Deep Learning

Past, present and future

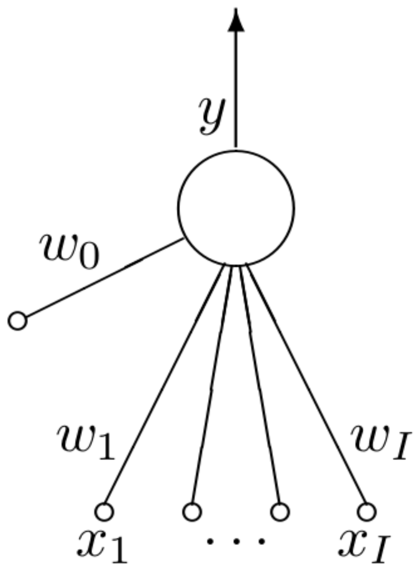
Prof. Gilles Louppe
g.louppe@uliege.be

Past

```
In [1]: from matplotlib.pyplot import imread
        imread("mushroom-small.png")

Out[1]: array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
                [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
                [0.4117647 , 0.34117648, 0.37254903, 1.        ],
                ...,
                [0.20392157, 0.23529412, 0.17254902, 1.        ],
                [0.16470589, 0.18039216, 0.12156863, 1.        ],
                [0.18039216, 0.18039216, 0.14117648, 1.        ]],

               [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
                [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
                [0.21176471, 0.2       , 0.20392157, 1.        ],
                ...,
                [0.1764706 , 0.24705882, 0.12156863, 1.        ],
                [0.10980392, 0.15686275, 0.07843138, 1.        ],
                [0.16470589, 0.20784314, 0.11764706, 1.        ]],

               [[0.14117648, 0.12941177, 0.10980392, 1.        ],
                [0.21176471, 0.1882353 , 0.16862746, 1.        ],
                [0.14117648, 0.13725491, 0.12941177, 1.        ],
                ...,
                [0.10980392, 0.15686275, 0.08627451, 1.        ],
                [0.0627451 , 0.08235294, 0.05098039, 1.        ],
                [0.14117648, 0.2       , 0.09803922, 1.        ]],

               ...,
```

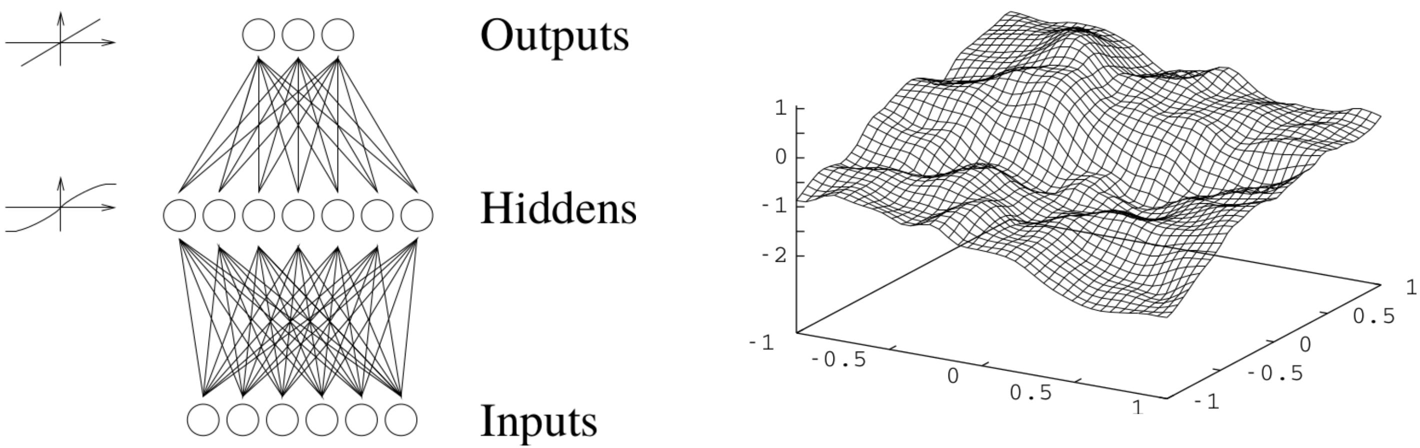
How to write a computer program that performs tasks what we can all easily do, yet all fail to describe precisely how?

Perceptron



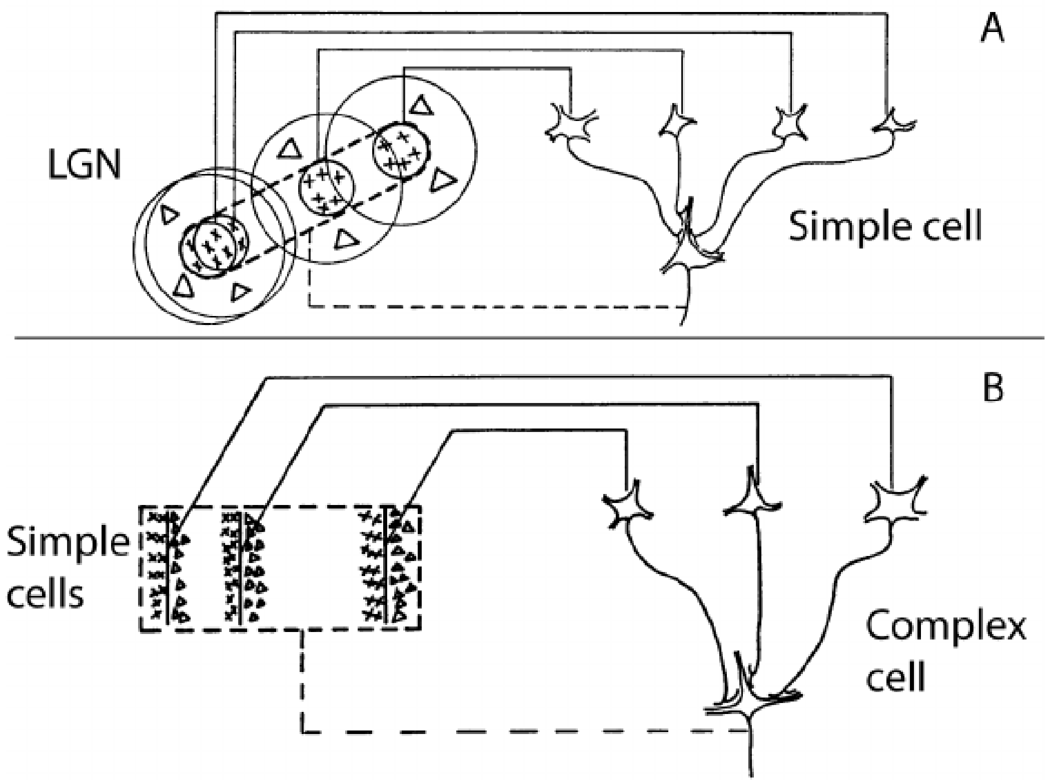
$$f(\mathbf{x}; \mathbf{w}, b) = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

The Perceptron (Rosenblatt, 1957)

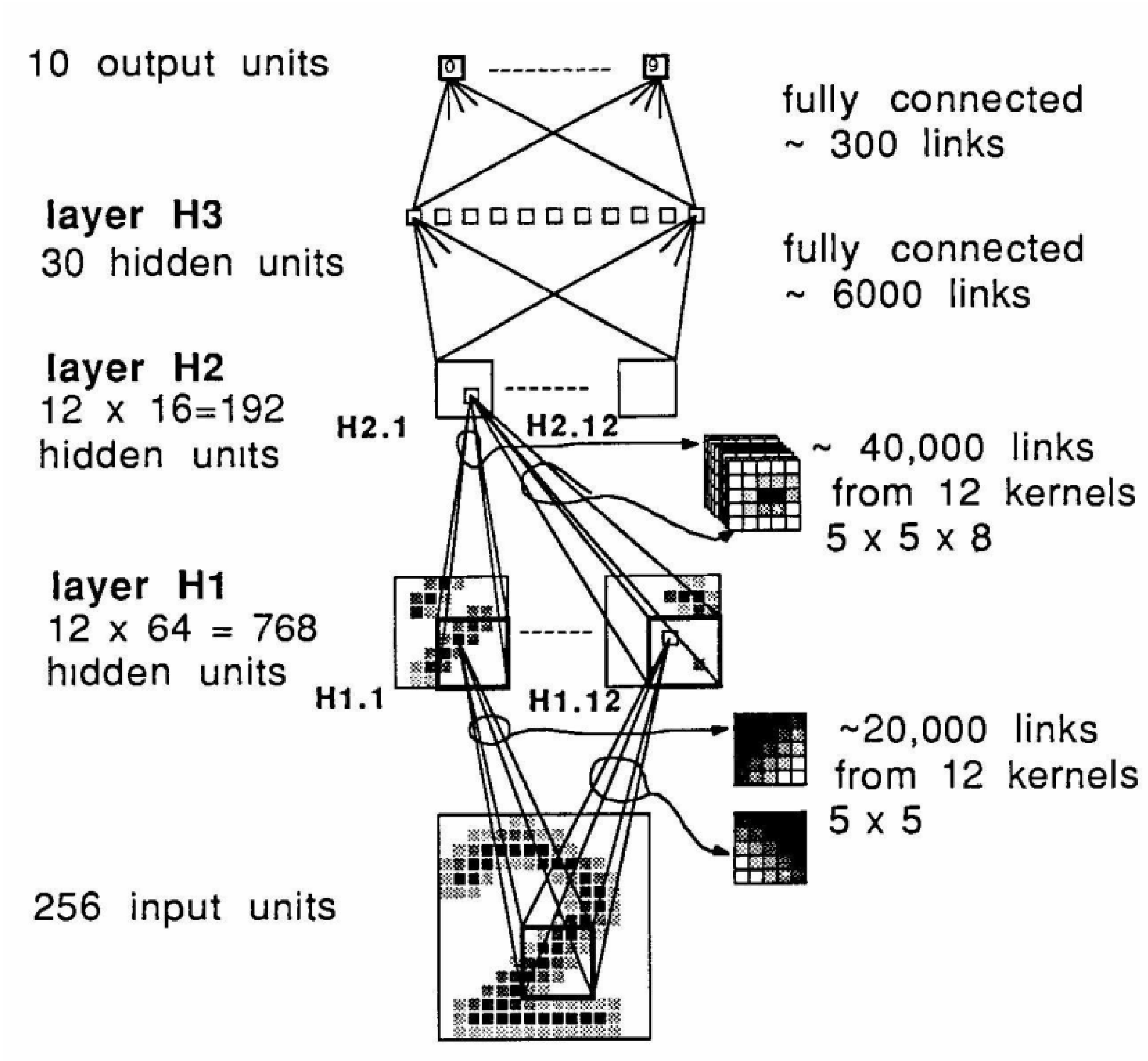


The Multi-Layer Perceptron (Rumelhart et al, 1986)

Convolutional networks



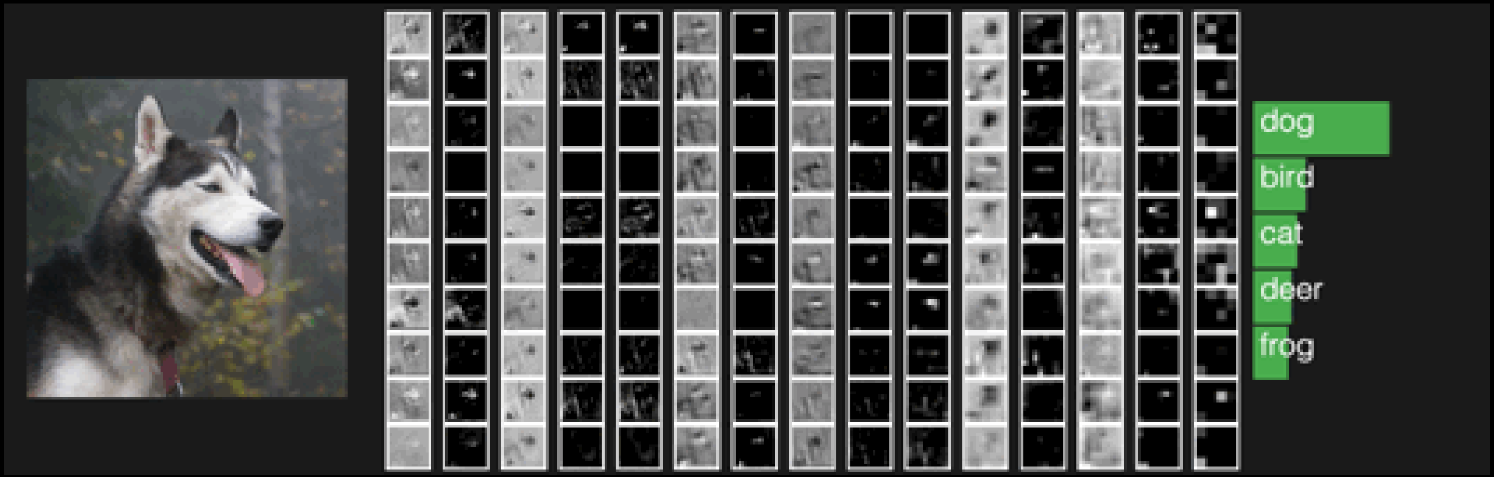
Hubel and Wiesel, 1962



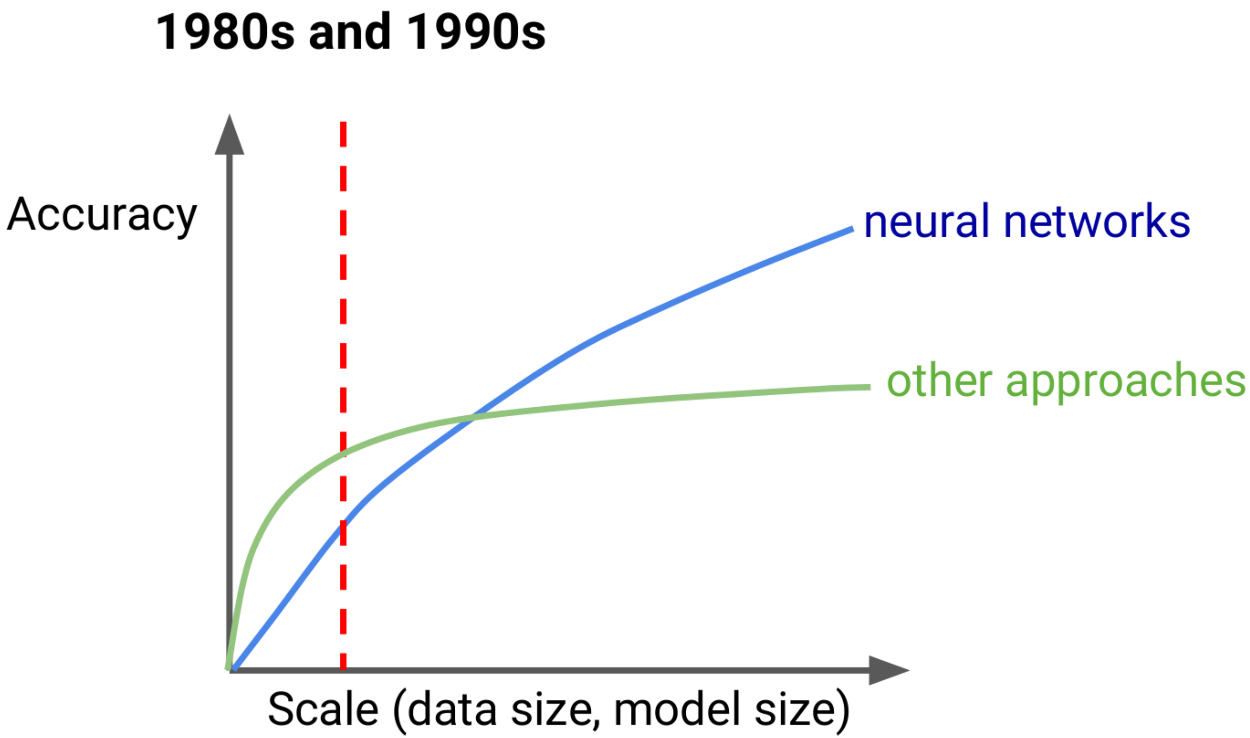
Convolutional network (LeCun et al, 1989)

Learning

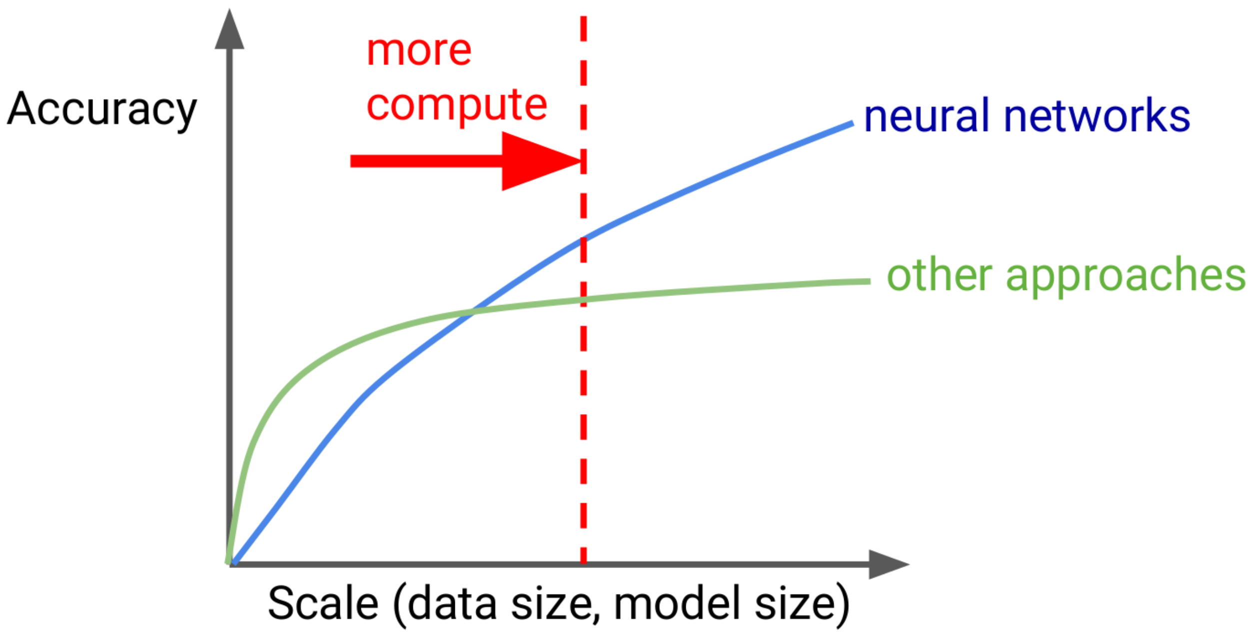
$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \mathcal{L}(\theta_t)$$


dog
bird
cat
deer
frog

Present

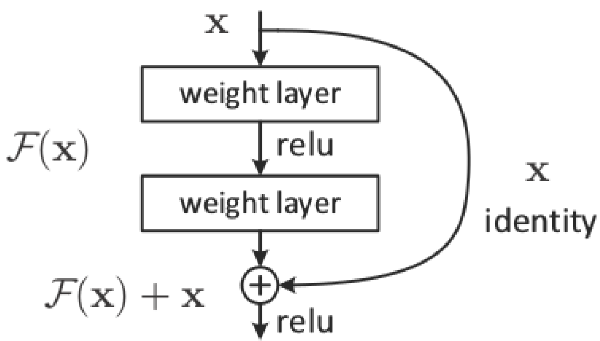

1980s and 1990s
Accuracy
neural networks
other approaches
Scale (data size, model size)

Now


Accuracy
more
compute
neural networks
other approaches
Scale (data size, model size)

What has changed?

Algorithms



Data

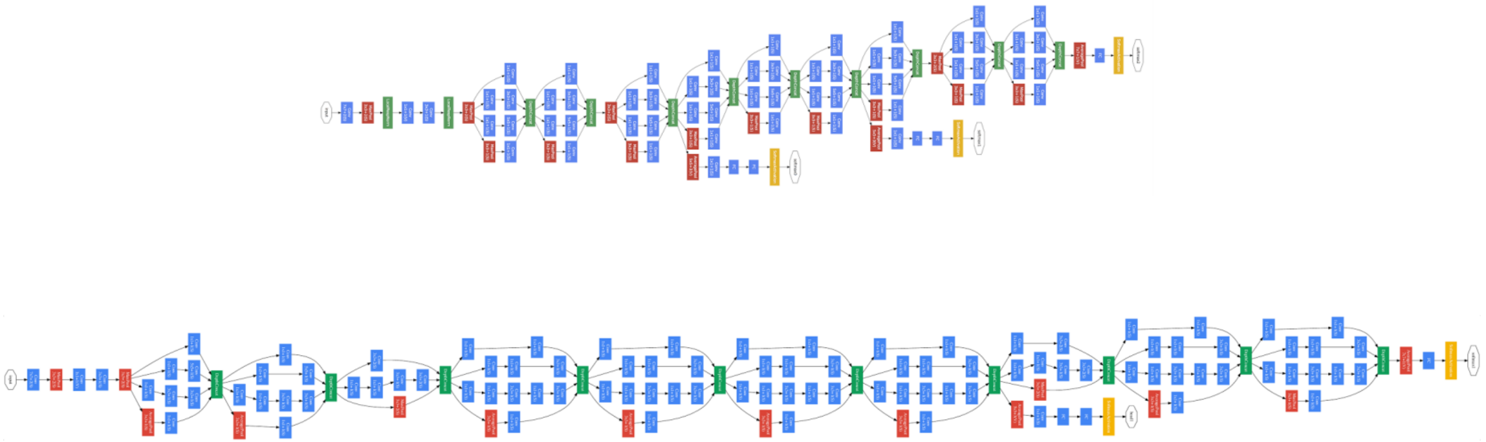
Software

Caffe2 · Microsoft CNTK · TensorFlow · theano

Chainer · PYTORCH · dmlc mxnet · K

Compute engines

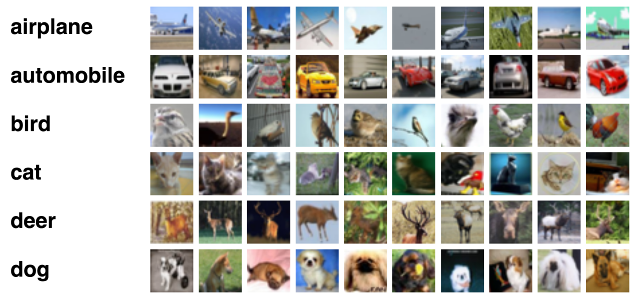
Depth



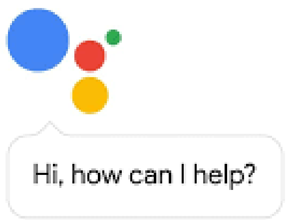
Szegedy et al, 2014

Beyond domain-based approaches

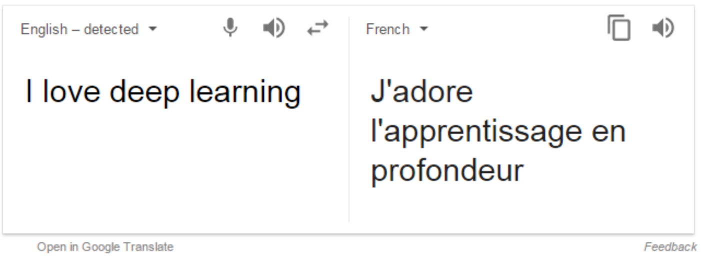
Pixel data
(e.g., visual recognition)



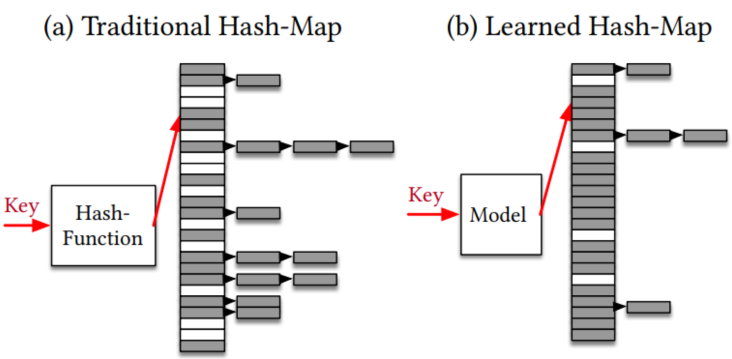
Audio data
(e.g., speech recognition and synthesis)



Text data
(e.g., machine translation)

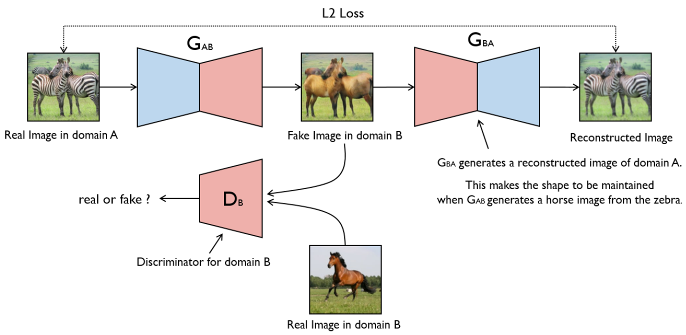


System applications
(e.g., databases)

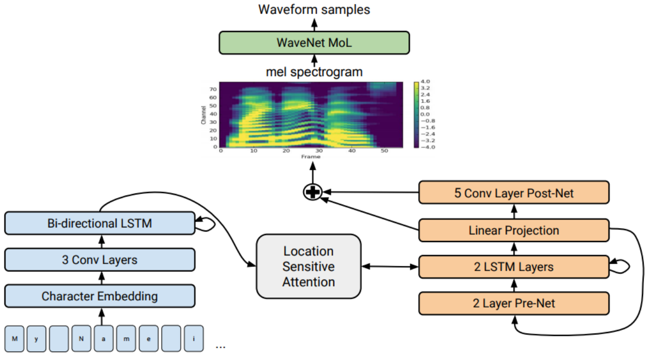


Beyond supervised learning

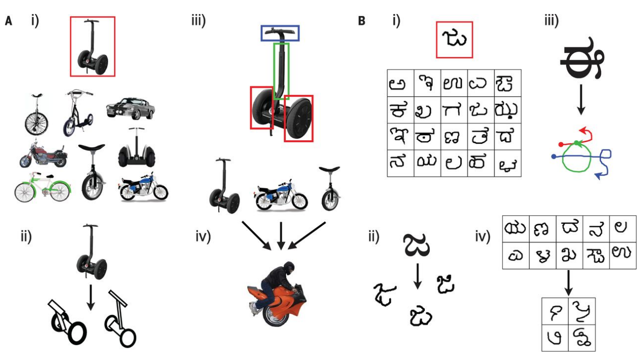
Adversarial training



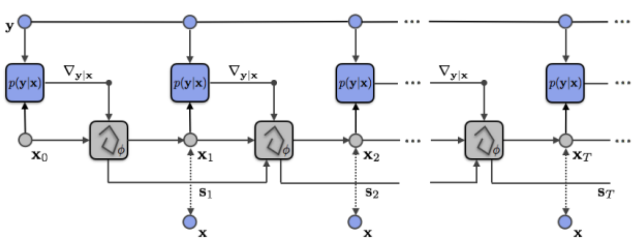
Generative models



Few-shot learning



Learning to learn



Autonomous cars (NVIDIA)

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

Future

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

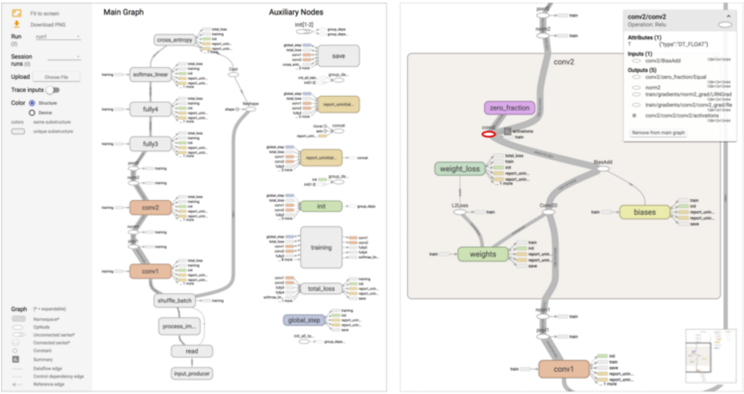
- Programs are written in languages such as Python, C or Java.
- They consist of explicit instructions to the computer written by a programmer.
- The programmer identifies a specific point in program space with some desirable behavior.

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.

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.

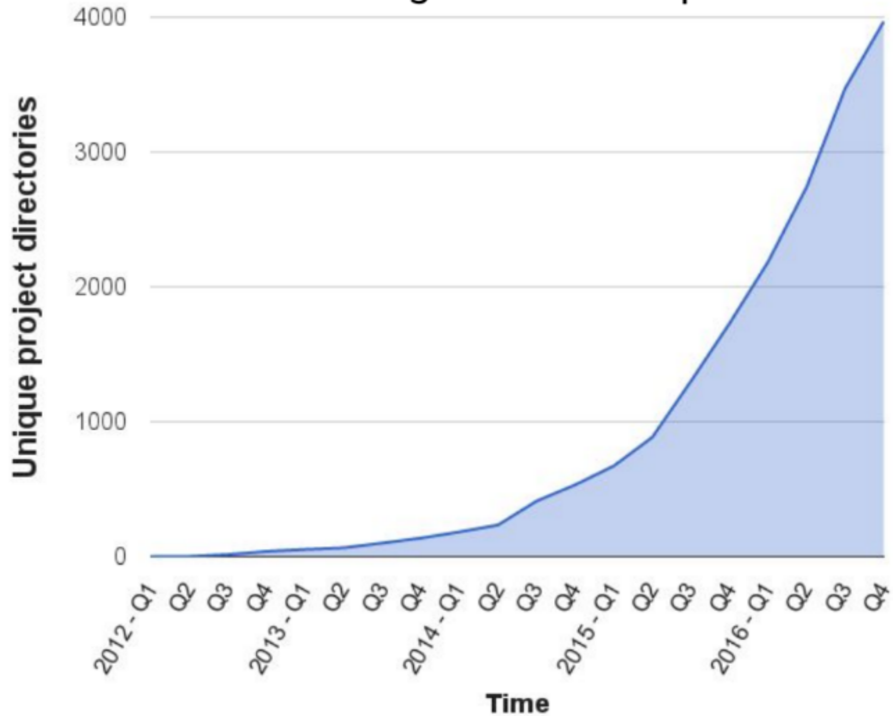
Therefore,

- 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.

Fundamentally, deep learning enables a new methodology towards problem solving.

Growing Use of Deep Learning at Google

of directories containing model description files



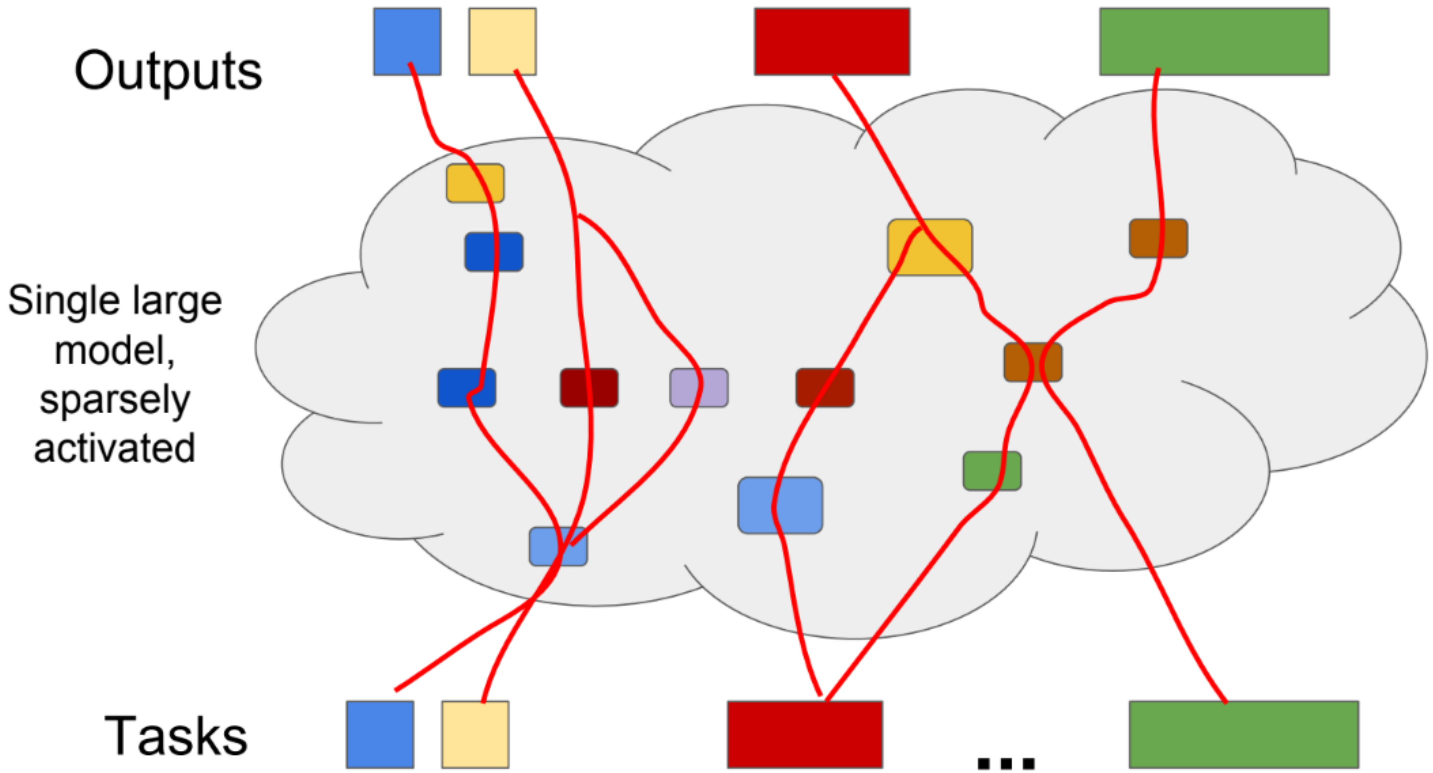
Across many products/areas:

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 and memory use
- It is highly portable
- It is very agile
- It is better than you

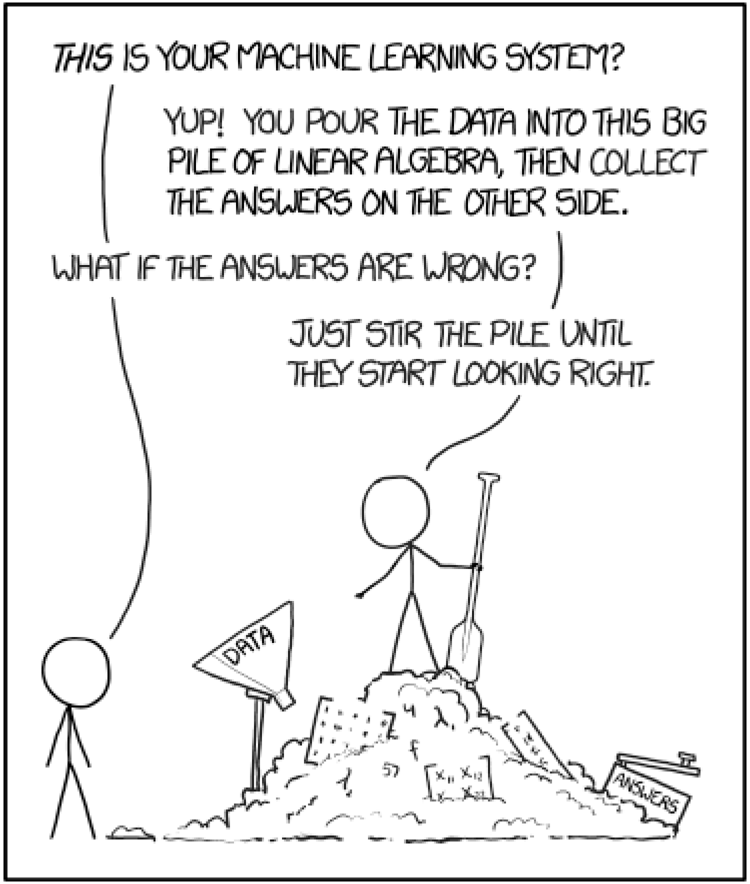


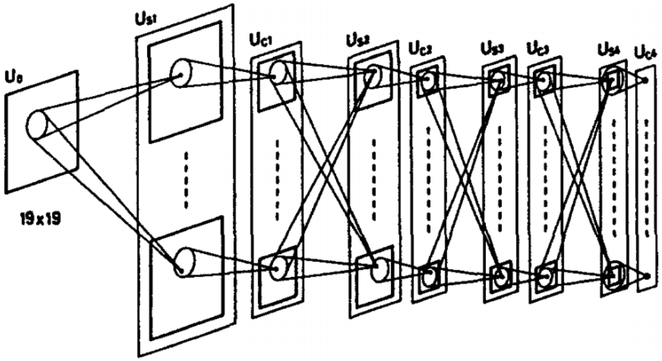
Modules can meld into an optimal whole (Jeff Dean, Lead of Google.ai, 2017)

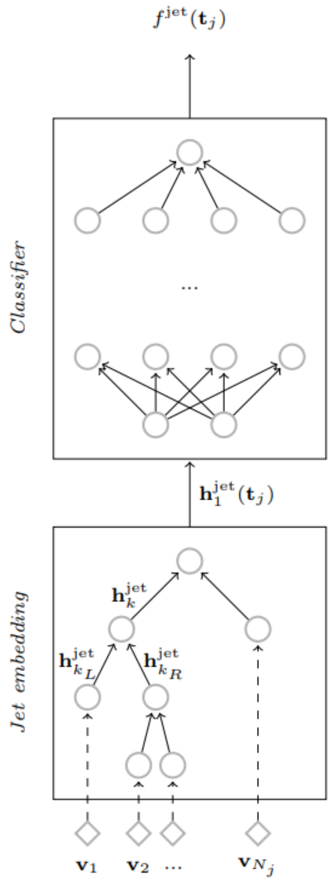
Trust

How do you trust systems made of opaque neural networks, for which domain knowledge seems to have disappeared?

- interpretability issues
- accountability issues
- security issues







Domain knowledge should not be abandoned.

Instead, use it to design neural networks,
thereby gaining in understanding and trust.

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.

The end.