

Course 4: Deep Learning



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

2024-10-25

Course 4: Deep Learning

Course 4: Deep Learning



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

Summary

Last session

- Unsupervised learning - discover structure from unlabeled data
- Clustering
- Decomposition - sparse dictionary learning
- Practical ethics

Today's session

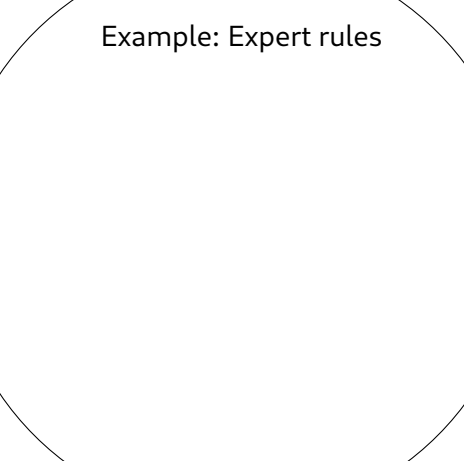
- Multi-Layer Perceptron
- Convolutional Neural Networks
- Transformers

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



AI

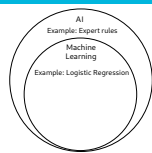
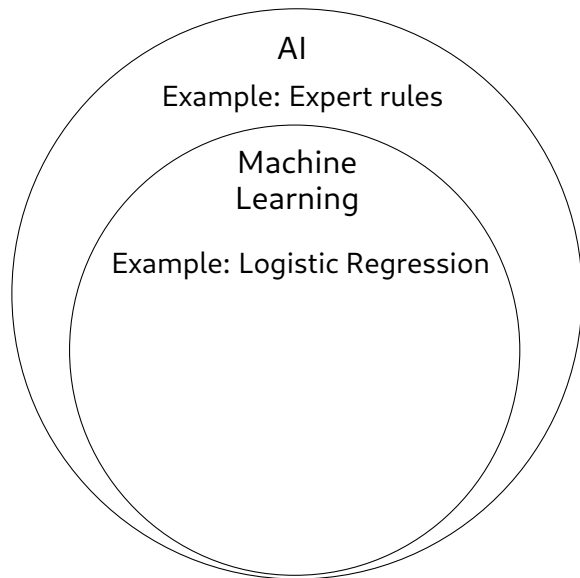
Example: Expert rules

2024-10-25

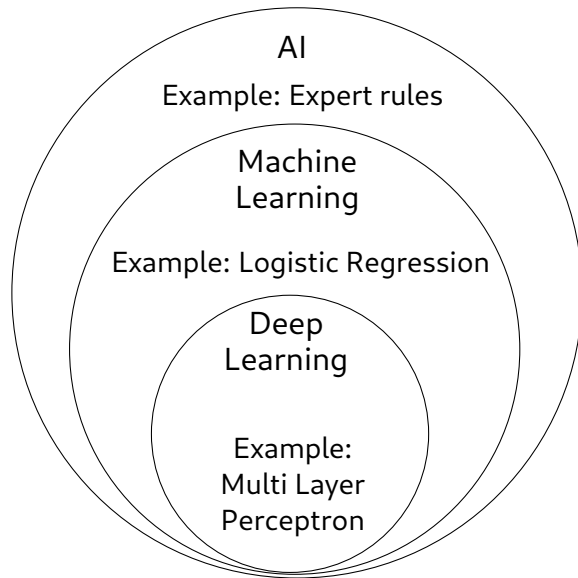
L

AI
Example: Expert rules

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

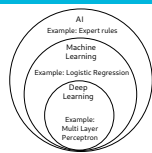


Deep Learning is a particular case of Machine Learning.



└ Global overview...

Deep Learning is a particular case of Machine Learning.



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.

└ Deep Learning in a nutshell (1/3)

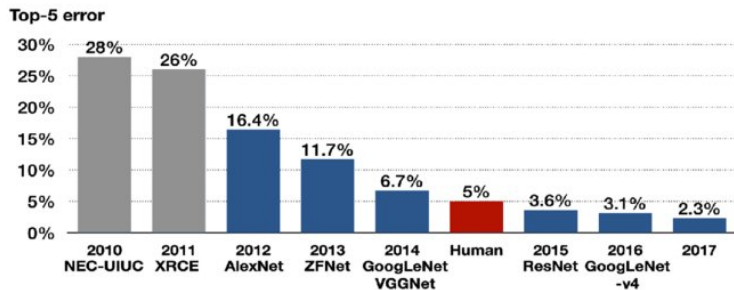
Definition

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: Russakovsky, Dieg et al.. ImageNet Large Scale Visual Recognition Challenge, <https://arxiv.org/pdf/1409.0575.pdf>

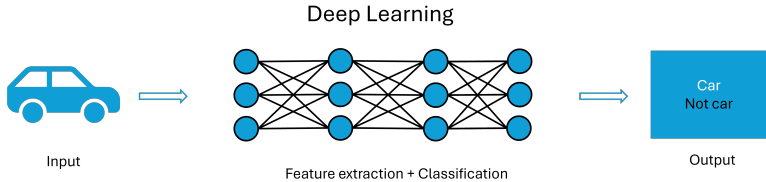
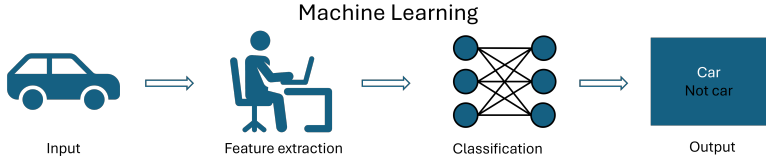
Deep Learning in a nutshell (2/3)

A major breakthrough in image classification:



The landscape of Machine Learning changed in 2012: Deep Neural Networks, a technique used in a minority of cases until then, suddenly won the Image Classification contest on ImageNet, a standard image classification dataset. From this point, Deep Neural Networks became mainstream, and the performance of Deep Neural Network models skyrocketed.

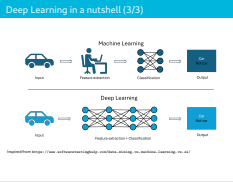
Deep Learning in a nutshell (3/3)



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

2024-10-25 Course 4: Deep Learning

└ Deep Learning in a nutshell (3/3)



1 Multi-layer Perceptron

2 Convolutional Neural Networks (CNN)

3 Transformers

2024-10-25

Course 4: Deep Learning
└ Multi-layer Perceptron
└ Outline

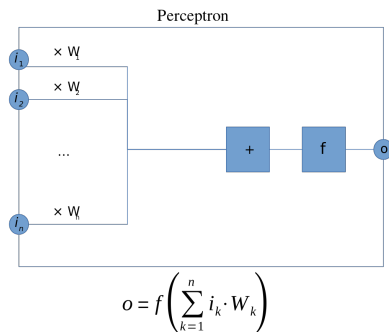
Outline

- Multi-layer Perceptron
- Convolutional Neural Networks (CNN)
- Transformers

Perceptron

1943, implementation in 1957

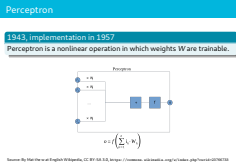
Perceptron is a nonlinear operation in which weights W are trainable.



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

2024-10-25

Course 4: Deep Learning
└ Multi-layer Perceptron
└ Perceptron



The Perceptron is a matrix multiplication, followed by a nonlinear function $f()$. It is important to say that the Perceptron is old!! Hence, AI is not a brand new thing, but an old research domain.

Perceptron

1943, implementation in 1957

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

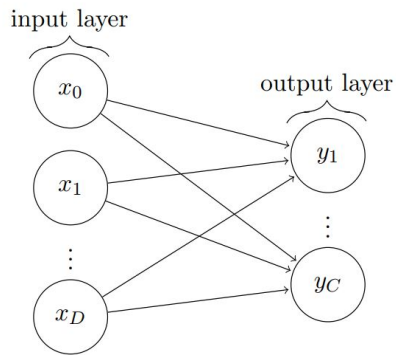


Figure: The arrows represent the weights W .

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

2024-10-25

Course 4: Deep Learning

└ Multi-layer Perceptron

└ Perceptron

Perceptron

1943, implementation in 1957

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

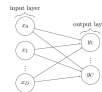


Figure: The arrows represent the weights W .

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

The Perceptron is a matrix multiplication, followed by a nonlinear function $f()$. It is important to say that the Perceptron is old!! Hence, AI is not a brand new thing, but an old research domain.

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: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron

- └ Optimizing the weights

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

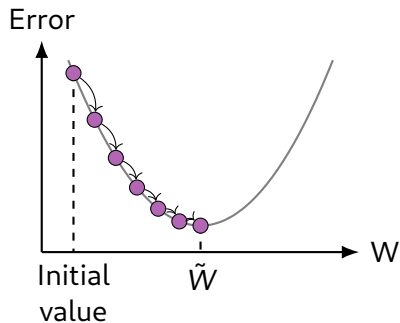
- Compute the gradient: $\frac{\partial J(W)}{\partial W}$ (high dimensional derivative)
- Update weights: $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$

The Perceptron is a matrix multiplication, followed by a nonlinear function $f()$. It is important to say that the Perceptron is old!! Hence, AI is not a brand new thing, but an old research domain.

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>

2024-10-25

Course 4: Deep Learning
└ Multi-layer Perceptron

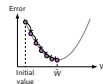
└ Gradient descent

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>

The gradient follows the increase of the error function, hence the inverse of the gradient follows its decrease. Parameter η , the learning rate, gives the size of the step to take at each iteration. If too small, the model will slowly converge. If too large, the model can be unstable and never reach the optimal solution. In practice, setting the learning rate is not easy.

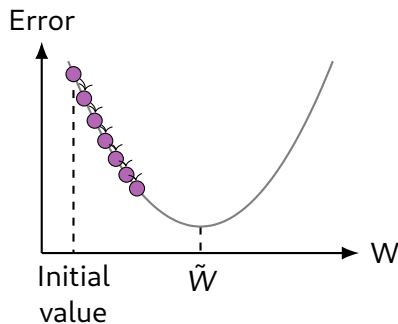
Gradient descent

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:



2024-10-25

Course 4: Deep Learning

└ Multi-layer Perceptron

└ Gradient descent

Gradient descent

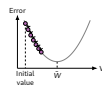
Intuition behind the gradient descent

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:



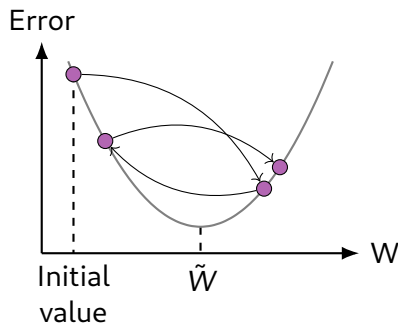
The gradient follows the increase of the error function, hence the inverse of the gradient follows its decrease. Parameter η , the learning rate, gives the size of the step to take at each iteration. If too small, the model will slowly converge. If too large, the model can be unstable and never reach the optimal solution. In practice, setting the learning rate is not easy.

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:



2024-10-25

Course 4: Deep Learning

└ Multi-layer Perceptron

└ Gradient descent

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

Large step:

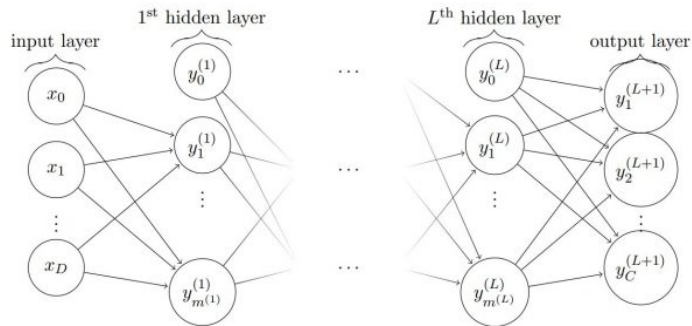


The gradient follows the increase of the error function, hence the inverse of the gradient follows its decrease. Parameter η , the learning rate, gives the size of the step to take at each iteration. If too small, the model will slowly converge. If too large, the model can be unstable and never reach the optimal solution. In practice, setting the learning rate is not easy.

Multi-Layer Perceptron

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

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



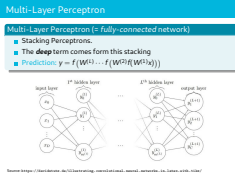
Source: <https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/>

2024-10-25

Course 4: Deep Learning

└ Multi-layer Perceptron

└ Multi-Layer Perceptron



The bias are removed from the equations for simplicity, but say that they exist orally.

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.

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.

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron
 - └ 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.

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron
 - └ Neural Networks

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.

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron
 - └ 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.

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron
 - └ 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.

2024-10-25

Course 4: Deep Learning

- └ Multi-layer Perceptron
 - └ Neural Networks

Definitions

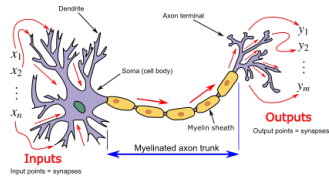
$$\mathbf{y}^{(l+1)} = f(\mathbf{W}^{(l)}\mathbf{y}^{(l)} + \mathbf{b}^{(l)}) = \phi^{(l)}(\mathbf{y}^{(l)})$$

- 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**.

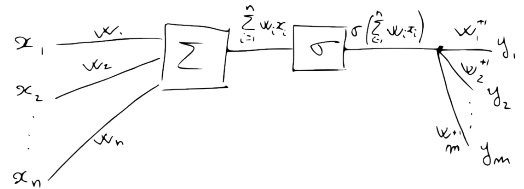
Note: while each layer can have a different activation function f , it is standard that each layer uses the same.

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



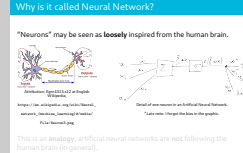
Detail of one neuron in an Artificial Neural Network.
* Late note: I forgot the bias in the graphic.

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

2024-10-25

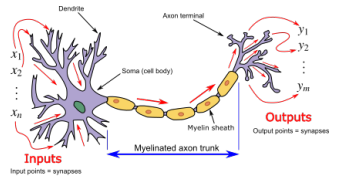
Course 4: Deep Learning └ Multi-layer Perceptron

└ Why is it called Neural Network?



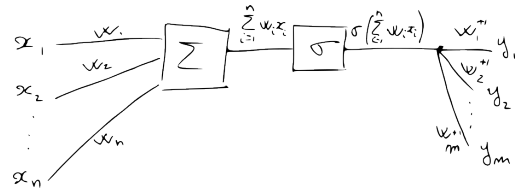
Why is it called Neural Network?

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

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



Detail of one neuron in an Artificial Neural Network.
* Late note: I forgot the bias in the graphic.

2024-10-25

Course 4: Deep Learning └ Multi-layer Perceptron

└ Why is it called Neural Network?

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

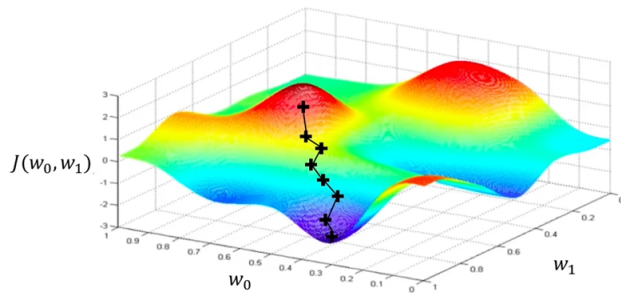
Detail of one neuron in an Artificial Neural Network.
*Late note: I forgot the bias in the graphic.

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: $\frac{\partial J(W)}{\partial W} = \frac{\partial J(W)}{\partial W^{(L)}} \frac{\partial W^{(L)}}{\partial W^{(L-1)}} \frac{\partial W^{(L-1)}}{\partial W^{(L-2)}} \dots \frac{\partial W^{(2)}}{\partial W^{(1)}}$
- The error **backpropagates** through the network (reverse path)
- Computationally efficient, but finds a local minimum (at best)



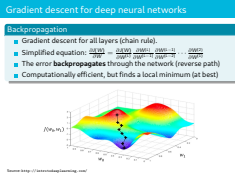
Source: <http://introtodeeplearning.com/>

2024-10-25

Course 4: Deep Learning

└ Multi-layer Perceptron

└ Gradient descent for deep neural networks



Careful: the given equation is not correct, just conveniently simplified. For better details, refer to <https://en.wikipedia.org/wiki/Backpropagation>.

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 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

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

1 Multi-layer Perceptron

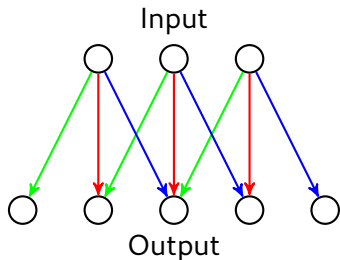
2 Convolutional Neural Networks (CNN)

3 Transformers

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



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

2024-10-25

Course 4: Deep Learning

Convolutional Neural Networks (CNN)

Convolutional Neural Network (1/5)

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

Convolution neural networks are the most common architecture for neural networks nowadays. They are particularly successful for image processing, were not challenged until very recently, with Vision Transformers (see Transformers in the last slides). The translation invariance is really important for image processing, as objects can be anywhere on the image. It contributed to the success of Convolutional Neural Networks.

Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

$I \qquad K \qquad I * K$

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

2024-10-25

Course 4: Deep Learning

└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

$I \qquad K \qquad I * K$

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

Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

$I \qquad K \qquad I * K$

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

2024-10-25

Course 4: Deep Learning

└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (2/5)

Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

$I \qquad K \qquad I + K$

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

Convolutional Neural Network (2/5)

Example of 2D convolution:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 \\ 1 & 2 & 4 & 3 & 3 \\ 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 3 & 1 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix}$$

I K $I * K$

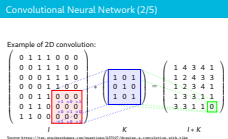
Source: <https://tex.stackexchange.com/questions/437007/drawing-a-convolution-with-tikz>

2024-10-25

Course 4: Deep Learning

└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (2/5)



Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	

maxpool, kernel 2, stride 2

2024-10-25

Convolutional Neural Network (3/5)

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	

maxpool, kernel 2, stride 2

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	3

maxpool, kernel 2, stride 2

2024-10-25

Convolutional Neural Network (3/5)

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	3

maxpool, kernel 2, stride 2

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	3
8	

maxpool, kernel 2, stride 2

2024-10-25

Convolutional Neural Network (3/5)

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5

2	3
8	

maxpool, kernel 2, stride 2

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5



2	3
8	6

maxpool, kernel 2, stride 2

2024-10-25

Course 4: Deep Learning

└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (3/5)

Example of 2D pooling:

1	2	3	1
1	1	1	1
2	3	1	6
8	1	4	5



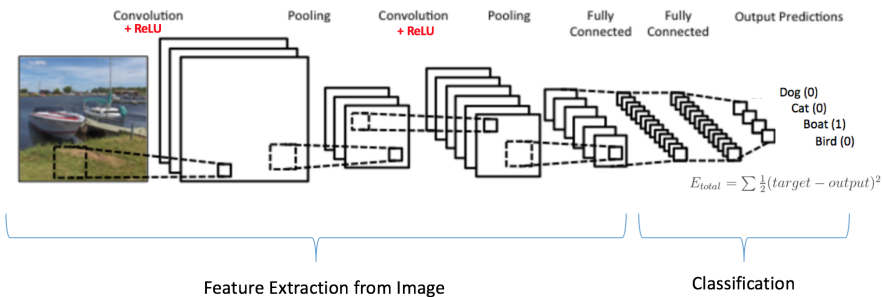
2	3
8	6

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, ...)



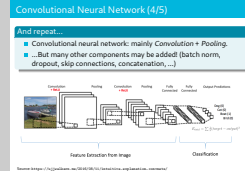
Source: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

2024-10-25

Course 4: Deep Learning

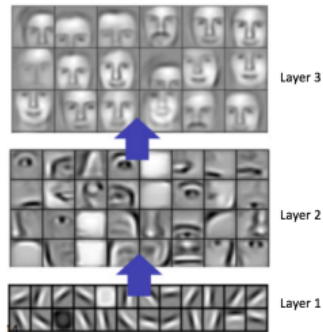
└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (4/5)



Why convolutions?

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



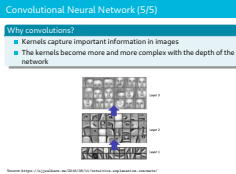
Source: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

2024-10-25

Course 4: Deep Learning

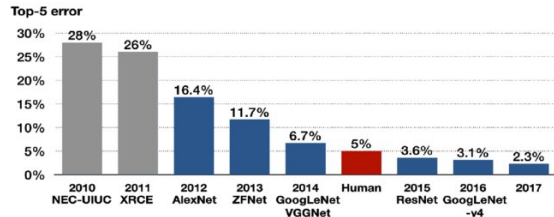
└ Convolutional Neural Networks (CNN)

└ Convolutional Neural Network (5/5)



Convolution are able to catch simple shapes (lines, edges) which turn into complex shapes in the subsequent layers.

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

2024-10-25

Course 4: Deep Learning

└ Convolutional Neural Networks (CNN)

└ What happened in 2012?

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

3 Transformers

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, ...

2024-10-25

Course 4: Deep Learning

└ Transformers

└ State-of-the-Art nowadays

State-of-the-Art nowadays	
Image classification	<ul style="list-style-type: none">■ 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	<ul style="list-style-type: none">■ 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, ...

2024-10-25

Course 4: Deep Learning

└ Transformers

└ State-of-the-Art nowadays

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, ...

2024-10-25

Course 4: Deep Learning

└ Transformers

└ State-of-the-Art nowadays

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, ...

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? → 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)

2024-10-25

Course 4: Deep Learning

└ Transformers

└ 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? → 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? → 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)

2024-10-25

Course 4: Deep Learning

└ Transformers

└ Focus on Large Language Models

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? → 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)

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? → 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)

2024-10-25

Course 4: Deep Learning

└ Transformers

└ Focus on Large Language Models

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

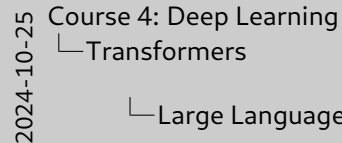
How are **you** doing today? → How are ... doing today?

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

Model sizes

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

* = parameters undisclosed // see the data



- Large Language Models are greedy

Standard architecture nowadays

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

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

Standard architecture nowadays

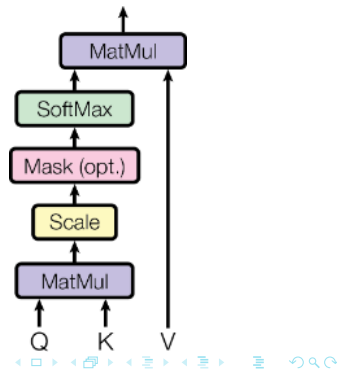
- No convolution
- Based on *attention*: what should be important for context?
- Used for text, image, audio, ...

Transformer block

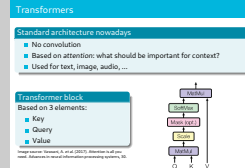
Based on 3 elements:

- Key
- Query
- Value

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



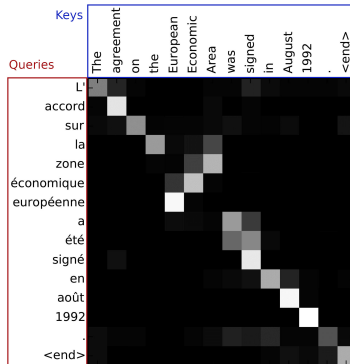
2024-10-25 Course 4: Deep Learning
└ Transformers
└ Transformers



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.

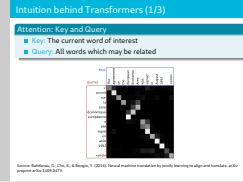
2024-10-25

Course 4: Deep Learning

Transformers

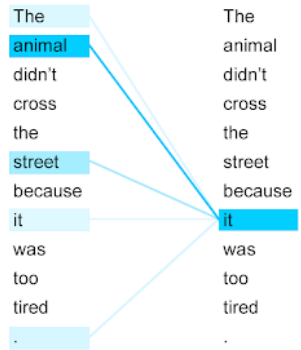
Intuition behind Transformers (1/3)

Attention consists in asking: which word is important to explain my current word?



From Attention to Self-Attention

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

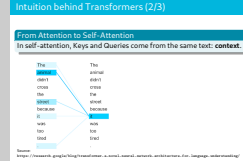


Source:
<https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/>

2024-10-25

Transformers

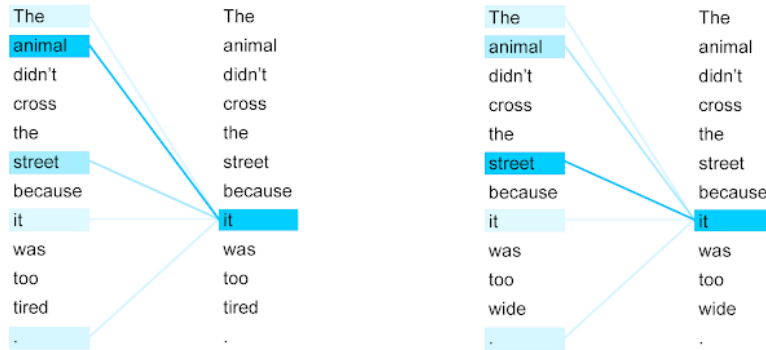
Intuition behind Transformers (2/3)



Notice how the model is able to understand the context of the "it" in the examples: "it" refers to the animal in the first example, and to the street in the second. The model has caught this behavior.

From Attention to Self-Attention

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

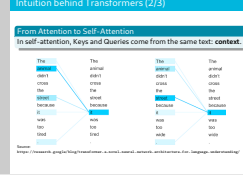


Source:
<https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/>

2024-10-25

Transformers

Intuition behind Transformers (2/3)



Notice how the model is able to understand the context of the “it” in the examples: “it” refers to the animal in the first example, and to the street in the second. The model has caught this behavior.

Intuition behind Transformers (3/3)

Transformer block

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

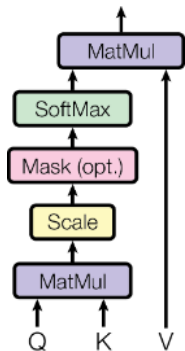
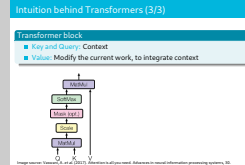


Image source: Vaswani, A. et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.

2024-10-25 Course 4: Deep Learning
└ Transformers
└ Intuition behind Transformers (3/3)



Finally, a Transformer block will modify each word for it to integrate the context learned with Keys and Queries. In the end, the whole sentence is encoded through the embeddings of each word.

Transformer block

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

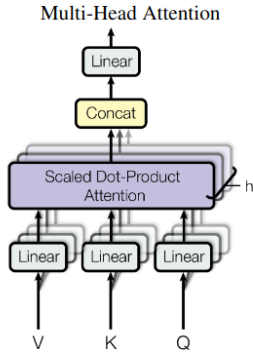
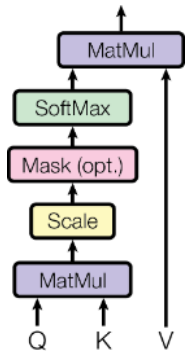
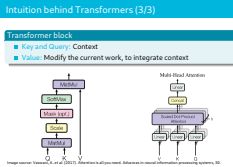


Image source: Vaswani, A. et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.



Finally, a Transformer block will modify each word for it to integrate the context learned with Keys and Queries. In the end, the whole sentence is encoded through the embeddings of each word.

Repeat Transformer blocks: Deep model

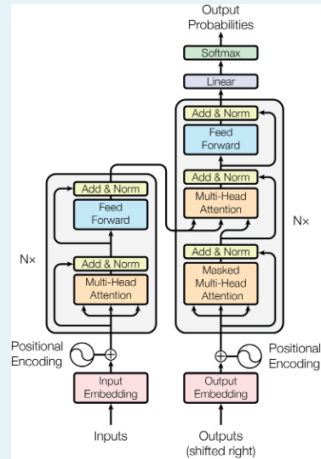
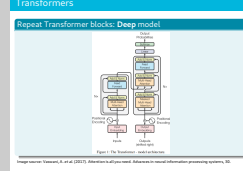


Figure 1: The Transformer - model architecture.

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

2024-10-25 Course 4: Deep Learning
└ Transformers
└ Transformers



doesn't need to go in details, just present the fact that blocks are repeated.

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/HbFadt0xs4k> | https://youtu.be/_XJsAQsTOBo

2024-10-25 Course 4: Deep Learning
└ Transformers
└ Deep Learning

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/HbFadt0xs4k> | 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/HbFadt0xs4k> | https://youtu.be/_XJsAQsTOBo

2024-10-25 Course 4: Deep Learning
└ Transformers
└ 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/HbFadt0xs4k> | 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/HbFadt0xs4k> | https://youtu.be/_XJsAQsT0Bo

2024-10-25 Course 4: Deep Learning
└ Transformers
└ Deep Learning

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

Lab

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