Neural Networks: Introduction

LING572 Advanced Statistical Methods for NLP

February 25 2020

Unit Overview

- Introduction: History; Main Ideas / Basic Computation; Landscape
- Computation in feed-forward networks + Beginning of Learning
- Backpropagation
- Recurrent networks
- Transformers + transfer learning

Overview of Today

- Overview / Motivation
- History
- Computation: Simple Example
- Landscape

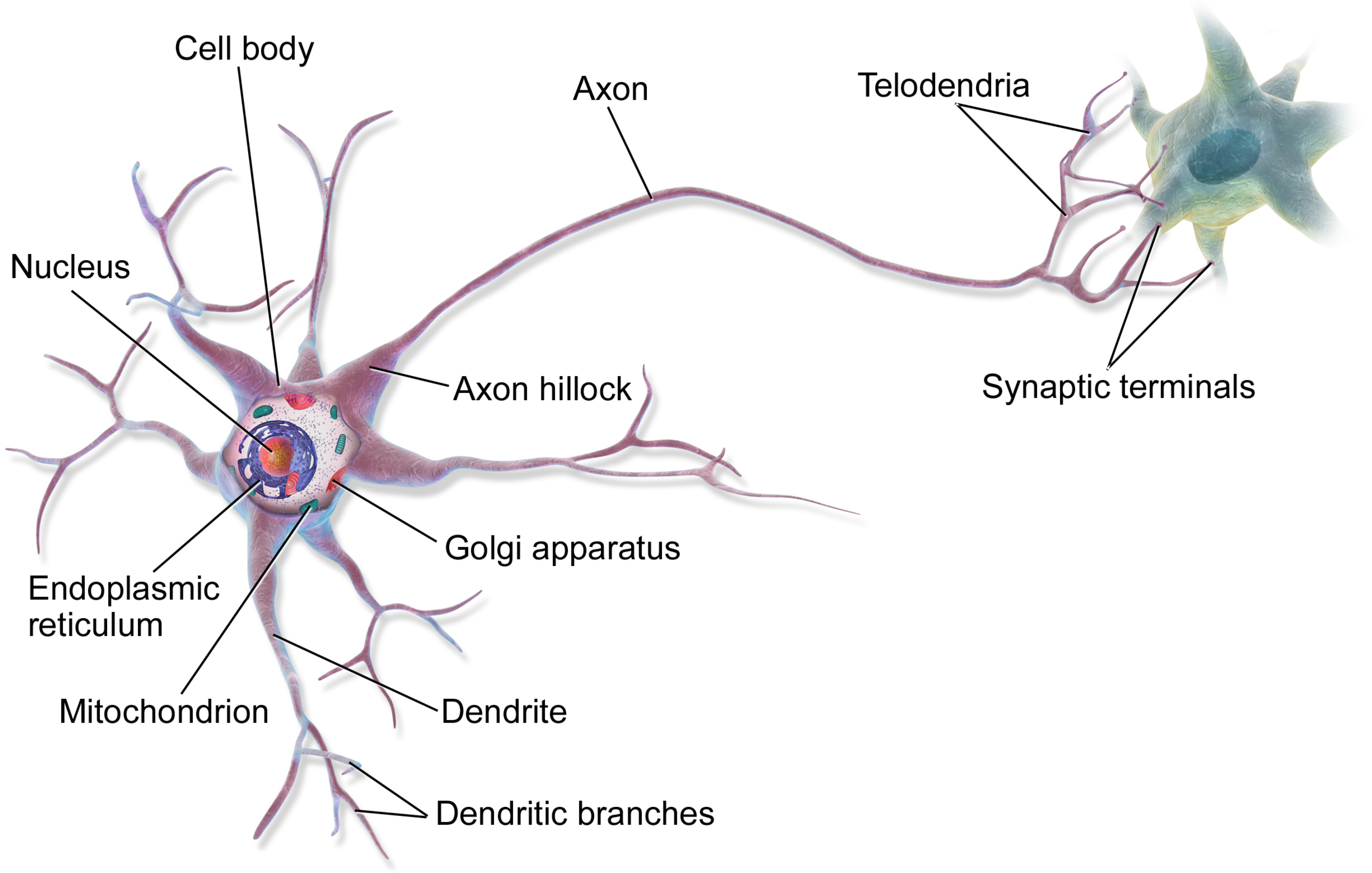
High-level Overview

What is a neural network?

- A *network* of artificial "*neurons*"
  - What's a neuron?
  - How are they connected in a network? Why do that?
- The networks *learns representations* of its input that are helpful in predicting desired outputs.
- In many cases, they are *universal function approximators*. (To be made precise later.)
  - But getting good approximations in practice is non-trivial.
- Dominating applied AI in many areas, including NLP, at the moment.

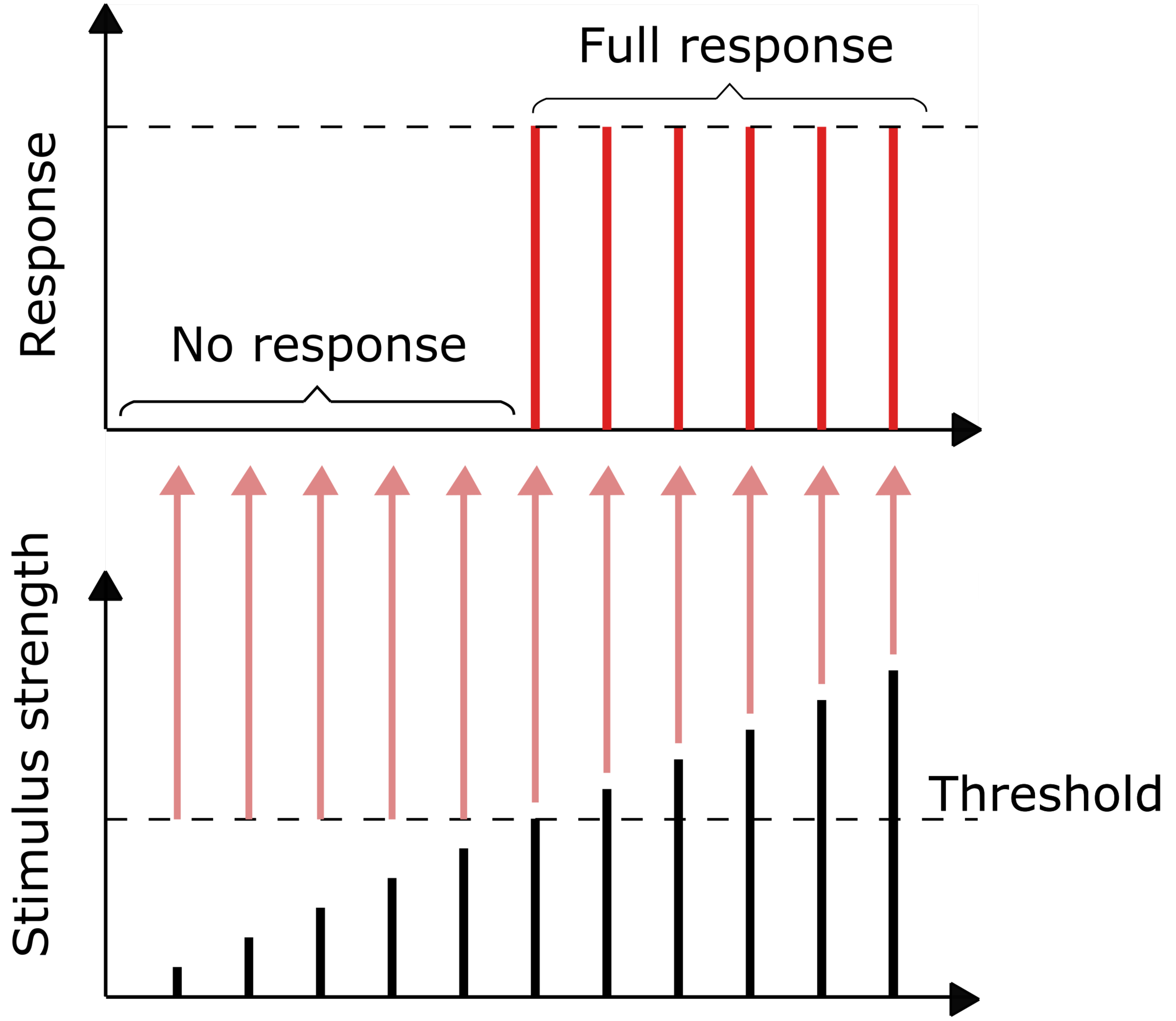
“Biological” Motivation

- Neuron: receives electrical impulses from others through its synapses.
  - Different connections have different strengths.
- Integrates these signals in its cell body.
- “Activates” if threshold passed.
  - Sends signal down dendrites to others that it’s connected to.



All-or-none Response

- Neuron: receives electrical impulses from others through its synapses.
  - Different connections have different strengths.
- Integrates these signals in its cell body.
- “Activates” if threshold passed.
  - Sends signal down dendrites to others that it’s connected to.



Some stats

- Number of neurons: ~100 billion
- Connections per neuron: ~10,000
- Strength of each connection adapts in the course of learning.

Engineering perspective

- MaxEnt (i.e. multinomial logistic regression):

$$y = \text{softmax}(w \cdot \boxed{f(x, y)})$$

Engineered feature vector

- Feed-forward neural network:

$$y = \text{softmax}(w \cdot \boxed{f_n(W_n(\cdots f_2(W_2 f_1(W_1 x))\cdots))})$$

Learned (and "hierarchical") feature vector

Why neural networks?

- Distributed representations:
  - Earlier NLP systems can be fragile, because of atomic symbol representations
    - e.g. “king” is as different from “queen” as from “bookshelf”
  - Learned word representations help enormously (cf 570, 571):
    - Lower dimensionality: breaks curse of dimensionality, *and* hopefully represents similarity structure
    - Can use larger contexts, beyond small *n*-grams
  - Beyond words: sentences, documents, …

Why neural networks? Learning Representations

- Handcrafting / engineering features is time-consuming
  - With no guarantee that the features you design will be the “right” ones for solving your task
- Representation learning: automatically learn good/useful features
  - (NB: one of the top ML conferences is ICLR = International Conference on Learning Representations)
- Deep learning: attempts to learn multiple levels of representation of increasing complexity/abstraction
- Good intermediate representations can be shared across tasks and languages (e.g. multi-task learning, transfer learning)

History

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

Turing Award: 2018

A.M. TURING AWARD WINNERS BY...

ALPHABETICAL LISTING | YEAR OF THE AWARD | RESEARCH SUBJECT

Yoshua Bengio

Geoffrey E Hinton

Yann LeCun

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

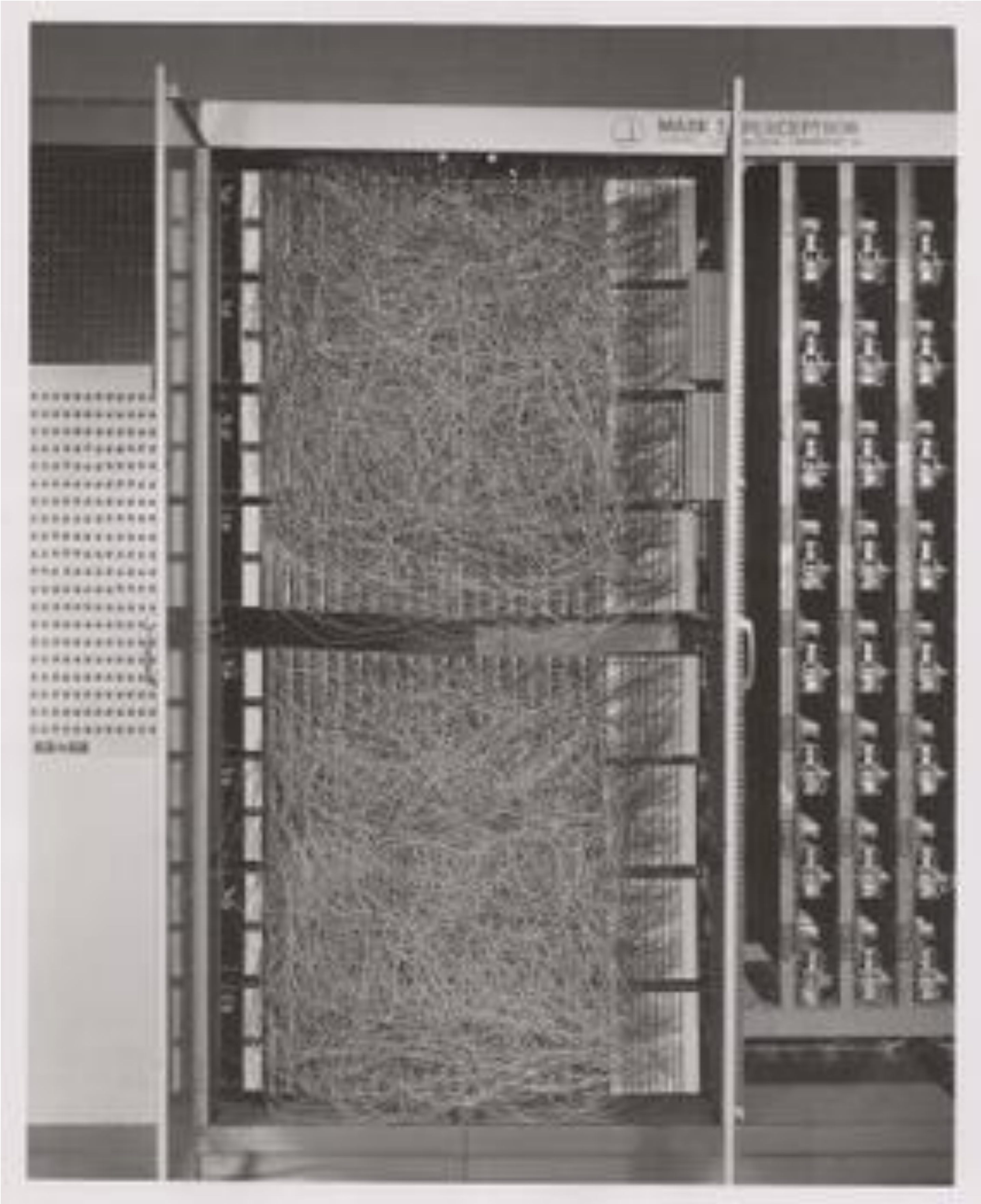
FATHERS OF THE DEEP LEARNING REVOLUTION RECEIVE ACM A.M. TURING AWARD

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)

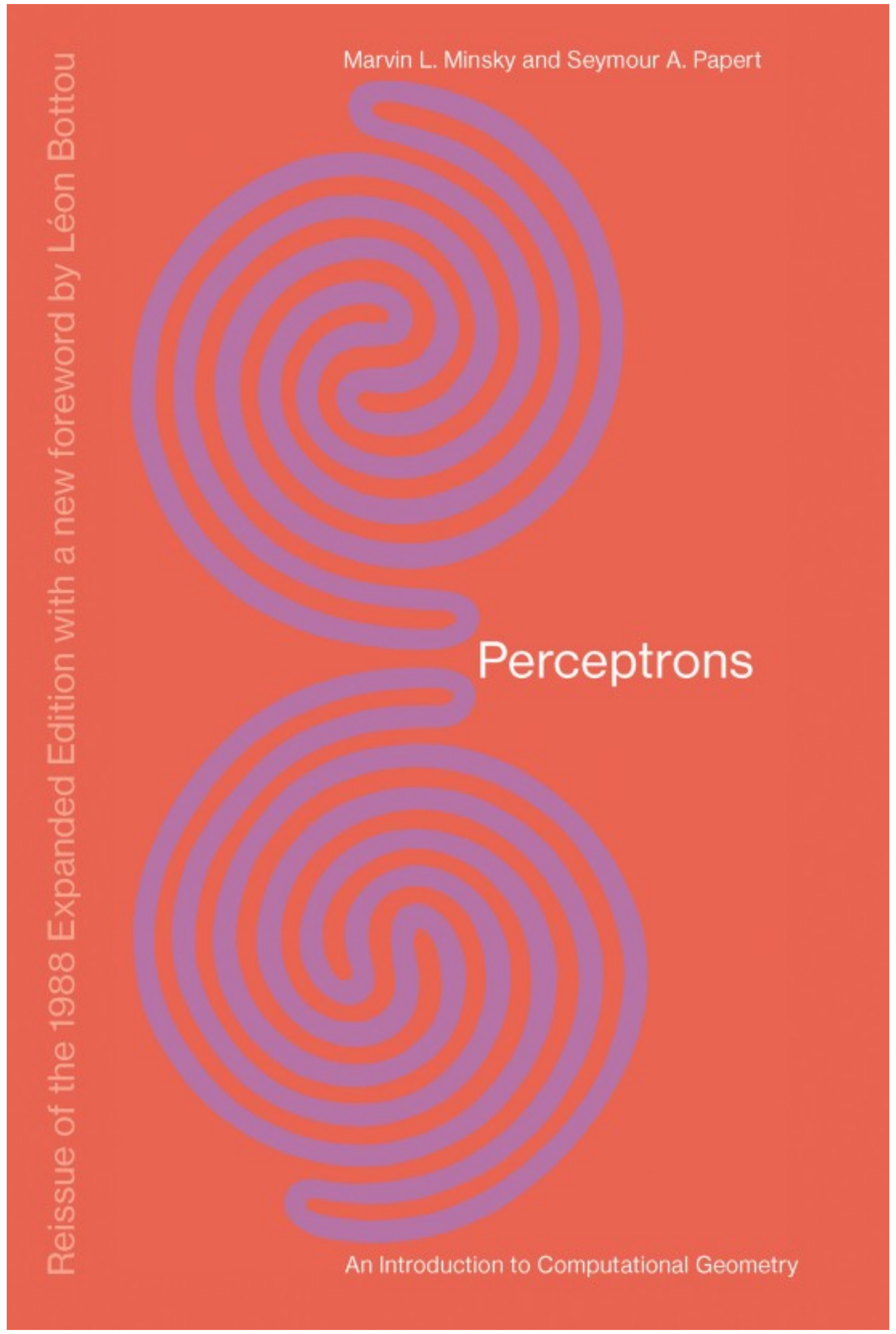


source

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

“"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

Perceptrons (1969)

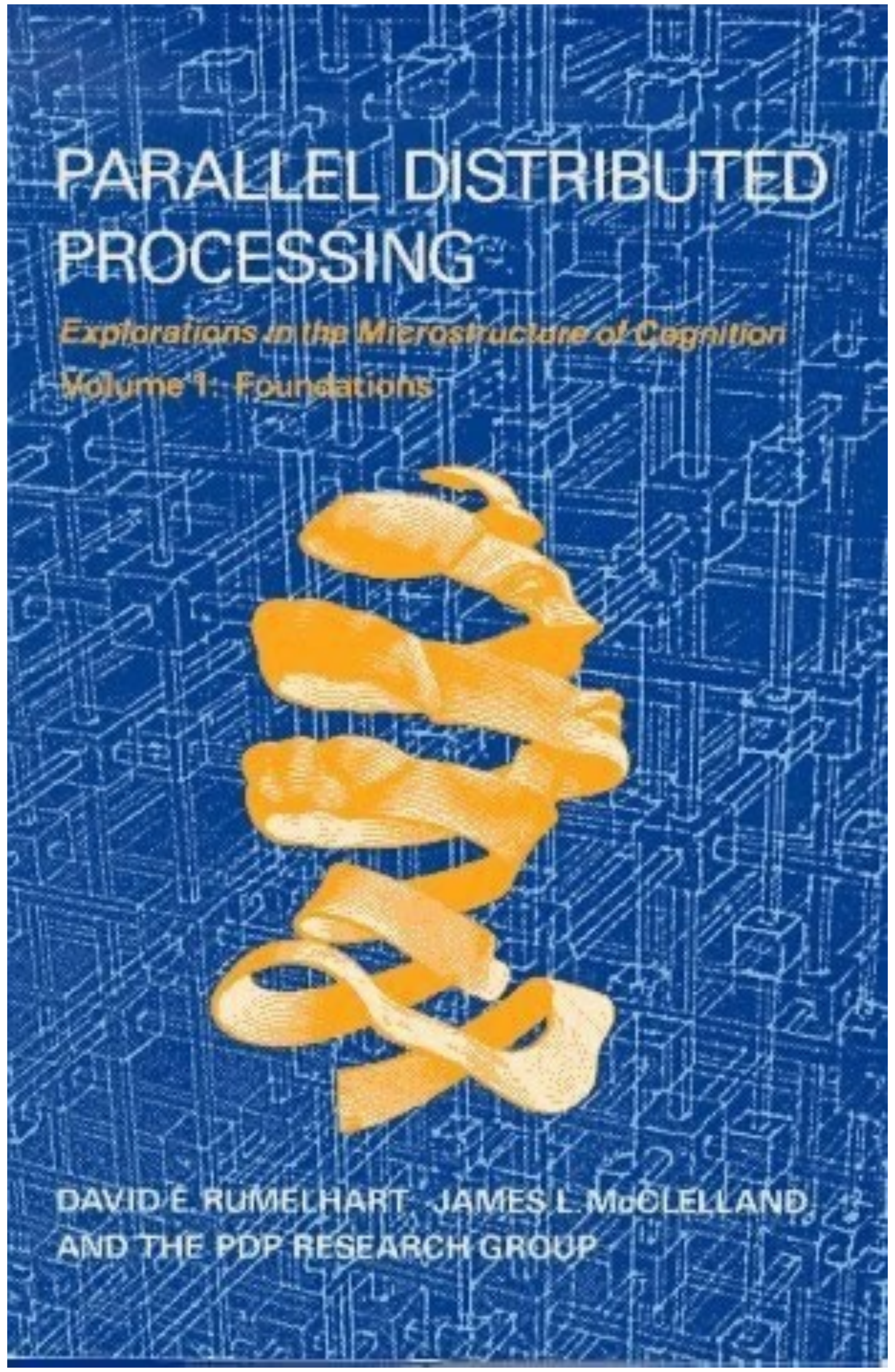


- Limitative results on functions computable by the basic perceptron
- Famous example (we'll return to it later):
  - Exclusive disjunction (XOR) is not computable
- Other examples that are uncomputable assuming *local connectivity*

AI Winter

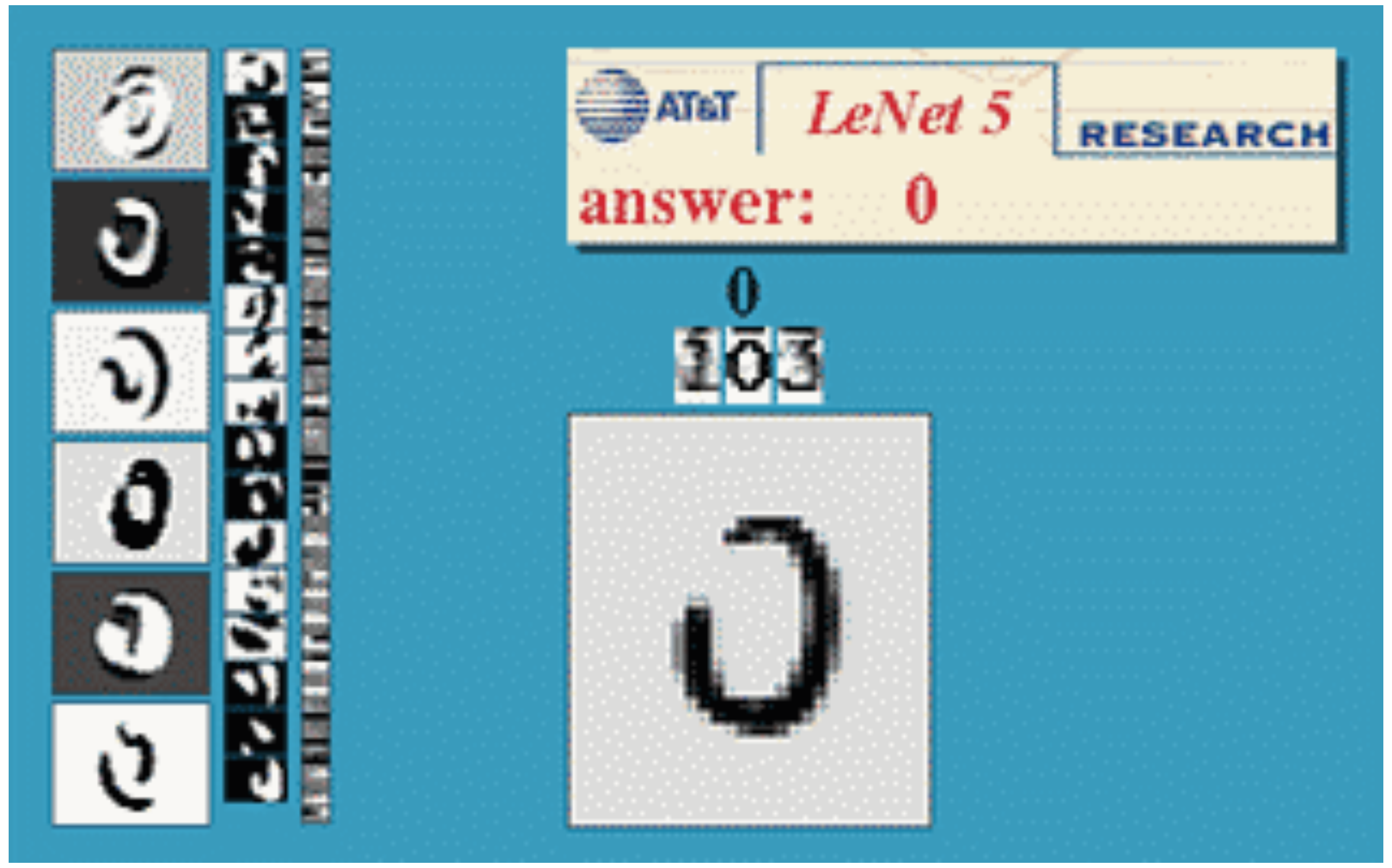
- Reaction to the results:
  - The approach of learning perceptrons for data cannot deliver on the promises
  - Funding from e.g. government agencies dried up significantly
  - Community lost interest in the approach
- Very unfortunate:
  - Already known from McCulloch and Pitts that any boolean function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

Deeper Backpropagation (1986)



- Multi-layer networks, trained by backpropagation, applied to cognitive tasks
- "Efficient applications of the chain rule based on dynamic programming began to appear in the 1960s and 1970s, mostly for control applications (Kelley, 1960; Bryson and Denham, 1961; Dreyfus, 1962; Bryson and Ho, 1969; Dreyfus, 1973) …. The idea was finally developed in practice after being independently rediscovered in different ways (LeCun, 1985; Parker, 1985; Rumelhart et al., 1986a). 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)

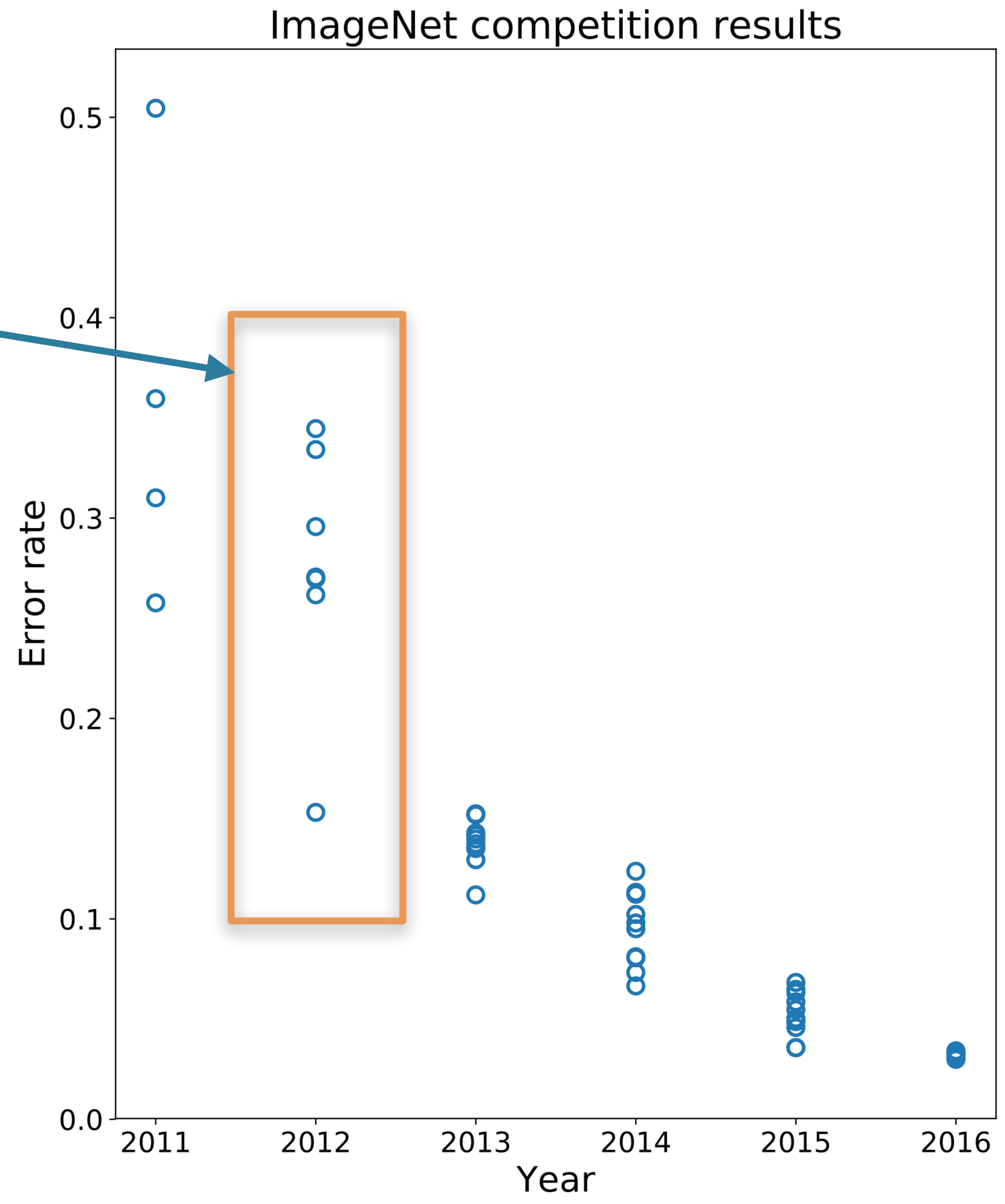


original website

- *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.

ImageNet (ILSVRC) results (2012)

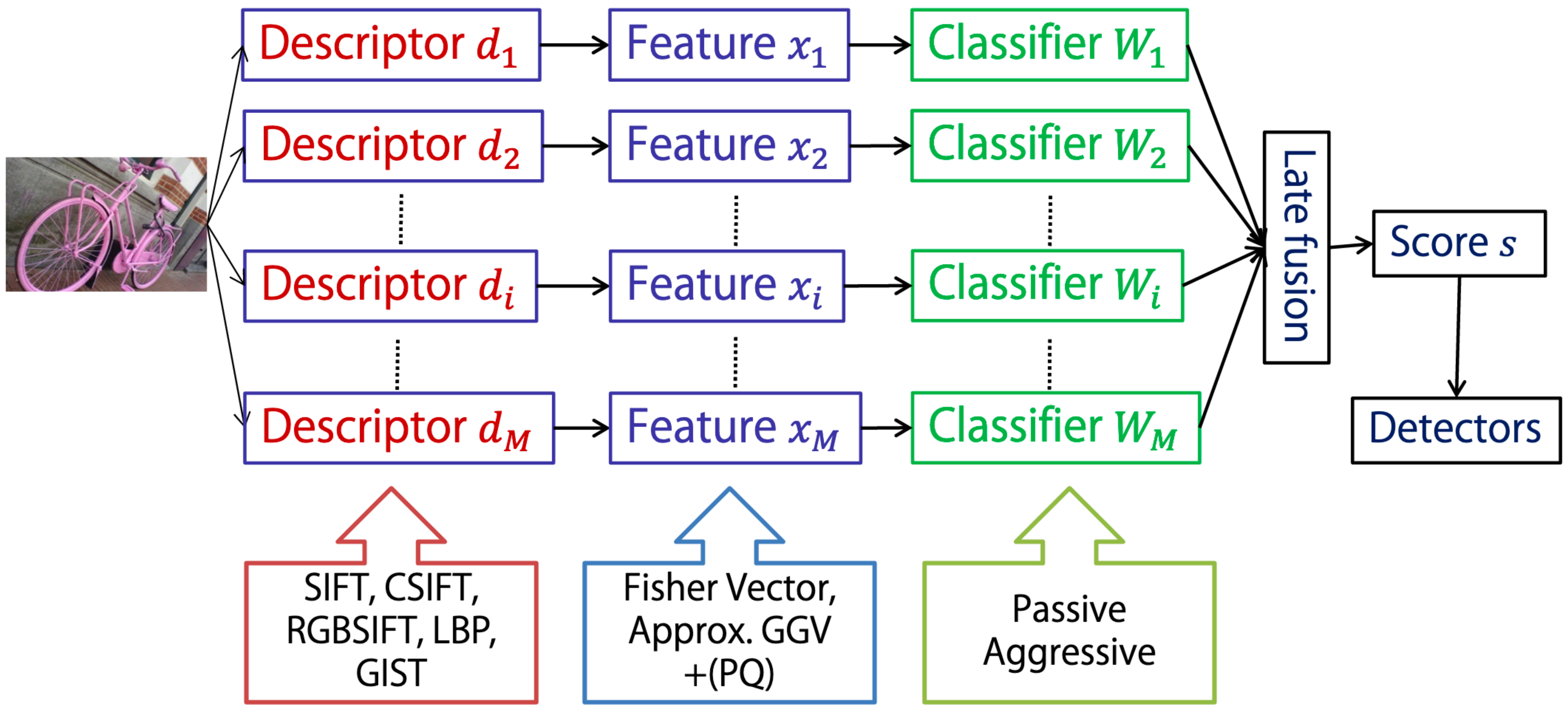
What happened in 2012?



source

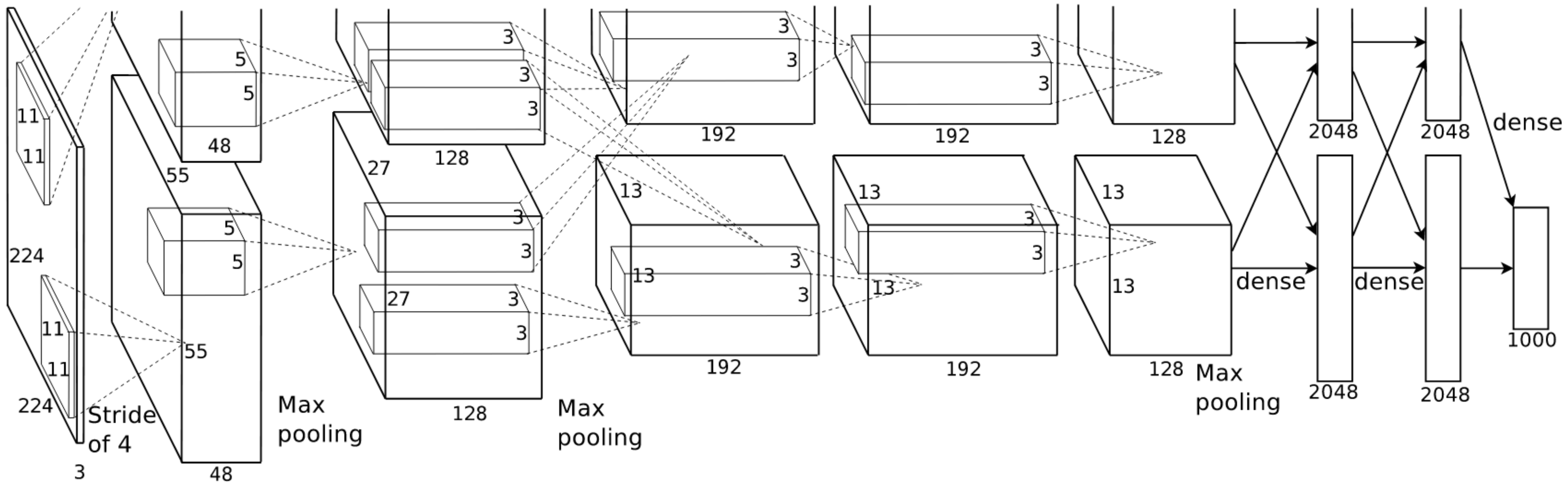
ILSVRC 2012: runner-up

Fisher based features + Multi class linear classifiers



source

ILSVRC 2012: winner



"AlexNet"

ImageNet Classification with Deep Convolutional Neural Networks

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

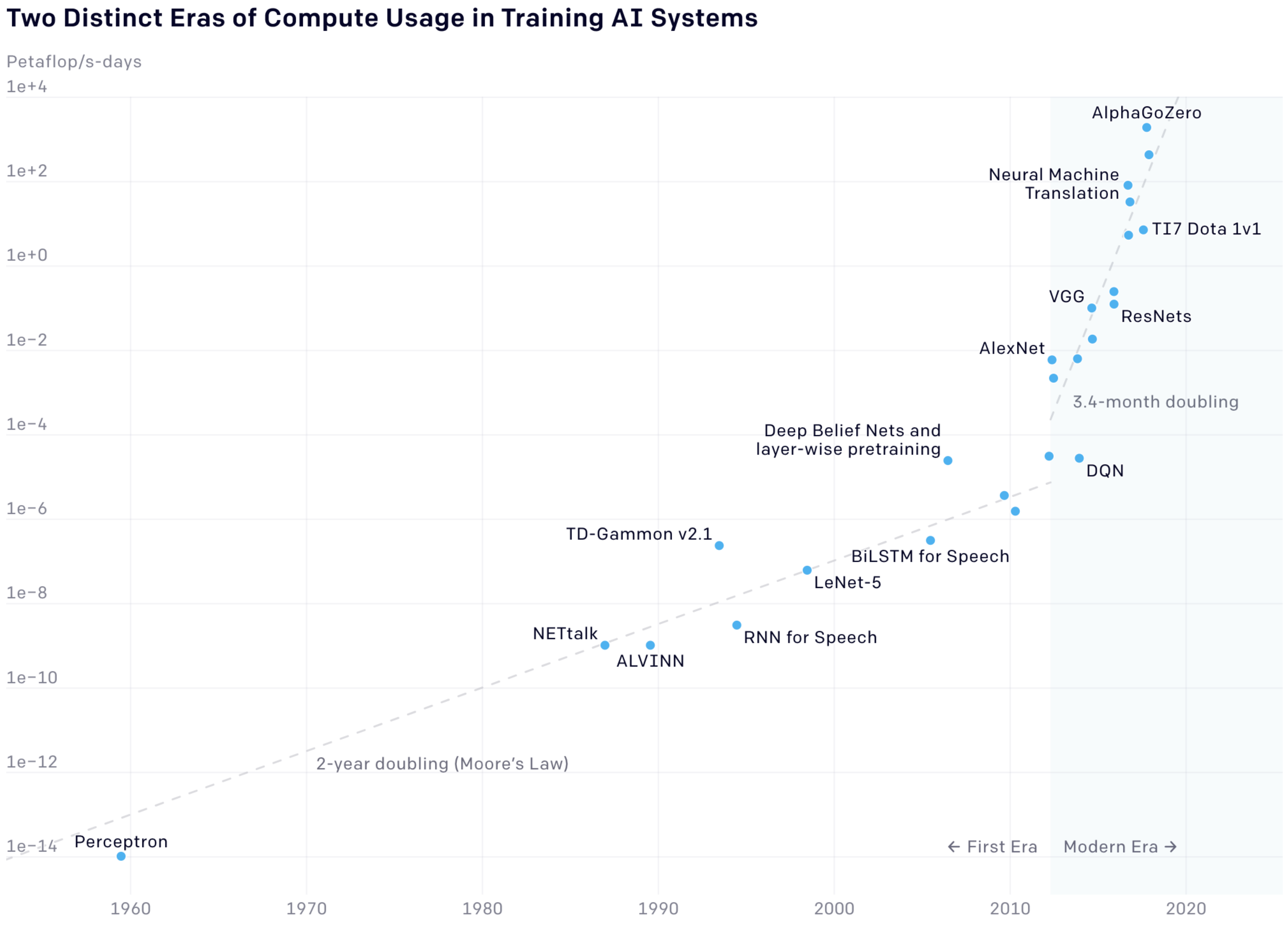
NeurIPS 2012 paper

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)

Compute in Deep Learning

log-scale!!



source

Caveat Emptor

- Some areas are an ‘arms race’ between e.g. Google, Facebook, OpenAI, MS, 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* ◊ Jesse Dodge*◊♣ Noah A. Smith◊♡ Oren Etzioni◊

◊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.

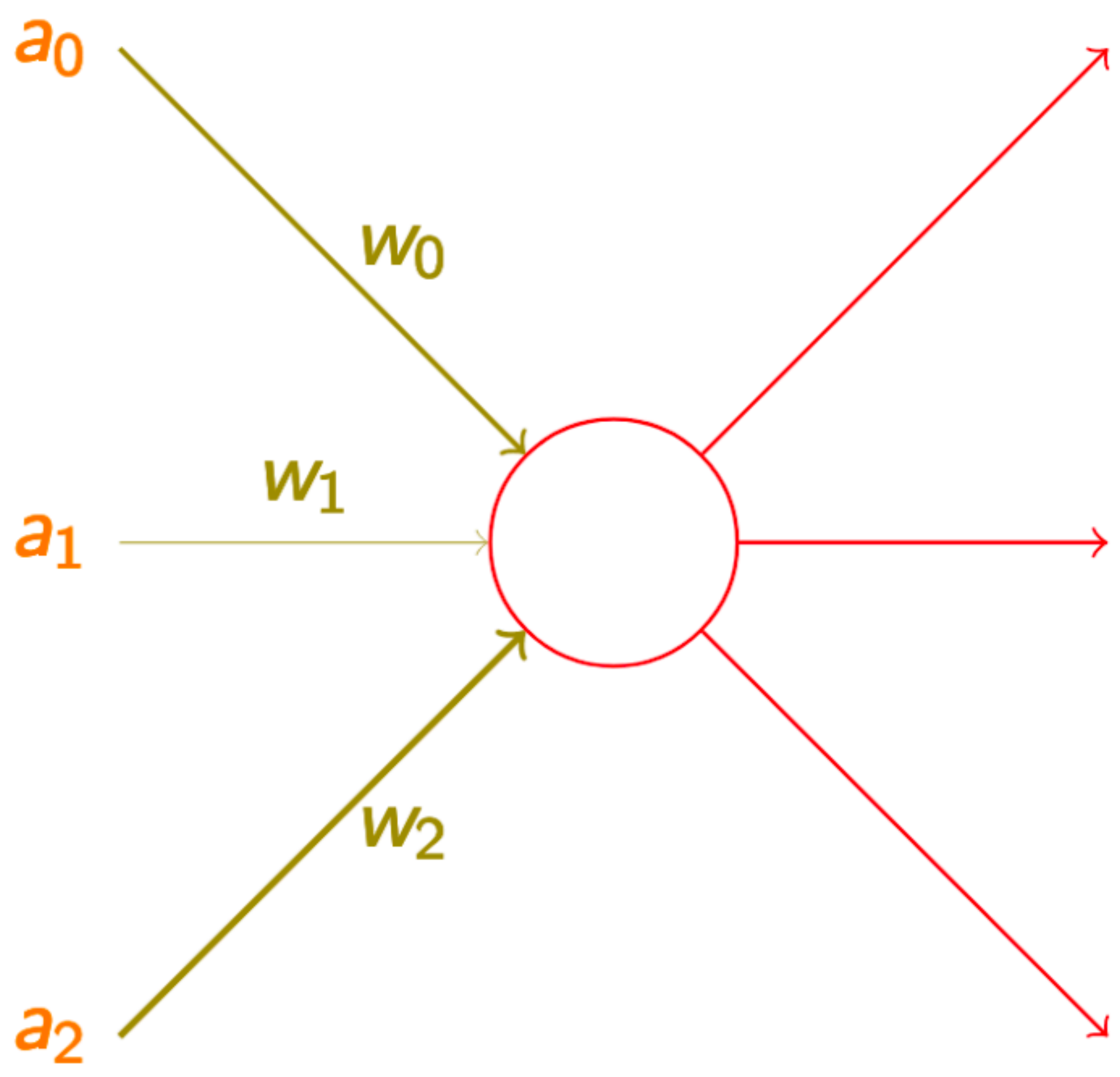
This position paper advocates a practical solution by making **efficiency** an evaluation criterion for research alongside accuracy and related measures. In addition, we propose reporting the financial cost or “price tag” of developing, training, and running models to provide baselines for the investigation 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_2e (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_2 emissions from training common NLP models, compared to familiar consumption.[1]

Computation: Basic Example

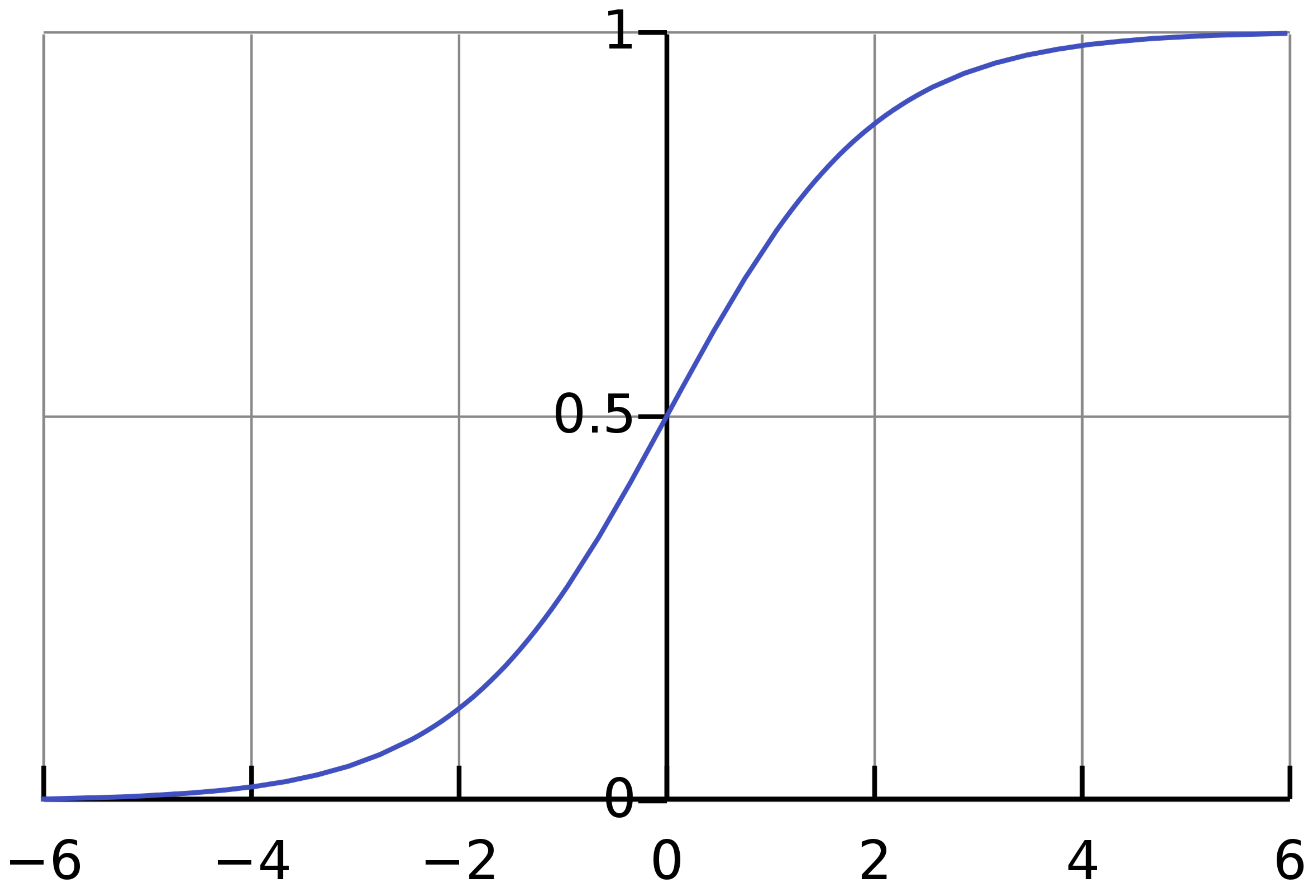
Artificial Neuron



$$a = f(a_0 \cdot w_0 + a_1 \cdot w_1 + a_2 \cdot w_2)$$

https://github.com/shanest/nn-tutorial

Activation Function: Sigmoid

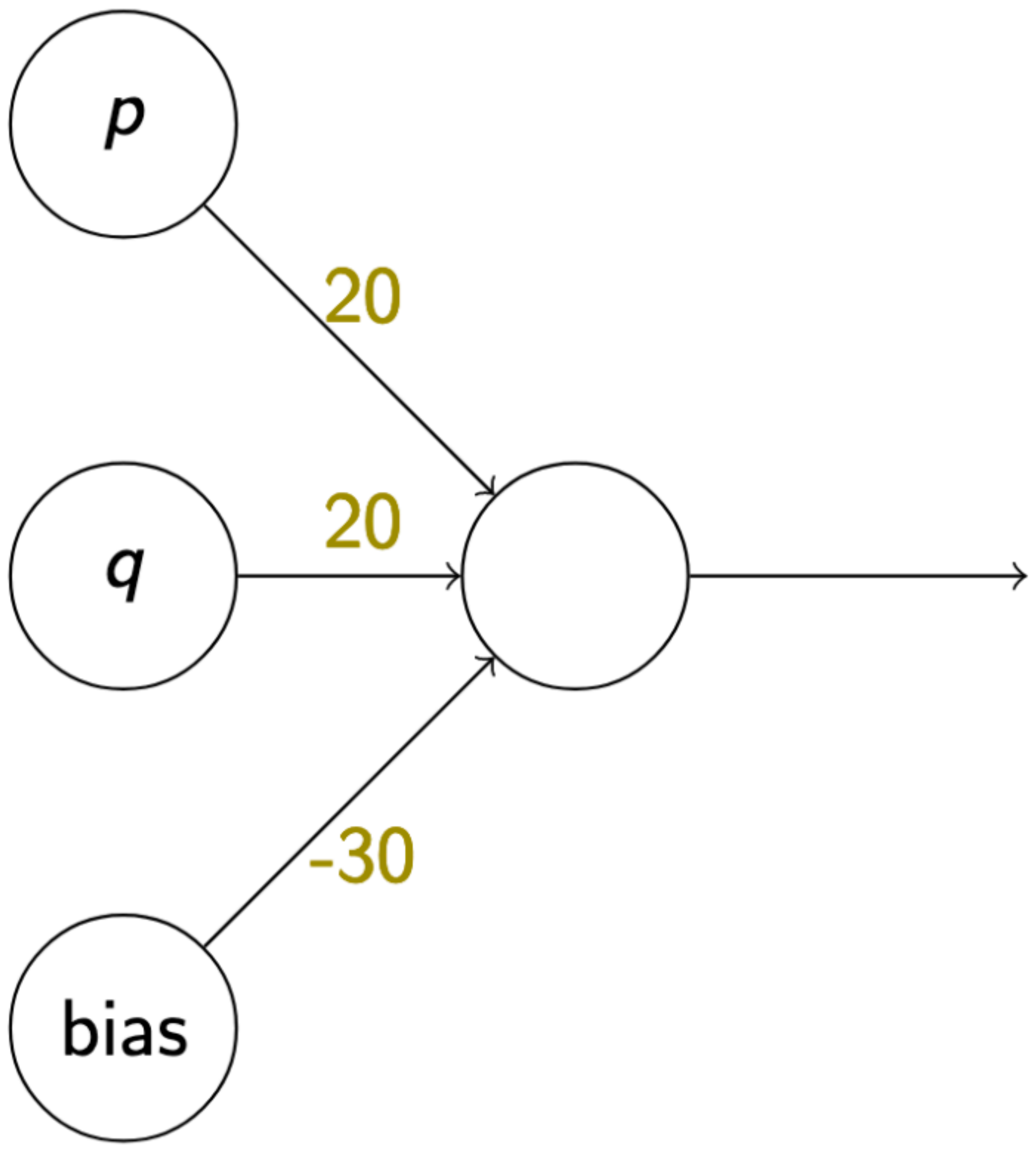


$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

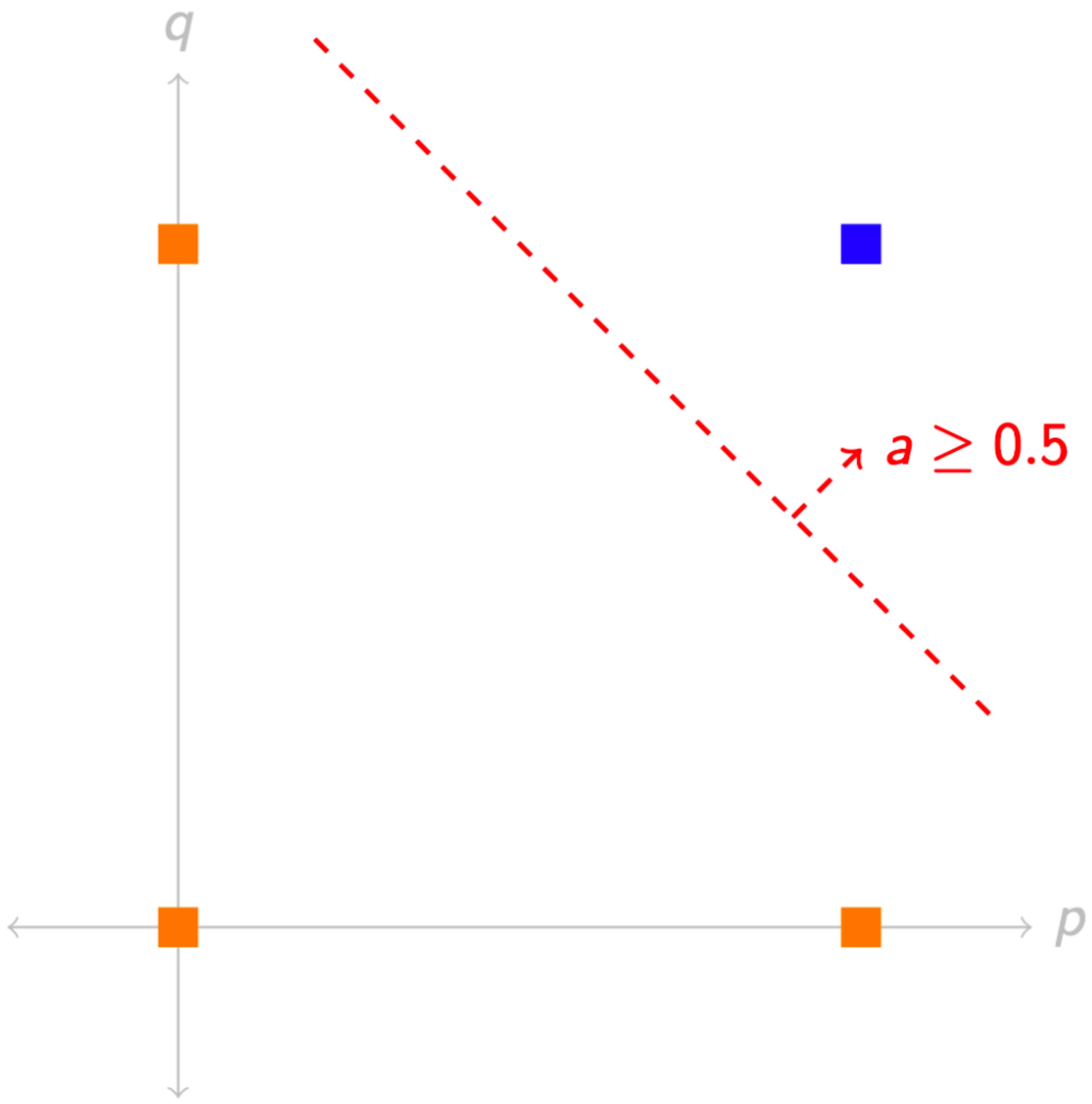
(more on this next time)

Computing a Boolean function

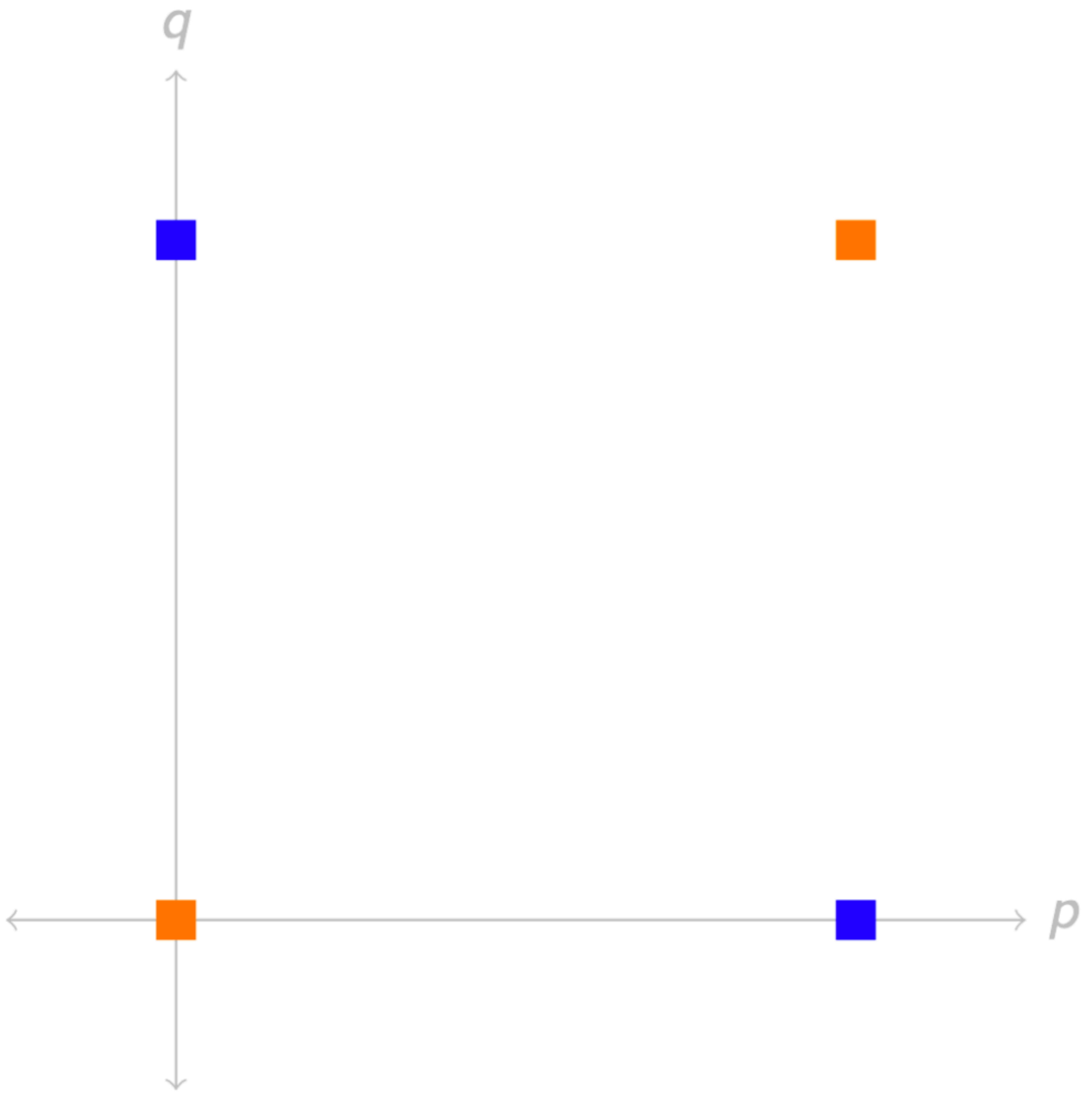
p	q	a
1	1	1
1	0	0
0	1	0
0	0	0



Computing ‘and’

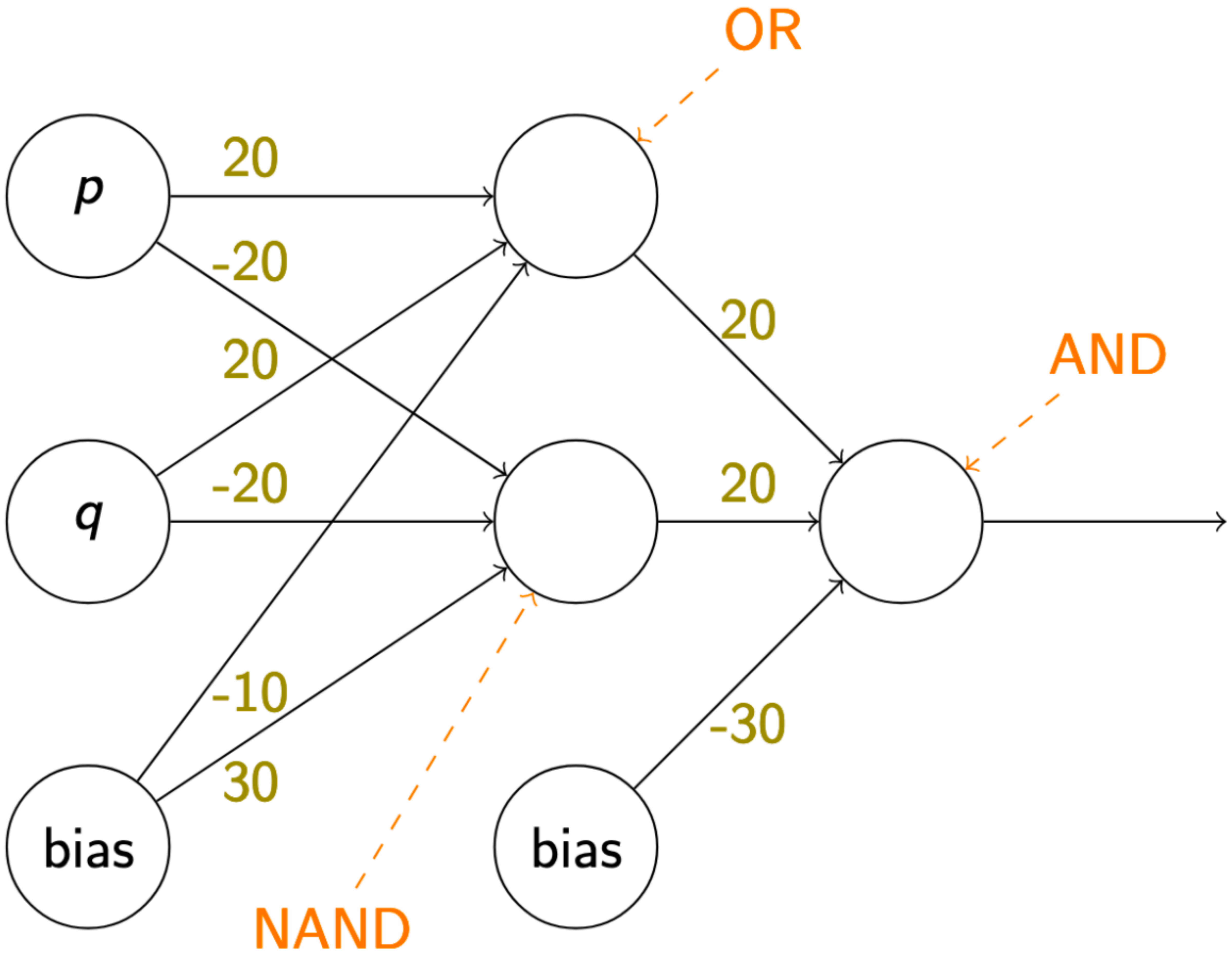


The XOR problem



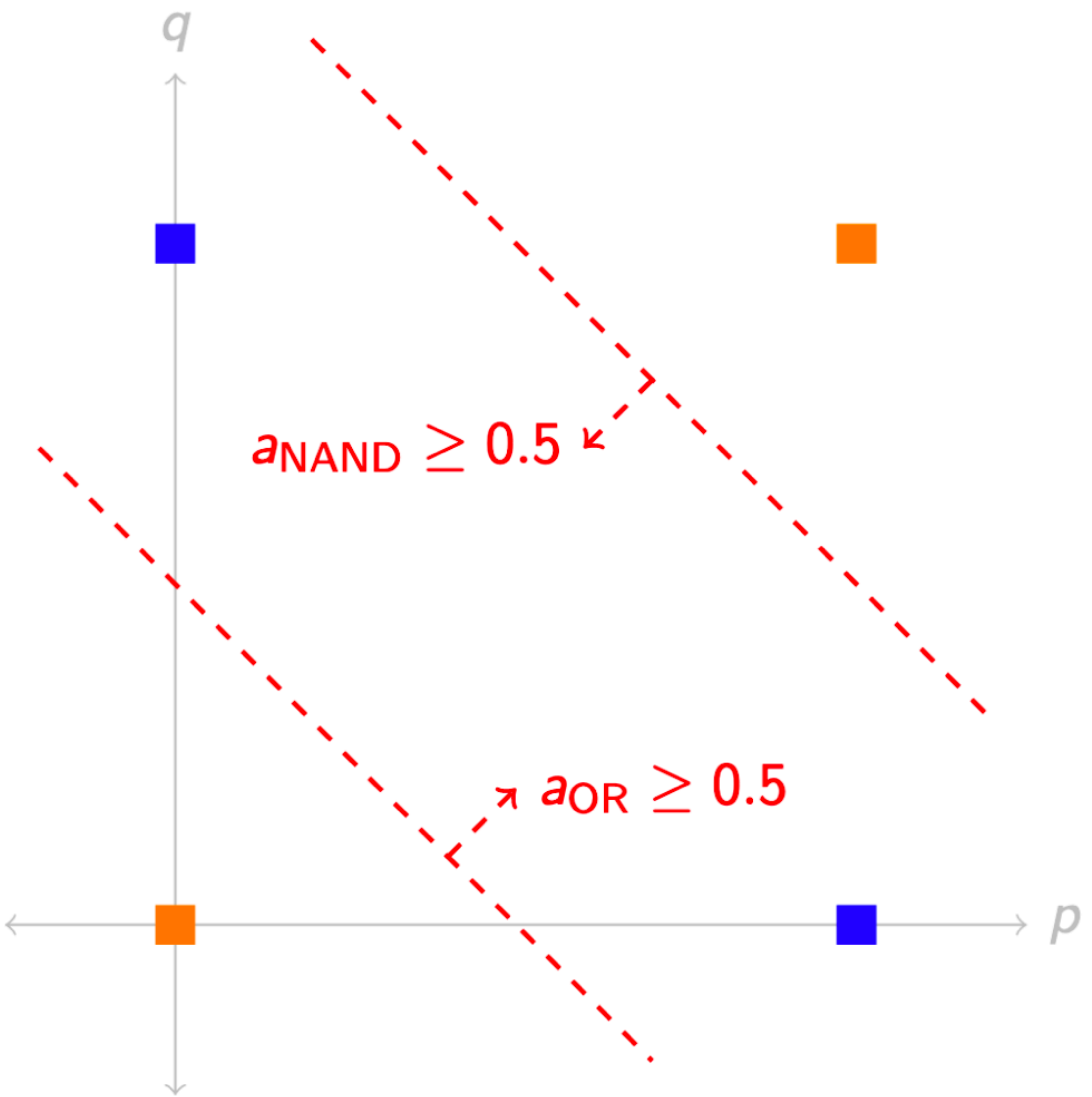
XOR is not linearly separable

Computing XOR



Exercise: show that
NAND behaves as described.

Computing XOR



Key Ideas

- Hidden layers compute high-level / abstract features of the input
  - Via training, will *learn which features* are helpful for a given task
  - Caveat: doesn't always learn much more than shallow features
- Doing so *increases the expressive power* of a neural network
  - Strictly more functions can be computed with hidden layers than without

Expressive Power

- Neural networks with *one* hidden layer are *universal function approximators*
- Let $f : [0,1]^m \to \mathbb{R}$ be continuous and $\epsilon > 0$. Then there is a one-hidden-layer neural network g with sigmoid activation such that $|f(\mathbf{x}) - g(\mathbf{x})| < \epsilon$ for all $\mathbf{x} \in [0,1]^m$.
- Generalizations (diff activation functions, less bounded, etc.) exist.
- But:
  - Size of the hidden layer is *exponential* in *m*
  - How does one *find*/learn such a good approximation?
- Nice walkthrough: http://neuralnetworksanddeeplearning.com/chap4.html

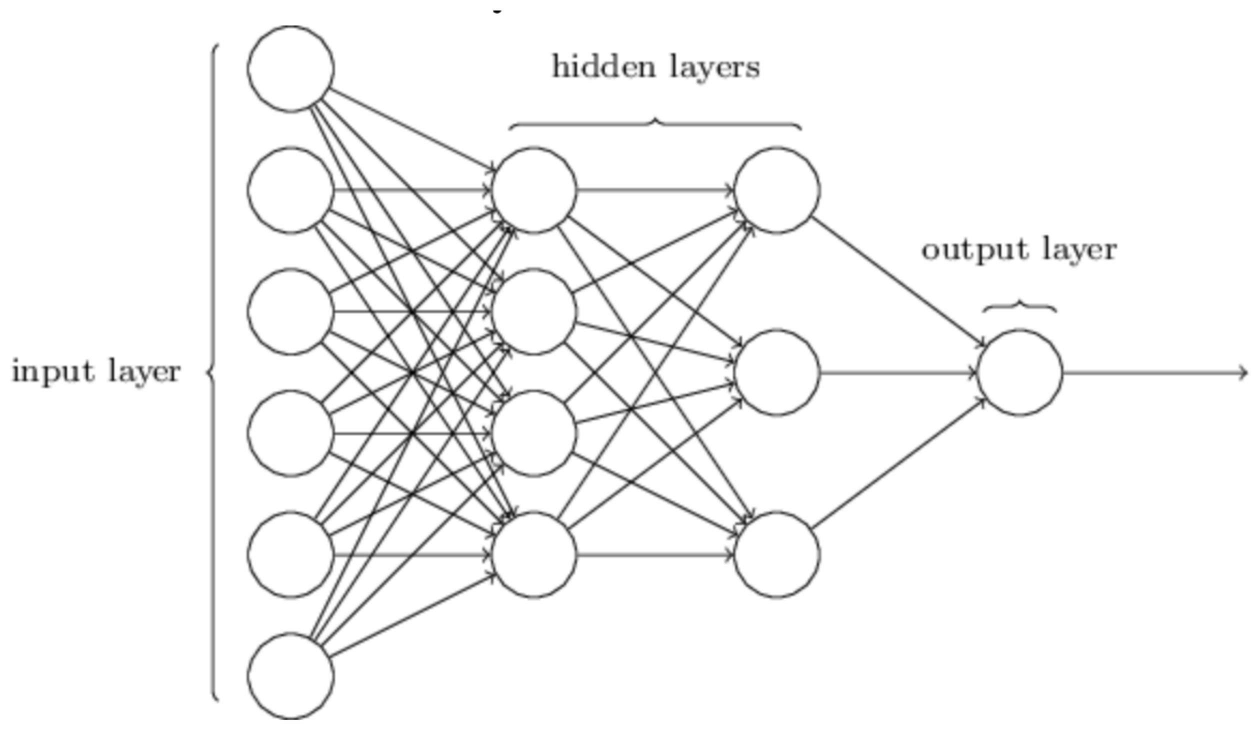
Landscape

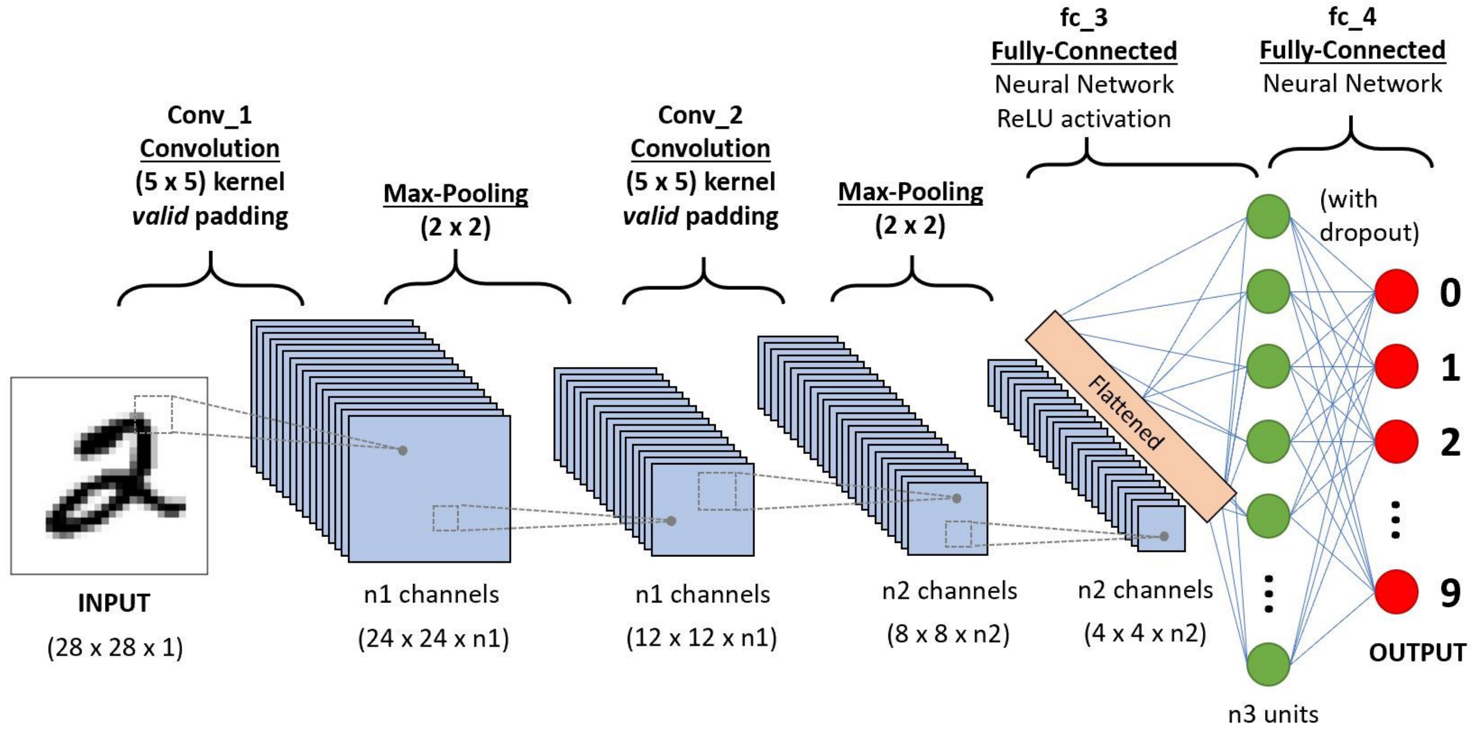
Next steps

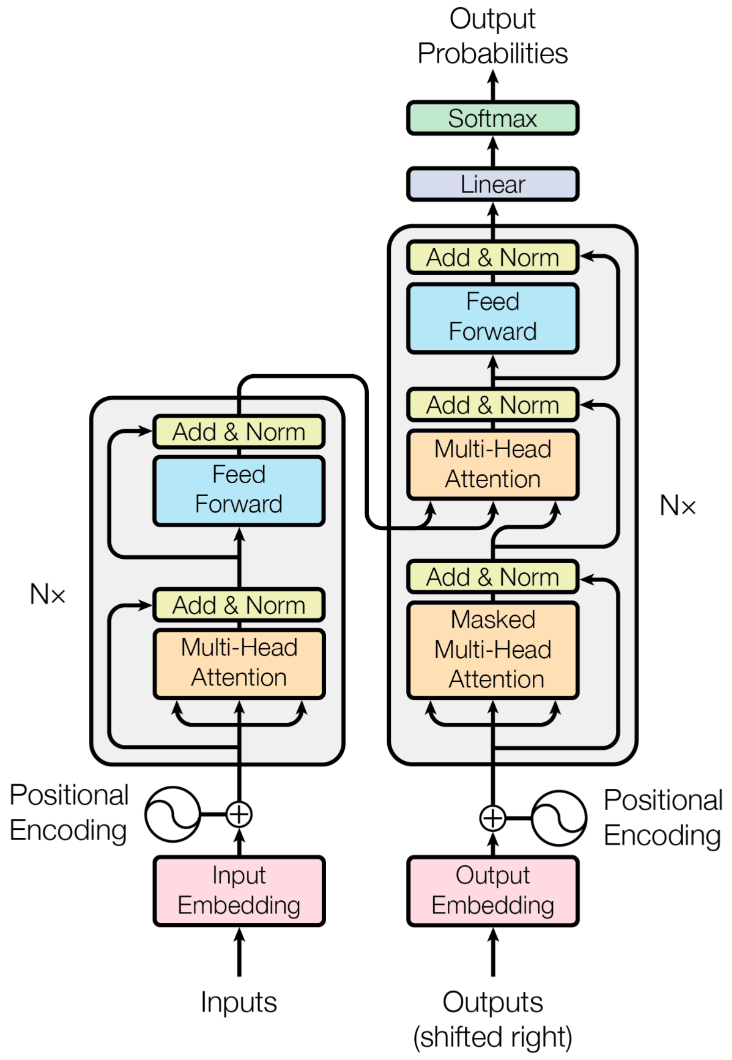
- More detail about computation, how to build and implement networks
- Where do the weights and biases come from?
  - (Stochastic) gradient descent
  - Backpropagation for gradients
- Various hyper-parameters around both of those
- NLP “specific” topics:
  - Sequence models
  - Pre-training

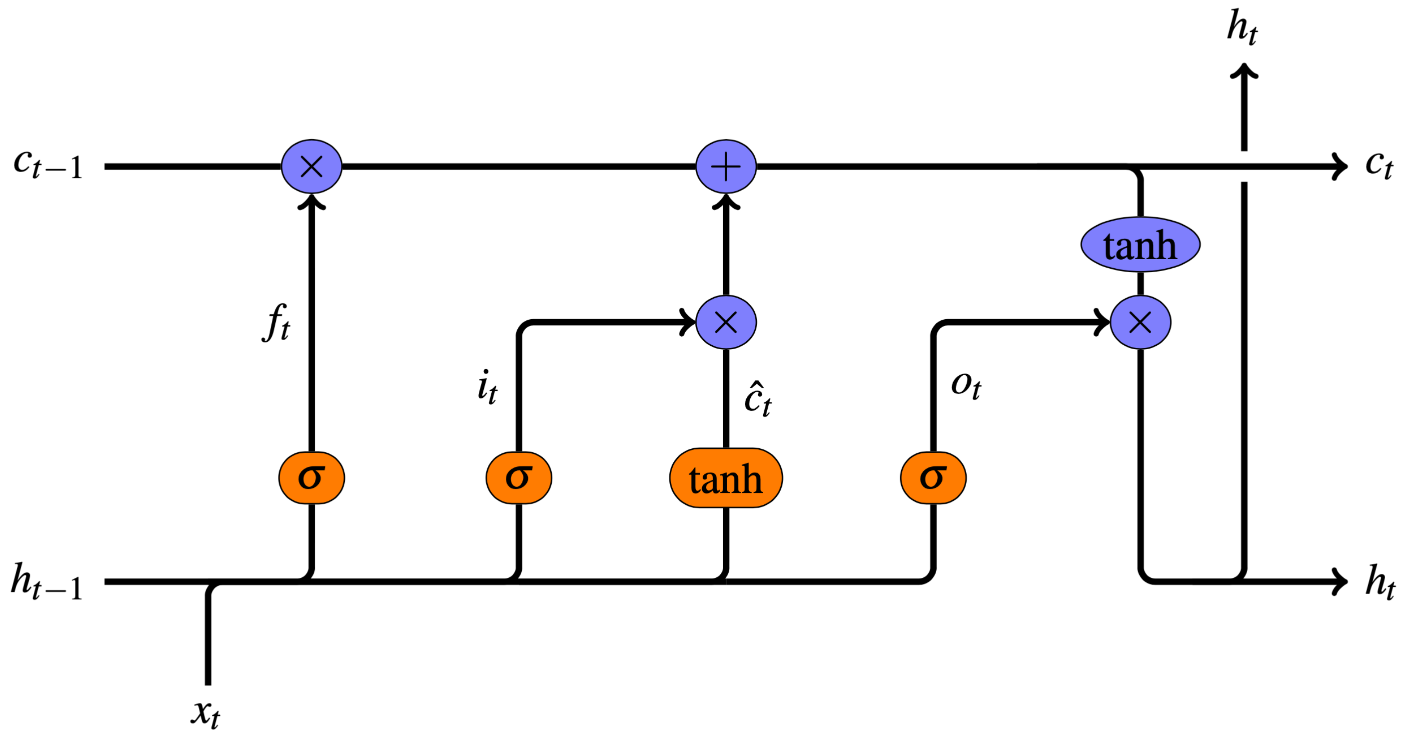
Broad architecture types

- Feed-forward (multi-layer perceptron)
  - Today and next time
- Convolutional (mainly for images, but also text applications)
- Recurrent (sequences; LSTM the most common)
- Transformers









Resources

- 3blue1brown videos: useful introduction, well animated
- Neural Networks and Deep Learning free e-book
  - A bit heavy on the notation, but useful
- Deep Learning book (free online): very solid, presupposes some mathematical maturity
- Various other course materials (e.g. CS231n and CS224n from Stanford)
- Blog posts
  - NB: hit or miss! Some are *amazing*, some are….not

Libraries

- General libraries:
  - PyTorch
  - TensorFlow
- Received wisdom: PyTorch the best for research; TF slightly better for deployment.
  - But, both are converging on the same API, just from different ends
  - I have a strong preference for PyTorch; it's also a more consistent API
- NLP specific: AllenNLP, fairseq, HuggingFace Transformers