Course 4: Deep Learning

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

Summary

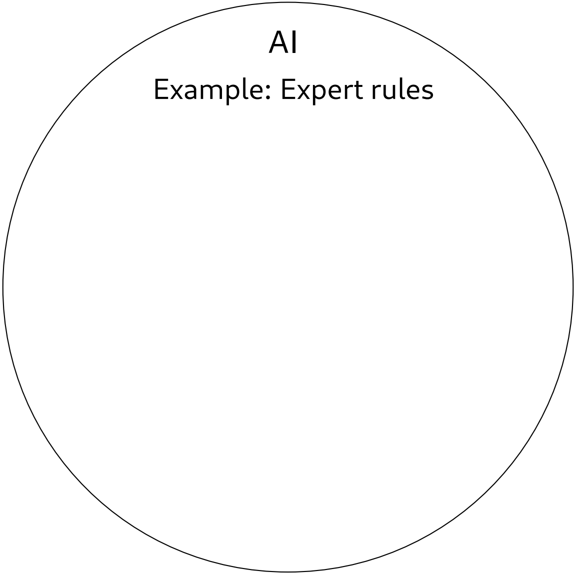
Last session

1. Unsupervised learning - discover structure from unlabeled data
2. Clustering
3. Decomposition - sparse dictionary learning
4. Practical ethics

Today's session

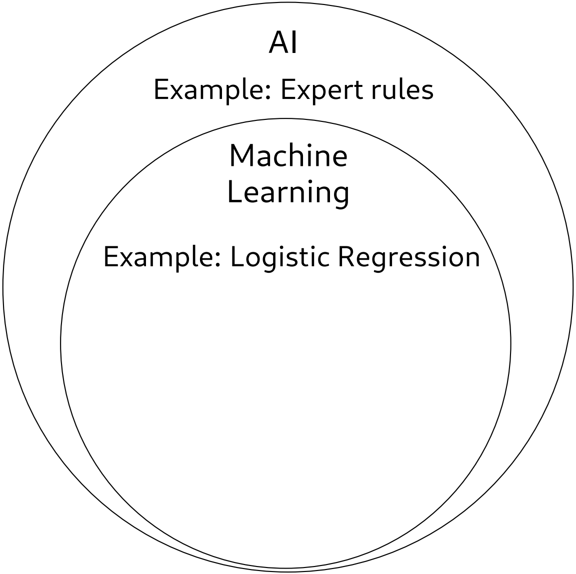
- Multi-Layer Perceptron
- Convolutional Neural Networks
- Transformers

Global overview…



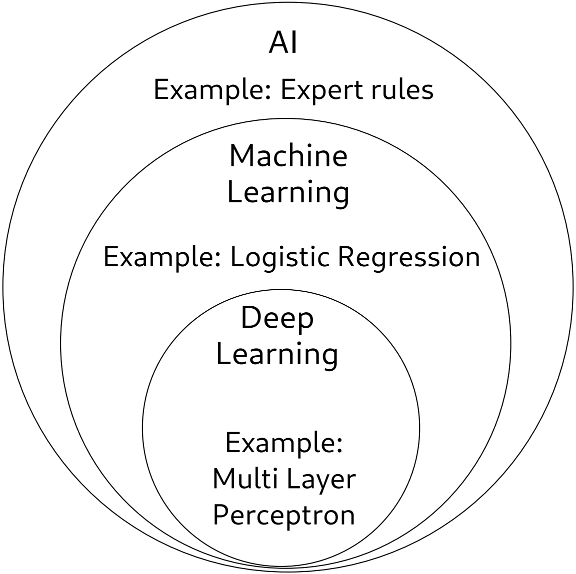
Adapted from *Deep Learning, 2016* by A. Courville, I. Goodfellow and Y. Bengio

Global overview...



Adapted from *Deep Learning, 2016* by A. Courville, I. Goodfellow and Y. Bengio

Global overview…



Adapted from *Deep Learning, 2016* by A. Courville, I. Goodfellow and Y. Bengio

Deep Learning in a nutshell (1/3)

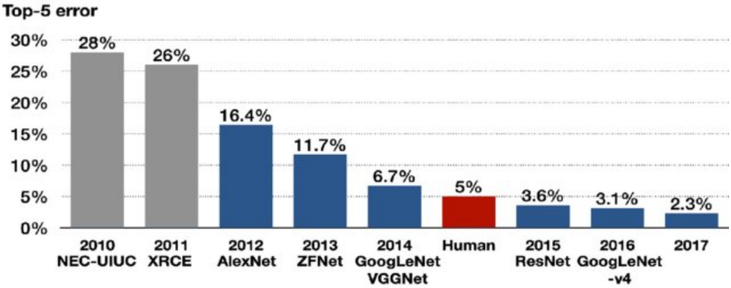
Definition

Using *deep* Artificial Neural Networks.

We generally talk about “Neural Networks instead of “Artificial Neural Networks” (but this is the most correct terminology!)

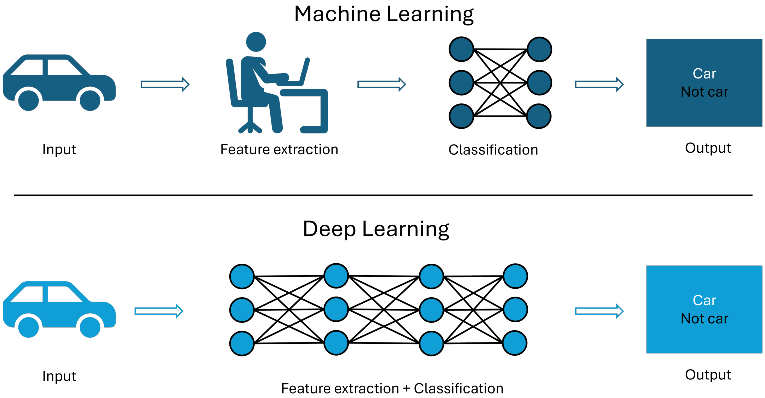
The strength of Deep Learning lies in using a lot *(a **lot**)* of data.

A major breakthrough in image classification:



Source: Kang, D. Y., Duong, H. P., & Park, J. C. (2020). Application of deep learning in dentistry and implantology. Journal of implantology and applied sciences, 24(3), 148-181.
Details for the human evaluation: Russakovsk, Dieg *et al.*. ImageNet Large Scale Visual Recognition Challenge, https://arxiv.org/pdf/1409.0575.pdf

Deep Learning in a nutshell (3/3)



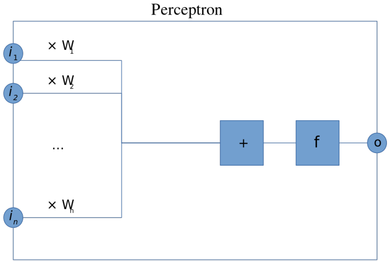
Inspired from `https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/`

Outline

Perceptron

1943, implementation in 1957

Perceptron is a nonlinear operation in which weights W are trainable.



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

Source: By Mat the w at English Wikipedia, CC BY-SA 3.0, `https://commons.wikimedia.org/w/index.php?curid=23766733`

Perceptron

1943, implementation in 1957

Perceptron is a nonlinear operation in which weights W are trainable.

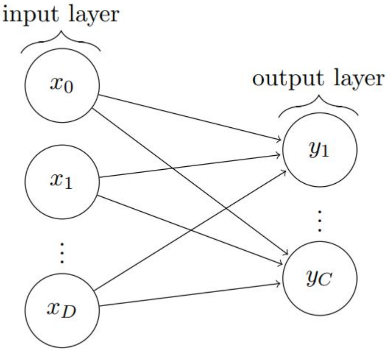


Figure: The arrows represent the weights W.

Source: `https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/`

Optimizing the weights

Loss

- Prediction: $y = f\left(\sum_{d=0}^{D} x_d W_d\right)$
- Ground truth: $\hat{y}$
- Loss (one example:) $\mathcal{L}(x, W, \hat{y}) = d(y, \hat{y})$
  (ex: $d(y, \hat{y}) = \|y - \hat{y}\|_2^2$)
- Loss (i examples): $J(W) = \sum_i \mathcal{L}(x^{(i)}, W, \hat{y}^{(i)})$

Gradient descent

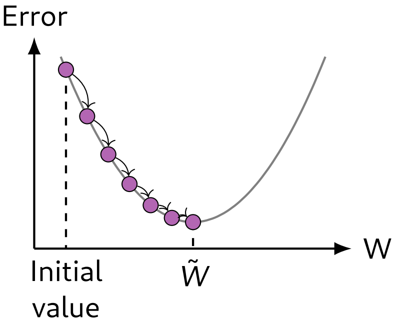
- Compute the gradient: $\frac{\partial J(W)}{\partial W}$ (high dimensional derivative)
- Update weights: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

Gradient descent

Intuition behind the gradient descent

Update is given as: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

- $\partial J(W)$ gives the direction
- η gives the size of the step



Adapted from `https://tex.stackexchange.com/questions/561921/replicating-a-plot-using-tikz`

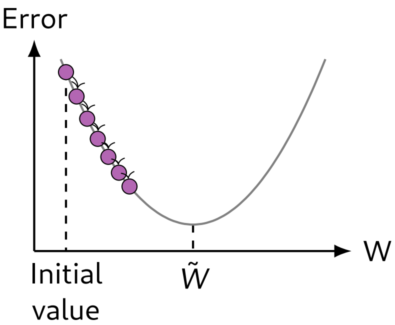
Gradient descent

Intuition behind the gradient descent

Update is given as: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

- $\partial J(W)$ gives the direction
- η gives the size of the step

Small step:

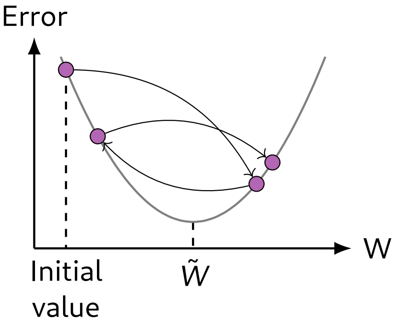


Intuition behind the gradient descent

Update is given as: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

- $\partial J(W)$ gives the direction
- η gives the size of the step

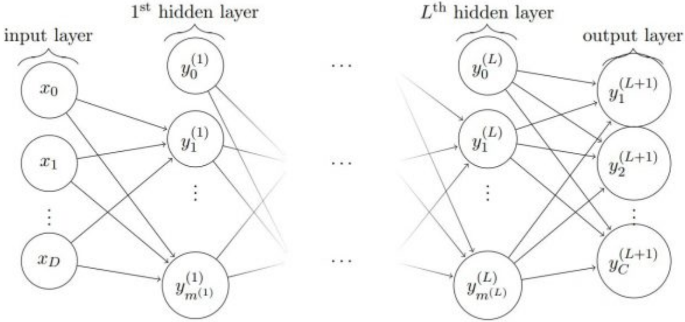
Large step:



Multi-Layer Perceptron

Multi-Layer Perceptron (= *fully-connected* network)

- Stacking Perceptrons.
- The ***deep*** term comes form this stacking
- Prediction: $y = f\left(W^{(L)} \cdots f\left(W^{(2)} f(W^{(1)} x)\right)\right)$



Source: `https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/`

Neural Networks

Definitions

$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element i of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function f is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function f, it is standard that each layer uses the same.

Neural Networks

Definitions

$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element i of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function f is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function f, it is standard that each layer uses the same.

Definitions

$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element i of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function f is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function f, it is standard that each layer uses the same.

Neural Networks

Definitions

$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element i of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function f is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function f, it is standard that each layer uses the same.

Neural Networks

Definitions

$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element *i* of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function *f* is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function *f*, it is standard that each layer uses the same.

Neural Networks

Definitions

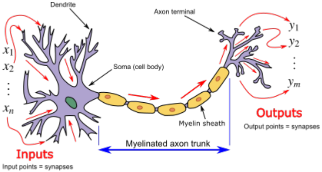
$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- Each building block $\mathbf{y}^{(l+1)}$ is called a **layer**.
- One element i of a layer ($y_i^{(l)}$) is called a **neuron** (both as input and output).
- The nonlinear function f is called the **activation function**.
- $\mathbf{W}^{(l)}$ are called **weights**.
- $\mathbf{b}(l)$ is called the **bias**.

Note: while each layer can have a different activation function f, it is standard that each layer uses the same.

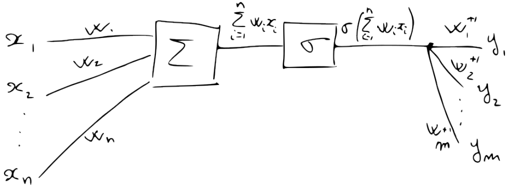
Why is it called Neural Network?

"Neurons" may be seen as **loosely** inspired from the human brain.



Attribution: Egm4313.s12 at English Wikipedia,

`https://en.wikipedia.org/wiki/Neural_network_(machine_learning)#/media/File:Neuron3.png`



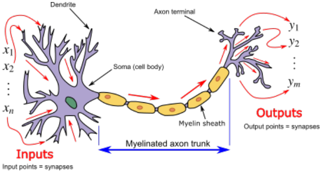
Detail of one neuron in an Artificial Neural Network.

* Late note: I forgot the bias in the graphic.

This is an **analogy**, artificial neural networks are **not** following the human brain (in general).

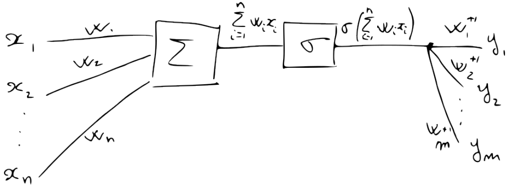
Why is it called Neural Network?

"Neurons" may be seen as **loosely** inspired from the human brain.



Attribution: Egm4313.s12 at English Wikipedia,

`https://en.wikipedia.org/wiki/Neural_network_(machine_learning)#/media/File:Neuron3.png`



Detail of one neuron in an Artificial Neural Network.

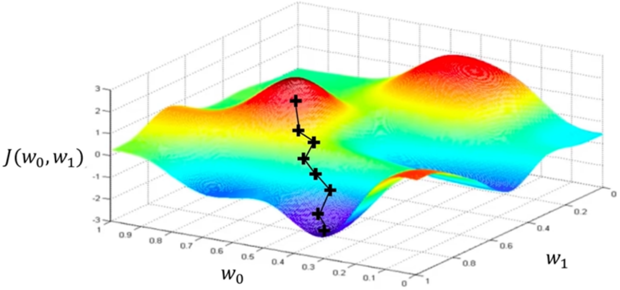
* Late note: I forgot the bias in the graphic.

This is an **analogy**, artificial neural networks are **not** following the human brain (in general).

Gradient descent for deep neural networks

Backpropagation

- Gradient descent for all layers (chain rule).
- Simplified equation: $\frac{\partial J(W)}{\partial W} = \frac{\partial J(W)}{\partial W^{(L)}} \frac{\partial W^{(L)}}{\partial W^{(L-1)}} \frac{\partial W^{(L-1)}}{\partial W^{(L-2)}} \cdots \frac{\partial W^{(2)}}{\partial W^{(1)}}$
- The error **backpropagates** through the network (reverse path)
- Computationally efficient, but finds a local minimum (at best)



Source: `http://introtodeeplearning.com/`

Batch

Batch

- The i examples are divided in *batches* (small excerpt)
- Allows one to train without loading the whole dataset in memory
- Accelerate the learning phase

Limits of MLP

Limits of Multi-Layer Perceptrons

- Computationally heavy for large inputs
- Large number of parameters: prone to overfitting
- No notion of structure in the input: everything is a vector

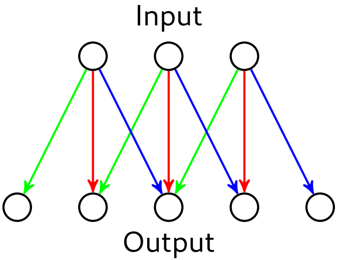
Outline

Convolutional Neural Network (1/5)

Principle

- Applying a kernel to the input, on small parts of the image at a time.
- Weights of the kernel are **learned** and **shared**!
- 2D convolution was a game changer for image processing
- Translation invariance

Convolutional layer



$$\begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{pmatrix}$$

Example of 2D convolution:

$$\underbrace{\begin{pmatrix} 0 & 1 & 1 & 1_{\times 1} & 0_{\times 0} & 0_{\times 1} & 0 \\ 0 & 0 & 1 & 1_{\times 0} & 1_{\times 1} & 0_{\times 0} & 0 \\ 0 & 0 & 0 & 1_{\times 1} & 1_{\times 0} & 1_{\times 1} & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{I} * \underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}}_{K} = \underbrace{\begin{pmatrix} 1 & 4 & 3 & \boxed{4} & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}}_{I * K}$$

Source: `https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz`

Example of 2D convolution:

$$
\underbrace{\begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0_{\times 1} & 0_{\times 0} & 0_{\times 1} & 1 & 1 & 0 & 0 \\
0_{\times 0} & 0_{\times 1} & 1_{\times 0} & 1 & 0 & 0 & 0 \\
0_{\times 1} & 1_{\times 0} & 1_{\times 1} & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}}_{I}
*
\underbrace{\begin{pmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{pmatrix}}_{K}
=
\underbrace{\begin{pmatrix}
1 & 4 & 3 & 4 & 1 \\
1 & 2 & 4 & 3 & 3 \\
1 & 2 & 3 & 4 & 1 \\
\boxed{1} & 3 & 3 & 1 & 1 \\
3 & 3 & 1 & 1 & 0
\end{pmatrix}}_{I * K}
$$

Source: `https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz`

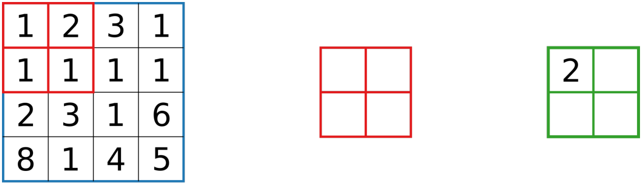
Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\underset{I}{\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0_{\times 1} & 0_{\times 0} & 0_{\times 1} \\ 0 & 1 & 1 & 0 & 0_{\times 0} & 0_{\times 1} & 0_{\times 0} \\ 1 & 1 & 0 & 0 & 0_{\times 1} & 0_{\times 0} & 0_{\times 1} \end{pmatrix}} * \underset{K}{\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}} = \underset{I * K}{\begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & \boxed{0} \end{pmatrix}}$$

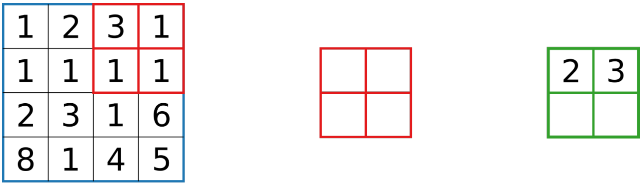
Source: `https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz`

Example of 2D pooling:



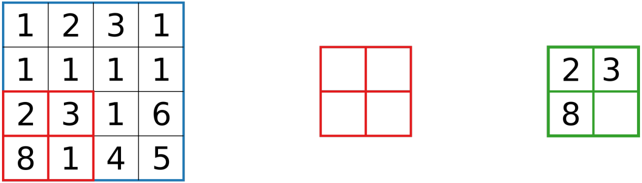
maxpool, kernel 2, stride 2

Example of 2D pooling:



maxpool, kernel 2, stride 2

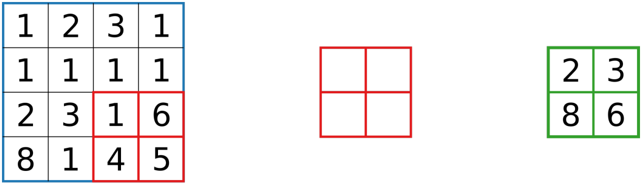
Example of 2D pooling:



maxpool, kernel 2, stride 2

Convolutional Neural Network (3/5)

Example of 2D pooling:



maxpool, kernel 2, stride 2

Convolutional Neural Network (4/5)

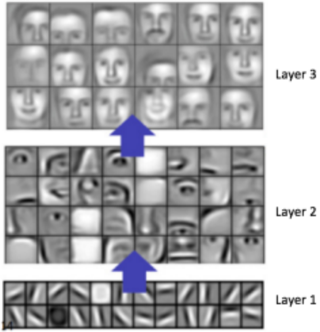
And repeat...

- Convolutional neural network: mainly *Convolution* + *Pooling*.
- ...But many other components may be added! (batch norm, dropout, skip connections, concatenation, ...)

Convolutional Neural Network (5/5)

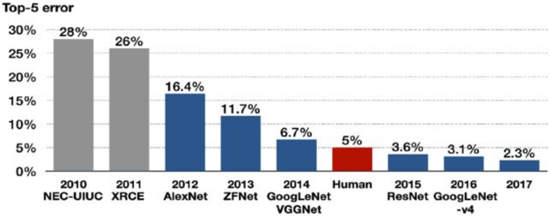
Why convolutions?

- Kernels capture important information in images
- The kernels become more and more complex with the depth of the network



Source: `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`

What happened in 2012?



A combination of...

- Convolutional neural networks
- A very large dataset (ImageNet)
- Clever tricks (ex: data augmentation, *i.e.* altering image during training, very standard in Deep Learning)
- The use of GPUs for computation

Outline

State-of-the-Art nowadays

Image classification

- Image classification for a single dataset is (almost) solved
- Challenges of adapting models to unseen datasets
- Challenges when data is scarce
- Specific domains with few variability or complex classification are still challenging (ex: medical imaging)

Large Language Models

- Large Language Models caught everyone's attention (ChatGPT)
- Challenges of reducing their resources (data/power)
- May hallucinate: lack of robustness

Many other domains

Multimodal models (DALL-E, ...), Audio, Games, Video, ...

State-of-the-Art nowadays

Image classification

- Image classification for a single dataset is (almost) solved
- Challenges of adapting models to unseen datasets
- Challenges when data is scarce
- Specific domains with few variability or complex classification are still challenging (ex: medical imaging)

Large Language Models

- Large Language Models caught everyone's attention (ChatGPT)
- Challenges of reducing their resources (data/power)
- May hallucinate: lack of robustness

Many other domains

Multimodal models (DALL-E, ...), Audio, Games, Video, ...

State-of-the-Art nowadays

Image classification

- Image classification for a single dataset is (almost) solved
- Challenges of adapting models to unseen datasets
- Challenges when data is scarce
- Specific domains with few variability or complex classification are still challenging (ex: medical imaging)

Large Language Models

- Large Language Models caught everyone's attention (ChatGPT)
- Challenges of reducing their resources (data/power)
- May hallucinate: lack of robustness

Many other domains

Multimodal models (DALL-E, ...), Audio, Games, Video, ...

Focus on Large Language Models

Many models

- GPT (Open-AI)
- LLaMA (Meta)
- Gemini (Google)
- Mistral 8x7B (MistralAI)
- Many others... And more to come!

Masked Language Modeling

*How are **you** doing today?* $\rightarrow$ *How are ... doing today?*

- The network learns to reconstruct masked words
- No supervision!
- Allows to leverage immense datasets (ex: GPT-3 was learned on an **Internet scale** dataset)

Focus on Large Language Models

Many models

- GPT (Open-AI)
- LLaMA (Meta)
- Gemini (Google)
- Mistral 8x7B (MistralAI)
- Many others... And more to come!

Masked Language Modeling

*How are **you** doing today?* $\rightarrow$ *How are ... doing today?*

- The network learns to reconstruct masked words
- No supervision!
- Allows to leverage immense datasets (ex: GPT-3 was learned on an **Internet scale** dataset)

Focus on Large Language Models

Many models

- GPT (Open-AI)
- LLaMA (Meta)
- Gemini (Google)
- Mistral 8x7B (MistralAI)
- Many others... And more to come!

Masked Language Modeling

*How are **you** doing today?* $\rightarrow$ *How are ... doing today?*

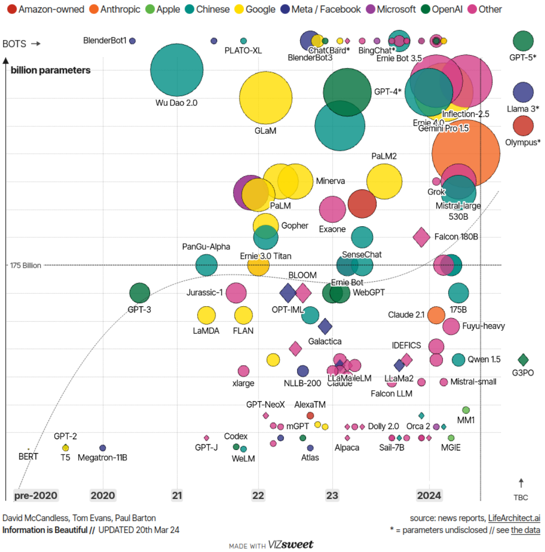
- The network learns to reconstruct masked words
- No supervision!
- Allows to leverage immense datasets (ex: GPT-3 was learned on an **Internet scale** dataset)

Large Language Models are greedy

Model sizes

- AlexNet (2012): 62 Million parameters
- GPT-3 (2020): 175 Billion parameters

Image source:
`https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-langua`



Transformers

Standard architecture nowadays

- No convolution
- Based on *attention*: what should be important for context?
- Used for text, image, audio, ...

Standard architecture nowadays

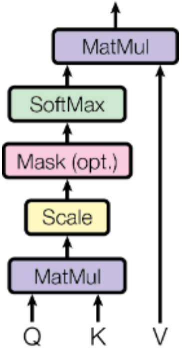
- No convolution
- Based on *attention*: what should be important for context?
- Used for text, image, audio, ...

Transformer block

Based on 3 elements:

- Key
- Query
- Value

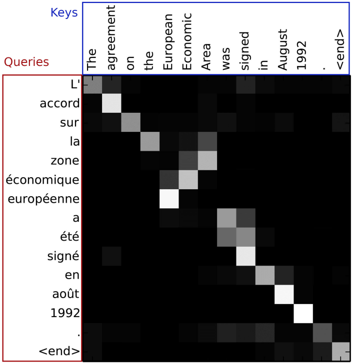
Image source: Vaswani, A. *et al.* (2017). Attention is all you need. Advances in neural information processing systems, 30.



Intuition behind Transformers (1/3)

Attention: Key and Query

- Key: The current word of interest
- Query: All words which may be related

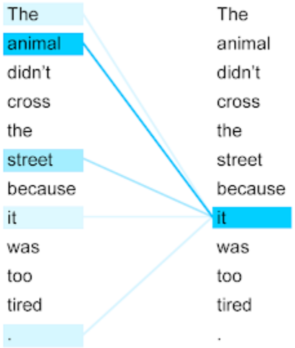


Source: Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Intuition behind Transformers (2/3)

From Attention to Self-Attention

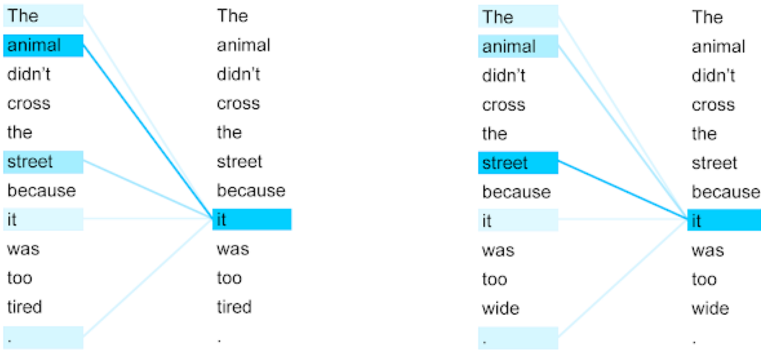
In self-attention, Keys and Queries come from the same text: **context**.



Source:
`https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/`

From Attention to Self-Attention

In self-attention, Keys and Queries come from the same text: **context**.



Source:
`https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/`

Intuition behind Transformers (3/3)

Transformer block

- Key and Query: Context
- Value: Modify the current work, to integrate context

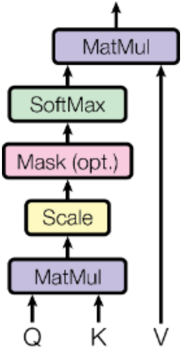
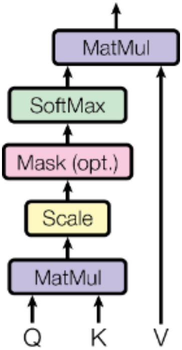


Image source: Vaswani, A. *et al.* (2017). Attention is all you need. Advances in neural information processing systems, 30.

Transformer block

- Key and Query: Context
- Value: Modify the current work, to integrate context



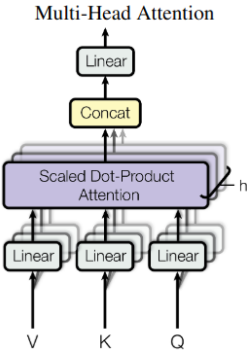


Image source: Vaswani, A. *et al.* (2017). Attention is all you need. Advances in neural information processing systems, 30.

Repeat Transformer blocks: **Deep** model

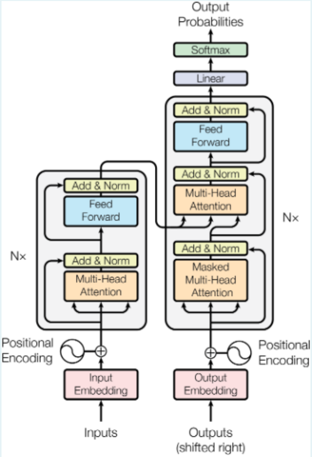


Figure 1: The Transformer - model architecture.

Image source: Vaswani, A. *et al.* (2017). Attention is all you need. Advances in neural information processing systems, 30.

Deep Learning

Conclusion

- Deep Learning algorithms: **powerful** without feature extraction
- They require **a lot** of data to be trained
- The architecture plays an important role

Common criticisms

- Hard to interpret
- Reproduce biases from data
- May require massive amounts of energetic consumption

Going further

- Details and maths behind IA: https://youtu.be/aircAruvnKk
- Ethics and reflexions (french): Science4all & M.Phi

https://youtu.be/HbFadtOxs4k | https://youtu.be/_XJsAQsTOBo

Conclusion

- Deep Learning algorithms: **powerful** without feature extraction
- They require **a lot** of data to be trained
- The architecture plays an important role

Common criticisms

- Hard to interpret
- Reproduce biases from data
- May require massive amounts of energetic consumption

Going further

- Details and maths behind IA: https://youtu.be/aircAruvnKk
- Ethics and reflexions (french): Science4all & M.Phi

https://youtu.be/HbFadtOxs4k | https://youtu.be/_XJsAQsTOBo

Conclusion

- Deep Learning algorithms: **powerful** without feature extraction
- They require **a lot** of data to be trained
- The architecture plays an important role

Common criticisms

- Hard to interpret
- Reproduce biases from data
- May require massive amounts of energetic consumption

Going further

- Details and maths behind IA: https://youtu.be/aircAruvnKk
- Ethics and reflexions (french): Science4all & M.Phi

https://youtu.be/HbFadtOxs4k | https://youtu.be/_XJsAQsTOBo

Practical session

Lab

- Lab Pytorch: manipulating the basics of PyTorch
- Lab Modality: try a first baseline on your modality