Course 1: Generalities about AI

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Global overview…

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview…

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview...

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Global overview…

What is AI?

- **Intelligence**: ability to **extract knowledge** from observations
- This knowledge is used to **solve tasks in different contexts and environments** (automation)

Old way: Memorize

- Human experts code the machines
- Goods: we know what we are doing.
- Bads: requires **explicit** solutions (not available for some problems).

Modern way: Generalize

- Let machines teach themselves how to solve a problem (**implicit**).
- Goods: universally applicable
- Bads: lack of understandability/robustness.
- Requires **training**.

Memorizing (explicit) **vs** Learning (implicit)

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples ...

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples …

Generalization and machine learning

Machine learning

- Supervised: Infer a function from inputs/outputs
- Difficulties:
  - Ill-posed problem (infinity of potential solutions)
  - Main approach: seek for particular solutions

Generalization

- Generalization refers to the ability to infer **good decisions or representations** from **examples**, **trials** and/or **interactions**.
- In computer science, we talk about **machine learning**.

Examples

- Learning to play chess through playing games,
- Learning to recognize dogs and cats in images from annotated examples ...

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

- $f = f_W$ contains **parameters W** to be trained,
- In most cases, an ideal f_W exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

- $f = f_W$ contains **parameters W** to be trained,
- In most cases, an ideal f_W exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

Input/output

- Goal: infer a function from an input (often tensor) space to an output (often tensor) space, $\mathbf{y} = f(\mathbf{x})$,
- Example: input can be an image, output a vector where the largest value indicate the category the image belongs to.

Error/Loss

- Loss $\mathcal{L}$: nonnegative measure of the discrepancy between expected output $\hat{\mathbf{y}}$ and obtained output $\mathbf{y}$.
- Example: output should be $[0, 1]$ but is $[0.2, 0.8]$.

Parameters

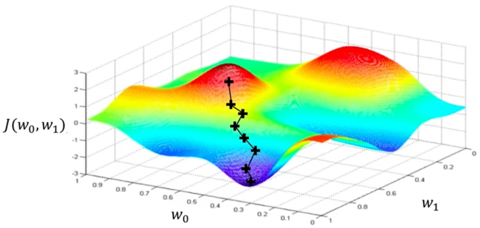
- $f = f_w$ contains **parameters W** to be trained,
- In most cases, an ideal f_w exists but is **hard to find in practice**,
- Learning is a **regression ill-posed** problem.

Global formalism

- Loss: $J(\mathbf{W}) = \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}, \mathbf{W}), \mathbf{y}^{(i)}), i = examples$
- Model parameters: $\mathbf{W}^* = argmin(J(\mathbf{W}))$

Training Algorithm

- Randomly Initialize model weights
- Compute Gradient of the Loss $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Update weights $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Repeat until convergence

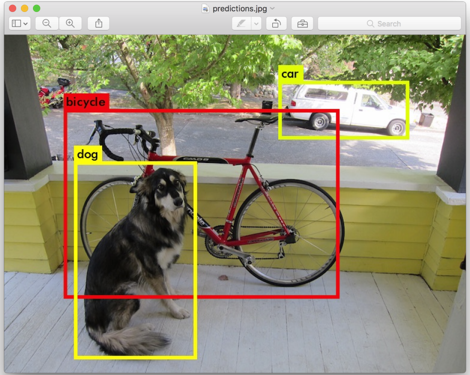


from MIT course `introtodeeplearning.com`

Main application domains of AI

Vision

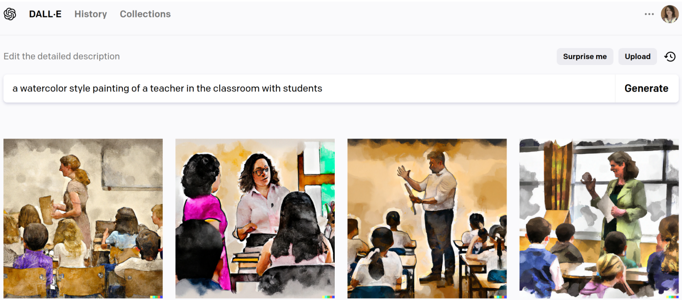
- Object/face recognition,
- Detection,
- Autonomous vehicles,
- Automatic diagnostic,
- Defects identification,
- Video applications...



Main application domains of AI

Recent breakthrough: Image Synthesis

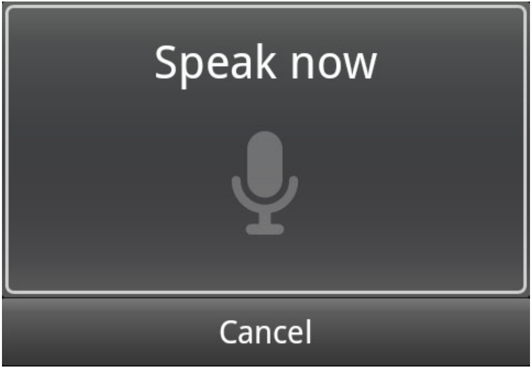
- Stable Diffusion Models: state of the art image generation (https://arxiv.org/abs/2112.10752)
- CLIP: Learning visual models from Natural Language Supervision (https://arxiv.org/abs/2103.00020)
- DALL-E (https://labs.openai.com/), Midjourney (https://midjourney.gitbook.io/docs/), Stable Diffusions



Main application domains of AI

Natural Language Processing (NLP)

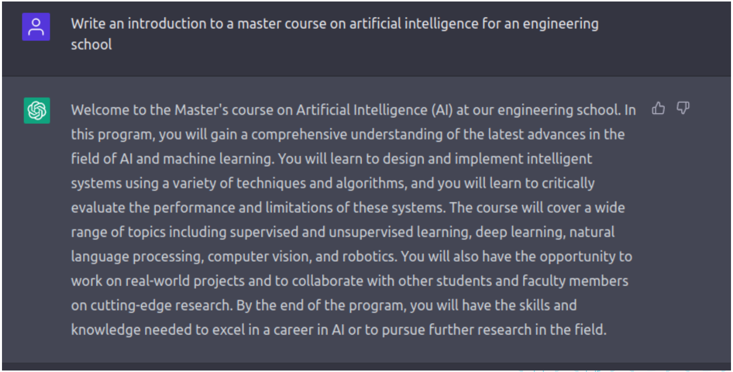
- Automatic assistant,
- Voice-to-text,
- Automatic translation,
- Automatic summarizing,
- Sentiment analysis,
- Text indexing…



Main application domains of AI

Recent breakthrough: Text Synthesis with Large Language Models

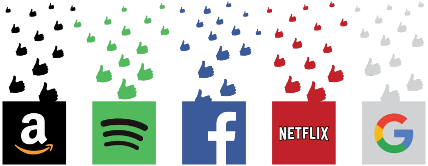
- GPT-3, 175 billion parameters trained without explicit supervision on over 600GB of text `https://arxiv.org/abs/2005.14165`
- ChatGPT `https://openai.com/blog/chatgpt/`



Main application domains of AI

Tons of other domains…

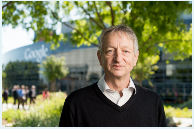
- Medical imaging,
- Decision aid,
- Data mining,
- Visualization,
- Recommender systems,
- Market analysis…



https://alphafold.ebi.ac.uk/

The great elders of modern AI (Turing Prize 2018)

Geoffrey Hinton



- Cognitive psychologist and computer scientist,
- Prof. at University of Toronto and works for Google,
- Known for back-propagation and Boltzmann machines.

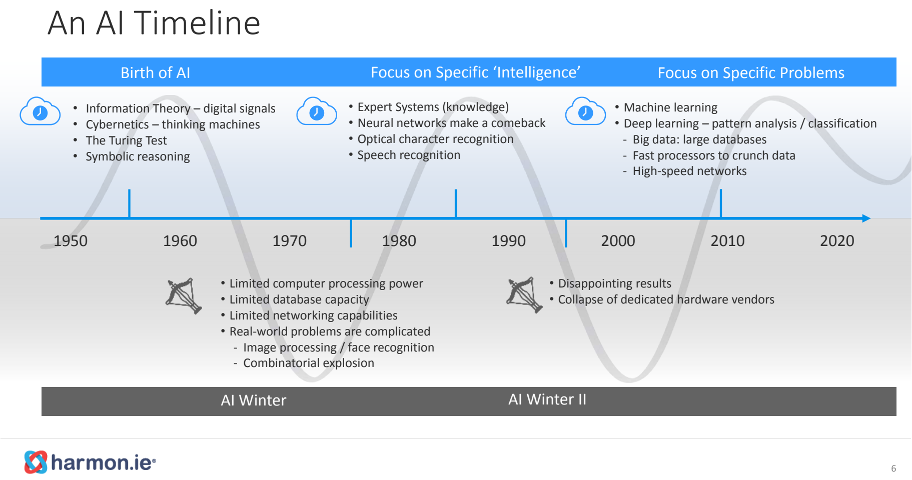
Yoshua Bengio

- Computer scientist,
- Prof. at Université de Montréal and head of MILA,
- Known for his work on deep learning.

Yann le Cun

- Computer scientist,
- Prof. at New York University then he joins FAIR,
- Known for his work on back-propagation and CNNs.

AI Timeline


An AI Timeline
Birth of AI
Focus on Specific 'Intelligence'
Focus on Specific Problems
• Information Theory – digital signals
• Cybernetics – thinking machines
• The Turing Test
• Symbolic reasoning
• Expert Systems (knowledge)
• Neural networks make a comeback
• Optical character recognition
• Speech recognition
• Machine learning
• Deep learning – pattern analysis / classification
- Big data: large databases
- Fast processors to crunch data
- High-speed networks
1950
1960
1970
1980
1990
2000
2010
2020
• Limited computer processing power
• Limited database capacity
• Limited networking capabilities
• Real-world problems are complicated
- Image processing / face recognition
- Combinatorial explosion
• Disappointing results
• Collapse of dedicated hardware vendors
AI Winter
AI Winter II
harmon.ie
6

Where did the revolution in AI come from?

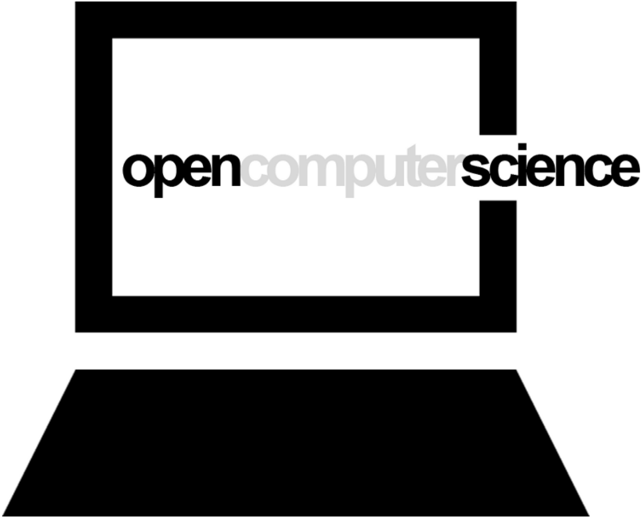
- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.

Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.

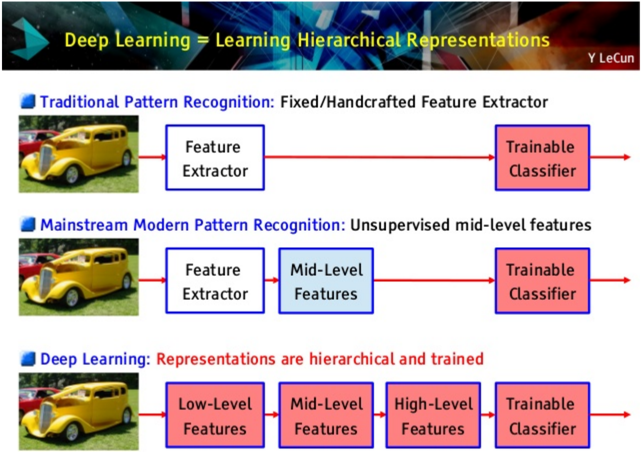
Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.



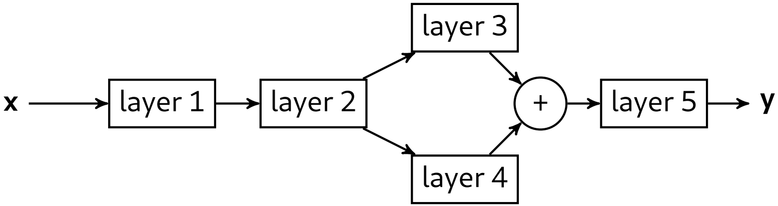
Where did the revolution in AI come from?

- The use of GPUs for computation.
- The share of huge datasets on Internet.
- Github/Arxiv new ways of sharing research.
- The return of representation learning.



Main idea

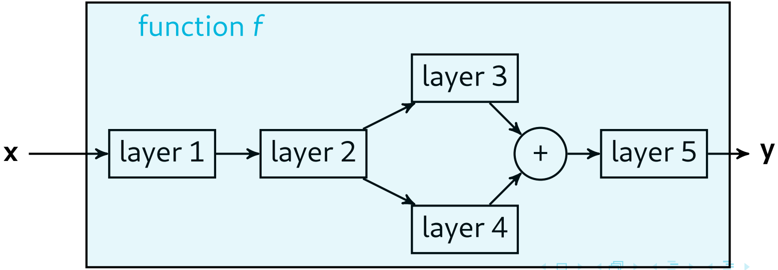
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

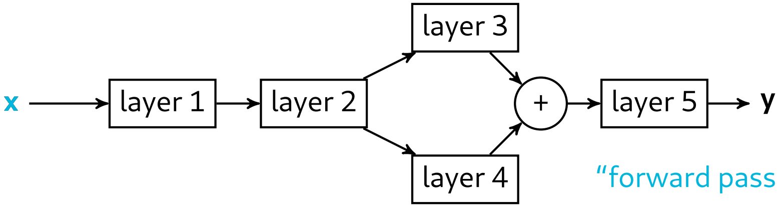
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)

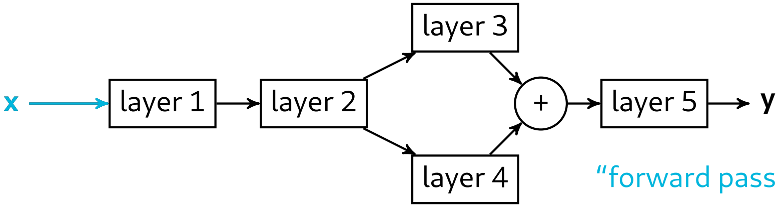


“forward pass”

Deep learning

Main idea

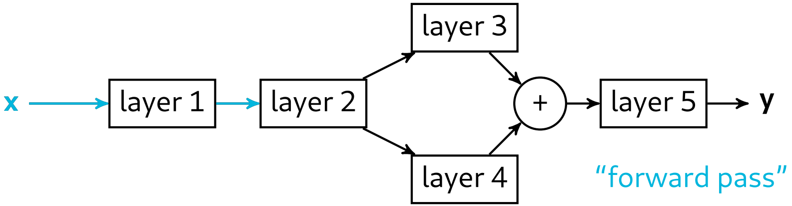
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



"forward pass"

Main idea

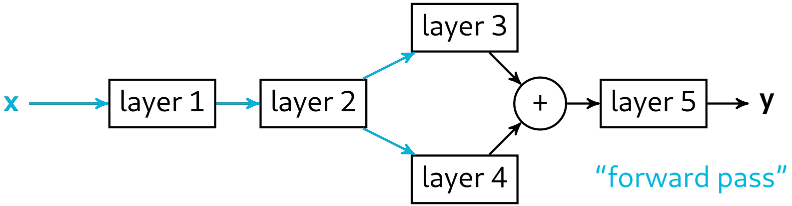
- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

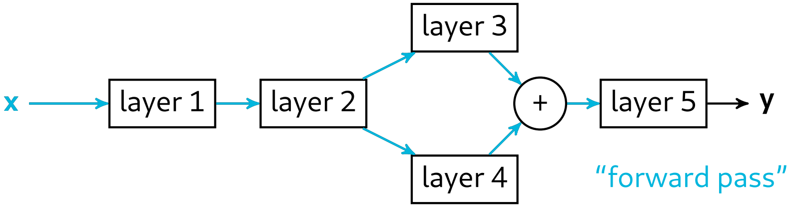
Main idea

- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



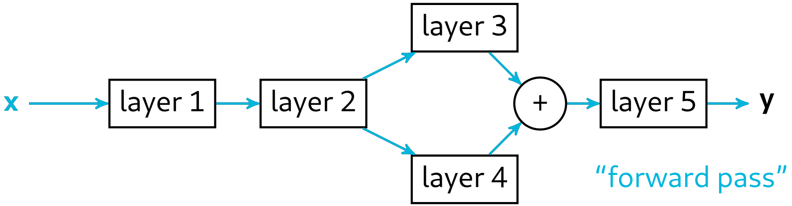
Main idea

- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Main idea

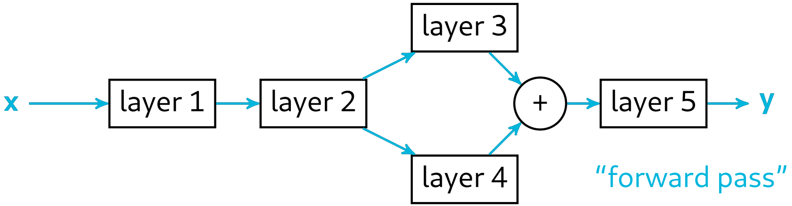
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

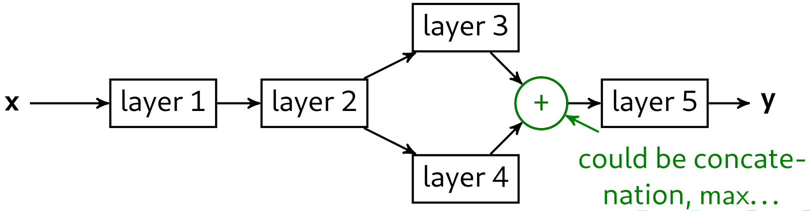
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

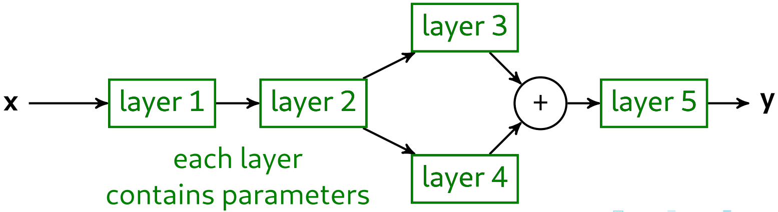
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

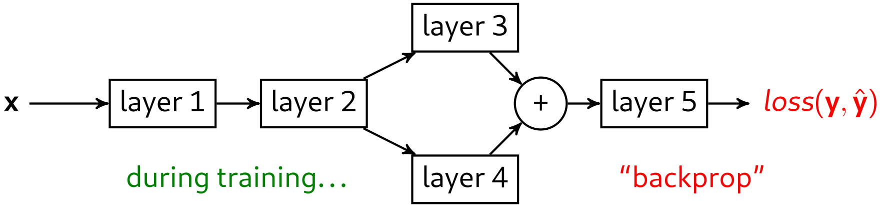
Main idea

- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Main idea

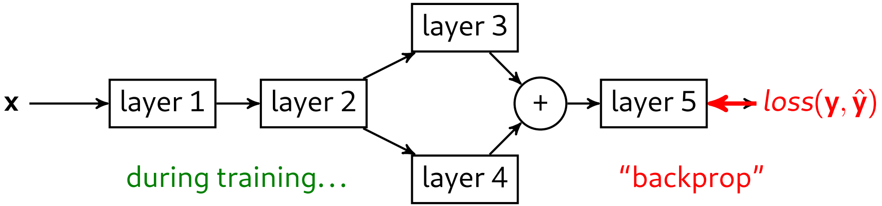
- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

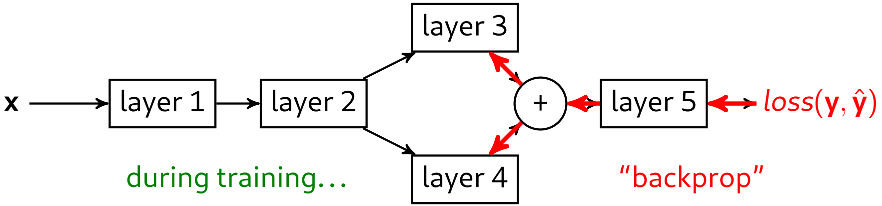
Main idea

- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



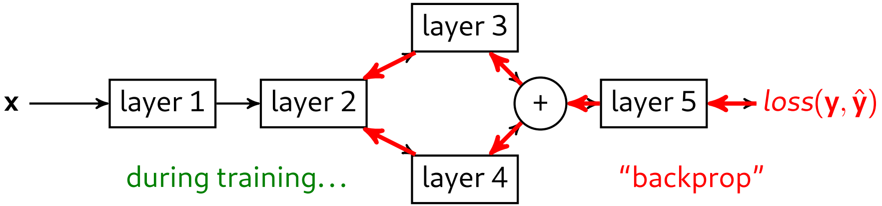
Main idea

- Compositional Approach: Instead of directly mapping $\mathbf{x}$ to $\mathbf{y}$, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Main idea

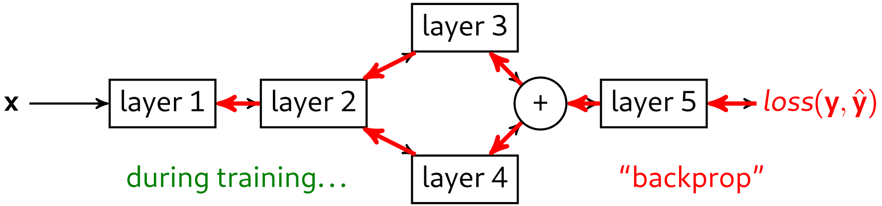
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

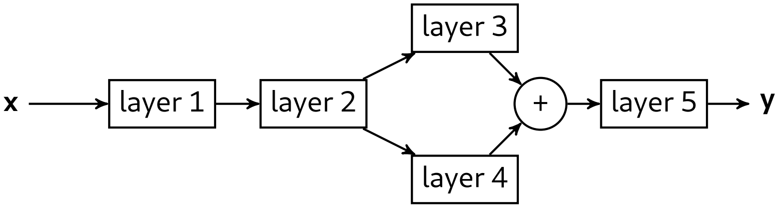
- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Deep learning

Main idea

- Compositional Approach: Instead of directly mapping **x** to **y**, express solutions as an assembly of simple mathematical functions called **layers**
- End-to-end learning: Tune all atomic functions together
- Training: Backpropagate throughout the architecture (to compute the gradient of the loss wrt all layers parameters)



Number of layers, choice of the architecture are **hyperparameters**

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

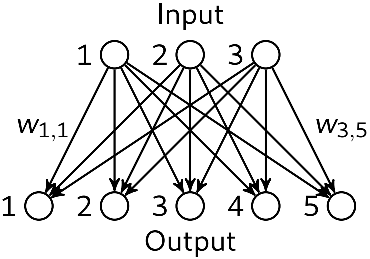
- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

Fully connected layer



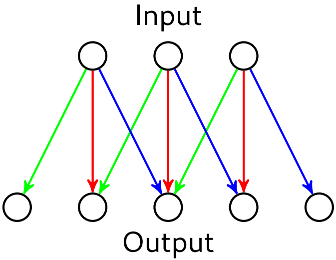
$$\begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} \end{pmatrix}$$

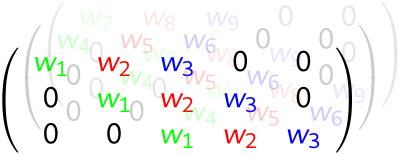
Some additional details

Layers

- $\mathbf{x} \mapsto h(\mathbf{W}\mathbf{x} + \mathbf{b})$.
  - h is a nonlinear parameterwise function (often without parameters),
  - $\mathbf{W}$ is a tensor:
    - Can be agnostic of the structure: **fully-connected layers**,
    - Can be structure-dependent: **convolutional layers**.

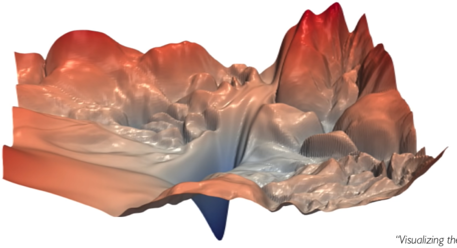
Convolutional layer





Some additional details

Training Neural Networks is Difficult

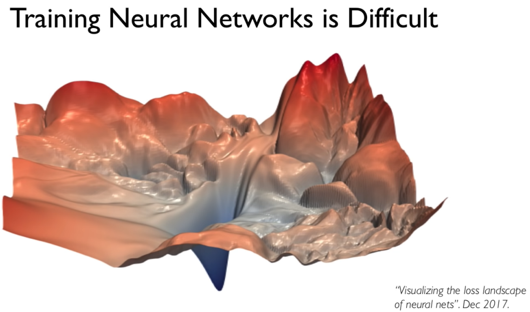


"Visualizing the loss landscape of neural nets". Dec 2017.

Optimization with Differentiable Algorithmic

- Learning rate η : $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Variants of the **Stochastic Gradient Descent (SGD)** algorithm are used:
  - Use of **moments**,
  - Use of **regularizers**.

Some additional details



Batches

- To accelerate computations, inputs are often treated **concurrently** using small **batches**.

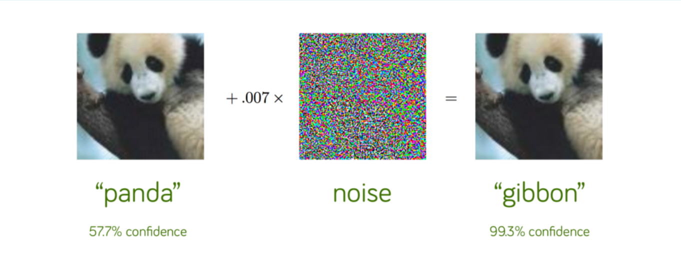
Some key open challenges (core AI research)

Learning from few examples / few shots / few labels

"How to grow a mind: statistics, structure, and abstraction", Science, 2011.

Some key open challenges (core AI research)

Learning what should be learned (robustness / adversarial attacks)

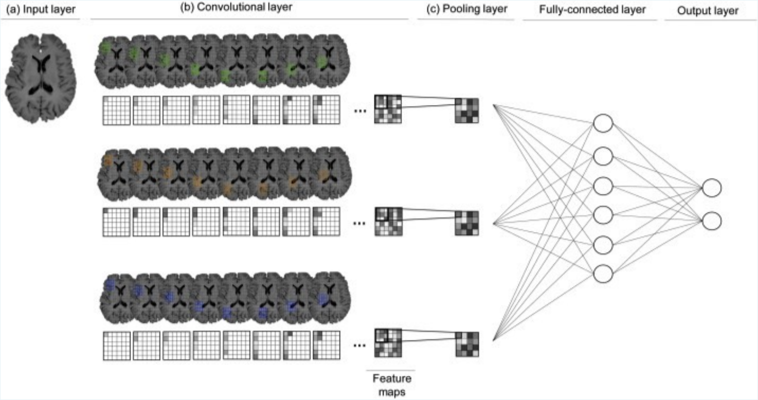


Random noise added to input images can dramatically change the result.

"Intriguing properties of neural networks", Arxiv research report, 2013.

Some key open challenges (core AI research)

Interpretability



A trained model might be very accurate, but how does it take its decision ?

“Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications“, Vieira et al. 2017.

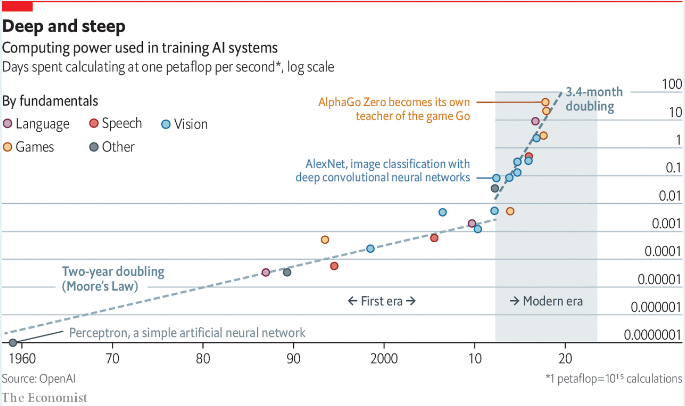
Some key open challenges (core AI research)

Incremental (or continual) learning

Adding new classes of object one by one without forgetting previous knowledge.

Some key open challenges (core AI research)

Computational and memory footprints



Training a large algorithm: thousands to millions of parameters using Gigabytes of data.

Course organisation

Sessions

1. Generalities about AI (today),
2. Supervised learning,
3. Unsupervised learning,
4. Combinatorial game theory,
5. Reinforcement learning,
6. Work on final project,
7. Work on final project,
8. Ethics in AI + Challenge.

Sessions schedule

Each session has (roughly) the same structure:

- Short written exam about the previous lesson (10 min),
- Short lesson (20 min),
- Lab Session ,
- Project
- Sessions 2, 3, 4 and 8 include students' presentations

Course Evaluation

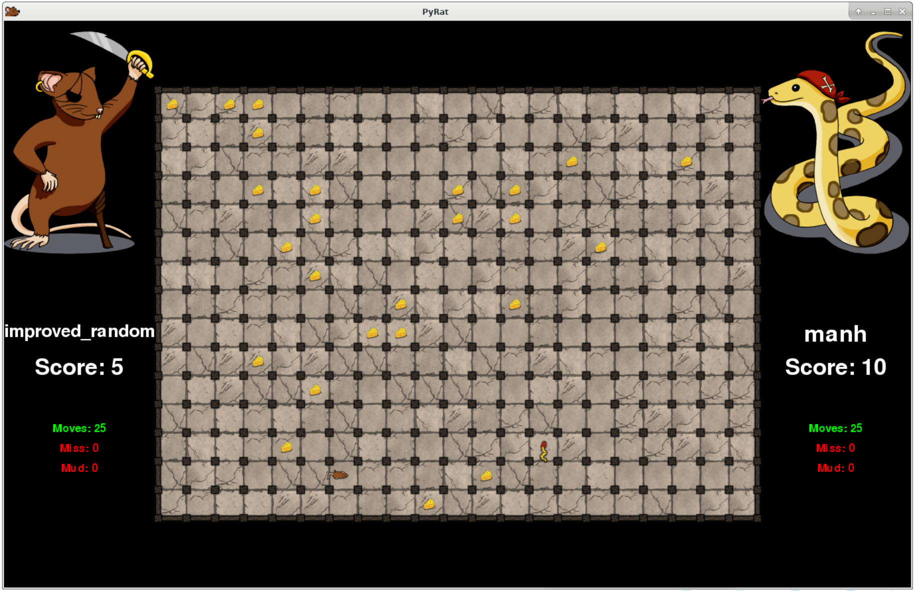
Lab Sessions and Challenge

- By groups of two, you will be working on campus machines (python, deeplearningu20 environment).
- Lessons, practicals and presentations in room K01-112

Evaluations

- QCM on moodle at 9:30 am **compulsory**
- Presentations - sessions 2,3,4 + **Final Deliverable (Challenge)** - session 8

Non-symmetric PyRat without walls / mud



Lab Session 1 and assignment

Lab Session

- Introduction to Jupyter Notebook, numpy, scipy
- Generating Pyrat games
- Visualisation using Matplotlib

Project 0 (P0) (oral presentation)

You will choose a topic on an application of AI.
You have to prepare a 7 minutes presentation (for session 2) in which you quickly explain :

- What the topic is about
- What solutions already exist
- Examples of companies / existing products on this topic
- Example of ethical considerations related to the topic
- Current limitations and hard problems