

# Introduction: History + Overview

Ling 282/482: Deep Learning for Computational Linguistics

C.M. Downey

Fall 2024

# Today's Plan

- Brief general introduction
- Potted history of Deep Learning
- Potted history of models in NLP
- Course information / logistics

# What is deep learning for language?

# What is deep learning for language?

- Language is an amazingly **flexible** system for communicating complex information
  - Novel expressions
  - Arbitrarily complex
  - Systematic **generalization**

# What is deep learning for language?

- Language is an amazingly **flexible** system for communicating complex information
  - Novel expressions
  - Arbitrarily complex
  - Systematic **generalization**
- Prime example of a **symbolic** system

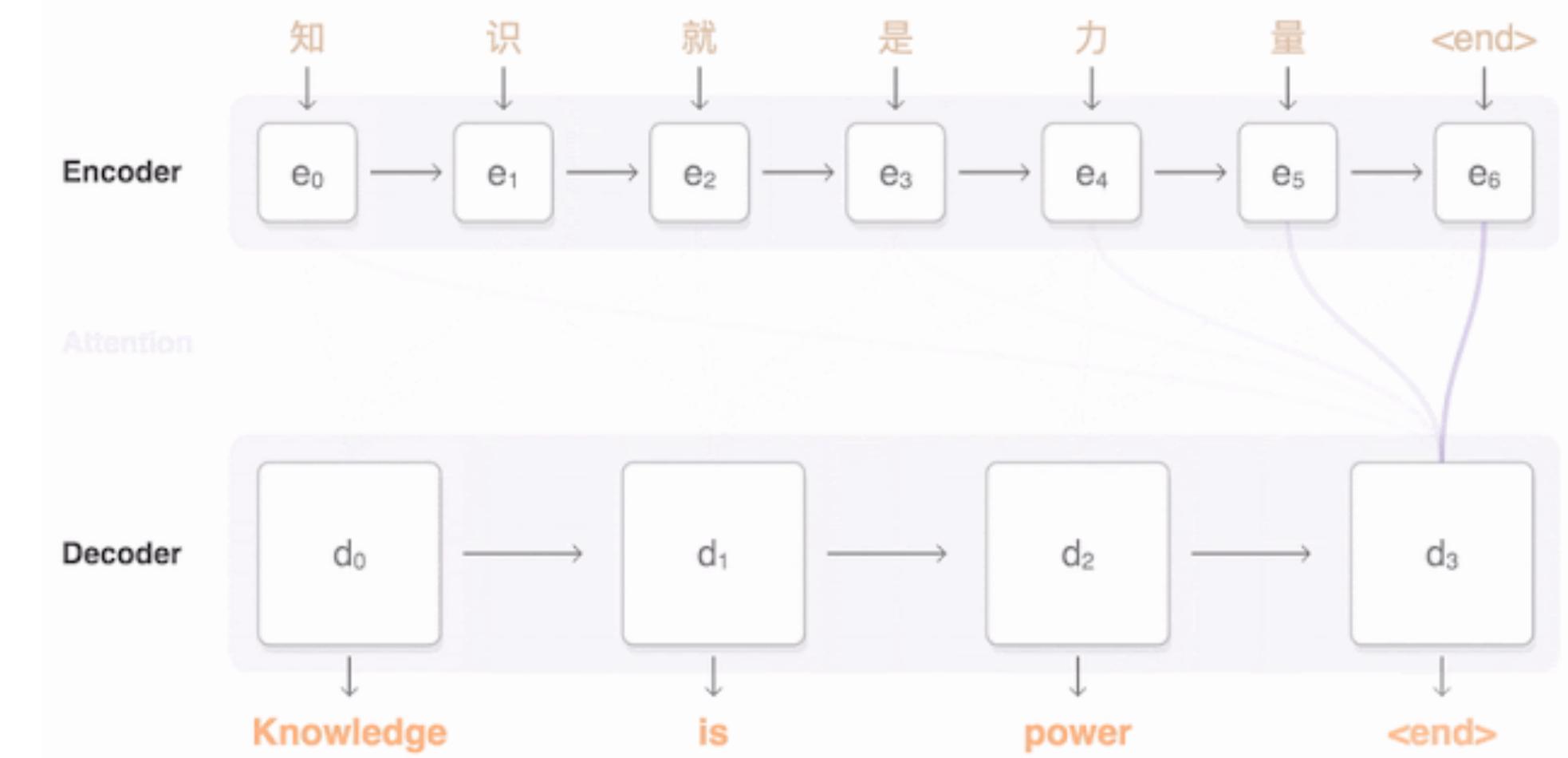
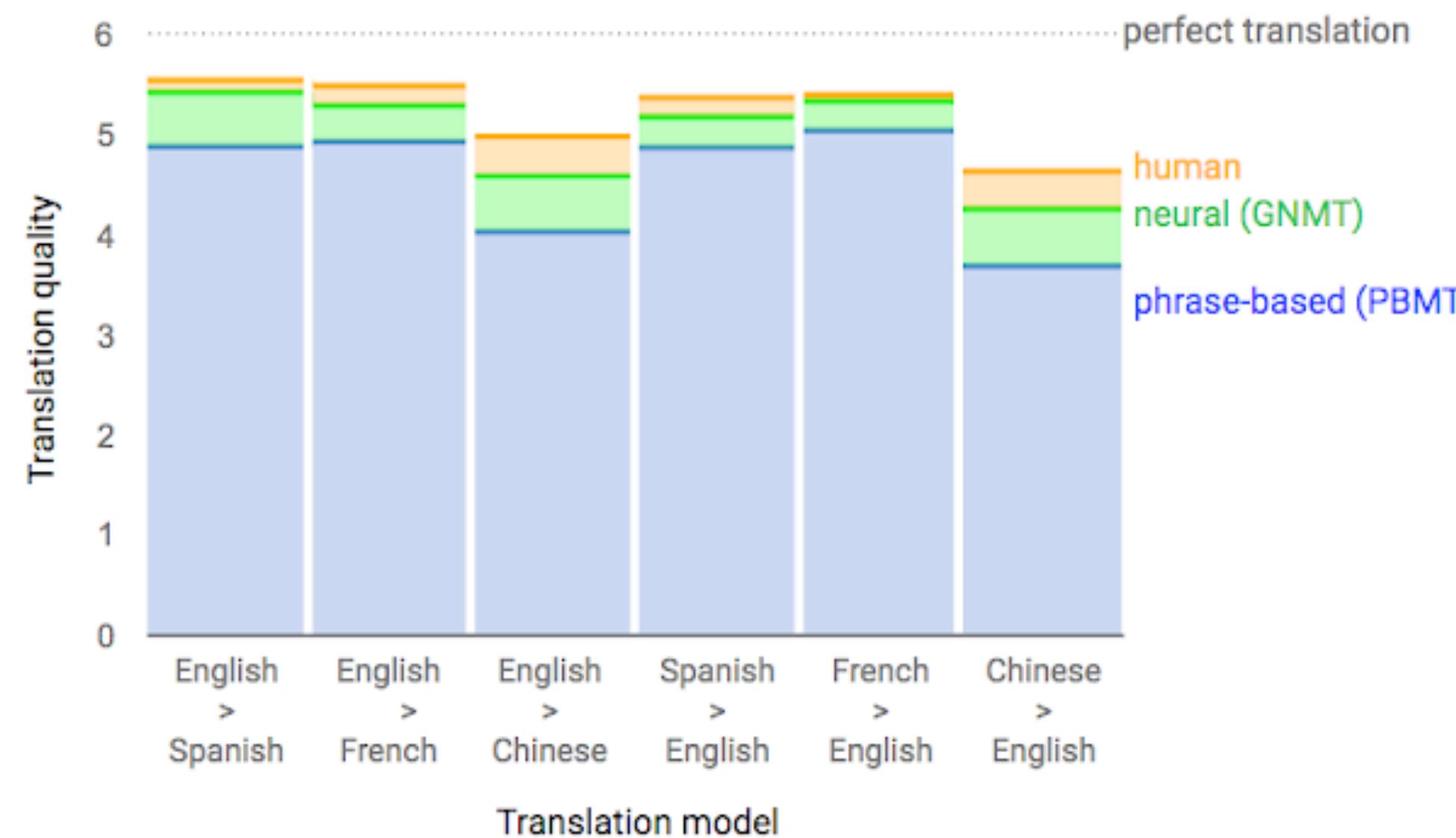
# What is deep learning for language?

- Language is an amazingly **flexible** system for communicating complex information
  - Novel expressions
  - Arbitrarily complex
  - Systematic **generalization**
- Prime example of a **symbolic** system
- How do we enable computers to understand and process language?
  - Traditional approach: by **manipulating symbols**

# What is deep learning for language?

- Application of neural networks specifically to language data and tasks
- Discrete symbols are replaced by *continuous vectors*
  - Large models build “deep” (hopefully **hierarchical**) representations of text
- But can they successfully mimic human language understanding?

# “Early” Success: Neural Machine Translation



when was the university of washington founded

X |

All News Maps Images Shopping More Settings Tools

About 219,000,000 results (1.10 seconds)

University of Washington / Founded

November 4, 1861



(The University building is on the hill in the center of the photograph.) On **November 4, 1861**, the Territorial University of Washington began classes in a building located on a site now occupied by the Olympic Hotel. The University was on the outskirts of the village of Seattle, which had a population of 250.

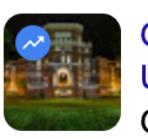
<https://www.lib.washington.edu> > exhibits > site > early

## I. The University of Washington's Early Years — UW Libraries

People also search for



Gonzaga University  
September 17, 1887



Oregon State University  
October 27, 1868, Oregon



Washington State University  
March 28, 1890

Feedback

People also ask

Who founded University of Washington?

## University of Washington

Public university in Seattle, Washington

The University of Washington is a public research university in Seattle, Washington. Founded in 1861, Washington is one of the oldest universities on the West Coast; it was established in downtown Seattle approximately a decade after the city's founding to aid its economic development. [Wikipedia](#)

Avg cost after aid	Graduation rate	Acceptance rate
\$12K	84%	52%

Graduation rate is for first-time, full-time undergraduate [more](#)  
Source: US Dept of Education · Learn more

**Undergraduate enrollment:** 30,905 (2018–19)

**Average annual in-state cost**

Before aid	After aid
\$28,394	\$12,001

Aid includes grants and scholarships from the institution, state, and federal government

Source: US Dept of Education (IPEDS) · Learn more

Dinner next week

Evan Brown, Maalika Patel

Dinner next week

Hey Evan and Maalika,

Haven't seen you in a while! Wanna grab dinner next week at my place? My address is 34 Smith Street, Somers, CT 06071.

Text Documents

DETECT LANGUAGE GERMAN ENGLISH SPANISH HEBREW GERMAN ENGLISH

The University of Washington was founded in 1861 in Seattle. × ☆ אוניברסיטת וושינגטון נוסדה בשנת 1861 בסיאטל.

60 / 5000

Send feedback

when was the university of washington founded

X |

All News Maps Images Shopping More

Settings Tools

About 219,000,000 results (1.10 seconds)

University of Washington / Founded

November 4, 1861

(The University building is on the hill in the center of the photograph.) On November 4, 1861, the Washington Territorial University began classes in a building located on the grounds of the Olympic Hotel. The University was on the outskirts of the village of Seattle, which had a population of 250.

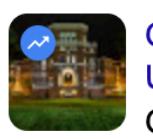
<https://www.lib.washington.edu> > exhibits > site > early

## I. The University of Washington's Early Years — University of Washington

### People also search for



Gonzaga University  
September 17, 1887



Oregon State University  
October 27, 1868, Oregon



Washington State University  
March 1890

### People also ask :

Who founded University of Washington?

Text

Doc

DETECT LANGUAGE

The University of Washington was founded in 1861 in Seattle.



60 / 5000



University of Washington

Public university in Seattle, Washington



ChatGPT  
UI Mockup

Y Explain quantum computing in simple terms

Quantum computing is a type of computing where information is processed using quantum-mechanical phenomena, such as superposition and entanglement. In traditional computing, information is processed using bits, which can have a value of either 0 or 1. In quantum computing, information is processed using quantum bits, or qubits, which can exist in multiple states simultaneously. This allows quantum computers to perform certain types of calculations much faster than traditional computers.

Y Explain quantum computing in simple terms

Quantum computing is a type of computing where information is processed using quantum-mechanical phenomena, such as superposition and entanglement. In traditional computing, information is processed using bits, which can have a value of either 0 or 1. In quantum computing, information is processed using quantum bits, or qubits, which can exist in multiple states simultaneously. This allows quantum computers to perform certain types of calculations much faster than traditional computers.

Send feedback

VERSITY of ROCHESTER

Send feedback VERSITY of ROCHESTER

# What This Course Is and Is Not

# What This Course Is and Is Not

- Provide a firm theoretical understanding of how to apply deep learning methods to natural language tasks

# What This Course Is and Is Not

- Provide a firm theoretical understanding of how to apply deep learning methods to natural language tasks
- From the ground up, progressing in complexity

# What This Course Is and Is Not

- Provide a firm theoretical understanding of how to apply deep learning methods to natural language tasks
- From the ground up, progressing in complexity
- We will apply different kinds of models to interesting linguistic tasks, but this course is **not** simply:
  - How to use the latest libraries
  - End-to-end application development

# What This Course Is and Is Not

- Provide a firm theoretical understanding of how to apply deep learning methods to natural language tasks
- From the ground up, progressing in complexity
- We will apply different kinds of models to interesting linguistic tasks, but this course is **not** simply:
  - How to use the latest libraries
  - End-to-end application development
- By understanding the theory behind and building blocks of progressively complex systems, you will be able to:
  - Process new developments, diagnose / debug perplexing errors, understand why things work the way they do (in the good and the bad case)

# A Potted History of NNs

# The first artificial neural network: 1943

BULLETIN OF  
MATHEMATICAL BIOPHYSICS  
VOLUME 5, 1943

A LOGICAL CALCULUS OF THE  
IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. McCULLOCH AND WALTER PITTS

FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE,  
DEPARTMENT OF PSYCHIATRY AT THE ILLINOIS NEUROPSYCHIATRIC INSTITUTE,  
AND THE UNIVERSITY OF CHICAGO



UNIVERSITY OF ROCHESTER

# Turing Award: 2018

The screenshot shows the official website for the ACM A.M. Turing Award. At the top, the ACM logo and the text "MORE ACM AWARDS" are visible. To the right, a large image of a man's face is displayed with the text "A.M. TURING AWARD". Below this, a grid of smaller portrait photos of previous Turing Award winners is shown. A navigation bar at the bottom includes links for "ALPHABETICAL LISTING", "YEAR OF THE AWARD", and "RESEARCH SUBJECT".

**GEOFFREY HINTON AND YANN LECUN TO DELIVER TURING LECTURE AT FCRC 2019**  
June 23, 5:15 - 6:30 P.M., Symphony Hall

We are pleased to announce that Geoffrey Hinton and Yann LeCun will deliver the Turing Lecture at FCRC 2019. Hinton's talk, "The Deep Learning Revolution," and LeCun's talk, "The Deep Learning Revolution: The Sequel," will be presented June 23rd from 5:15-6:30pm in Symphony Hall, Phoenix, Arizona.

No registration or tickets necessary to attend.

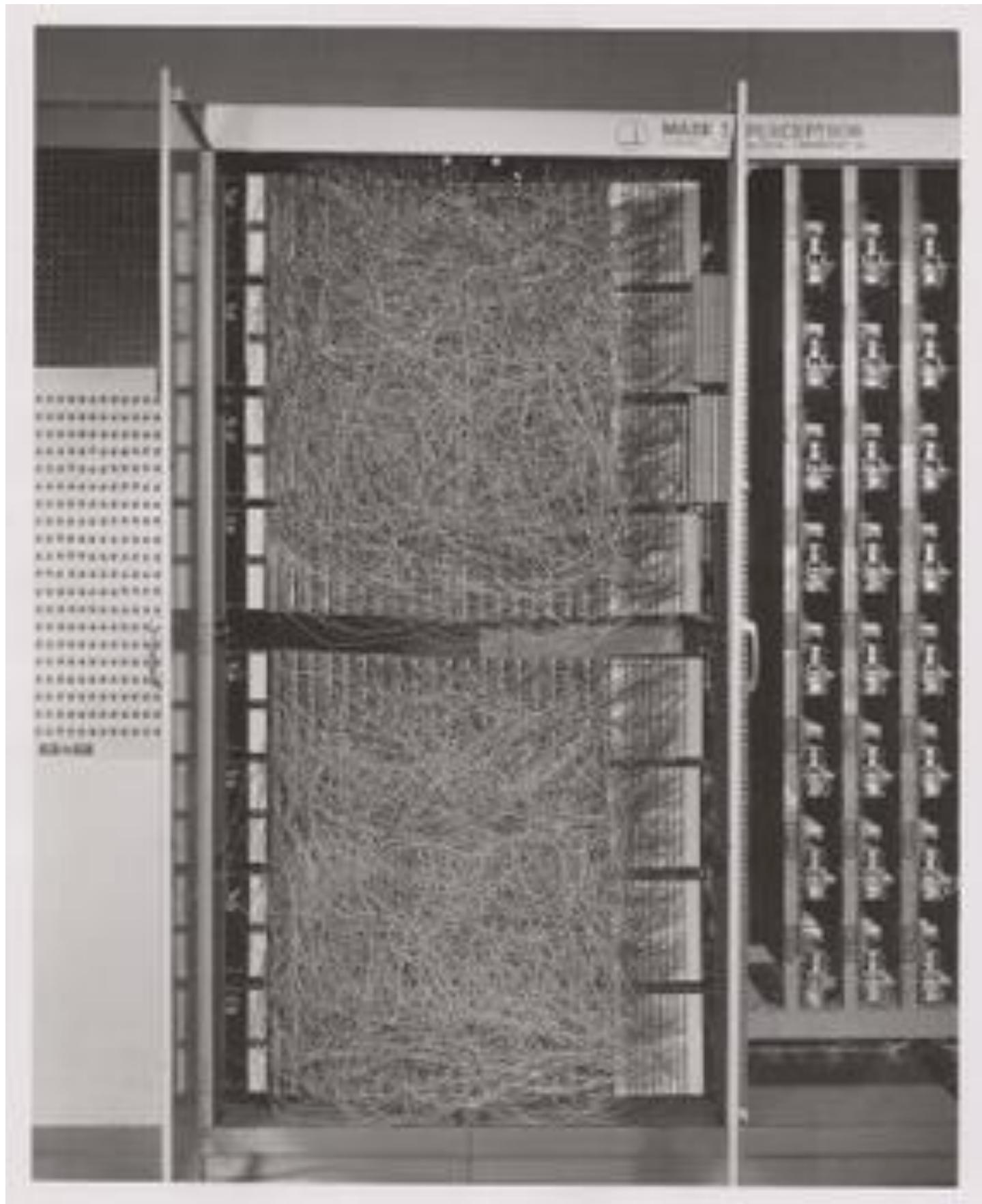
*[View the Livestream](#)*

**FATHERS OF THE DEEP LEARNING REVOLUTION RECEIVE ACM A.M. TURING AWARD**  
Bengio, Hinton, and LeCun Ushered in Major Breakthroughs in Artificial Intelligence

ACM named [Yoshua Bengio](#), [Geoffrey Hinton](#), and [Yann LeCun](#) recipients of the 2018 ACM A.M. Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing. Bengio is Professor at the University of Montreal and Scientific Director at Mila, Quebec's Artificial Intelligence Institute; Hinton is VP and Engineering Fellow of Google, Chief Scientific Adviser of The Vector Institute, and University Professor Emeritus at the University of Toronto; and LeCun is Professor at New York University and VP and Chief AI Scientist at Facebook.

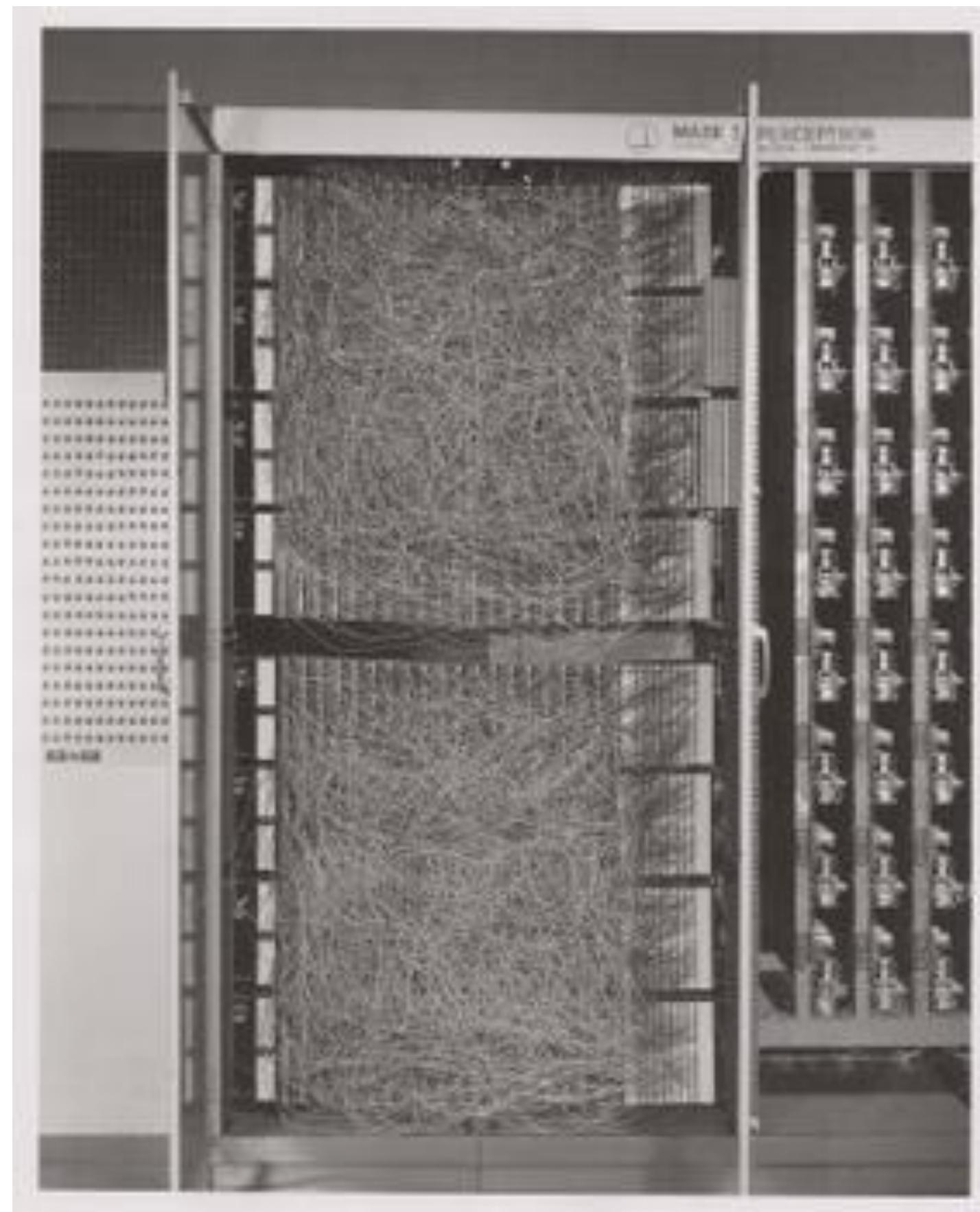
Working independently and together, Hinton, LeCun and Bengio developed conceptual foundations for the field, identified surprising phenomena through experiments, and contributed engineering advances that demonstrated the practical advantages of deep neural networks. In recent years, deep learning methods have been

# Perceptron (1958)



[source](#)

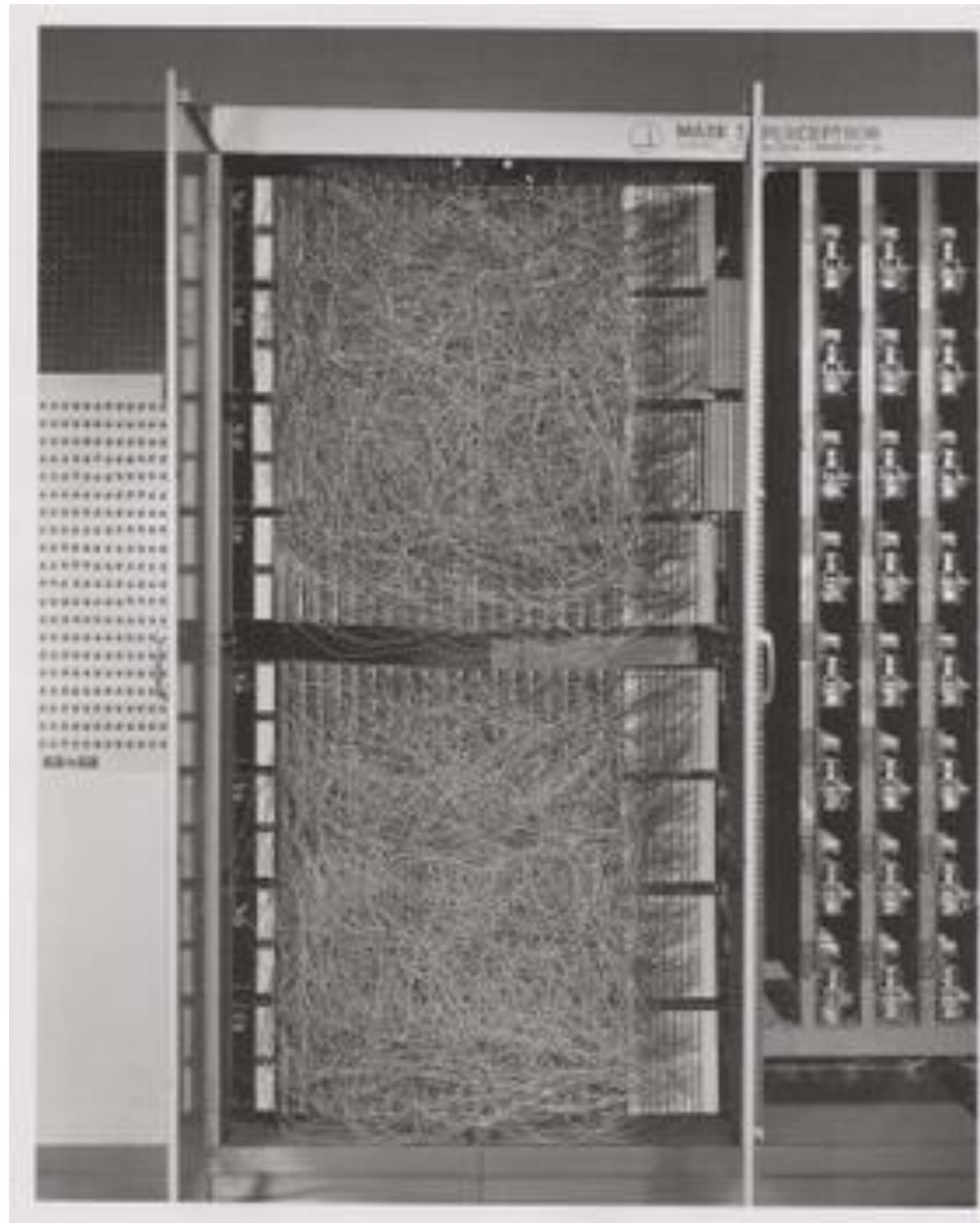
# Perceptron (1958)



[source](#)

$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

# Perceptron (1958)



$$f(\mathbf{x}) = \begin{cases} 1 & \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

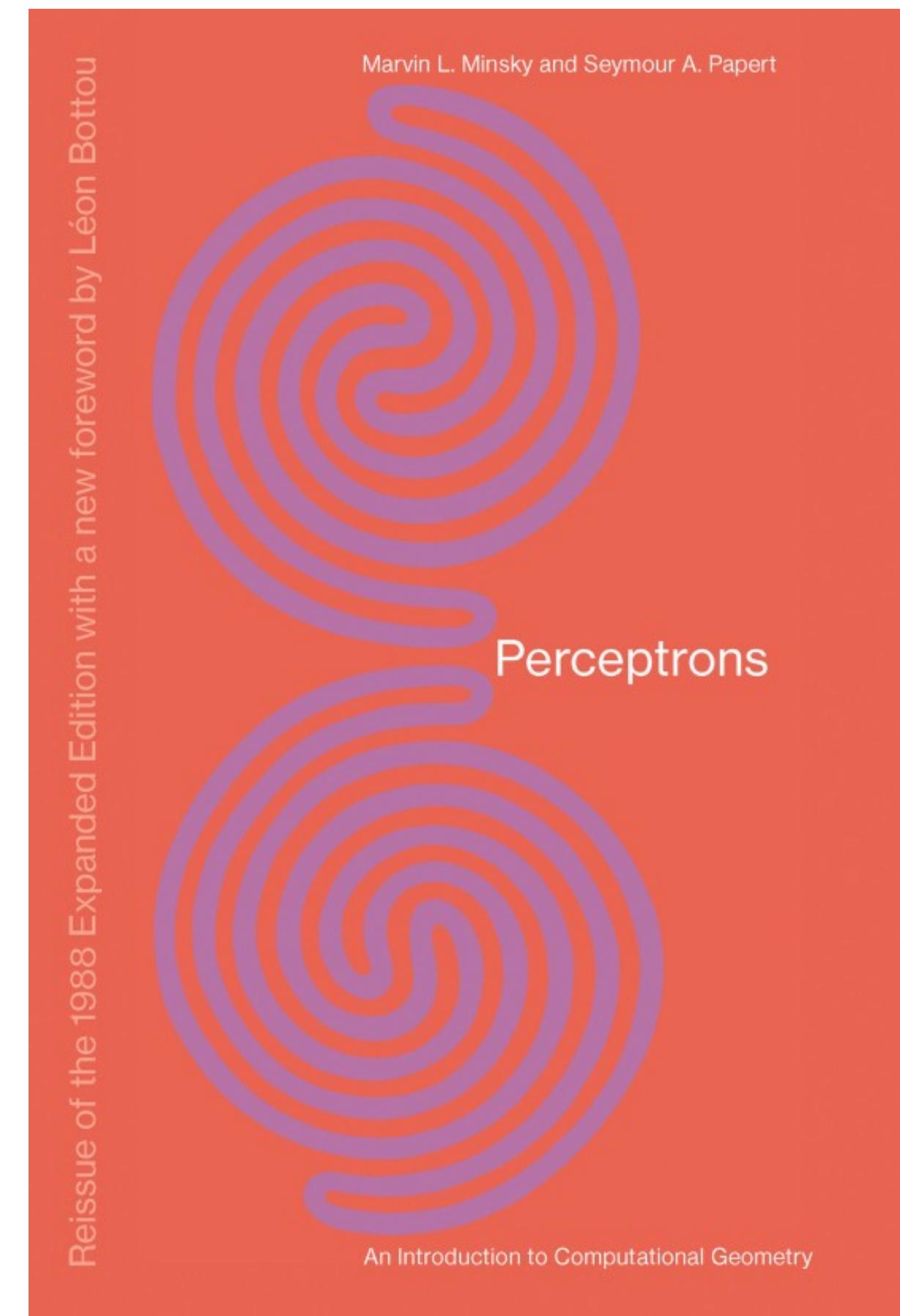
“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

—New York Times

[source](#)

# Perceptrons (1969)

- Limitative results on functions computable by the basic perceptron
- Famous example (we'll return to it later):
  - Exclusive disjunction (XOR) is not computable
- Other examples that are uncomputable assuming *local connectivity*



# AI Winter

# AI Winter

- Reaction to the results:
  - The approach of learning perceptrons for data cannot deliver on the promises
  - Funding from e.g. government agencies dried up significantly
  - Community lost interest in the approach

# AI Winter

- Reaction to the results:
  - The approach of learning perceptrons for data cannot deliver on the promises
  - Funding from e.g. government agencies dried up significantly
  - Community lost interest in the approach
- Very unfortunate:
  - Already known from McCulloch and Pitts that any boolean function can be computed by “deeper” networks of perceptrons
  - Negative consequences of the results were significantly over-blown

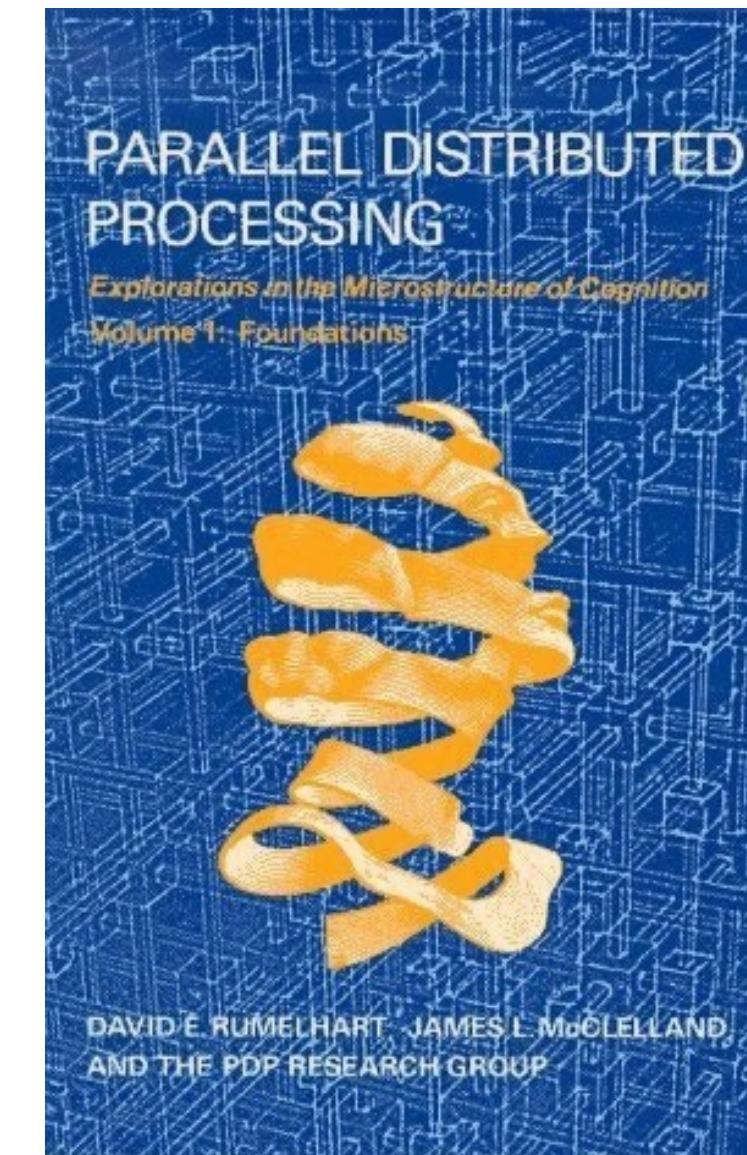
# Deeper Backpropagation (1986)

# Deeper Backpropagation (1986)

- Multi-layer networks, trained by backpropagation, applied to cognitive tasks

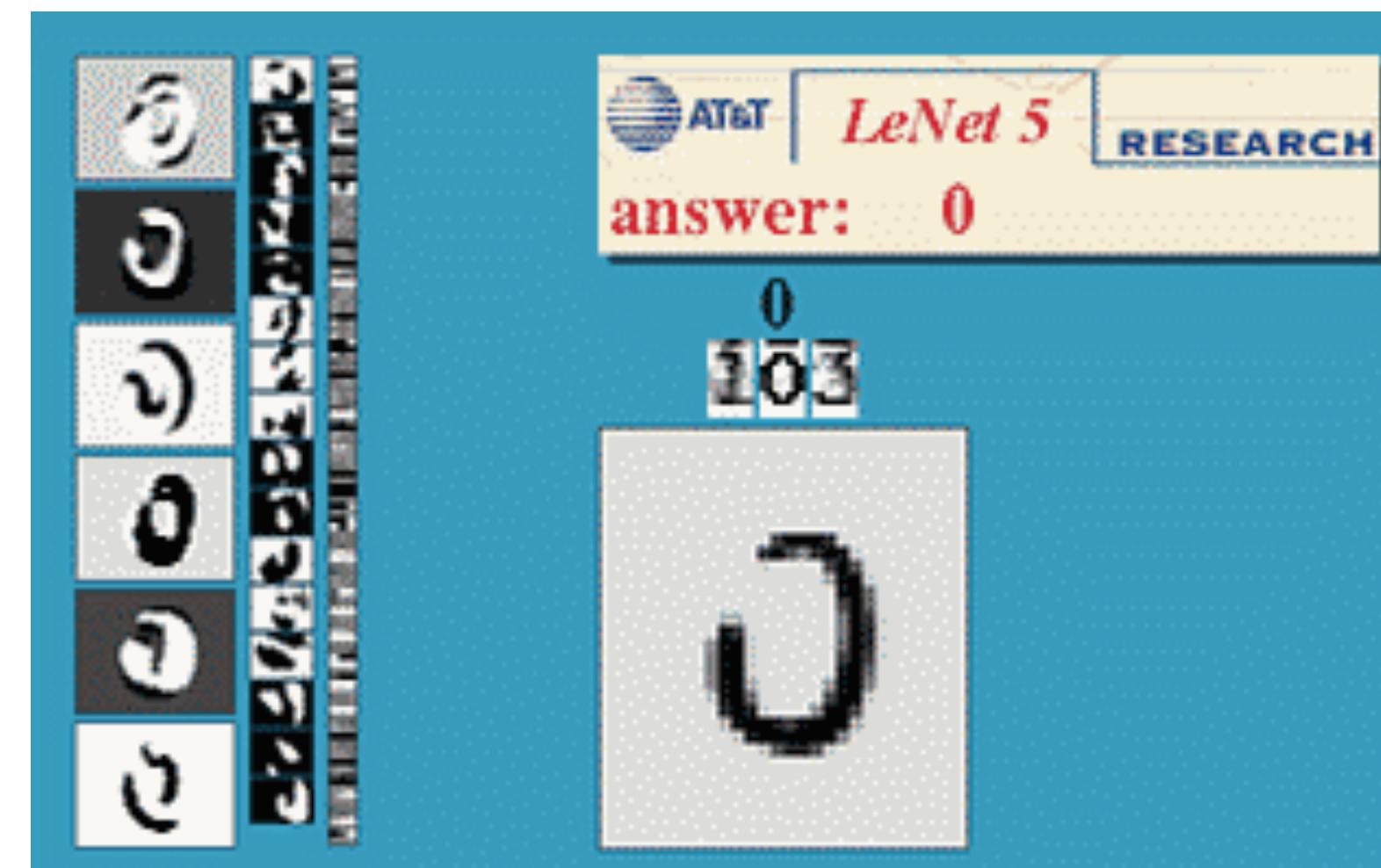
# Deeper Backpropagation (1986)

- Multi-layer networks, trained by backpropagation, applied to cognitive tasks
- “The book *Parallel Distributed Processing* presented the results of some of the first successful experiments with back-propagation in a chapter (Rumelhart et al., 1986b) that contributed greatly to the popularization of back-propagation and initiated a very active period of research in multilayer neural networks.”



# Successful Engineering Application (1989)

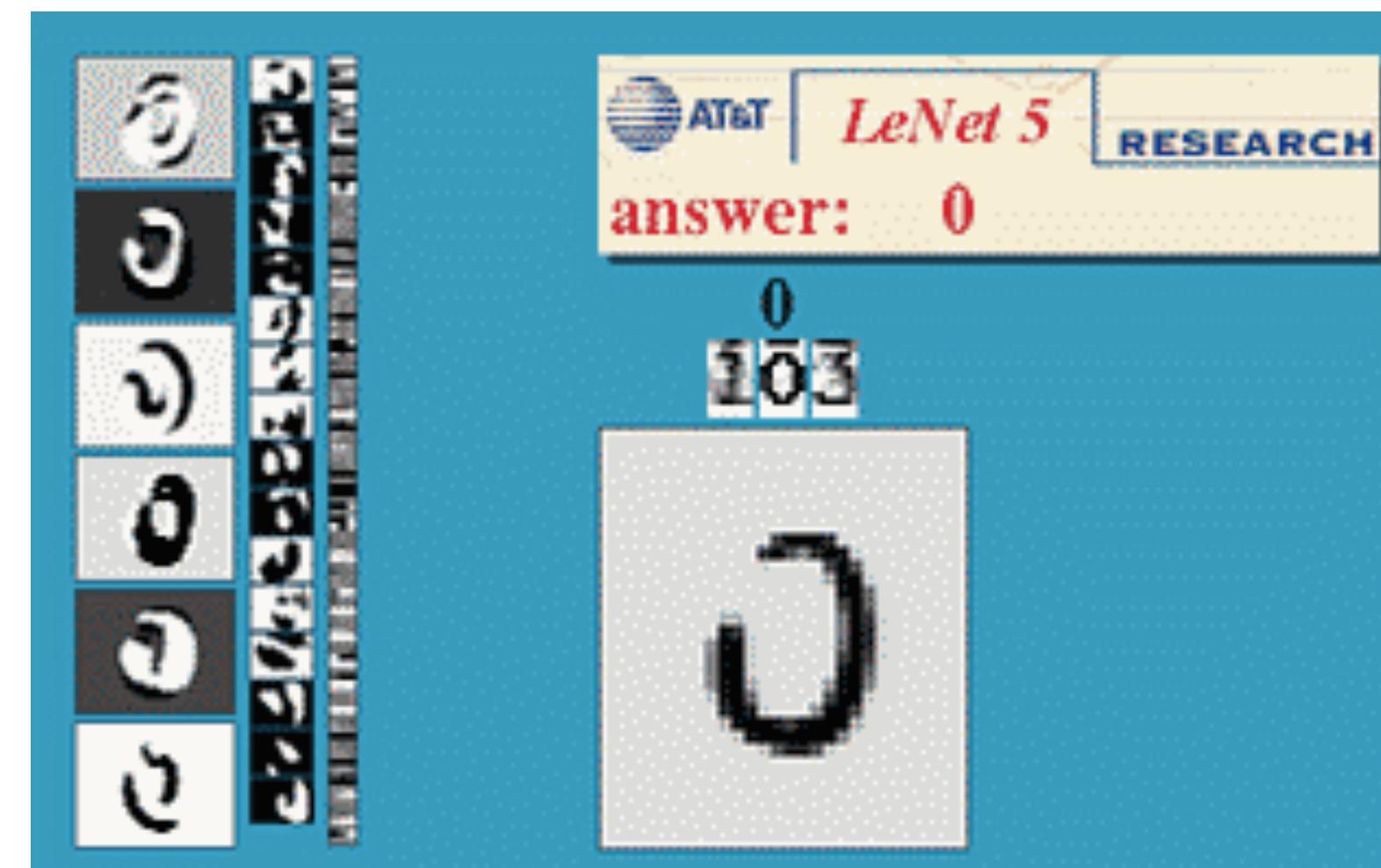
- *Convolutional* networks (“LeNet”, after Yann LeCun) applied to recognizing hand-written digits
  - MNIST dataset
  - Still useful for setting up pipelines, testing simple baselines, etc.
  - Deployed for automatic reading of mailing addresses, check amounts, etc.



original website

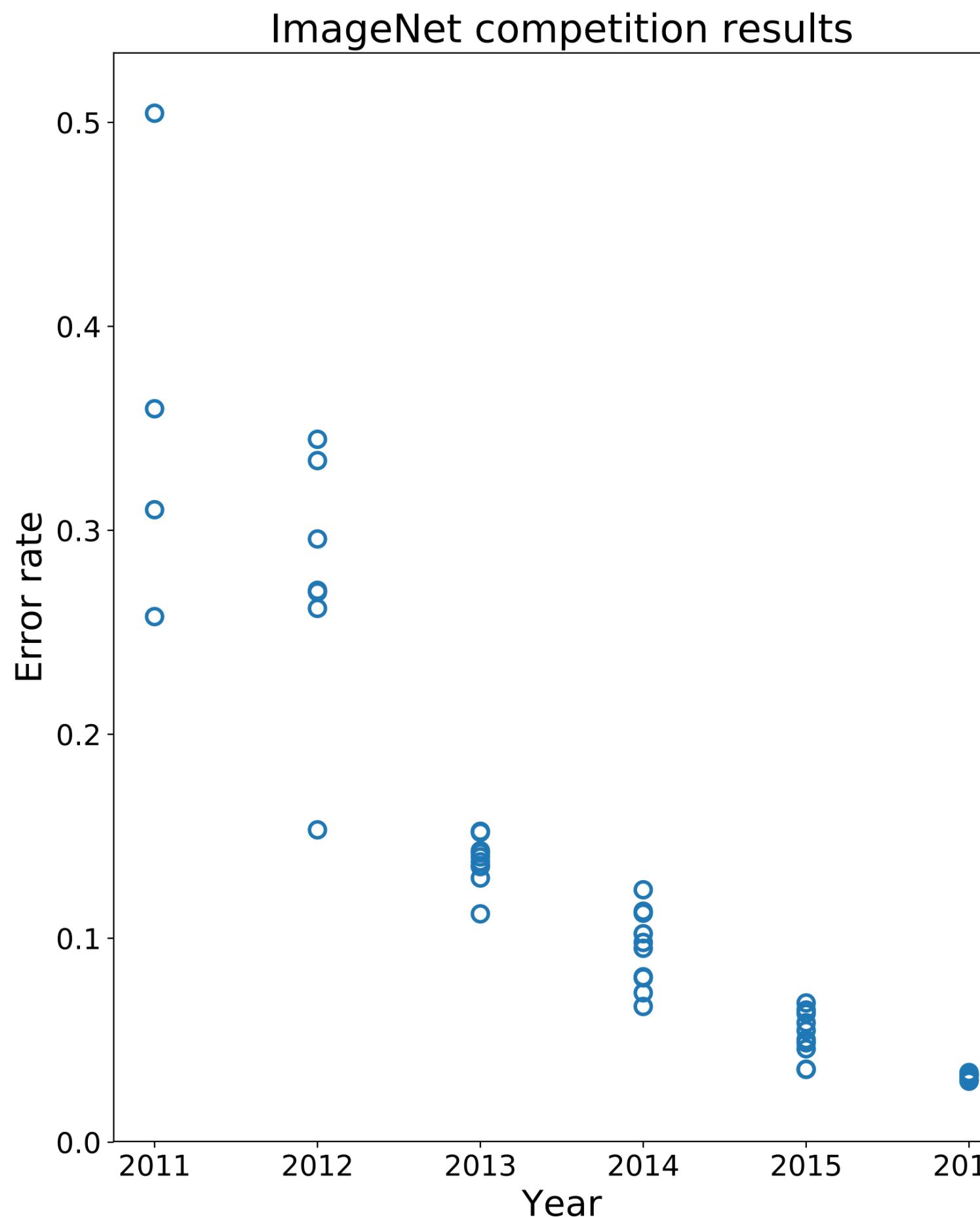
# Successful Engineering Application (1989)

- *Convolutional* networks (“LeNet”, after Yann LeCun) applied to recognizing hand-written digits
  - MNIST dataset
  - Still useful for setting up pipelines, testing simple baselines, etc.
  - Deployed for automatic reading of mailing addresses, check amounts, etc.



original website

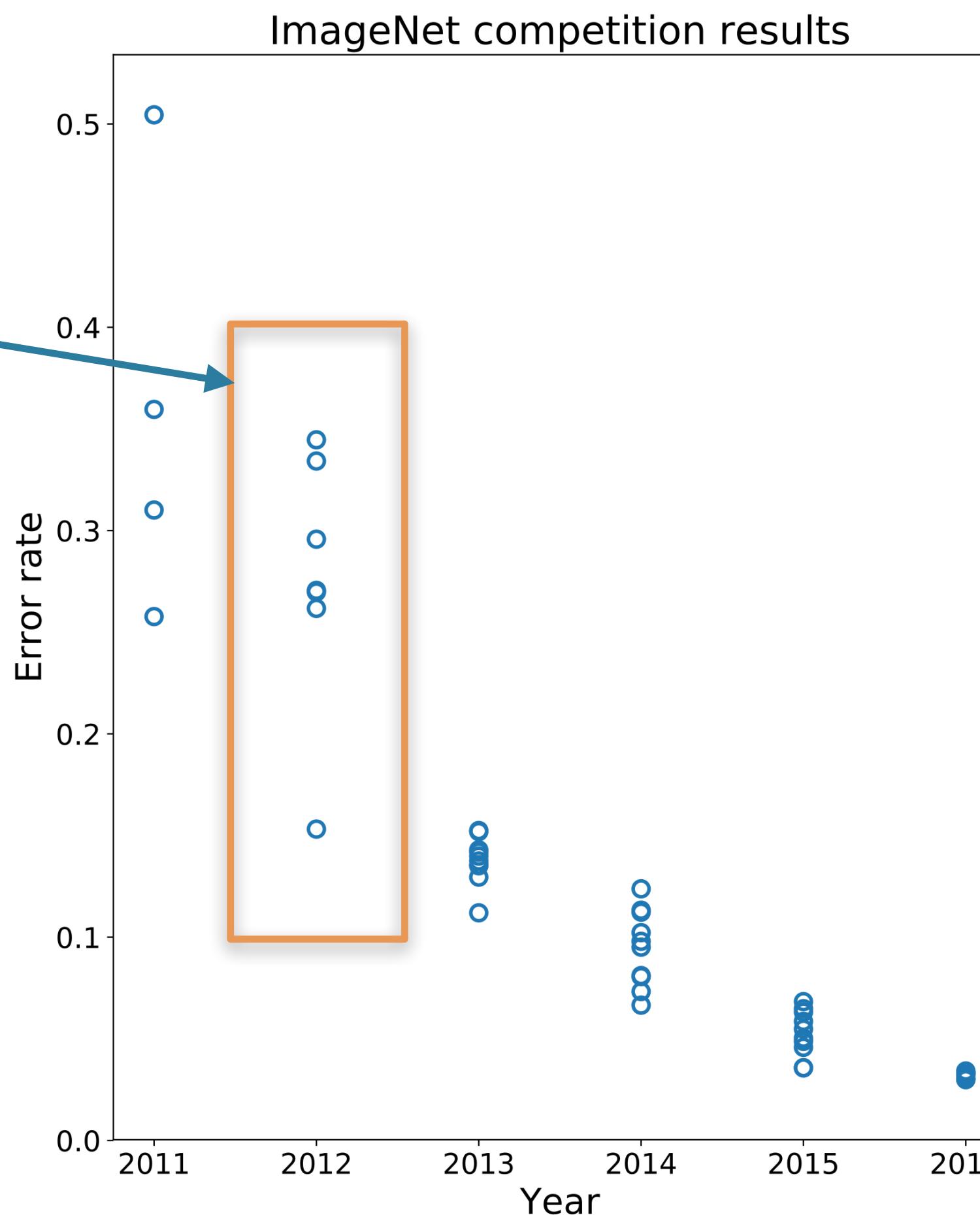
# ImageNet (ILSVRC) results (2012)



[source](#)

# ImageNet (ILSVRC) results (2012)

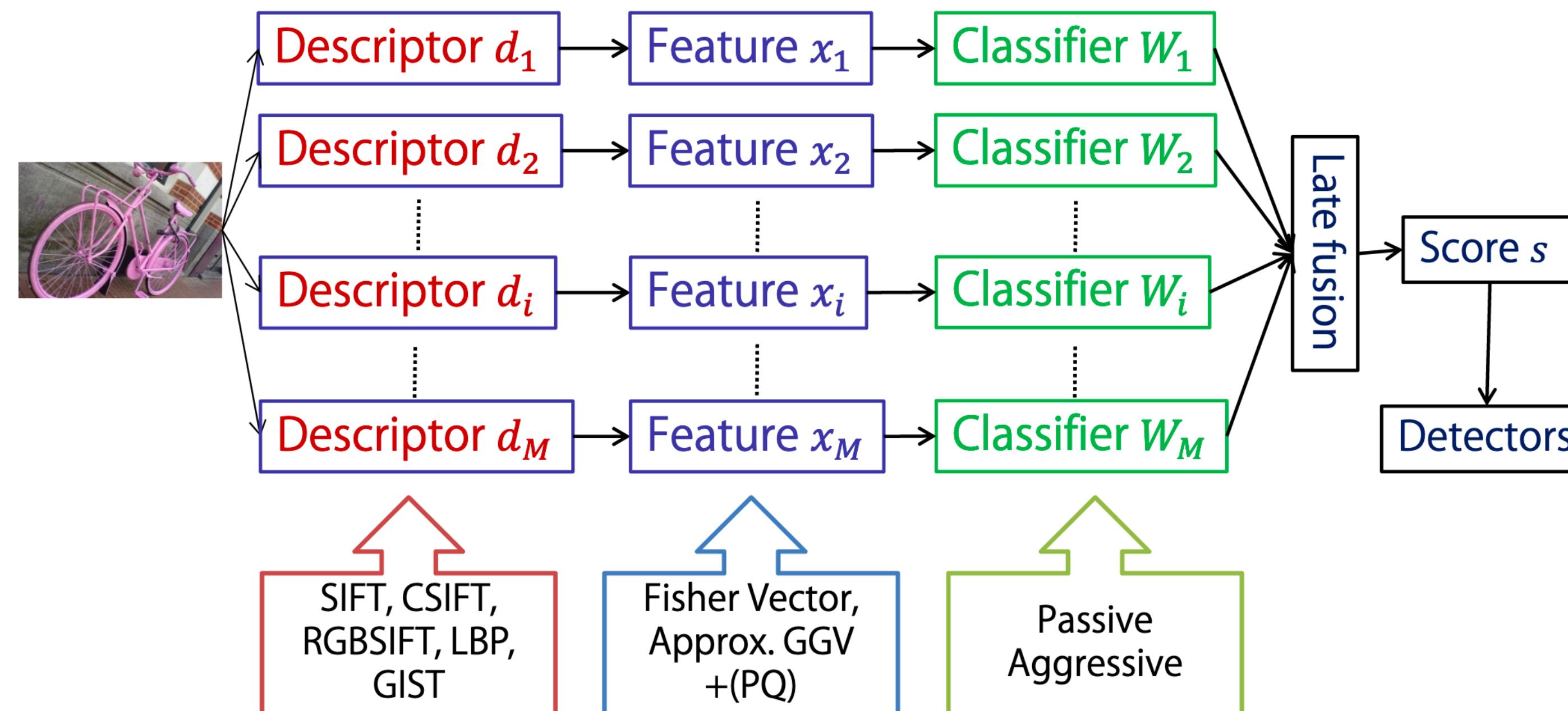
What happened in 2012?



[source](#)

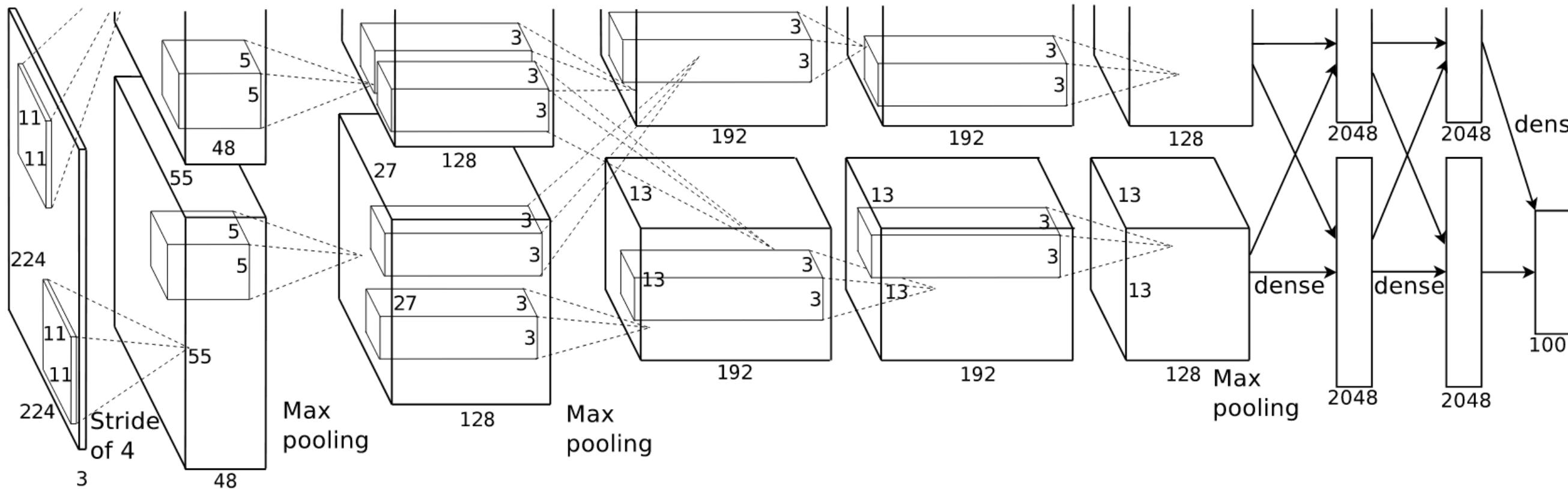
# ILSVRC 2012: runner-up

Fisher based features + Multi class linear classifiers



[source](#)

# ILSVRC 2012: winner



---

## ImageNet Classification with Deep Convolutional Neural Networks

---

[NeurIPS 2012 paper](#)

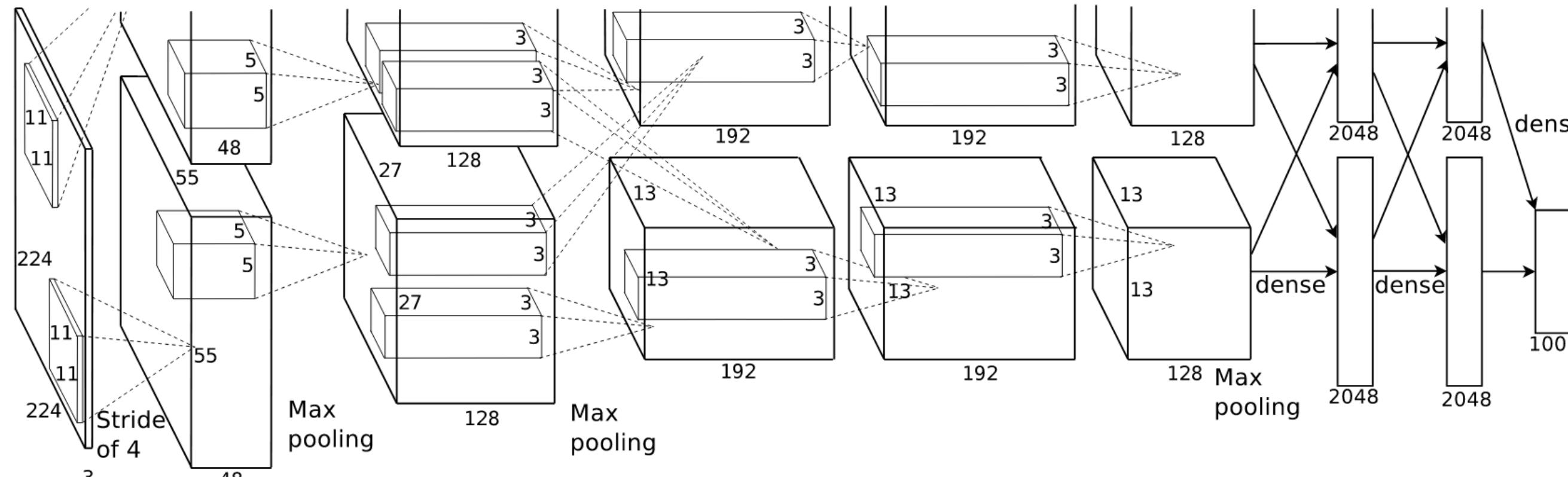
**Alex Krizhevsky**  
University of Toronto  
kriz@cs.utoronto.ca

**Ilya Sutskever**  
University of Toronto  
ilya@cs.utoronto.ca

**Geoffrey E. Hinton**  
University of Toronto  
hinton@cs.utoronto.ca

# ILSVRC 2012: winner

“AlexNet”



---

## ImageNet Classification with Deep Convolutional Neural Networks

---

[NeurIPS 2012 paper](#)

Alex Krizhevsky  
University of Toronto  
kriz@cs.utoronto.ca

Ilya Sutskever  
University of Toronto  
ilya@cs.utoronto.ca

Geoffrey E. Hinton  
University of Toronto  
hinton@cs.utoronto.ca

# 2012-now



# 2012-now

- Widespread adoption of deep neural networks across a range of domains / tasks
  - Image processing of various kinds
  - Reinforcement learning (e.g. AlphaGo/AlphaZero, ...)
  - NLP!

# 2012-now

- Widespread adoption of deep neural networks across a range of domains / tasks
  - Image processing of various kinds
  - Reinforcement learning (e.g. AlphaGo/AlphaZero, ...)
  - NLP!
- What happened?
  - Better learning algorithms / training regimes
  - Larger and larger, standardized datasets
  - Compute! GPUs, now dedicated hardware (TPUs)
  - Videogames?

# Videogames and Neural Nets

- As it turns out, both 3D graphics and neural networks involve lots of **matrix multiplications**
- The demand for better gaming graphics drove better **Graphics Processing Units (GPUs)**
- The Deep Learning “Revolution” was partially driven by this progress in hardware



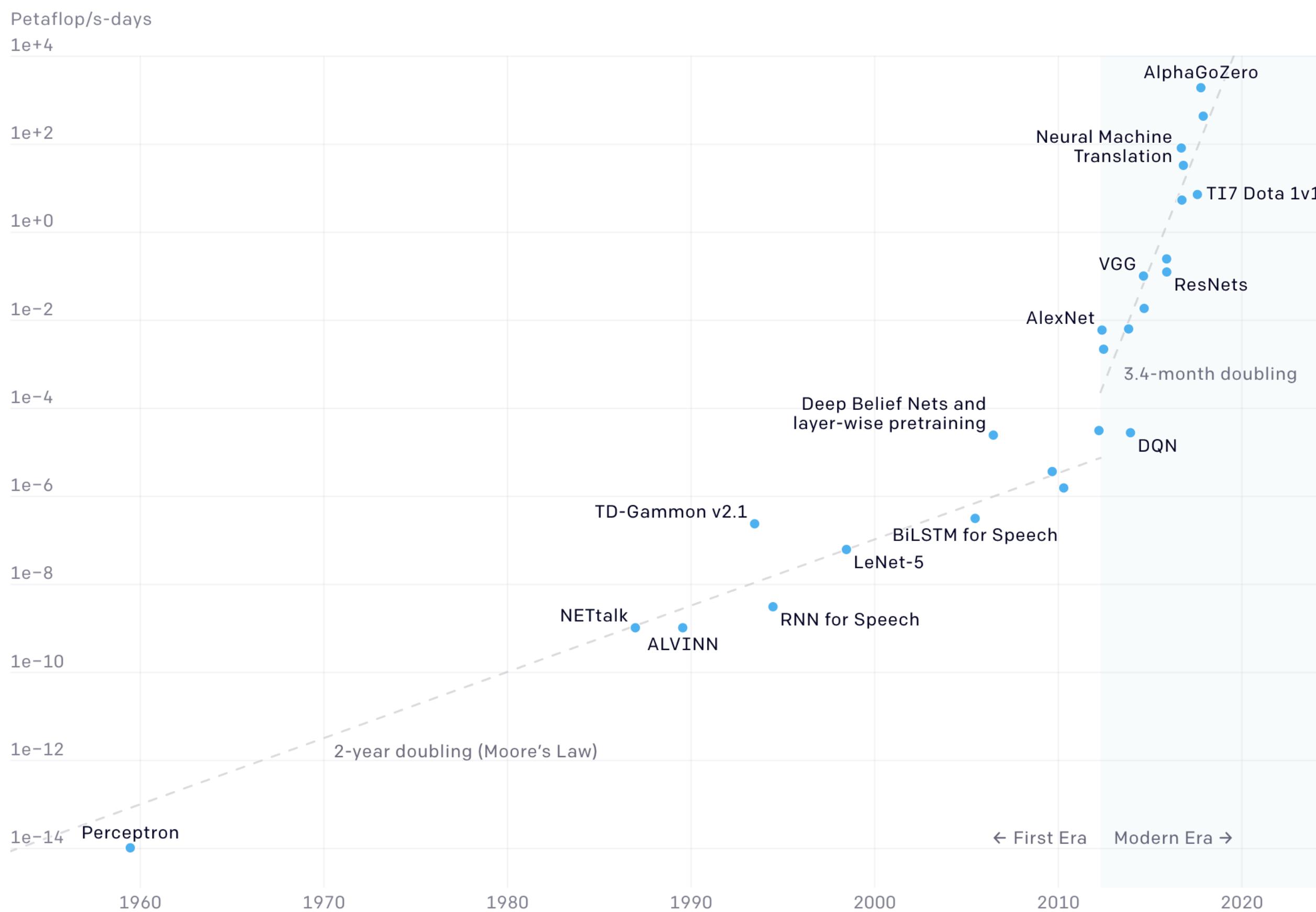
# Videogames and Neural Nets

- As it turns out, both 3D graphics and neural networks involve lots of **matrix multiplications**
- The demand for better gaming graphics drove better **Graphics Processing Units (GPUs)**
- The Deep Learning “Revolution” was partially driven by this progress in hardware



# Compute in Deep Learning

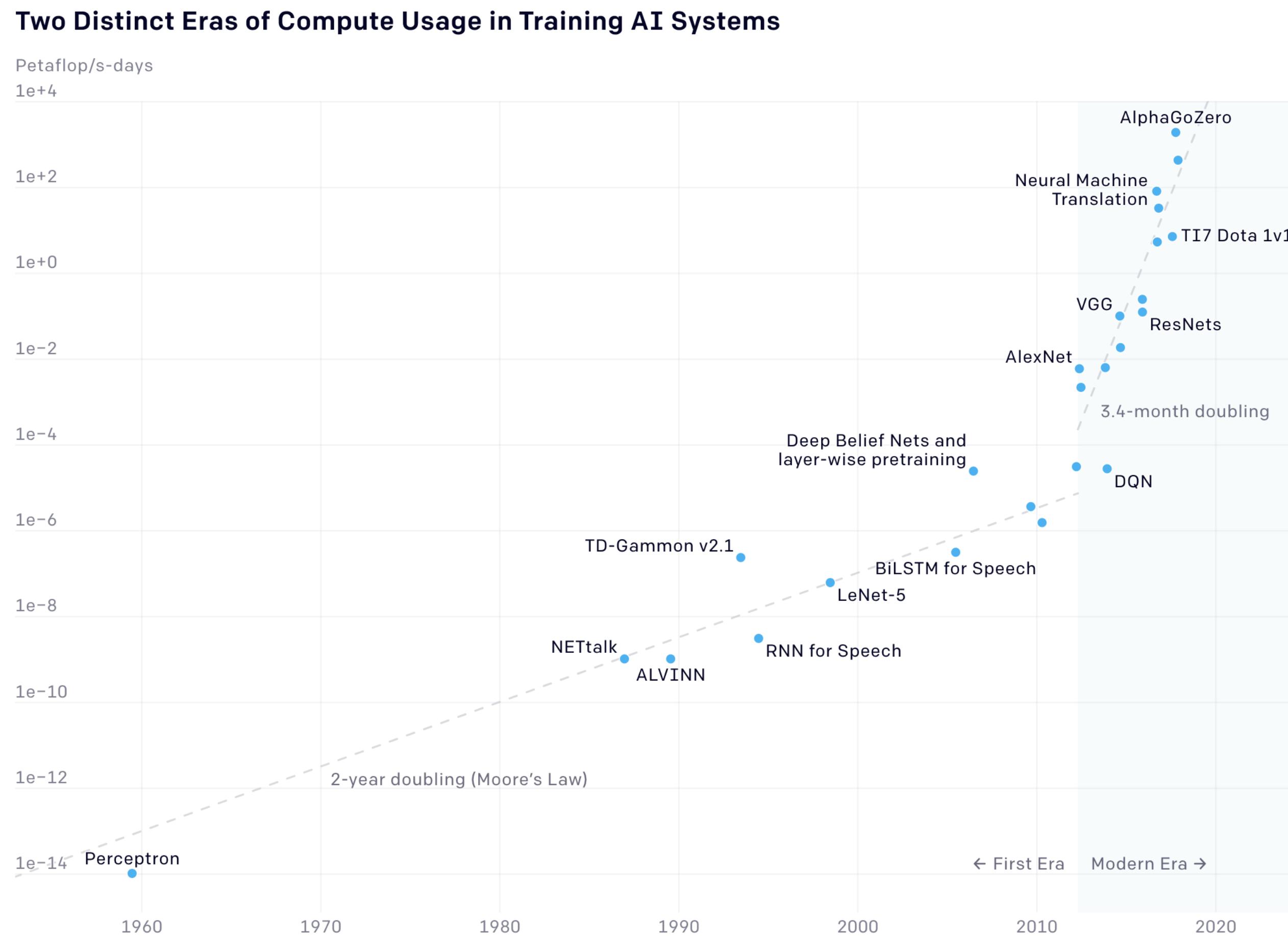
Two Distinct Eras of Compute Usage in Training AI Systems



[source](#)

# Compute in Deep Learning

log-scale!!



source

# Problems

# Problems

- Some areas are an ‘arms race’ between e.g. OpenAI, Meta, Google, MS, Baidu, ...

# Problems

- Some areas are an ‘arms race’ between e.g. OpenAI, Meta, Google, MS, Baidu, ...
- Hugely expensive
  - Carbon emissions
  - Monetarily
    - Inequitable access

# Problems

- Some areas are an ‘arms race’ between e.g. Baidu, ...
- Hugely expensive
  - Carbon emissions
  - Monetarily
  - Inequitable access

## Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell   Ananya Ganesh   Andrew McCallum  
College of Information and Computer Sciences  
University of Massachusetts Amherst  
`{strubell, aganesh, mccallum}@cs.umass.edu`

### Abstract

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large networks trained on abundant data. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements depend on the availability of exceptionally large computational resources that necessitate similarly substantial energy consumption. As a result these models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor

	Consumption	CO <sub>2</sub> e (lbs)
Air travel, 1 person, NY↔SF		1984
Human life, avg, 1 year		11,023
American life, avg, 1 year		36,156
Car, avg incl. fuel, 1 lifetime		126,000
Training one model (GPU)		
NLP pipeline (parsing, SRL)	39	
w/ tuning & experiments	78,468	
Transformer (big)	192	
w/ neural arch. search	626,155	

Table 1: Estimated CO<sub>2</sub> emissions from training common NLP models, compared to familiar consumption.<sup>1</sup>

# Problems

- Some areas are an ‘arms race’ between e.g. Baidu, ...
- Hugely expensive
  - Carbon emissions
  - Monetarily
  - Inequitable access

## Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell   Ananya Ganesh   Andrew McCallum  
College of Information and Computer Sciences  
University of Massachusetts Amherst  
`{strubell, aganesh, mccallum}@cs.umass.edu`

### Green AI

Roy Schwartz\*<sup>◊</sup>   Jesse Dodge\*<sup>◊♣</sup>   Noah A. Smith<sup>◊♡</sup>   Oren Etzioni<sup>◊</sup>

<sup>◊</sup>Allen Institute for AI, Seattle, Washington, USA  
♣ Carnegie Mellon University, Pittsburgh, Pennsylvania, USA  
♡ University of Washington, Seattle, Washington, USA

July 2019

#### Abstract

The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018 [2]. These computations have a surprisingly large carbon footprint [40]. Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Moreover, the financial cost of the computations can make it difficult for academics, students, and researchers, in particular those from emerging economies, to engage in deep learning research.

This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or “price tag” of developing, training, and running models to provide baselines for the investigation of increasingly efficient methods. Our goal is to make AI both greener and more inclusive—enabling any inspired undergraduate with a laptop to write high-quality research papers. **Green AI** is an emerging focus at the Allen Institute for AI.

Consumption	CO <sub>2</sub> e (lbs)
Air travel, 1 person, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experiments	78,468
Transformer (big)	192
w/ neural arch. search	626,155

Table 1: Estimated CO<sub>2</sub> emissions from training common NLP models, compared to familiar consumption.<sup>1</sup>

# Potted History of Models in NLP

# Four Broad “Eras”

# Four Broad “Eras”

- 100% rule-based systems (1960s on)

# Four Broad “Eras”

- 100% rule-based systems (1960s on)
- Early Machine Learning (mid-80s - mid-90s)
  - Decision trees, naive bayes, etc

# Four Broad “Eras”

- 100% rule-based systems (1960s on)
- Early Machine Learning (mid-80s - mid-90s)
  - Decision trees, naive bayes, etc
- Log-linear (i.e. maxent) models (mid-90s - mid-2010s)

# Four Broad “Eras”

- 100% rule-based systems (1960s on)
- Early Machine Learning (mid-80s - mid-90s)
  - Decision trees, naive bayes, etc
- Log-linear (i.e. maxent) models (mid-90s - mid-2010s)
- Neural networks (2013 - now)

# Four Broad “Eras”

- 100% rule-based systems (1960s on)
- Early Machine Learning (mid-80s - mid-90s)
  - Decision trees, naive bayes, etc
- Log-linear (i.e. maxent) models (mid-90s - mid-2010s)
- Neural networks (2013 - now)
- **All** of these are still used in applications in every area!
  - They all have different strengths and weaknesses

# Early NLP Systems (1960s-1990s)

- SHRDLU, e.g.:

Person: Pick up a big red block.

Computer: OK.

Person: Grasp the pyramid.

Computer: I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.

Person (changing their mind): Find a block which is taller than the one you are holding and put it into the box.

Computer: BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN THE ONE I AM HOLDING.

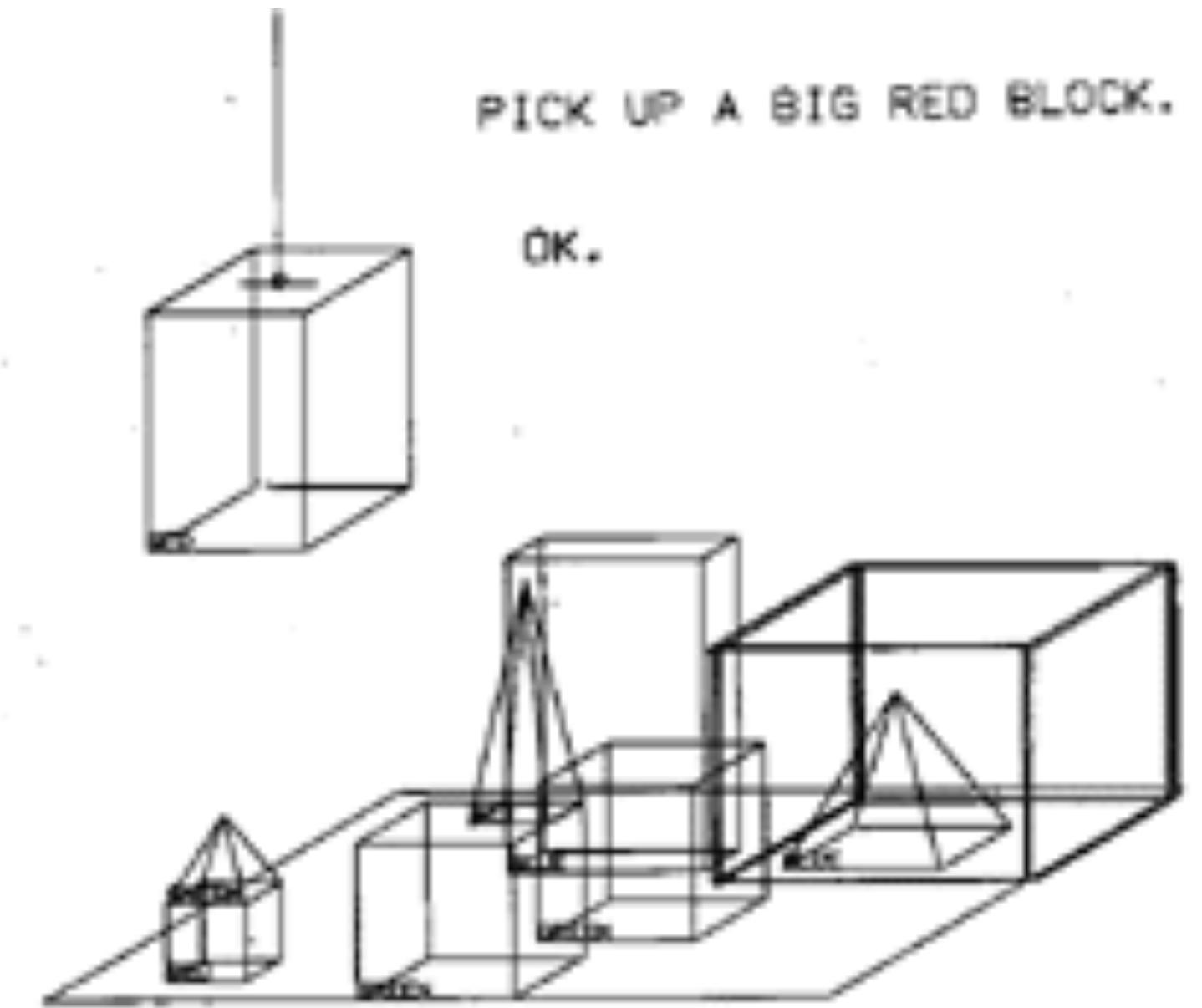
Computer: OK.

Person: What does the box contain?

Computer: THE BLUE PYRAMID AND THE BLUE BLOCK.

Person: What is the pyramid supported by?

Computer: THE BOX.



# Early NLP Systems (1960s-1990s)

# Early NLP Systems (1960s-1990s)

- 100% rule-based, hand-written algorithms

# Early NLP Systems (1960s-1990s)

- 100% rule-based, hand-written algorithms
- Lots of energy in ontology development / knowledge representation

# Early NLP Systems (1960s-1990s)

- 100% rule-based, hand-written algorithms
- Lots of energy in ontology development / knowledge representation
- Exhibit many core features of human linguistic competence
  - Compositional generalization
  - Abstract representations of meaning

# Early NLP Systems (1960s-1990s)

- 100% rule-based, hand-written algorithms
- Lots of energy in ontology development / knowledge representation
- Exhibit many core features of human linguistic competence
  - Compositional generalization
  - Abstract representations of meaning
- Fully “interpretable”, because fully engineered

# Early NLP Systems (1960s-1990s)

- 100% rule-based, hand-written algorithms
- Lots of energy in ontology development / knowledge representation
- Exhibit many core features of human linguistic competence
  - Compositional generalization
  - Abstract representations of meaning
- Fully “interpretable”, because fully engineered
- But: brittle, no graceful degradation, domain-specific

# Early ML (80s-90s)

# Early ML (80s-90s)

- Increase in compute power, availability of larger corpora for parameter estimation

# Early ML (80s-90s)

- Increase in compute power, availability of larger corpora for parameter estimation
- Generally, *generative models* (i.e. models of joint distribution  $P(x, y)$ )
  - N-grams, Naive Bayes, HMMs, PCFGs, ...

# Early ML (80s-90s)

- Increase in compute power, availability of larger corpora for parameter estimation
- Generally, *generative models* (i.e. models of joint distribution  $P(x, y)$ )
  - N-grams, Naive Bayes, HMMs, PCFGs, ...
- Parameter estimation via counting = very simple training

# Early ML (80s-90s)

- Increase in compute power, availability of larger corpora for parameter estimation
- Generally, *generative models* (i.e. models of joint distribution  $P(x, y)$ )
  - N-grams, Naive Bayes, HMMs, PCFGs, ...
- Parameter estimation via counting = very simple training
- Generally relies on heavy use of feature engineering

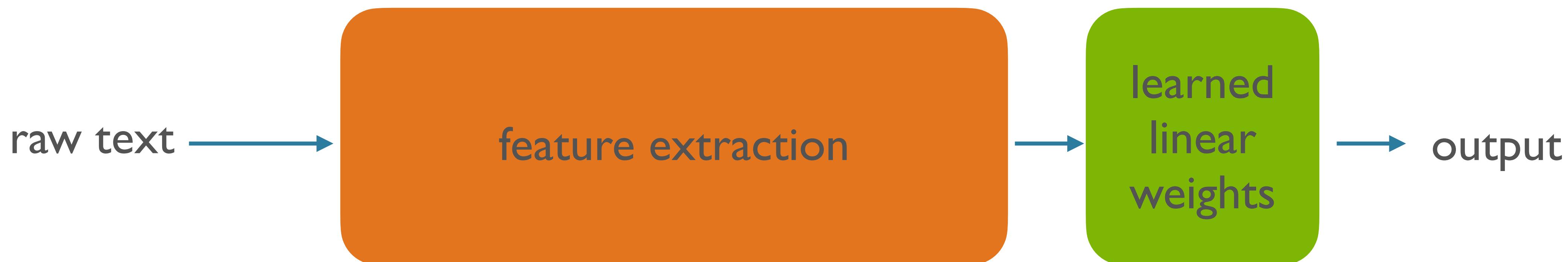
# Early ML (80s-90s)

- Increase in compute power, availability of larger corpora for parameter estimation
- Generally, *generative models* (i.e. models of joint distribution  $P(x, y)$ )
  - N-grams, Naive Bayes, HMMs, PCFGs, ...
- Parameter estimation via counting = very simple training
- Generally relies on heavy use of feature engineering
- Still work surprisingly well! Always try them first.

# Log-linear models

- Aka maximum entropy (maxent), multinomial classifiers, softmax, ...
- **Discriminative** models (i.e. of  $P(y | x)$ )

$$P(y | x) \propto e^{\sum_j w_j f_j(x, y)}$$



# Log-linear models

	O	LOC	MISC	ORG	PER
WORDS					
PWORD:at	-0.18	0.94	-0.31	0.28	-0.73
CWORD:Grace	-0.01	0	0	-0.02	0.03
NWORD:Road	0.02	0.27	-0.01	-0.25	-0.03
PWORD-CWORD:at-Grace	0	0	0	0	1.0
CWORD-NWORD:Grace-Road	0	0	0	0	0
NGRAMS (prefix/suffix only here)					
⟨G	-0.57	-0.04	0.26	-0.04	0.45
⟨Gr	0.27	-0.06	0.12	-0.17	-0.16
⟨Gra	-0.01	-0.37	0.19	-0.09	0.28
⟨Grac	-0.01	0	0	-0.02	0.03
⟨Grace	-0.01	0	0	-0.02	0.03
⟨Grace⟩	-0.01	0	0	-0.02	0.03
Grace⟩	-0.01	0	0	-0.02	0.03
race⟩	0	0	0	-0.02	0.03
ace⟩	0.08	0.24	0.07	-0.30	-0.10
ce⟩	0.44	0.31	-0.34	-0.02	-0.38
e⟩	0.38	-0.14	-0.18	-0.06	0
TAGS					
PTAG:IN	-0.40	0.24	0.16	0.08	-0.08
CTAG>NNP	-1.09	0.45	-0.26	0.43	0.47
NTAG>NNP	0.05	-0.19	0.18	-0.12	0.08
PTAG-CTAG:IN-NNP	0	0.14	-0.03	-0.01	-0.10
CTAG-NTAG>NNP-NNP	-0.11	-0.05	0	-0.38	-0.54
TYPES					
PTYPE:x:2	-0.07	-0.15	0.35	0.18	-0.31
CTYPE:Xx	-2.02	0.46	0.19	0.57	0.80
NTYPE:Xx	-0.22	-0.42	-0.19	0.29	0.54
PTYPE-CTYPE:x:2-Xx	-0.20	0.08	0.10	0.10	-0.09
CTYPE-NTYPE:Xx-Xx	0.55	-0.13	-0.55	-0.13	0.26
PTYPE-CTYPE-NTYPE:x:2-Xx-Xx	0.10	0.37	0.10	0.12	-0.69
WORDS/TYPES					
PWORD-CTYPE:at-Xx	-0.21	0.57	-0.21	0.41	-0.56
CTYPE-NWORD:Xx-Road	-0.01	0.27	-0.01	-0.23	-0.03
STATES					
PSTATE:O	2.91	-0.92	-0.72	-0.58	-0.70
PPSTATE-PSTATE:O-O	1.14	-0.60	-0.08	-0.43	-0.04
WORDS/STATES					
PSTATE-CWORD:O-Grace	-0.01	0	0	-0.02	0.03
TAGS/STATES					
PSTATE-PTAG-CTAG:O-IN-NNP	0.12	0.59	-0.29	-0.28	-0.14
PPSTATE-PPTAG-PSTATE-PTAG-CTAG:O-NN-O-IN-NNP	0.01	-0.03	-0.31	0.31	0.01
TYPES/STATES					
PSTATE-CTYPE:O-Xx	-1.13	0.37	-0.12	0.20	0.68
PSTATE-NTYPE:O-Xx	-0.69	-0.3	0.29	0.39	0.30
PSTATE-PTYPE-CTYPE:O-x:2-Xx	-0.28	0.82	-0.10	-0.26	-0.20
PPSTATE-PPTYPE-PSTATE-PTYPE-CTYPE:O-x-O-x:2-Xx	-0.22	-0.04	-0.04	-0.06	0.22
Total:	-1.40	2.68	-1.74	-0.19	-0.58



# Log-linear models

- Learnable using standard optimization methods

	O	LOC	MISC	ORG	PER
WORDS					
PWORD:at	-0.18	0.94	-0.31	0.28	-0.73
CWORD:Grace	-0.01	0	0	-0.02	0.03
NWORD:Road	0.02	0.27	-0.01	-0.25	-0.03
PWORD-CWORD:at-Grace	0	0	0	0	1.0
CWORD-NWORD:Grace-Road	0	0	0	0	0
NGRAMS (prefix/suffix only here)					
⟨G	-0.57	-0.04	0.26	-0.04	0.45
⟨Gr	0.27	-0.06	0.12	-0.17	-0.16
⟨Gra	-0.01	-0.37	0.19	-0.09	0.28
⟨Grac	-0.01	0	0	-0.02	0.03
⟨Grace	-0.01	0	0	-0.02	0.03
⟨Grace⟩	-0.01	0	0	-0.02	0.03
Grace⟩	-0.01	0	0	-0.02	0.03
race⟩	0	0	0	-0.02	0.03
ace⟩	0.08	0.24	0.07	-0.30	-0.10
ce⟩	0.44	0.31	-0.34	-0.02	-0.38
e⟩	0.38	-0.14	-0.18	-0.06	0
TAGS					
PTAG:IN	-0.40	0.24	0.16	0.08	-0.08
CTAG>NNP	-1.09	0.45	-0.26	0.43	0.47
NTAG>NNP	0.05	-0.19	0.18	-0.12	0.08
PTAG-CTAG:IN-NNP	0	0.14	-0.03	-0.01	-0.10
CTAG-NTAG>NNP-NNP	-0.11	-0.05	0	-0.38	-0.54
TYPES					
PTYPE:x:2	-0.07	-0.15	0.35	0.18	-0.31
CTYPE:Xx	-2.02	0.46	0.19	0.57	0.80
NTYPE:Xx	-0.22	-0.42	-0.19	0.29	0.54
PTYPE-CTYPE:x:2-Xx	-0.20	0.08	0.10	0.10	-0.09
CTYPE-NTYPE:Xx-Xx	0.55	-0.13	-0.55	-0.13	0.26
PTYPE-CTYPE-NTYPE:x:2-Xx-Xx	0.10	0.37	0.10	0.12	-0.69
WORDS/TYPES					
PWORD-CTYPE:at-Xx	-0.21	0.57	-0.21	0.41	-0.56
CTYPE-NWORD:Xx-Road	-0.01	0.27	-0.01	-0.23	-0.03
STATES					
PSTATE:O	2.91	-0.92	-0.72	-0.58	-0.70
PPSTATE-PSTATE:O-O	1.14	-0.60	-0.08	-0.43	-0.04
WORDS/STATES					
PSTATE-CWORD:O-Grace	-0.01	0	0	-0.02	0.03
TAGS/STATES					
PSTATE-PTAG-CTAG:O-IN-NNP	0.12	0.59	-0.29	-0.28	-0.14
PPSTATE-PPTAG-PSTATE-PTAG-CTAG:O-NN-O-IN-NNP	0.01	-0.03	-0.31	0.31	0.01
TYPES/STATES					
PSTATE-CTYPE:O-Xx	-1.13	0.37	-0.12	0.20	0.68
PSTATE-NTYPE:O-Xx	-0.69	-0.3	0.29	0.39	0.30
PSTATE-PTYPE-CTYPE:O-x:2-Xx	-0.28	0.82	-0.10	-0.26	-0.20
PPSTATE-PPTYPE-PSTATE-PTYPE-CTYPE:O-x-O-x:2-Xx	-0.22	-0.04	-0.04	-0.06	0.22
Total:	-1.40	2.68	-1.74	-0.19	-0.58

# Log-linear models

- Learnable using standard optimization methods
- Interpretable: can see feature importance
  - e.g. [Klein et al 2003](#) on Named Entity Recognition:
    - Weight for class PER for feature CURWORD:Grace: 0.03
    - Weight for class PER for prefix “<G”: 0.45

	O	LOC	MISC	ORG	PER
WORDS					
PWORD:at	-0.18	0.94	-0.31	0.28	-0.73
CWORD:Grace	-0.01	0	0	-0.02	0.03
NWORD:Road	0.02	0.27	-0.01	-0.25	-0.03
PWORD-CWORD:at-Grace	0	0	0	0	1.0
CWORD-NWORD:Grace-Road	0	0	0	0	0
NGRAMS (prefix/suffix only here)					
<G	-0.57	-0.04	0.26	-0.04	0.45
<Gr	0.27	-0.06	0.12	-0.17	-0.16
<Gra	-0.01	-0.37	0.19	-0.09	0.28
<Grac	-0.01	0	0	-0.02	0.03
<Grace	-0.01	0	0	-0.02	0.03
<Grace>	-0.01	0	0	-0.02	0.03
Grace>	-0.01	0	0	-0.02	0.03
race>	0	0	0	-0.02	0.03
ace>	0.08	0.24	0.07	-0.30	-0.10
ce>	0.44	0.31	-0.34	-0.02	-0.38
e>	0.38	-0.14	-0.18	-0.06	0
TAGS					
PTAG:IN	-0.40	0.24	0.16	0.08	-0.08
CTAG>NNP	-1.09	0.45	-0.26	0.43	0.47
NTAG>NNP	0.05	-0.19	0.18	-0.12	0.08
PTAG-CTAG:IN-NNP	0	0.14	-0.03	-0.01	-0.10
CTAG-NTAG>NNP-NNP	-0.11	-0.05	0	-0.38	-0.54
TYPES					
PTYPE:x:2	-0.07	-0.15	0.35	0.18	-0.31
CTYPE:Xx	-2.02	0.46	0.19	0.57	0.80
NTYPE:Xx	-0.22	-0.42	-0.19	0.29	0.54
PTYPE-CTYPE:x:2-Xx	-0.20	0.08	0.10	0.10	-0.09
CTYPE-NTYPE:Xx-Xx	0.55	-0.13	-0.55	-0.13	0.26
PTYPE-CTYPE-NTYPE:x:2-Xx-Xx	0.10	0.37	0.10	0.12	-0.69
WORDS/TYPES					
PWORD-CTYPE:at-Xx	-0.21	0.57	-0.21	0.41	-0.56
CTYPE-NWORD:Xx-Road	-0.01	0.27	-0.01	-0.23	-0.03
STATES					
PSTATE:O	2.91	-0.92	-0.72	-0.58	-0.70
PPSTATE-PSTATE:O-O	1.14	-0.60	-0.08	-0.43	-0.04
WORDS/STATES					
PSTATE-CWORD:O-Grace	-0.01	0	0	-0.02	0.03
TAGS/STATES					
PSTATE-PTAG-CTAG:O-IN-NNP	0.12	0.59	-0.29	-0.28	-0.14
PPSTATE-PPTAG-PSTATE-PTAG-CTAG:O-NN-O-IN-NNP	0.01	-0.03	-0.31	0.31	0.01
TYPES/STATES					
PSTATE-CTYPE:O-Xx	-1.13	0.37	-0.12	0.20	0.68
PSTATE-NTYPE:O-Xx	-0.69	-0.3	0.29	0.39	0.30
PSTATE-PTYPE-CTYPE:O-x:2-Xx	-0.28	0.82	-0.10	-0.26	-0.20
PPSTATE-PPTYPE-PSTATE-PTYPE-CTYPE:O-x-O-x:2-Xx	-0.22	-0.04	-0.04	-0.06	0.22
Total:	-1.40	2.68	-1.74	-0.19	-0.58



# Log-linear models

- Learnable using standard optimization methods
- Interpretable: can see feature importance
  - e.g. [Klein et al 2003](#) on Named Entity Recognition:
    - Weight for class PER for feature CURWORD:Grace: 0.03
    - Weight for class PER for prefix “<G”: 0.45
- Feature engineering:
  - Expensive
  - Incomplete
  - Sparse (= wasted compute as well)

	O	LOC	MISC	ORG	PER
WORDS					
PWORD:at	-0.18	0.94	-0.31	0.28	-0.73
CWORD:Grace	-0.01	0	0	-0.02	0.03
NWORD:Road	0.02	0.27	-0.01	-0.25	-0.03
PWORD-CWORD:at-Grace	0	0	0	0	1.0
CWORD-NWORD:Grace-Road	0	0	0	0	0
NGRAMS (prefix/suffix only here)					
<G	-0.57	-0.04	0.26	-0.04	0.45
<Gr	0.27	-0.06	0.12	-0.17	-0.16
<Gra	-0.01	-0.37	0.19	-0.09	0.28
<Grac	-0.01	0	0	-0.02	0.03
<Grace	-0.01	0	0	-0.02	0.03
<Grace>	-0.01	0	0	-0.02	0.03
Grace>	-0.01	0	0	-0.02	0.03
race>	0	0	0	-0.02	0.03
ace>	0.08	0.24	0.07	-0.30	-0.10
ce>	0.44	0.31	-0.34	-0.02	-0.38
e>	0.38	-0.14	-0.18	-0.06	0
TAGS					
PTAG:IN	-0.40	0.24	0.16	0.08	-0.08
CTAG>NNP	-1.09	0.45	-0.26	0.43	0.47
NTAG>NNP	0.05	-0.19	0.18	-0.12	0.08
PTAG-CTAG:IN>NNP	0	0.14	-0.03	-0.01	-0.10
CTAG-NTAG>NNP>NNP	-0.11	-0.05	0	-0.38	-0.54
TYPES					
PTYPE:x:2	-0.07	-0.15	0.35	0.18	-0.31
CTYPE:Xx	-2.02	0.46	0.19	0.57	0.80
NTYPE:Xx	-0.22	-0.42	-0.19	0.29	0.54
PTYPE-CTYPE:x:2-Xx	-0.20	0.08	0.10	0.10	-0.09
CTYPE-NTYPE:Xx-Xx	0.55	-0.13	-0.55	-0.13	0.26
PTYPE-CTYPE-NTYPE:x:2-Xx-Xx	0.10	0.37	0.10	0.12	-0.69
WORDS/TYPES					
PWORD-CTYPE:at-Xx	-0.21	0.57	-0.21	0.41	-0.56
CTYPE-NWORD:Xx-Road	-0.01	0.27	-0.01	-0.23	-0.03
STATES					
PSTATE:O	2.91	-0.92	-0.72	-0.58	-0.70
PPSTATE-PSTATE:O-O	1.14	-0.60	-0.08	-0.43	-0.04
WORDS/STATES					
PSTATE-CWORD:O-Grace	-0.01	0	0	-0.02	0.03
TAGS/STATES					
PSTATE-PTAG-CTAG:O-IN>NNP	0.12	0.59	-0.29	-0.28	-0.14
PPSTATE-PPTAG-PSTATE-PTAG-CTAG:O-NN-O-IN>NNP	0.01	-0.03	-0.31	0.31	0.01
TYPES/STATES					
PSTATE-CTYPE:O-Xx	-1.13	0.37	-0.12	0.20	0.68
PSTATE-NTYPE:O-Xx	-0.69	-0.3	0.29	0.39	0.30
PSTATE-PTYPE-CTYPE:O-x:2-Xx	-0.28	0.82	-0.10	-0.26	-0.20
PPSTATE-PPTYPE-PSTATE-PTYPE-CTYPE:O-x-O-x:2-Xx	-0.22	-0.04	-0.04	-0.06	0.22
Total:	-1.40	2.68	-1.74	-0.19	-0.58

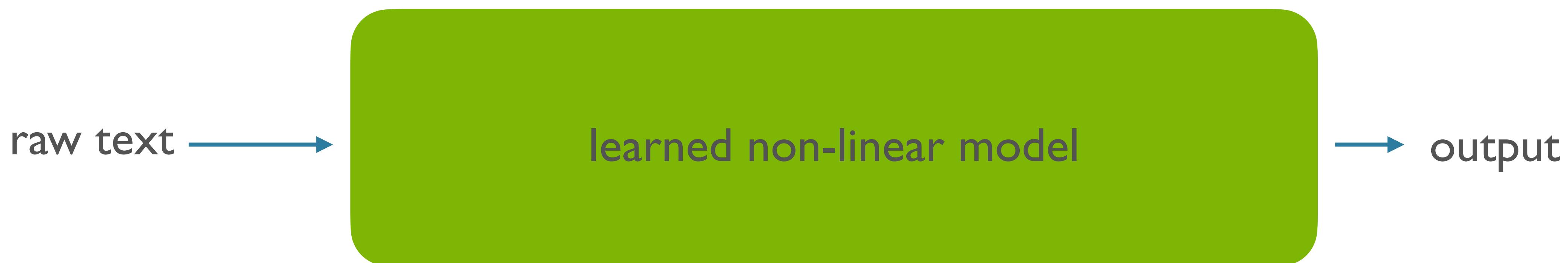
# Neural Networks

# Neural Networks

- Key idea: no feature engineering
  - Have a larger model *learn which features are useful*
  - (but can be combined with feature extraction as well)

# Neural Networks

- Key idea: no feature engineering
  - Have a larger model *learn which features are useful*
  - (but can be combined with feature extraction as well)
- “End-to-end” learning paradigm:



# Neural Networks (drawbacks)

# Neural Networks (drawbacks)

- “Black box”:
  - How do we know *what* the model has learned?
  - How can we trust it in deployment?
  - Often learns to solve a dataset, not a task; may be very different from our linguistic competence

# Neural Networks (drawbacks)

- “Black box”:
  - How do we know *what* the model has learned?
  - How can we trust it in deployment?
  - Often learns to solve a dataset, not a task; may be very different from our linguistic competence
- Larger and larger compute needs (equity, environmental costs)

# Neural Networks (drawbacks)

- “Black box”:
  - How do we know *what* the model has learned?
  - How can we trust it in deployment?
  - Often learns to solve a dataset, not a task; may be very different from our linguistic competence
- Larger and larger compute needs (equity, environmental costs)
- Larger and larger data needs
  - Documentation debt
  - Privacy concerns
  - Amplifying biases

# Neural Networks (drawbacks)

- “Black box”:
  - How do we know *what* the model has learned?
  - How can we trust it in deployment?
  - Often learns to solve a dataset, not a task
- Larger and larger compute needs (equitability)
- Larger and larger data needs
  - Documentation debt
  - Privacy concerns
  - Amplifying biases

## On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?

Emily M. Bender\*  
ebender@uw.edu  
University of Washington  
Seattle, WA, USA

Angelina McMillan-Major  
aymm@uw.edu  
University of Washington  
Seattle, WA, USA

Timnit Gebru\*  
timnit@blackinai.org  
Black in AI  
Palo Alto, CA, USA

Shmargaret Shmitchell  
shmargaret.shmitchell@gmail.com  
The Aether

### ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask:

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learn-



# Course Information / Overview

# Learning Objectives

# Learning Objectives

- Provide hands-on experience with building neural networks and using them for NLP tasks

# Learning Objectives

- Provide hands-on experience with building neural networks and using them for NLP tasks
- Theoretical understanding of building blocks
  - Linear Algebra
  - Computation graphs + gradient descent
  - Forward/backward API
  - Chain rule for computing gradients [backpropagation]

# Learning Objectives

- Provide hands-on experience with building neural networks and using them for NLP tasks
- Theoretical understanding of building blocks
  - Linear Algebra
  - Computation graphs + gradient descent
  - Forward/backward API
    - Chain rule for computing gradients [backpropagation]
- Various network architectures; their structure and biases

# Content

# Content

- Model architectures
  - Feed-forward networks
  - Recurrent networks
  - Transformers

# Content

- Model architectures
  - Feed-forward networks
  - Recurrent networks
  - Transformers
- Primary tasks:
  - Language modeling
  - Text classification (sentiment analysis in particular)
  - Translation

# Content

- Model architectures
  - Feed-forward networks
  - Recurrent networks
  - Transformers
- Primary tasks:
  - Language modeling
  - Text classification (sentiment analysis in particular)
  - Translation
- Pre-training + fine-tuning, “Large Language Models”

# Content, cont.

- Special topics:
  - Model interpretability
  - Multilingual models
  - Ethics/criticism
  - Application to Linguistics

# Course resources

# Course resources

- Web page: <https://cmdowney88.github.io/teaching/ling282/fall24>
  - Up-to-date syllabus
  - **Lecture slides** posted
  - Basic course information and policies
  - **Homeworks posted**

# Course resources

- Web page: <https://cmdowney88.github.io/teaching/ling282/fall24>
  - Up-to-date syllabus
  - **Lecture slides** posted
  - Basic course information and policies
  - **Homeworks posted**
- Blackboard
  - Signup link to **course Zulip** (messaging)
  - Homeworks **submitted via blackboard**

# Course resources

- Zulip
  - Need **UR email** to sign up
  - Use for **class-wide discussions**, e.g. on lectures and homeworks
  - Can direct-message me and other students
  - Please **remember to be respectful**. Consider this to be a classroom setting
  - Because of FERPA, I **cannot discuss grades** over Zulip

# Communication

# Communication

- Contacting me
  - If you prefer, you can use **Zulip** for course-related direct messages. Remember that I **cannot discuss grades** over Zulip
  - If you do send **email**, please include “Ling482” in your subject line
  - I will try to respond within 24 hours, but only during “business hours” during the week

# Communication

- Contacting me
  - If you prefer, you can use **Zulip** for course-related direct messages. Remember that I **cannot discuss grades** over Zulip
  - If you do send **email**, please include “Ling482” in your subject line
  - I will try to respond within 24 hours, but only during “business hours” during the week
- Zulip
  - All content and logistics questions
  - If you have the question, someone else does too. Someone else besides me might have the answer!

# Office hours



# Office hours

- Prof. C.M. Downey

# Office hours

- Prof. C.M. Downey
- Email: [c.m.downey@rochester.edu](mailto:c.m.downey@rochester.edu)

# Office hours

- Prof. C.M. Downey
- Email: [c.m.downey@rochester.edu](mailto:c.m.downey@rochester.edu)
- Office hours
  - TBD: please fill out the **when2meet on Zulip!**
  - Lattimore 507
  - Drop-in (no need to make an appointment)

# Homework assignments

# Homework assignments

- Due date: **Wednesdays at 11pm** unless specified otherwise
- The submission area closes two days after the due date
- Late penalty:
  - 5% for the 1st hour
  - 10% for the 1st 24 hours
  - 20% for the 1st 48 hours
- Your code must run, and will be tested, in the course Github codespace  
(tentative)

# Final grade

# Final grade

- Undergraduate (Ling 282)
  - Homeworks: 80%
  - Special topic presentation: 15%
  - Participation/attendance: 5%

# Final grade

- Undergraduate (Ling 282)
  - Homeworks: 80%
  - Special topic presentation: 15%
  - Participation/attendance: 5%
- Graduate (Ling 482)
  - Homeworks: 70%
  - Special topic presentation: 12.5%
  - **Critical review short paper:** 12.5%
  - Participation/attendance: 5%

# Assignment Overview

- Assignment 1: Linear Algebra and gradients basics (no coding)
- Assignment 2: Word2Vec, Backpropagation, and computational graphs
- Assignment 3: Feedforward NNs, language modeling
- Assignment 4: RNN language models
- Assignment 5: RNN variants
- Assignment 6: Seq2Seq, attention, tokenization, and NMT
- Assignment 7: Transformers and pre-training
- Assignment 8: “Large Language Models”

# Next Time

- Linear Algebra basics
  - vectors
  - matrices
  - matrix multiplication
  - span, matrix rank
  - linear transformations

# Time for questions/discussion



# Looking forward to a great semester!