Deep Learning

Past, present and future

Prof. Gilles Louppe
g.louppe@uliege.be
How to write a computer program that performs tasks what we can all easily do, yet all fail to describe precisely how?
The Perceptron (Rosenblatt, 1957)

\[ f(x; w, b) = \sigma(w^T x + b) \]
The Multi-Layer Perceptron (Rumelhart et al, 1986)
Convolutional networks

Hubel and Wiesel, 1962
10 output units

Layer H3
30 hidden units

Layer H2
12 x 16 = 192 hidden units

Layer H1
12 x 64 = 768 hidden units

256 input units

Fully connected
~ 300 links

Fully connected
~ 6000 links

~ 40,000 links from 12 kernels
5 x 5 x 8

~ 20,000 links from 12 kernels
5 x 5

Convolutional network (LeCun et al, 1989)
Learning

\[ \theta_{t+1} = \theta_t - \gamma \nabla_{\theta} \mathcal{L}(\theta_t) \]
Present
1980s and 1990s

Accuracy

Scale (data size, model size)

neural networks

other approaches
What has changed?

Algorithms

$$\mathcal{F}(x)$$

- weight layer
- relu

$$\mathcal{F}(x) + x$$

- weight layer
- identity
- relu

Data

Software

- Caffe2
- CNTK
- TensorFlow
- theano

Compute engines
Depth

Szegedy et al, 2014
Beyond domain-based approaches

- **Pixel data** (e.g., visual recognition)
  - airplane
  - automobile
  - bird
  - cat
  - deer
  - dog

- **Audio data** (e.g., speech recognition and synthesis)
  - Hi, how can I help?

- **Text data** (e.g., machine translation)
  - I love deep learning
  - J'adore l'apprentissage en profondeur

- **System applications** (e.g., databases)
  - (a) Traditional Hash-Map
  - (b) Learned Hash-Map
    - Key
    - Hash-Function
    - Model
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.

```c
string input;
int length, IN;
double dblTemp;
bool again = true;

while (again) {
    IN = 1;
    again = false;
    getline(stdin, input);
    system("cls");
    stringstream(input) >> dblTemp;
    length = string length();
    if (length < 4) {
        again = true;
        continue;
    } else if (input[length - 3] != '.') {
        again = true;
        continue;
    }
    while (valid < length) {
        if (isdigit(input[valid]) == 1) {
            continue;
        } else if (valid < length - 1) {
            continue;
        }
    }
}
```
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.

```
string sigma; int length, M;
double dlength;
bool again = true;
while (again) {
    if (again) {
        again = false;
        getline(cin, sigma);
        system("chmod 777 ");
        stringstream sigma>>dlength;
        length += dlength.length();
        if (length > 40) {
            again = true;
            continue;
        } else if ((length[0] == 'i' && length < 0) || length < 0) {
            again = true;
            again = true;
            continue;
        } while (sigma < length) {
            if (isdigit(sigma)) {
                if (isdigit((sigma - 2))) {
                    if (isdigit((sigma - 3))) {
                        break;
                    } else {
                        continue;
                    }
                } else {
                    break;
                }
            } else {
                break;
            }
        }
    }
    cin >> sigma;
    if (sigma[0] == 'i') {
        if (isdigit(sigma)) {
            if (isdigit((sigma - 2))) {
                if (isdigit((sigma - 3))) {
                    break;
                } else {
                    continue;
                }
            } else {
                break;
            }
        } else {
            break;
        }
    }
}
```
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.