A gentle introduction to Deep Learning

Part 1: Computer Vision

Alina Miron

alina.miron@brunel.ac.uk

9th of November 2021

In today’s lecture we’ll discuss:

- What is deep learning
- Recap Neural Networks
- Image classification task
- Convolutional Neural Networks
  - Convolution
  - Pooling
  - Dropout
  - RELU activation function
- Basic deep learning software libraries
- Deep learning advantages & disadvantages

A short history of deep learning

A short history of deep learning

The term of deep learning is introduced

1986

LEARNING WHILE SEARCHING IN CONSTRAINT-SATISFACTION-PROBLEMS*

Rina Dechter

Artificial Intelligence Center
Hughes Aircraft Company, Calabasas, CA 91302
and
Cognitive Systems Laboratory, Computer Science Department
University of California, Los Angeles, CA 90024

ABSTRACT

The popular use of backtracking as a control strategy for theorem proving in PROLOG and in Truth-Maintenance-Systems (TMS) led to increased interest in various schemes for enhancing the efficiency of backtrack search. Researchers have referred to these enhancement schemes by the names

2. **Look-back schemes:** affecting the decision of where and how to go in case of a a dead-end situation. Look-back schemes are centered around two fundamental ideas:

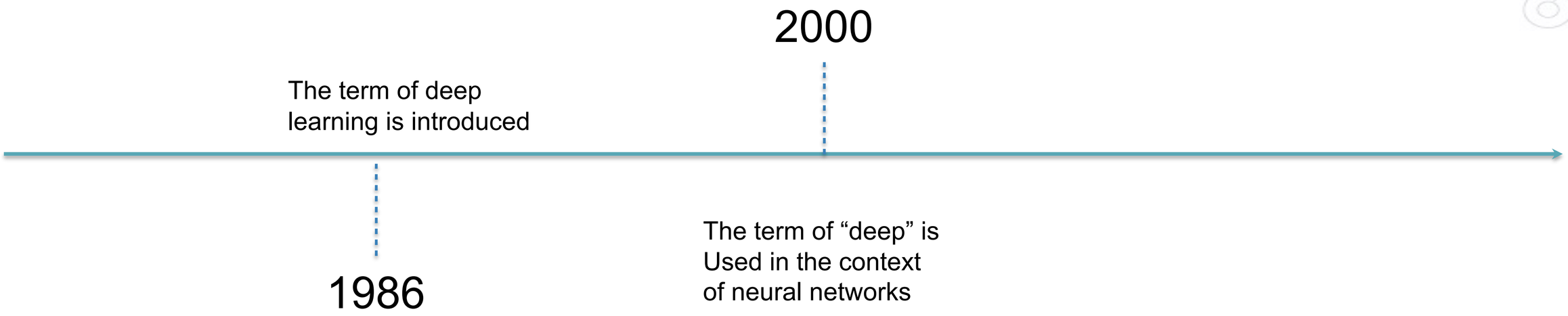
   a. **Go-back to source of failure:** an attempt is made to detect and change previous decisions

Discovering all minimal conflict-sets amounts to acquiring all the possible information out of a dead-end. Yet, such **deep learning** may require considerable amount of work. While the number of minimal conflict-sets is less then 2^r, where r is the cardinality of the Conf-set, we can envision a worst case where all subsets of Conf(S,X_i) having $\frac{r}{2}$ elements are in conflict with X_i. The number of minimal conflict-sets should then satisfy

$$\#\text{min-conflict-sets} = \left[\begin{matrix} r \\ \frac{r}{2} \end{matrix}\right] \cong 2^r \,, \qquad (2)$$

which is still exponential in the size of the Conf-set. If the size of this Conf-set is small it may still be reasonable to recognize all minimal conflict-sets.

A short history of deep learning



A short history of deep learning

2000

The term of deep
learning is introduced

1986

The term of “deep” is
Used in the context
of neural networks

Co-Evolving Recurrent Neurons Learn Deep Memory POMDPs

Faustino J. Gomez[1]
tino@idsia.ch

Jürgen Schmidhuber[1,2]
juergen@idsia.ch

[1] IDSIA, Galleria 2, 6928 Lugano, Switzerland

[2] TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany

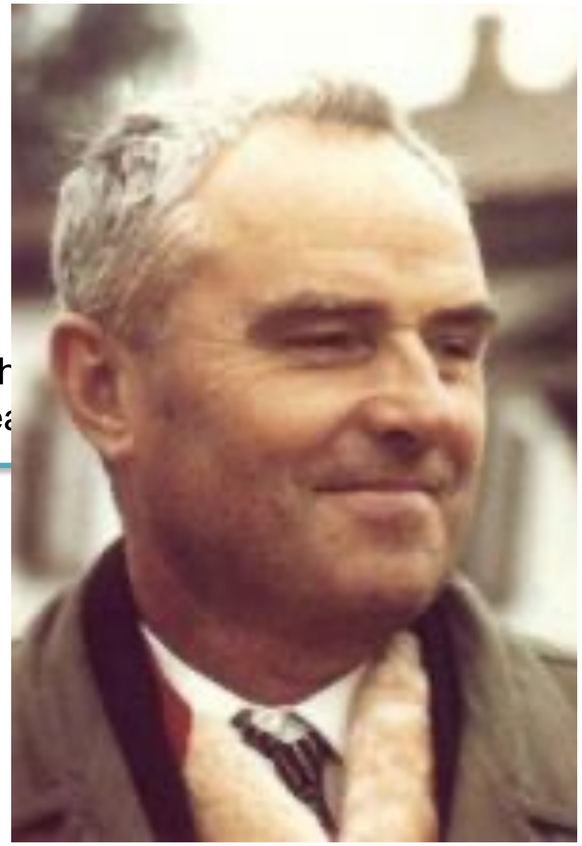
ABSTRACT

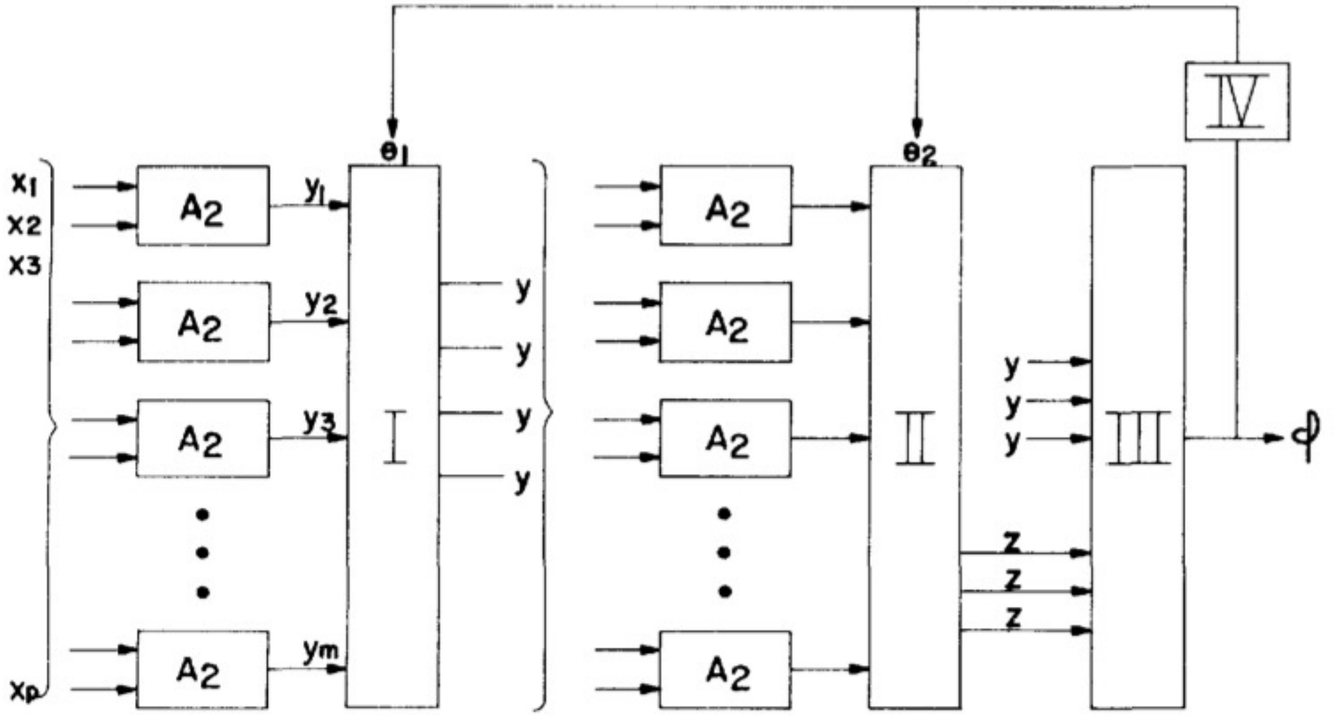
Recurrent neural networks are theoretically capable of learning complex temporal sequences, but training them through gradient-descent is too slow and unstable for practical use in reinforcement learning environments. Neuroevolution, the evolution of artificial neural networks using genetic algorithms, can potentially solve real-world reinforcement learning tasks that require deep use of memory, i.e. memory spanning hundreds or thousands of inputs, by searching the space of recurrent neural networks directly. In this paper, we introduce a new neuroevolution algorithm called Hierarchical Enforced SubPopulations that simultaneously evolves networks at two levels of granularity: full networks and network components or *neurons*. We demonstrate the method previous network inputs. This ability to act based upon the recollection of past events is essential for truly complex behavior. Unfortunately, training RNNs with standard gradient-descent methods such as Real-Time Recurrent Learning (RTRL; [11, 14]) or Back Propagation Through Time (BPTT; [12]) is not possible when actions depend on inputs from more than as few as 10 time-steps in the past. The error signal used to adjust synaptic weights either vanishes or explodes as it is propagated back through time, making learning prohibitively slow or preventing it altogether [5, 3]. For this reason, using conventional RNN architectures to approximate a value-function or policy for reinforcement learning (i.e. Q-learning, SARSA) is not practical when the environment exhibits long temporal dependencies.

A short history of deep learning

Ivakhnenko: “working algorithm for supervised, deep, feedforward, multilayer perceptrons”

1965



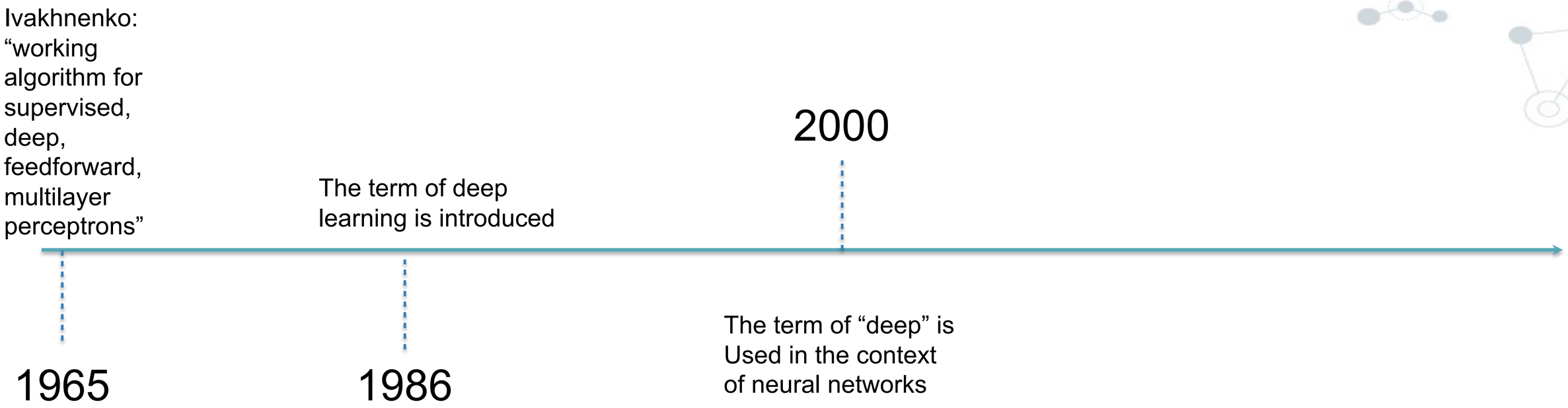


Algorithm of group method of data handling (GMDH) using second-degree polynomials. I—first threshold self-selection; II—second threshold self-selection; III—selection from all solutions; IV—threshold optimization.

https://kpi.ua/en/ivakhnenko

https://www.gmdh.net/articles/history/polynomial.pdf

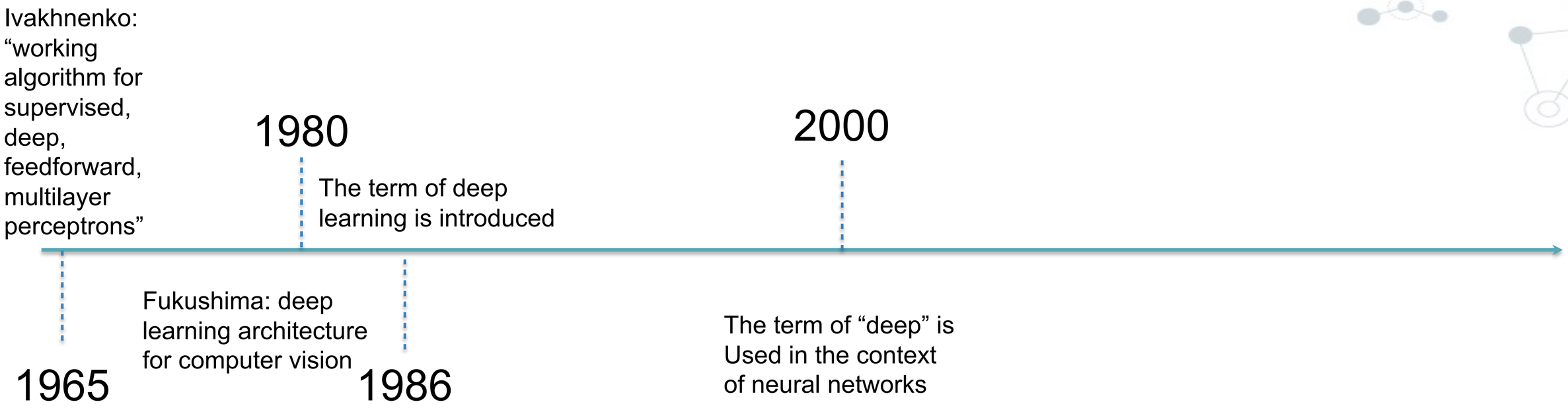
A short history of deep learning



https://kpi.ua/en/ivakhnenko

https://www.gmdh.net/articles/history/polynomial.pdf

A short history of deep learning



Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." *Competition and cooperation in neural nets*. Springer, Berlin, Heidelberg, 1982. 267-285.

A short history of deep learning

Ivakhnenko: “working algorithm for supervised, deep, feedforward, multilayer perceptrons”

1965

1980

The term of deep learning is introduced

Fukushima: deep learning architecture for computer vision

1986

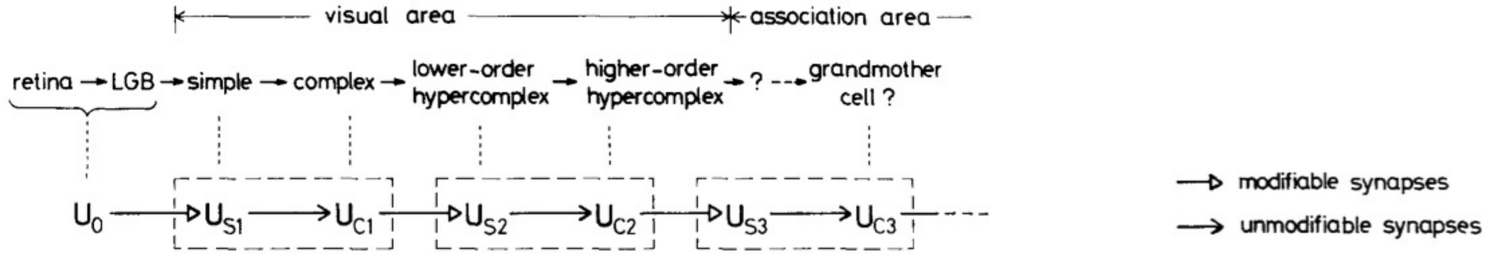


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

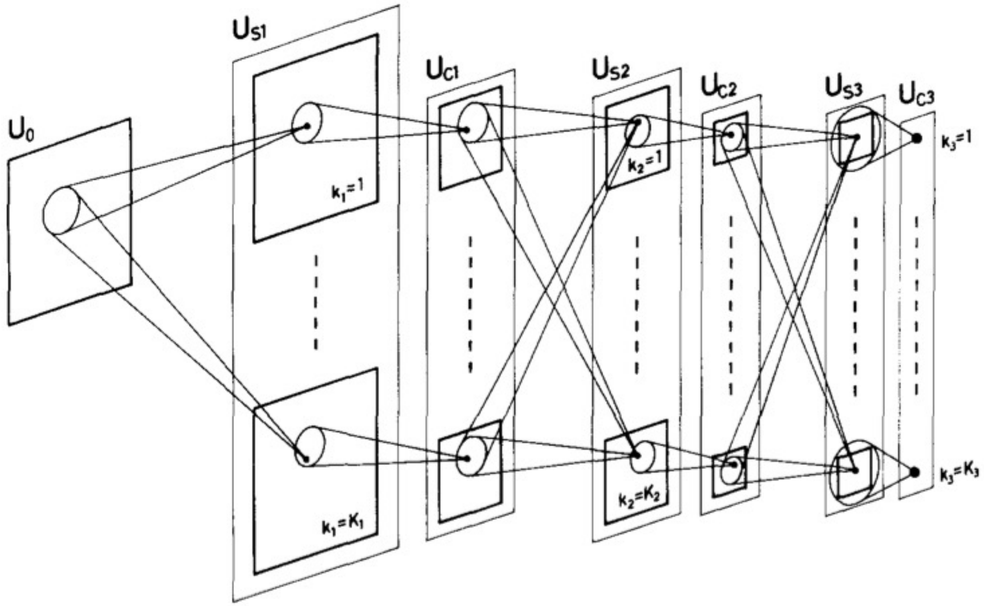
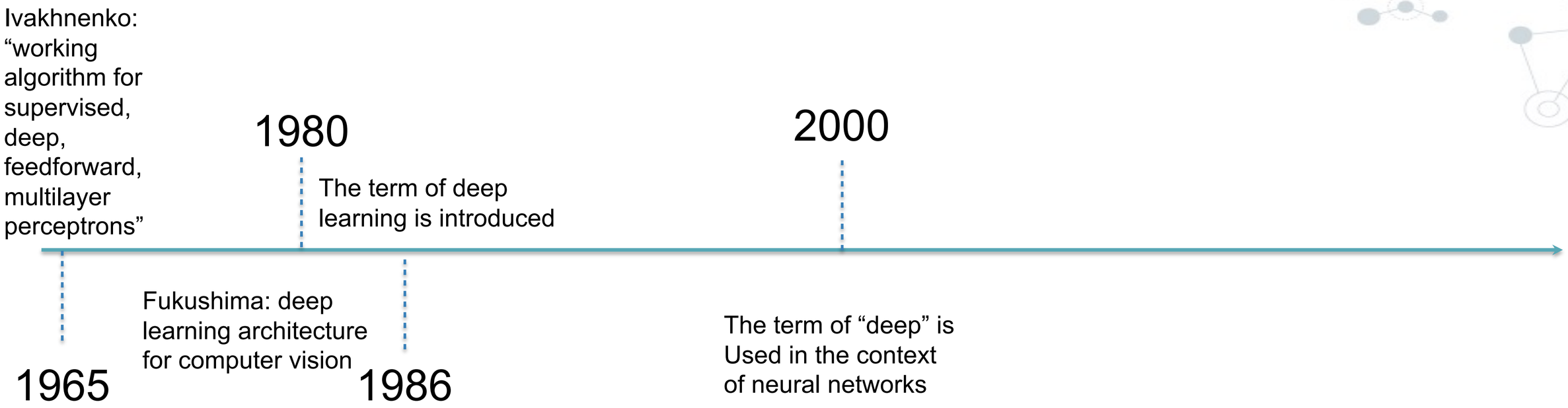


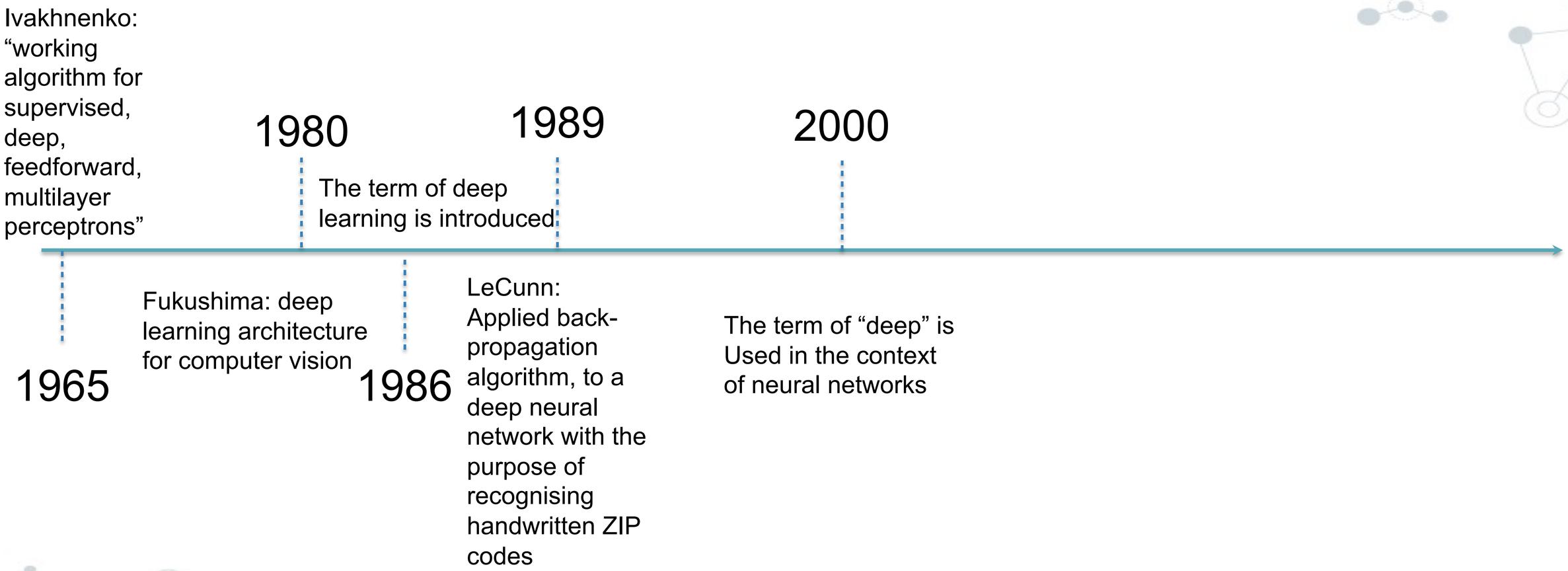
Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." *Competition and cooperation in neural nets*. Springer, Berlin, Heidelberg, 1982. 267-285.

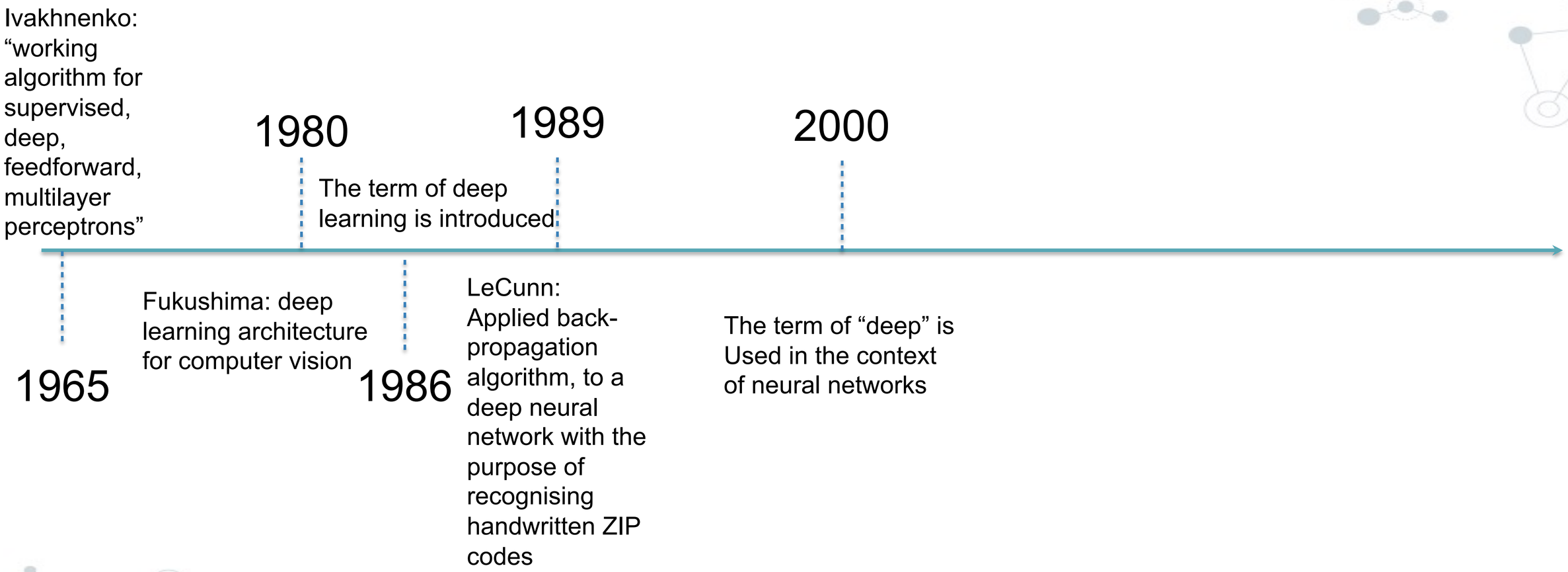
A short history of deep learning



A short history of deep learning



A short history of deep learning



LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), pp.541-551.

A short history of deep learning

1965

Ivakhnenko: “working algorithm for supervised, deep, feedforward, multilayer perceptrons”

1980

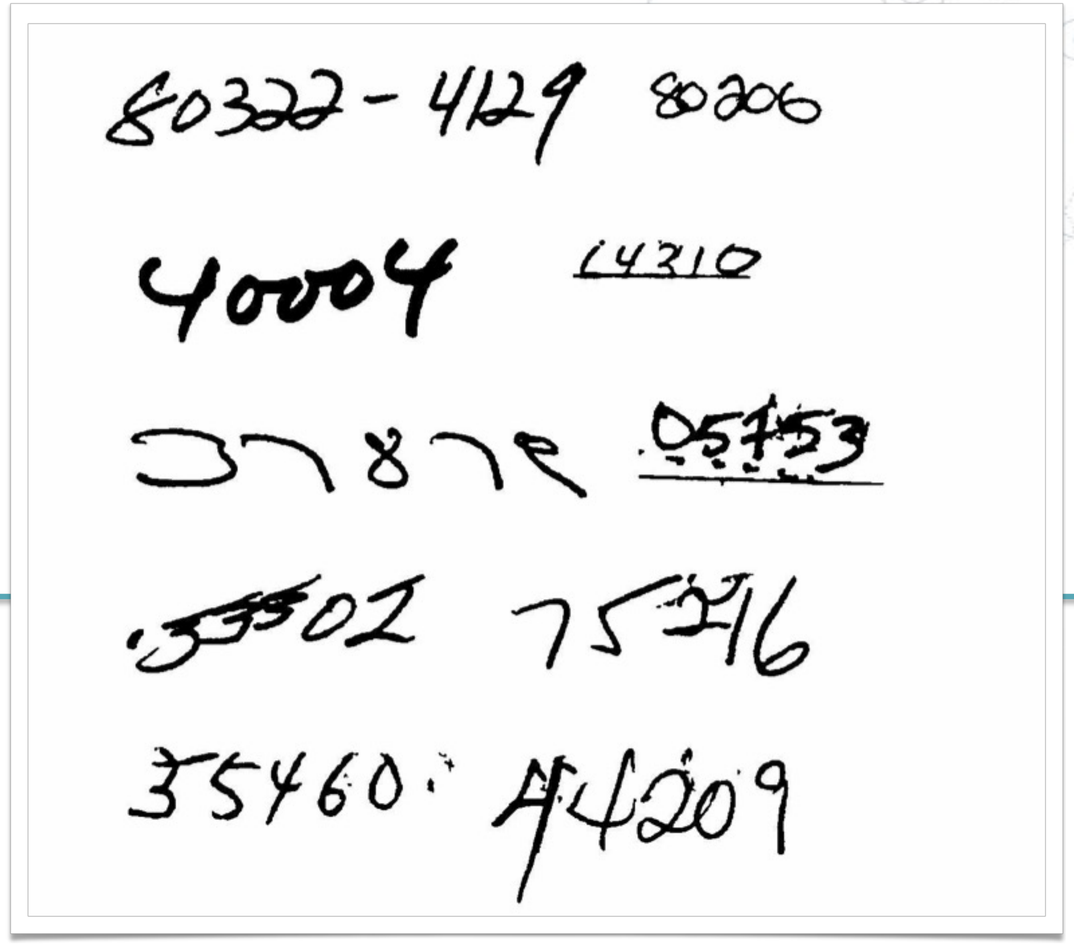
Fukushima: deep learning architecture for computer vision

1986

1989

The term of deep learning is introduced

LeCunn: Applied back-propagation algorithm, to a deep neural network with the purpose of recognising handwritten ZIP codes



LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), pp.541-551.

A short history of deep learning

1965

Ivakhnenko: “working algorithm for supervised, deep, feedforward, multilayer perceptrons”

1980

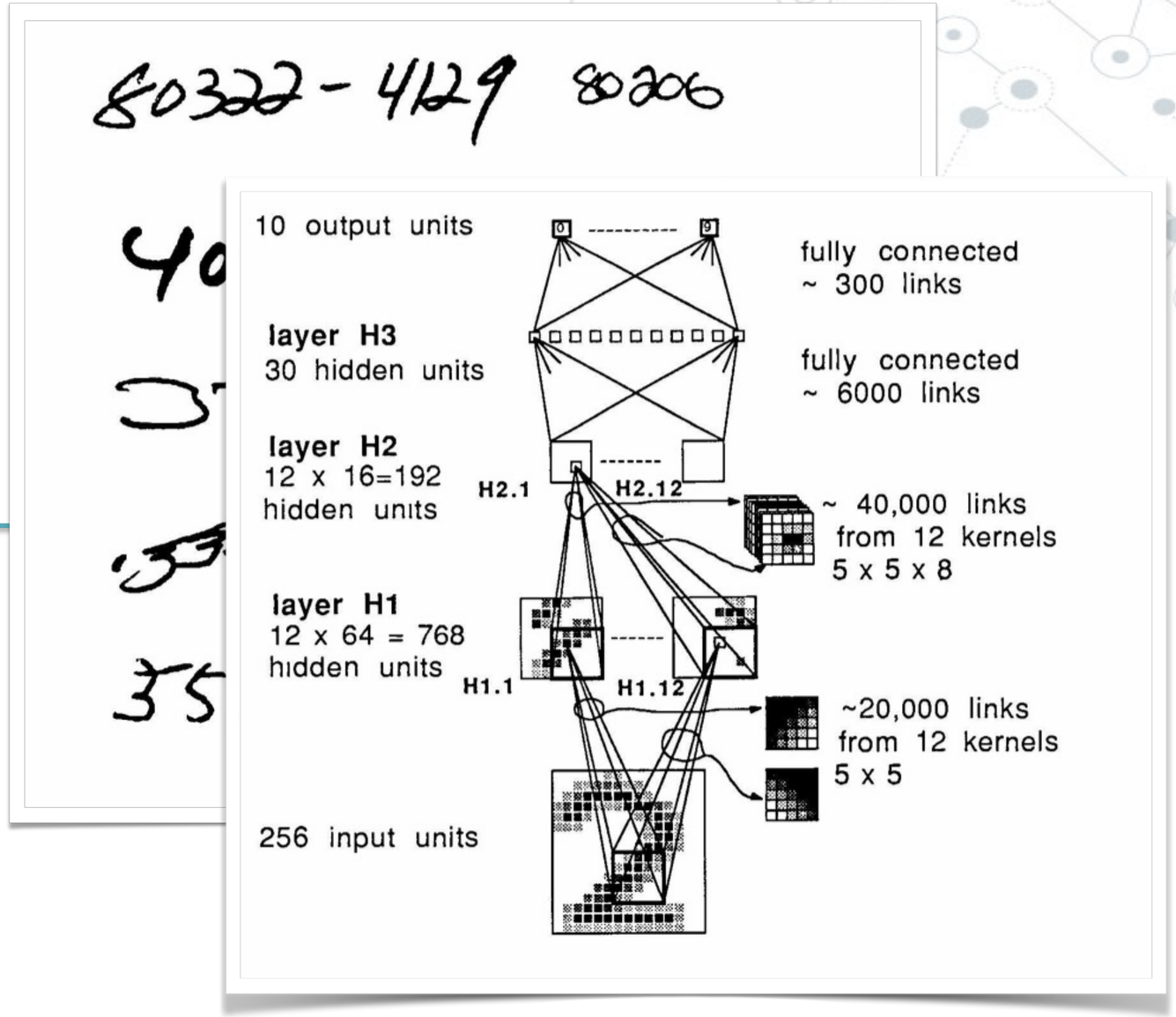
Fukushima: deep learning architecture for computer vision

1986

The term of deep learning is introduced

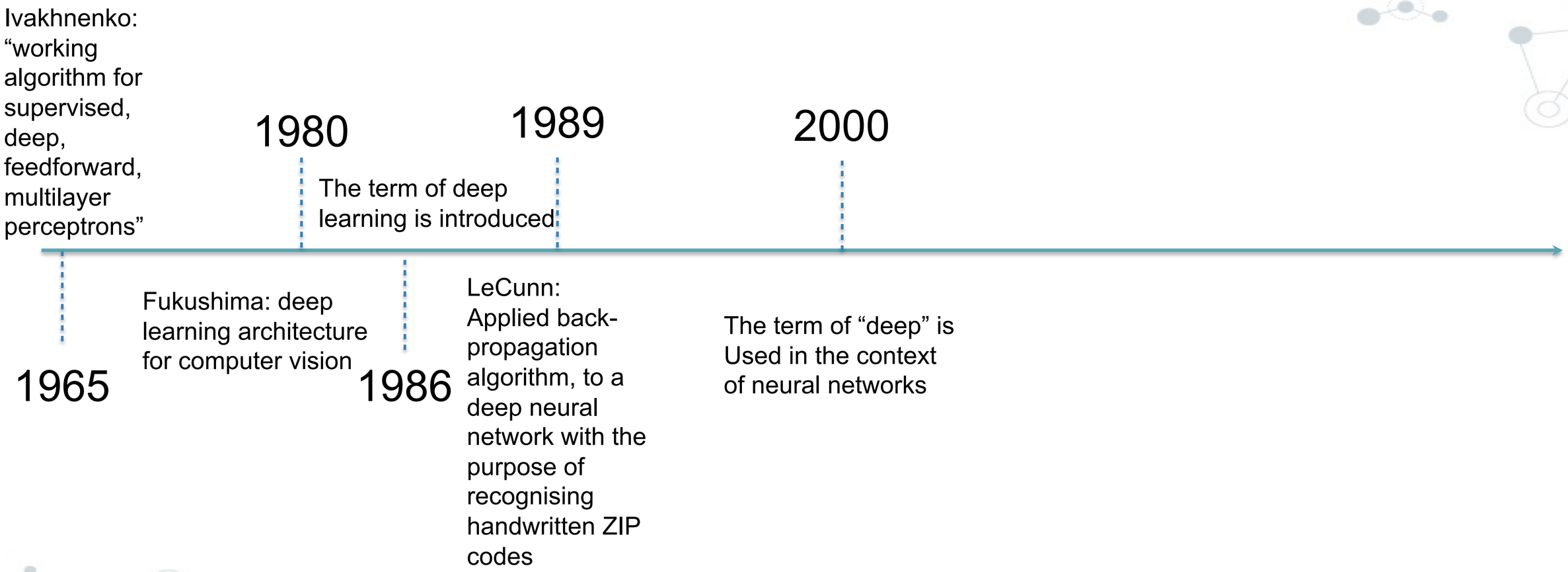
1989

LeCunn: Applied back-propagation algorithm, to a deep neural network with the purpose of recognising handwritten ZIP codes



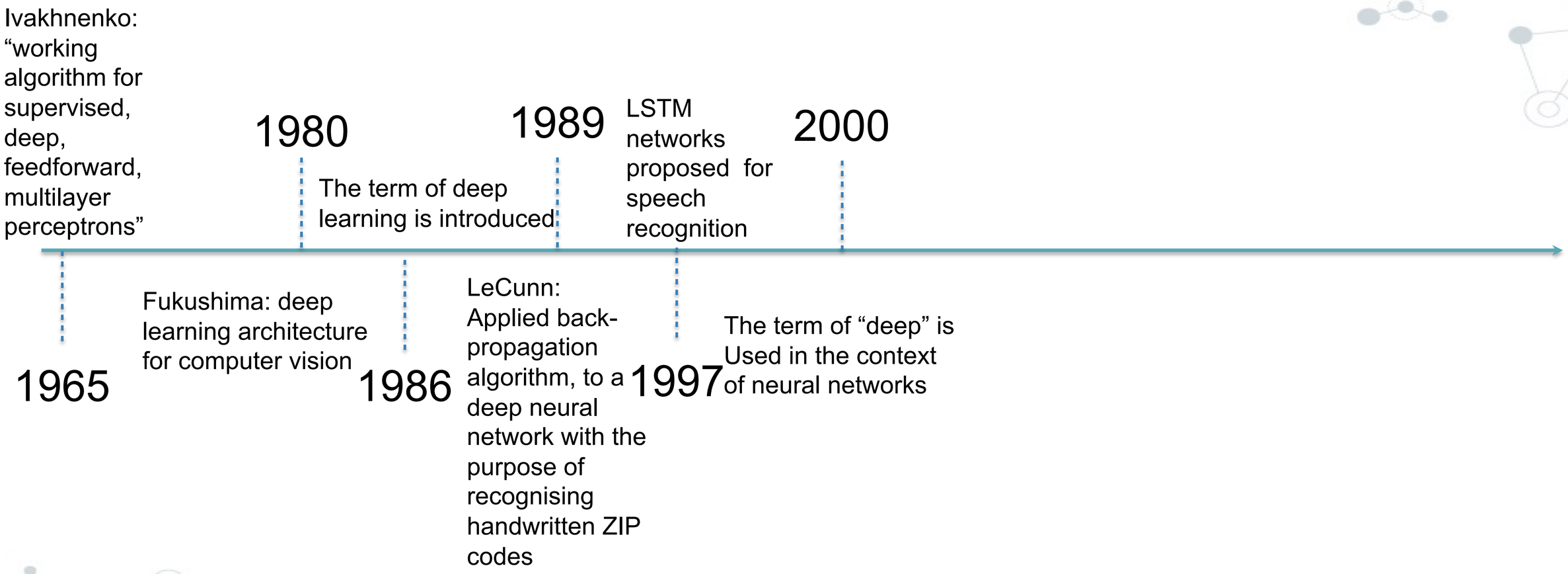
LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), pp.541-551.

A short history of deep learning



LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), pp.541-551.

A short history of deep learning



Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

A short history of deep learning

Ivakhnenko: "working algorithm for supervised, deep, feedforward, multilayer perceptrons"

1965

Fukushima: deep learning architecture for computer vision

1980

1986

The term of deep learning is introduced

LeCunn: Applied back-propagation algorithm, to a deep neural network with the purpose of recognising handwritten ZIP codes

1989

1997

LSTM networks proposed for speech recognition

The t
Usec
of ne

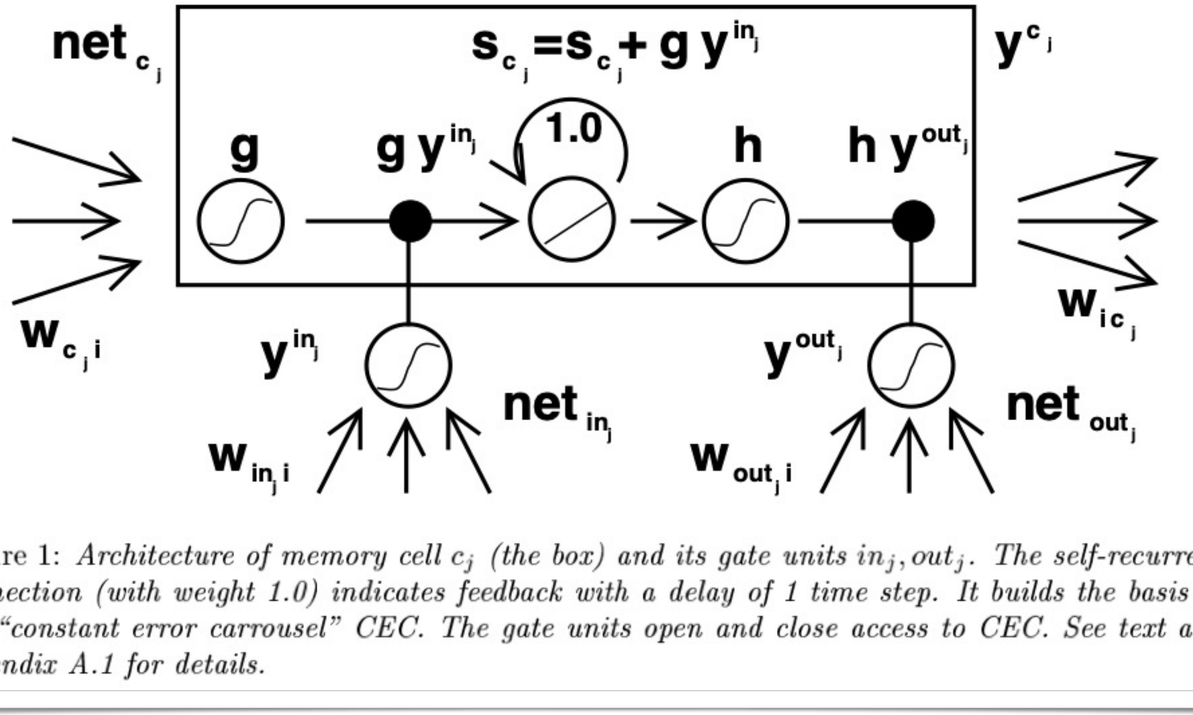
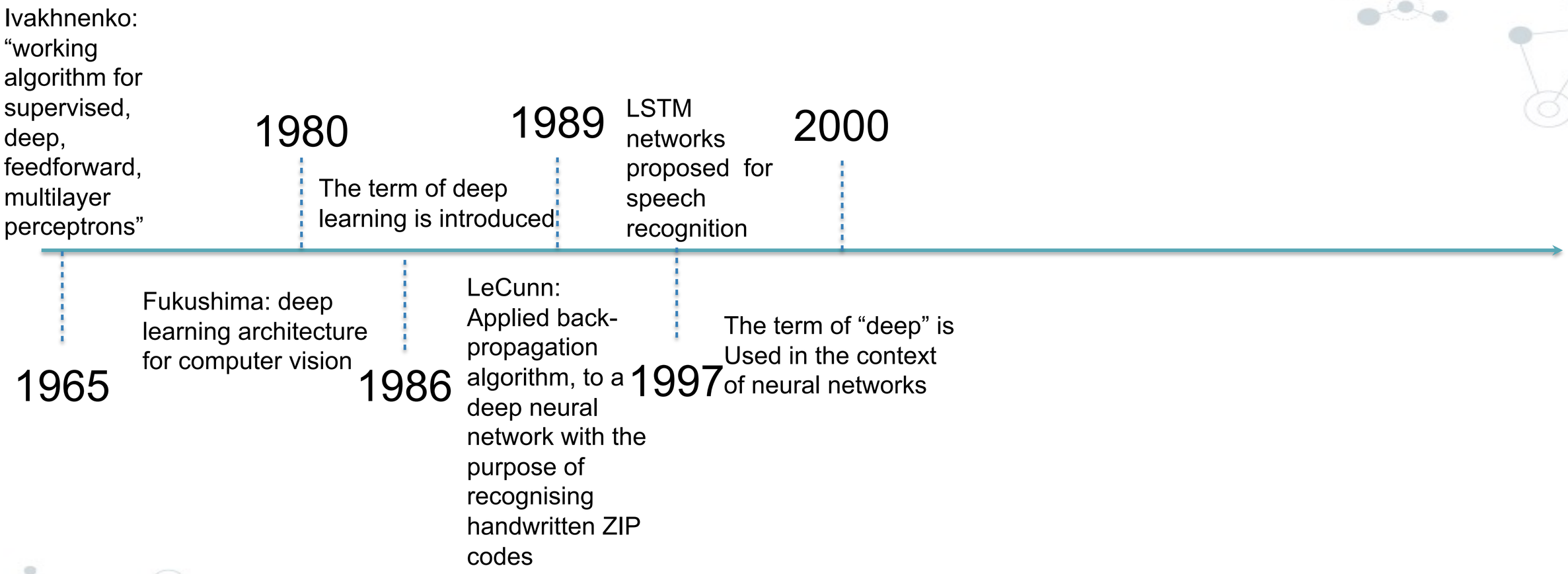


Figure 1: *Architecture of memory cell c_j (the box) and its gate units in_j, out_j. The self-recurre connection (with weight 1.0) indicates feedback with a delay of 1 time step. It builds the basis the "constant error carrousel" CEC. The gate units open and close access to CEC. See text a appendix A.1 for details.*

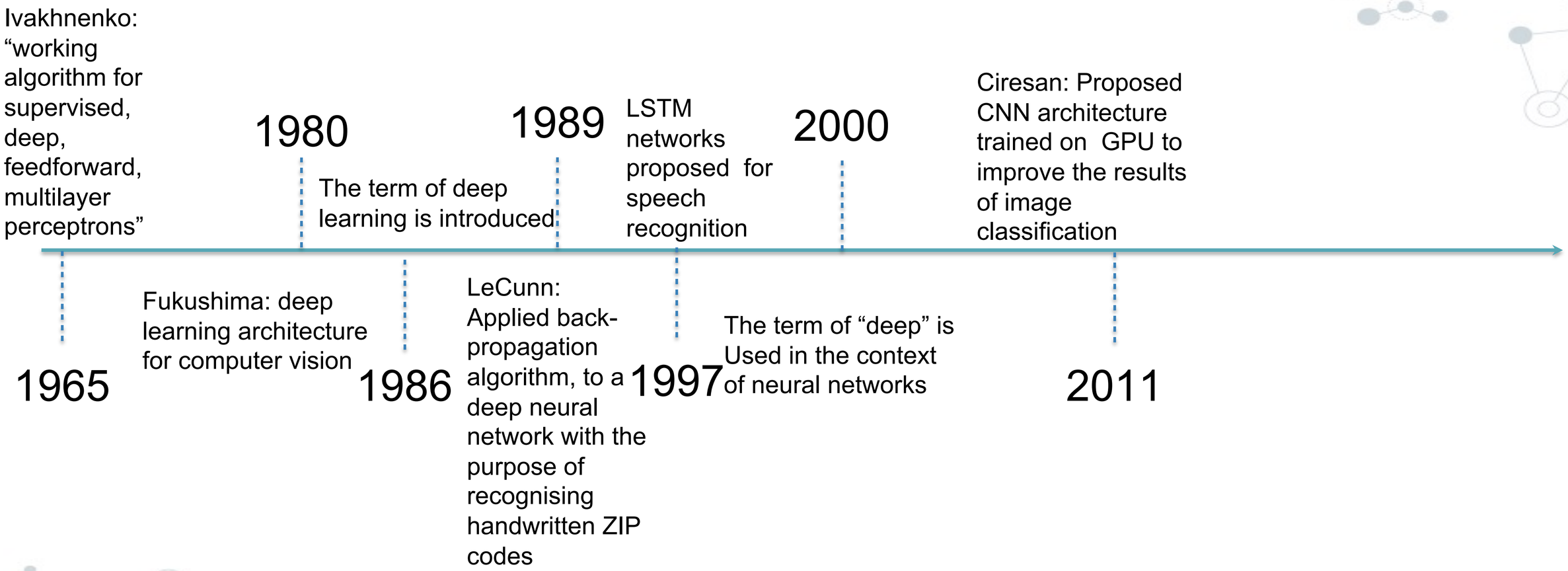
Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

A short history of deep learning



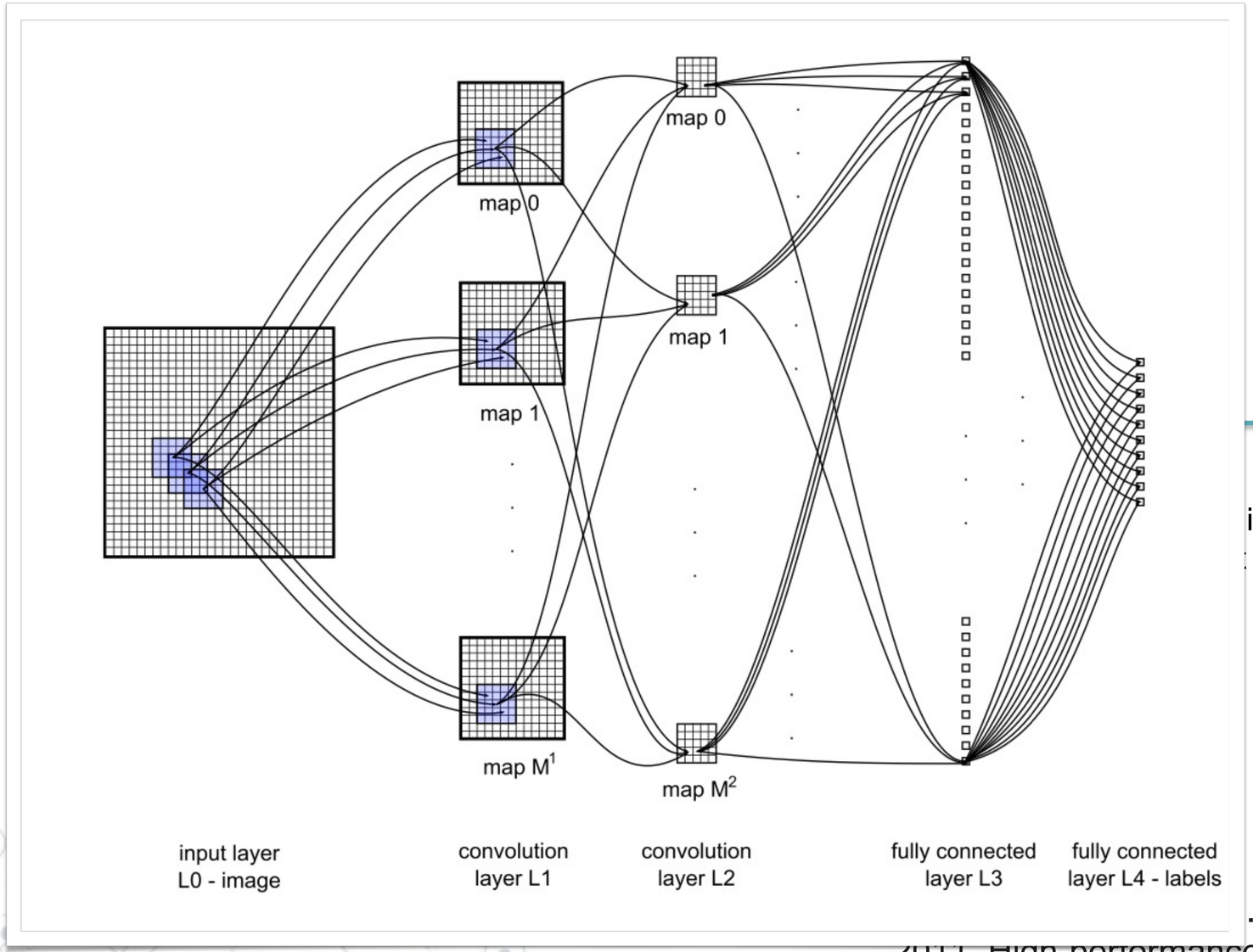
Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

A short history of deep learning



Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M. and Schmidhuber, J., 2011. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.

A short history of deep learning



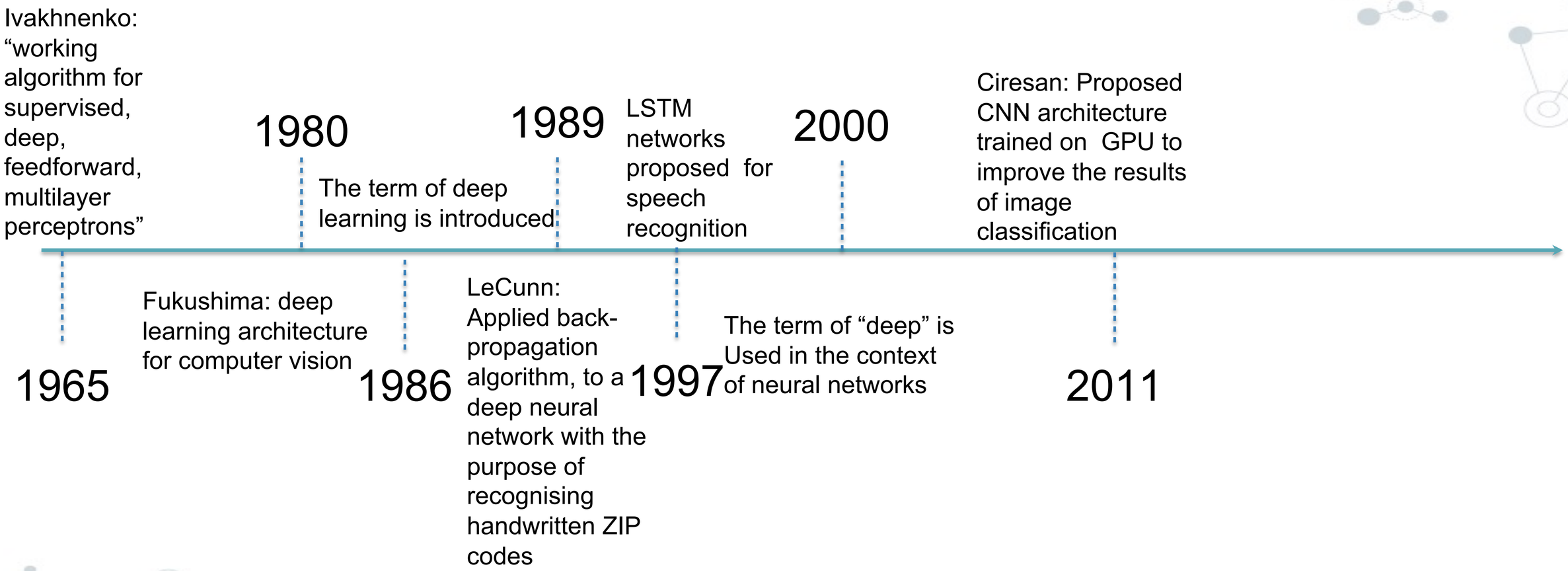
Ciresan: Proposed CNN architecture trained on GPU to improve the results of image classification

is

2011

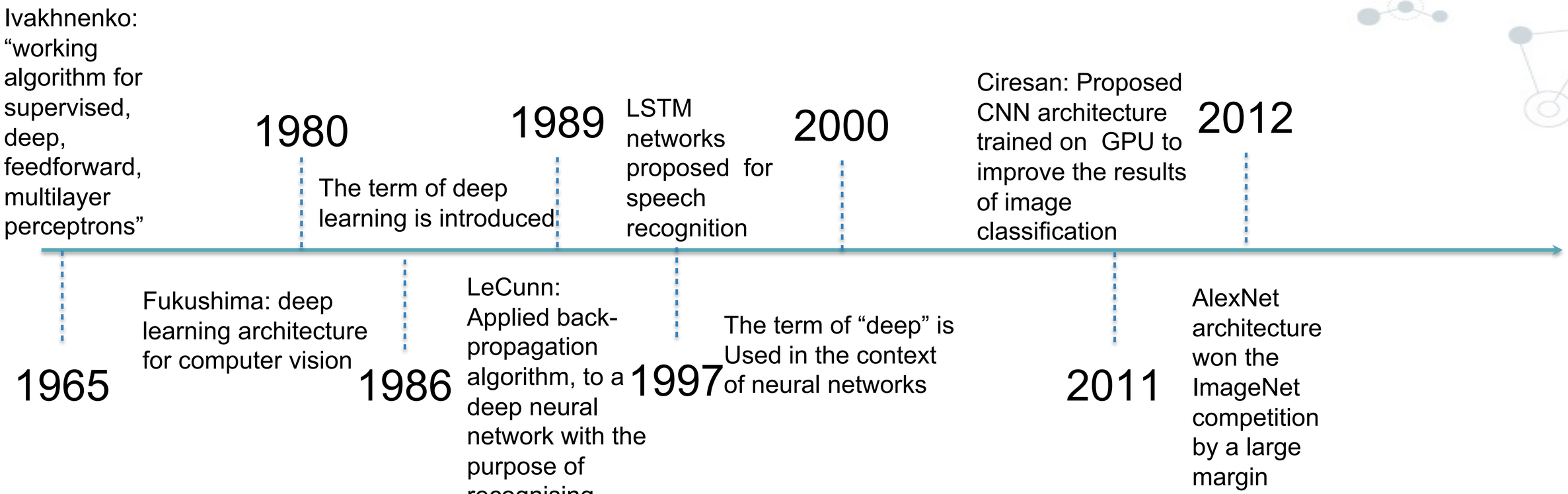
., Masci, J., Gambardella, L.M. and Schmidhuber, J., 2011. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.

A short history of deep learning



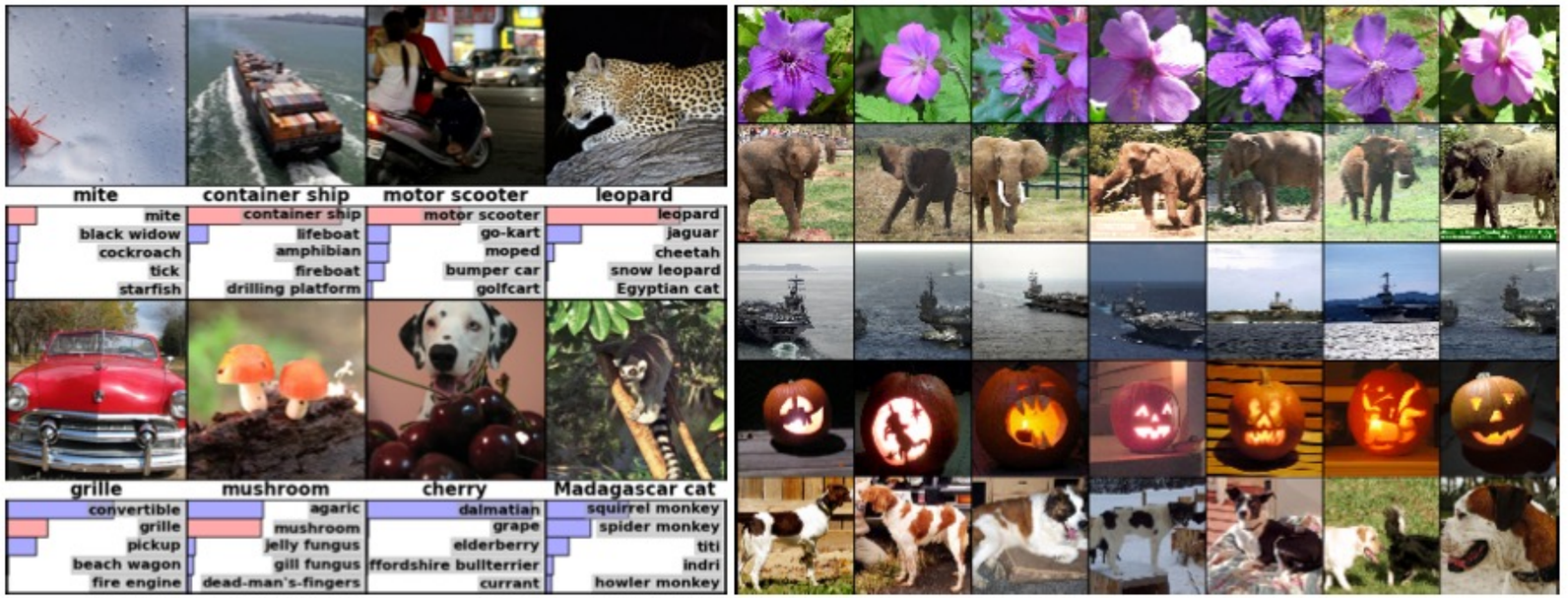
Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M. and Schmidhuber, J., 2011. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.

A short history of deep learning



Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), pp.84-90.

A short history of deep learning



osed
ture
PU to
esults

2012

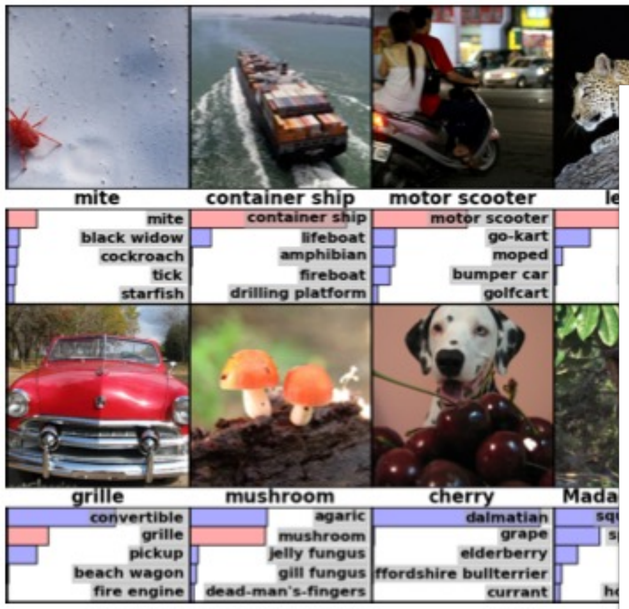
AlexNet architecture won the ImageNet competition by a large margin

l1

network with the purpose of recognising handwritten ZIP codes

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), pp.84-90.

A short history of deep learning



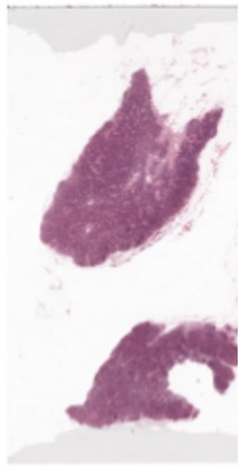
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

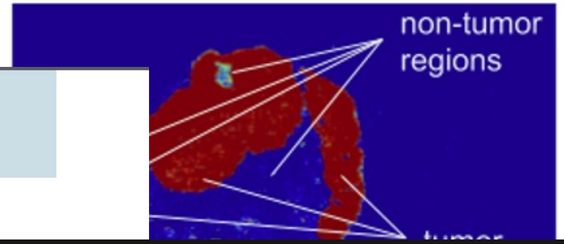
Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), pp.84-90.


GARRY
KASPAROV
DEEP
BLUE
15:09:49

Deep Learning to detect tumours





MENU SCIENTIFIC REPORTS

Article | Open Access | Published: 29 August 2019

Deep Learning to
Detection on Scre

Li Shen, Laurie R. Margolies, Jos
McBride & Weiva Sieh

Scientific Reports 9, Article numbe

Abstract

The rapid development of deep
techniques, has spurred much
imaging problems. Here, we de
accurately detect breast cance
"end-to-end" training approac
datasets with either complete

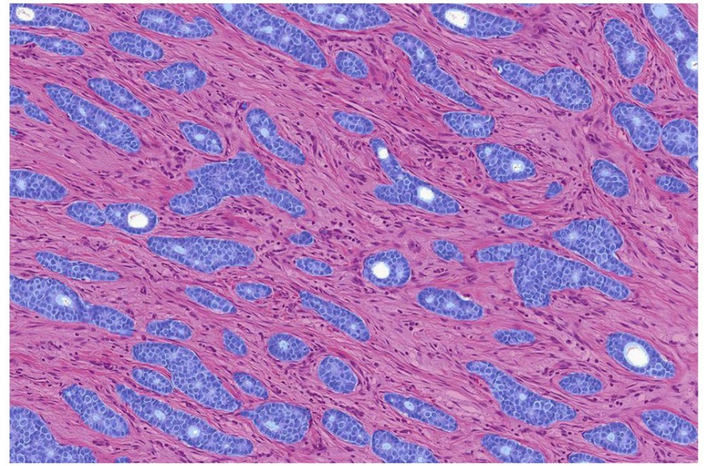
Cookies on Forbes

Forbes Billionaires Innovation Leadership Money Business Small Business Lifestyle Lis

7,494 views | Apr 30, 2019, 11:46am

The Cutting-Edge Of AI Cancer Detection

Charles Towers-Clark Contributor
AI & Big Data
I write about AI, data, deep tech & self-management in the digital age



Detecting and treating cancer could benefit from a dose of AI, whether in pathological H&E slides or... [+] LUNIT.IO

https://ai.googleblog.com/2017/03/assisting-pathologists-in-detecting.html


1
2
3
4

Check out these links:

- https://www.nytimes.com/interactive/2020/11/21/science/artificial-intelligence-fake-people-faces.html
- https://www.thispersondoesnotexist.com

What changed?

Deep learning vs Traditional neural networks

Google Trends | Explore

Sign in

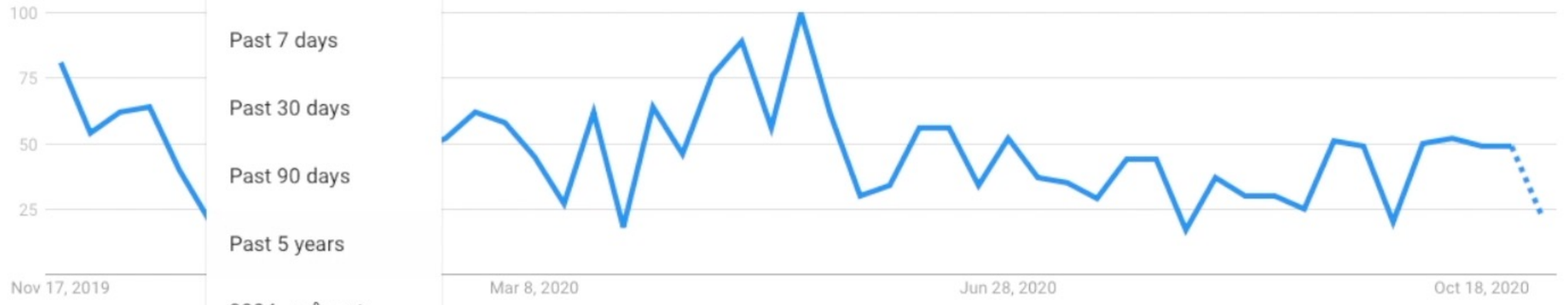
Decision Trees
Search term

+ Compare

United States | Past 12 months | All categories | Web Search

Past hour
Past 4 hours
Past day
Past 7 days
Past 30 days
Past 90 days
Past 5 years
2004 - present
Custom time rang...

Interest over time



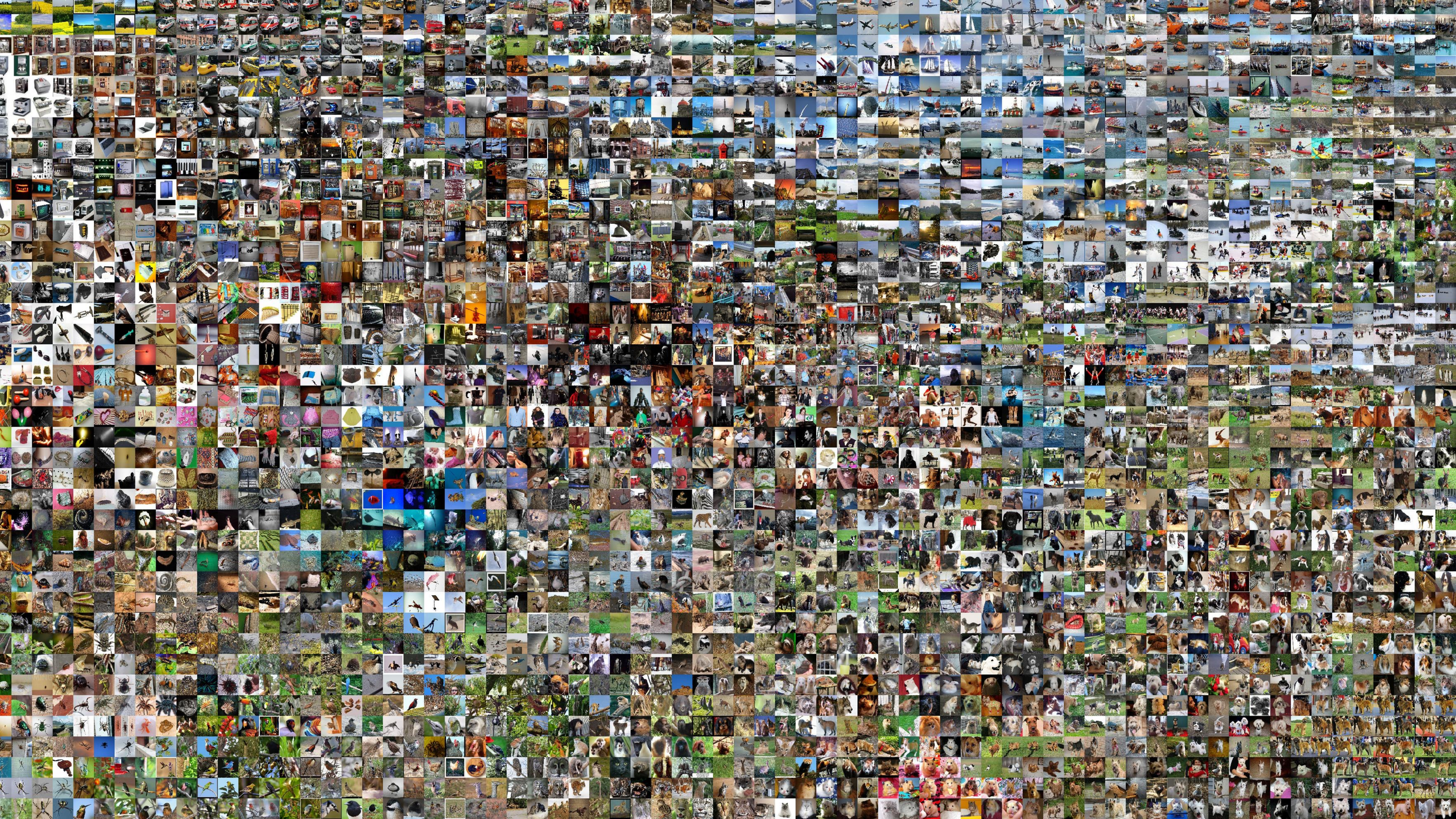
Interest by subregion

Subregion

1	Massachusetts	100
2	New Jersey	94

Lots and lots of

DATA



COMPUTATION POWER

GPU

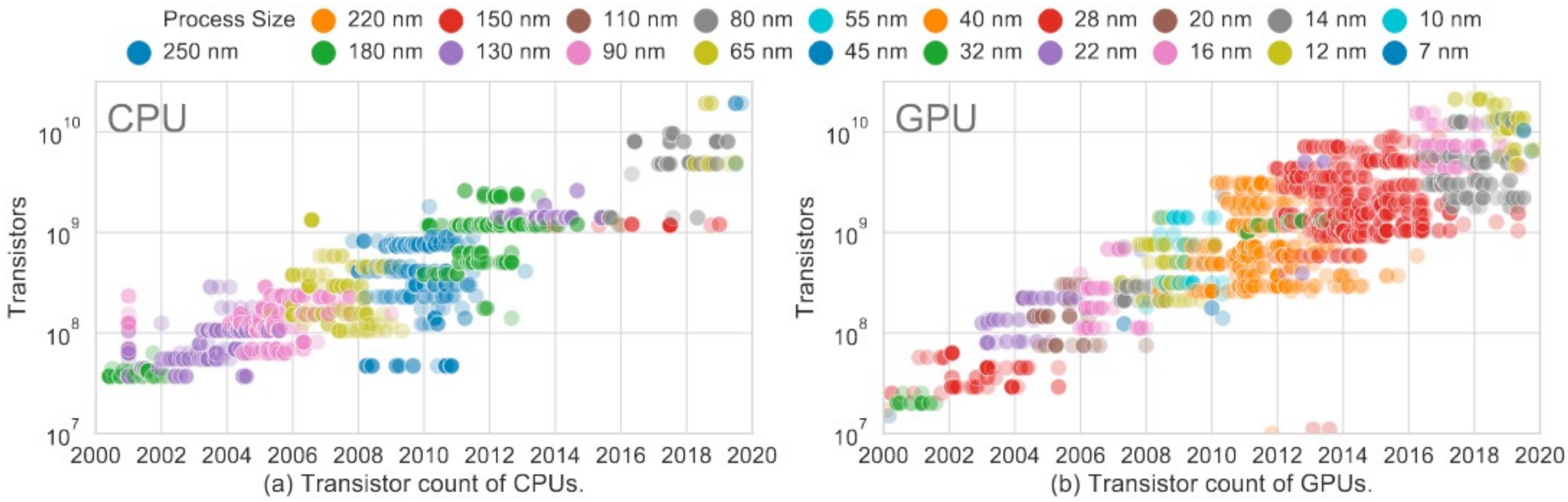
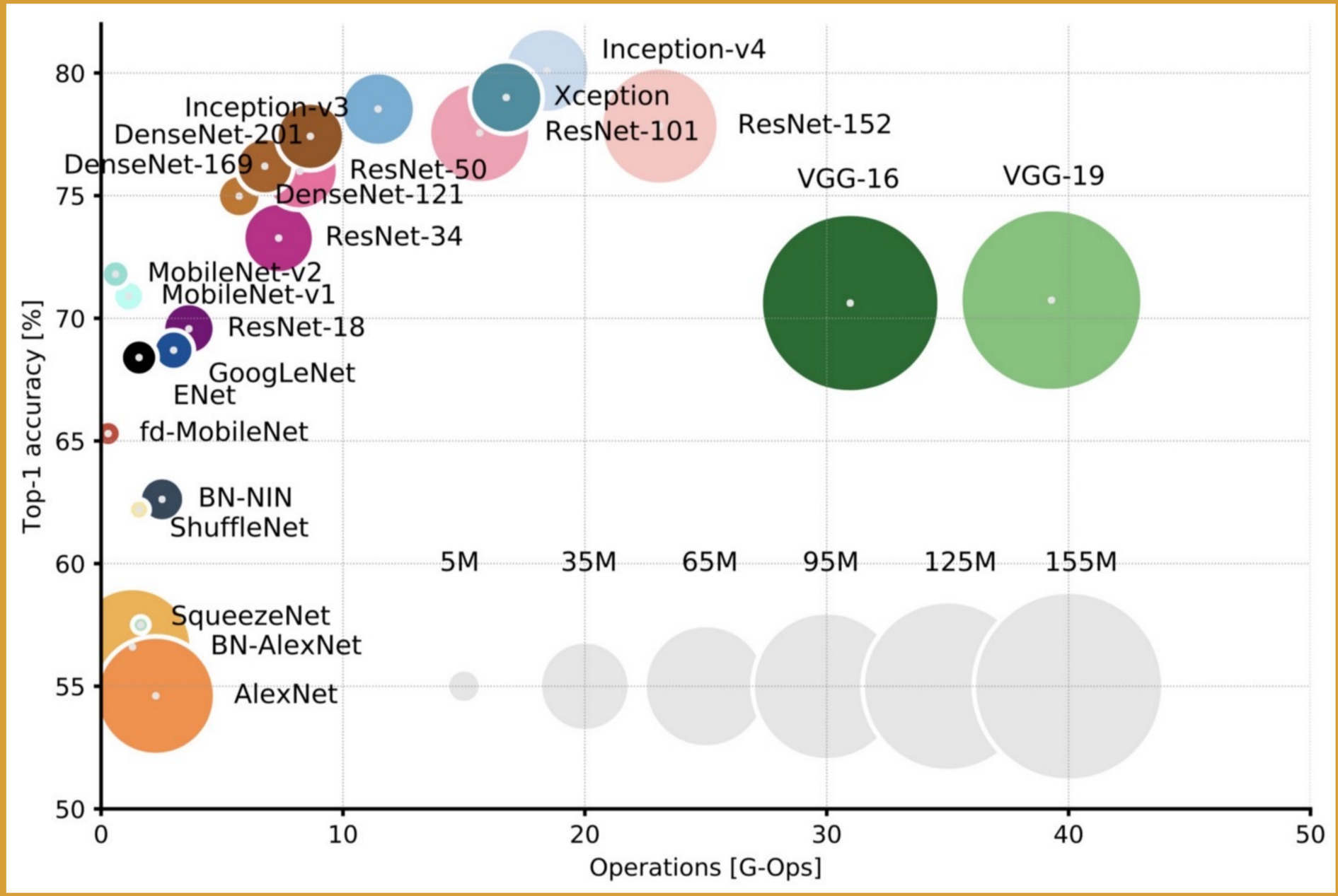


Fig. 1. Moore's Law is still valid for both CPUs and GPUs.

Sun, Y., Agostini, N.B., Dong, S. and Kaeli, D., 2019. Summarizing CPU and GPU Design Trends with Product Data. *arXiv preprint arXiv:1911.11313*.

Size of the Neural Network



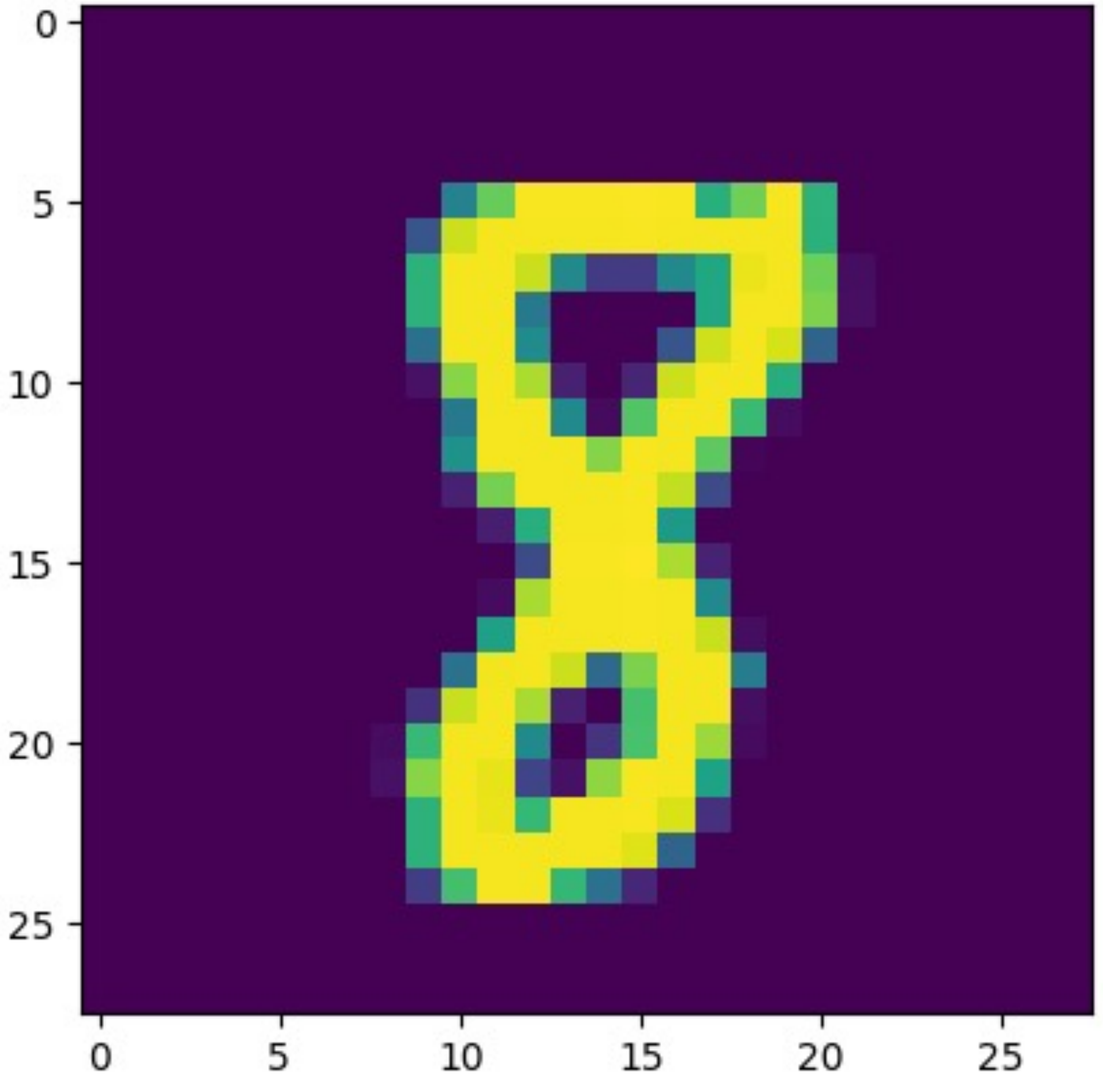
Canziani, A., Paszke, A. and Culurciello, E., 2016. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

A few tweaks about the neural network itself

(activation function, weight initialisation)

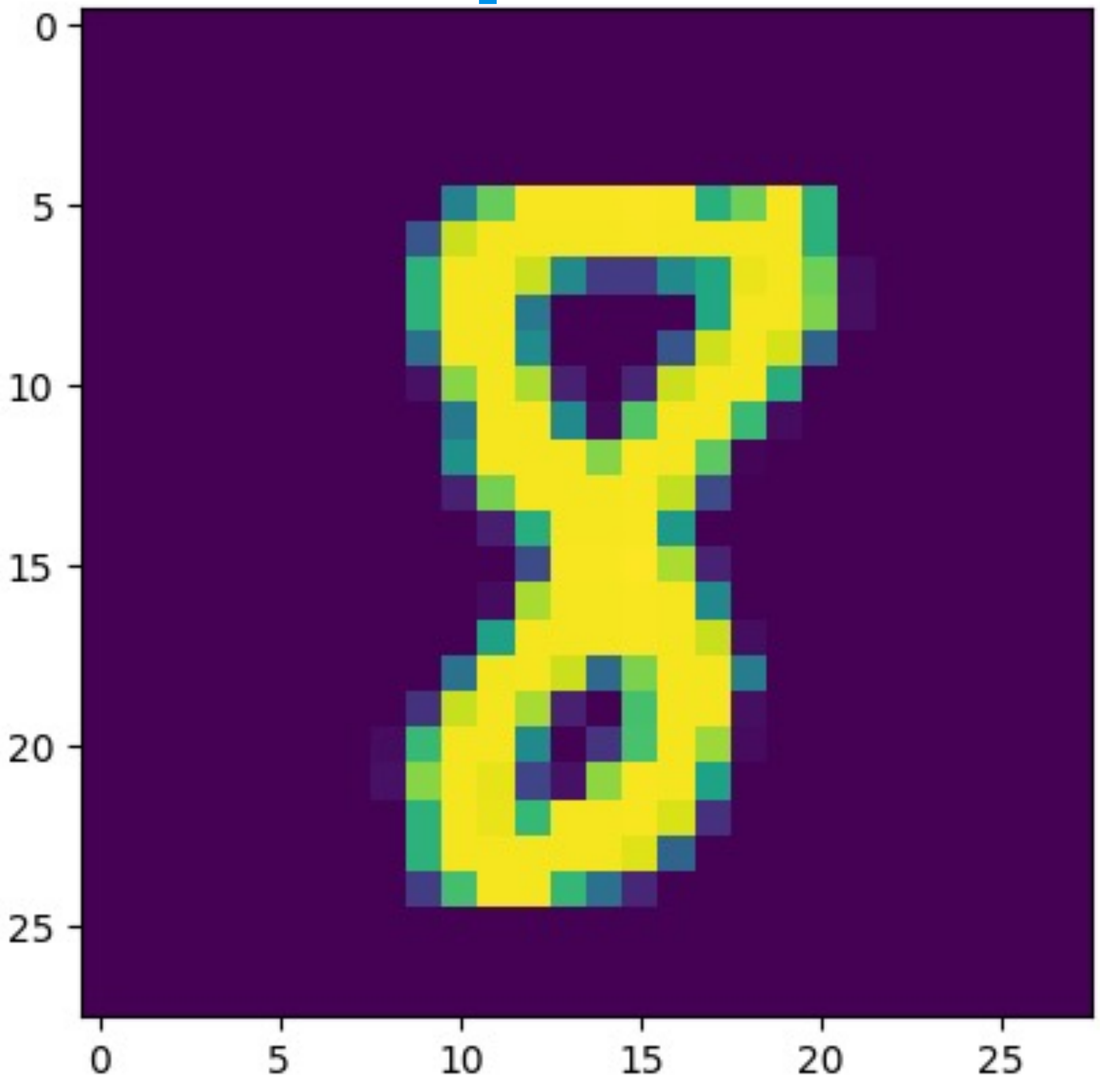
Notations

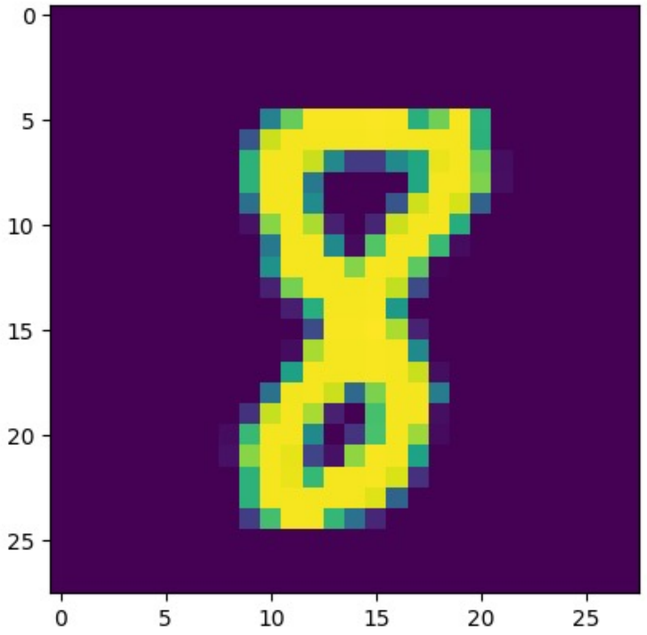
MNIST dataset - “Hello World” for image recognition


0
5
10
15
20
25
0
5
10
15
20
25

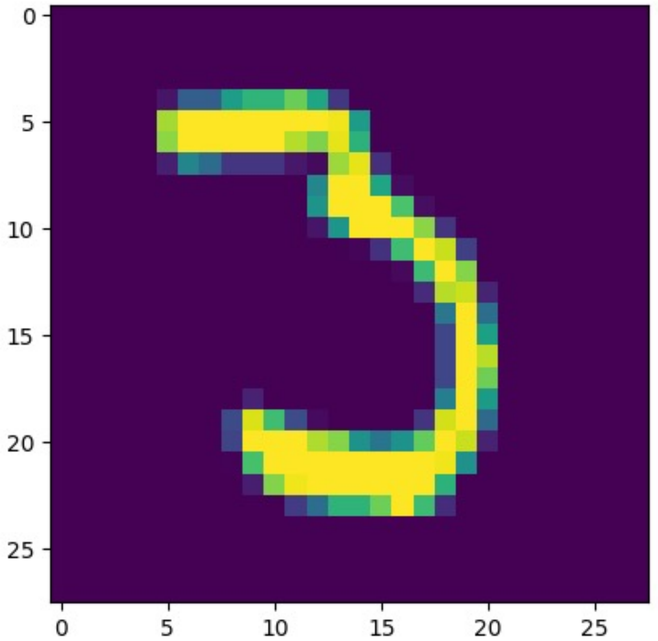
28 pixels

28

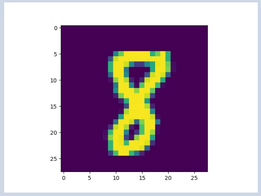
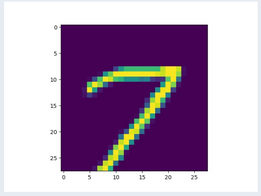
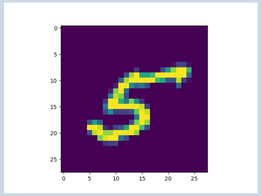
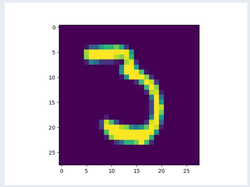




vs.



0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44	0.77	0.99	0.99	0.99	1.00	0.99	0.64	0.79	0.99	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.93	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.92	0.47	0.17	0.17	0.47	0.60	0.97	0.99	0.78	0.04	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.40	0.00	0.00	0.00	0.00	0.60	0.99	0.99	0.81	0.04	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.99	0.99	0.48	0.00	0.00	0.00	0.27	0.93	0.99	0.94	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.82	0.99	0.88	0.09	0.00	0.11	0.93	0.99	0.99	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.99	0.99	0.48	0.03	0.73	0.99	0.99	0.68	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.51	0.99	0.99	0.99	0.82	0.99	0.99	0.75	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.80	0.99	0.99	0.99	0.99	0.91	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.63	0.99	0.99	0.99	0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.99	0.99	1.00	0.88	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.87	0.99	0.99	0.99	0.99	0.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.99	0.99	0.99	0.99	0.99	0.92	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.98	0.99	0.93	0.34	0.80	0.99	0.99	0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.92	0.99	0.87	0.09	0.00	0.71	0.99	0.99	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.67	0.99	0.99	0.48	0.00	0.15	0.71	0.99	0.85	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.82	0.99	0.97	0.21	0.05	0.84	0.99	0.99	0.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.97	0.67	0.99	0.99	0.99	0.95	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.99	0.99	0.99	0.95	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.71	0.99	0.99	0.67	0.37	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

INPUT (X)	OUPUT (Y)
	8
	7
	5
	3

x - the set examples

y - the set of labels

$x^{(1)}$ - refers to the first example

$x_1^{(1)}$ - first feature from the first example

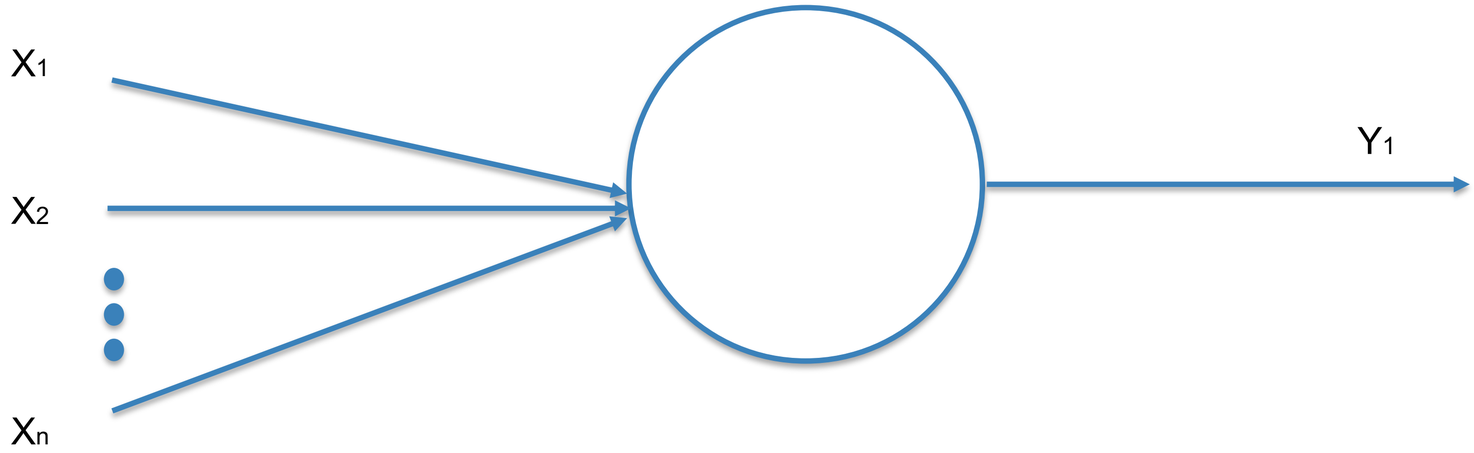
$y^{(1)}$ - label of the first example

N - number of features

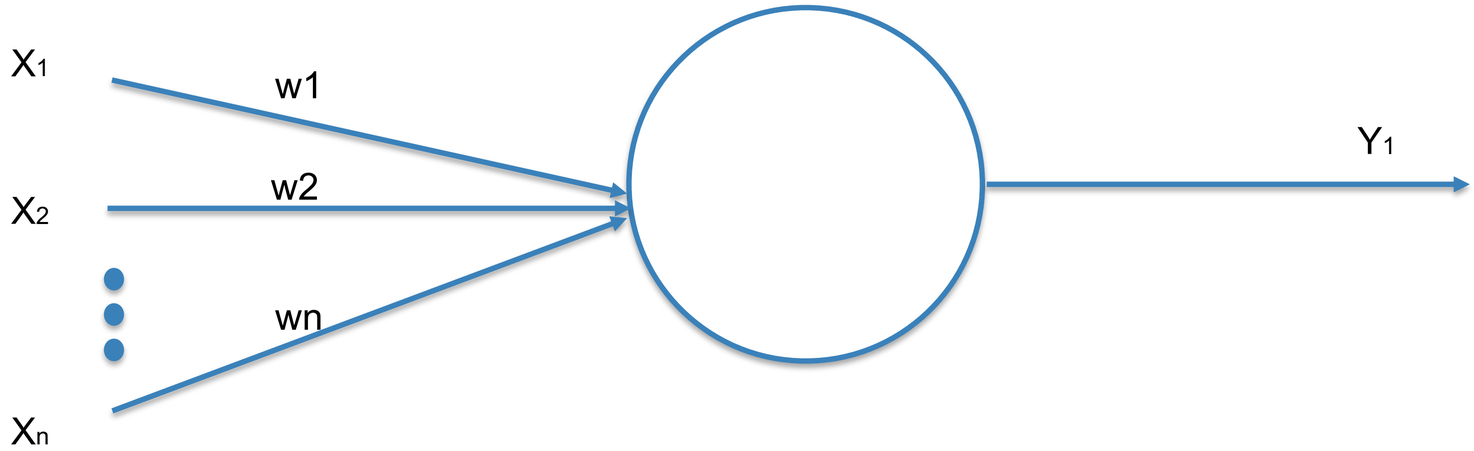
M - number of examples

Recap: Neural Networks

Neural Networks

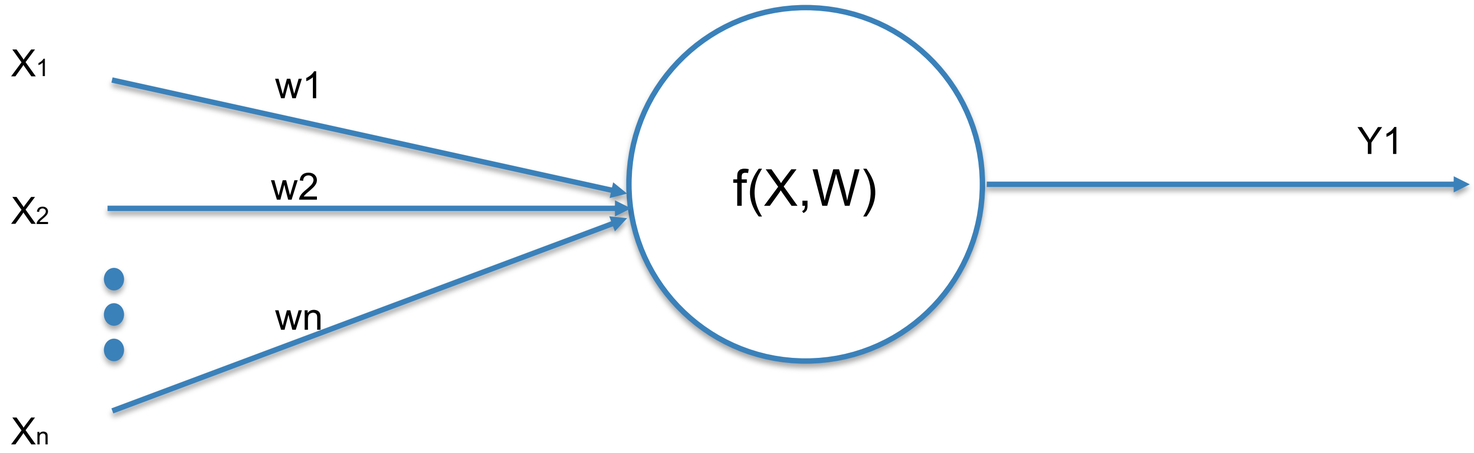


Neural Networks



The **weights** (w) give the importance of a input

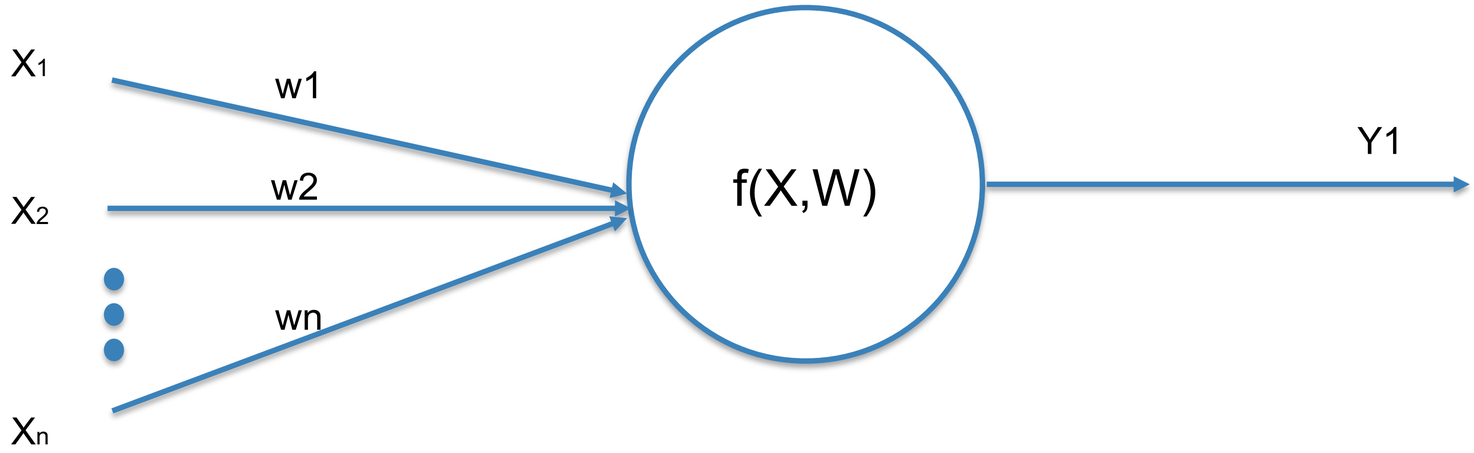
Neural Networks



A NEURON:

- **Receives input (x)**
- **Computes output (y) based on an activation function f**

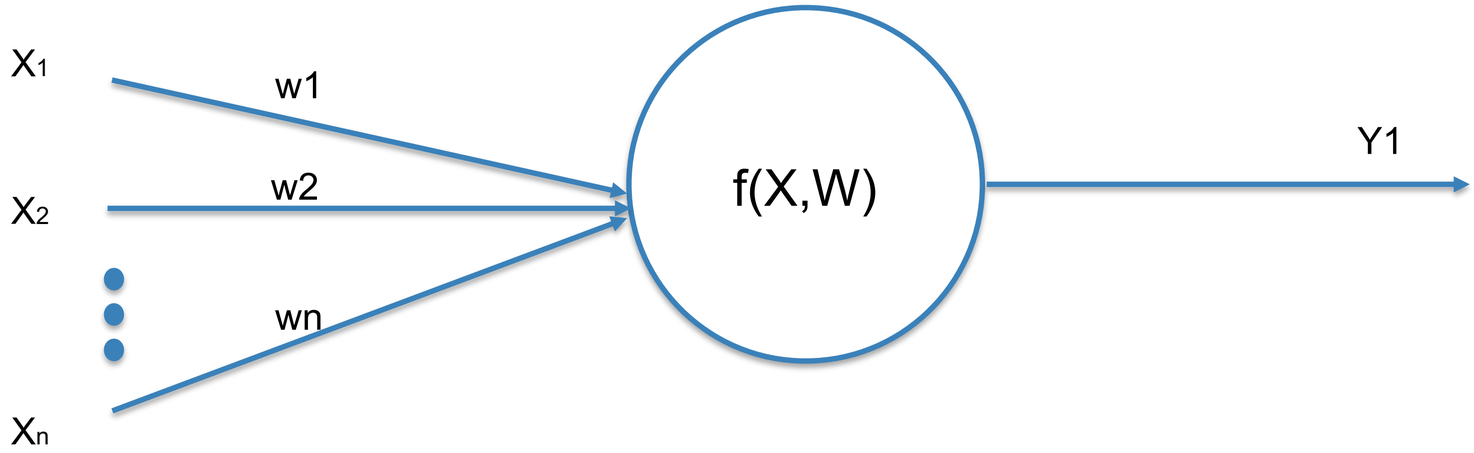
Artificial neuron



A NEURON:

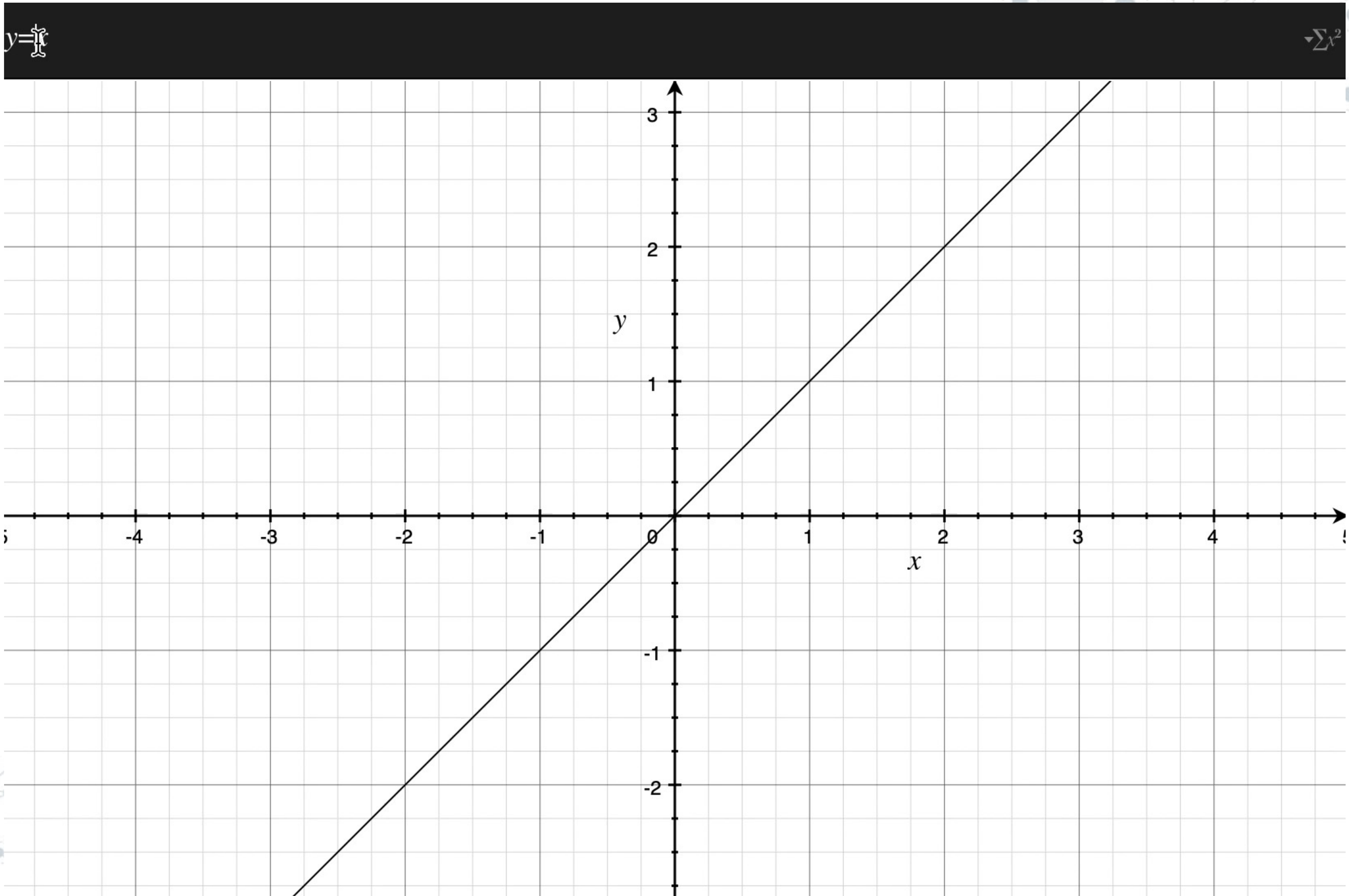
- **Receives input (x)**
- **Computes output (y) based on an activation function f**

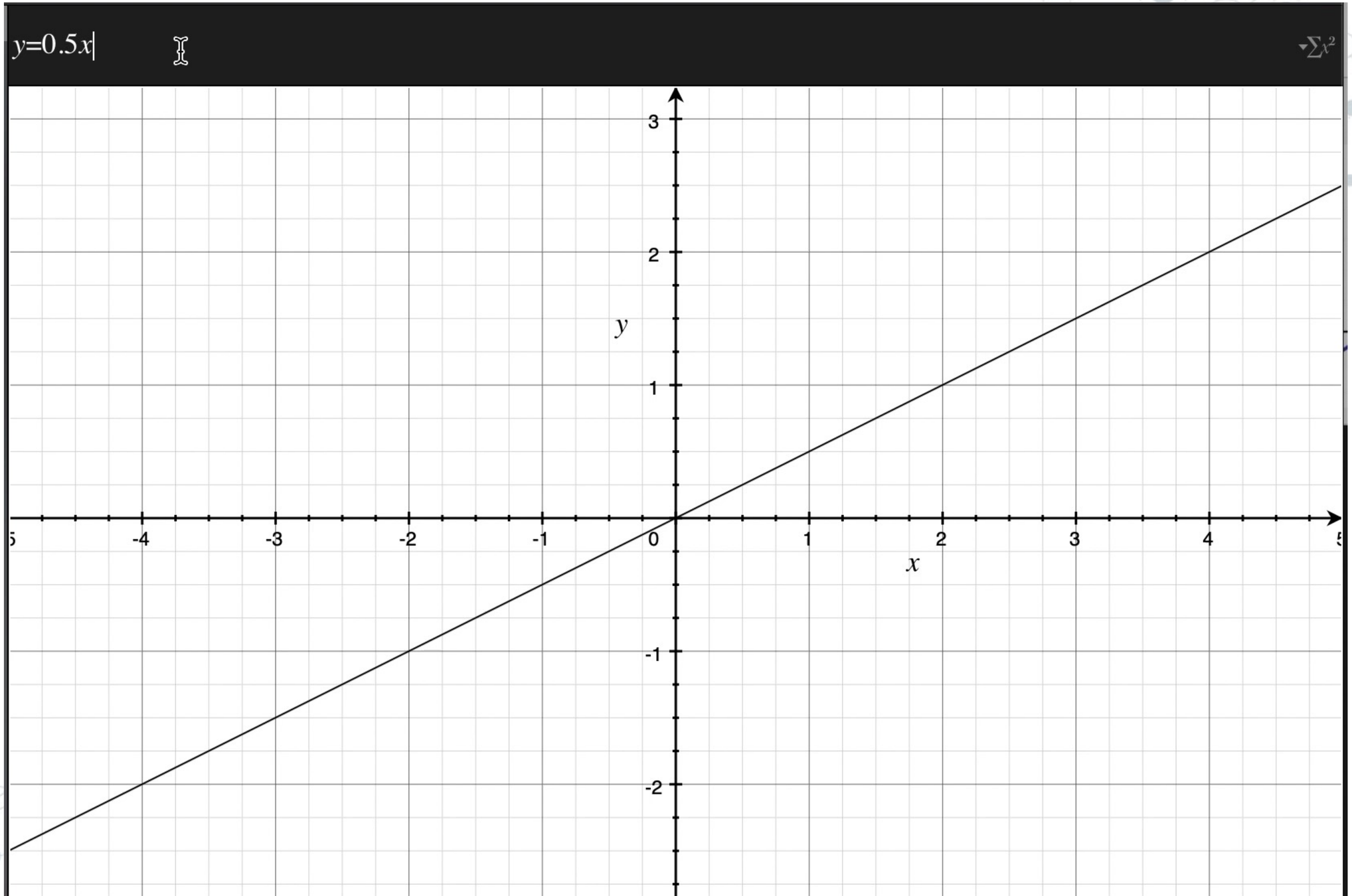
Neural Networks



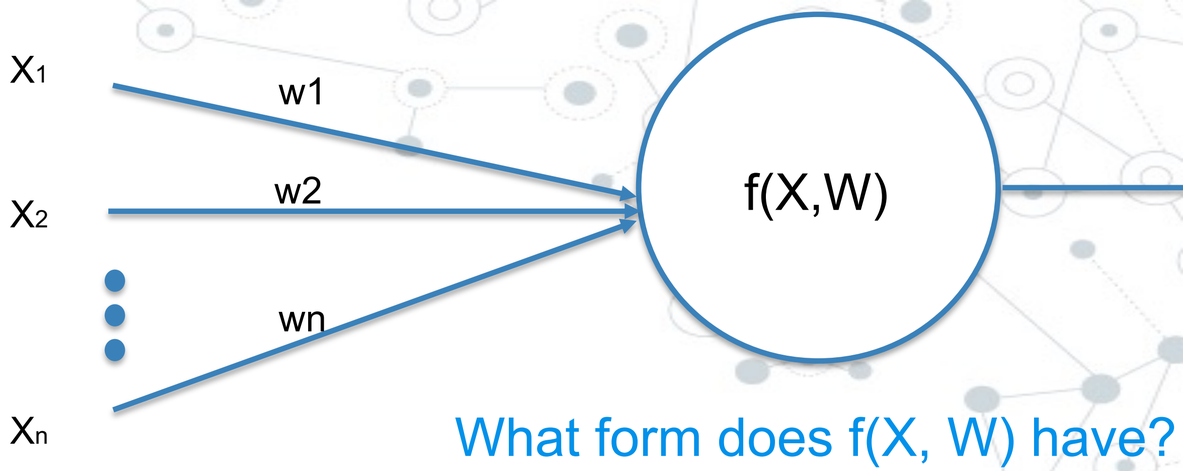
Activation function: $f(w_1x_1+w_2x_2+w_3x_3+\ldots + w_nx_n + b)$

b is a bias term. Think of it as the it is somehow similar to the constant b of a linear function (y = ax+b)


y=x
3
2
y
1
-4
-3
-2
-1
0
1
2
3
4
x
-1
-2

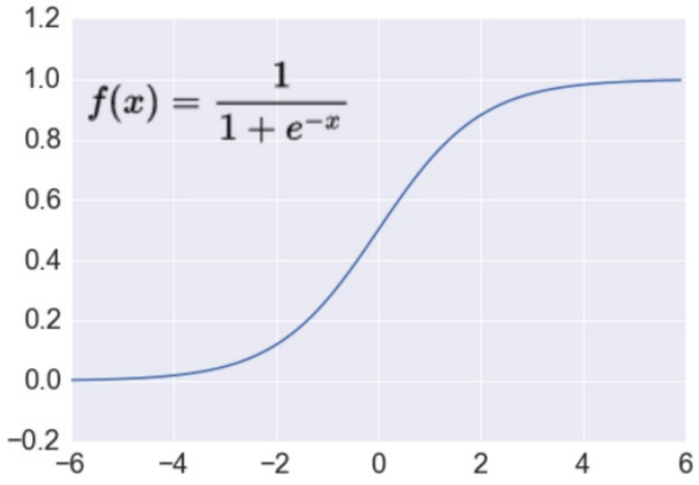

y=0.5x
3
2
y
1
-4
-3
-2
-1
0
1
2
3
4
x
-1
-2

Neural Network - Activation functions

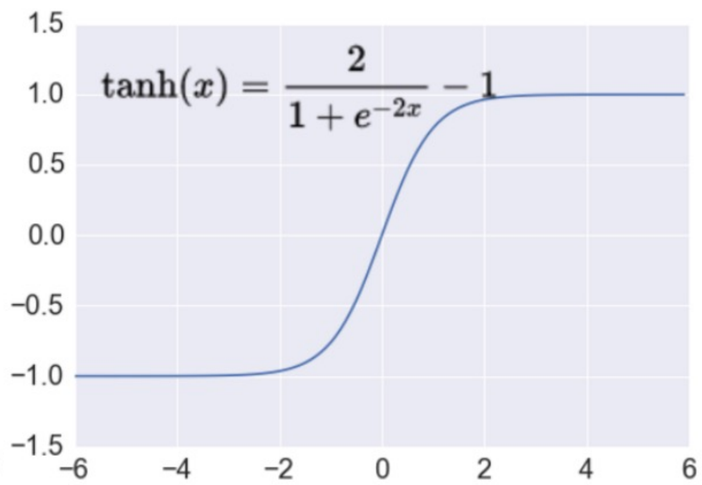


What form does f(X, W) have?

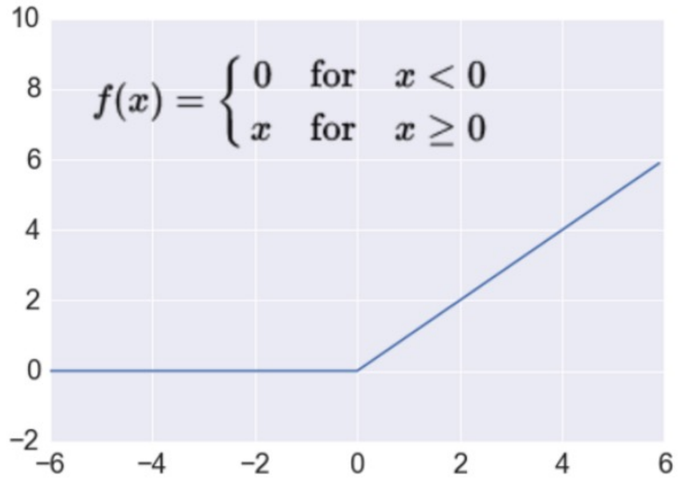
Sigmoid



TanH



ReLU

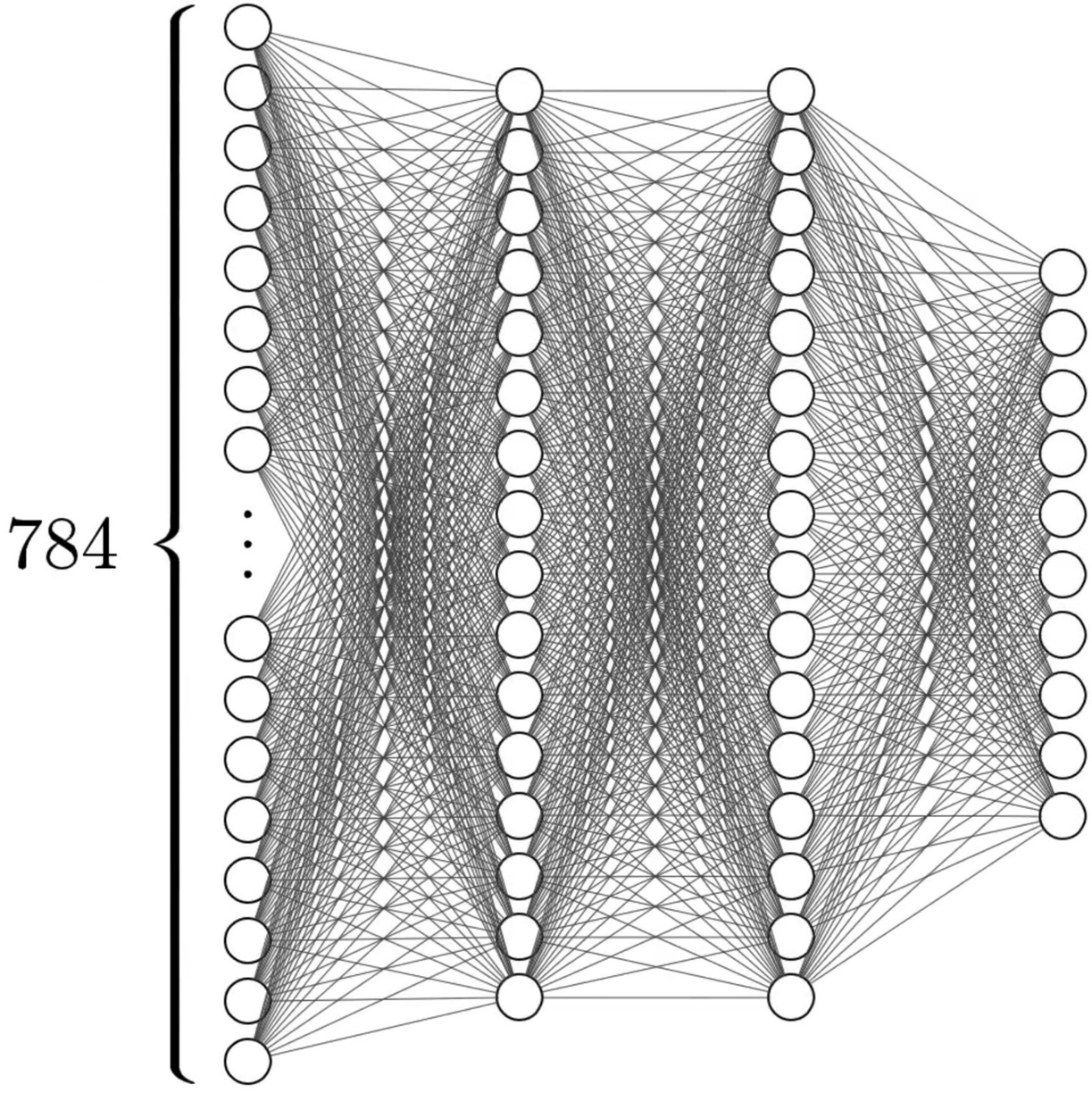


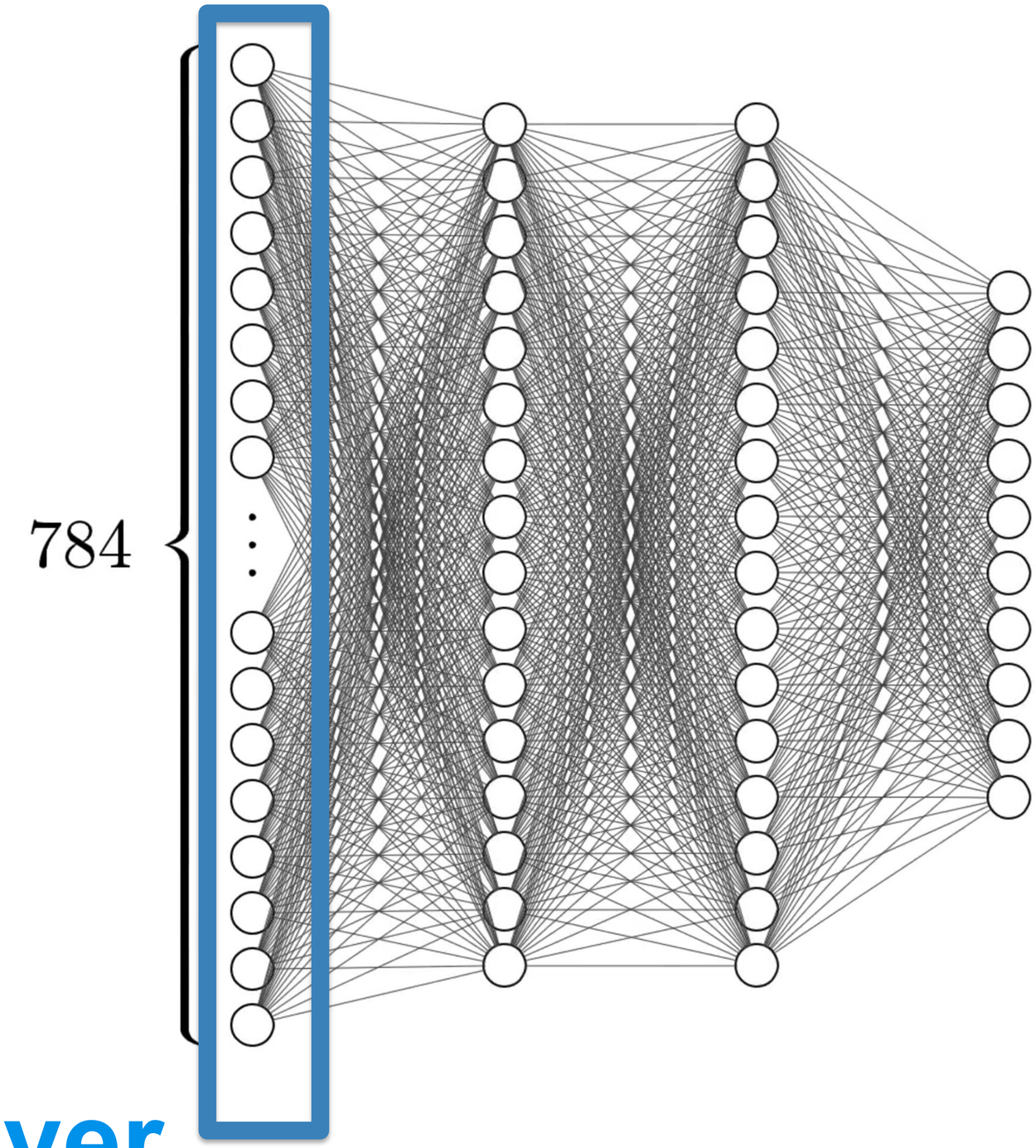
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44	0.77	0.99	0.99	0.99	1.00	0.99	0.64	0.79	0.99	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.93	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.92	0.47	0.17	0.17	0.47	0.60	0.97	0.99	0.78	0.04	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.40	0.00	0.00	0.00	0.00	0.60	0.99	0.99	0.81	0.04	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.99	0.99	0.48	0.00	0.00	0.00	0.27	0.93	0.99	0.94	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.82	0.99	0.88	0.09	0.00	0.11	0.93	0.99	0.99	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.99	0.99	0.48	0.03	0.73	0.99	0.99	0.68	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.51	0.99	0.99	0.99	0.82	0.99	0.99	0.75	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.80	0.99	0.99	0.99	0.99	0.91	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.63	0.99	0.99	0.99	0.54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.99	0.99	1.00	0.88	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.87	0.99	0.99	0.99	0.99	0.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.99	0.99	0.99	0.99	0.99	0.92	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.98	0.99	0.93	0.34	0.80	0.99	0.99	0.42	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.92	0.99	0.87	0.09	0.00	0.71	0.99	0.99	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.67	0.99	0.99	0.48	0.00	0.15	0.71	0.99	0.85	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.82	0.99	0.97	0.21	0.05	0.84	0.99	0.99	0.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.97	0.67	0.99	0.99	0.99	0.95	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.99	0.99	0.99	0.99	0.99	0.95	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.71	0.99	0.99	0.67	0.37	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

28x28 => 784 pixels

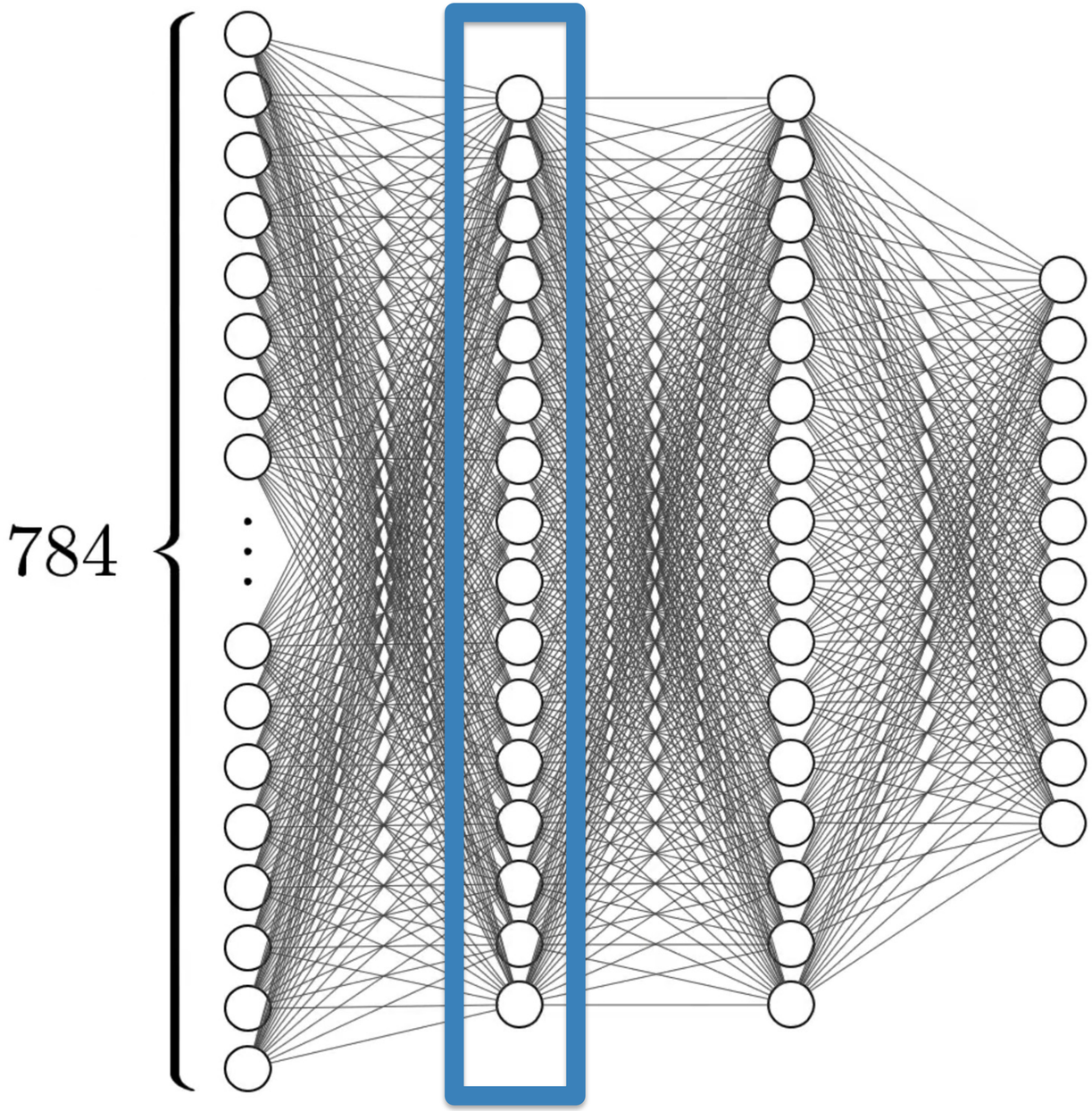
The value of the pixels are concatenated in a vector.

The 784 numbers represent the input layer to our network

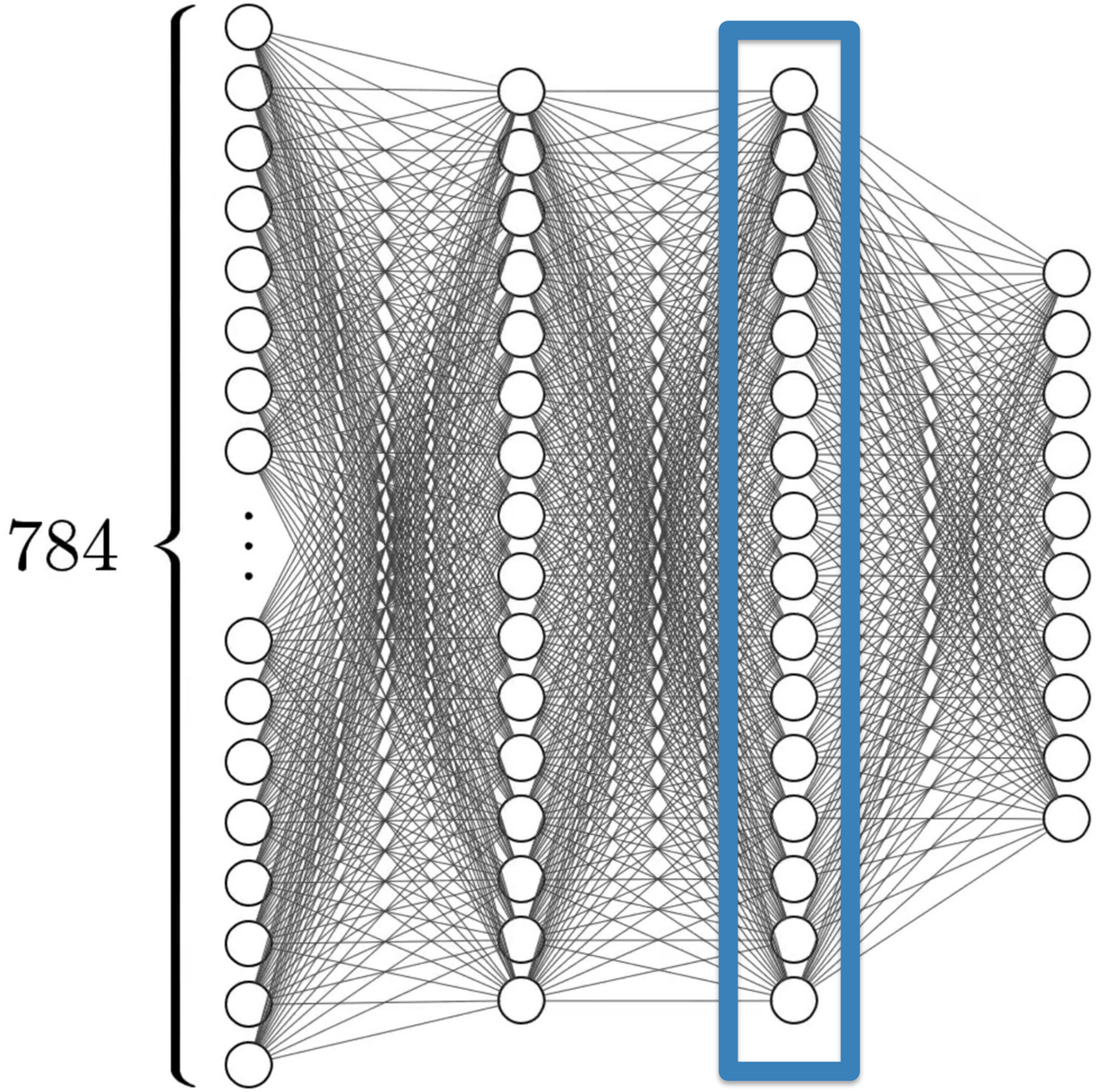

784



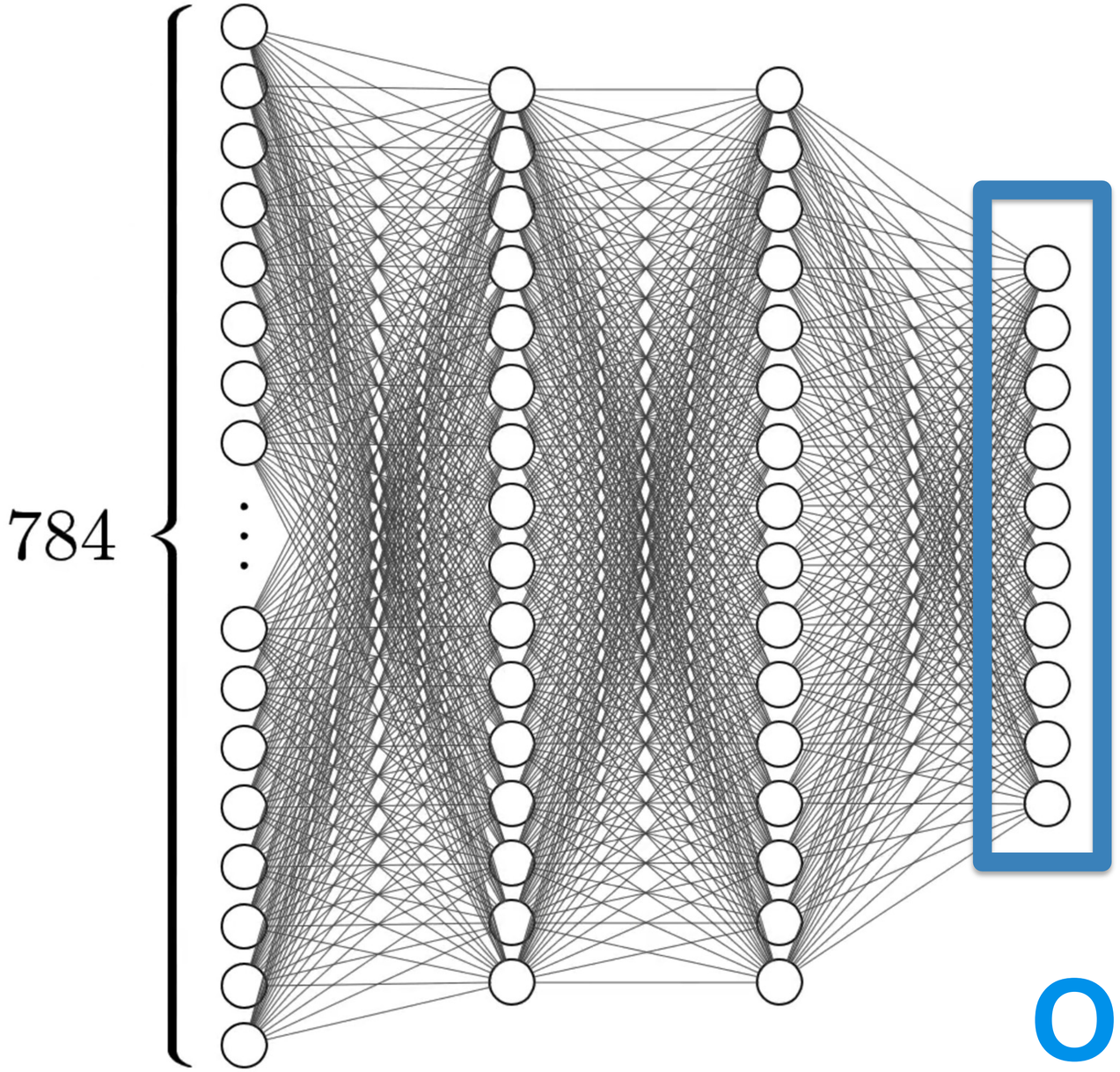
Input Layer



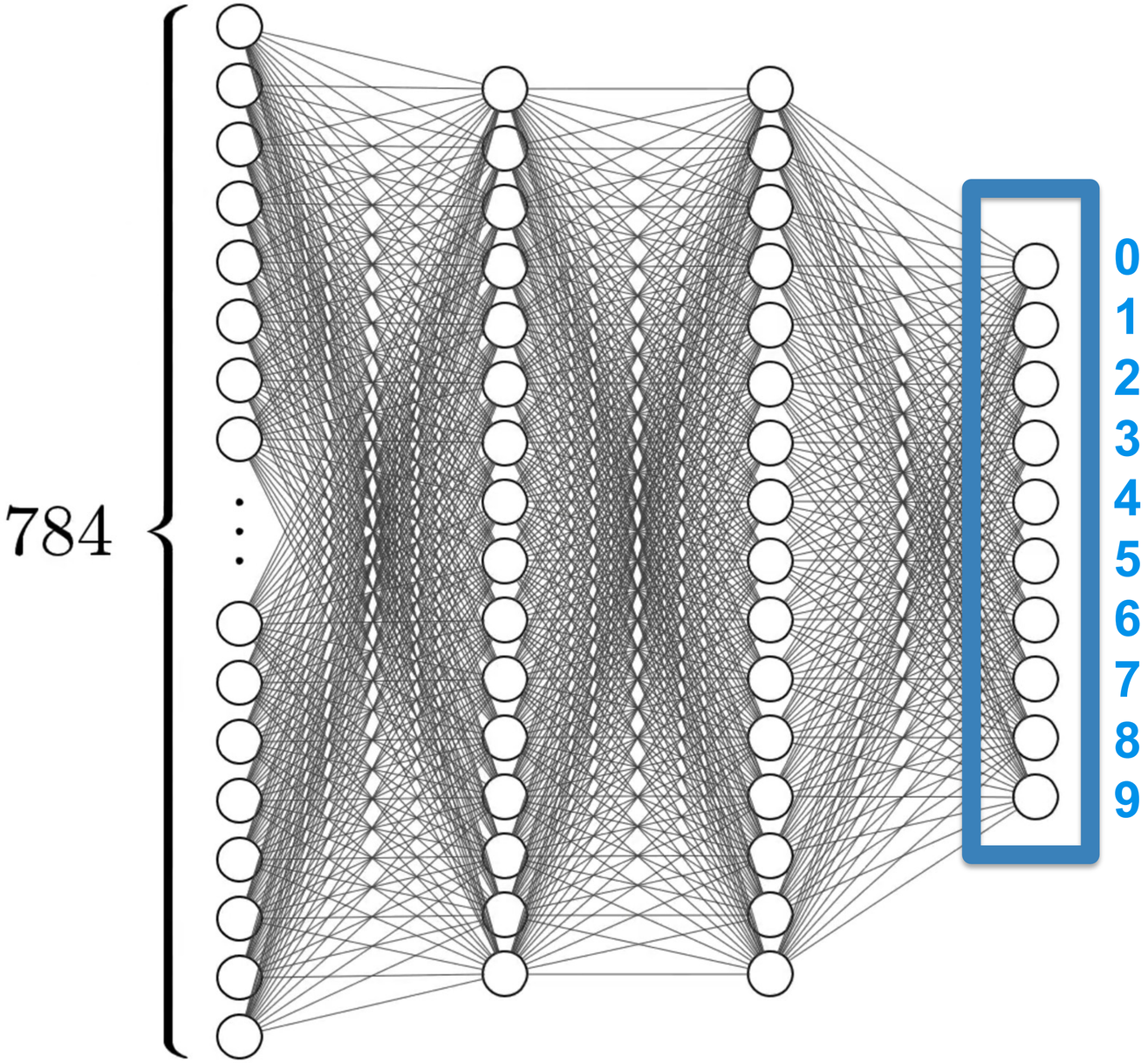
Hidden Layer 1



Hidden Layer 2

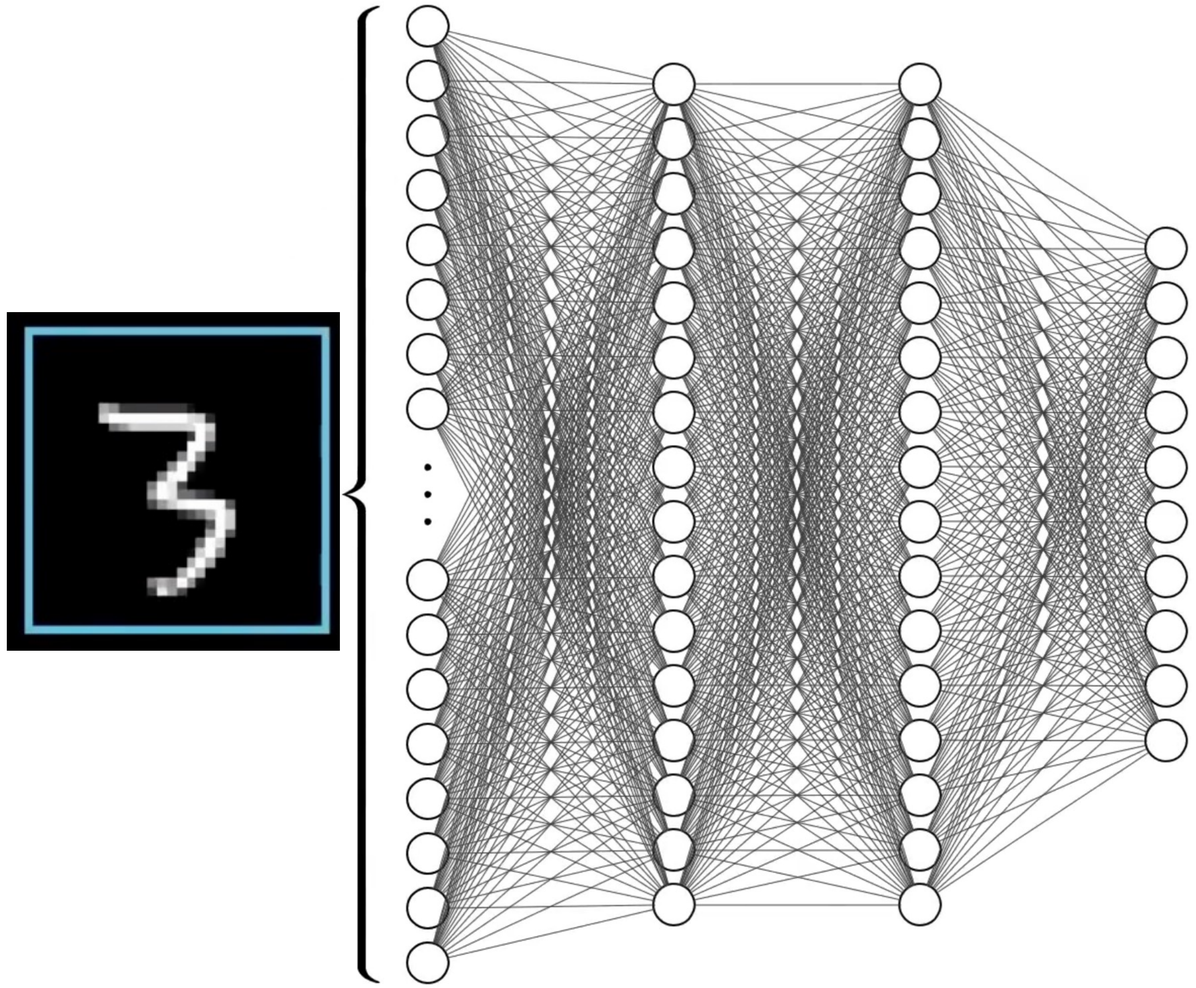


Output Layer



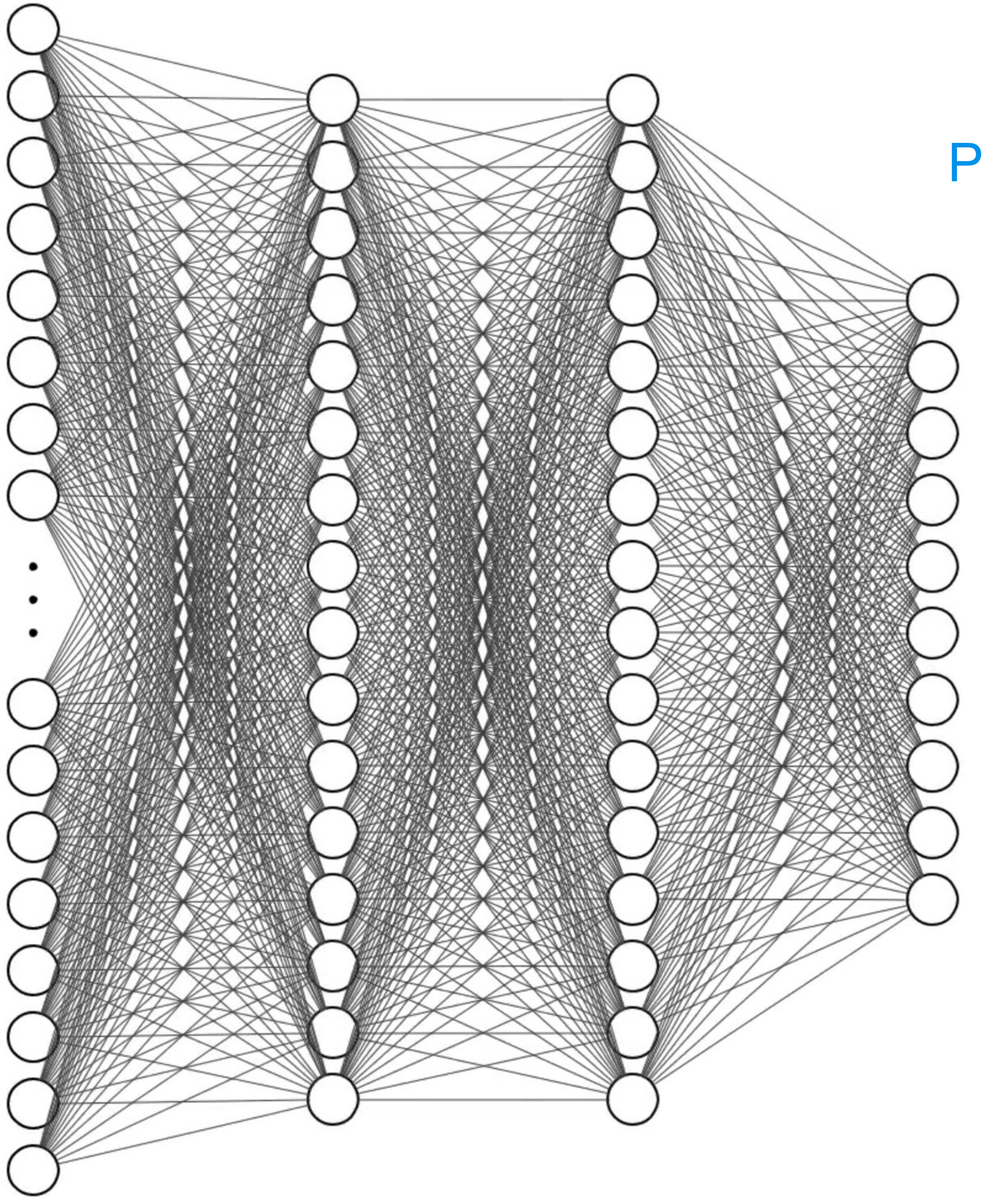
- Each node in the output layer corresponds to a label in the dataset

Training



At the beginning of the training process, the weights are randomly initialised

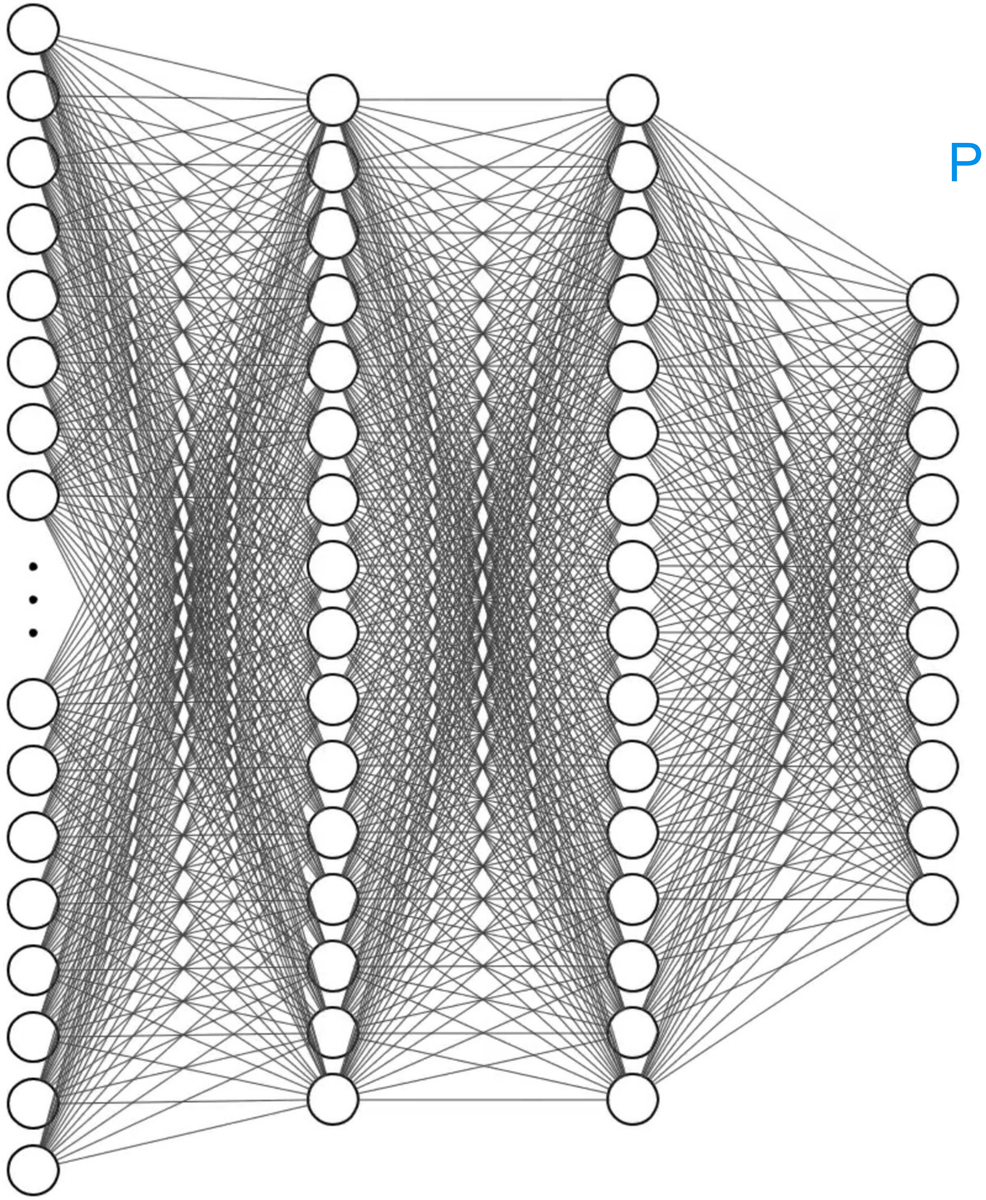
Cost function



Predicted

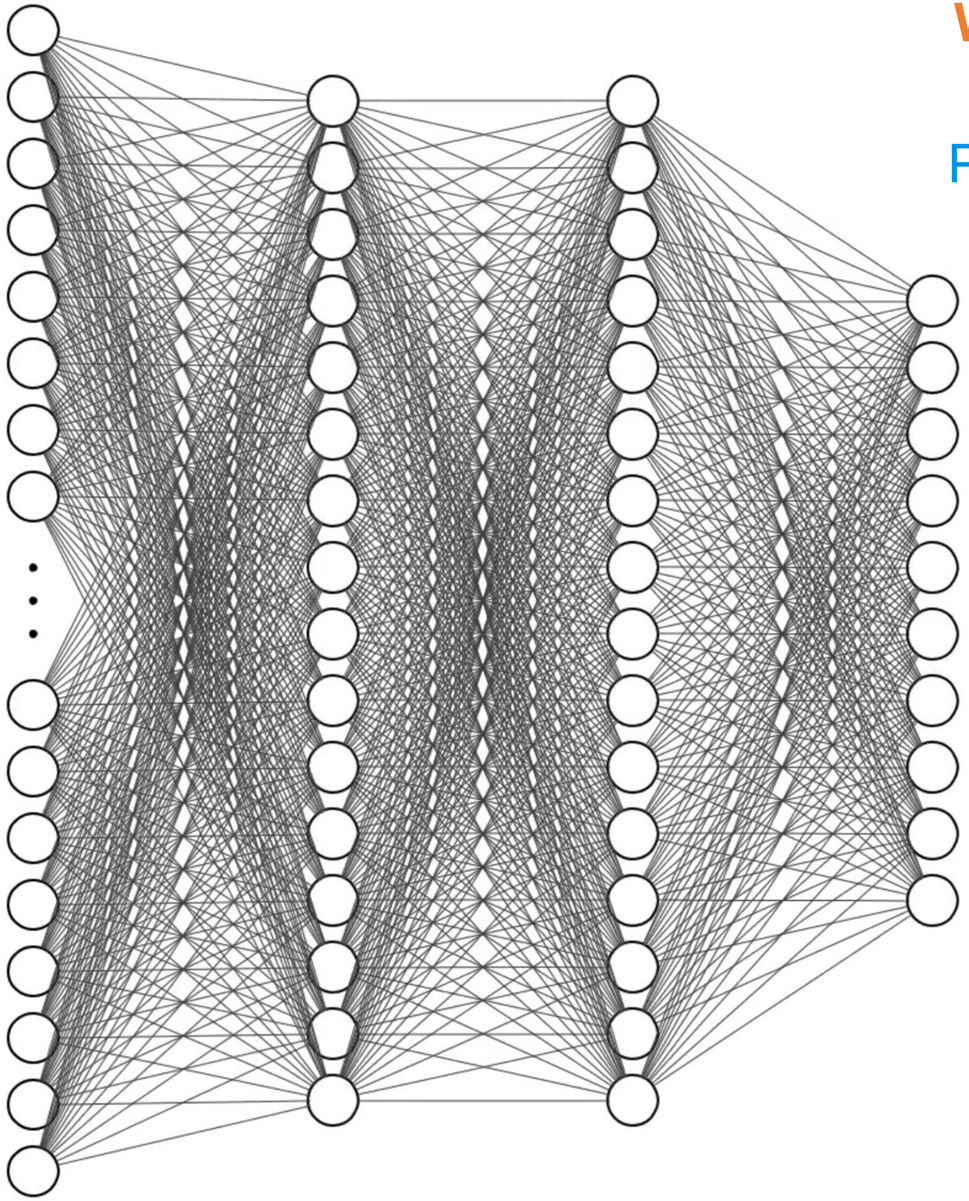
0.1
0.2
0.5
0.7
0.3
0.9
0.55
0.2
0.5
0.1

Cost function



Predicted	Ground truth values
0.1	0
0.2	0
0.5	0
0.7	1
0.3	0
0.9	0
0.55	0
0.2	0
0.5	0
0.1	0

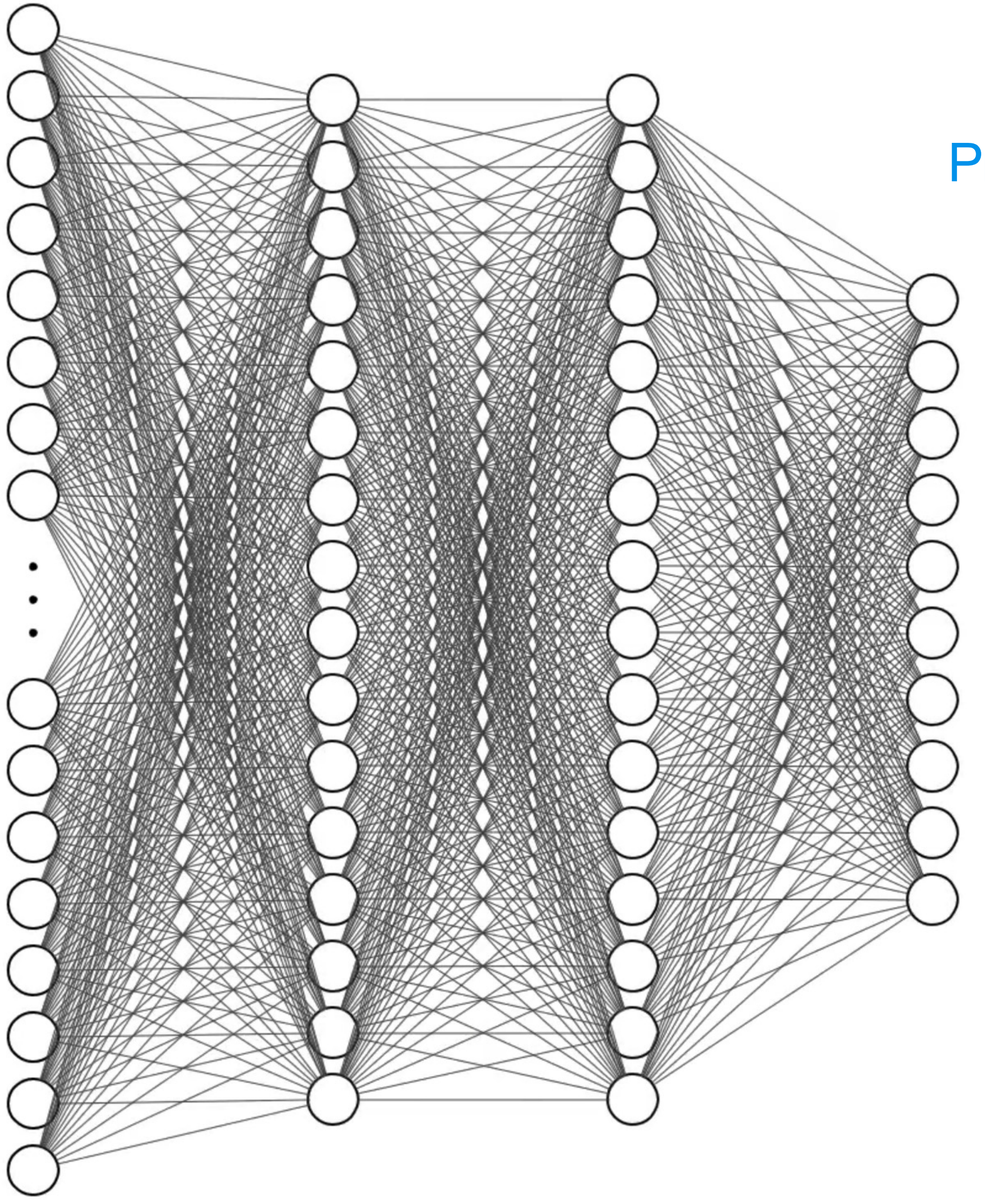
Cost function



What is the cost difference?

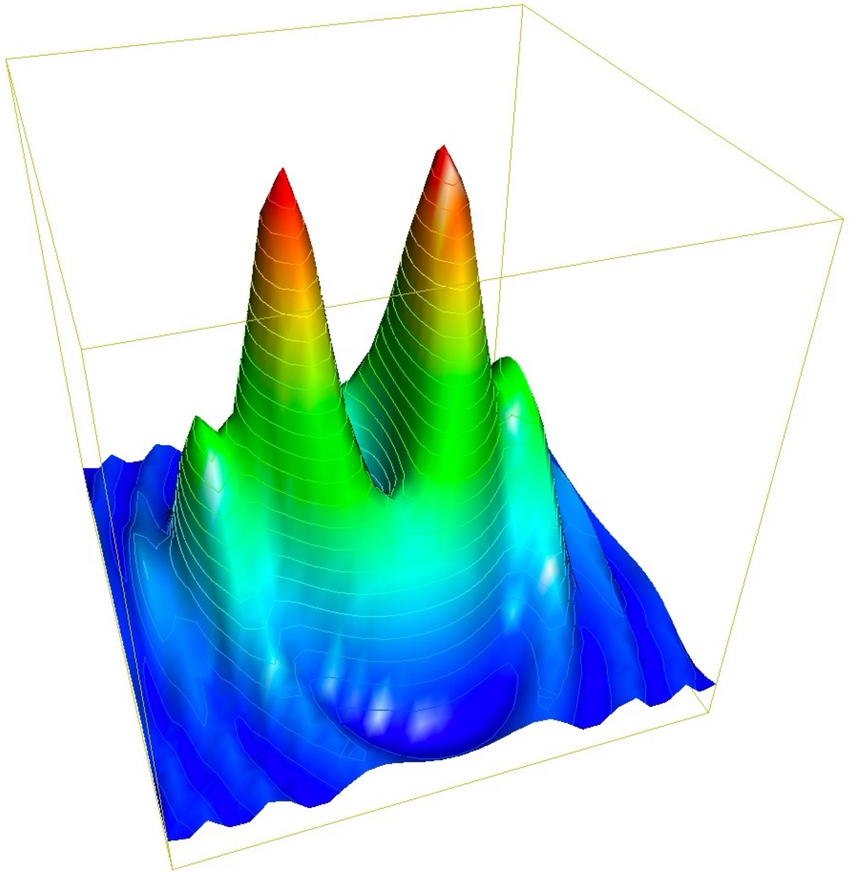
Predicted	Ground truth values
0.1	0
0.2	0
0.5	0
0.7	1
0.3	0
0.9	0
0.55	0
0.2	0
0.5	0
0.1	0

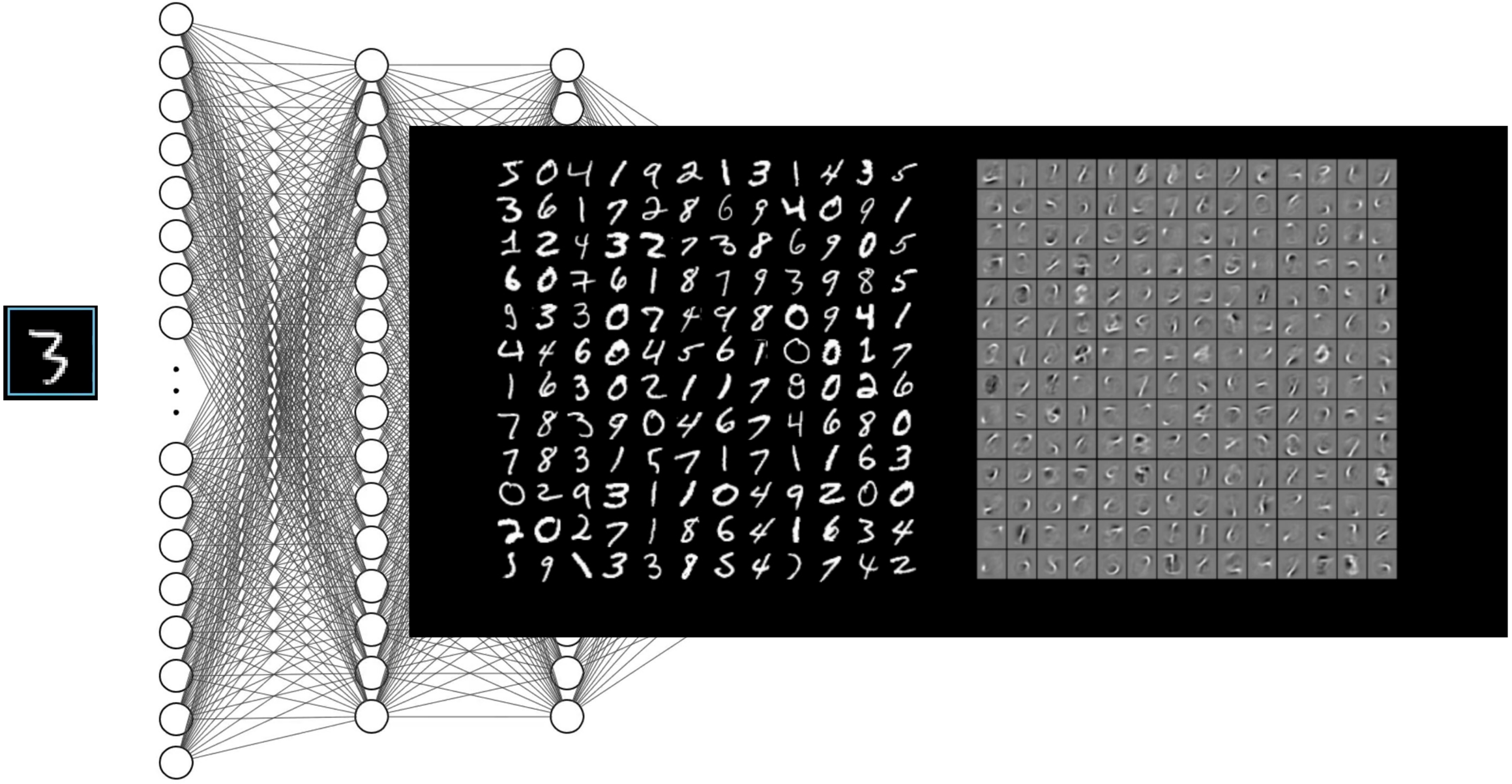
Cost function



Predicted	Ground truth values	Cost
0.1	0	$(0.1 - 0)^2 +$
0.2	0	$(0.2 - 0)^2 +$
0.5	0	$(0.5 - 0)^2 +$
0.7	1	$(0.7 - 1)^2 +$
0.3	0	$(0.3 - 0)^2 +$
0.9	0	$(0.9 - 0)^2 +$
0.55	0	$(0.55 - 0)^2 +$
0.2	0	$(0.2 - 0)^2 +$
0.5	0	$(0.5 - 0)^2 +$
0.1	0	$(0.1 - 0)^2$

Gradient descent





Deep learning

How many layers to transform a neural network into **deep?**

How many layers to transform a neural network into **deep?**

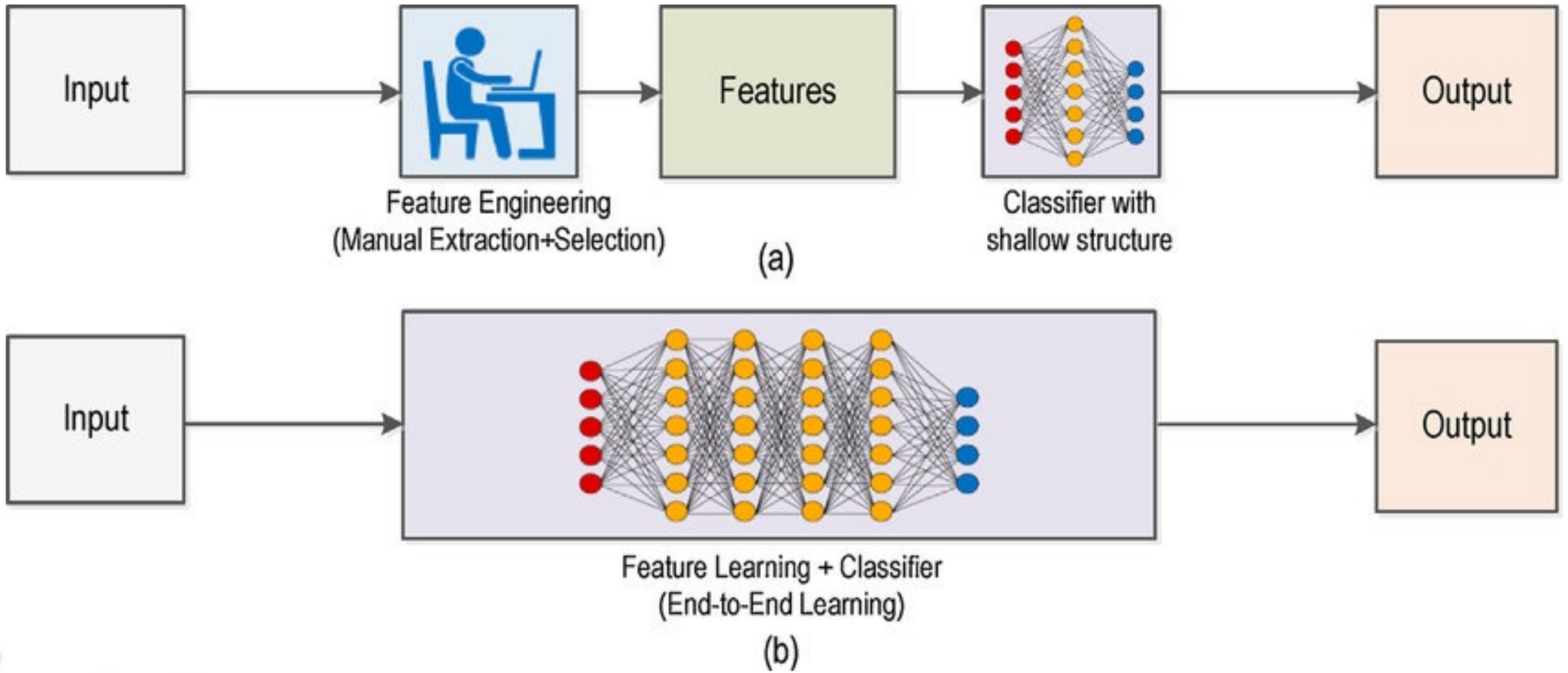
One of the earliest deep neural networks has **three** densely connected hidden layers (Hinton et al. (2006)).

How many layers to transform a neural network into **deep?**

One of the earliest deep neural networks has **three** densely connected hidden layers (Hinton et al. (2006)).

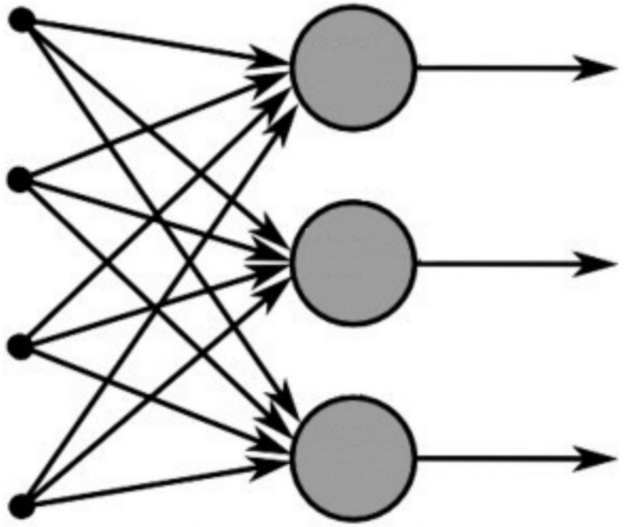
There is no universally agreed upon threshold of depth dividing shallow learning from deep learning

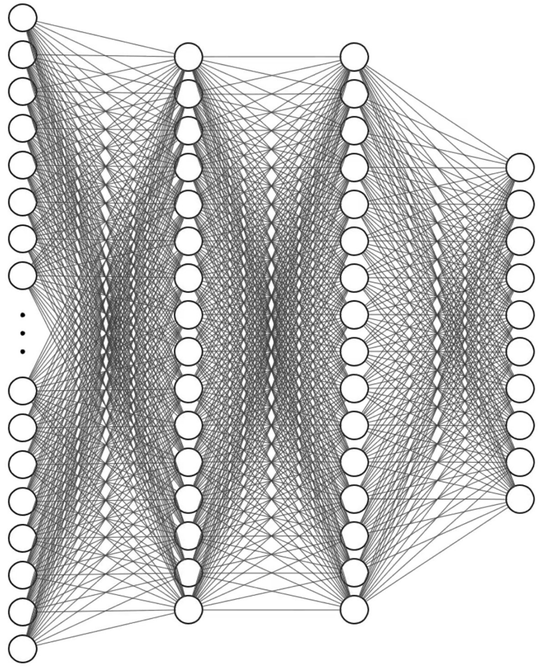
Deep Learning vs Other Machine Learning Techniques



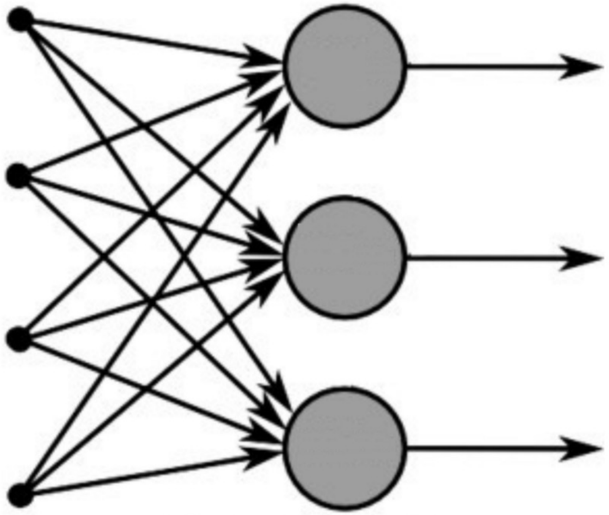
Wang, Jinjiang, et al. "Deep learning for smart manufacturing: Methods and applications." *Journal of Manufacturing Systems* (2018).

Examples of Neural Networks: Feed forward neural network



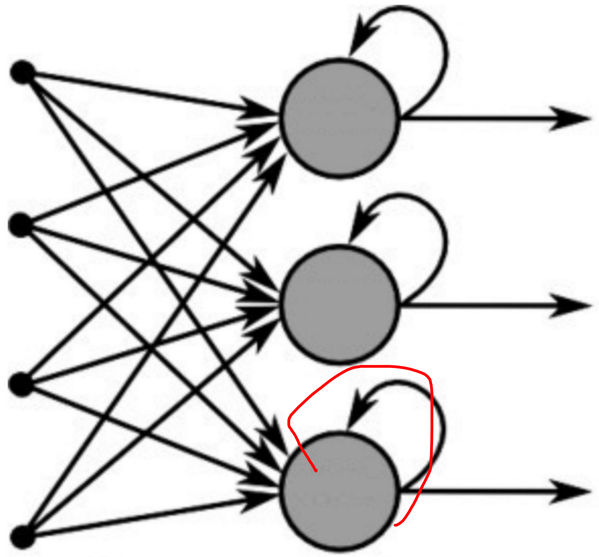


Examples of Neural Networks
Recurrent Neural Networks



Feed-Forward Neural Network

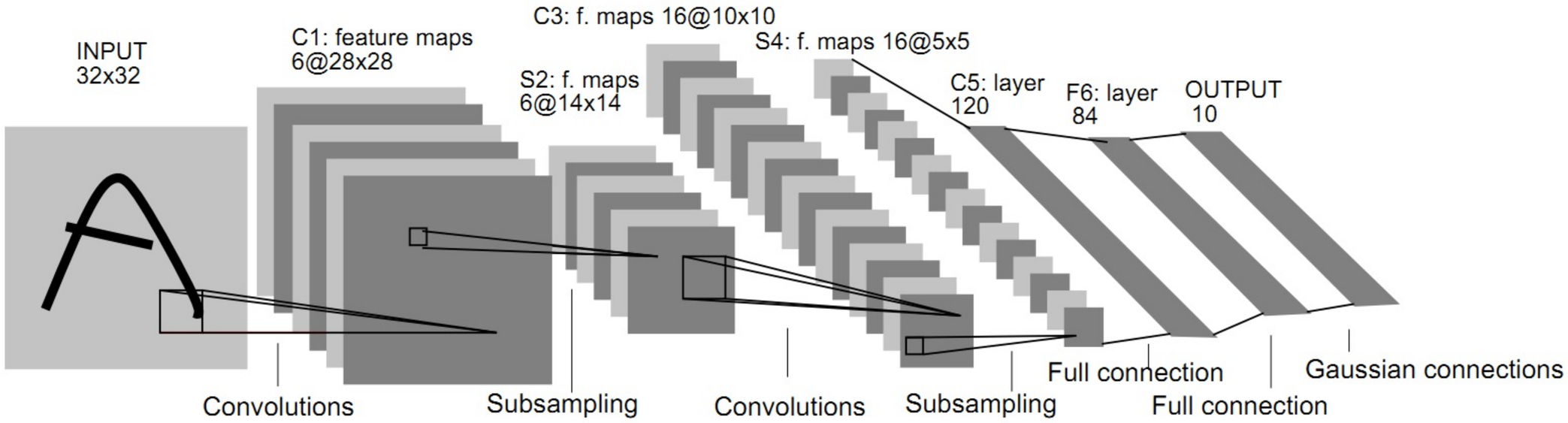
vs.



Recurrent Neural Network

Recurrent Neural Networks -> good for speech recognition

Examples of Neural Networks: Convolutional Neural Networks

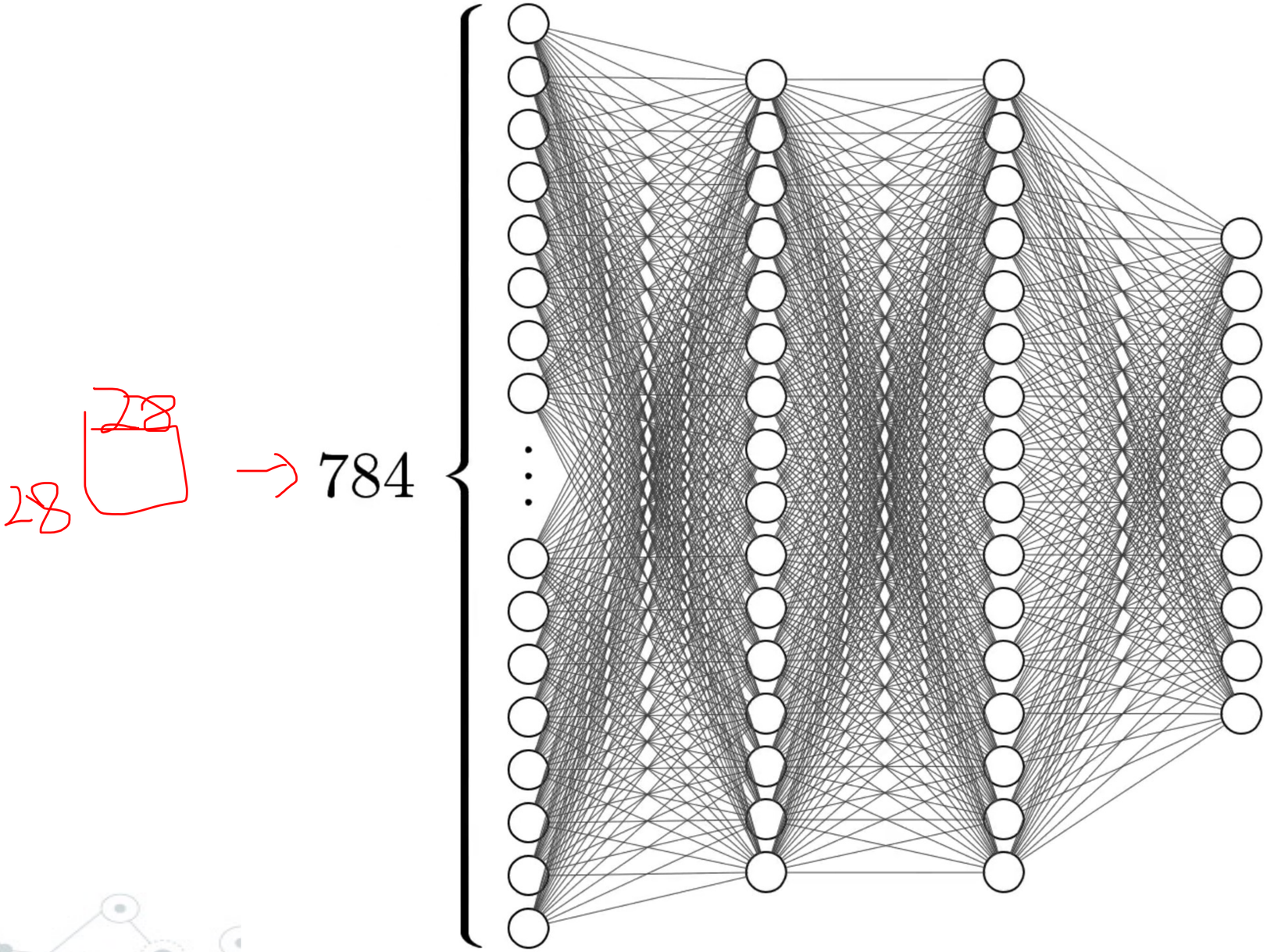


Convolutional Neural Networks -> good for image recognition

http://yann.lecun.com/exdb/lenet/

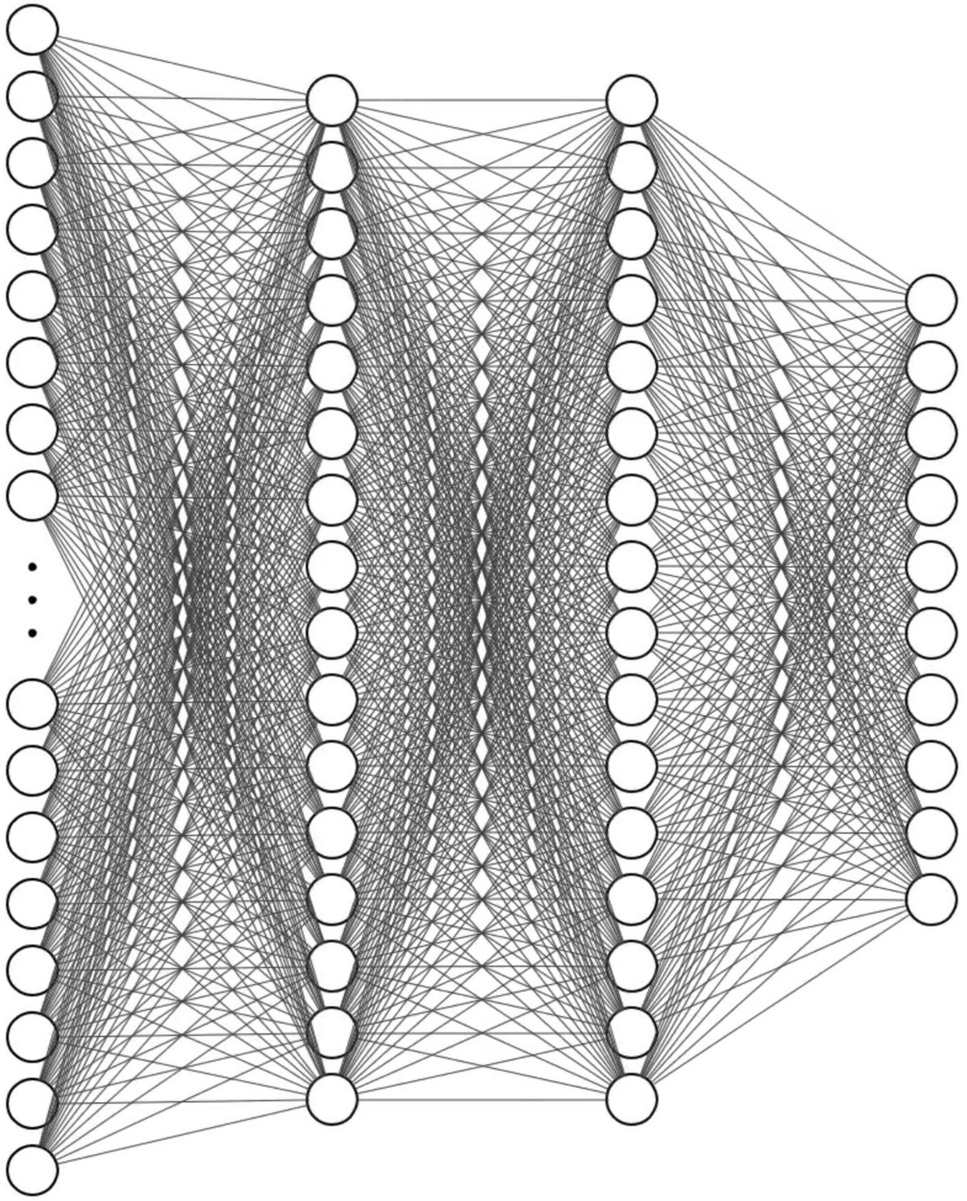
Image classification with

Convolutional Neural Networks


28
28
784

Why do we need convolutions?

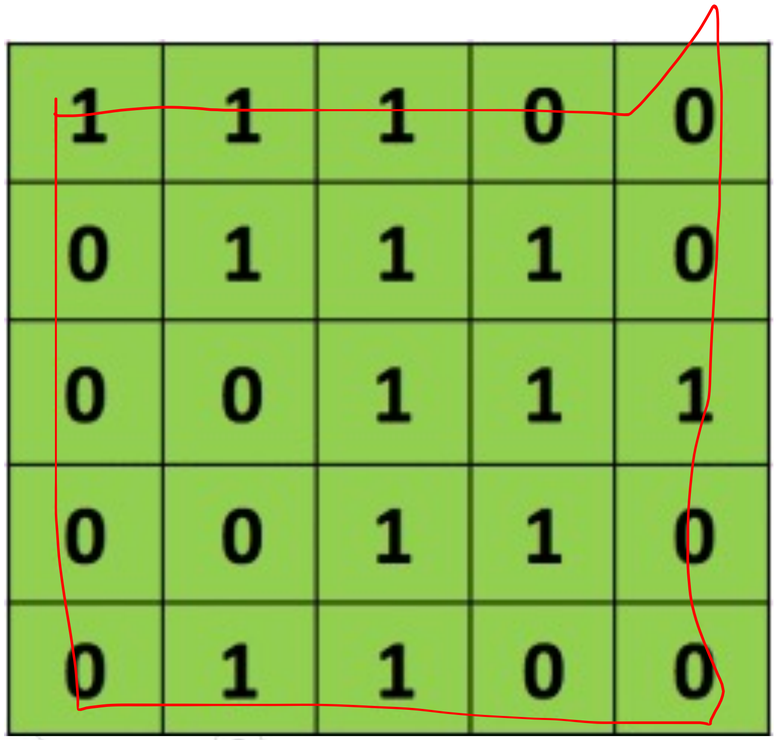
- Even for small images, the number of a fully connected network is computationally expensive
- Here comes in play the convolution and pooling processes



Brian Amberg derivative work: Tinos (talk)

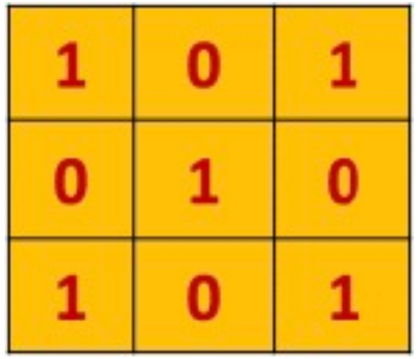
What is a convolution?

Original Image

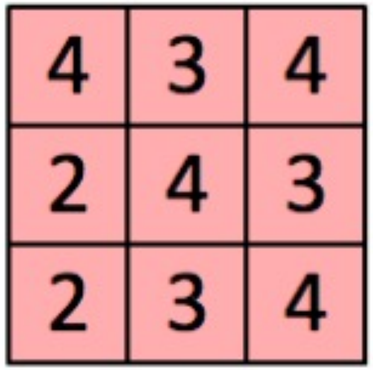


*

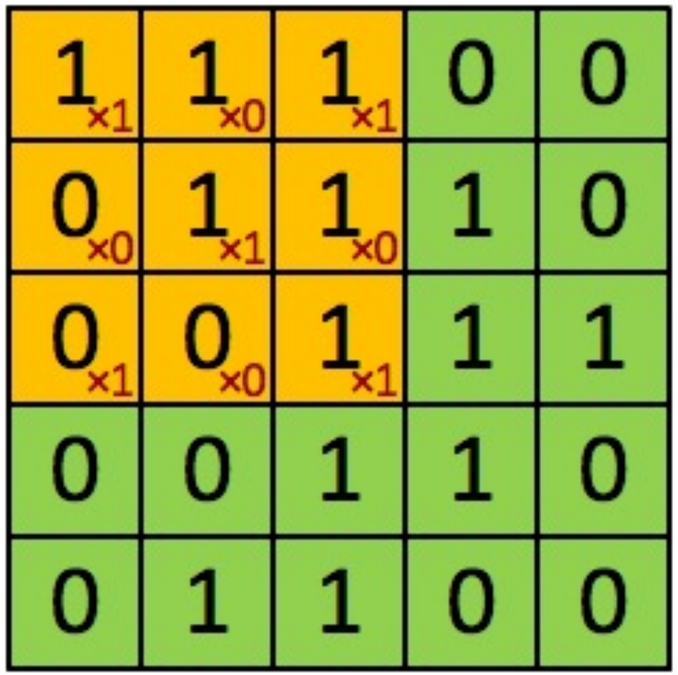
Filter (or Kernel)

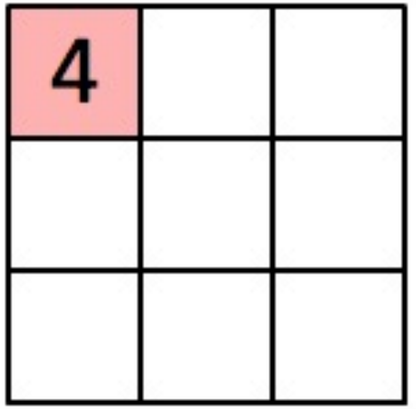


Feature Map (Activation Map; Convolved feature)



What is a convolution?



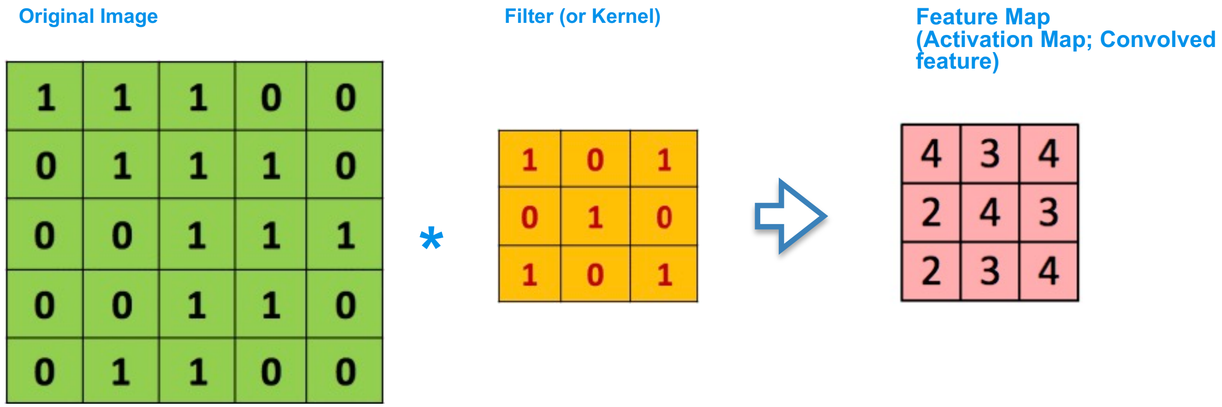


http://deeplearning.stanford.edu/wiki/index.php/File:Convolution_schematic.gif

Image

Convolved Feature

Convolution & training



- Think of the Filter (Kernel) as a pattern detector!
- When training the CNN we will learn the parameters of these filters
- For a filter of size 3*3, we have to learn 10 parameters (9 weights + 1 bias)
- You have to define the Filter size and the Step size

W1	W2	W3
W4	W5	W6
W7	W8	W9

What is a convolution?

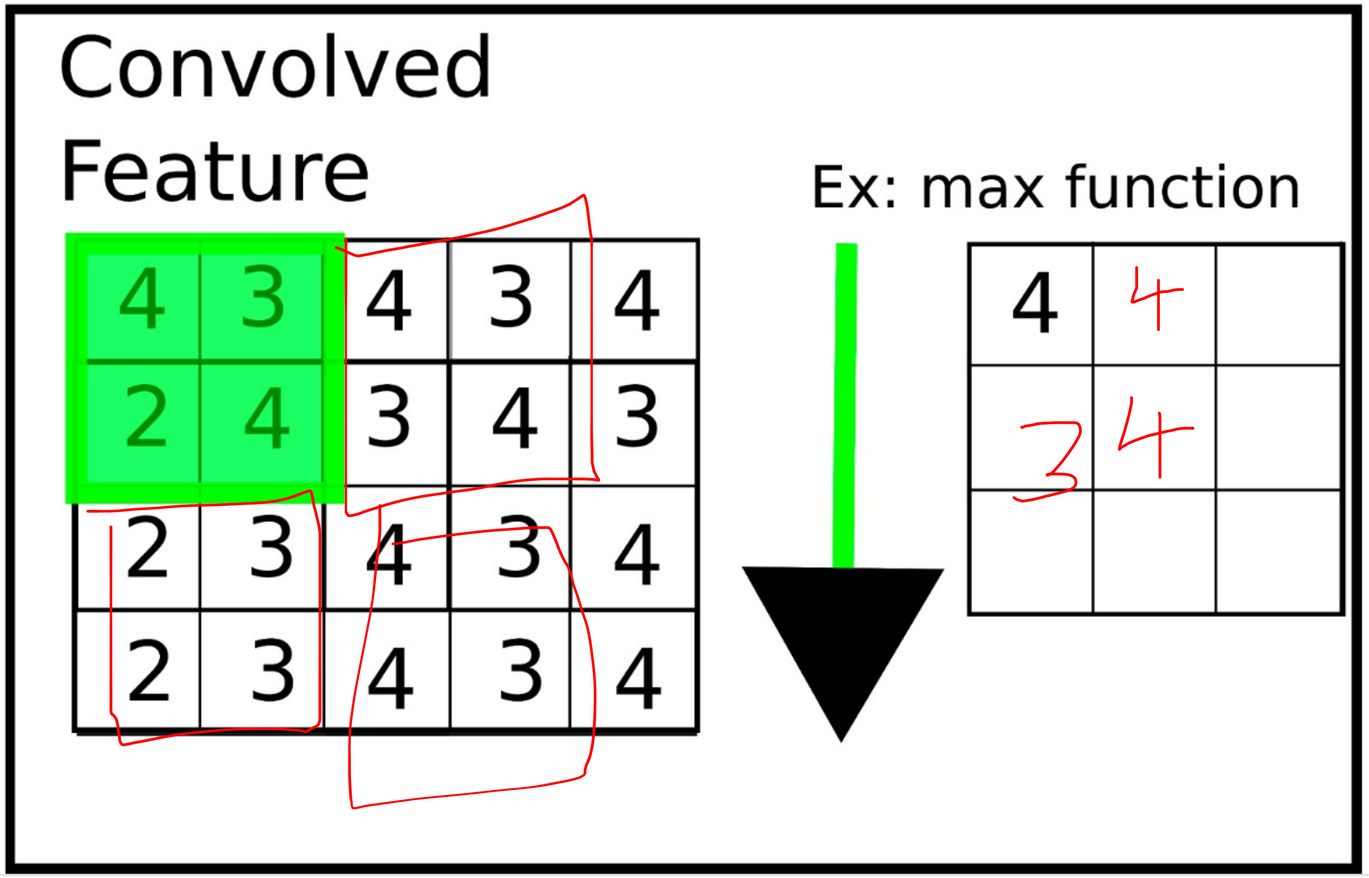
https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Pooling/subsampling?

- Pooling is a way of sub-sampling, meaning reducing the dimension of the input
- The role of pooling is to reduce the resolution of the feature map, while retaining the important features for classification
- The pooling will have a pooling size, step and type of operator
- Operators can be different, the most popular are max or average

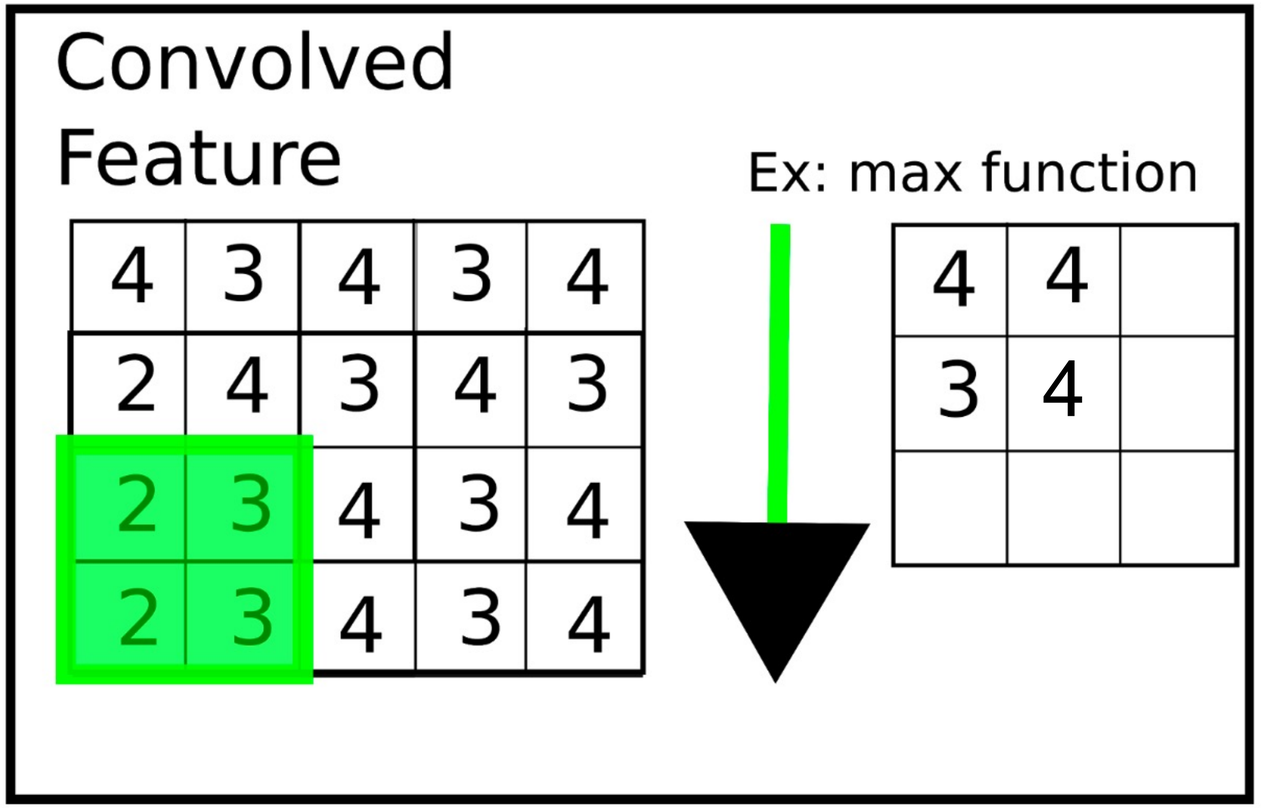
Pooling/subsampling : max pooling

- Filter size: 2x2
- Step: 2
- Operator: max



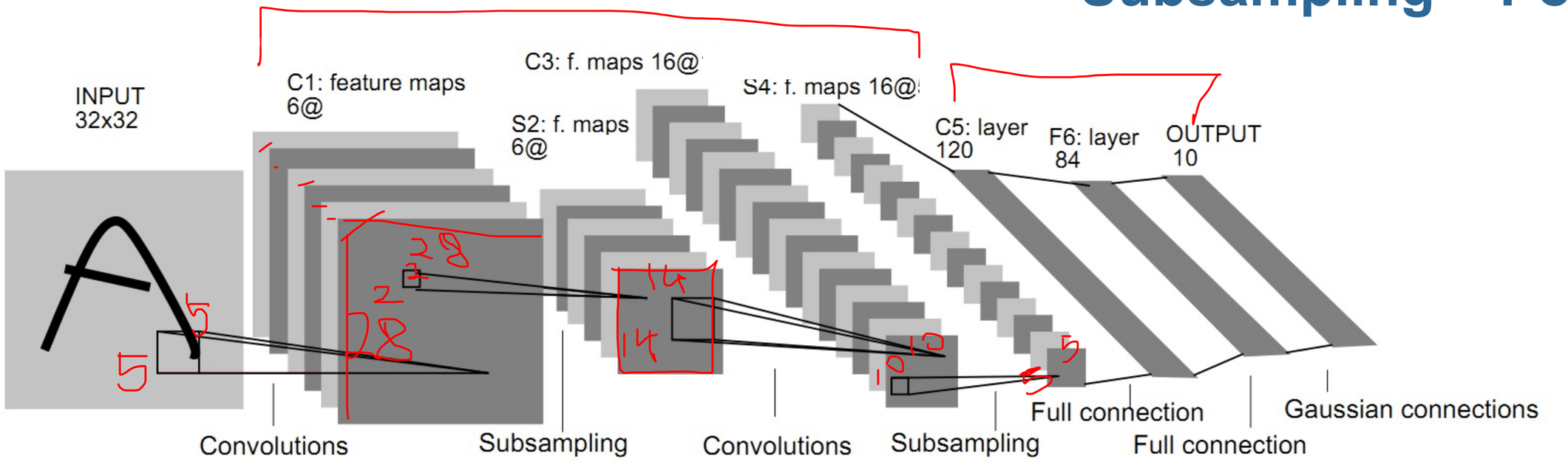
Pooling/subsampling : max pooling

- Filter size: 2x2
- Step: 2
- Operator: max



Examples of Neural Networks: Convolutional Neural Networks

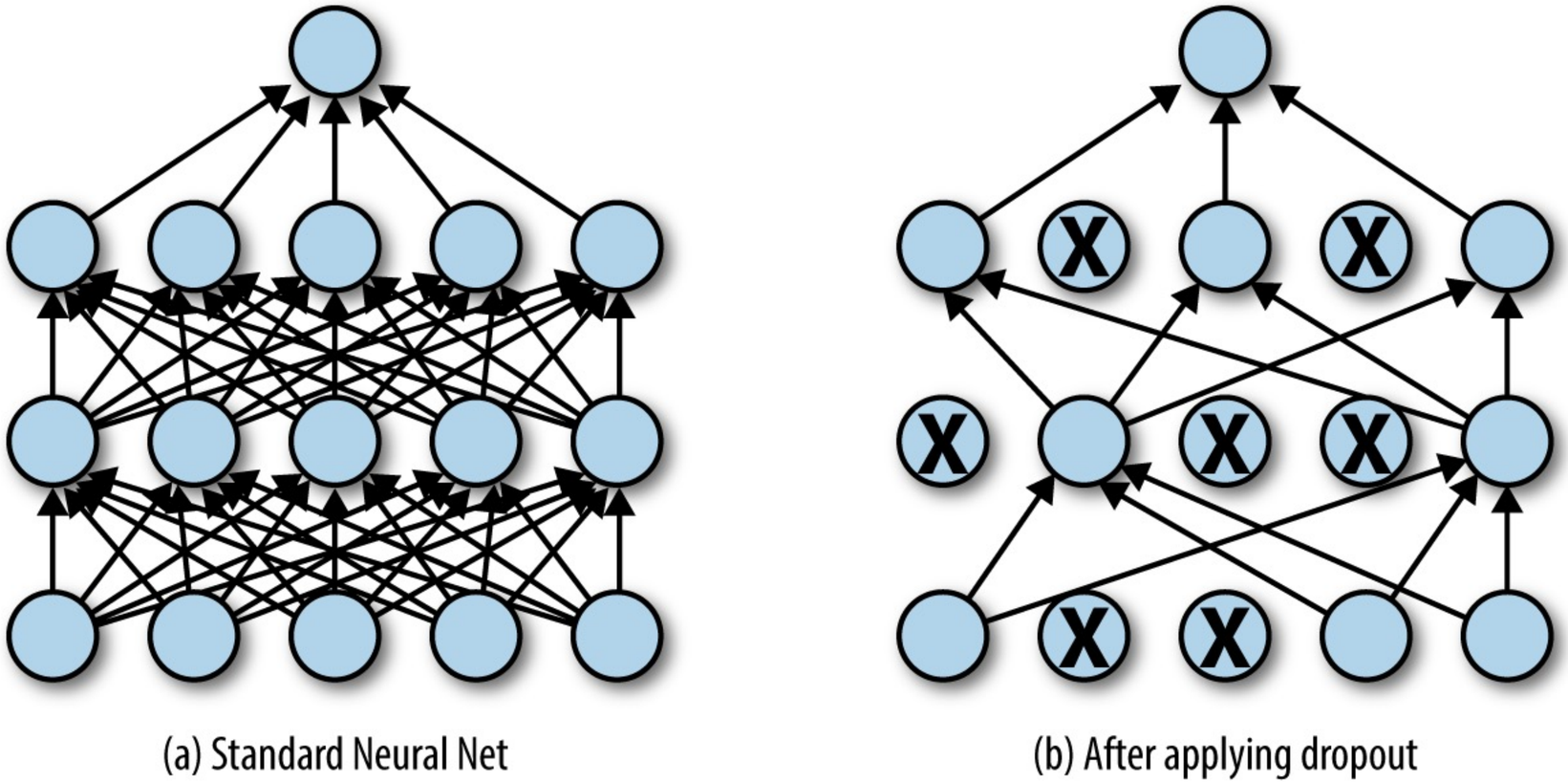
Subsampling = Pooling



Convolutional Neural Networks -> good for image recognition

http://yann.lecun.com/exdb/lenet/

Dropout?

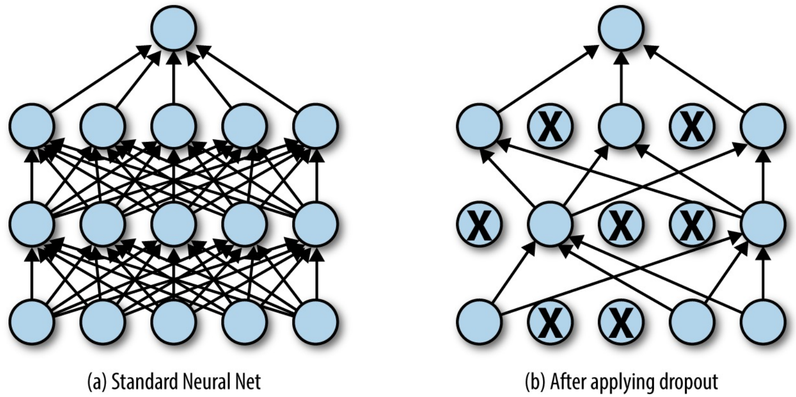


(a) Standard Neural Net

(b) After applying dropout

Ramsundar, Bharath, and Reza Bosagh Zadeh. *TensorFlow for deep learning: from linear regression to reinforcement learning*. " O'Reilly Media, Inc.", 2018.

Dropout?



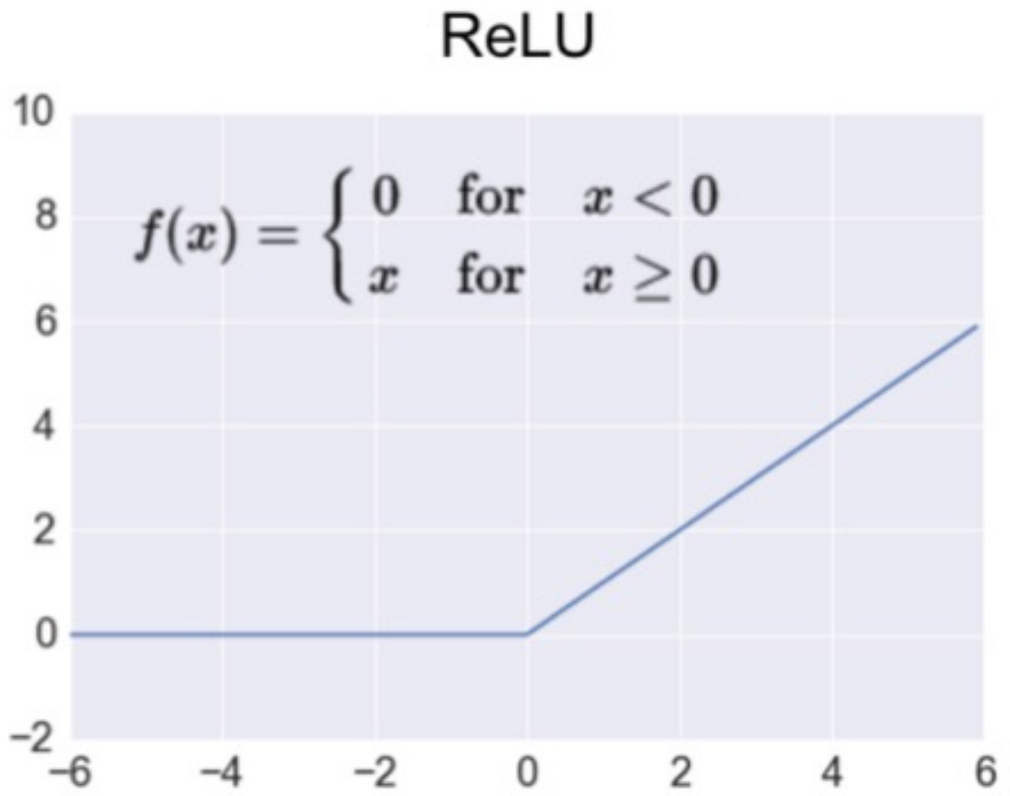
(a) Standard Neural Net

(b) After applying dropout

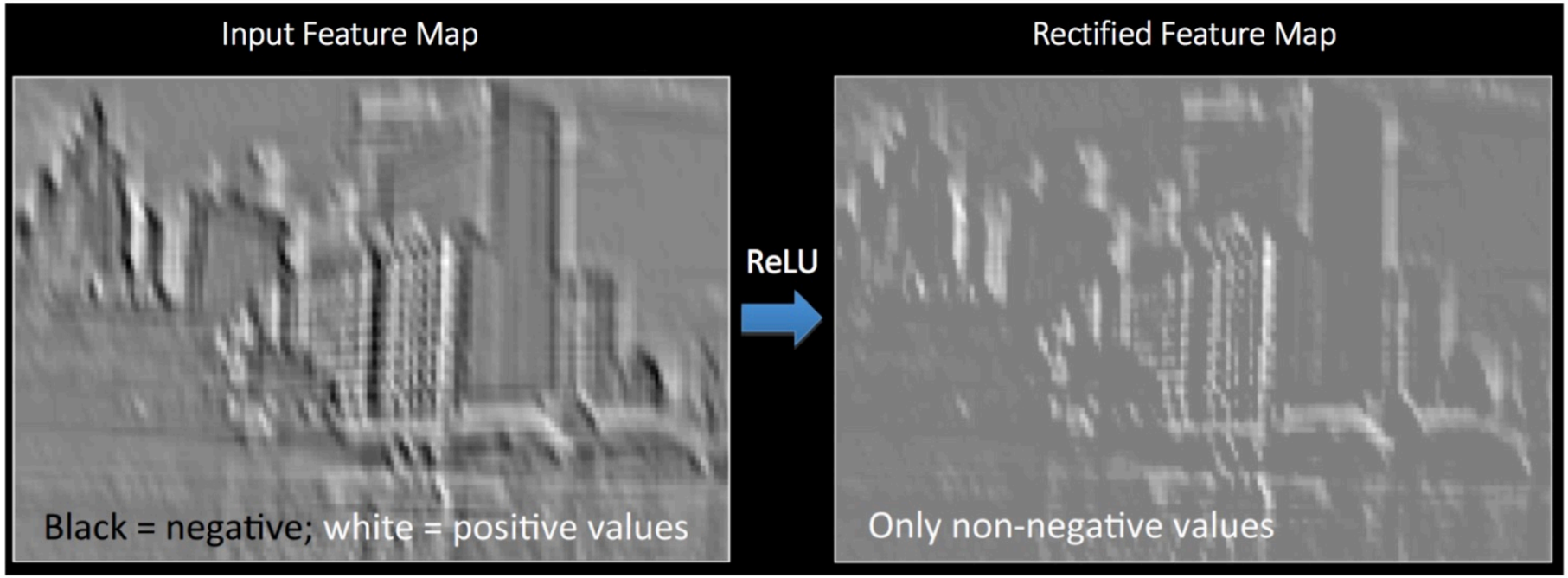
- Dropping a node means that we consider its activation 0
- It prevents the network memorising the training data (i.e. overfitting the data)
- It will help the network generalising better
- It is used only in training! When making predictions dropout is turned off.

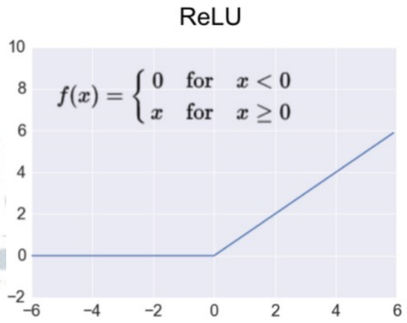
Ramsundar, Bharath, and Reza Bosagh Zadeh. *TensorFlow for deep learning: from linear regression to reinforcement learning*. " O'Reilly Media, Inc.", 2018.

Activation function:RELU function



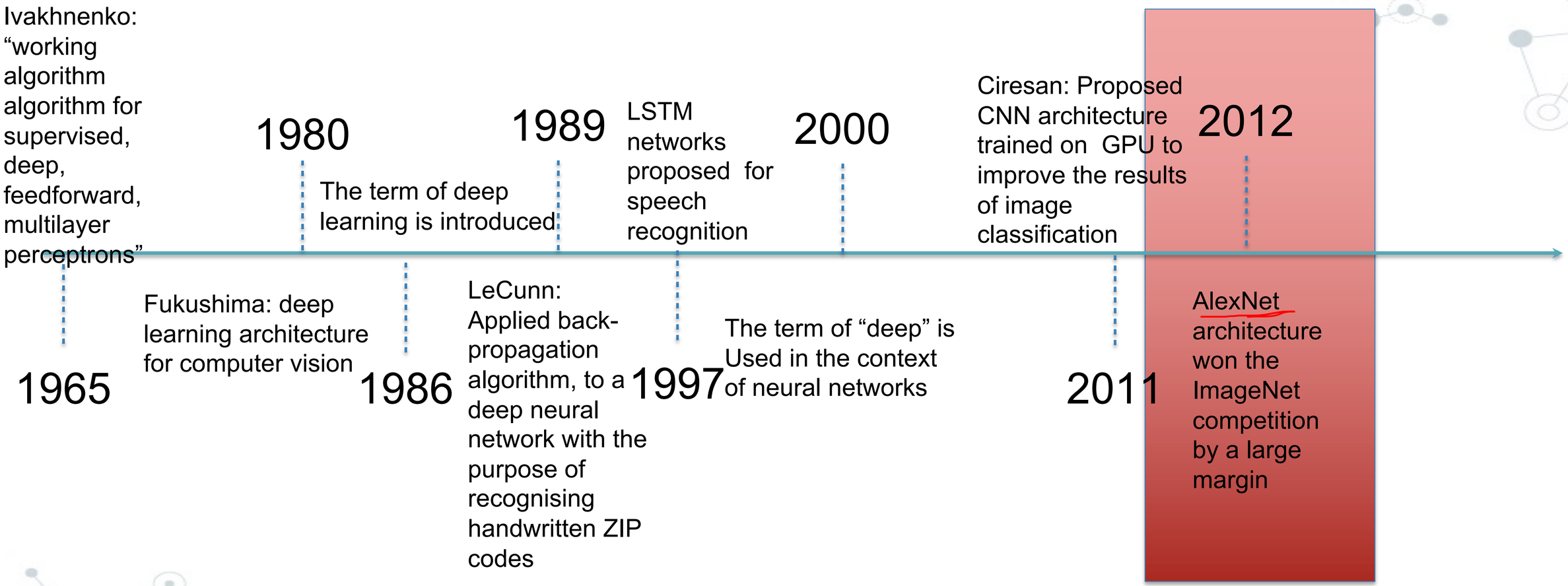
Activation function:RELU function



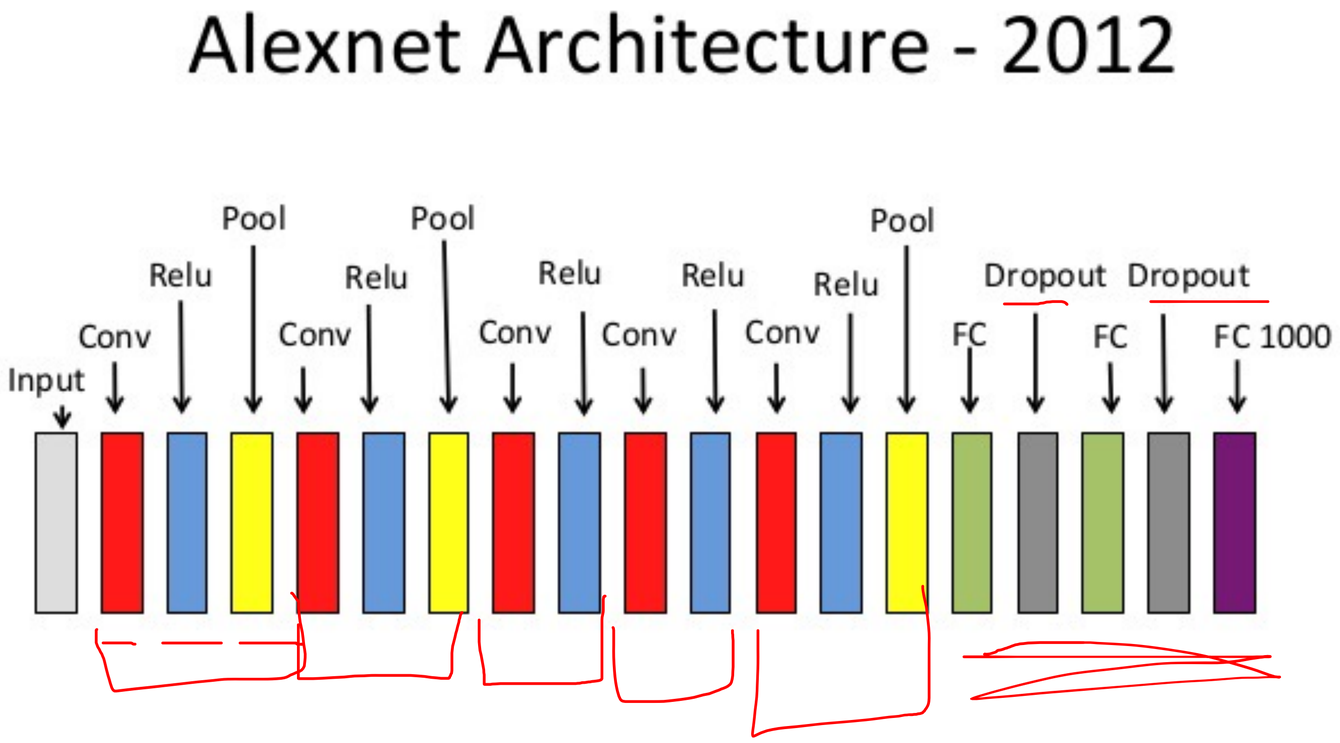


http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

A short history of deep learning



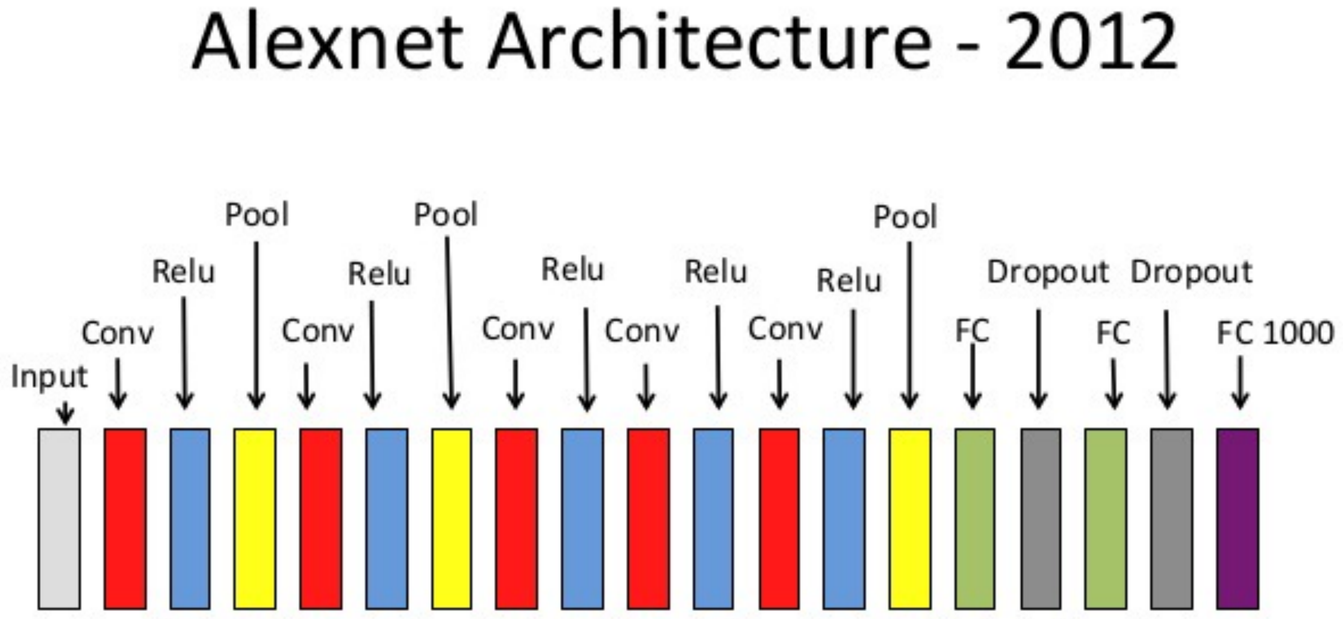
The network that showed the power of convolutions: AlexNet 2012



ImageNet Classification with Deep Convolutional Neural Networks Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton Advances in Neural Information Processing Systems 25 eds.F. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger pp. 1097-1105, 2012

http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

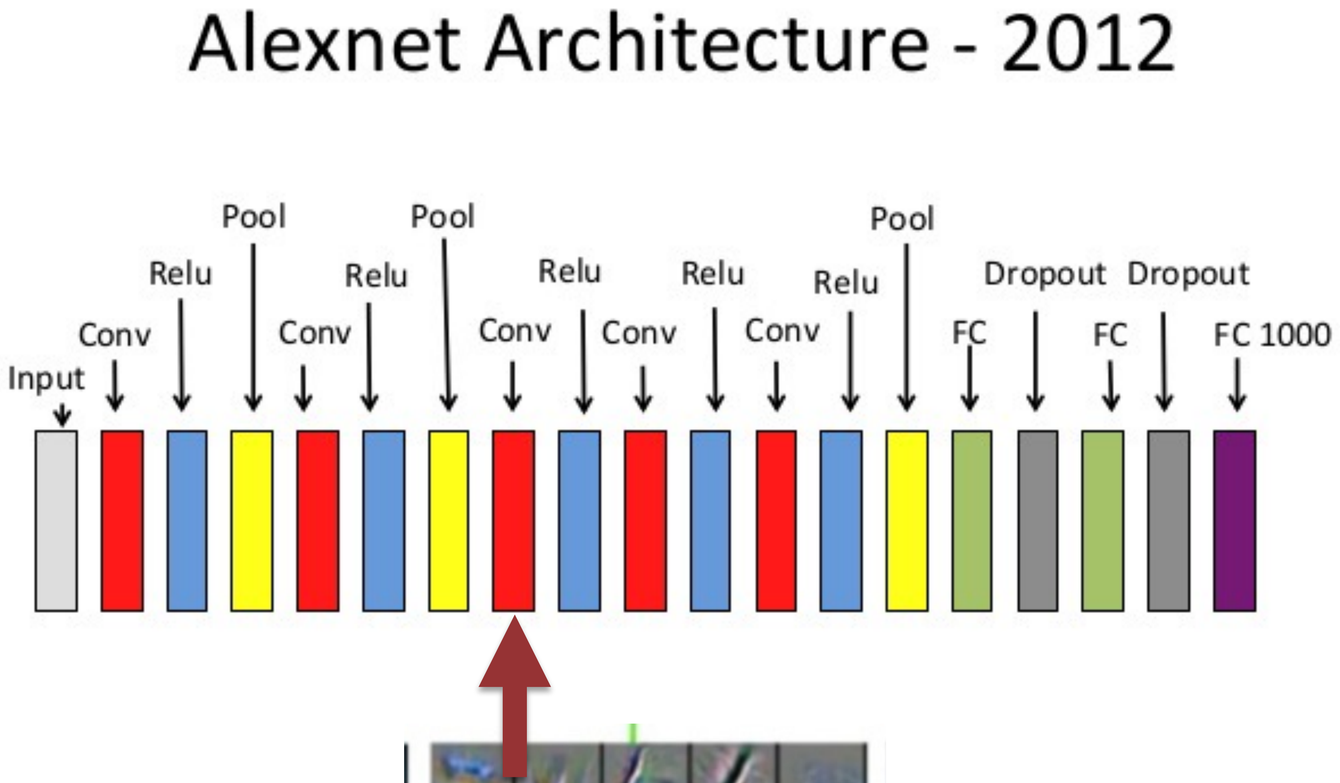
AlexNet 2012



utional Neural Networks Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton
ng Systems 25 eds.F. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger pp.
1097-1105, 2012

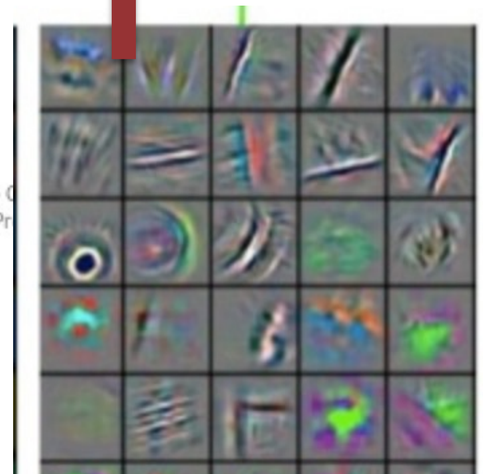
http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

AlexNet 2012



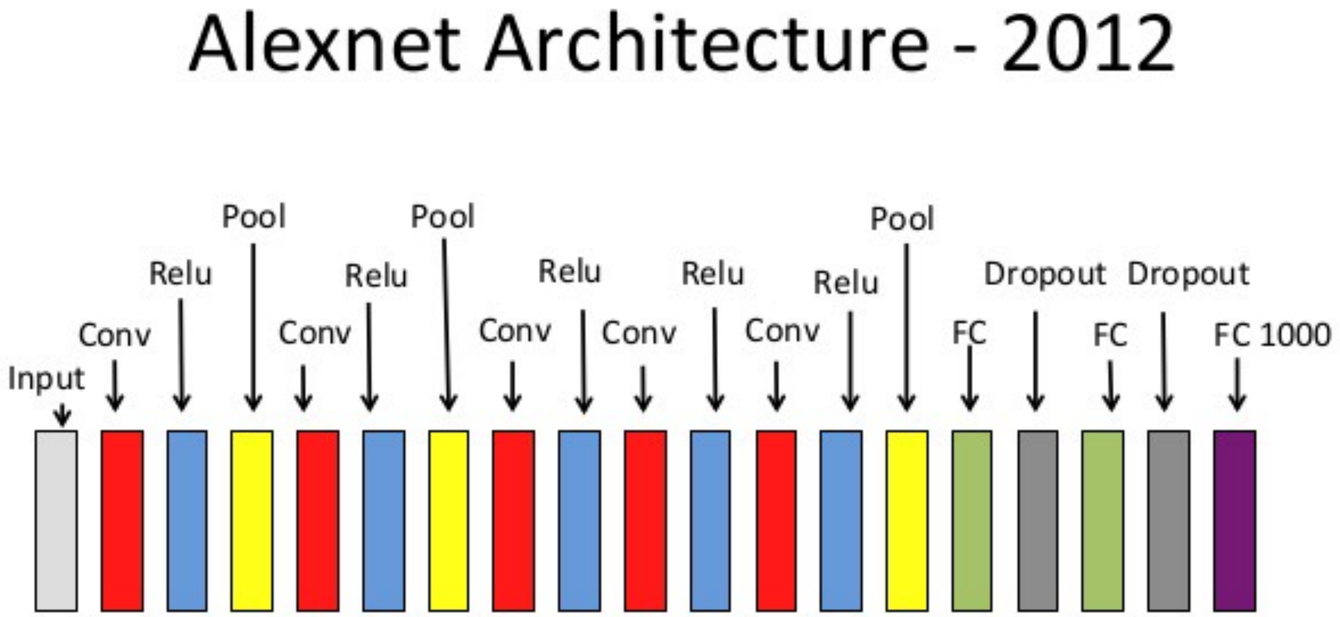
ImageNet Classification with Deep C
Advances in Neural Information Pr

ya Sutskever and Geoffrey E. Hinton
L. Bottou and K.Q. Weinberger pp.



ahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

AlexNet 2012

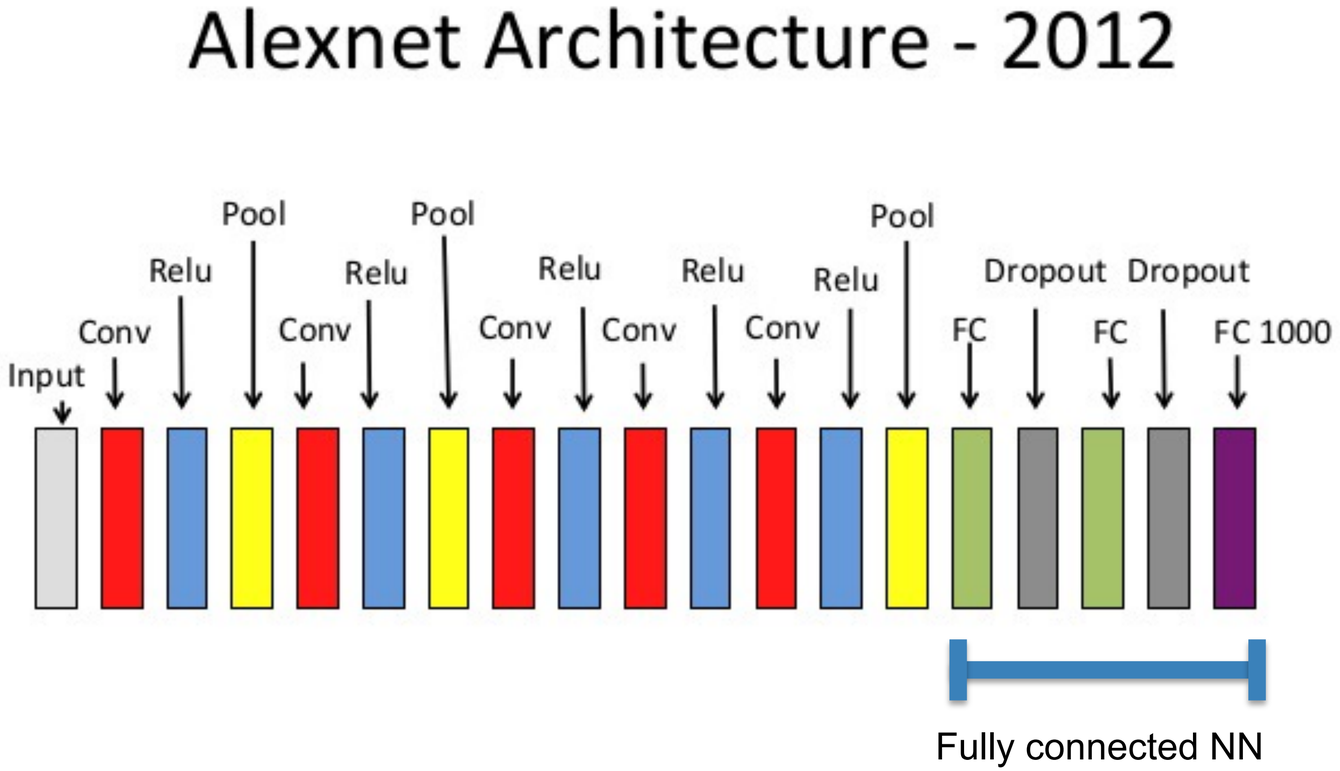


ImageNet Classification with Deep Convoluti
Advances in Neural Information Processing

tskever and Geoffrey E. Hinton
ttou and K.Q. Weinberger pp.

d.com/posts/2016/06/introduction-deep-learning-python-caffe/

AlexNet 2012



ImageNet Classification with Deep Convolutional Neural Networks Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton Advances in Neural Information Processing Systems 25 eds.F. Pereira, C.J.C. Burges, L. Bottou and K.Q. Weinberger pp. 1097-1105, 2012

http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

How many layers can we have?

Deep Residual Learning for Image Recognition

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

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis*

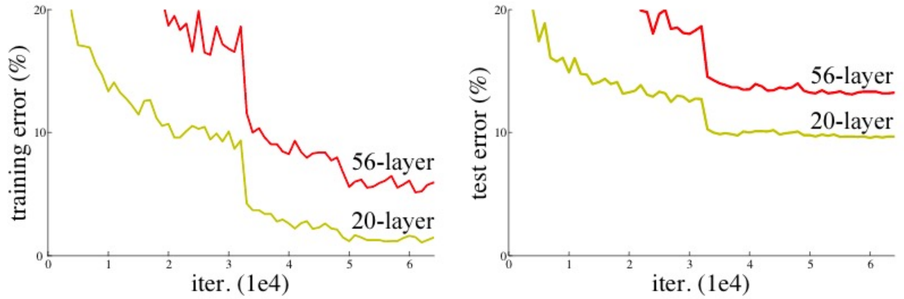


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: *Is learning better networks as easy as stacking more layers?* An obstacle to answering this question was the notorious

How many layers can we have?

Deep Residual Learning for Image Recognition

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

stead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test *set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.*

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative im-

Software libraries

Existing frameworks and libraries

Library	Language(s)	Licence	Author	URL
TensorFlow	Python/C++/Java	Apache 2.0	Google et al.	tensorflow.org
Keras	Python	MIT	François Chollet et al.	keras.io
Torch	Lua/C++	Modified BSD	Idiap Research Institute et al.	torch.ch
PyTorch	Python/C++	Modified BSD	Facebook et al.	pytorch.org
Microsoft Cognitive Toolkit (previously known as CNTK)	Python/C#/C++/Brain Script	MIT	Microsoft	docs.microsoft.com/en-gb/cognitive-toolkit/
Caffe	C++/Python/Matlab	BSD 2-Clause	Berkeley AI Research (BAIR)	caffe.berkeleyvision.org
Apache MXNet	Python/C++/Julia/Closure/R/Scala/Perl	Apache 2.0	Apache Foundation	mxnet.incubator.apache.org
DeepLearning4j	Java	Apache 2.0	Skymind	deeplearning4j.org
Theano	Python	BSD 3-Clause	MILA	deeplearning.net/software/theano
TFLearn	Python	MIT	Aymeric Damien	tflearn.org
Cafe2	Python/C++	Apache 2.0	Facebook	caffe2.ai
Deep Learning Toolbox	Matlab/C++	Proprietary	MatWorks	uk.mathworks.com/products/deep-learning.html
SKLearn	Python	BSD	scikit-learn.org/stable/about.html	scikit-learn.org

Existing frameworks and libraries

Library	Language(s)	Licence	Author	URL
TensorFlow	Python/C++/Java	Apache 2.0	Google et al.	tensorflow.org
Keras	Python	MIT	François Chollet et al.	keras.io
Torch	Lua/C++	Modified BSD	Idiap Research Institute et al.	torch.ch
PyTorch	Python/C++	Modified BSD	Facebook et al.	pytorch.org
Microsoft Cognitive Toolkit (previously known as CNTK)	Python/C#/C++/Brain Script	MIT	Microsoft	docs.microsoft.com/en-gb/cognitive-toolkit/
Caffe	C++/Python/Matlab	BSD 2-Clause	Berkeley AI Research (BAIR)	caffe.berkeleyvision.org
Apache MXNet	Python/C++/Julia/Closure/R/Scala/Perl	Apache 2.0	Apache Foundation	mxnet.incubator.apache.org
DeepLearning4j	Java	Apache 2.0	Skymind	deeplearning4j.org
Theano	Python	BSD 3-Clause	MILA	deeplearning.net/software/theano
TFLearn	Python	MIT	Aymeric Damien	tflearn.org
Cafe2	Python/C++	Apache 2.0	Facebook	caffe2.ai
Deep Learning Toolbox	Matlab/C++	Proprietary	MatWorks	uk.mathworks.com/products/deep-learning.html
SKLearn	Python	BSD	scikit-learn.org/stable/about.html	scikit-learn.org

TensorFlow

- DeepLearning Framework developed by Google
- Widely available and used by companies in production
- Many models (pre-trained or code) available on Github
- Models trained on GPU can be easily queried on CPU
- Tools for distributed training, online parameters tracking, pre-processing, etc.

Deep Learning Advantages

- It works better than other methods for image recognition, speech recognition etc.
- Reduces the need of feature engineering
- The network architectures can be reused for a variety of problems

Deep Learning Disadvantages

- Computational expensive to train
- Requires a LARGE amount of data (with few examples, deep learning is very unlikely to outperform other machine learning methods)
- Sometimes it is a black box, difficult to debug and understand what is actually going on

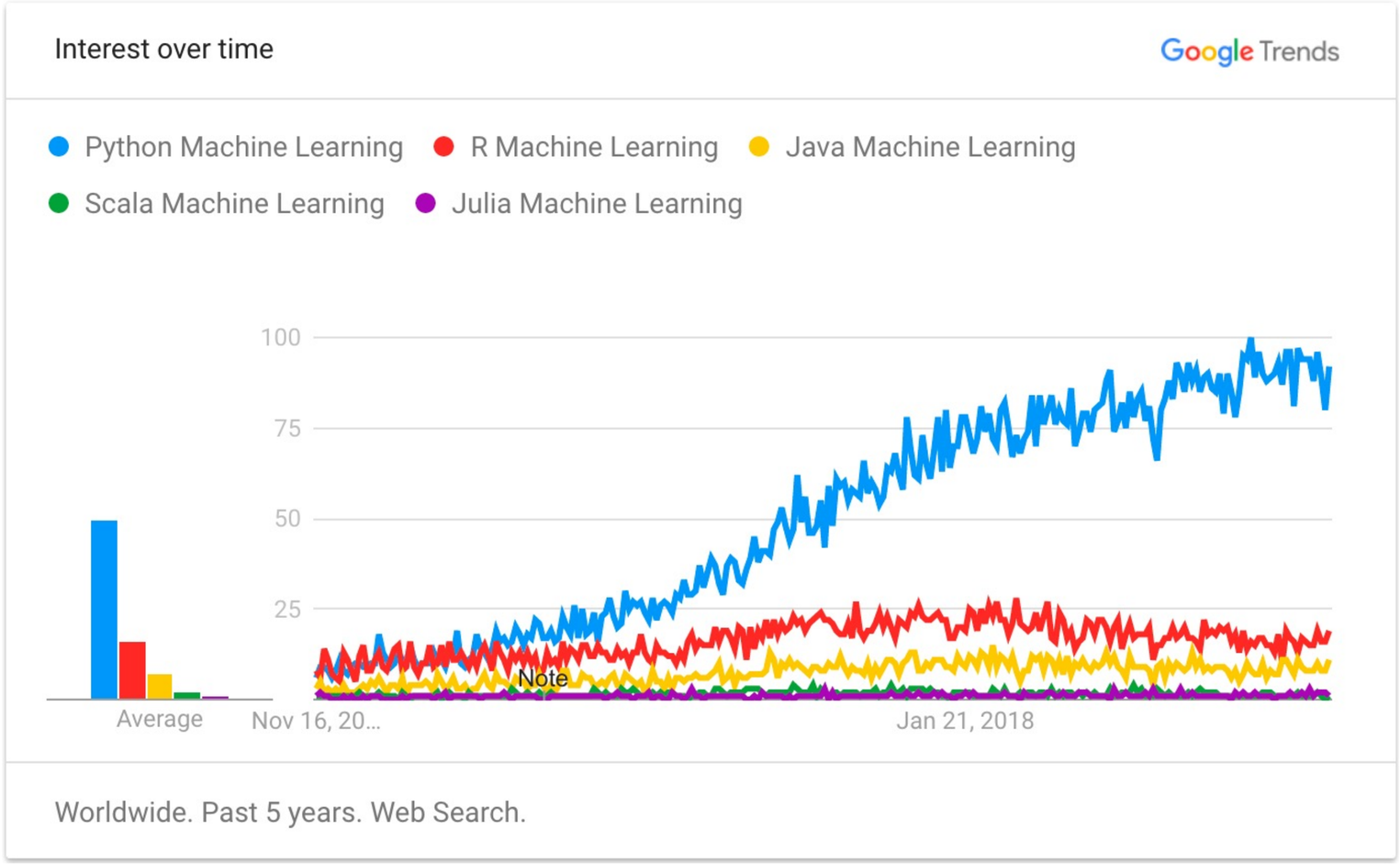
If your DeepLearning project fails

- You might have too few examples
- You didn’t trained it long enough
- You trained it too long (overfitting)
- Testing data distribution is different from the training data
- It will not fail because your neural network achieved sentience!
- It will not fail because daemons and zombies*!

*https://en.wikipedia.org/wiki/Zombie_process

Python: a short overview

Python for machine learning?



https://www.geeksforgeeks.org/why-is-python-the-best-suited-programming-language-for-machine-learning/

Python vs Java

- Python and Java are both object-oriented languages
- Python is “ducked typed” i.e. dynamically typed, while java is a statically typed
    - Ducked typed: ”If it looks like a duck and quacks like a duck, it's a duck”
    - you don't need a type in order to invoke an existing method on an object
- In Python, it is easy to write and read, but can be difficult to analyse

Python syntax: indentation

```
if 5 > 2:
  print("Five is greater than two!")
```

```
if 5 > 2:
print("Five is greater than two!")
```

=> Syntax error

Python syntax: data types

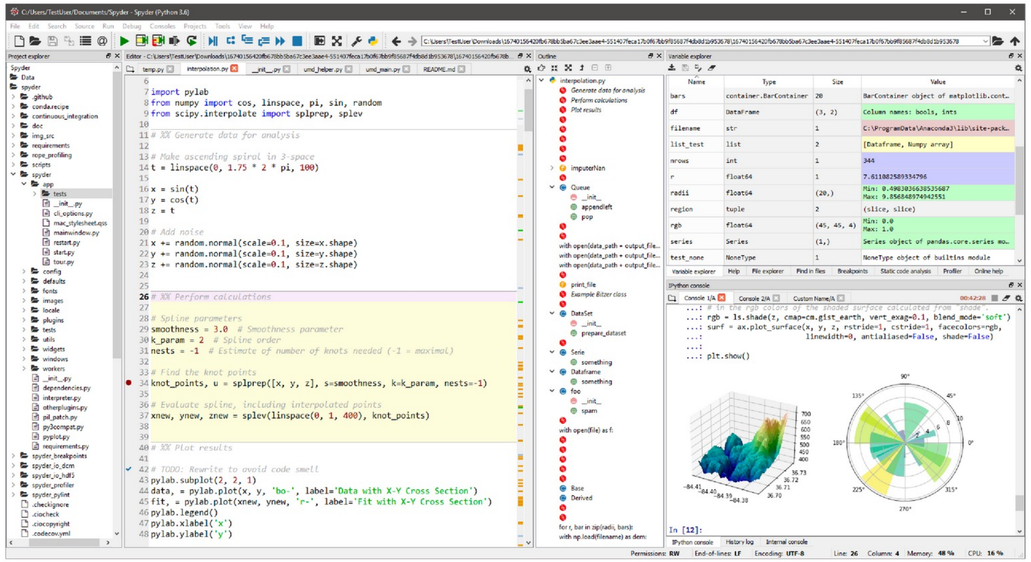
Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool

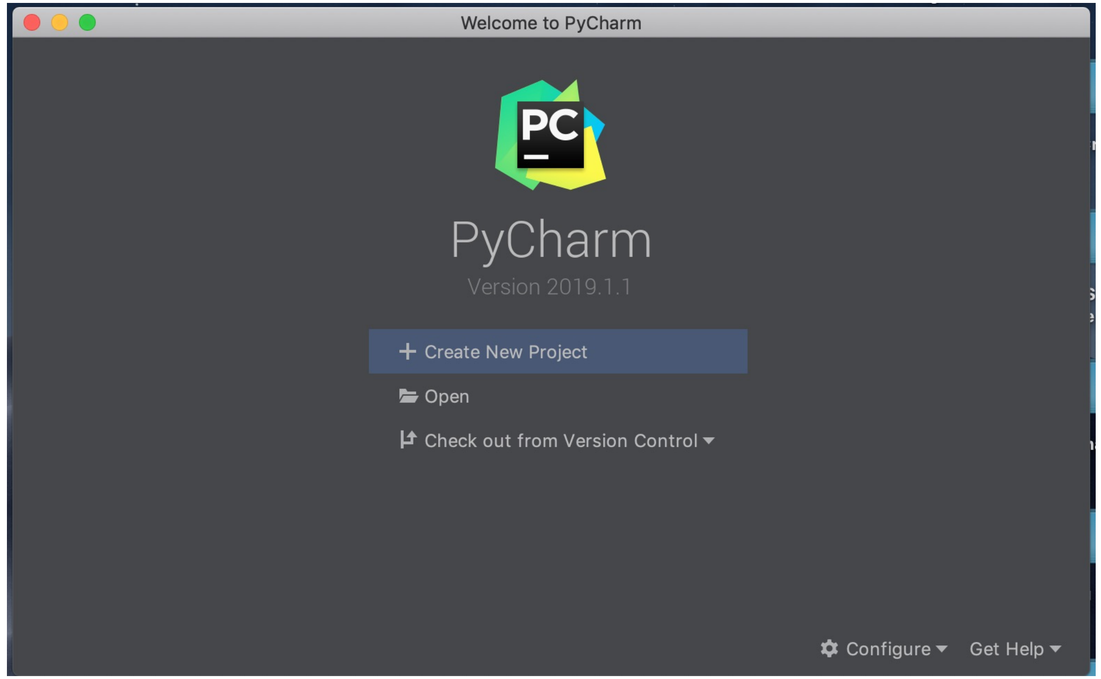
https://www.w3schools.com/python/python_datatypes.asp

Python IDEs

SPYDER

The Scientific Python Development Environment





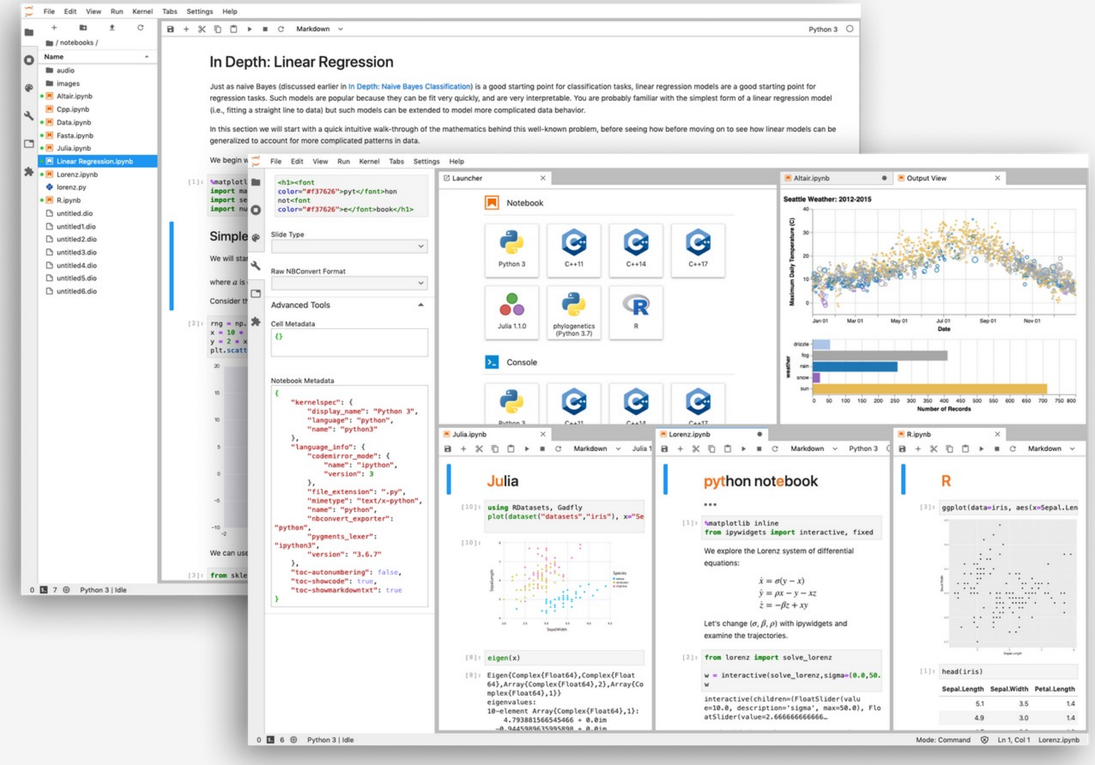
Python Tutorials

https://www.w3schools.com/python/default.asp
https://docs.python.org/3/tutorial/
https://www.datacamp.com/courses/intro-to-python-for-data-science

https://jupyter.org/

Python IDEs

“Jupyter Notebook is a web-based interactive computational environmen



JupyterLab 1.0: Jupyter’s Next-Generation Notebook Interface

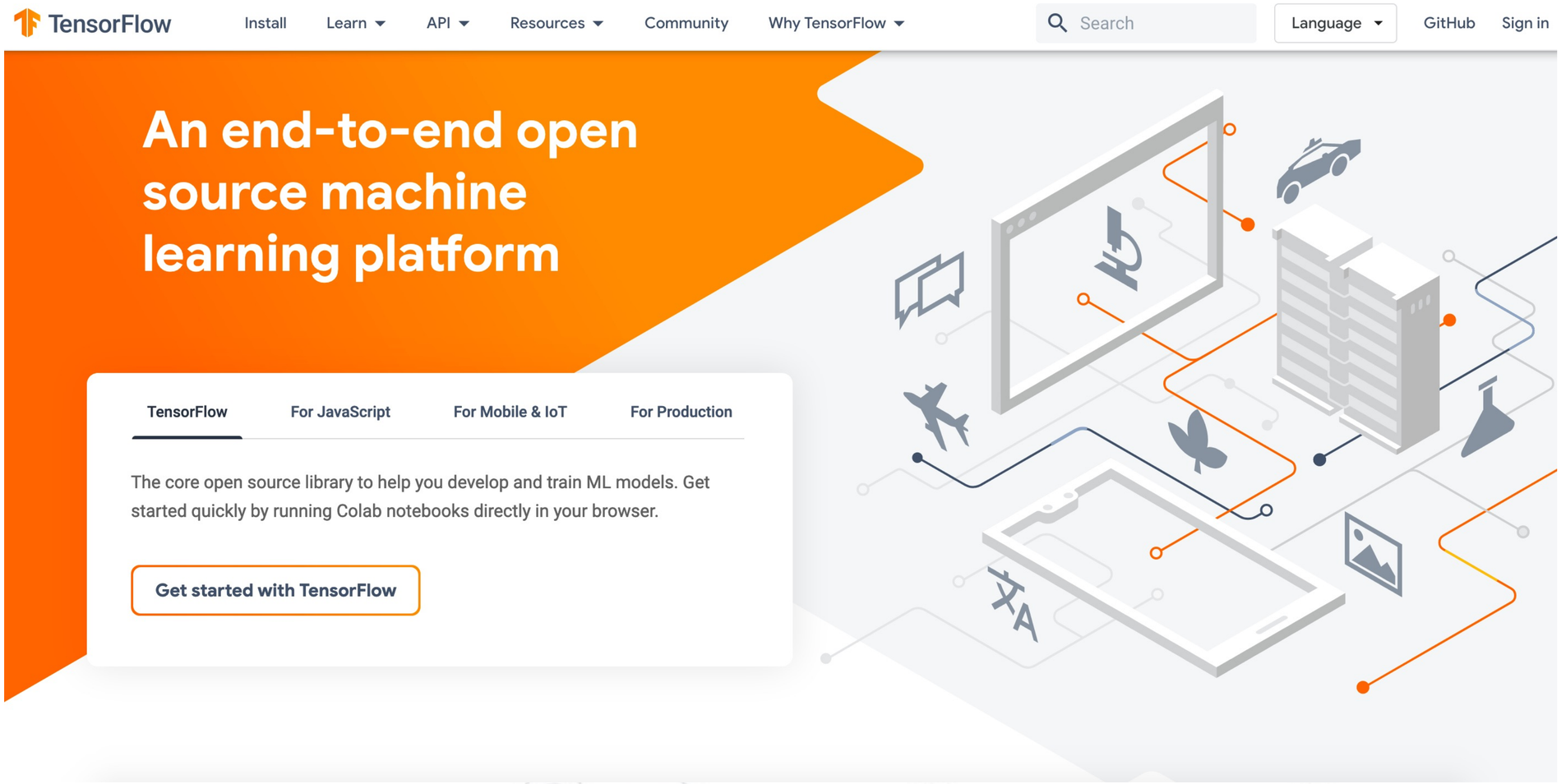
JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

Try it in your browser | Install JupyterLab

https://jupyter.org/

TensorFlow: a short overview

TensorFlow V2.0 was released on the September 30, 2019



TensorFlow V2.0

Tensor

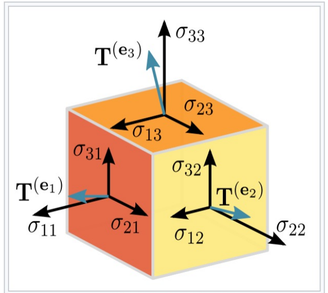
From Wikipedia, the free encyclopedia

This article is about tensors on a single vector space. For vector fields, see vector field. For tensor fields, see Tensor field. For other uses, see Tensor (disambiguation).

In mathematics, a **tensor** is an algebraic object that describes a linear mapping from one set of algebraic objects to another. Objects that tensors may map between include, but are not limited to, vectors and scalars, and, recursively, even other tensors (for example, a matrix is a map between vectors, and is thus a tensor. Therefore a linear map between matrices is also a tensor). Tensors are inherently related to vector spaces and their dual spaces, and can take several different forms – for example: a scalar, a vector, a dual vector at a point, or a multi-linear map between vector spaces. Euclidean vectors and scalars (which are often used in elementary physics and engineering applications where general relativity is irrelevant) are the simplest tensors.[1] While tensors are defined independent of any basis, the literature on physics often refers to them by their components in a basis related to a particular coordinate system.

An elementary example of a mapping describable as a tensor is the dot product, which maps two vectors to a scalar. A more complex example is the Cauchy stress tensor **T**, which takes a directional unit vector **v** as input and maps it to the stress vector $\mathbf{T}^{(\mathbf{v})}$, which is the force (per unit area) exerted by material on the negative side of the plane orthogonal to **v** against the material on the positive side of the plane, thus expressing a relationship between these two vectors, shown in the figure (right). The cross product, where two vectors are mapped to a third one, is strictly speaking not a tensor because it changes its sign under those transformations that change the orientation of the coordinate system. The totally anti-symmetric symbol ε_{ijk} nevertheless allows a convenient handling of the cross product in equally oriented three dimensional coordinate systems.

Assuming a basis of a real vector space, e.g., a coordinate frame in the ambient space, a tensor can be represented as an organized multidimensional array of numerical values with respect to this specific basis. Changing the basis transforms the values in the array in a characteristic way that allows to *define* tensors as objects adhering to this transformational behavior. For example, there are invariants of



The second-order Cauchy stress tensor in the basis ($\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$): $\mathbf{T} = \left[\mathbf{T}^{(\mathbf{e}_1)}\mathbf{T}^{(\mathbf{e}_2)}\mathbf{T}^{(\mathbf{e}_3)}\right]$, or $\mathbf{T} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$.

https://en.wikipedia.org/wiki/Tensor

Dataflow programming

From Wikipedia, the free encyclopedia

In computer programming, **dataflow programming** is a programming paradigm that models a program as a directed graph of the data flowing between operations, thus implementing dataflow principles and architecture. Dataflow programming languages share some features of functional languages, and were generally developed in order to bring some functional concepts to a language more suitable for numeric processing. Some authors use the term *datastream* instead of *dataflow* to avoid confusion with dataflow computing or dataflow architecture, based on an indeterministic machine paradigm. Dataflow programming was pioneered by Jack Dennis and his graduate students at MIT in the 1960s.

Contents [hide]

https://en.wikipedia.org/wiki/Dataflow_programming

TensorFlow V2.0

- Can run on multiple GPUs & CPUs
- Works on Linux, MacOS, Windows, iOS, Android
- It has a stable Python & C implementation, but could be integrated with JavaScript, C++, Java, C#, Haskell, R (third-party packages)

TensorFlow V2.0

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

In TensorFlow V2.0 Keras is the official API !!!

TensorFlow comes with some simple datasets

In the original dataset a pixel value is a grayscale colour from 0-255. We need to normalise this in the interval [0,1]

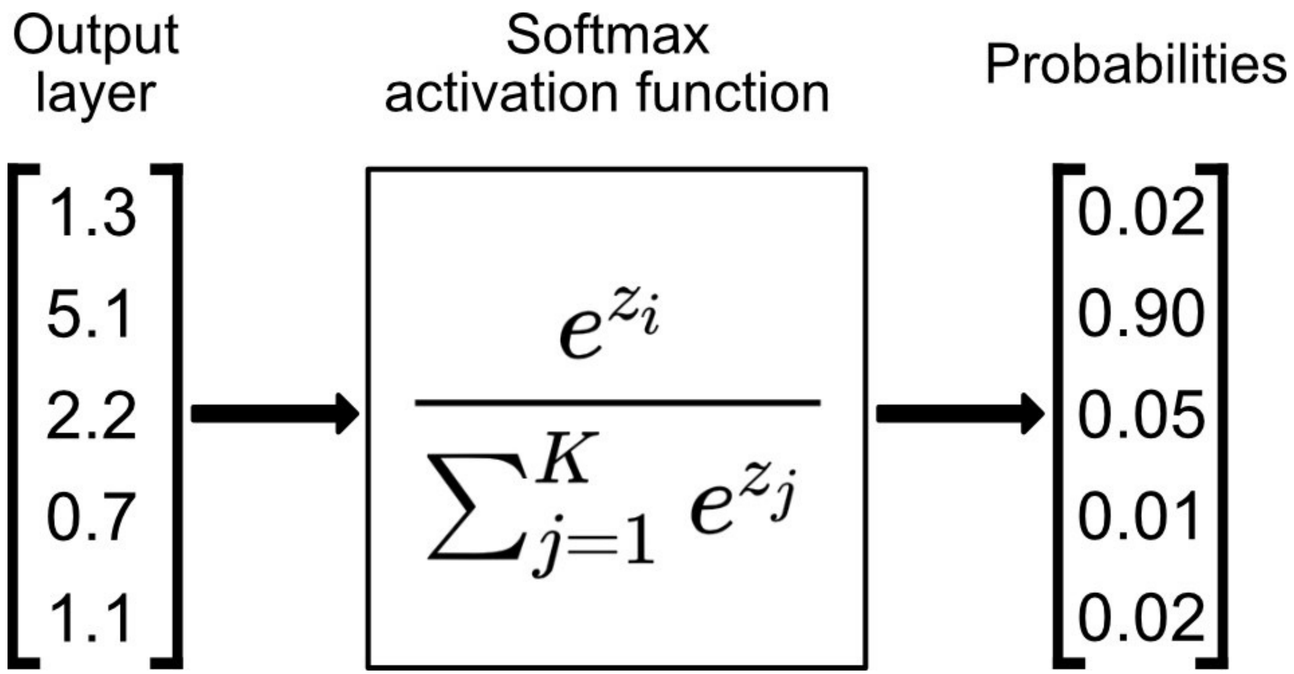
We define a simple neural network, with
One input layer
One hidden layer
Dropout is applied to the layer defined previously, in our case the hidden layer
One output layer

Compile defines the **loss function**, the **optimiser** and the **. You compile only when training!**

Epochs represents how many times we go through the entire dataset in the training

Evaluate will return the accuracy on the test set

TensorFlow V2.0



https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

The lab

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

Epoch
000,000

Learning rate: 0.03
Activation: Tanh
Regularization: None
Regularization rate: 0
Problem type: Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

X1
X2
X12
X22
X1X2

2 HIDDEN LAYERS

4 neurons

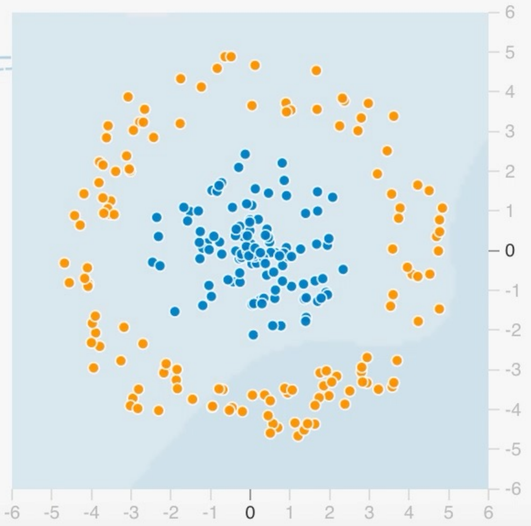
2 neurons

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

OUTPUT

Test loss 0.499
Training loss 0.520



HelloWorld-MNIST.ipynb

File Edit View Insert Runtime Tools Help

OPEN IN PLAYGROUND

Let's get started, import the TensorFlow library into your program, along with the library to plot graphs (matplotlib)

```
%matplotlib inline
import tensorflow as tf
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (7,7) # Make the figures a bit bigger

import numpy as np
```

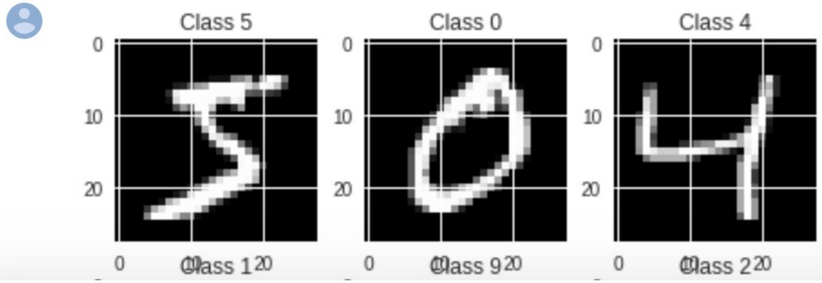
Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

```
[ ]  mnist = tf.keras.datasets.mnist

     (x_train, y_train), (x_test, y_test) = mnist.load_data()
     x_train, x_test = x_train / 255.0, x_test / 255.0
```

Visualize some of the dataset images

```
[ ]  for i in range(9):
         plt.subplot(3,3,i+1)
         plt.imshow(x_train[i], cmap='gray', interpolation='none')
         plt.title("Class {}".format(y_train[i]))
```



Thank you!