

# Deep Learning

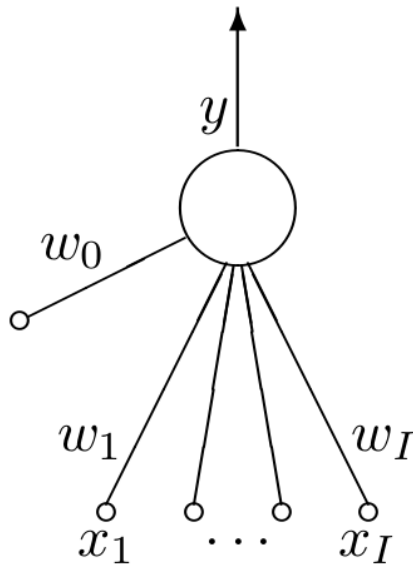
Past, present and future

Prof. Gilles Louppe, Chaire NRB  
[g.louppe@uliege.be](mailto:g.louppe@uliege.be)

**Past**

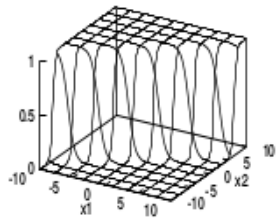


# Perceptron

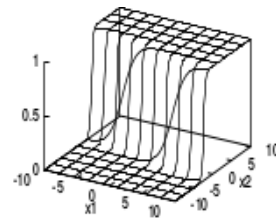


$$f(\mathbf{x}; \mathbf{w}) = \begin{cases} 1 & \text{if } w_0 + \mathbf{w}^T \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

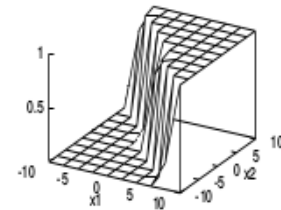
The Perceptron (Rosenblatt, 1957)



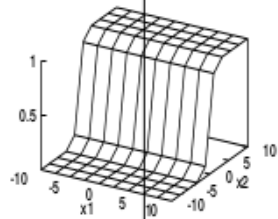
$$\mathbf{w} = (-2, 3)$$



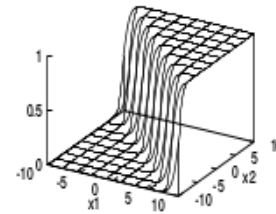
$$\mathbf{w} = (1, 4)$$



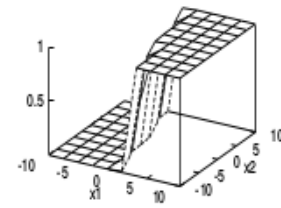
$$\mathbf{w} = (5, 4)$$



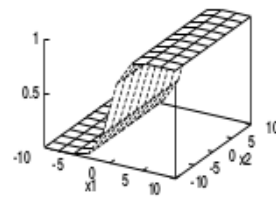
$$\mathbf{w} = (0, 2)$$



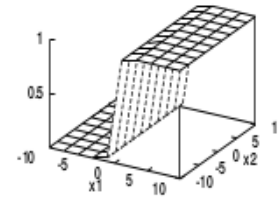
$$\mathbf{w} = (2, 2)$$



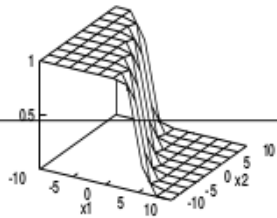
$$\mathbf{w} = (5, 1)$$



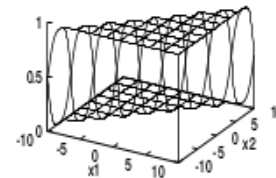
$$\mathbf{w} = (1, 0)$$



$$\mathbf{w} = (3, 0)$$

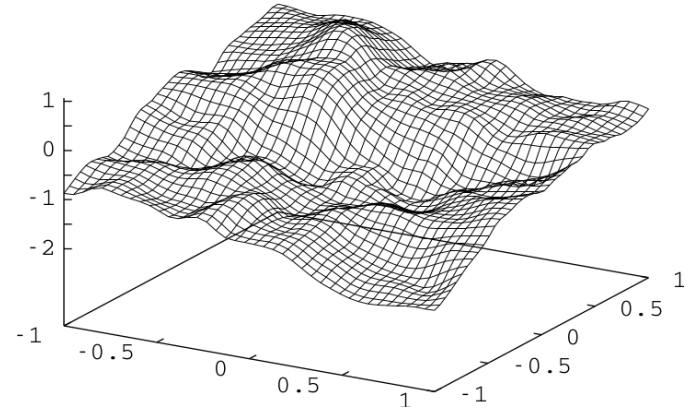
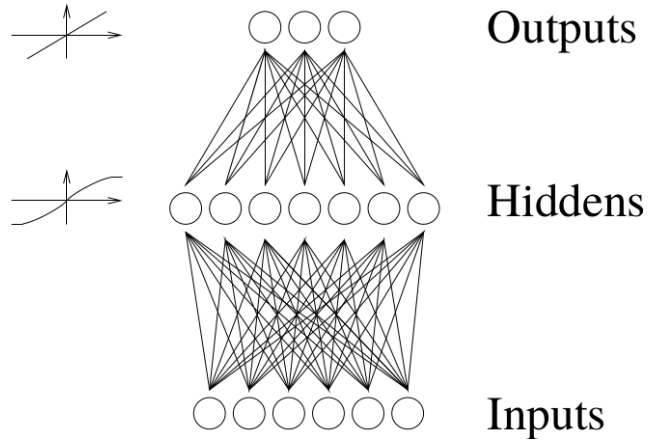


$$\mathbf{w} = (-2, -1)$$



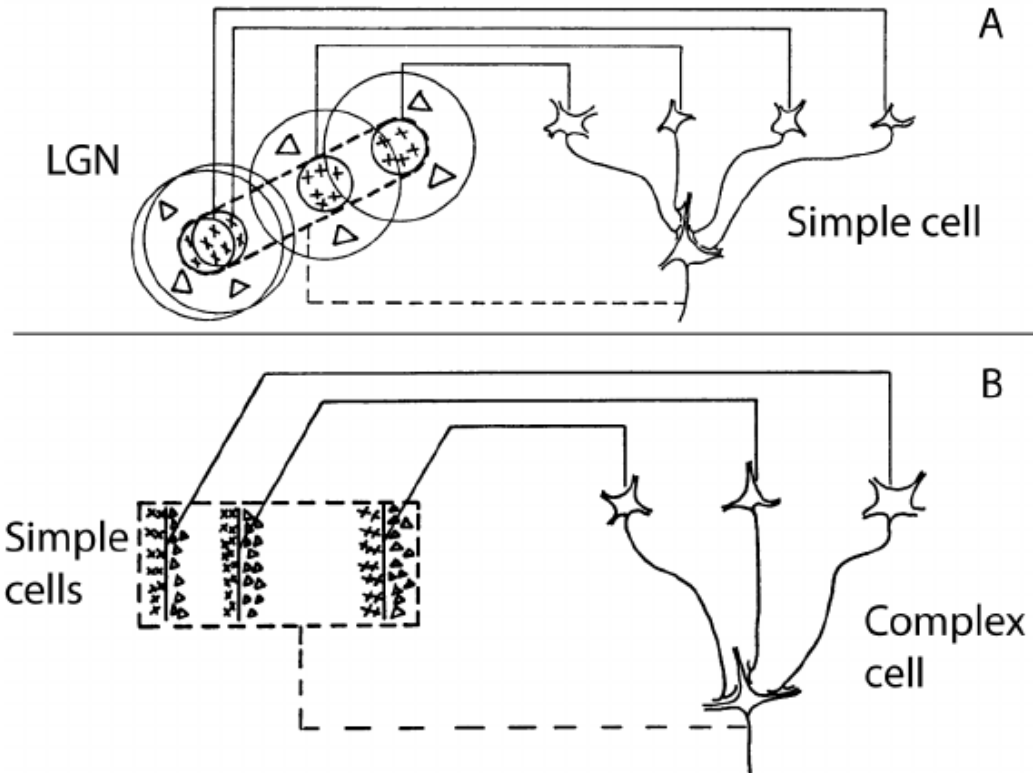
$$\mathbf{w} = (2, -2)$$

$w_1$

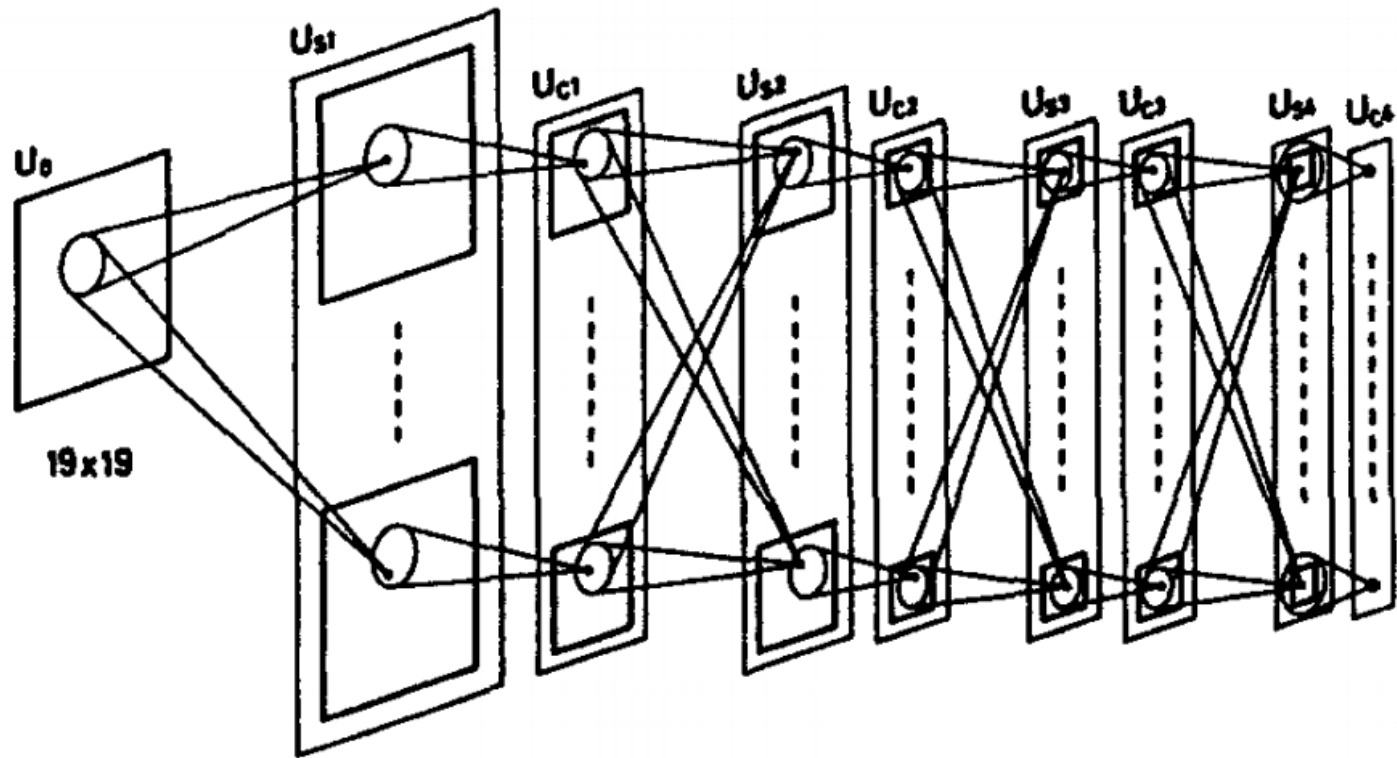


The Multi-Layer Perceptron (Rumelhart et al, 1986)

# Convolutional networks

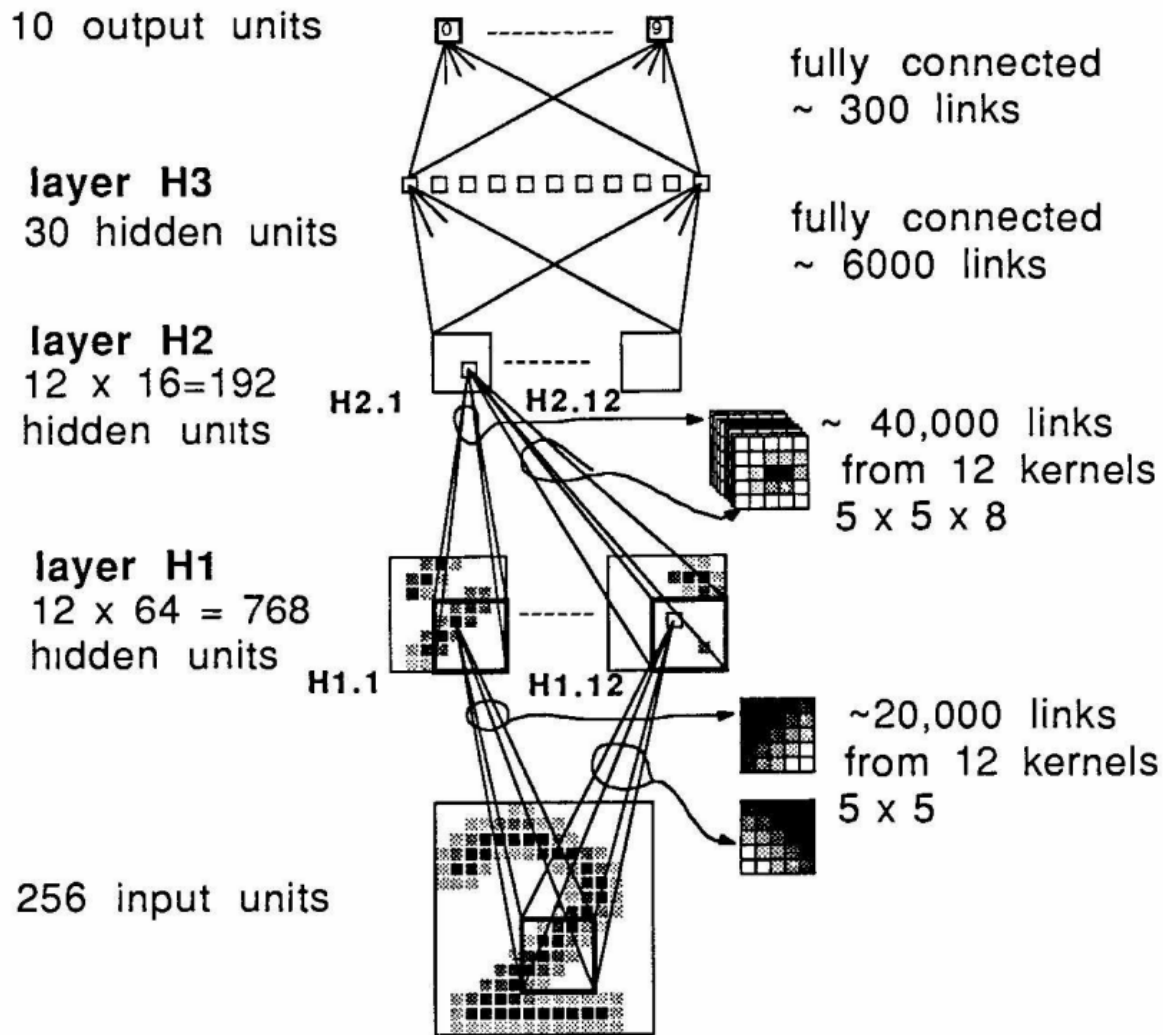


Hubel and Wiesel, 1962



The Neocognitron (Fukushima, 1987)





Convolutional network (LeCun et al, 1989)





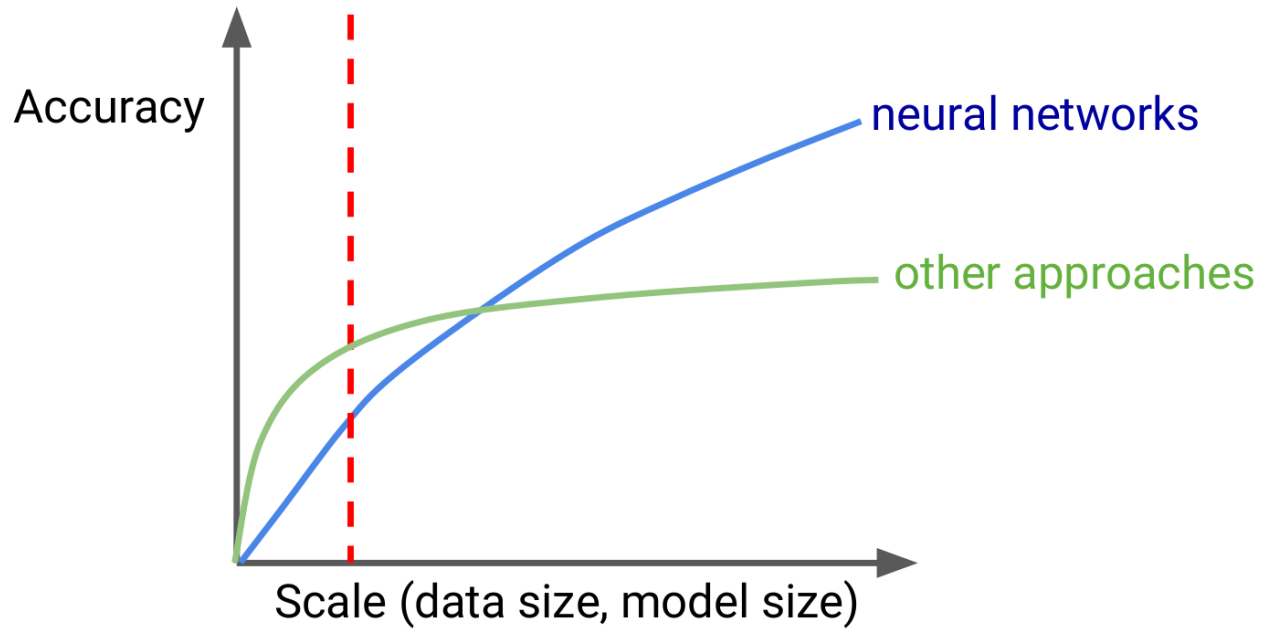
LeCun et al, 1993

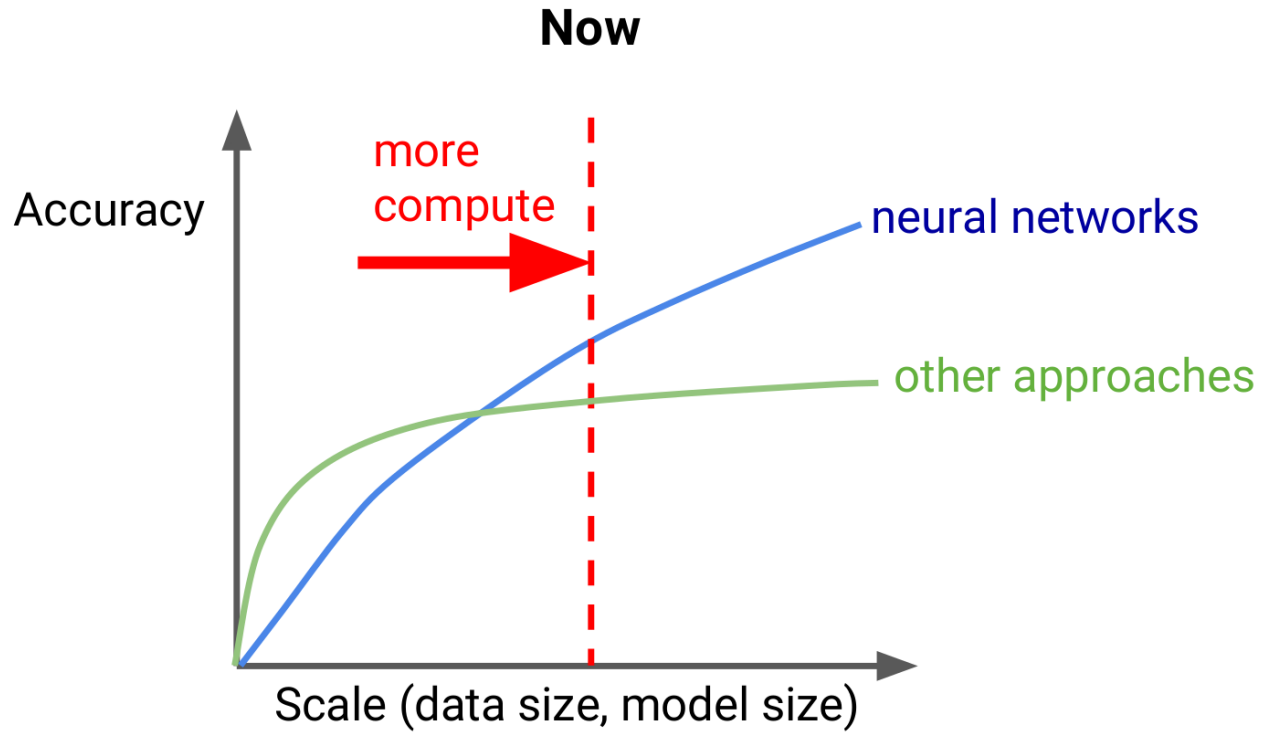
# Learning

$$\theta_{t+1} = \theta_t - \gamma \nabla_{\theta} \mathcal{L}(\theta_t)$$

**Present**

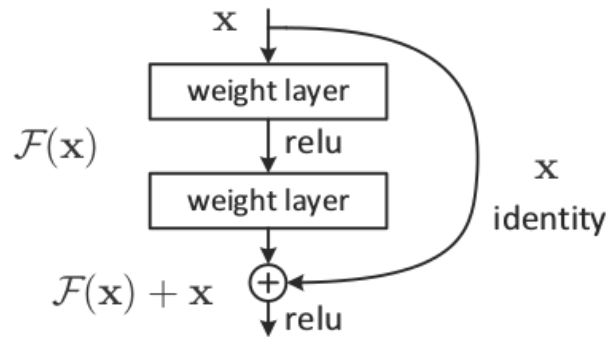
## 1980s and 1990s





# What has changed?

Algorithms



Data



Software



theano



PYTORCH

dmlc  
mxnet

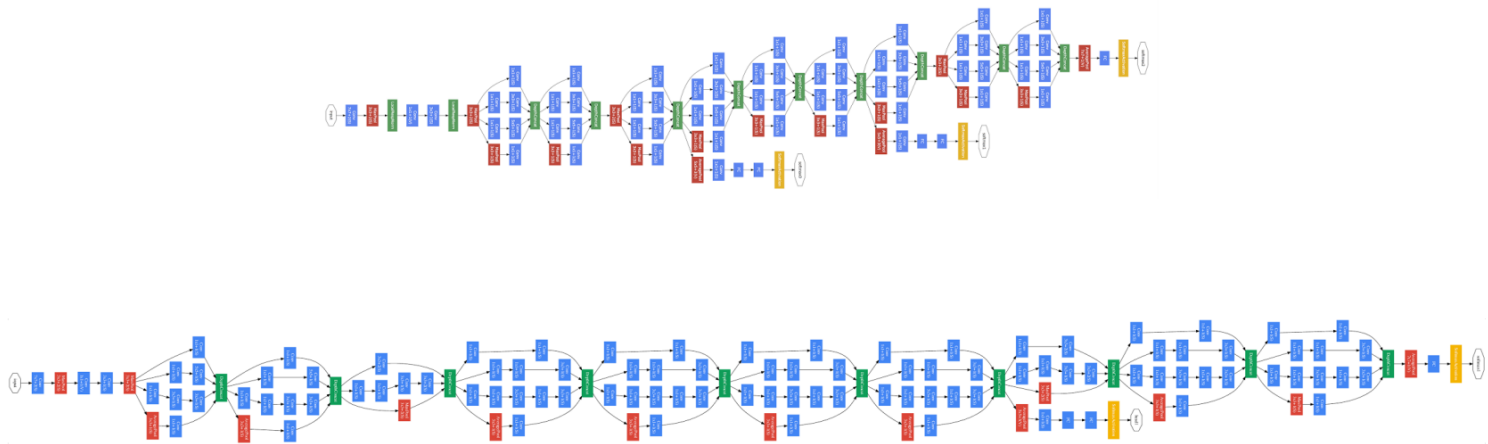


Compute engines





# Deep networks



Szegedy et al, 2014

# Applications

airplane



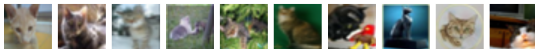
automobile



bird



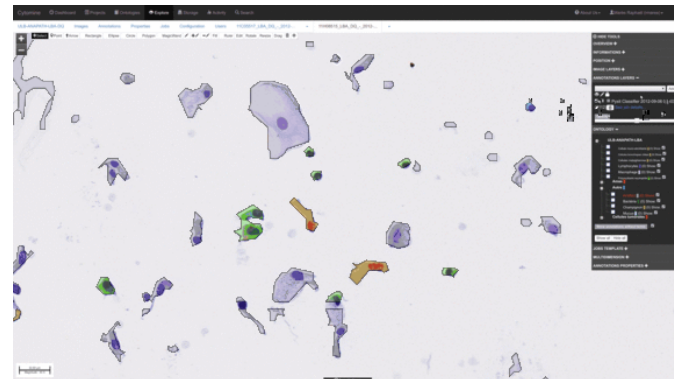
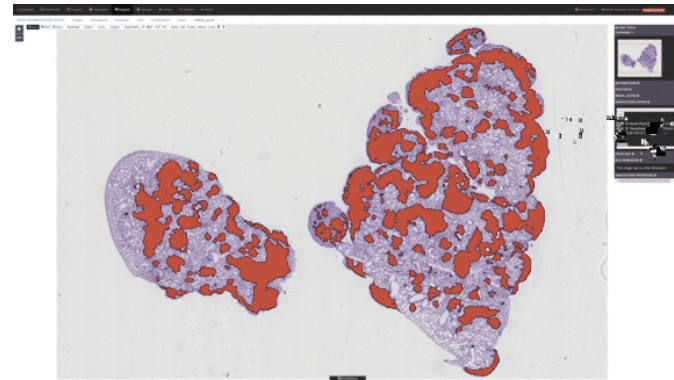
cat



deer

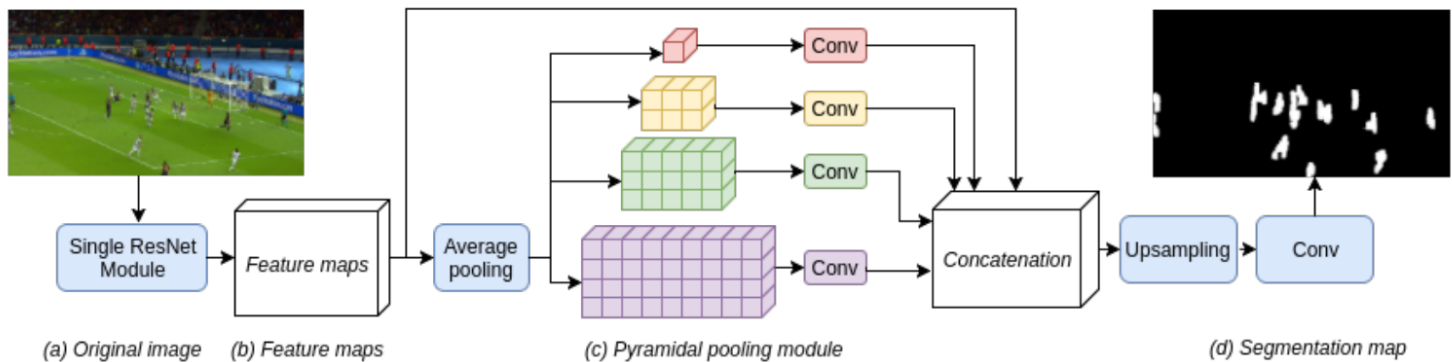
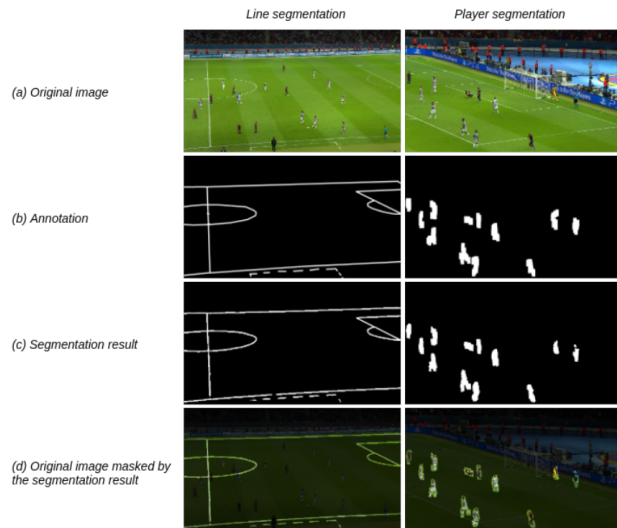


dog



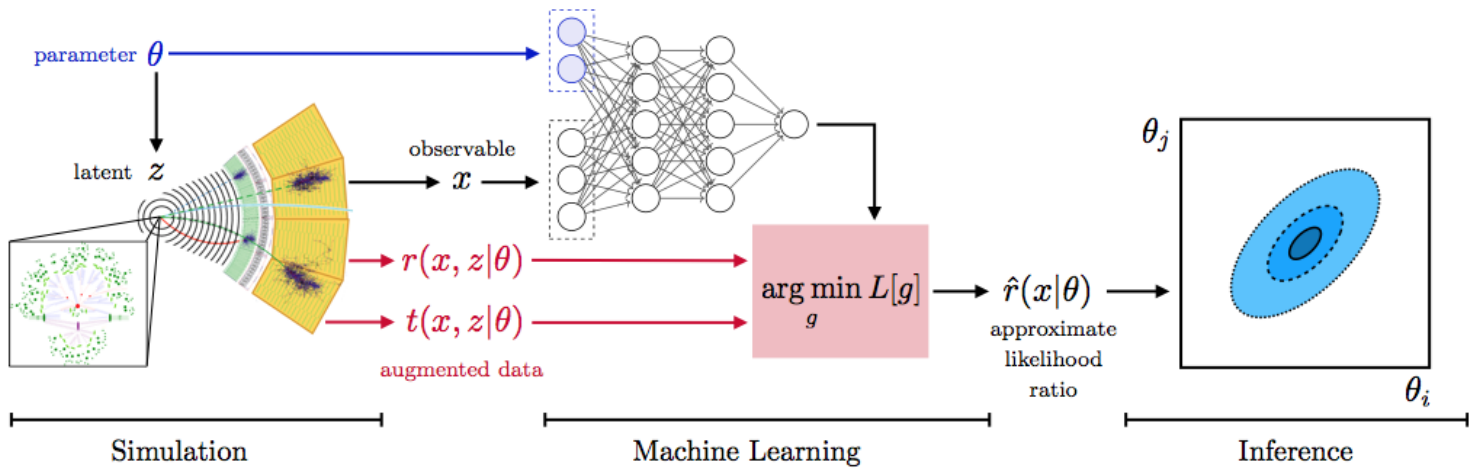
(Left) Image classification

(Right) Biomedical image segmentation (Cytomine, 2010-2018, ULiège)



Soccer games analysis (Cioppa et al, 2018, ULiège)





Analysis of scientific data at the LHC (Brehmer et al, 2018)



Autonomous drones (Smolianskiy et al, 2017)



Autonomous cars (NVIDIA)



Learning to play video games (Mnih et al, 2013)





Learning to perform tasks (Levine et al, 2015)

# Future

## Automatisation : 4 belges sur 10 craignent de perdre leur job. A tort ?


Trends Tendances

08/05/18 à 15:58 - Mise à jour à 16:01  
Source: Trends-Tendances

L'automatisation galopante au sein des entreprises inquiète de nombreux travailleurs belges. Quatre Belges sur dix craignent ainsi de voir leur fonction disparaître au cours des dix prochaines années. À tort ?

12  
Fois partagé



 Lire plus tard





*Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we write software. **They are Software 2.0.***

Andrej Karpathy (Director of AI, Tesla, 2017)

# Software 1.0

- Programs are written in languages such as [Python](#), [C](#) or [Java](#).
- They consist of **explicit instructions** to the computer **written by a programmer**.
- The programmer identifies **a specific point** in program space with some desirable behavior.

```
17 string sInput;
18 int iLength;
19 double dblTemp;
20 bool again = true;
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     stringstream(sInput) >> iLength;
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     } while (++iN < iLength) {
36         if (isdigit(sInput[iN])) {
37             continue;
38         } else if (iN == (iLength - 3)) {
39             continue;
40         }
41     }
42 }
```

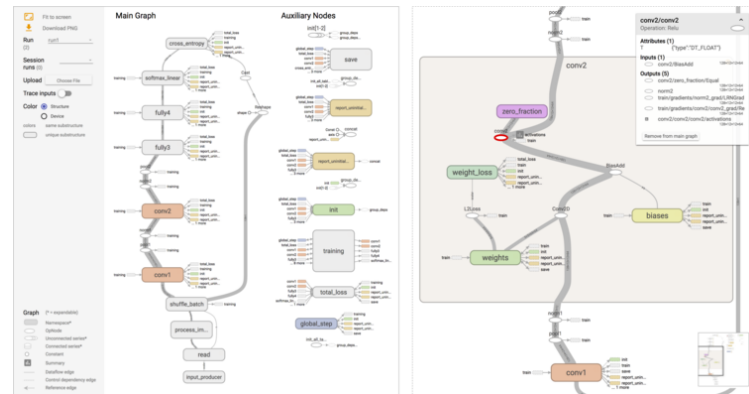
*Software 1.0*

# Software 2.0

- Programs are written in **neural network weights**
- **No human is involved** in writing those weights!
- Instead, **specify constraints** on the behavior of a desirable program (e.g., through data).
- **Search** the program space through optimization.

```
17 string sInput;
18 int iLength, iN;
19 double dblTemp;
20 bool again = true;
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     } while (++iN < iLength) {
36         if (isdigit(sInput[iN])) {
37             continue;
38         } else if (iN == (iLength - 3)) {
```

Software 1.0



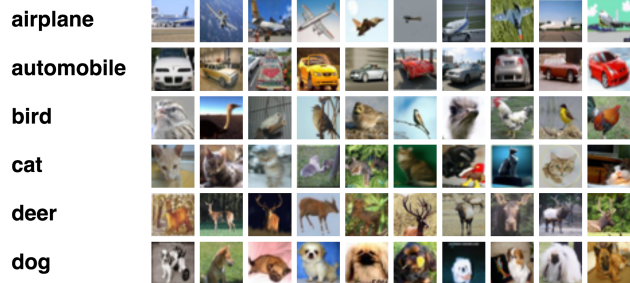
Software 2.0

For many real-world problems, it is often significantly **easier to collect the data** than to explicitly write the program.

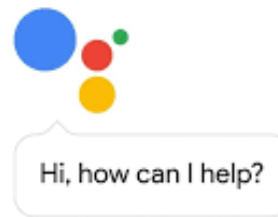
Therefore,

- programmers of tomorrow do not maintain complex software repositories, write intricate programs or analyze their running times.
- Instead, **programmers become teachers**. They collect, clean, manipulate, label, analyze and visualize the data that feeds neural nets.

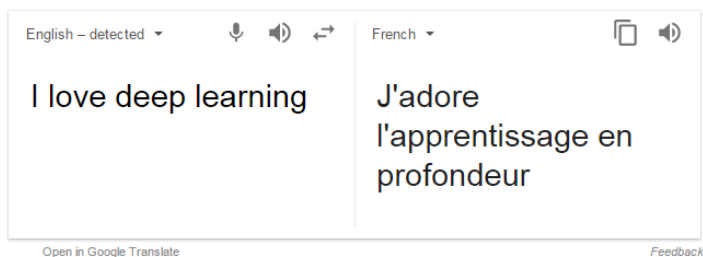
## Pixel data (e.g., visual recognition)



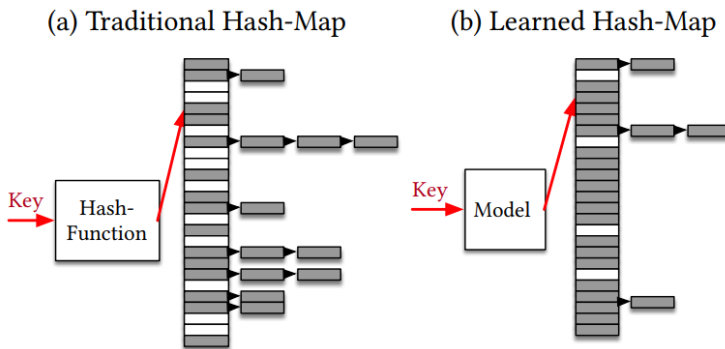
## Audio data (e.g., speech recognition and synthesis)



## Text data (e.g., machine translation)



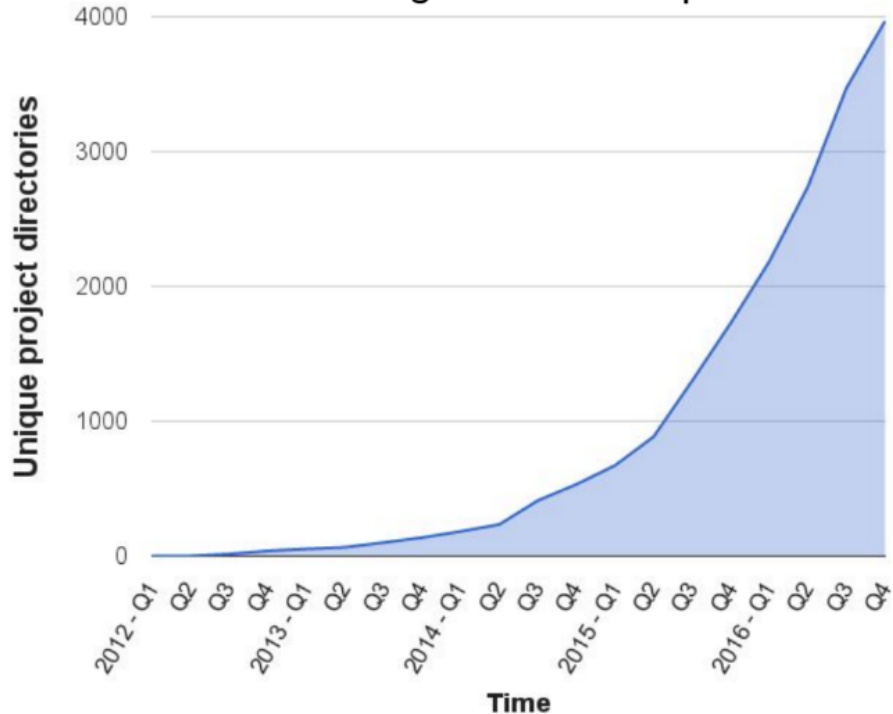
## System applications (e.g., databases)





# Growing Use of Deep Learning at Google

# of directories containing model description files



**Across many products/areas:**

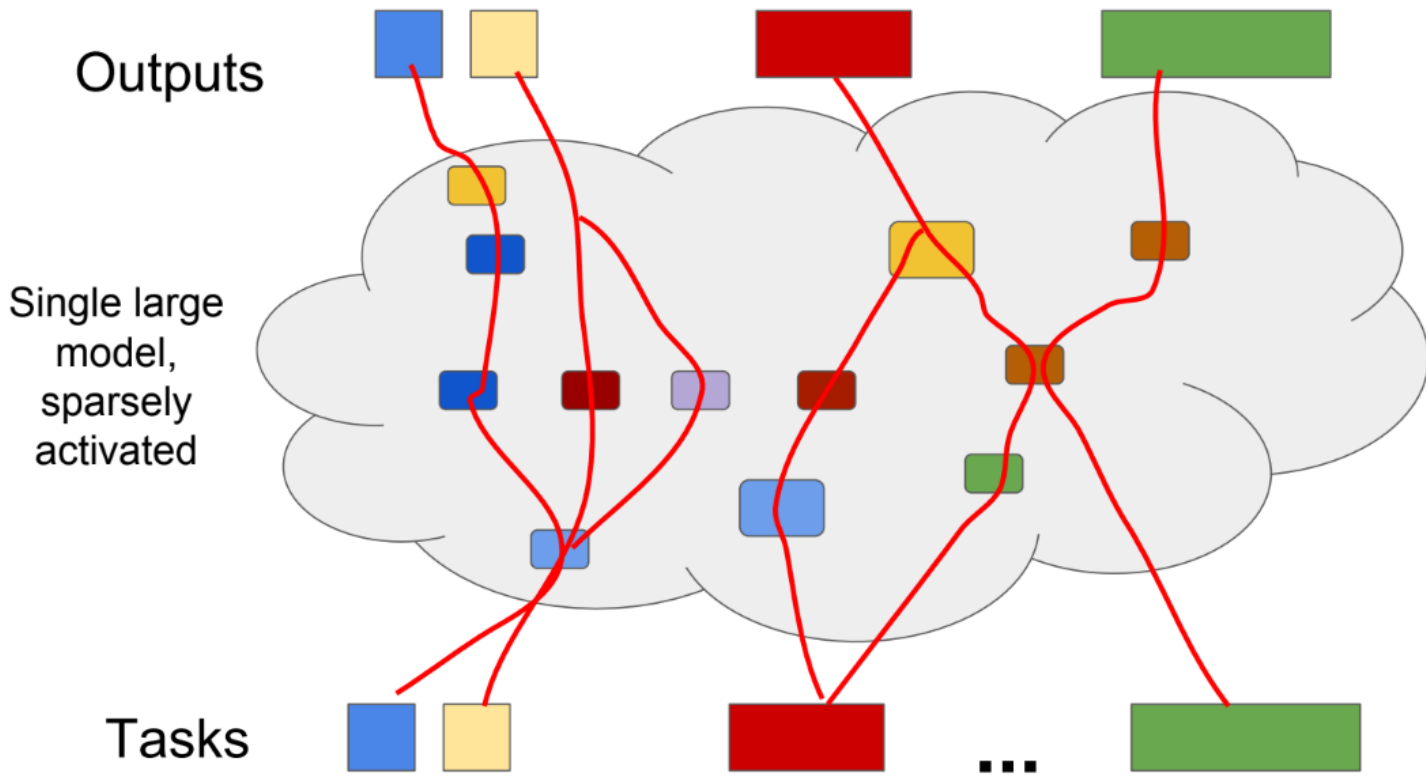
- Android
- Apps
- drug discovery
- Gmail
- Image understanding
- Maps
- Natural language understanding
- Photos
- Robotics research
- Speech
- Translation
- YouTube
- ... many others ...



(Jeff Dean, Lead of Google.ai, 2017)

# Benefits

- Computationally homogeneous
- Simple to bake in silicon
- Constant running time
- Constant memory use
- It is highly portable
- It is very agile
- **It is better than you**



Modules can meld into an optimal whole (Jeff Dean, Lead of Google.ai, 2017)

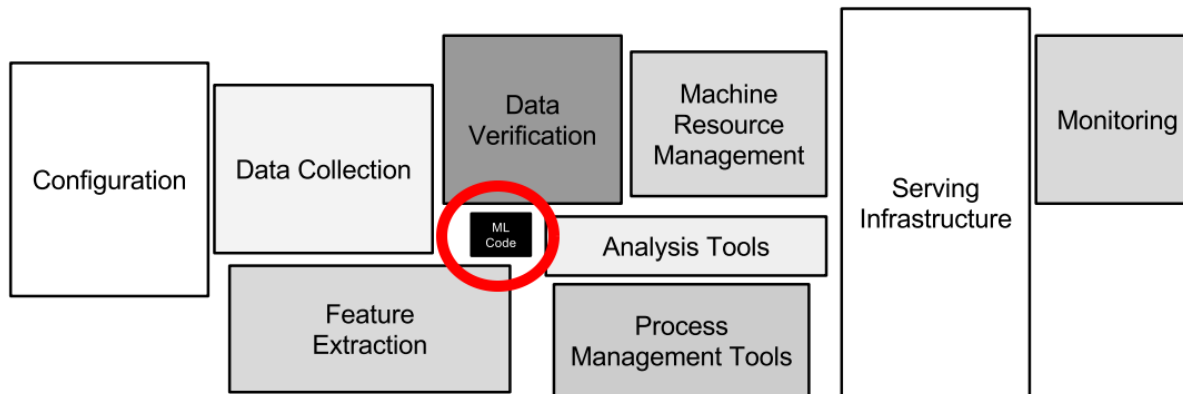


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Software 2.0 needs Software 1.0 (Sculley et al, 2015)



*People are now building a new kind of software by assembling networks of parameterized functional blocks and by training them from examples using some form of gradient-based optimization.*

*An increasingly large number of **people are defining the networks procedurally in a data-dependent way (with loops and conditionals)**, allowing them to change dynamically as a function of the input data fed to them. It's really **very much like a regular program, except it's parameterized.***

Yann LeCun (Director of AI Research, Facebook, 2018)

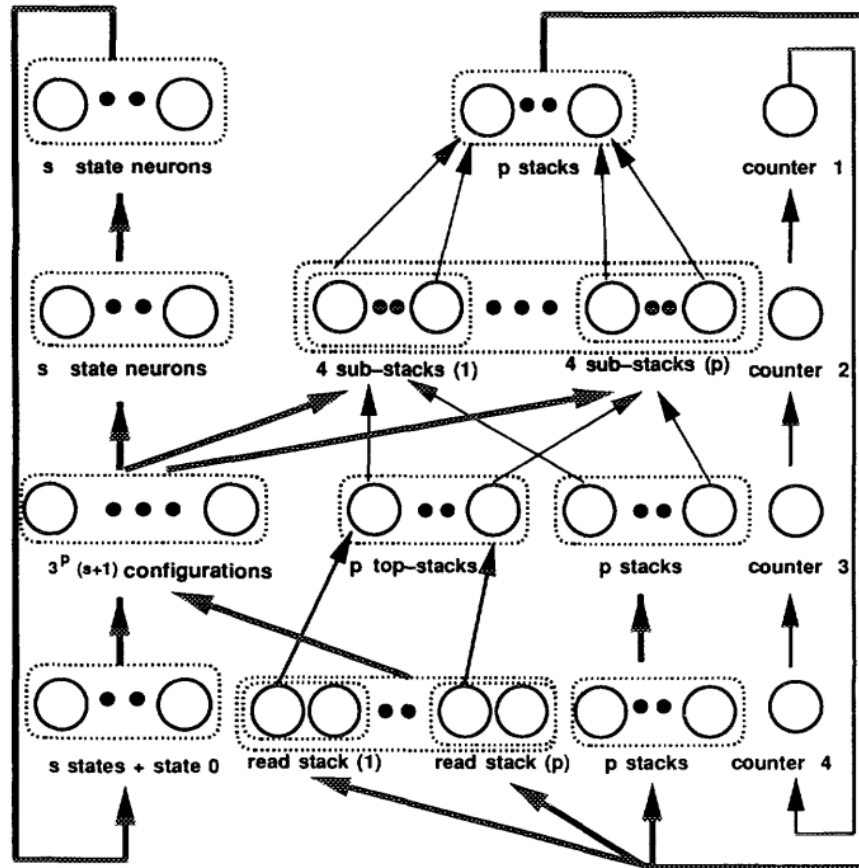
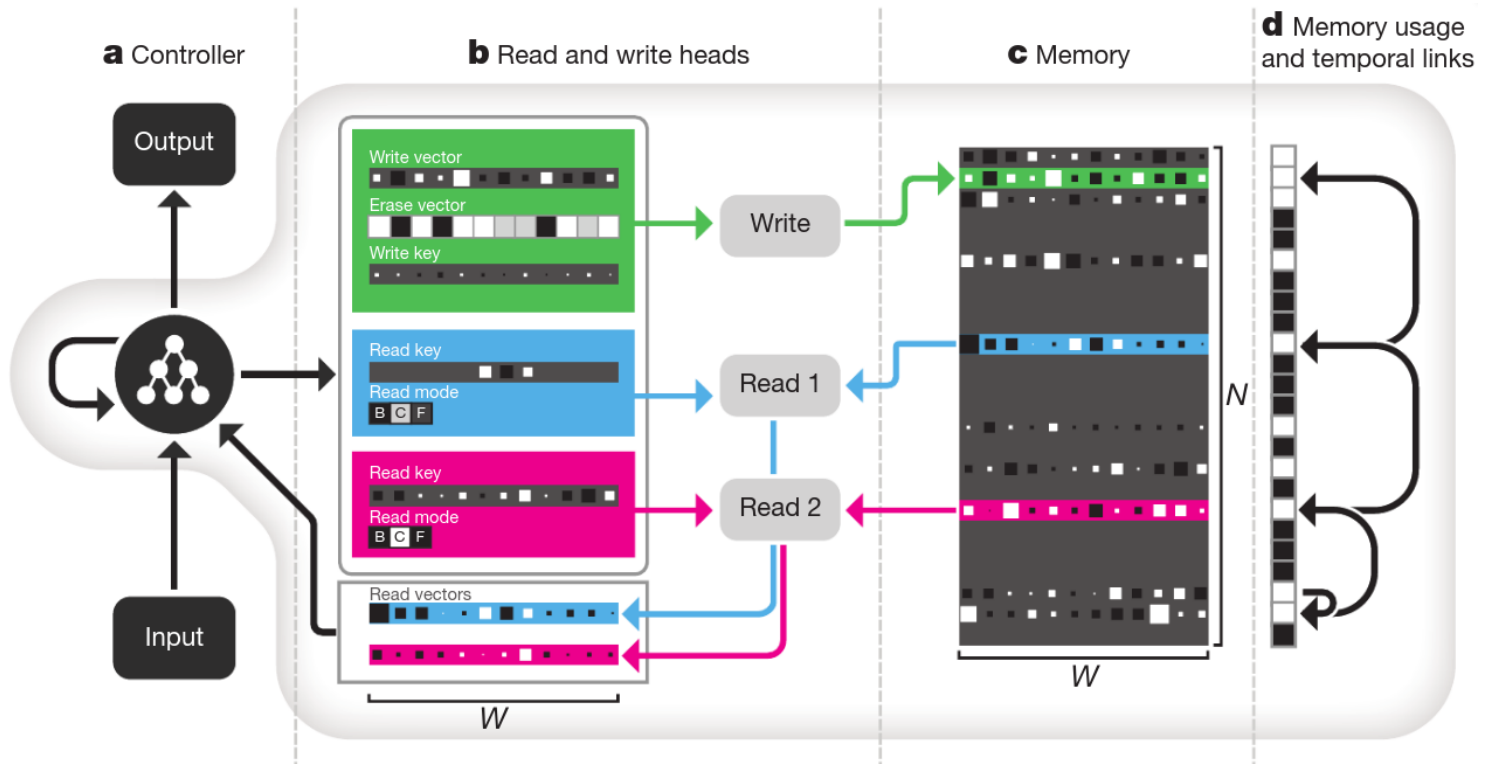


FIG. 1. The universal network.

Any Turing machine can be simulated by a recurrent neural network  
(Siegelmann and Sontag, 1995)



Differentiable Neural Computer (Graves et al, 2016)





# Summary

- Past: Deep Learning has a **long history**, fueled by contributions from neuroscience, control and computer science.
- Present: It is now mature enough to enable **applications with super-human level performance**, as already illustrated in many engineering disciplines.
- Future: Neural networks are **not just another classifier**. Sooner than later, they will take over increasingly large portions of what Software 1.0 is responsible for today.

