Brief history

- Inspired by 1940s understanding of neurons in animal brains (Mccullough-Pitt)
- Basic training algorithm (backpropagation) discovered in 1986 (Rumelhart, Hinton, Williams).
- Convolutional nets + modern training from late 1980s.

Published: 09 October 1986

Learning representations by back-propagating errors

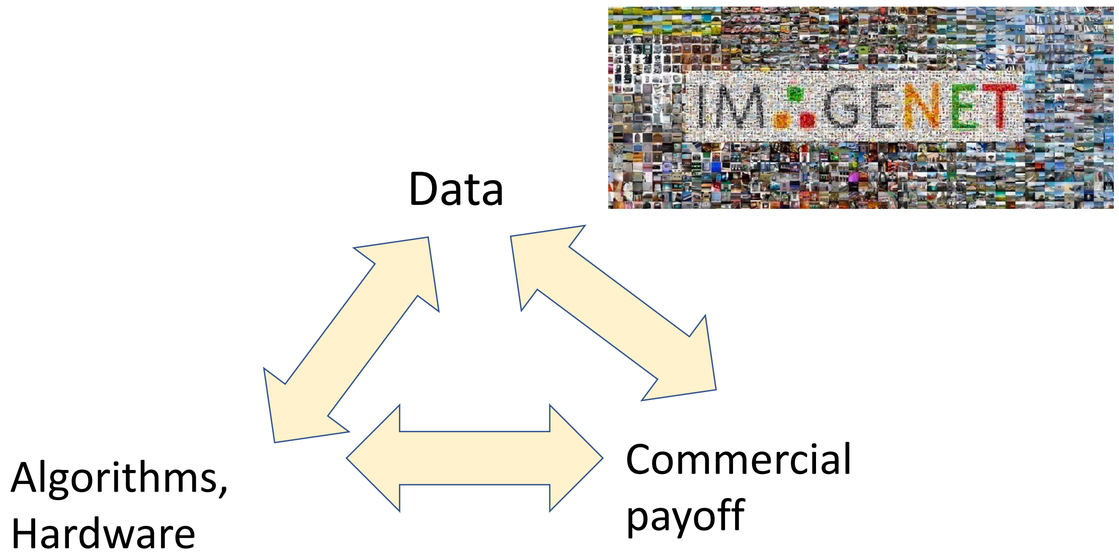
David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams

Nature **323**, 533–536 (1986) | Cite this article

Brief history

- Out of fashion for a decade; came back strong with around 2012. Now dominant in AI (computer vision, NLP, robotics, etc).

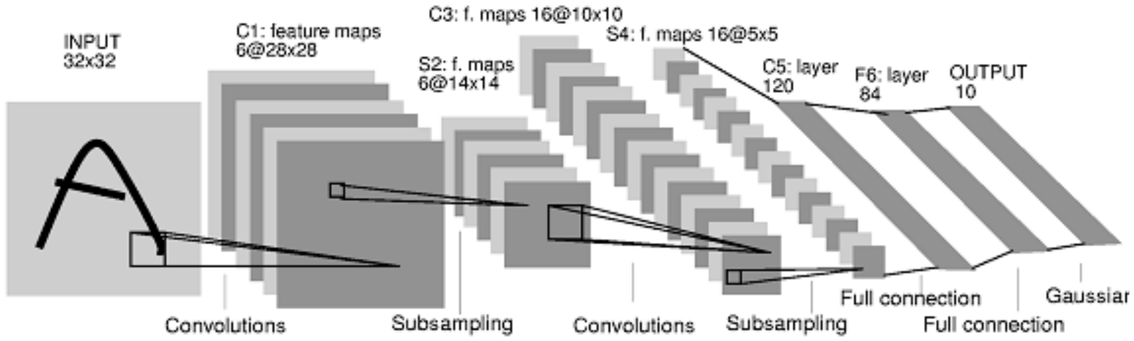
Why popular now?



hardware

1998

LeCun et al.



# of transistors

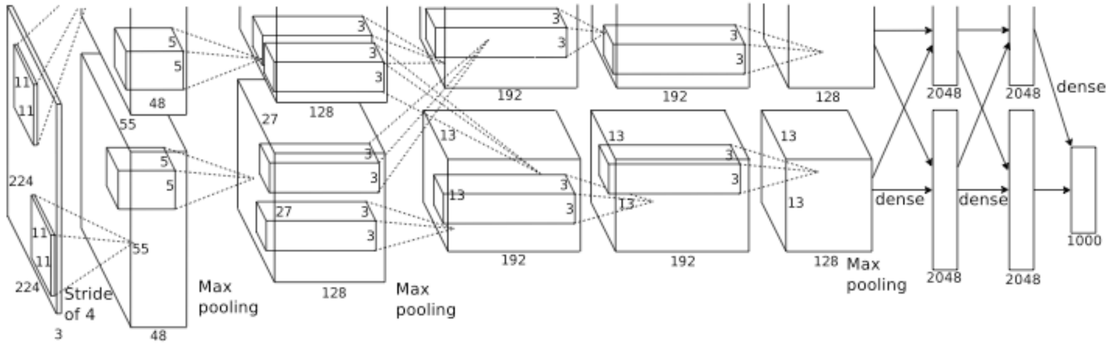
10^6

# of pixels used in training

NIST 10^7

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

IMAGENET 10^{14}

2022

GPUs have 8×10^{10} transistors.

Adapted from: Fei-Fei Li

Commercial payoff

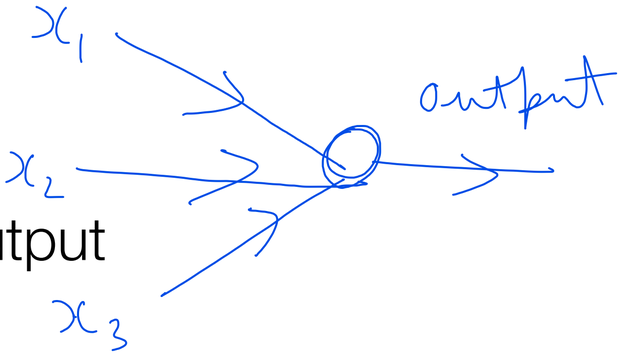
- Works well enough to monetize, across a range of applications
- Open-source tools enable democratization of access

Tensorflow

Pytorch

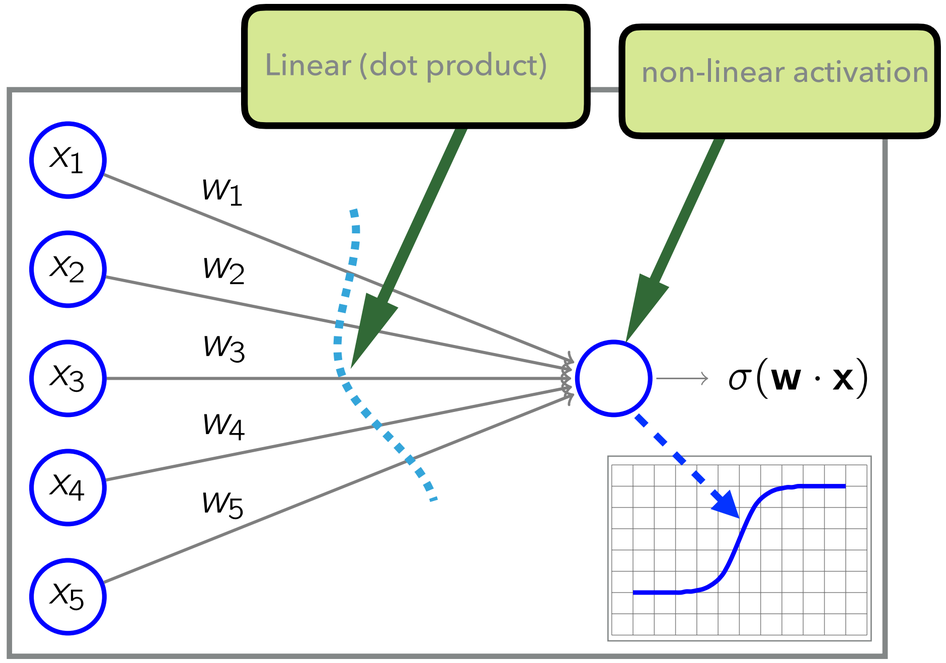
Neural Network Basics

An artificial neuron



- A neuron is a computational unit that has scalar inputs and an output
- Each input has an associated weight.
- The neuron multiples each input by its weight, sums them, applied a nonlinear activation function to the result, and passes it to its output.

$w_1 x_1 + w_2 x_2 + w_3 x_3$



$$\sigma(z) = 1/(1 + e^{-z})$$

Input: x_1, x_2, x_3, x_4, x_5

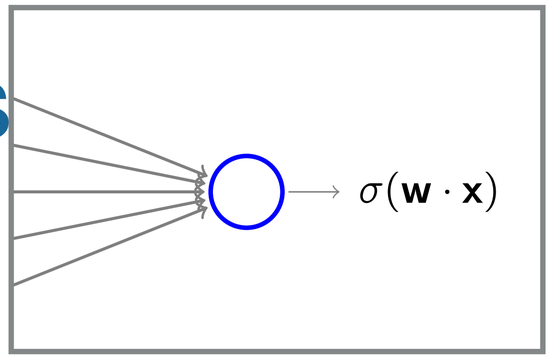
weights: w_1, w_2, w_3, w_4, w_5

output:

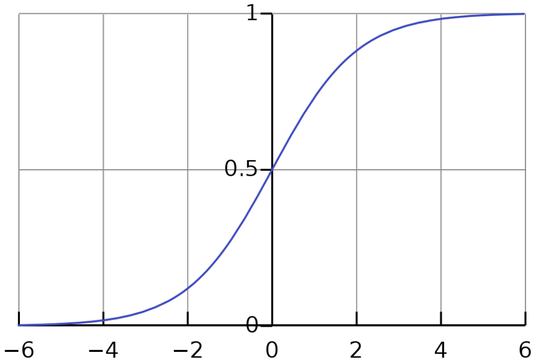
$$y = \sigma\left(\sum_{j=1}^{5} w_j x_j\right) \text{ or } y = \sigma(\mathbf{w} \cdot \mathbf{x})$$

A neuron can be a logistic regression unit!

Popular Activation functions

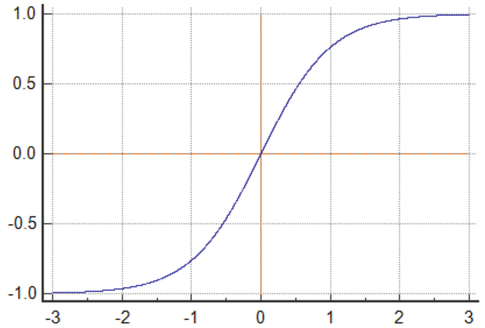


sigmoid $$\sigma(z) = \frac{1}{1 + e^{-z}}$$

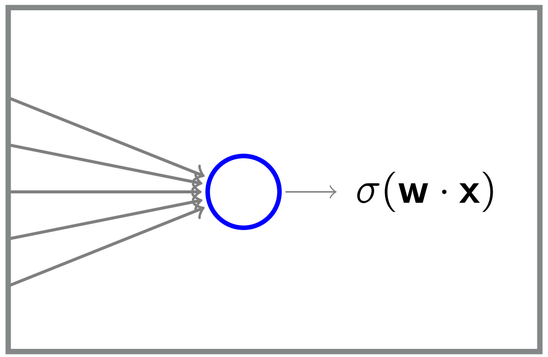


tanh: $$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$(\tanh(z) = 2\sigma(2z) - 1)$$



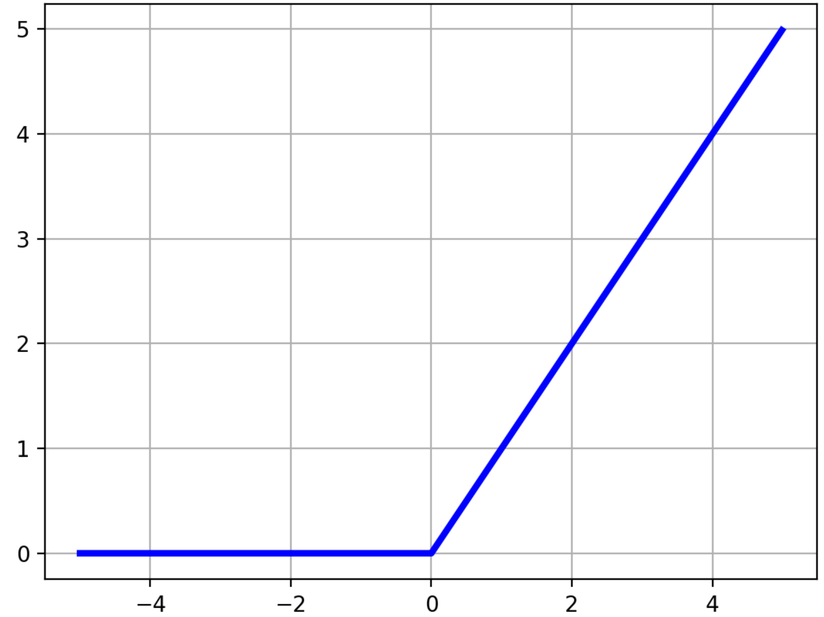
Activation function: ReLU



A popular choice: **Re**ctified **L**inear **U**nit (ReLU)

$$\text{ReLU}(z) = \max\{z, 0\} = \begin{cases} 0 & z \leq 0 \\ z & z > 0 \end{cases}$$

Shorthand: $\text{ReLU}(z) = [z]_+$

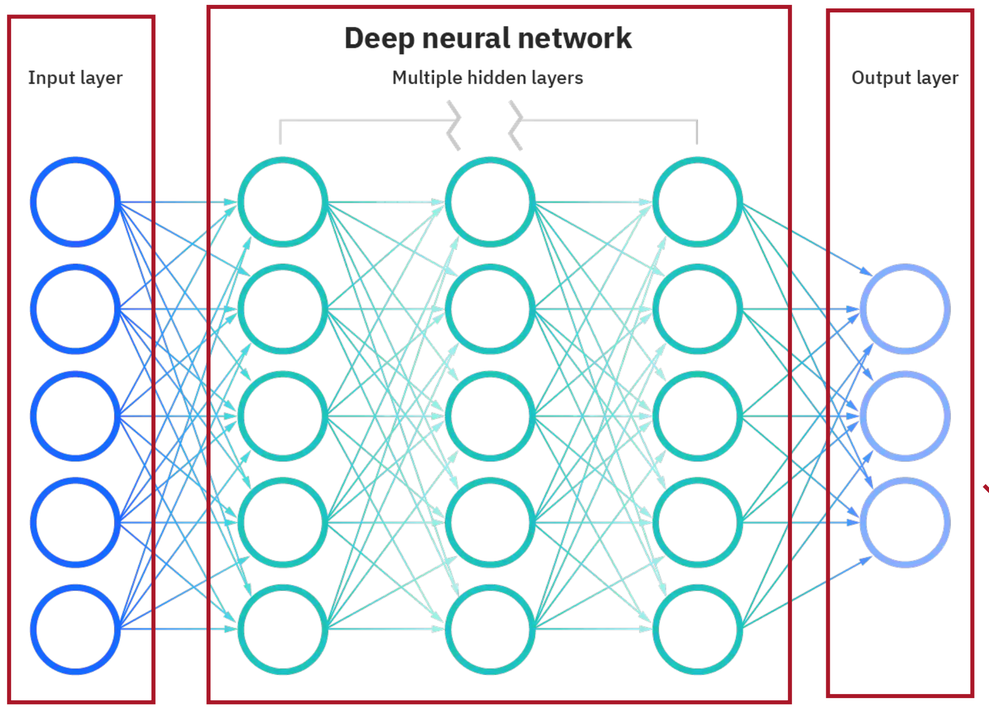


Q: What is the advantage of ReLU?

1) avoid gradient vanishing; 2) cheaper to compute

Neural networks (high level)

- The artificial neurons are connected to each other, forming a network
- The output of a neuron may feed into the inputs of other neurons

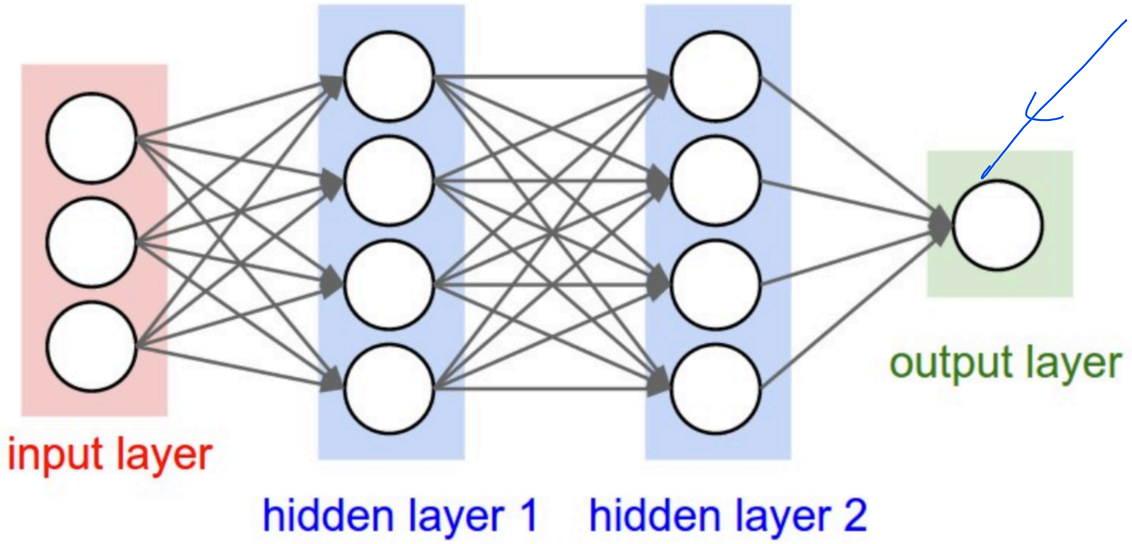


multiple outputs (next time)

The intermediate layers are called hidden layers

Feedforward neural networks

- A feedforward network is a multilayer feedforward network:
  - The units are connected with **no cycles**
  - The outputs from units in each layer are passed to units in the next higher layer
  - No outputs are passed back to lower layers



Fully-connected layers:

All the units from one layer are fully connected to every unit of the next layer.

Note: each edge has an associated weight