper@deepmind.com; dhcontact@deepmind.com

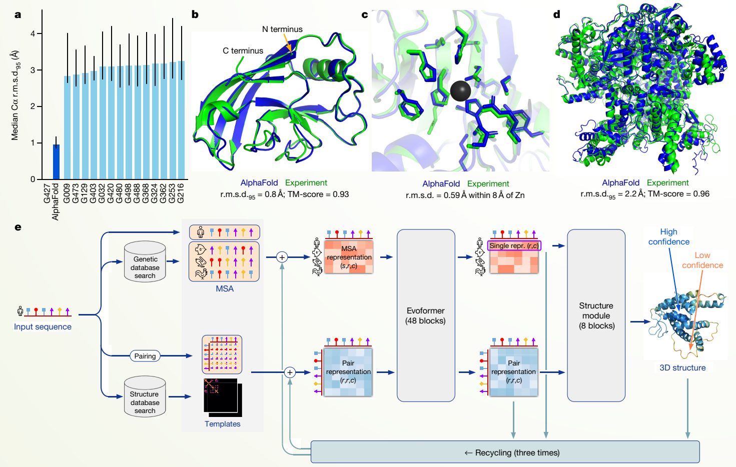


Fig. 1 | AlphaFold produces highly accurate structures. a, The performance of AlphaFold on the CASP14 dataset ($n = 87$ protein domains) relative to the top-15 entries (out of 146 entries), group numbers correspond to the numbers assigned to entrants by CASP. Data are median and the 95% confidence interval of the median, estimated from 10,000 bootstrap samples. **b**, Our prediction of CASP14 target T1049 (PDB 6Y4F, blue) compared with the true (experimental) structure (green). Four residues in the C terminus of the crystal structure are *B*-factor outliers and are not depicted. **c**, CASP14 target T1056 (PDB 6YJ1). An example of a well-predicted zinc-binding site (AlphaFold has accurate side chains even though it does not explicitly predict the zinc ion). **d**, CASP target T1044 (PDB 6VR4)—a 2,180-residue single chain—was predicted with correct domain packing (the prediction was made after CASP using AlphaFold without intervention). **e**, Model architecture. Arrows show the information flow among the various components described in this paper. Array shapes are shown in parentheses with *s*, number of sequences (N_{seq} in the main text); *r*, number of residues (N_{res} in the main text); *c*, number of channels.

Number of AI papers on arXiv, 2010-2019

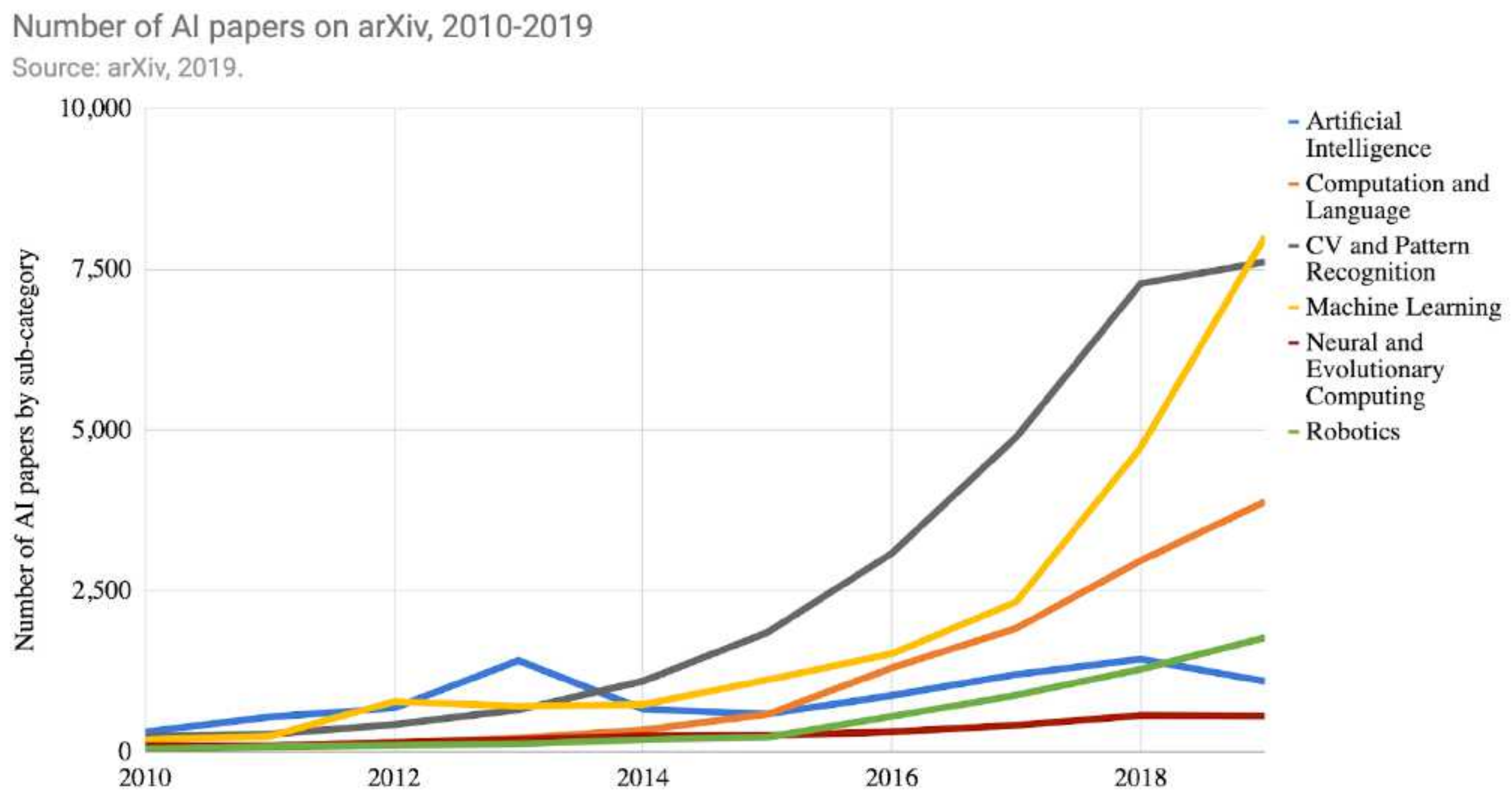
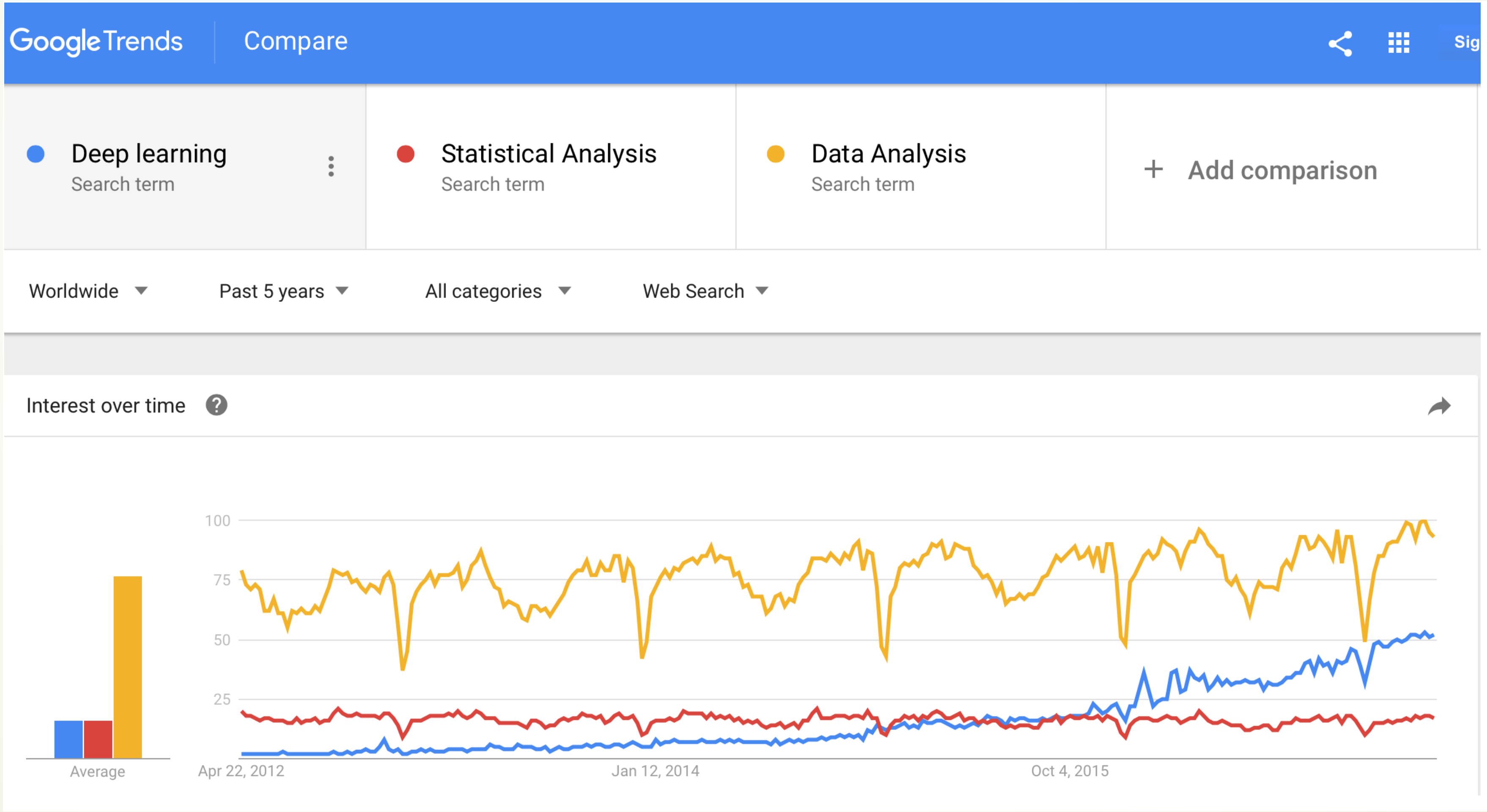


Fig. 1.6.

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.



● Deep learning
Search term

● Statistical Analysis
Search term

● Data Analysis
Search term

+ Add comparison

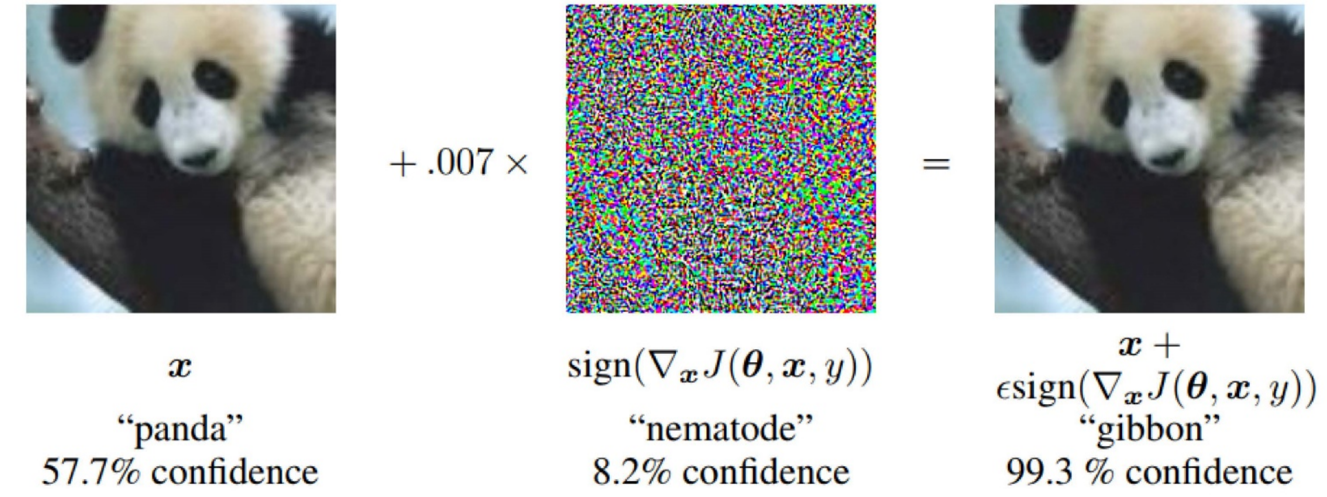
Worldwide ▾ Past 5 years ▾ All categories ▾ Web Search ▾

Interest over time

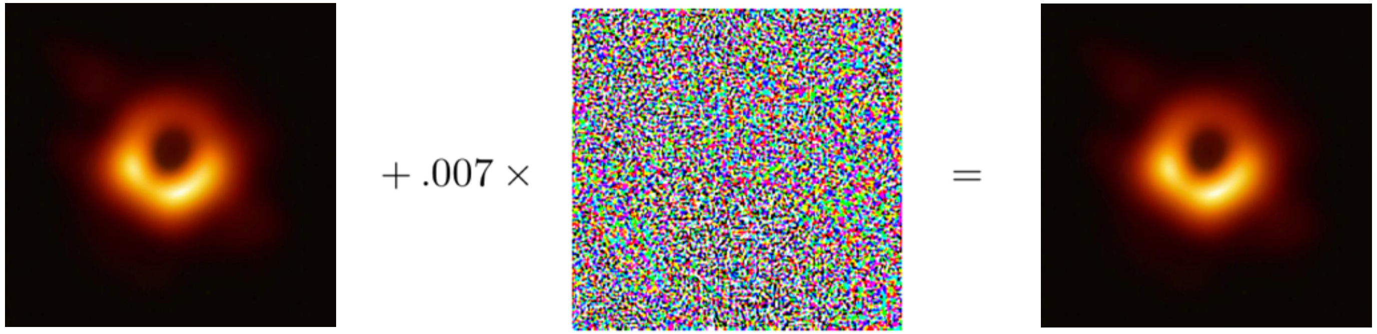
Some Cold Water: Tesla Autopilot Misclassifies Truck as Billboard

Problem: Why? How can you trust a blackbox?

Deep Learning may be fragile in generalization against noise!


+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

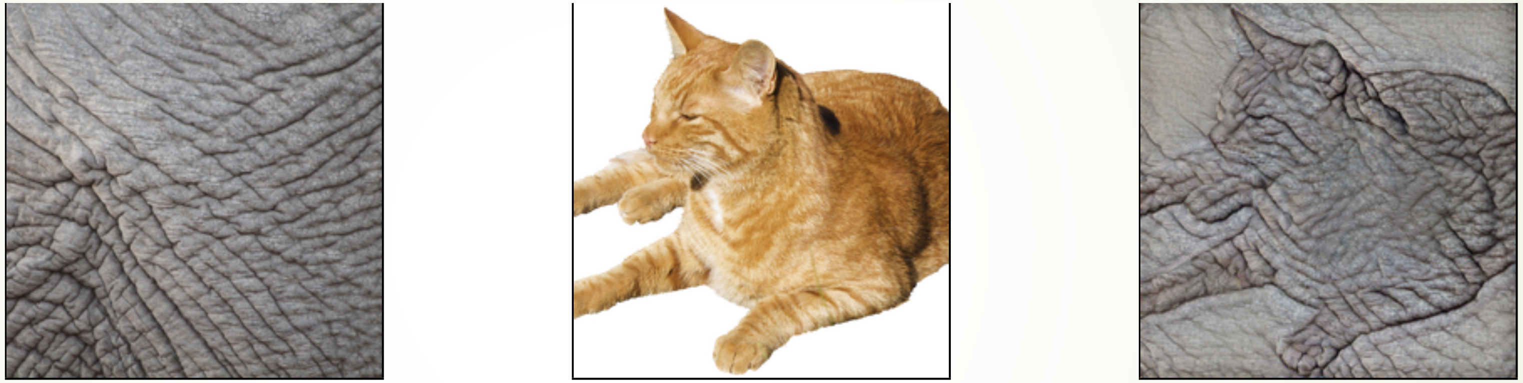
[Goodfellow et al., 2014]



"black hole"
87.7% confidence

"donut"
99.3% confidence

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

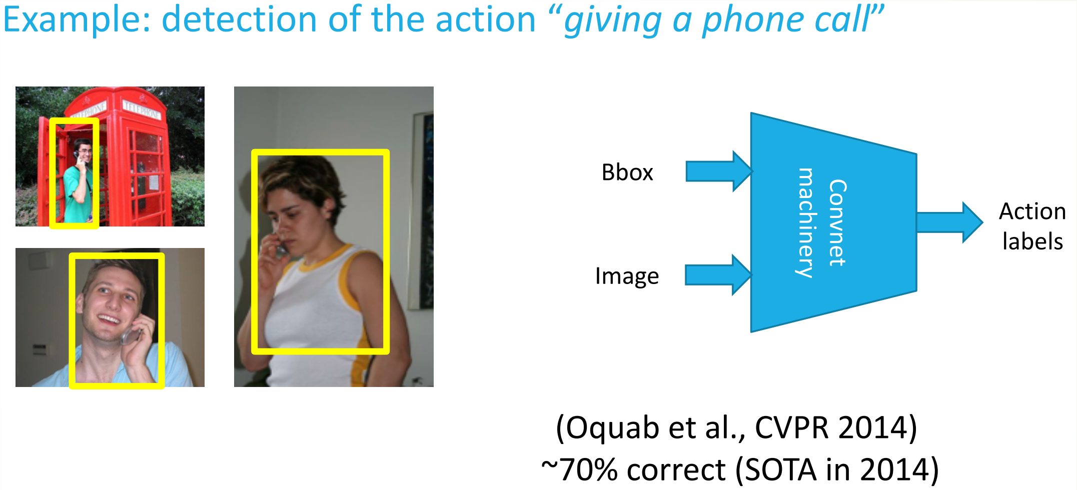
https://videoken.com/embed/W2HvLBMhCJQ?tocitem=46

1:16:47

Capture spurious correlations and can't do causal inference on counterfactuals

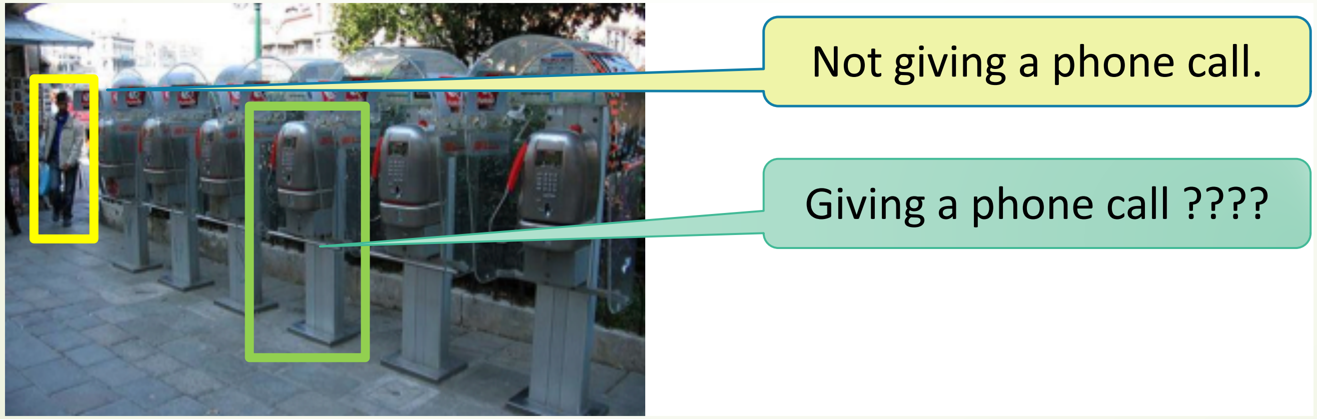
https://videoken.com/embed/8UxS4ls6g1g?tocitem=2

Leon Bottou, ICLR 2019



Example: detection of the action *"giving a phone call"*

(Oquab et al., CVPR 2014)
~70% correct (SOTA in 2014)



Not giving a phone call.

Giving a phone call ????

Overfitting causes privacy leakage

- Model inversion attack leaks privacy

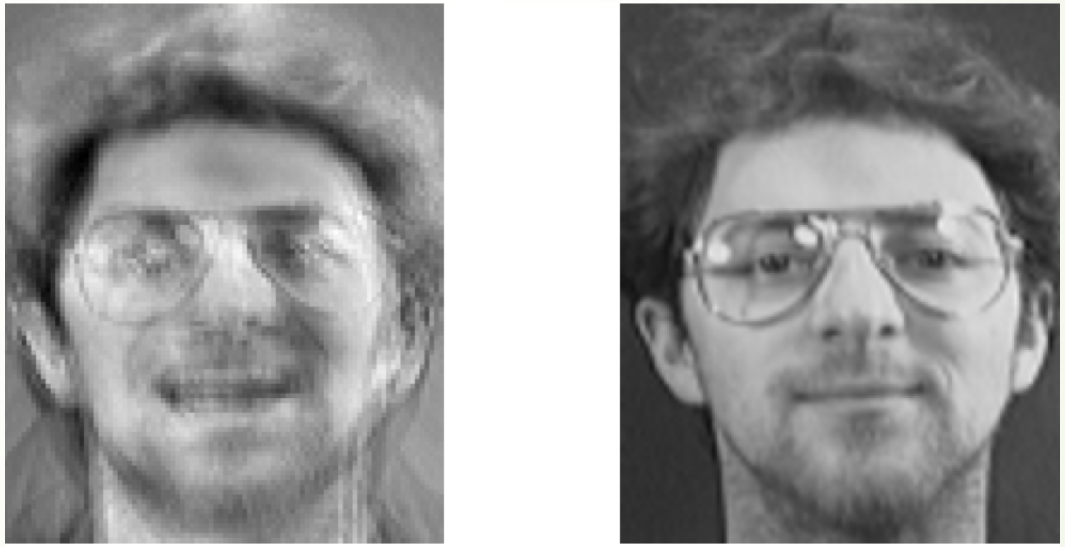


Figure: Recovered (Left), Original (Right)

Fredrikson et al. Proc. CCS, 2016

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 (clarity in understanding)**!

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**

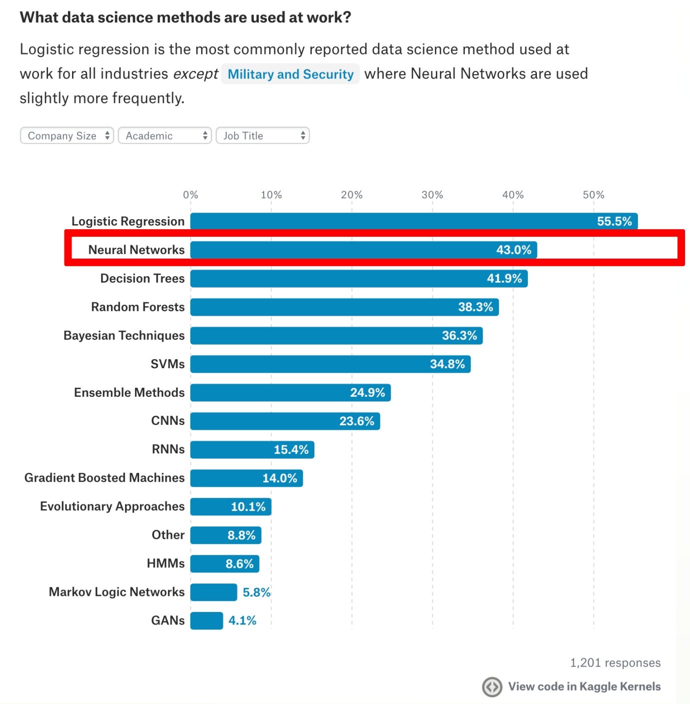
> “Shall we see soon an emergence
> from Alchemy to Science
> in deep leaning?”

How can we teach our students in the next generation science rather than alchemy?

Kaggle survey: Top Data Science Methods

https://www.kaggle.com/surveys/2017

Academic



What data science methods are used at work?

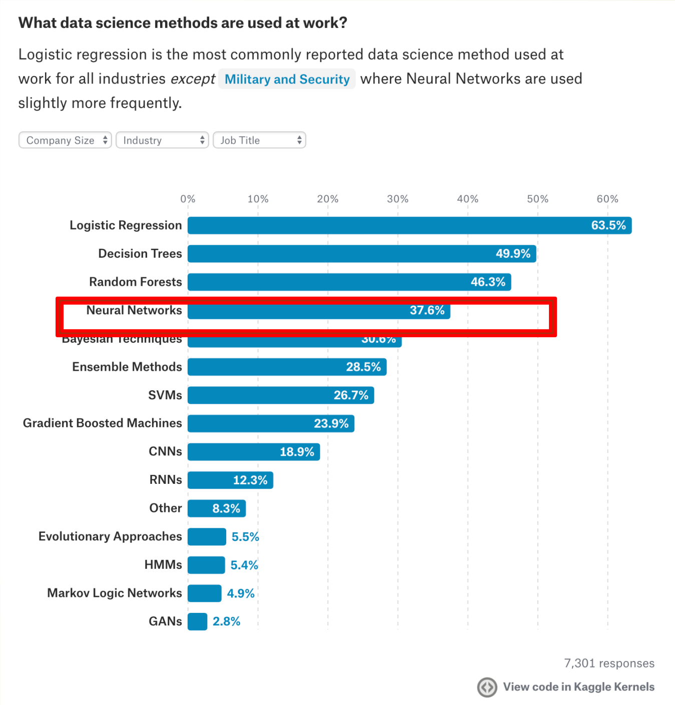
Logistic regression is the most commonly reported data science method used at work for all industries *except* Military and Security where Neural Networks are used slightly more frequently.

Company Size | Academic | Job Title

1,201 responses

View code in Kaggle Kernels

Industry



What data science methods are used at work?

Logistic regression is the most commonly reported data science method used at work for all industries *except* Military and Security where Neural Networks are used slightly more frequently.

Company Size | Industry | Job Title

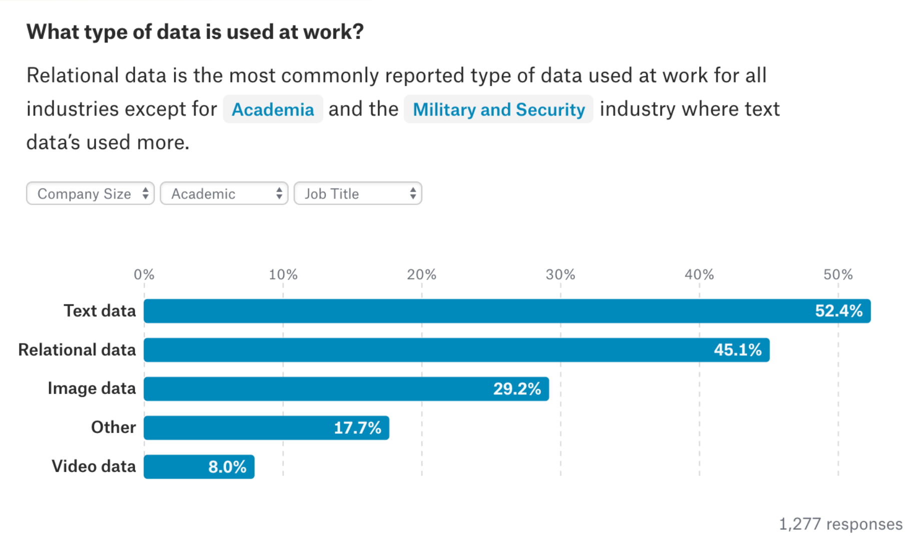
7,301 responses

View code in Kaggle Kernels

What type of data is used at work?

https://www.kaggle.com/surveys/2017

Academic



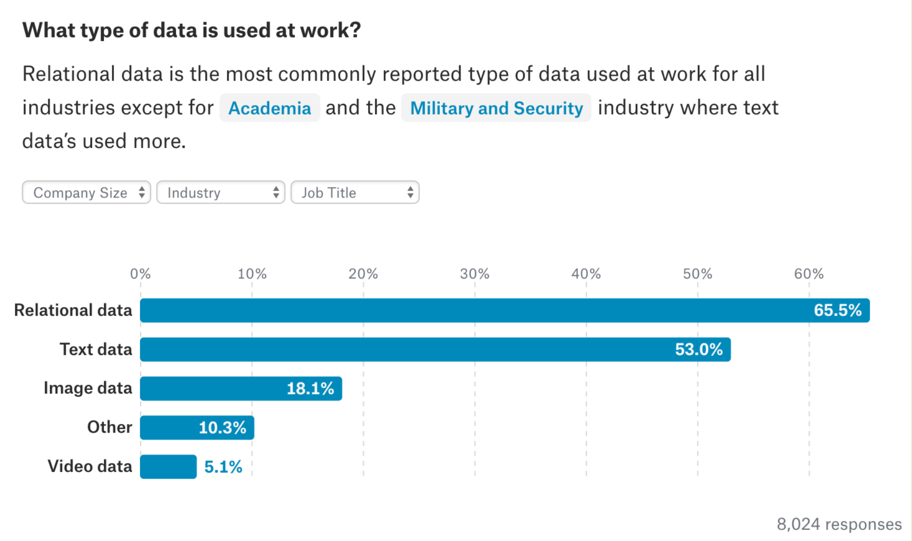
What type of data is used at work?

Relational data is the most commonly reported type of data used at work for all industries except for Academia and the Military and Security industry where text data's used more.

Company Size | Academic | Job Title

1,277 responses

Industry



What type of data is used at work?

Relational data is the most commonly reported type of data used at work for all industries except for Academia and the Military and Security industry where text data's used more.

Company Size | Industry | Job Title

8,024 responses

All models are wrong, but some are useful …

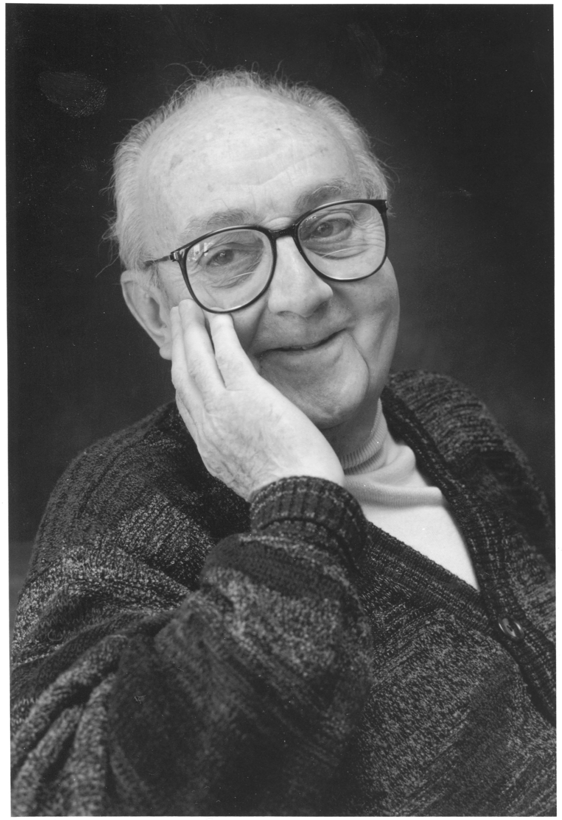


Figure 7: George Box: “Essentially, all models are wrong, but some are useful.”

In this class

- Understand its principles: statistics, optimization
- Analyze the real world data with the methods
- Team-work in projects