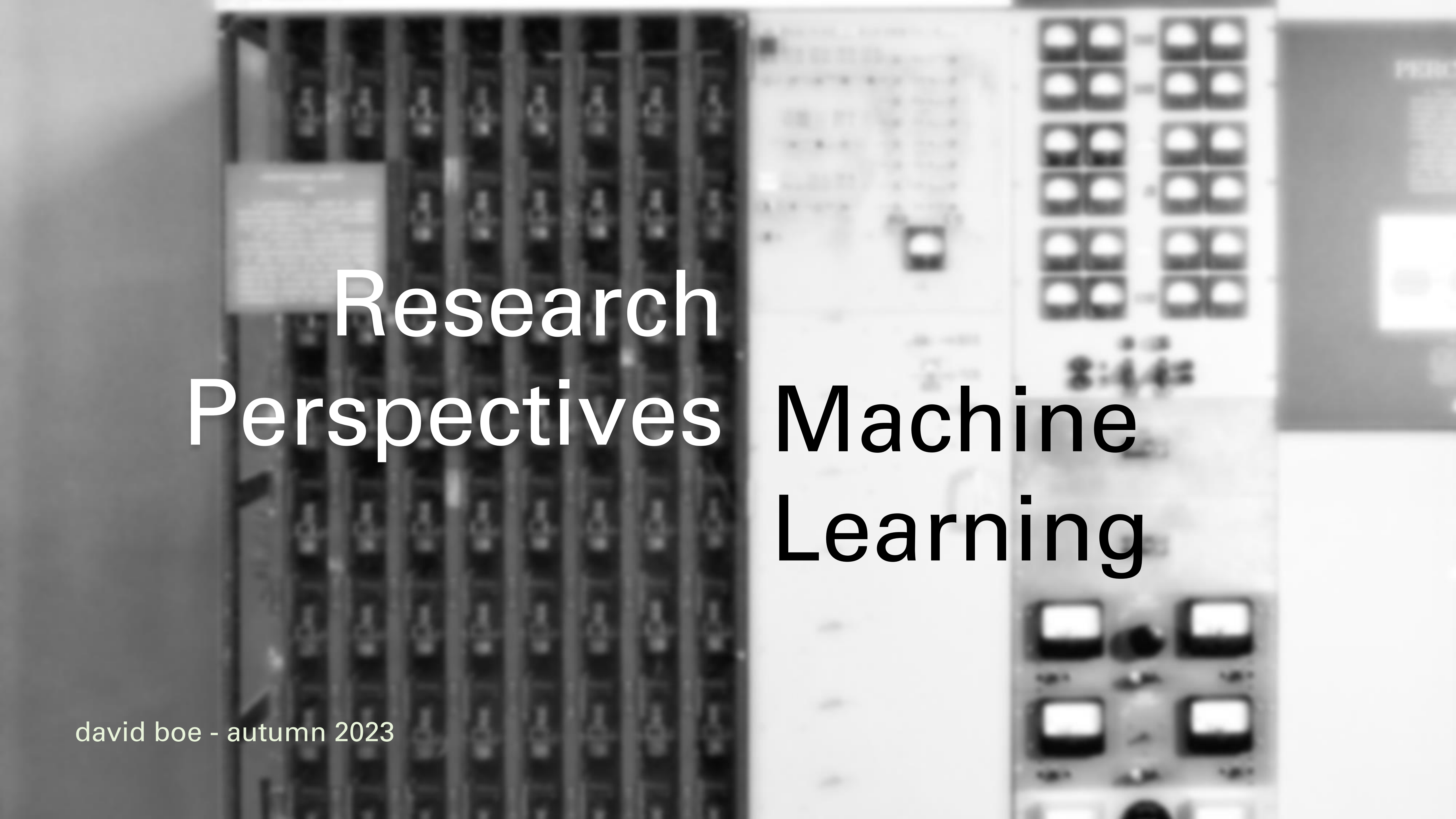


Research Perspectives Machine Learning

david boe - autumn 2023

"typical" career paths for ML

2010-2013

Undergrad in Neurobiology, UW

The forgotten years…

2015-2017

Masters in Prosthetics and Orthotics, UW

2017-2021

Research Engineer, UW

2022-now

ML Researcher, PostEra

agenda

- History of machine learning research (from a connectionist perspective)
- How to read a research paper
- Good and bad paper presentations to your peers

why does history matter

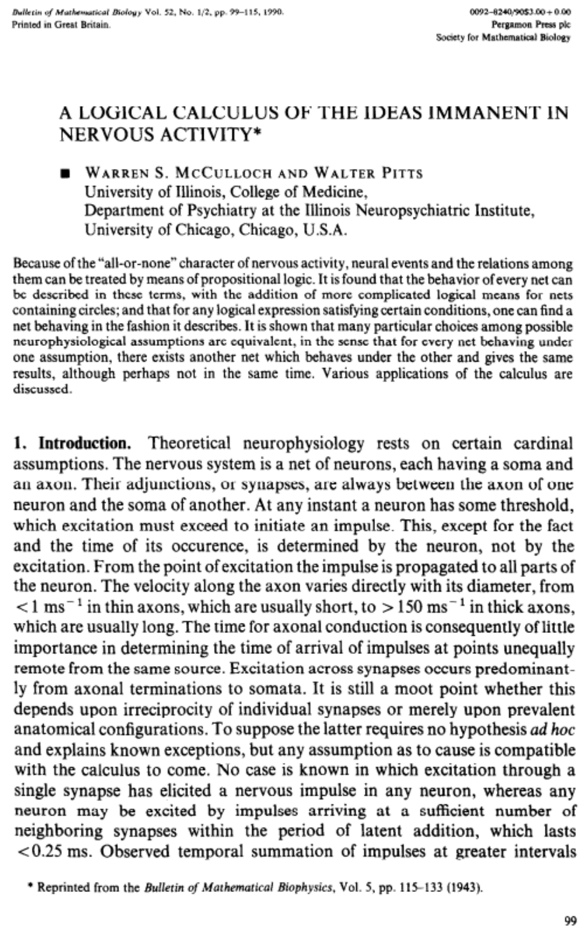
- Good research is about asking the right questions
- Many breakthroughs in ML are simply applications of old ideas with new technologies
- Gives you a competitive edge
- Informs your ability to search related disciplines for relevant innovations

1943 - perceptron

The “Perceptron” was first thought up

Warren McCulloch,
-philosopher
-psychologist
-neurophysiologist
-45 yrs old

Walter Pitts,
-mathematician
-homeless
-20 yrs old



Bulletin of Mathematical Biology Vol. 52, No. 1/2, pp. 99–115, 1990.
Printed in Great Britain.

0092–8240/90$3.00 + 0.00
Pergamon Press plc
Society for Mathematical Biology

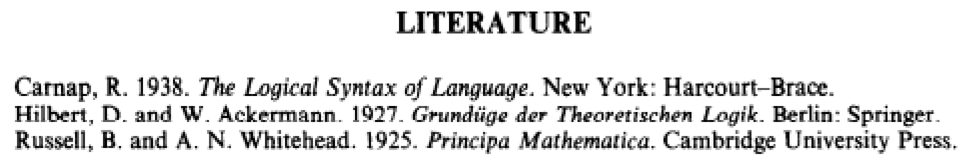
A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

■ WARREN S. MCCULLOCH AND WALTER PITTS
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.

Because of the “all-or-none” character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

1. Introduction. Theoretical neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from $<1\ \mathrm{ms}^{-1}$ in thin axons, which are usually short, to $>150\ \mathrm{ms}^{-1}$ in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time of arrival of impulses at points unequally remote from the same source. Excitation across synapses occurs predominantly from axonal terminations to somata. It is still a moot point whether this depends upon irreciprocity of individual synapses or merely upon prevalent anatomical configurations. To suppose the latter requires no hypothesis *ad hoc* and explains known exceptions, but any assumption as to cause is compatible with the calculus to come. No case is known in which excitation through a single synapse has elicited a nervous impulse in any neuron, whereas any neuron may be excited by impulses arriving at a sufficient number of neighboring synapses within the period of latent addition, which lasts <0.25 ms. Observed temporal summation of impulses at greater intervals

* Reprinted from the *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115–133 (1943).

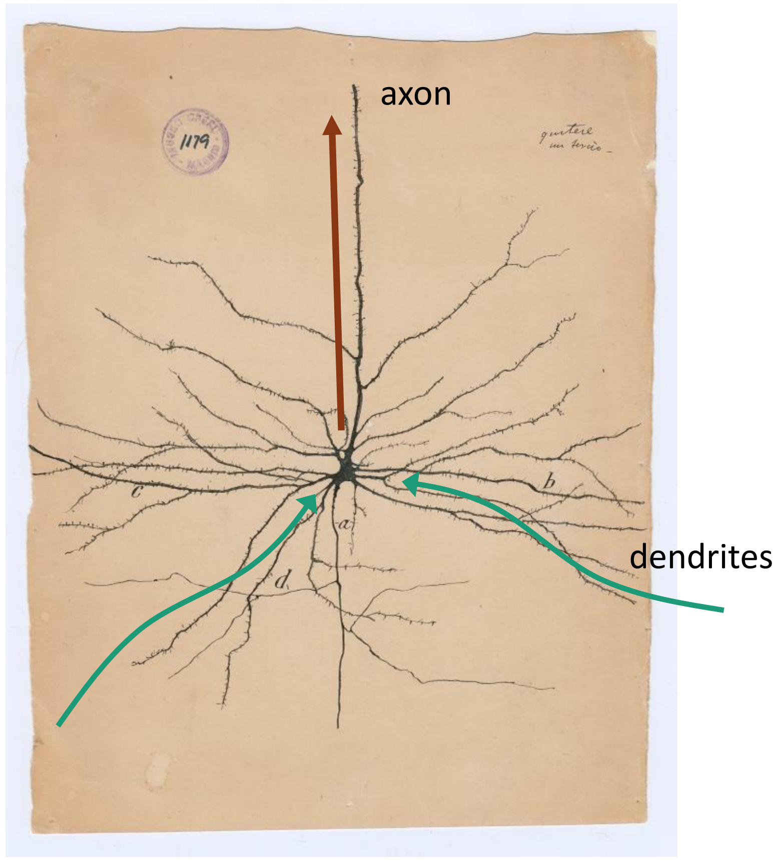


LITERATURE

Carnap, R. 1938. *The Logical Syntax of Language*. New York: Harcourt–Brace.
Hilbert, D. and W. Ackermann. 1927. *Grundüge der Theoretischen Logik*. Berlin: Springer.
Russell, B. and A. N. Whitehead. 1925. *Principa Mathematica*. Cambridge University Press.

1943 - perceptron

Inspired by physiology of the brain



Cajal, 1904, neuroanatomist and artist

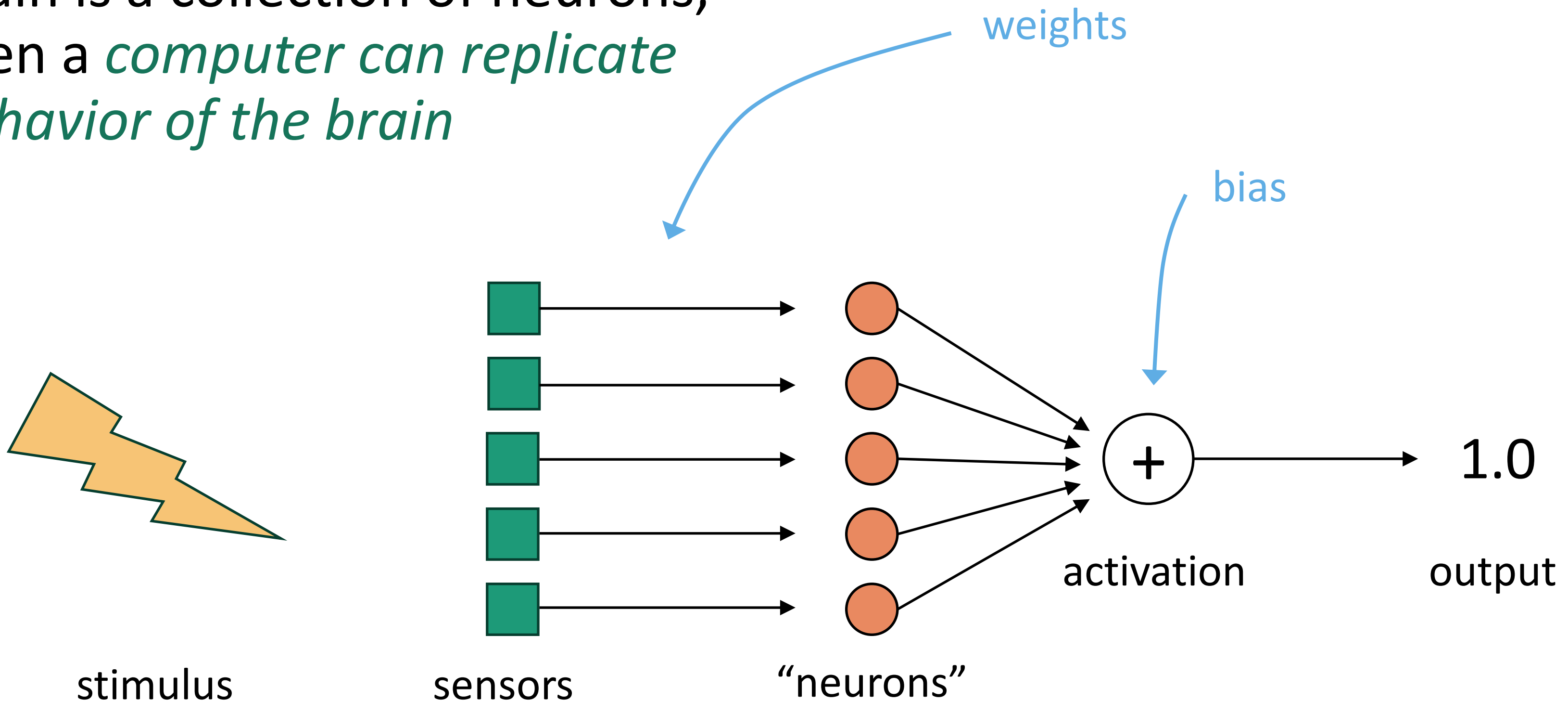
Every dendrite has a “weight”

Dendrites excite the neuron proportional to their weight

If the neuron excitation rises above a threshold, then the neuron “fires”

1943 - perceptron

If a computer could replicate behavior of a neuron, and the brain is a collection of neurons, then a *computer can replicate behavior of the brain*



1943 - perceptron

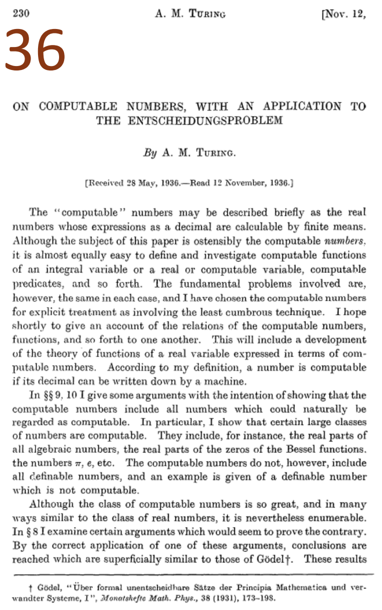
Formalized the concept of *training* (not *programming*) a computer. Idea that computers could *learn* on their own was revolutionary.

Gave rise to the field of neural networks, or connectionism

meanwhile...

Alan Turing is laying the theoretical foundation for computing, launching the field of artificial intelligence.

1936



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

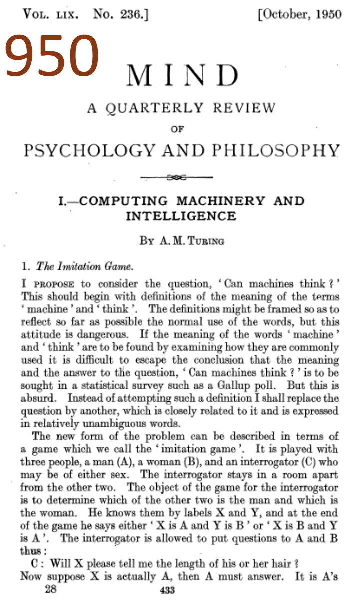
The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

1950



VOL. LIX. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

BY A. M. TURING

1. *The Imitation Game.*

I PROPOSE to consider the question, 'Can machines think ?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think ?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

The new form of the problem can be described in terms of a game which we call the 'imitation game'. It is played with three people, a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. He knows them by labels X and Y, and at the end of the game he says either 'X is A and Y is B' or 'X is B and Y is A'. The interrogator is allowed to put questions to A and B thus:

C: Will X please tell me the length of his or her hair ?

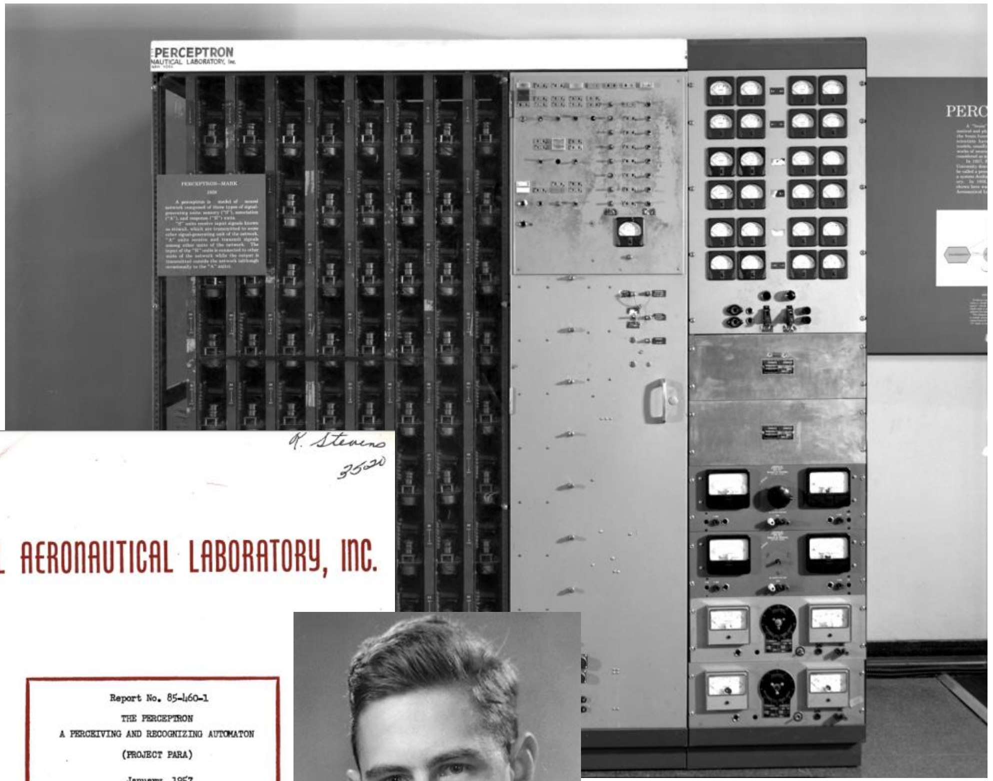
Now suppose X is actually A, then A must answer. It is A's

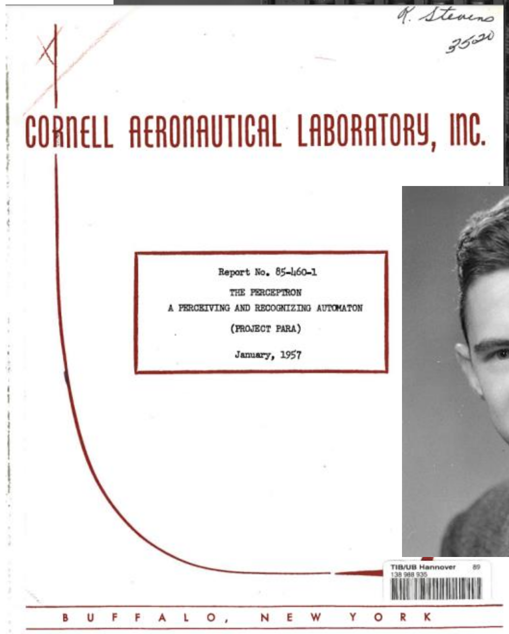
28 433

"Symbolic AI" rises as the dominant research paradigm throughout the 1940s and 1950s.

Symbolic AI relies on sets of hand-made rules and symbols, *"if this, then that"* embedded within an algorithm

1958 – perceptron v1




K. Stevens
35020

CORNELL AERONAUTICAL LABORATORY, INC.

Report No. 85-460-1
THE PERCEPTRON
A PERCEIVING AND RECOGNIZING AUTOMATON
(PROJECT PARA)
January, 1957

B U F F A L O, N E W Y O R K

Frank Rosenblatt

-cognitive scientist

-hype man

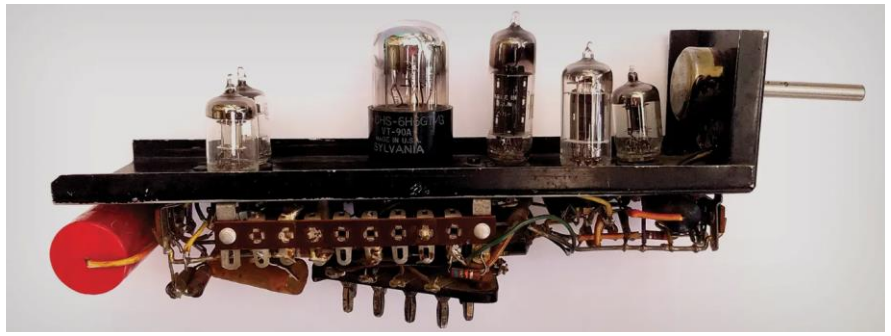
"The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence . . . Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted. (New York Times, 1958a, p. 25 :2)"

1958 – perceptron v1

Rosenblatt’s exuberant personality + sensational media reporting made him the enemy of symbolic AI.

Enter his nemesis, Marvin Minsky, mathematician.

Minsky built his own hardware implementation of a neural network in 1951, called SNARC (Stochastic Neural Analog Reinforcement Calculator).



1958 – 1965

Minsky and Rosenblatt were both well aware of the challenges of neural networks. To name a few…

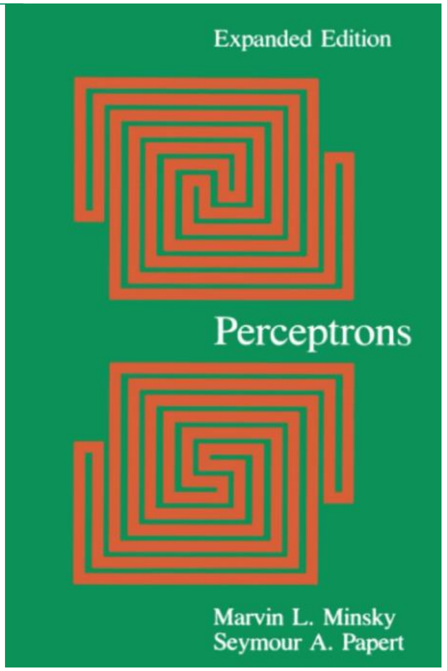
- There was no way to know if a perceptron could solve a problem *a priori*
- Could add layers, but nobody figured out how to adjust the weights
- Networks had poor generalization
- Not SE3 invariant
- Could not solve simple problems like XOR
- Could only work on linearly separable classes

1960s – the Minsky takedown

Marvin Minsky and Seymour Papert (also mathematician) published *Perceptrons* in 1969.

They set out to describe the shortcomings of the connectionist movement, once and for all (and divert funding towards their own work).

Connectionism was increasingly viewed with skepticism, and this book fully killed it off.



1970s – first AI winter

Connectionism failure to deliver brought down the whole AI field. Analog computers were discarded. Funding dried up. Scientists moved on to other areas.

But machine learning scientists quietly continued…

During this time, several new methods were born

- Digital computing gained power
- Least mean squares (LMS) – *Widrow and Hoff*
- Neocognitron for recognizing Japanese handwriting, *Kunihiko Fukushima*
- First self-organizing networks, *Shun'ichi Amari*, *William Little*
- Automatic differentiation, *Seppo Linnainmaa*

what did we learn the first time around?

Overhyping expectations can tank an entire scientific discipline (don’t cross mathematicians)

Translation from the lab (AI that plays checkers) to the real world (AI that does taxes) is non-trivial

Machine learning can draw assistance from many other fields

- Robotics
- Physics
- Computing
- Linguistics
- Mathematics
- Cognition
- …?

machine learning needs a savior

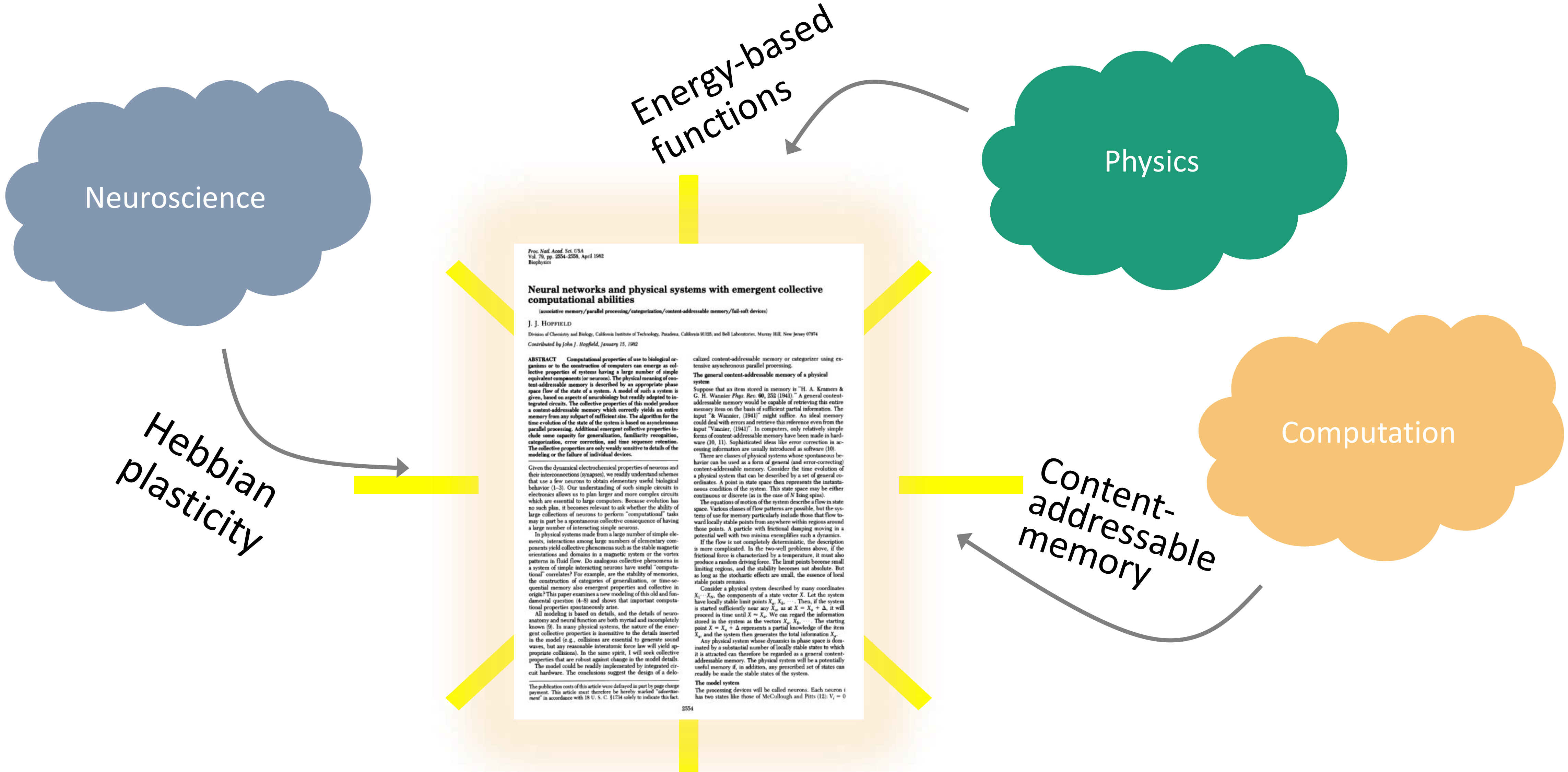
Symbolic AI work was hitting its ceiling. Clunky computers and man-made schemas couldn’t solve longstanding problems like speech recognition or machine translation.

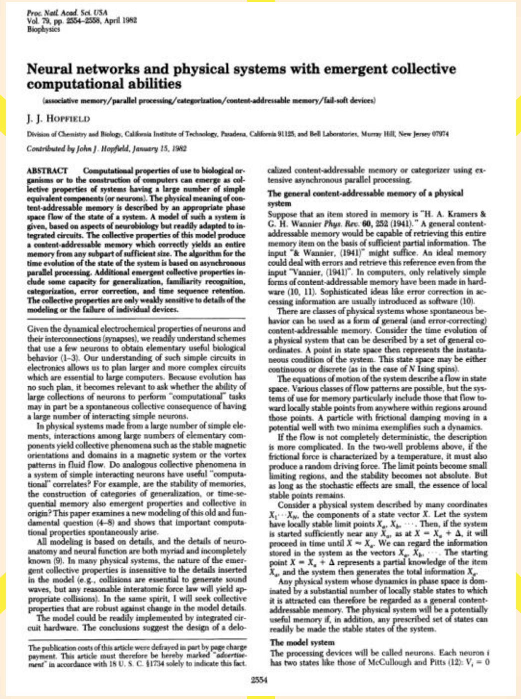
Then, a physicist to the rescue!

John Hopfield (physicist, chemist, biologist)

comes up with the next big thing.

1982 – hopfield networks





Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. Hopfield

Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125; and Bell Laboratories, Murray Hill, New Jersey 07974

Contributed by John J. Hopfield, January 15, 1982

ABSTRACT **Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of a system. A model of such a system is given, based on aspects of neurobiology but readily adapted to integrated circuits. The collective properties of this model produce a content-addressable memory which correctly yields an entire memory from any subpart of sufficient size. The algorithm for the time evolution of the state of the system is based on asynchronous parallel processing. Additional emergent collective properties include some capacity for generalization, familiarity recognition, categorization, error correction, and time sequence retention. The collective properties are only weakly sensitive to details of the modeling or the failure of individual devices.**

Given the dynamical electrochemical properties of neurons and their interconnections (synapses), we readily understand schemes that use a few neurons to obtain elementary useful biological behavior (1–3). Our understanding of such simple circuits in electronics allows us to plan larger and more complex circuits which are essential to large computers. Because evolution has no such plan, it becomes relevant to ask whether the ability of large collections of neurons to perform "computational" tasks may in part be a spontaneous collective consequence of having a large number of interacting simple neurons.

In physical systems made from a large number of simple elements, interactions among large numbers of elementary components yield collective phenomena such as the stable magnetic orientations and domains in a magnetic system or the vortex patterns in fluid flow. Do analogous collective phenomena in a system of simple interacting neurons have useful "computational" correlates? For example, are the stability of memories, the construction of categories of generalization, or time-sequential memory also emergent properties and collective in origin? This paper examines a new modeling of this old and fundamental question (4–8) and shows that important computational properties spontaneously arise.

All modeling is based on details, and the details of neuroanatomy and neural function are both myriad and incompletely known (9). In many physical systems, the nature of the emergent collective properties is insensitive to the details inserted in the model (e.g., collisions are essential to generate sound waves, but any reasonable interatomic force law will yield appropriate collisions). In the same spirit, I will seek collective properties that are robust against change in the model details.

The model could be readily implemented by integrated circuit hardware. The conclusions suggest the design of a delocalized content-addressable memory or categorizer using extensive asynchronous parallel processing.

The general content-addressable memory of a physical system

Suppose that an item stored in memory is "H. A. Kramers & G. H. Wannier *Phys. Rev.* **60**, 252 (1941)." A general content-addressable memory would be capable of retrieving this entire memory item on the basis of sufficient partial information. The input "& Wannier, (1941)" might suffice. An ideal memory could deal with errors and retrieve this reference even from the input "Vannier, (1941)". In computers, only relatively simple forms of content-addressable memory have been made in hardware (10, 11). Sophisticated ideas like error correction in accessing information are usually introduced as software (10).

There are classes of physical systems whose spontaneous behavior can be used as a form of general (and error-correcting) content-addressable memory. Consider the time evolution of a physical system that can be described by a set of general coordinates. A point in state space then represents the instantaneous condition of the system. This state space may be either continuous or discrete (as in the case of N Ising spins).

The equations of motion of the system describe a flow in state space. Various classes of flow patterns are possible, but the systems of use for memory particularly include those that flow toward locally stable points from anywhere within regions around those points. A particle with frictional damping moving in a potential well with two minima exemplifies such a dynamics.

If the flow is not completely deterministic, the description is more complicated. In the two-well problems above, if the frictional force is characterized by a temperature, it must also produce a random driving force. The limit points become small limiting regions, and the stability becomes not absolute. But as long as the stochastic effects are small, the essence of local stable points remains.

Consider a physical system described by many coordinates $X_1 \cdots X_N$, the components of a state vector X. Let the system have locally stable limit points $X_a, X_b, \cdots$. Then, if the system is started sufficiently near any X_a, as at $X = X_a + \Delta$, it will proceed in time until $X \approx X_a$. We can regard the information stored in the system as the vectors $X_a, X_b, \cdots$. The starting point $X = X_a + \Delta$ represents a partial knowledge of the item X_a, and the system then generates the total information X_a.

Any physical system whose dynamics in phase space is dominated by a substantial number of locally stable states to which it is attracted can therefore be regarded as a general content-addressable memory. The physical system will be a potentially useful memory if, in addition, any prescribed set of states can readily be made the stable states of the system.

The model system

The processing devices will be called neurons. Each neuron i has two states like those of McCullough and Pitts (12): $V_i = 0$

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U. S. C. §1734 solely to indicate this fact.

2554

1982 – hopfield networks

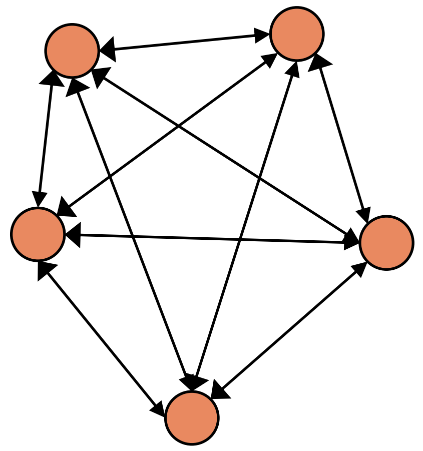
Like the Perceptron, Hopfield networks learn to recognize patterns. Unlike the Perceptron, they can still work in noisy conditions.

Operates over a fully connected graph, with symmetric edges.

Each node is connected to one input.

Learns the weights (edges) with Hebb's rule.

Prediction happens recursively.

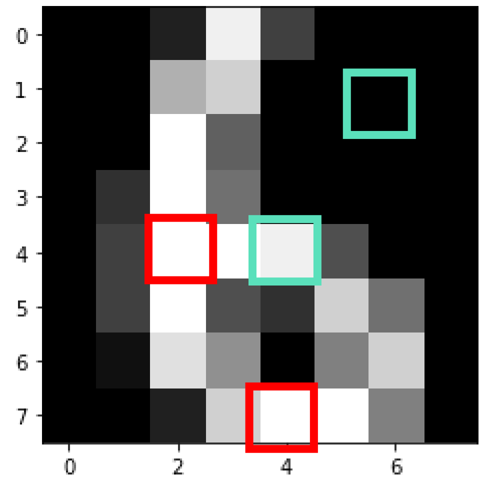


1982 – hopfield networks

For images, each pixel is assigned a node.

Weights between pixels i,j that have the same state are increased in strength (and decreased when pixels are different). Do this for every pattern p.

$$w_{ij} = \sum_p x_i^{(p)} x_j^{(p)}$$



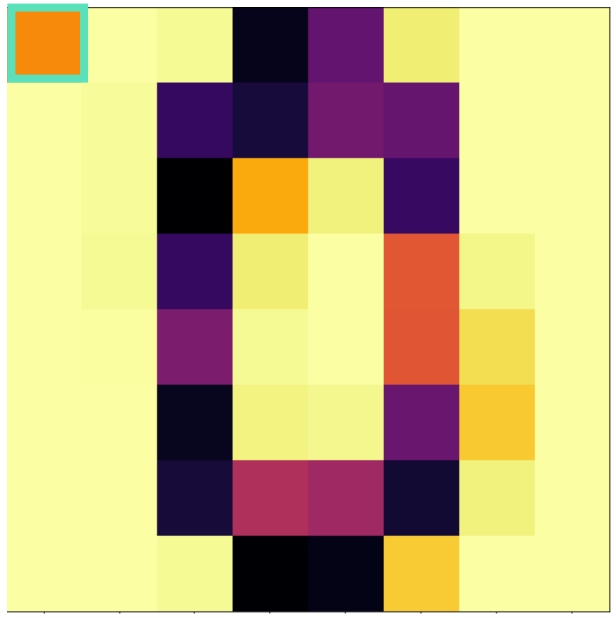
$w_{ij} = 1.0 * 1.0$

$w_{ij} = 1.0 * -1.0$

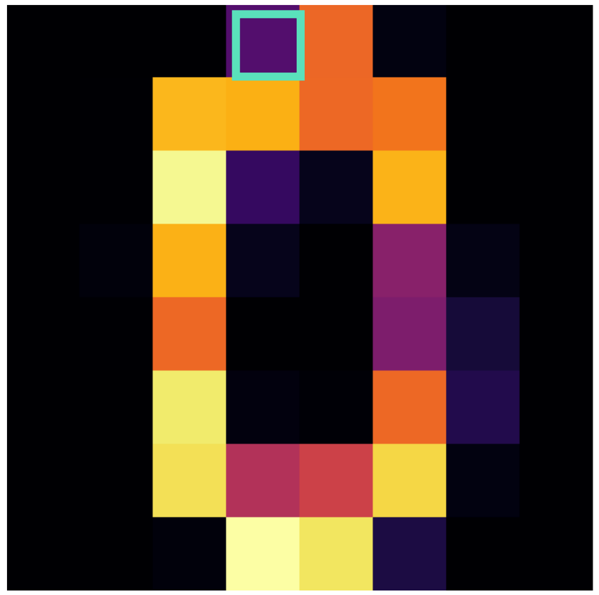
1982 – hopfield networks

If we teach a Hopfield network to learn zeros, here is what the weight matrix looks like

For neuron 1



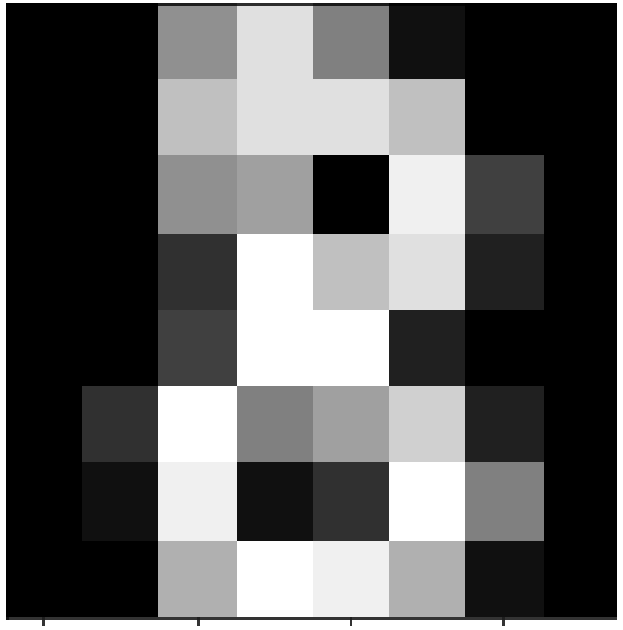
For neuron 4

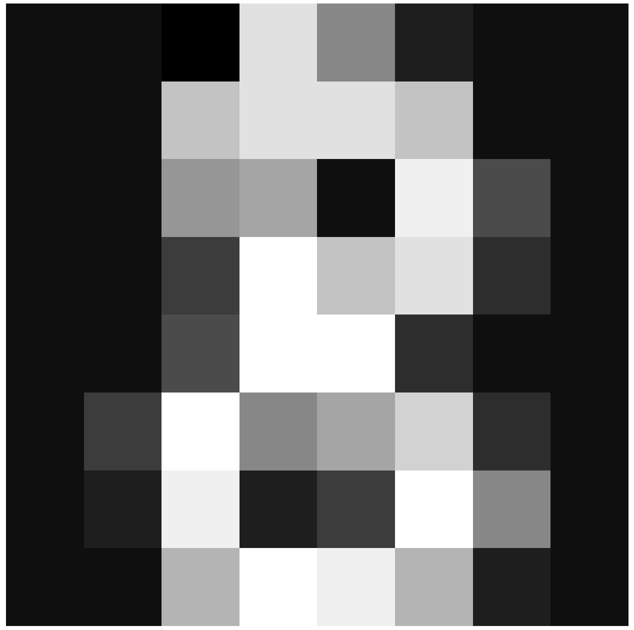


1982 – hopfield networks

If we feed a Hopfield network an 8, it will still converge to a stable energy state, a zero.

Viewed as a dynamical system, the zero state has become an “attractor”





Remind you of anything?

1982 – hopfield networks

Re-ignited interest in “connectionist” ideas as physicists, engineers, neuroscientists, etc. all saw their own ideas reflected in the algorithm.

Hebb’s rule – classic rule by which neurons learn

Energy based functions – typical way that physicists think of stateful problems

Content-addressable-memory – a sought after feature for building memory systems

Was one of the first “recurrent” neural networks, and was “unsupervised”

1982 – hopfield networks

Unfortunately, there was no killer application. Wasn’t a regressor, wasn’t a classifier. Was something different.

Capacity issues (couldn’t learn very many patterns) combined with 1980s computing power meant there was a real ceiling on what they could do.

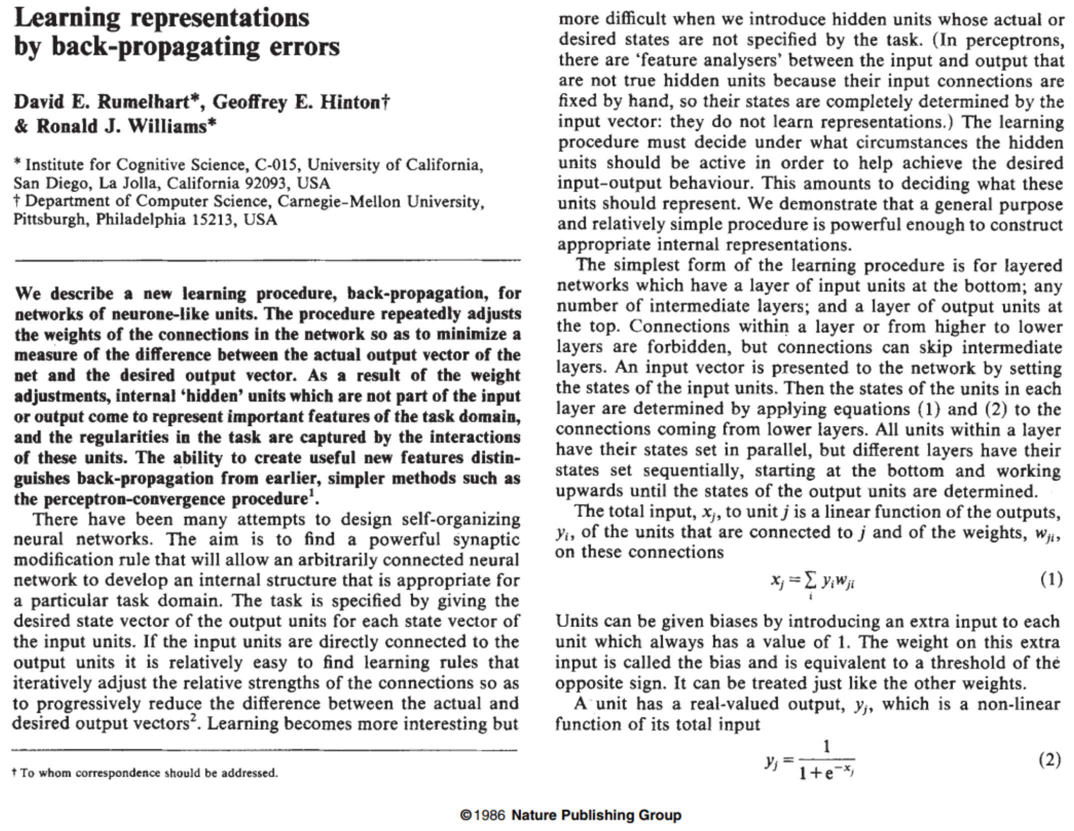
“Fully connected graph” architecture was not well suited to problems of the day like image recognition and time series forecasting.

1986 – backprop to the future

Scientists revisited the old set of perceptron challenges, namely, how do you update the weights of multi-layer perceptrons?

Inertia built across multiple people and institutions.

David Rumelhart (psychologist),
Geoffrey Hinton (psych, comp neuro, CS),
Ronald Williams (cog sci?)
published this landmark paper.



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.

1986 – backprop to the future

Suddenly, multi-layer perceptrons were all the rage again.

Backpropagation was widely applicable to many kinds of models, encouraging a new wave of research.

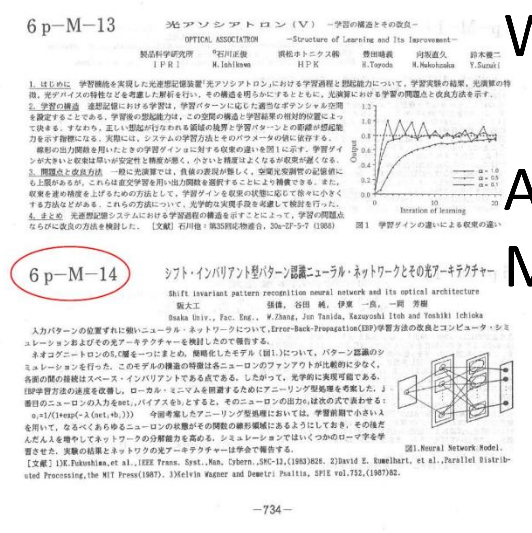
Apply the chain rule to partial derivatives to find each weight’s gradient with respect to the error.

1988 – convolutional neural networks

Hubel and Wiesel's famous experiments in 1959 on the visual cortex inspired the Neocognitron in 1979.

Now equipped with backpropagation, the CNN was born.

Proceedings of annual conference of the Japan Society of Applied Physics, Sept. 1988



6 p−M−13 光アソシアトロン（V） −学習の構造とその改良−

OPTICAL ASSOCIATRON −Structure of Learning and Its Improvement−

製品科学研究所 ○石川正俊 浜松ホトニクス㈱ 豊田晴義 向坂直久 鈴木義二

IPRI M.Ishikawa HPK H.Toyoda N.Mukohzaka Y.Suzuki

1. はじめに 学習機能を実現した光連想記憶装置「光アソシアトロン」における学習過程と想起能力について，学習実験の結果，光演算の特徴，光デバイスの特性などを考慮した解析を行い，その構造を明らかにするとともに，光演算における学習の問題点と改良方法を示す．

2. 学習の構造 連想記憶における学習は，学習パターンに応じた適当なポテンシャル空間を設定することである．学習後の想起能力は，この空間の構造と学習結果の相対的位置によって決まる．すなわち，正しい想起が行なわれる領域の境界と学習パターンとの距離が想起能力を示す指標になる．実際には，システムの学習方法とそのパラメータの値に依存する．

線形の出力関数を用いたときの学習ゲインαに対する収束の違いを図1に示す．学習ゲインが大きいと収束は早いが安定性と精度が悪く，小さいと精度はよくなるが収束が遅くなる．

3. 問題点と改良方法 一般に光演算では，負値の表現が難しく，空間光変調管の記憶値にも上限があるが，これらは直交学習を用い出力関数を選択することにより補償できる．また，収束を速め精度を上げるための方法として，学習ゲインを収束の状態に応じて徐々に小さくする方法などがある．これらの方法について，光学的な実現手段を考慮して検討を行った．

4. まとめ 光連想記憶システムにおける学習過程の構造を示すことによって，学習の問題点ならびに改良の方法を検討した． [文献] 石川他：第35回応物連合，30a-ZF-5-7 (1988)

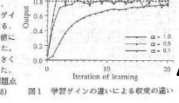


図1 学習ゲインの違いによる収束の違い

6 p−M−14 シフト・インバリアント型パターン認識ニューラル・ネットワークとその光アーキテクチャー

Shift invariant pattern recognition neural network and its optical architecture

阪大工 張偉，谷田 純，伊東 一良，一岡 芳樹

Osaka Univ., Fac. Eng., W.Zhang, Jun Tanida, Kazuyoshi Itoh and Yoshiki Ichioka

入力パターンの位置ずれに強いニューラル・ネットワークについて，Error-Back-Propagation(EBP)学習方法の改良とコンピュータ・シミュレーションおよびその光アーキテクチャーを検討したので報告する．

ネオコグニートロンのS,C層を一つにまとめ，簡略化したモデル（図1.）について，パターン認識のシミュレーションを行った．このモデルの構造の特徴は各ニューロンのファンアウトが比較的に少なく，各面の間の接続はスペース・インバリアントである点である．したがって，光学的に実現可能である．EBP学習方法の速度を改善し，ローカル・ミニマムを回避するためにアニーリング型処理を考案した．j番目のニューロンの入力をnet_j，バイアスをb_jとすると，そのニューロンの出力o_jは次の式で表わせる：

$o_j=1/(1+\exp(-\lambda(net_j+b_j)))$ 今回考案したアニーリング型処理においては，学習前期で小さいλを用いて，なるべくあらゆるニューロンの状態がその関数の線形領域にあるようにしておき，その後だんだんλを増やしてネットワークの分解能力を高める．シミュレーションではいくつかのローマ字を学習させた．実験の結果とネットワクの光アーキテクチャーは学会で報告する．

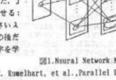


図1.Neural Network Model.

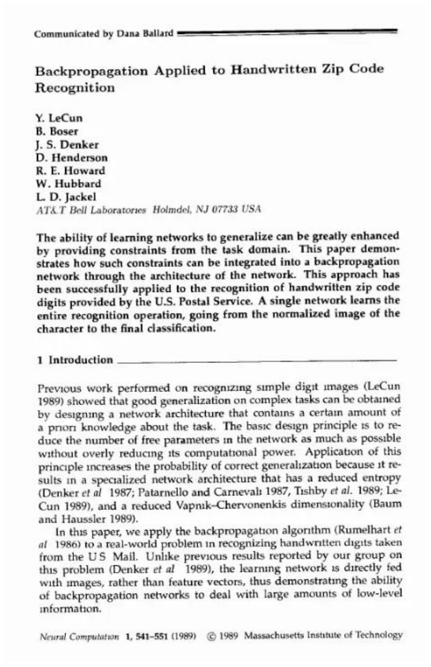
[文献] 1)K.Fukushima,et al.,IEEE Trans. Syst.,Man, Cybern.,SMC-13,(1983)826. 2)David E. Rumelhart, et al.,Parallel Distributed Processing,the MIT Press(1987). 3)Kelvin Wagner and Demetri Psaltis, SPIE vol.752,(1987)82.

−734−

Wei Zhang, 1988

Alphabet detection

Medical imaging



Communicated by Dana Ballard

Backpropagation Applied to Handwritten Zip Code Recognition

Y. LeCun
B. Boser
J. S. Denker
D. Henderson
R. E. Howard
W. Hubbard
L. D. Jackel
AT&T Bell Laboratories Holmdel, NJ 07733 USA

The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain. This paper demonstrates how such constraints can be integrated into a backpropagation network through the architecture of the network. This approach has been successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service. A single network learns the entire recognition operation, going from the normalized image of the character to the final classification.

1 Introduction

Previous work performed on recognizing simple digit images (LeCun 1989) showed that good generalization on complex tasks can be obtained by designing a network architecture that contains a certain amount of a priori knowledge about the task. The basic design principle is to reduce the number of free parameters in the network as much as possible without overly reducing its computational power. Application of this principle increases the probability of correct generalization because it results in a specialized network architecture that has a reduced entropy (Denker *et al* 1987; Patarnello and Carnevali 1987, Tishby *et al.* 1989; LeCun 1989), and a reduced Vapnik–Chervonenkis dimensionality (Baum and Haussler 1989).

In this paper, we apply the backpropagation algorithm (Rumelhart *et al* 1986) to a real-world problem in recognizing handwritten digits taken from the U S Mail. Unlike previous results reported by our group on this problem (Denker *et al* 1989), the learning network is directly fed with images, rather than feature vectors, thus demonstrating the ability of backpropagation networks to deal with large amounts of low-level information.

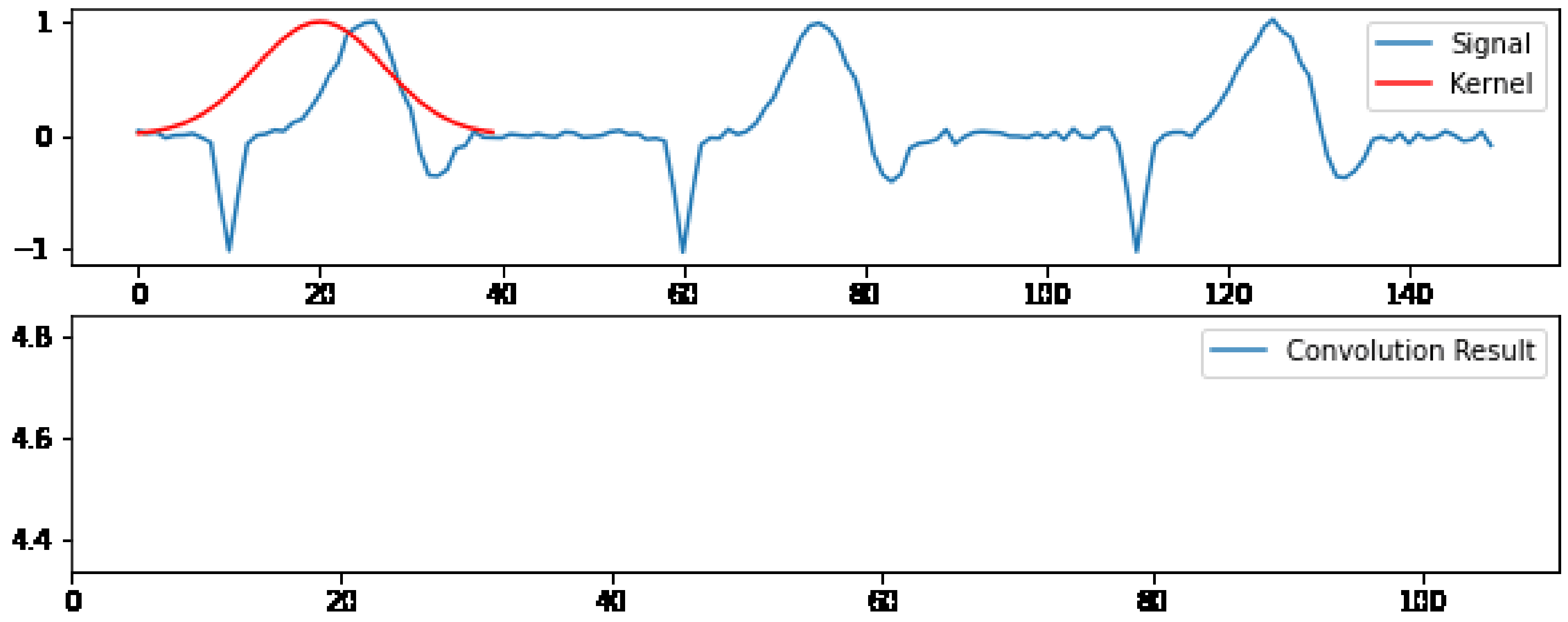
Neural Computation **1, 541–551** (1989) © 1989 Massachusetts Institute of Technology

Yann LeCun, 1989

Optical character recognition (OCR)

1988 – convolutional neural networks

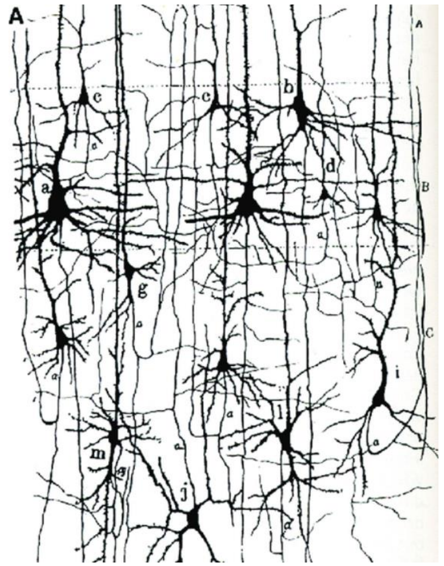
A convolution is when a signal is convolved with a *kernel* or *filter*

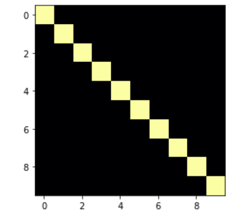


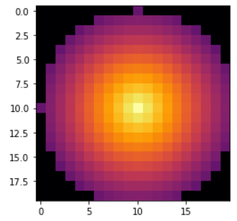
1988 – convolutional neural networks

A convolutional neural net simply *learns* those kernels, to solve some task.

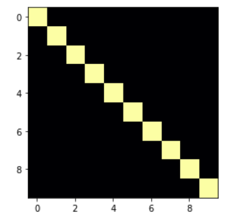
- Translation invariant
- Parameter sharing + pooling = efficiency on images

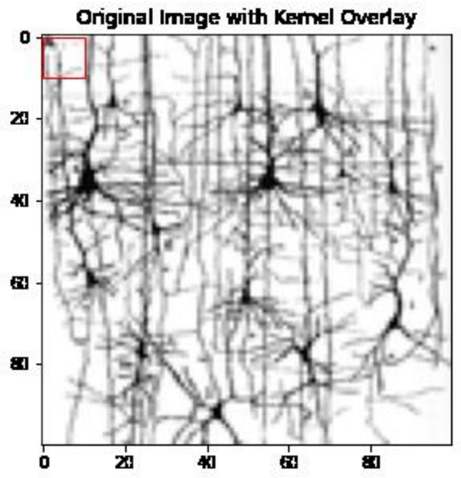


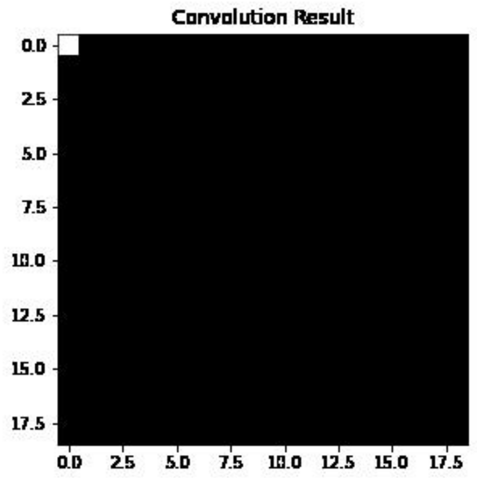

Edge kernel

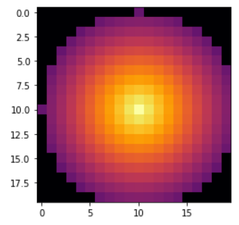

RBF kernel

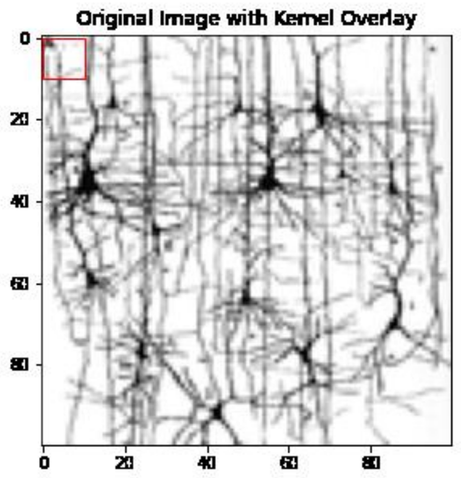
1988 – convolutional neural networks

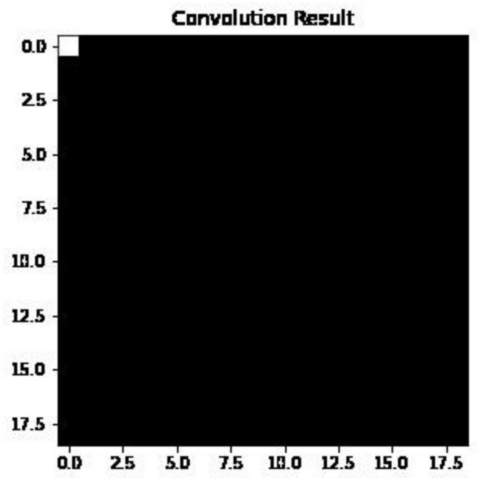












Can also stack convolutions on each other.

CNNs tend to learn primitive features like edges first, then learn more abstract features like “cat” in deeper layers.

Hubel and Wiesel showed that the visual cortex operates in much the same way, 30+ years prior!

1990s – here we go again

Demand for AI hardware drove innovations in massively parallel computers.

US and Japan competed to make the most powerful AI computing systems.

Symbolic AI was the dominant paradigm, and was making its move on corporate America.

1990s – the second AI winter

Much like the first AI winter, expectations exceeded reality. No real world market for AI.

Symbolic AI / expert systems couldn't handle real world data.

Connectionism was incredibly expensive and lacked power.

Bayesian statisticians got their time to shine.

Chip Group Urges Japan Sanctions

Belgium Bidder Must List Allies

2 Tax System Changes Favored by Greenspan

Ways to cut deficits and spur savings are seen.

U.S.-Built Vehicle Sales Surged in Late February

Productivity Gain for 1987 After Revision

Butler Is Seeking Larger Bear Stake

Gains Are Slow for Artificial Intelligence Industry

'Everyone was planning on growth that was unsustainable.'

Mexico Accepts $3.67 Billion In Debt Exchange for Bonds

"good academicians often do not make good business people"

'Most corporate programs have failed to fulfill their promise''

''People believed their own hype''

https://www.nytimes.com/1988/03/04/business/setbacks-for-artificial-intelligence.html

1990s – the second AI winter

Symbolic AI was largely abandoned by researchers.

Connectionism research continued on, albeit quietly. Several notable advances were

- Q-learning in reinforcement learning
- Support Vector Machines
- Long Short Term Memory networks

Probability and statistics exerted their influence on ML

2000s – the rise of big data

By 1996, storing data digitally was cheaper than on paper.

The internet grew up, and data became a commodity for online companies.

Still, machine learning research remained in the academic sphere.

2012 – AlexNet

Alex Krizhevsky and Ilya Sutskever entered the ImageNet competition.

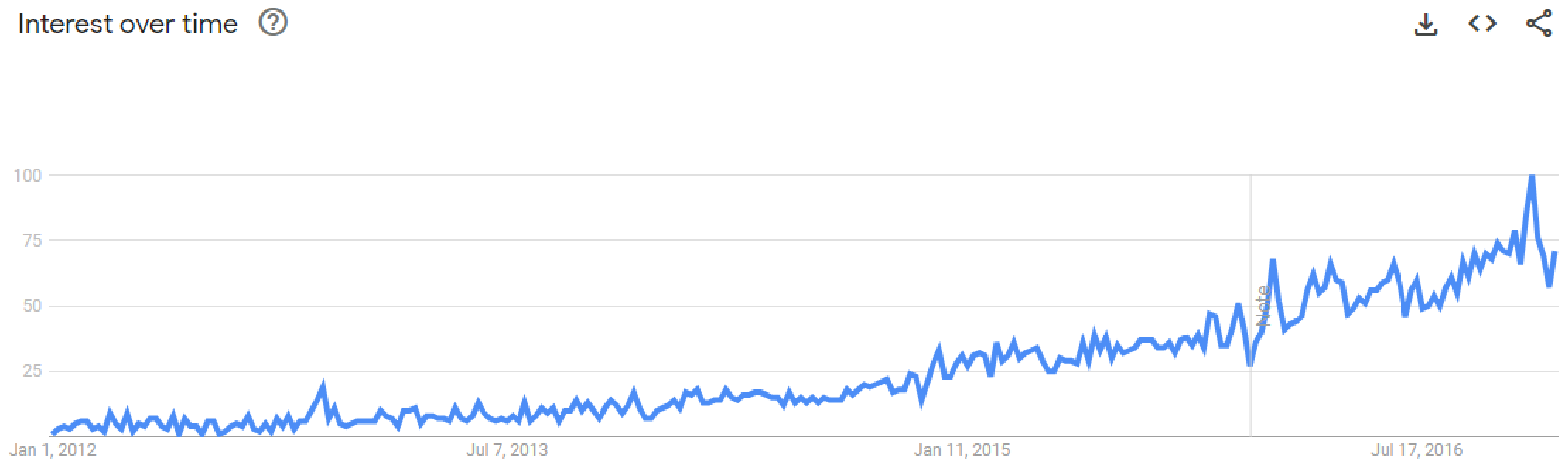
SVMs, segmentation methods, kernel methods, etc. were winners of prior years competitions.

They wanted to use a many-layered neural network, but training it on a million images would have taken months. So they used a pair of Nvidia GTX 580s, the current best parallel processing hardware available.

2012 – AlexNet

AlexNet won the competition handedly, with a top-5 error of 0.15. The next nearest was 0.26!

Amazingly, AlexNet had 62 *million* parameters over 8 layers. Google snatched them right up, and the term “deep learning” was coined.



2012 – 2016 – industry adoption

Tech companies all jumped on to the “deep learning” bandwagon.

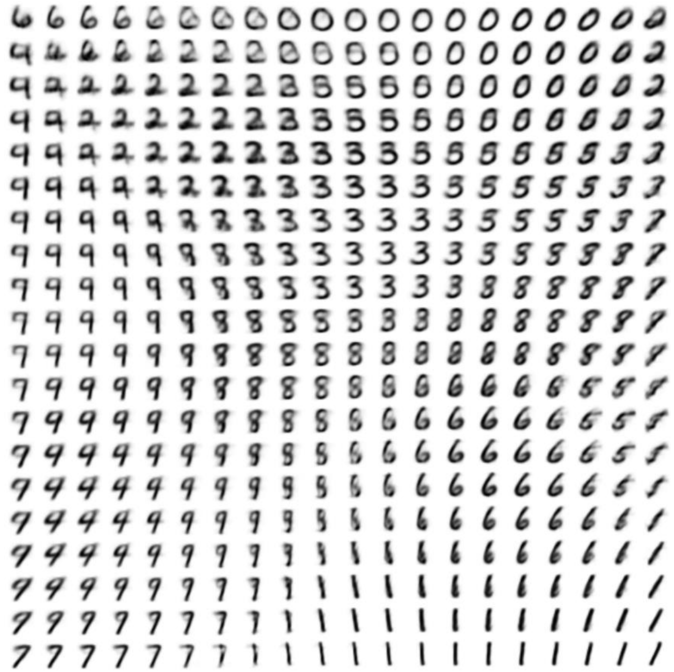
Access to GPU clusters and large datasets put industrial research on equal footing with academics.

Finally, machine learning found its market fit.

2014 – generative AI revival

Generative Adversarial Networks, first described in 1991, were revived by Ian Goodfellow.

Variational Autoencoders, an application of Bayesian theory to deep learning, also revived.



2017 – _____ is all you need

Transformer paper published by Google Brain.

Transformers dominated for autoregressive and sequence analysis tasks, like translation.

Transformer-style models have roots in the 90s. The “attention” method was built on prior work in RNNs. Merging the two methods on large datasets resulted in impressive results.

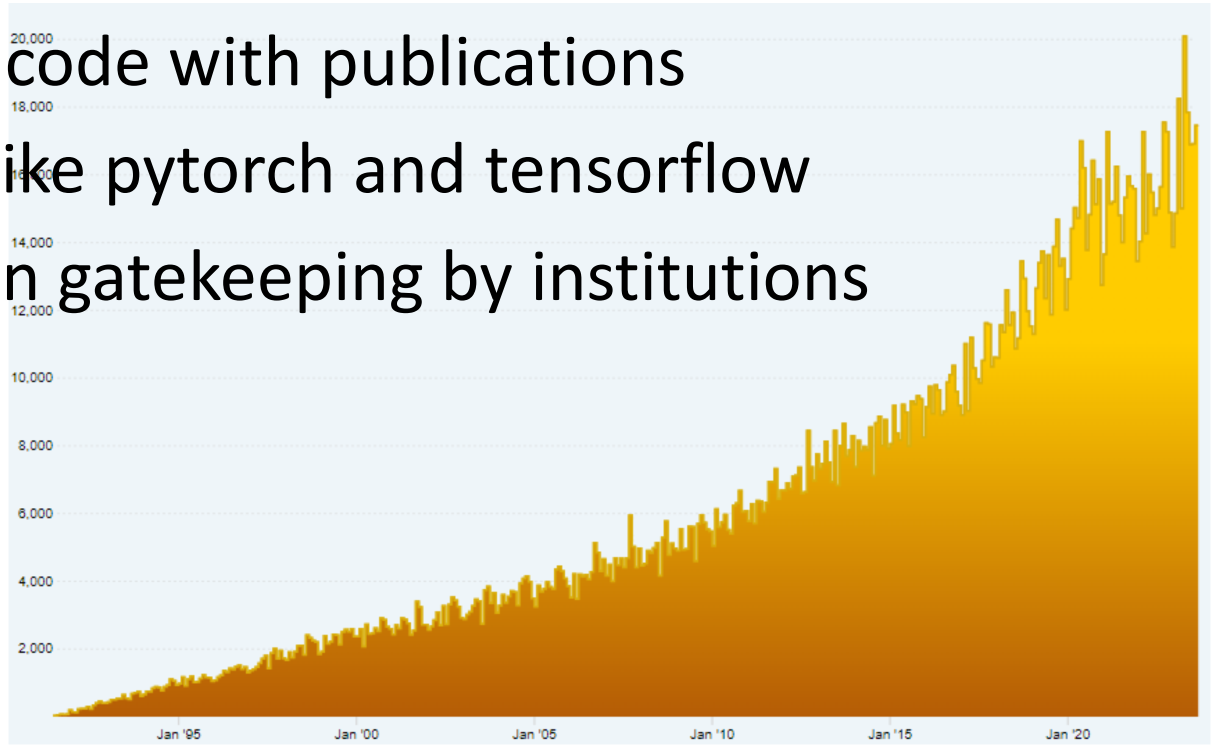
One problem though, the transformer architecture is data hungry and computationally demanding.

also started a super obnoxious naming trend

2018 to today – the diaspora

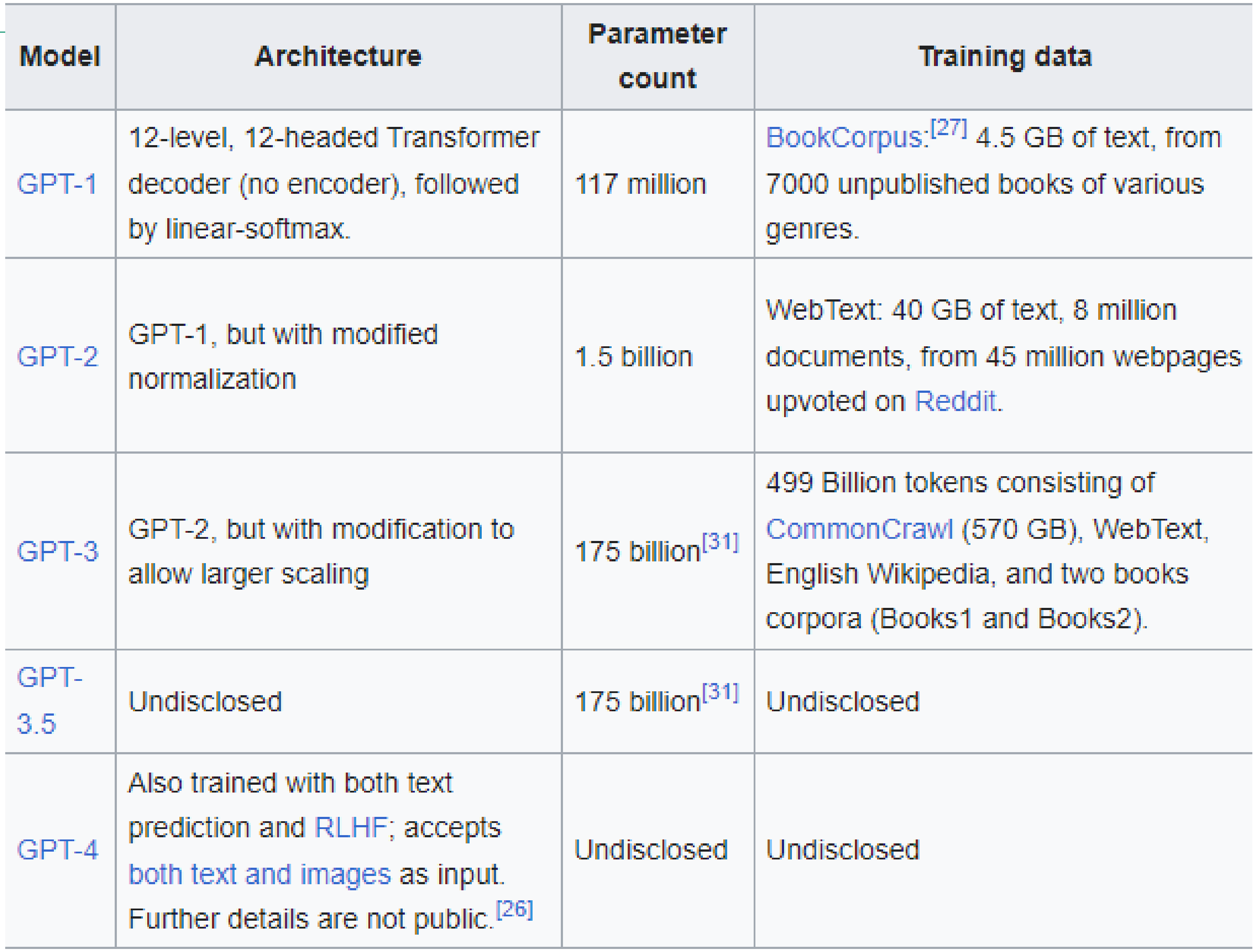
Machine learning became a more accessible field of research to enter

- arXiv made research available for everyone across the world
- availability of supplementary code with publications
- development of ML libraries like pytorch and tensorflow
- open source approach overran gatekeeping by institutions



2018 to today – GPT

GPT (generative pre-trained transformer) models are essentially transformers, with the bells and whistles that come with big data pretraining tasks.



Model	Architecture	Parameter count	Training data
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	117 million	BookCorpus:[27] 4.5 GB of text, from 7000 unpublished books of various genres.
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.
GPT-3	GPT-2, but with modification to allow larger scaling	175 billion[31]	499 Billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2).
GPT-3.5	Undisclosed	175 billion[31]	Undisclosed
GPT-4	Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public.[26]	Undisclosed	Undisclosed

2022 to today – Stable Diffusion

Diffusion models have their roots in earlier frameworks, like Hopfield Networks.

Stable Diffusion was first to train such a model on billions of images. Beats GANs in image generation capabilities.

Generative techniques are widely applicable to other domains like audio, video, 3d renderings

research landscape today

Research is highly fragmented into specialties by industry and model architectures.

Computational chemistry uses graph neural networks.

Self-driving cars use computer vision.

LLM research uses large pretrained models with finetuning methods.

Financial industry uses uncertainty modeling and time series methods

And so on and so forth.

research landscape today

The accessibility of machine learning research comes with risks

- Focus on public benchmarks incentivizes chasing metrics for glory
- Although code is often provided, it is rarely plug-and-play
- Most papers have novelty that is domain specific or narrow in scope
- Quality of research as a whole suffers without peer review

reading machine learning papers

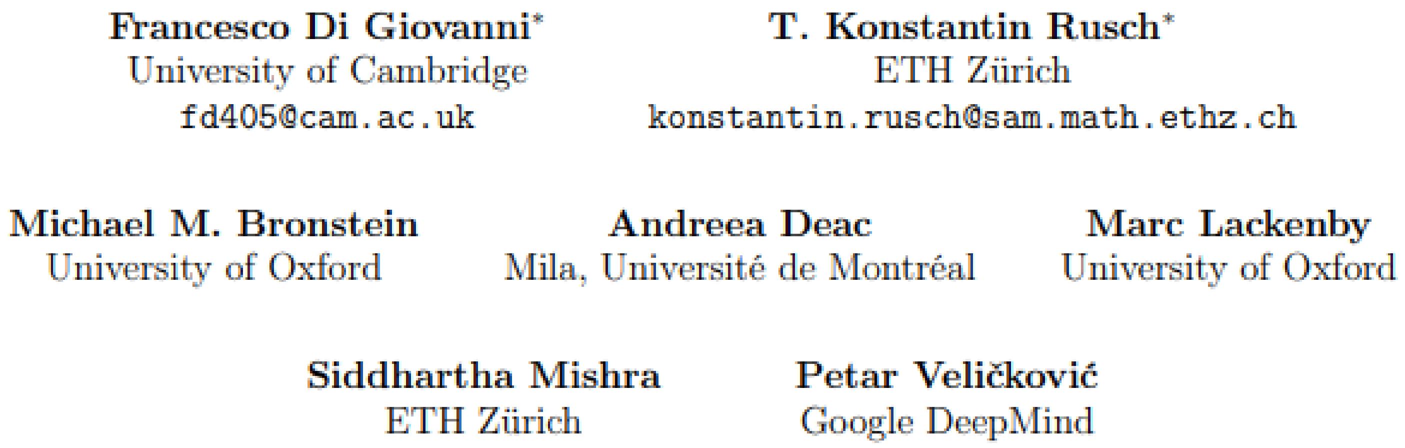
Some tips for reading research papers.

I do most of my literature searches with simple google searches. Finding the right keywords can sometimes be nontrivial.

Let’s say I want to understand more about Graph Neural Networks. This paper comes up: https://arxiv.org/abs/2306.03589

step 1: authors, institutions, publication

Before you do anything, check the authors and the institutions. Have you heard of them? Do you trust them?


Francesco Di Giovanni*
University of Cambridge
fd405@cam.ac.uk

T. Konstantin Rusch*
ETH Zürich
konstantin.rusch@sam.math.ethz.ch

Michael M. Bronstein
University of Oxford

Andreea Deac
Mila, Université de Montréal

Marc Lackenby
University of Oxford

Siddhartha Mishra
ETH Zürich

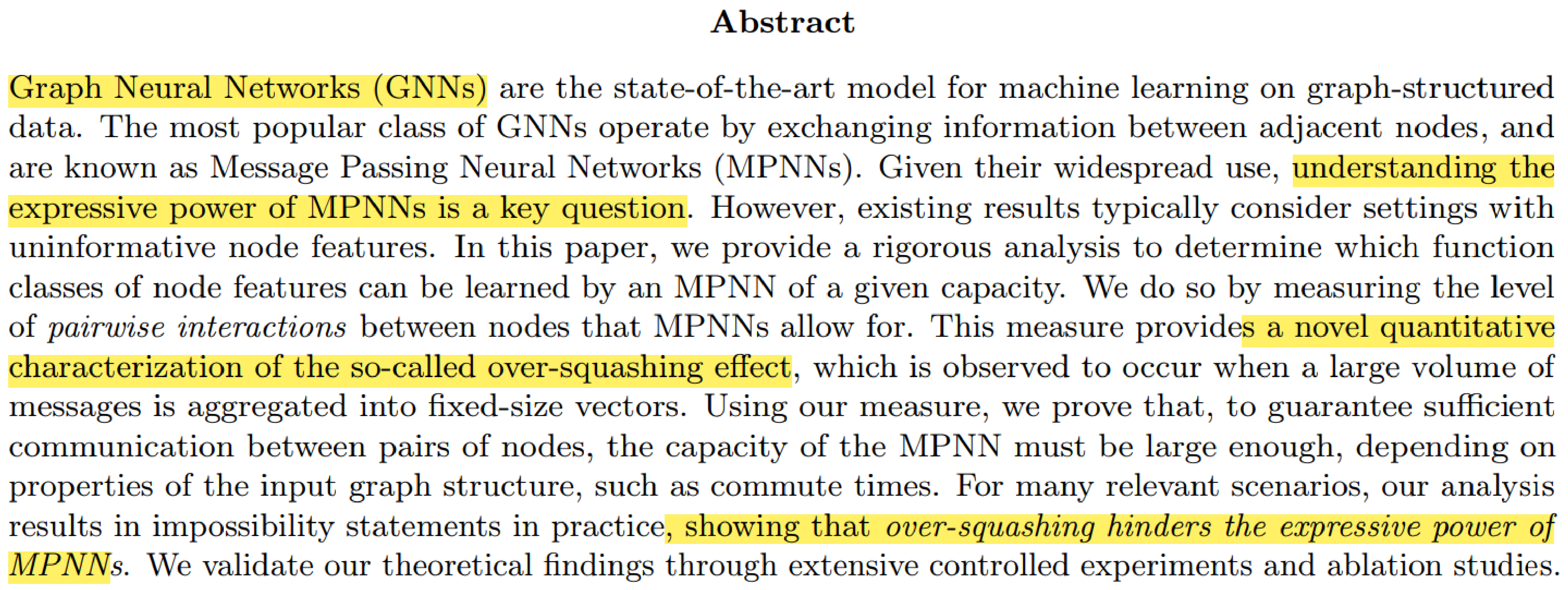
Petar Veličković
Google DeepMind

Yes! These are all heavy hitters in ML research.

But, this paper is in arXiv, which is not peer-reviewed. This means you should be very critical, regardless of the authors.

step 2: read the abstract

Read the abstract. You will probably not understand everything at first, but if the paper is well written, it will tell you if it is relevant to your interests.



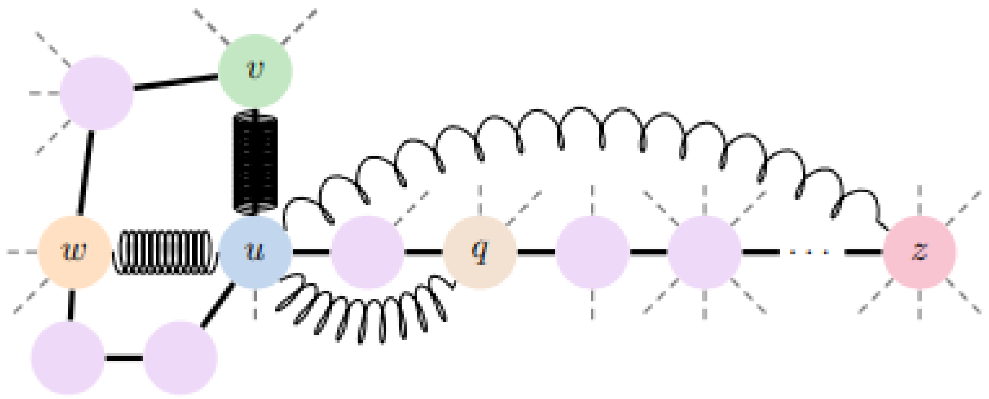
Abstract

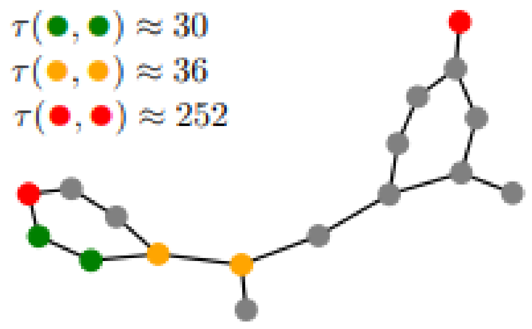
Graph Neural Networks (GNNs) are the state-of-the-art model for machine learning on graph-structured data. The most popular class of GNNs operate by exchanging information between adjacent nodes, and are known as Message Passing Neural Networks (MPNNs). Given their widespread use, understanding the expressive power of MPNNs is a key question. However, existing results typically consider settings with uninformative node features. In this paper, we provide a rigorous analysis to determine which function classes of node features can be learned by an MPNN of a given capacity. We do so by measuring the level of *pairwise interactions* between nodes that MPNNs allow for. This measure provides a novel quantitative characterization of the so-called over-squashing effect, which is observed to occur when a large volume of messages is aggregated into fixed-size vectors. Using our measure, we prove that, to guarantee sufficient communication between pairs of nodes, the capacity of the MPNN must be large enough, depending on properties of the input graph structure, such as commute times. For many relevant scenarios, our analysis results in impossibility statements in practice, showing that *over-squashing hinders the expressive power of MPNNs*. We validate our theoretical findings through extensive controlled experiments and ablation studies.

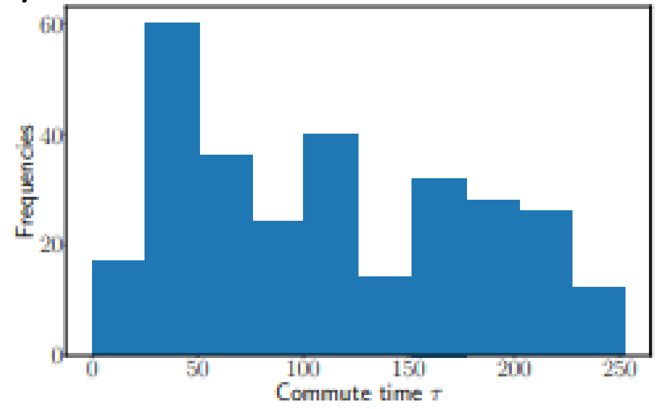
step 3: skim the figures

Do NOT read the introduction next! Instead, skim through the figures and captions of the paper. A good paper worth your read conveys its main message just in the figures.

Commute time is a new term to me, but it seems important here, so I should pay attention to that.







Modeling the relationship between nodes with springs is novel and interesting!

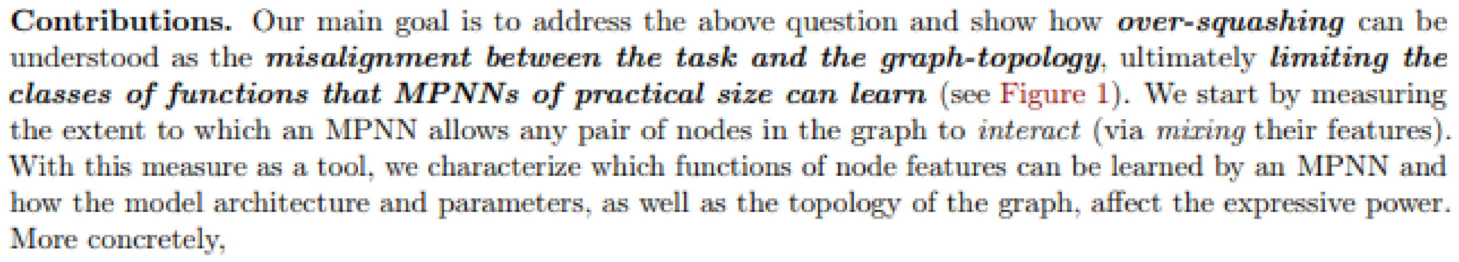
I'm gathering the main takeaway of this paper is really that GNNs need to have sufficient capacity in order to realize their full potential. Very relevant!

step 4: lightly read the introduction

Depending on your knowledge of that research domain, you could skim through to the *end* of the introduction. That is where authors describe the *novelty* of their method.

If you are reading to learn more about the domain, then read the introduction line by line. Go look up the references for everything you don’t understand. Skim those papers to get an understanding of why they matter.

step 4: lightly read the introduction



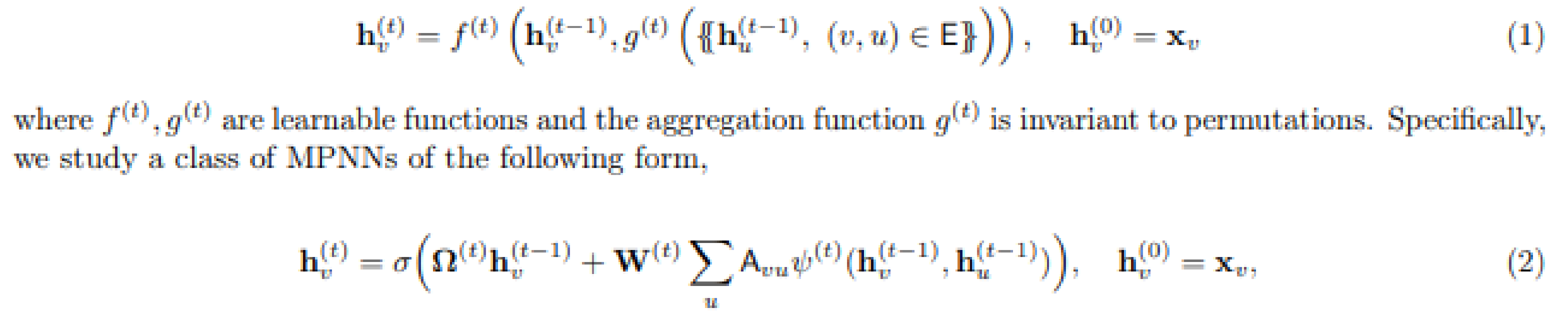
Contributions. Our main goal is to address the above question and show how ***over-squashing*** can be understood as the ***misalignment between the task and the graph-topology***, ultimately ***limiting the classes of functions that MPNNs of practical size can learn*** (see Figure 1). We start by measuring the extent to which an MPNN allows any pair of nodes in the graph to *interact* (via *mixing* their features). With this measure as a tool, we characterize which functions of node features can be learned by an MPNN and how the model architecture and parameters, as well as the topology of the graph, affect the expressive power. More concretely,

Authors did a good job at calling out their specific contributions. Always look for this!

Every citation is a full paper that cannot be summarized in a single sentence. It is common practice to set up a strawman with citations to make your own work look more impactful. Ask yourself – did this author accurately interpret those citations?

step 5: lightly read the rest of the paper

You’ve read the introduction, you know what the authors are trying to do. Read the rest of the paper, but don’t get hung up on things you don’t understand quite yet.



$$\mathbf{h}_v^{(t)} = f^{(t)}\left(\mathbf{h}_v^{(t-1)}, g^{(t)}\left(\{\!\{\mathbf{h}_u^{(t-1)},\ (v,u)\in \mathsf{E}\}\!\}\right)\right), \quad \mathbf{h}_v^{(0)} = \mathbf{x}_v \tag{1}$$

where $f^{(t)}, g^{(t)}$ are learnable functions and the aggregation function $g^{(t)}$ is invariant to permutations. Specifically, we study a class of MPNNs of the following form,

$$\mathbf{h}_v^{(t)} = \sigma\left(\boldsymbol{\Omega}^{(t)}\mathbf{h}_v^{(t-1)} + \mathbf{W}^{(t)}\sum_u \mathsf{A}_{vu}\psi^{(t)}(\mathbf{h}_v^{(t-1)},\mathbf{h}_u^{(t-1)})\right), \quad \mathbf{h}_v^{(0)} = \mathbf{x}_v, \tag{2}$$

80% of the time this level of notation is unnecessary for understanding an ML paper. Don’t bother reading it if it is rehashing old concepts or proofs. You can always come back to it later.

step 5: lightly read the rest of the paper

Often, the methods will be pretty obtuse. Don’t worry, its not you, its them. Rarely is a paper written so it can be understood easily.

For this paper, I read the section headers, and decided to skip to Section 4.

4 Over-squashing limits the expressive power of MPNNs

After skimming it, it seems they make their own metric. When the metric is large, the model is over-squashing. When its small, the model is expressing. Carry on!

They are interested in testing it in two scenarios – making models deeper or making models wider.

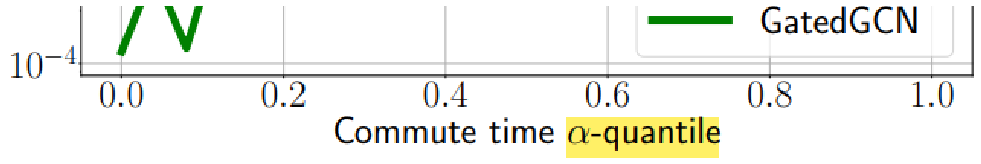
step 5: lightly read the rest of the paper

5 Experimental validation of the theoretical results

Experimental results time! If these don’t exist, I may not even read the paper. Deducing the value of the paper’s findings is difficult without some kind of experiment.

Here I am usually looking for a few things
- What is the dataset used
- What are the model architectures, training tasks, and loss functions
- What are the comparisons being made

I will make a note on the key notations used, so I can go back and figure them out.



step 6: read the conclusion

Nowadays, ML papers are not typically divided into traditional categories. Still, I want to read the authors’ summary of their paper. If I look at the end, I will often find this, along with a description of the limitations of their approach.

If there is no limitations section, feel free to boo.

In this paper, it is all inside the Discussion section. The discussion usually rehashes the introduction, and then adds the authors interpretation of the results in the context of their cited works.

step 7: (optional) read it all again, slower

You should already have a good idea of the central claims, the methods, and a sense of how much you believe in it. Often, that is enough to help you on your way (feel free to cite it now!).

If your motivation is deeper, now is when you go back and learn all that notation you missed. Don’t feel compelled to understand it *all.*

Write down or discuss your thoughts on the paper with your colleagues, everyone catches something different. You can learn it *much* faster this way.

your research presentation

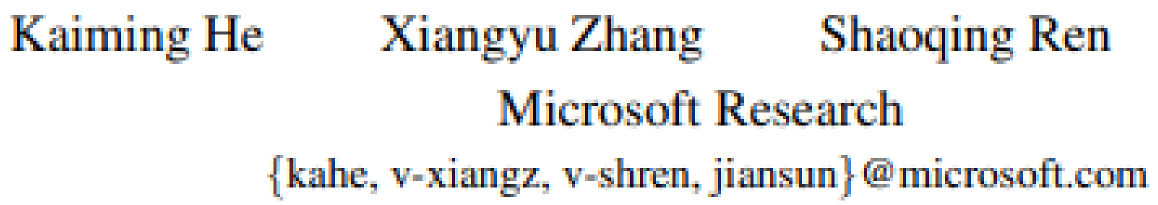
You will present a research paper to your classmates.

You will have 15 minutes to present, followed by a 5 min Q&A.

You will give and receive constructive evaluations for your peers.

Deep Residual Learning for Image Recognition

2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)



Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun

Microsoft Research

{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Introduction

- This paper is about deep residual learning for image recognition
- This technique got 1st place in ImageNet and COCO tasks
- Residual connections are also called shortcuts

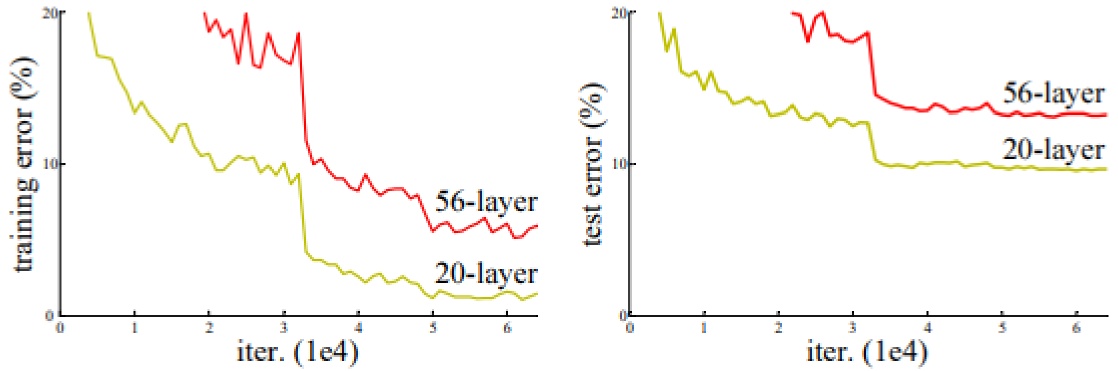


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Related Work

VLAD, Fisher Vectors, and Multigrid methods have also been used on residual connections

Lots of papers have studied shortcuts and tried many things like gated shortcuts

In this paper, shortcuts are always open

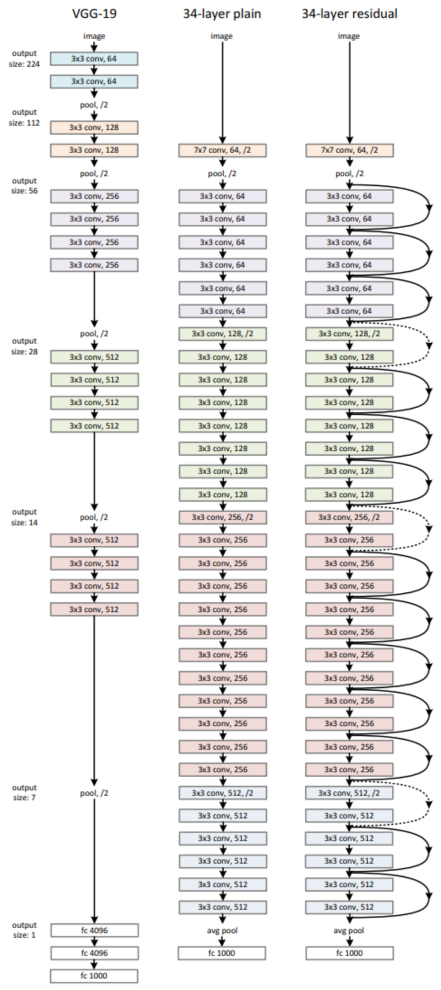
Methods

Model architecture.

CNNs stacked on each other.

Residual connections added in.

VGG nets don't have that.



Methods

They used the following implementation

Images randomly cropped and sampled at 224x224 pixels
Per-pixel mean subtracted
Standard color augmentation
Batch normalization after each convolution
Train from scratch
Use SGD with mini-batch size 256
Learning rate starts at 0.1 and is divided by 10 when the error plateaus
Trained 600,000 iterations with weight decay 0.0001
No dropout

Experiments

Evaluated on ImageNet 2012 classification datasets

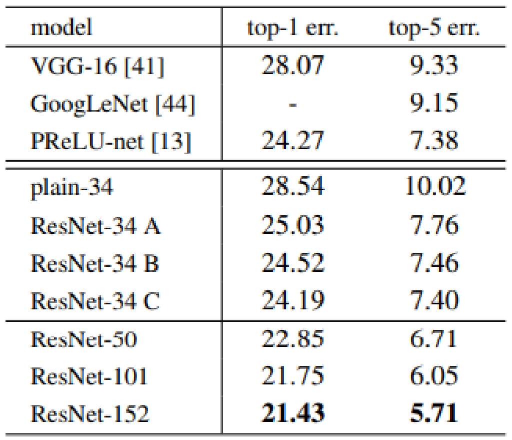
- 1000 classes

Evaluated with 18 layer and 34 layer models

Results

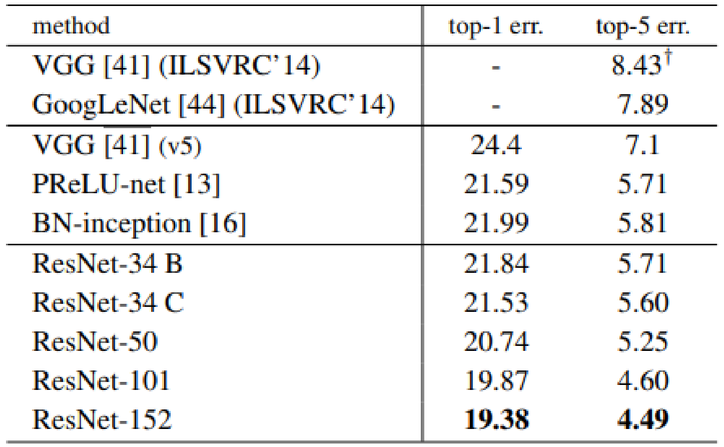
18 layer model did better than 34 layers model due to degradation

When residual connections were added, the 34 layer model did better than the 18 layer model, solving degradation.



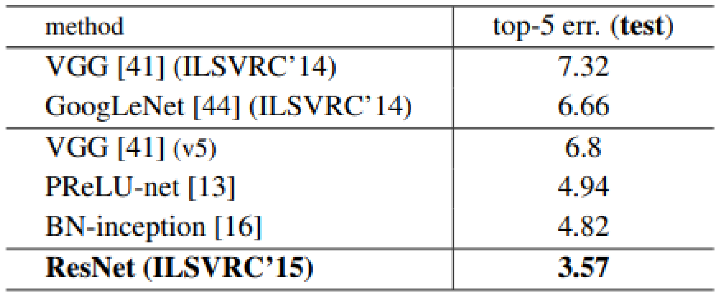
model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43[†]
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

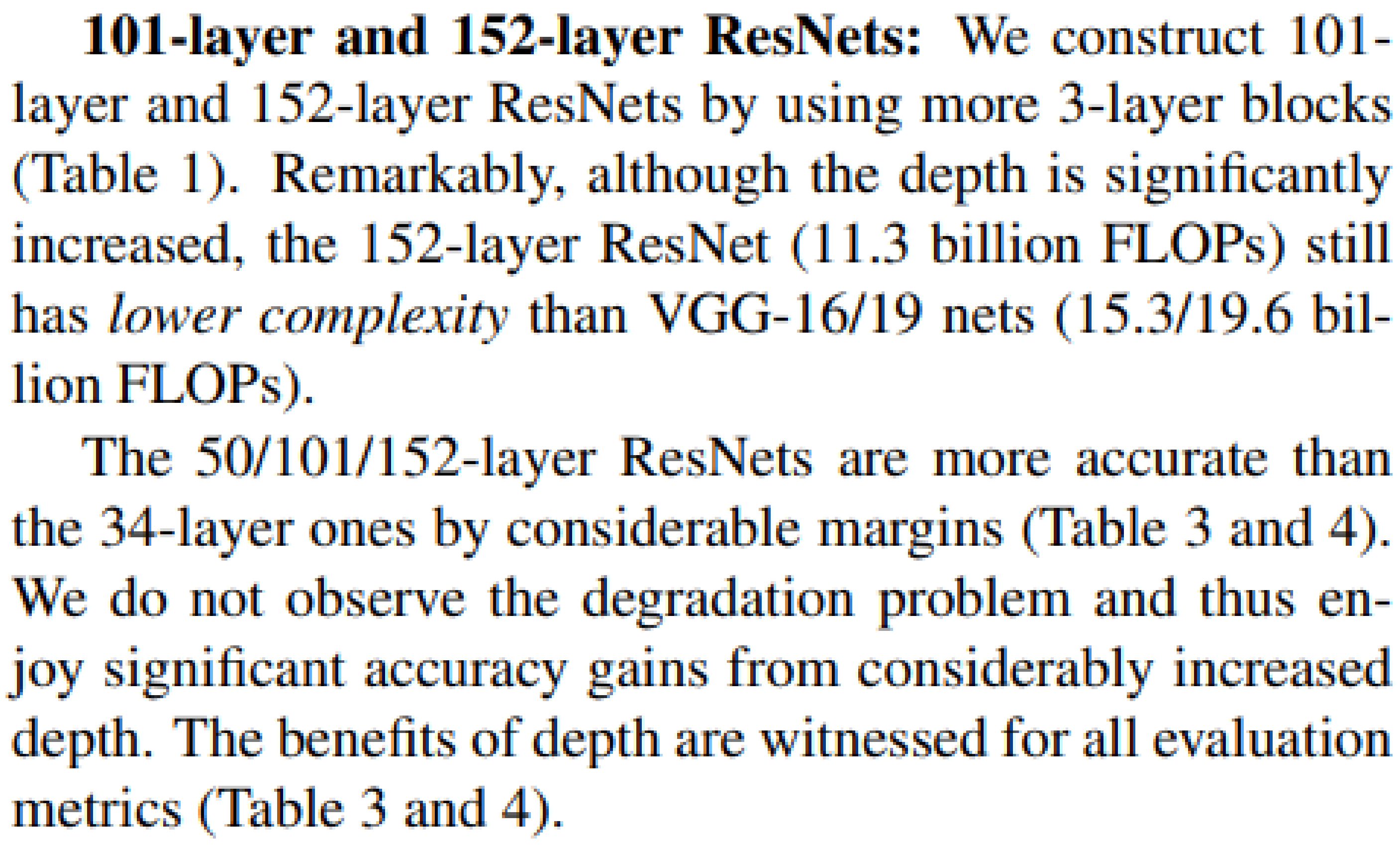
Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except [†] reported on the test set).



method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

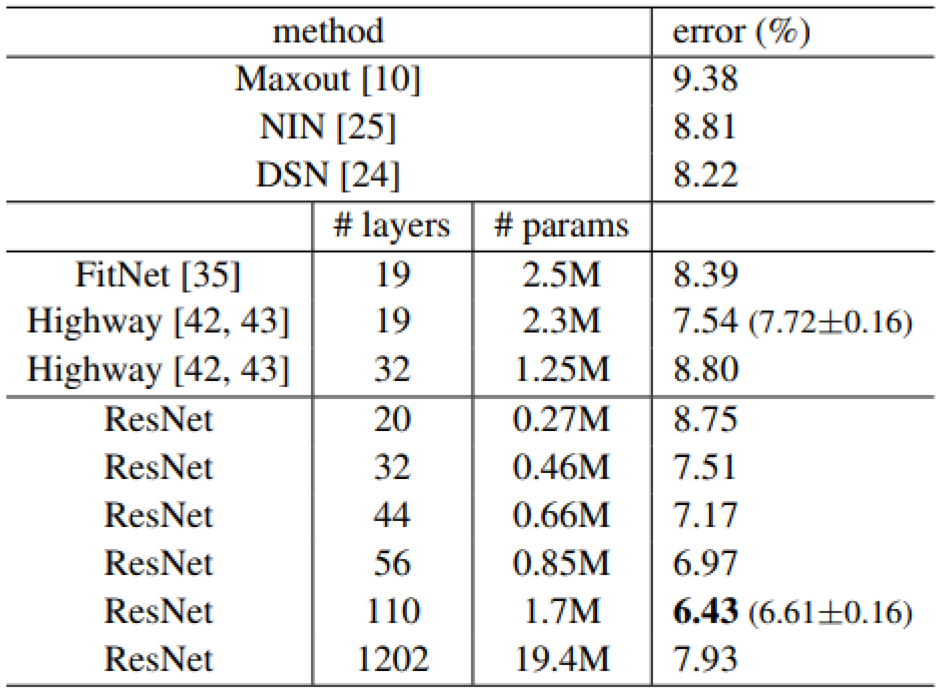
Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Results



101-layer and 152-layer ResNets: We construct 101-layer and 152-layer ResNets by using more 3-layer blocks (Table 1). Remarkably, although the depth is significantly increased, the 152-layer ResNet (11.3 billion FLOPs) still has *lower complexity* than VGG-16/19 nets (15.3/19.6 billion FLOPs).

The 50/101/152-layer ResNets are more accurate than the 34-layer ones by considerable margins (Table 3 and 4). We do not observe the degradation problem and thus enjoy significant accuracy gains from considerably increased depth. The benefits of depth are witnessed for all evaluation metrics (Table 3 and 4).



method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Summary

ResNets have lower training and validation losses.

ResNets don’t overfit as much.

Questions

why was that bad

Provided no useful context for the result

Assumed knowledge the audience didn’t have

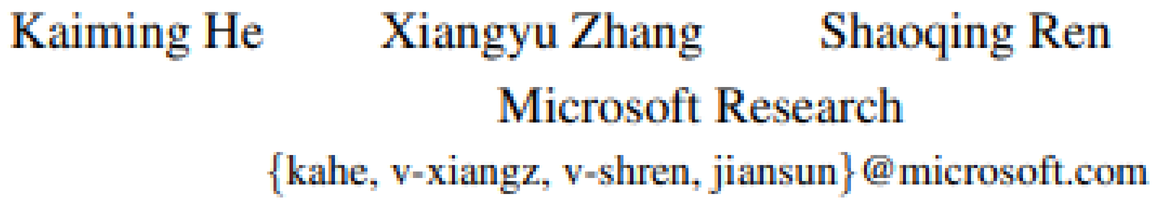
Focused on implementation details and not the core findings

Didn’t provide a critical analysis on results or qualify their impact

Takeaway summary is not actionable advice

Deep Residual Learning for Image Recognition

2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)



Kaiming He   Xiangyu Zhang   Shaoqing Ren   Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Agenda

Today I will present

- (Intro) The problem this paper is trying to solve
- (Methods) The residual connection method
- (Results) The experimental setup in the paper
- (Discussion) Author’s interpretation of the findings
- Impact on the field
- Questions

The Problem

The deeper a neural network is, the harder it is to train.

Technically, a model with more parameters should fit better than a model with fewer, right?

Wrong!

The Problem

"Degradation" of performance occurs in deep networks.

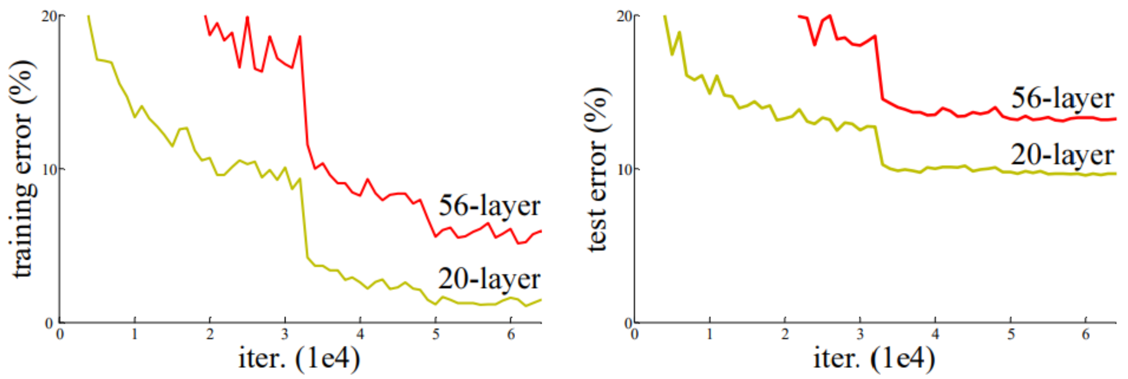


Figure 1. Training error (left) and test error (right) on CIFAR-10

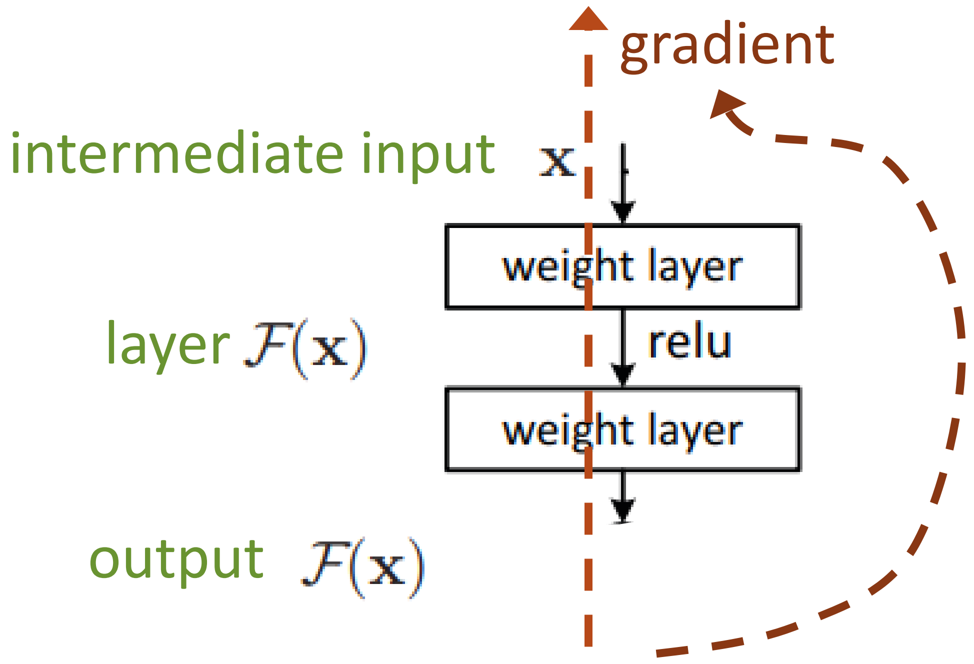
Training a very deep neural network (20+ layers) is always challenging.

Theoretical basis for this challenge is not yet clear. Lots of research has gone into studying this problem, including:

- Batch normalization - https://arxiv.org/pdf/1502.03167v3.pdf
- ReLU activation - https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
- Weights initialization - https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

The Method

Authors propose a mechanism to mitigate degradation, called “residual connections”


gradient
intermediate input $\mathbf{x}$
weight layer
layer $\mathcal{F}(\mathbf{x})$
relu
weight layer
output $\mathcal{F}(\mathbf{x})$

In this way, a multi-layer block trains like a single layer.

Note: a “residual” connection *adds* the input to the output as it pertains to this paper only. A “skip” connection is a more general form of this that skips multiple layers.

Hypotheses

Deeper models will degrade in performance compared to less deep models in identical setups

Addition of residual connections alone will improve benchmark performance on a deep model

Models trained with residual connections will outperform larger state of the art models

Experiments

Use the ImageNet dataset

- 14 million images
- 100,000 “classes” or “synsets”

Benchmark task is to train on a subset and learn to classify images by the objects present in them.

Experiments

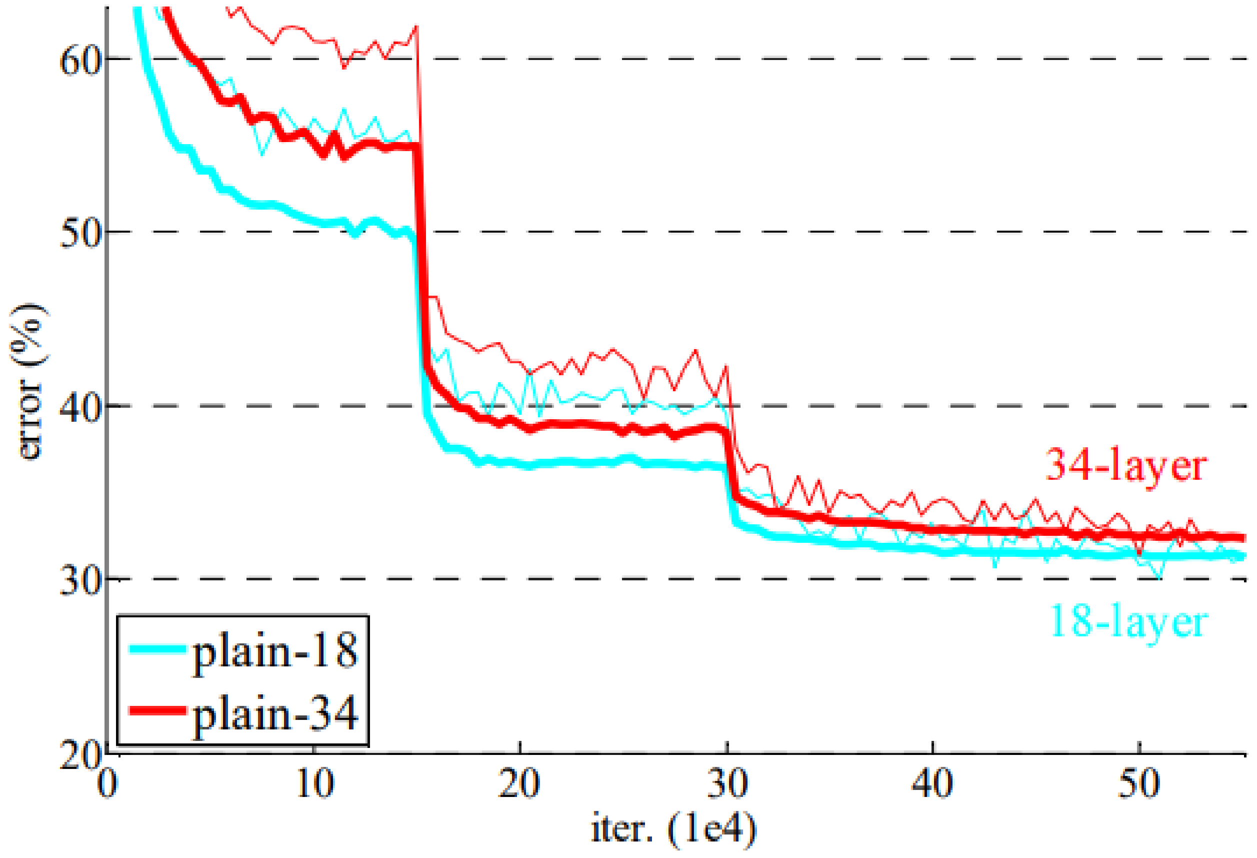
First, identify if degradation does indeed occur with deeper models.

Train two deep convolutional neural net models, an 18-layer and 34-layer.

Does the 34 layer model perform worse?

Thin = training Top-1 error

Bold = validation Top-1 error

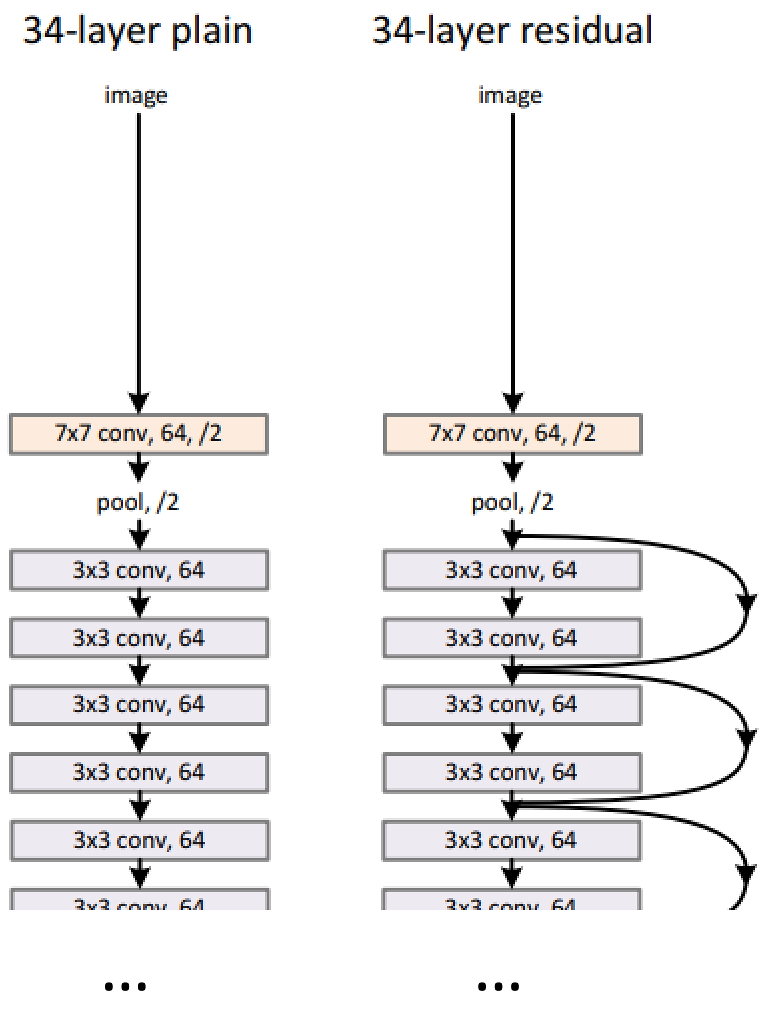


Experiments

The authors initially set up 2 models for comparison.

1. plain CNN (either 18 or 34 layers deep)

2. residual CNN (18 or 34, identical with addition of residual connections)

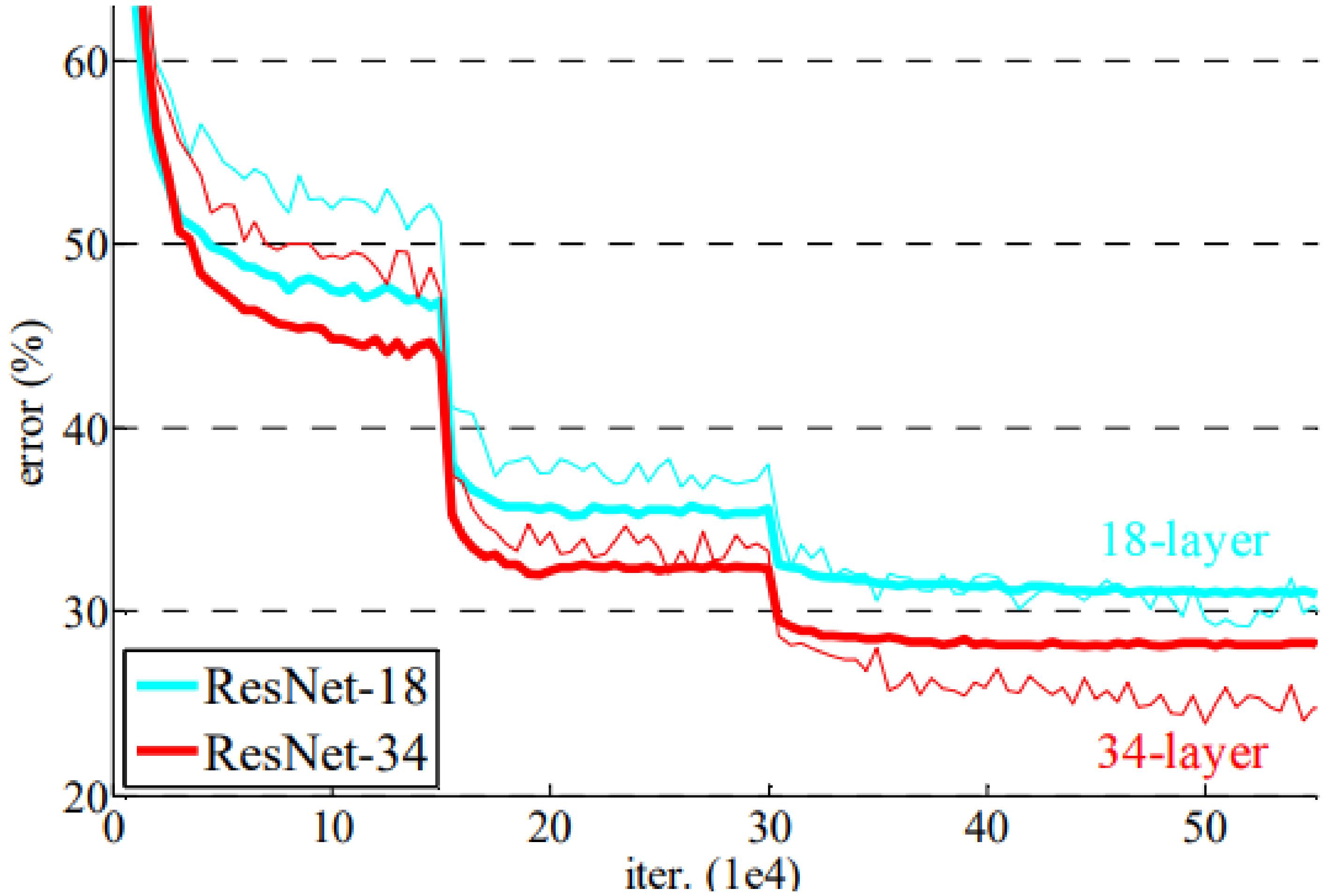


Experiments

Thin = training Top-1 error

Bold = validation Top-1 error

Adding residual connections to deep models improve performance



Top-1 error %

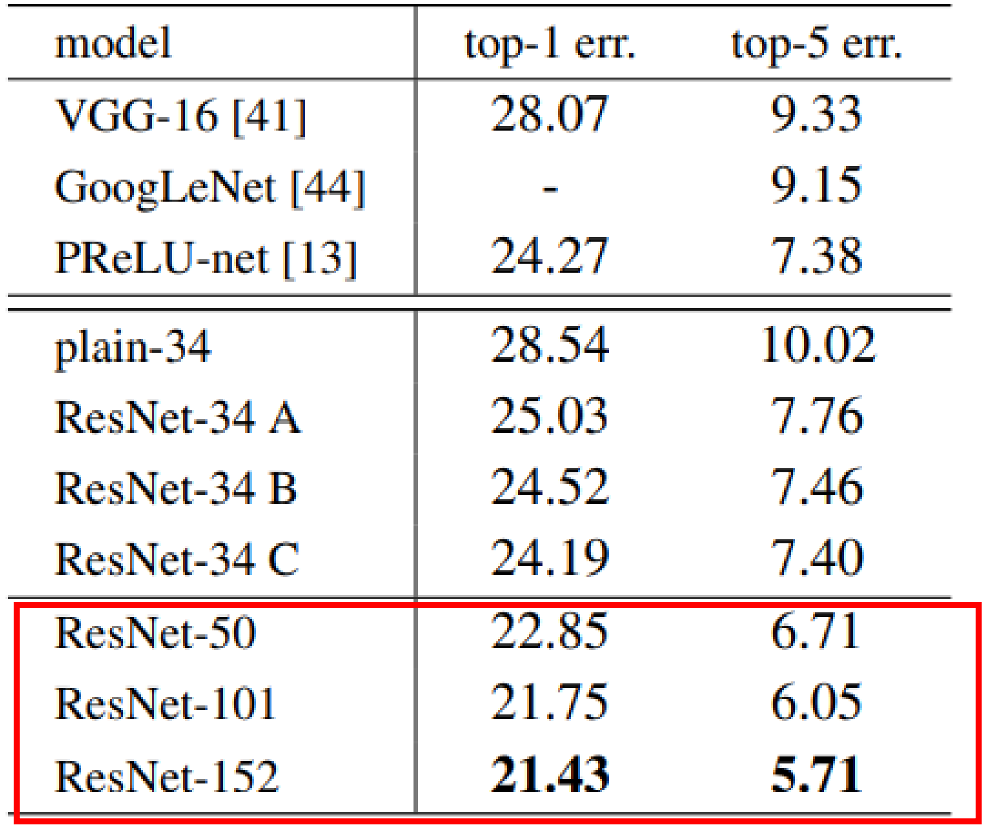
	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Experiments

How much better can performance get? Is it better than the state of the art model, VGG-16?

https://arxiv.org/pdf/1409.1556.pdf

How deep can you go?



model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

Interpretation

- Authors assert that degradation impacts deep models, and that residual connections mitigate it. Their evidence is compelling.
- “Degradation” remains a poorly defined concept. Perhaps better characterization of it would help formulate a more principled solution.
- Decisions on *how* to make residual connections was arbitrary.

Impact

Residual connections are an effective trick, but they lack a theoretical underpinning so far.

Top performance on ImageNet benchmark is impressive – but overfitting to a single benchmark dataset can be misleading.

Points us towards better questions to ask like:

- What *exactly* causes degradation?
- Why do residual connections improve deep models ability to learn?
- (engineering) how to choose the kind of residual/skip connection based on the model, data, or problem domain?

Questions

why was that better (hopefully)

Defined the problem that needed solving

Minimal use of jargon, citations where necessary

Did not dwell on the math – presentation time better spent elsewhere

Broke paper down into set of hypotheses and experiments to test them

Provided added value to the paper by lending my own analysis

Thank you!

:)