

# Les fondamentaux de l'apprentissage profond avec Python

CEPE / ENSAE-ENSAI Formation continue

Thomas Belhafaoui

2023

# Organisation prévisionnelle

Jour 1	Jour 2	Jour 3
<p>Intelligence artificielle, apprentissage statistique et réseaux de neurones</p> <p>Théorie</p>	<p>Réseaux convolutifs (CNN) et apprentissage par transfert</p> <p>Théorie</p>	<p>Auto-encodeurs</p> <p>Théorie et pratique</p>
	<p>Réseaux convolutifs (CNN) et apprentissage par transfert</p> <p>Pratique (images)</p>	<p>Auto-encodeurs</p> <p>Théorie et pratique</p>
<p>Intelligence artificielle, apprentissage statistique et réseaux de neurones</p> <p>Pratique - séance introductory</p>	<p>Réseaux récurrents (RNN) et traitements textuels (NLP)</p> <p>Théorie</p>	<p>Introduction aux Transformers et aux réseaux antagonistes (GAN)</p> <p>Théorie</p>
	<p>Réseaux récurrents (RNN) et traitements textuels (NLP)</p> <p>Pratique (texte)</p>	<p>Conclusion et ouverture</p> <p>MLOps, Transformers, éthique et risques de l'IA, ...</p>

# Intelligence Artificielle : tentative de définition.

"L'**IA** est la théorie et le développement de **systèmes informatiques** capables de réaliser des tâches **nécessitant habituellement de l'intelligence humaine.**"

Adapté de OxfordReference.com

À quels exemples  
d'IA pensez-vous ?

# Exemples de tâches.

Reconnaissance d'objet

Transcription de parole

Traduction

Prise de décision

Prédiction chiffrée

Génération d'image

Génération de texte

# Différents types de donnée en entrée et en sortie.

Reconnaissance d'objet	image → label
Transcription de parole	son → texte
Traduction	texte → texte
Prise de décision	tableau → oui/non + score
Prédiction chiffrée	tableau → nombre
Génération d'image	texte → image
Génération de texte	texte → texte

# IA, un terme ambigu.

## IA FORTE (GÉNÉRALE)

Un **système unique** qui se comporte comme un être humain.

Doué de **conscience**

(sentience and conscience de soi).

N'existe pas (encore?).

## IA FAIBLE (SPÉCIFIQUE)

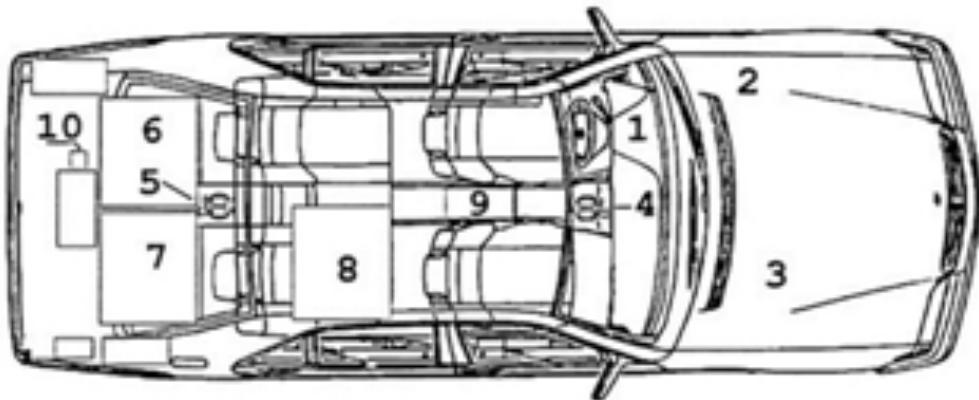
Une **variété de systèmes**, chacun effectuant une tâche **spécifique**.

Dénue de **conscience**.

Largement répandu.

L'IA ne date pas d'hier.

# En 1994, l'une des premières voitures autonomes.

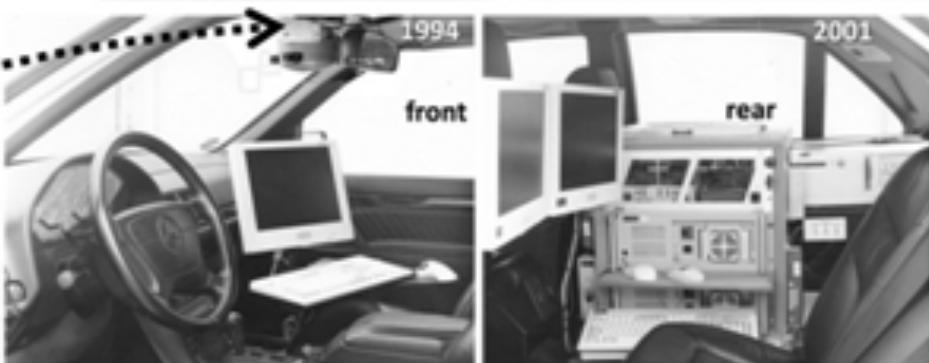


- 1 electrical steering motor  
2 electrical brake control  
3 electronic throttle  
4 front pointing platform for CCD-cameras  
5 rear pointing platform

- 6 Transputer Image Processing system  
7 platform and vehicle controllers  
8 electronics rack, human interface  
9 accelerometers (3orthogonal)  
10 inertial rate sensors



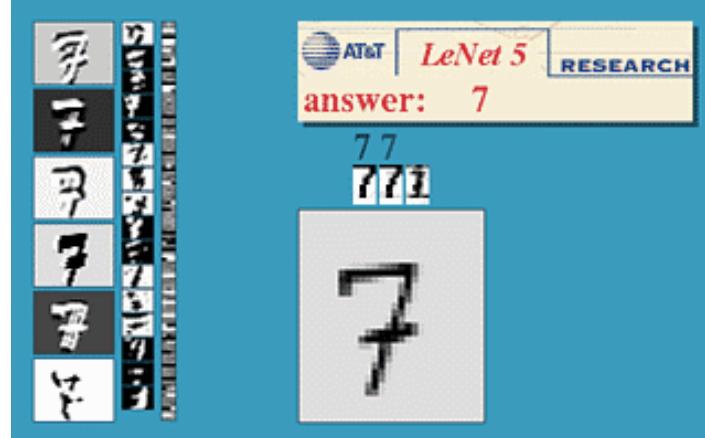
At distance  $L_s \sim 20$  m (~ 60 m),  
the resolution is 5 cm/pixel



# En 1997, Deep Blue gagne contre Garry Kasparov.



En 1998, LeNet-5 popularise les réseaux de neurones profonds.



# Qu'y a-t-il de nouveau ?

D'où vient l'engouement actuel autour de l'IA ?

# Un changement de signification du mot IA.

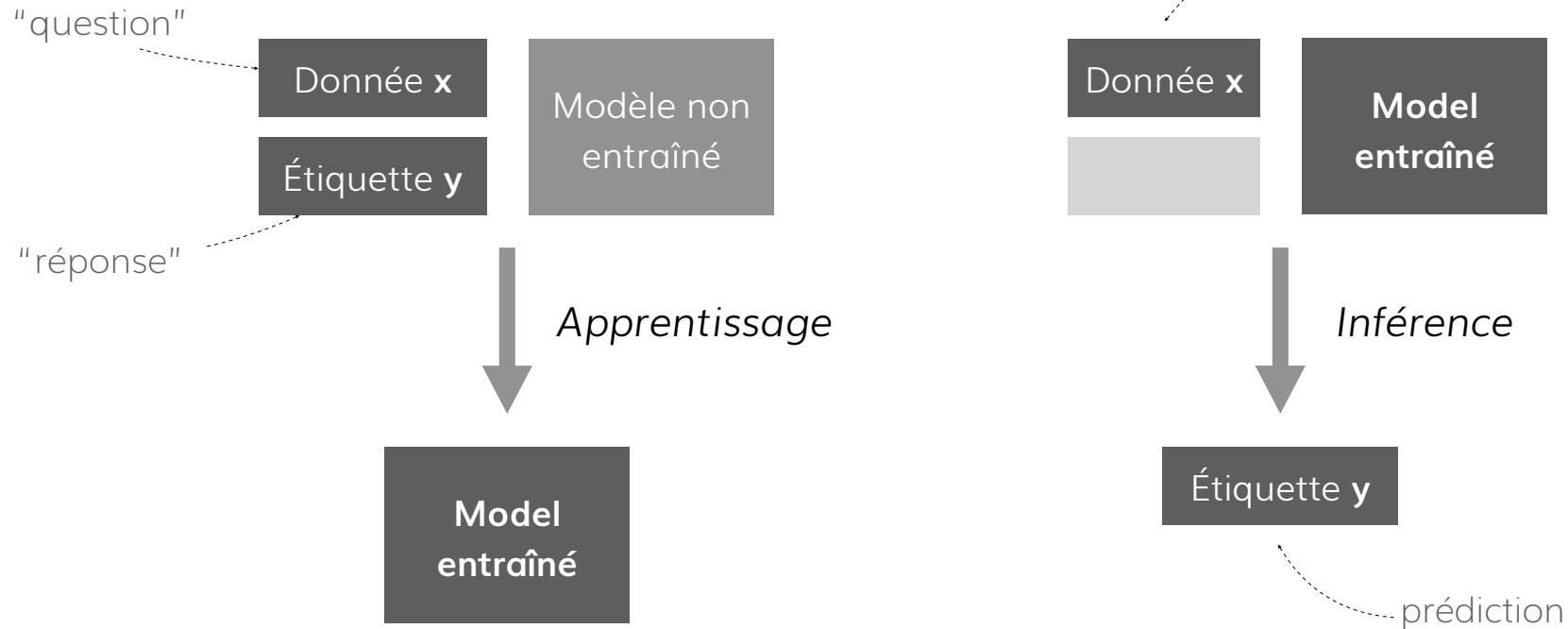


# Des systèmes experts à l'apprentissage automatique.

Ou comment apprendre les "règles" automatiquement  
à partir de données étiquetées.

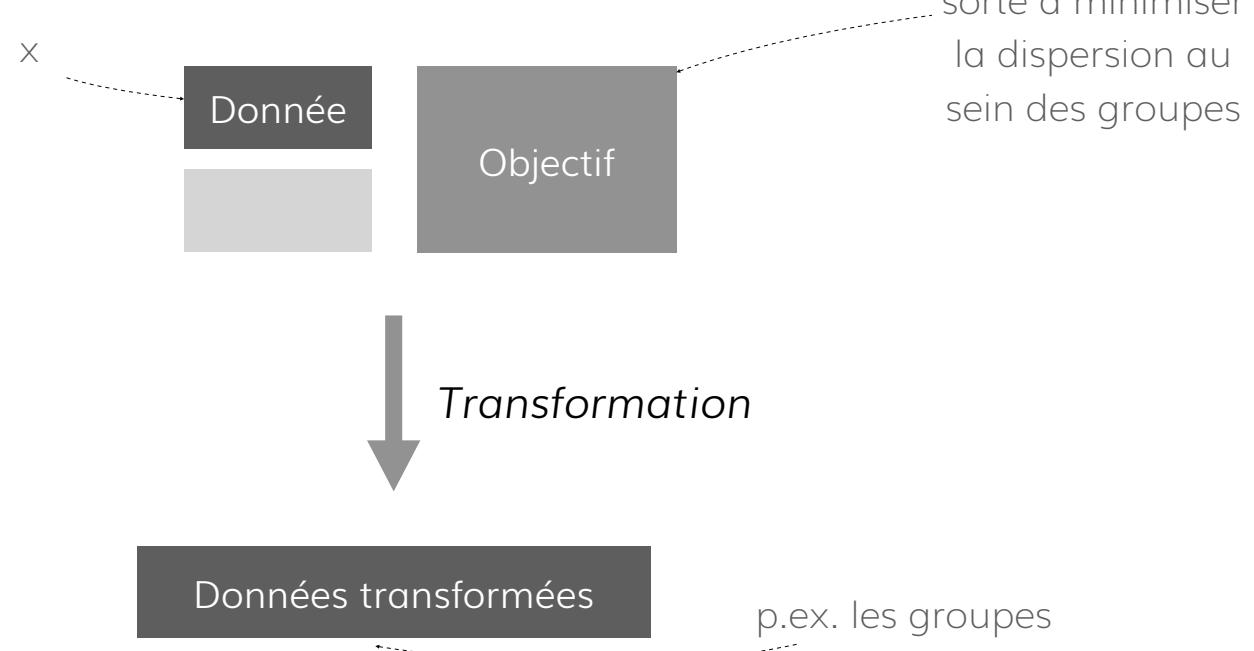
# L'apprentissage supervisé.

Apprendre à partir des données.



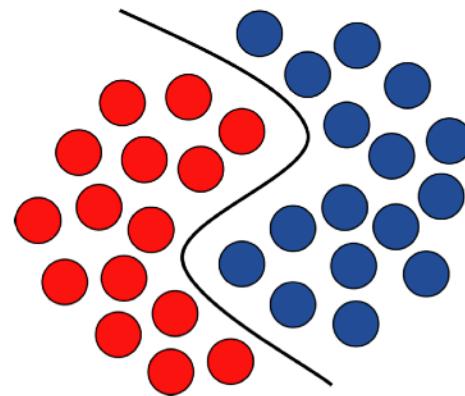
# L'apprentissage non supervisé.

Apprendre sans étiquettes.

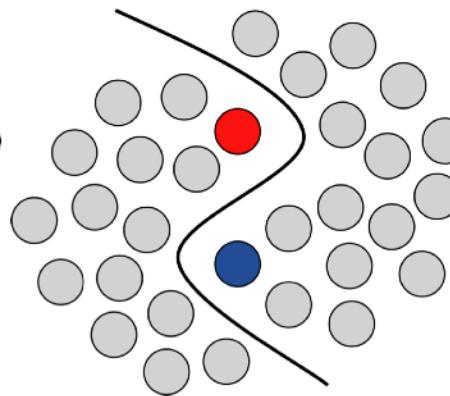


# L'apprentissage non supervisé.

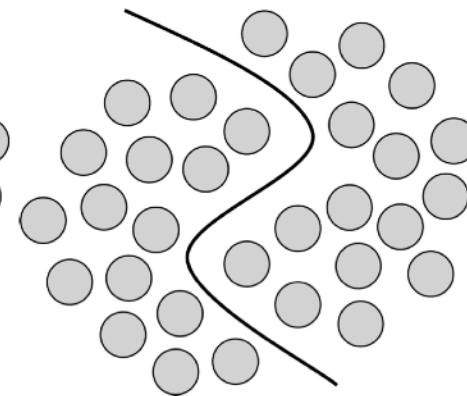
Apprendre sans étiquettes.



Supervisé



Semi-supervisé



Non-supervisé

# Quelques exemples de tâches.

Quelle est la donnée "x" et quelle est l'étiquette "y" ?

# Reconnaissance de chiffres manuscrits.

Jeu de données MNIST et réseau LeNet-5, 1998.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

# Transcription de parole.

Ex : assistants vocaux.



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua."

# Segmentation d'image.

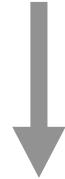
Segment Anything Model (SAM), META AI Research, 2023



# Traduction de texte.

EN

"A woman must have money and a room of her own  
if she is to write fiction."



FR

"Il est indispensable qu'une femme possède quelque argent  
et une chambre à soi si elle veut écrire une œuvre de fiction."

# Aide au diagnostic médical.

Résultats d'analyses sanguines,  
Radiographies,  
Questionnaire de santé,  
...



0.7

Nombres

Images

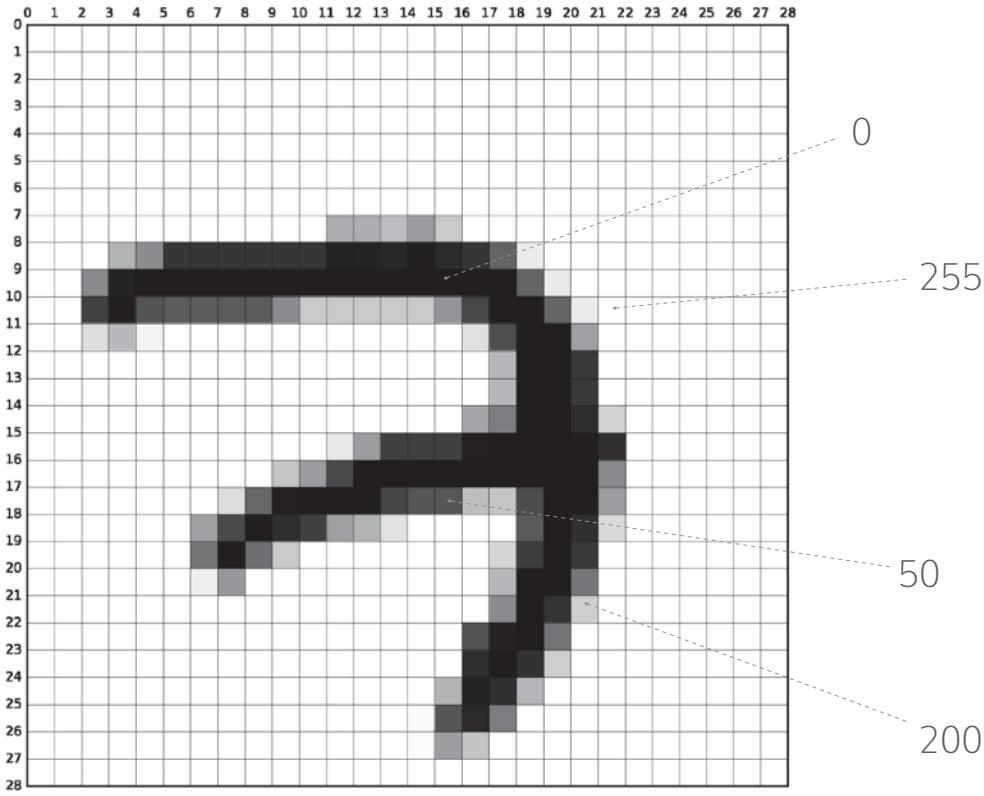
Données numériques  
et non-numériques  
("catégorielles")

Score de maladie entre 0 et 1

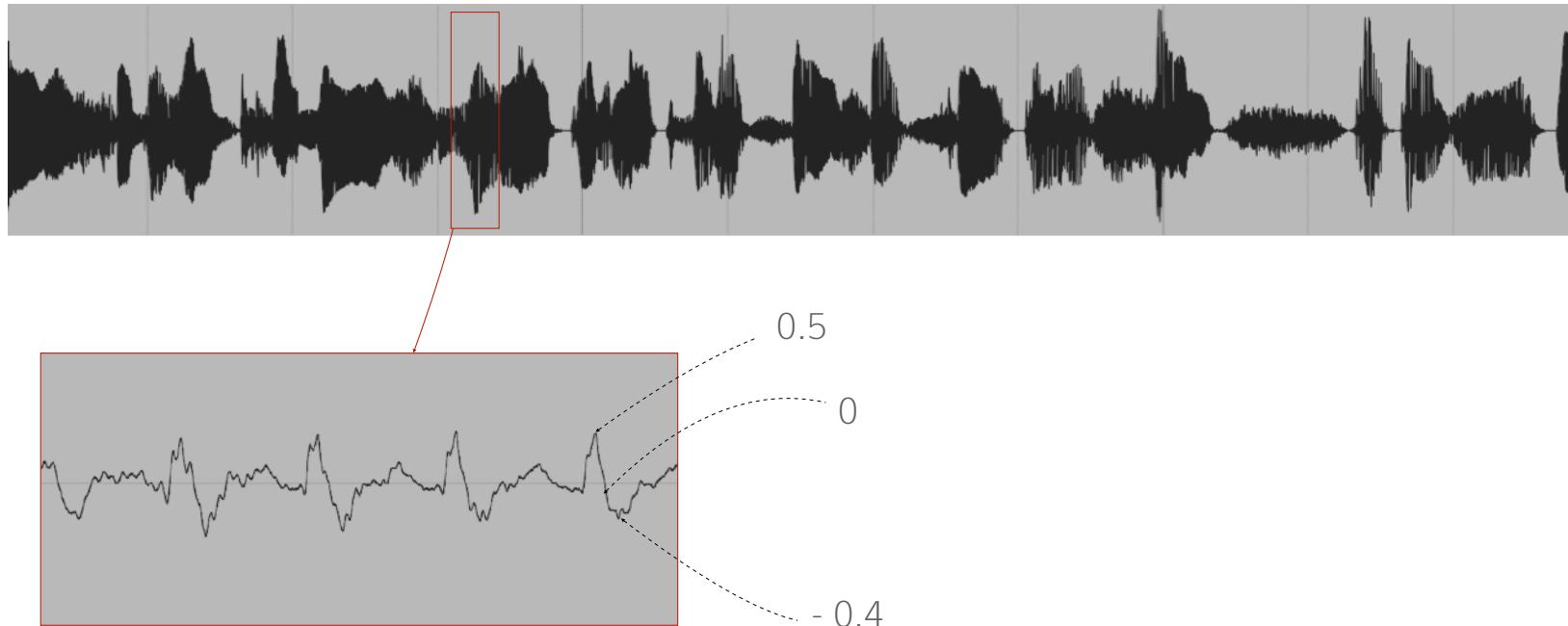
Quelle forme prennent  
les données ?

*Tout n'est que nombres.*

Une image en noir et blanc est un tableau de nombres.



Un son est une liste de nombres.



# Comment représenter des données non-numériques (catégorielles) ?

L'encodage "1 parmi n" ou "one hot".

Données "x"

	Paris	Marseille	Lyon	Toulouse	Nantes
Nantes	0	0	0	0	<b>1</b>
Lyon	0	0	<b>1</b>	0	0
Lyon	0	0	<b>1</b>	0	0
Nantes	0	0	0	0	<b>1</b>
Toulouse	0	0	0	<b>1</b>	0
Nantes	0	0	0	0	<b>1</b>
Lyon	0	0	<b>1</b>	0	0
Nantes	0	0	0	0	<b>1</b>

Toutes les valeurs possibles

Liste ("vecteur") de nombres représentant la donnée "Toulouse"

L'apprentissage automatique  
ne date pas non plus d'hier.

# Les mathématiques derrière l'IA — quelques dates.

- 1795 Méthode des **moindres carrés** (Carl Friedrich Gauss).
- 1847 **Descente de gradient** (Augustin-Louis Cauchy).
- 1957 **Perceptron**, ou premier "neurone artificiel" (Frank Rosenblatt).
- 1986 **Rétropropagation** (Geoffrey Hinton).
- 1995 Algorithme des **Forêts aléatoires** (Tin Kam Ho).
- 1992 Le **Fast Weight Controller**, ancêtre du Transformer  
(Jürgen Schmidhuber).

# Alors où est la nouveauté ?

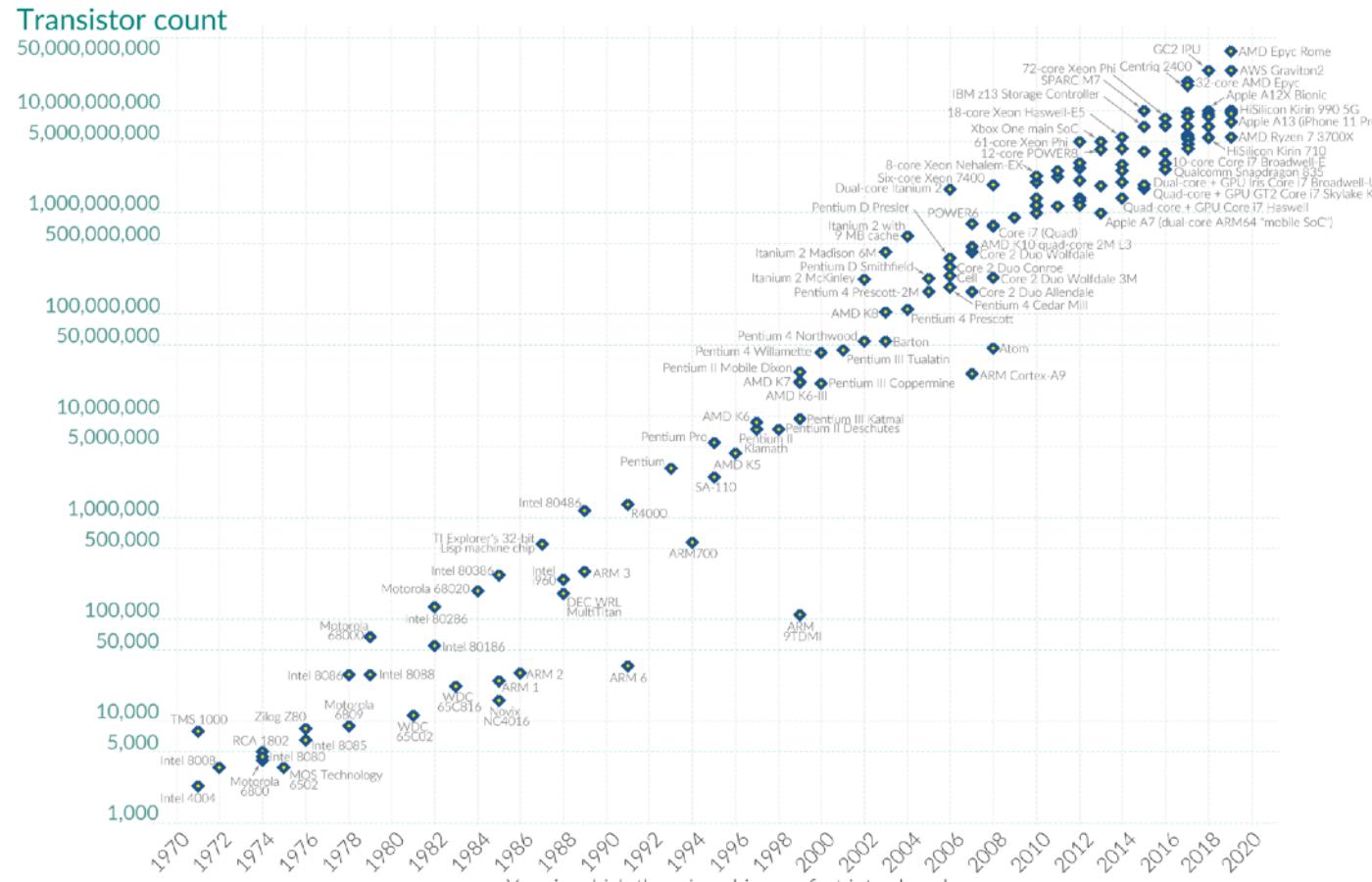
Pourquoi cette explosion de l'IA ?

La nouveauté c'est l'énorme  
**quantité de données** disponibles  
et la **puissance de calcul** bon  
marché.

Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

S Our World  
in Data



Data source: Wikipedia ([wikipedia.org/wiki/Transistor\\_count](https://en.wikipedia.org/w/index.php?title=Transistor_count&oldid=1000000000))

[OurWorldInData.org](http://OurWorldInData.org) – Research and data to make progress against the world's largest problems

Licensed under CC-BY by the authors Hannah Ritchie and Max Rose

# Quelques chiffres à propos de GPT-3

Le modèle

**175 milliards** de paramètres.

**800 GB** pour le stocker.

L'apprentissage

**500 milliards** de mots  
(60% issus de *Common Crawl*).

**4.6 millions** de dollars

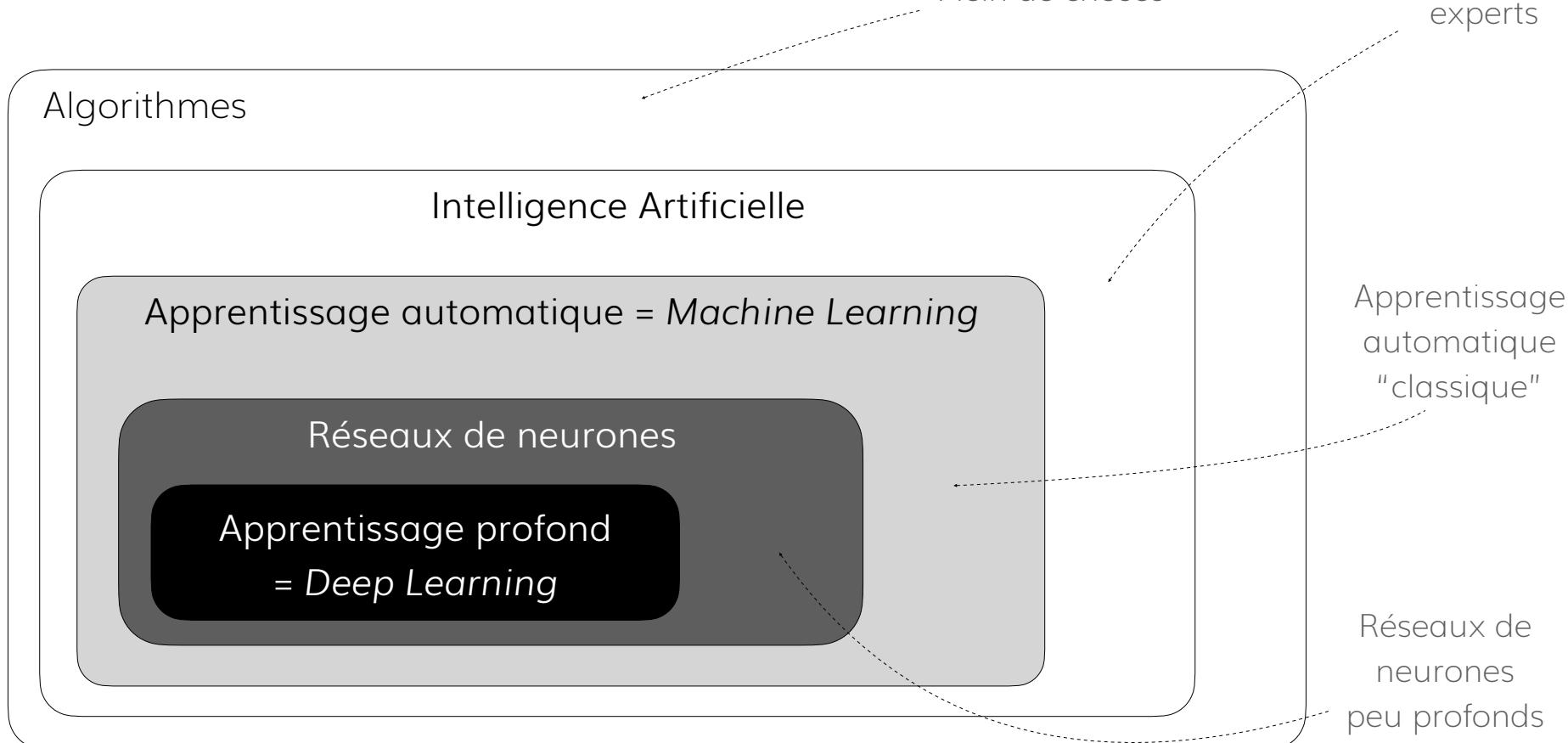
**355 GPU ans.**

**~ 1,2 GWh<sup>1</sup>**  
(~ consommation annuelle de  
180 Français·e·s moyen·ne·s<sup>2</sup>).

<sup>1</sup> <https://arxiv.org/ftp/arxiv/papers/2204/2204.05149.pdf>

<sup>2</sup> Qui est de 6,7 MWh pour un·e Français·e ([https://en.wikipedia.org/wiki/List\\_of\\_countries\\_by\\_electricity\\_consumption](https://en.wikipedia.org/wiki/List_of_countries_by_electricity_consumption))

# Synthèse du vocabulaire.

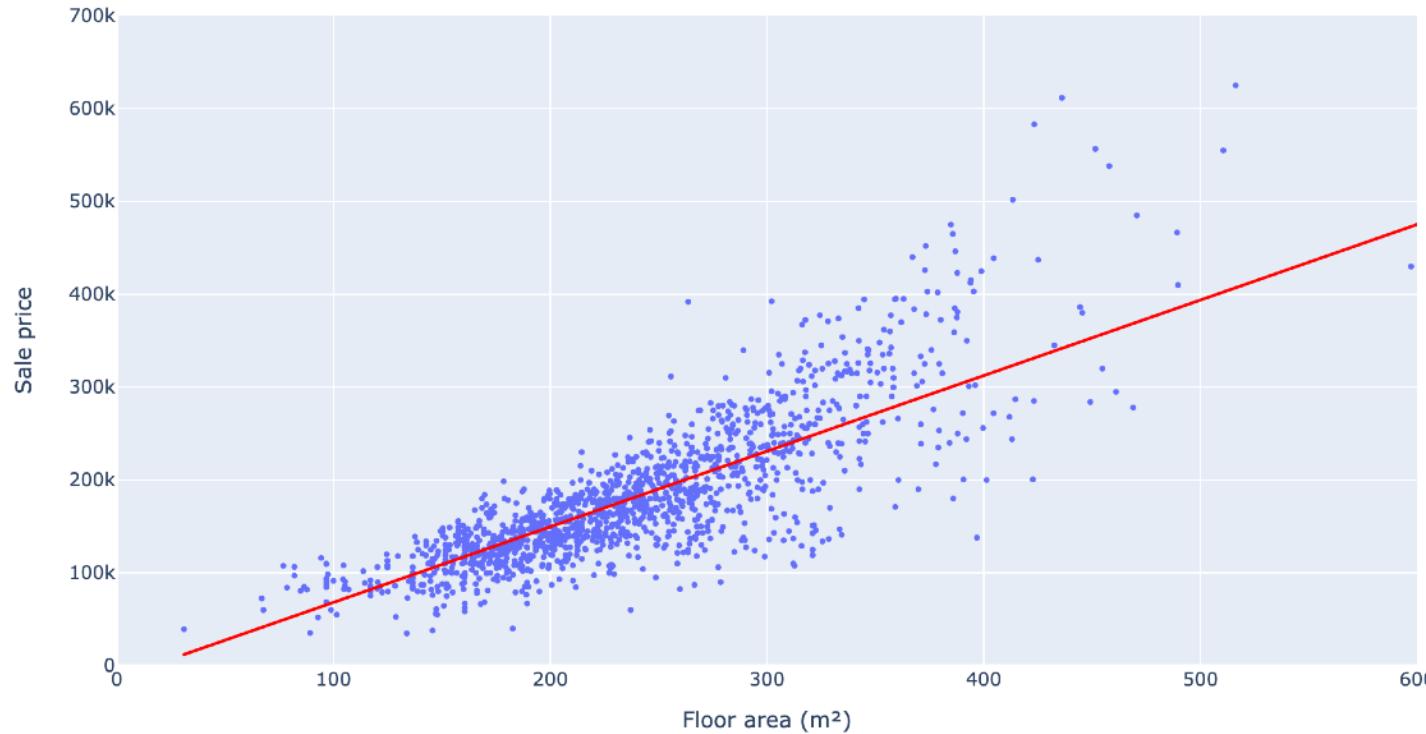


L'algorithme d'apprentissage  
automatique le plus simple :

la régression linéaire.

# La régression linéaire.

Houses sale price as a function of floor area



Prix de vente

Surface habitable

$$y = ax + b$$

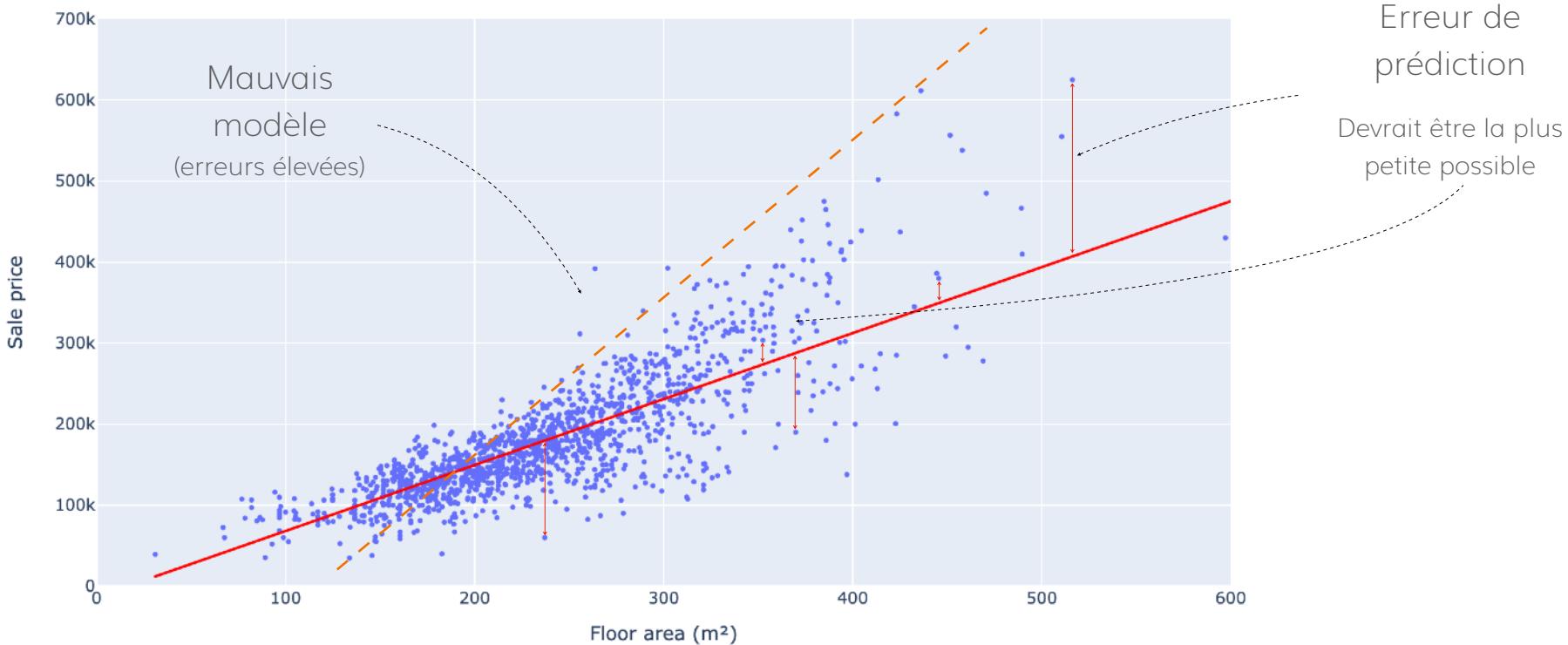
Paramètres  
(poids)

# Comment trouver les "bons" paramètres ?

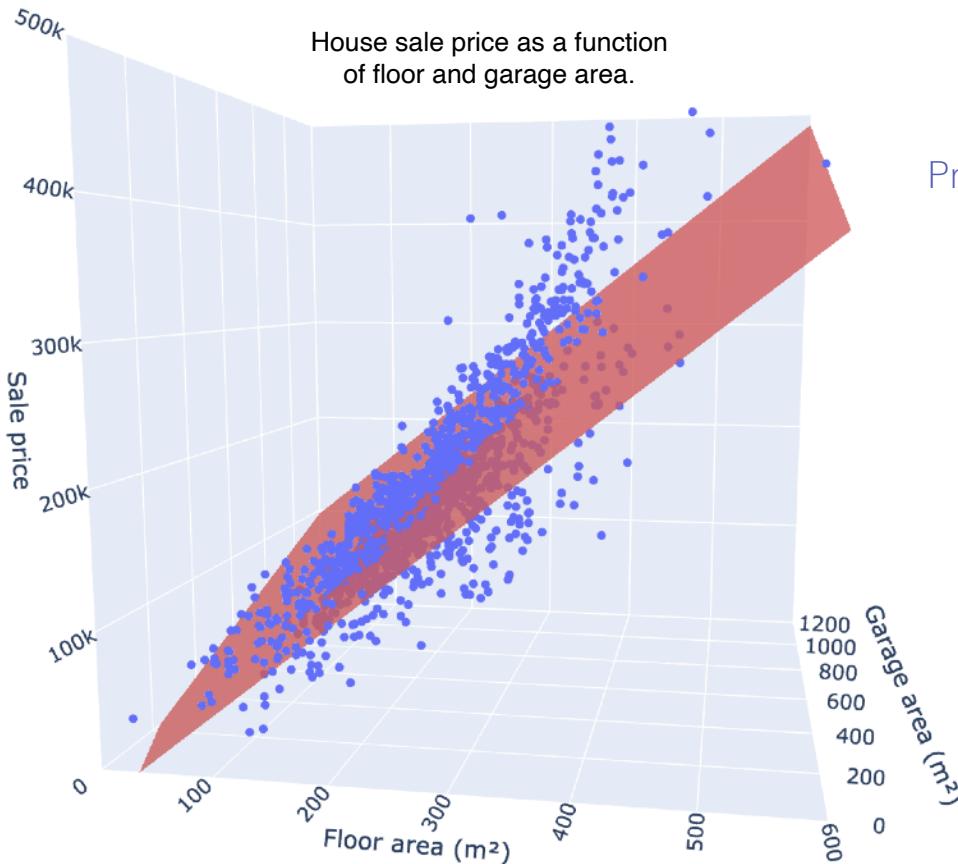
Minimiser la fonction de coût.

# Apprendre (entraîner) le modèle : minimiser la fonction de coût.

Ici, la fonction de coût est la **somme des erreurs au carré**.



# La régression linéaire en deux dimensions.



Prix de vente

Surface du garage

Surface habitable

Paramètres

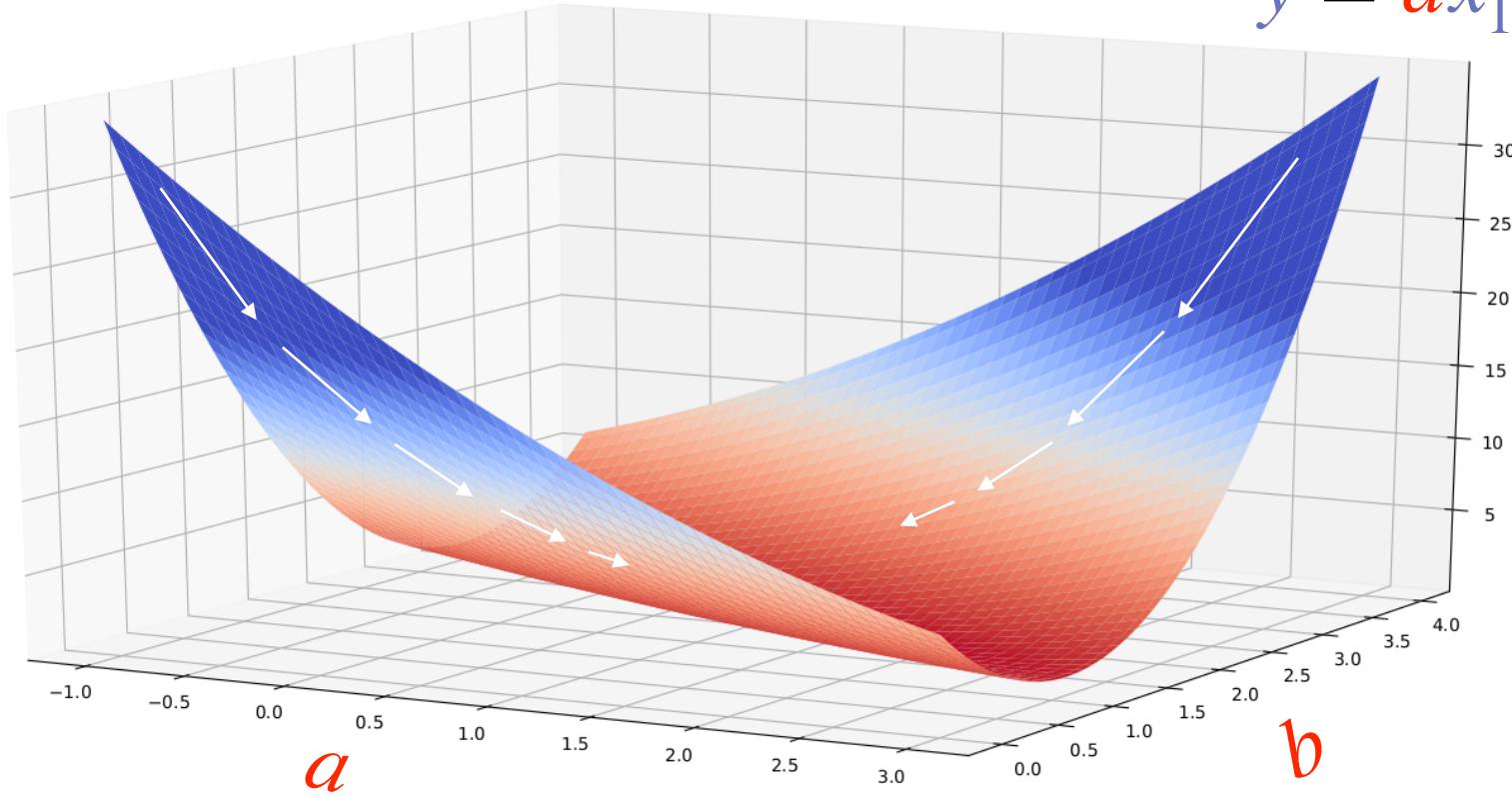
$$y = ax_1 + bx_2 + c$$

Comment trouver  
les paramètres  
qui minimisent  
la fonction de coût ?

La descente de gradient.

La descente de gradient : descendre le long de la plus forte pente.

$$y = ax_1 + bx_2 + c$$

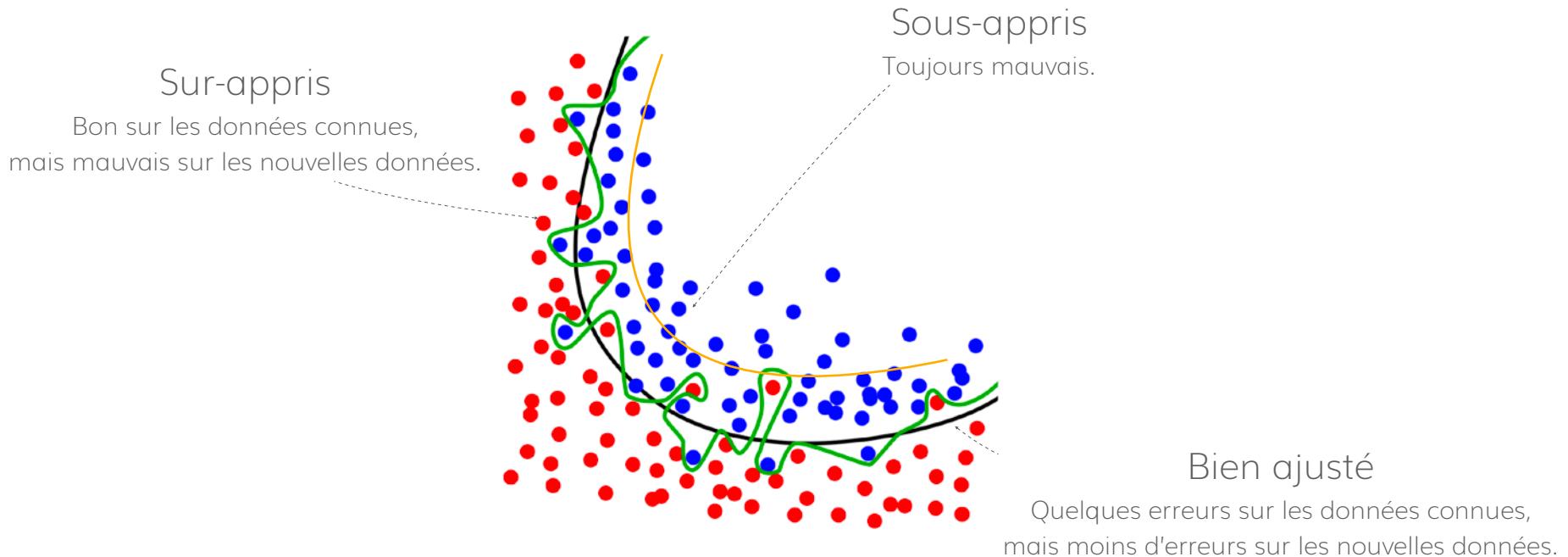


Pourquoi ne pas simplement  
apprendre les étiquettes  
par cœur ?

Le défi majeur de la **généralisation**.

# Un modèle sur-appris généralise mal à de nouvelles données.

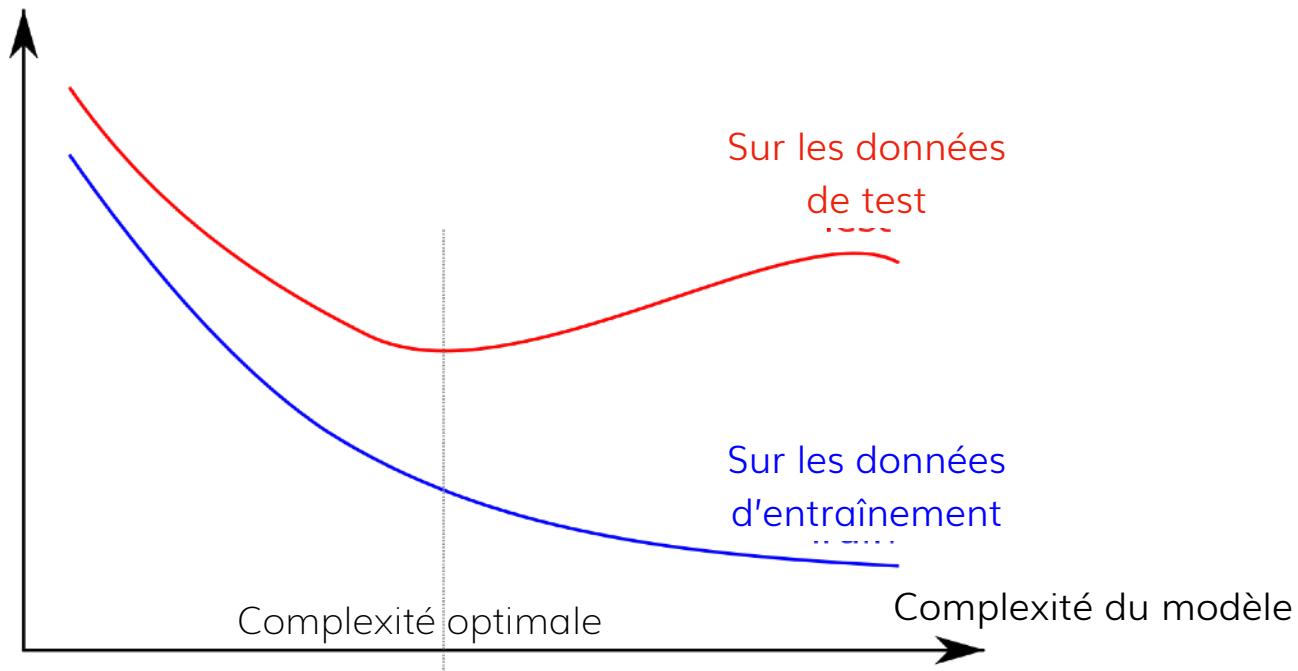
Apprendre aussi à s'écartier des étiquettes connues.



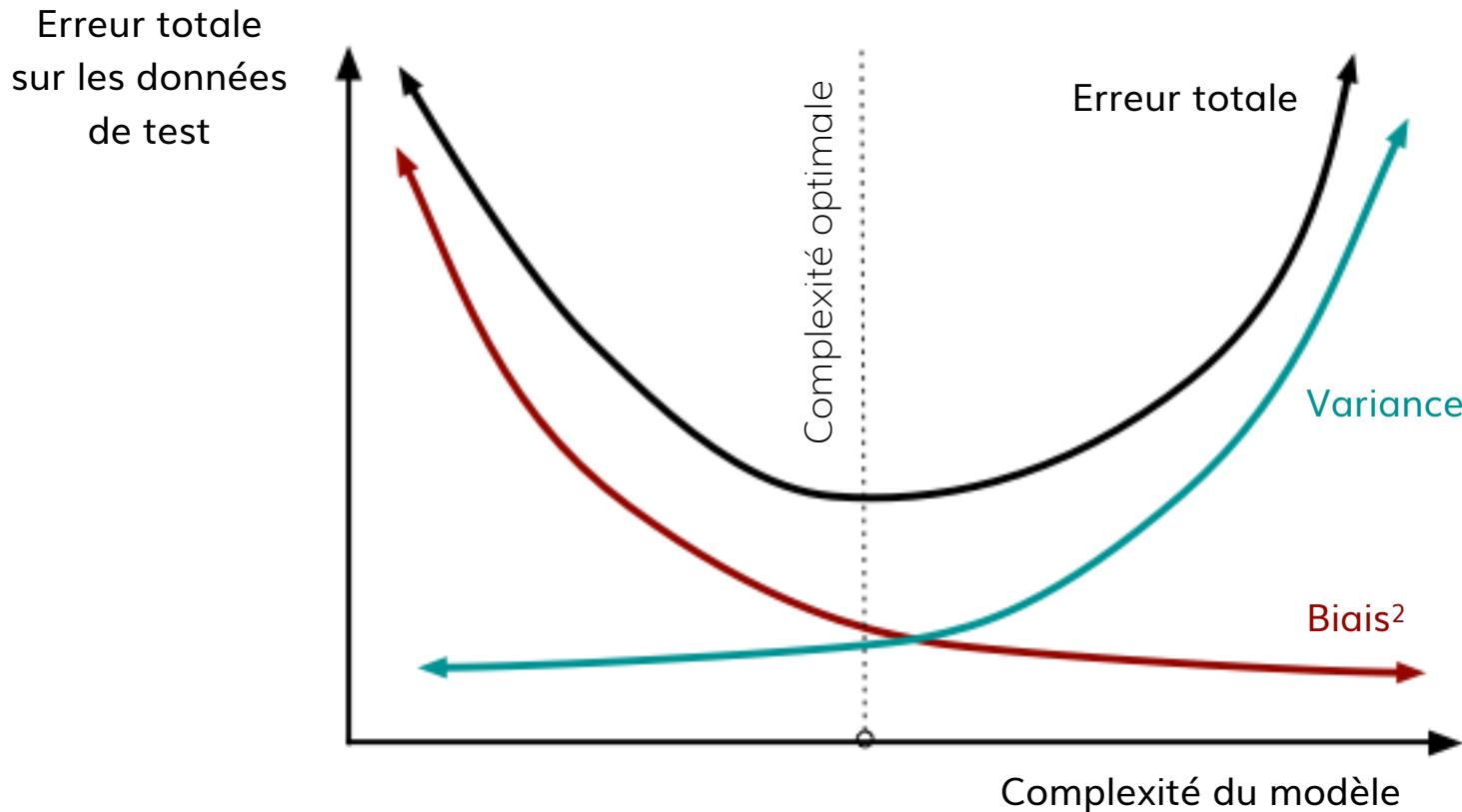
# Trouver le juste compromis.

## Courbes d'apprentissage

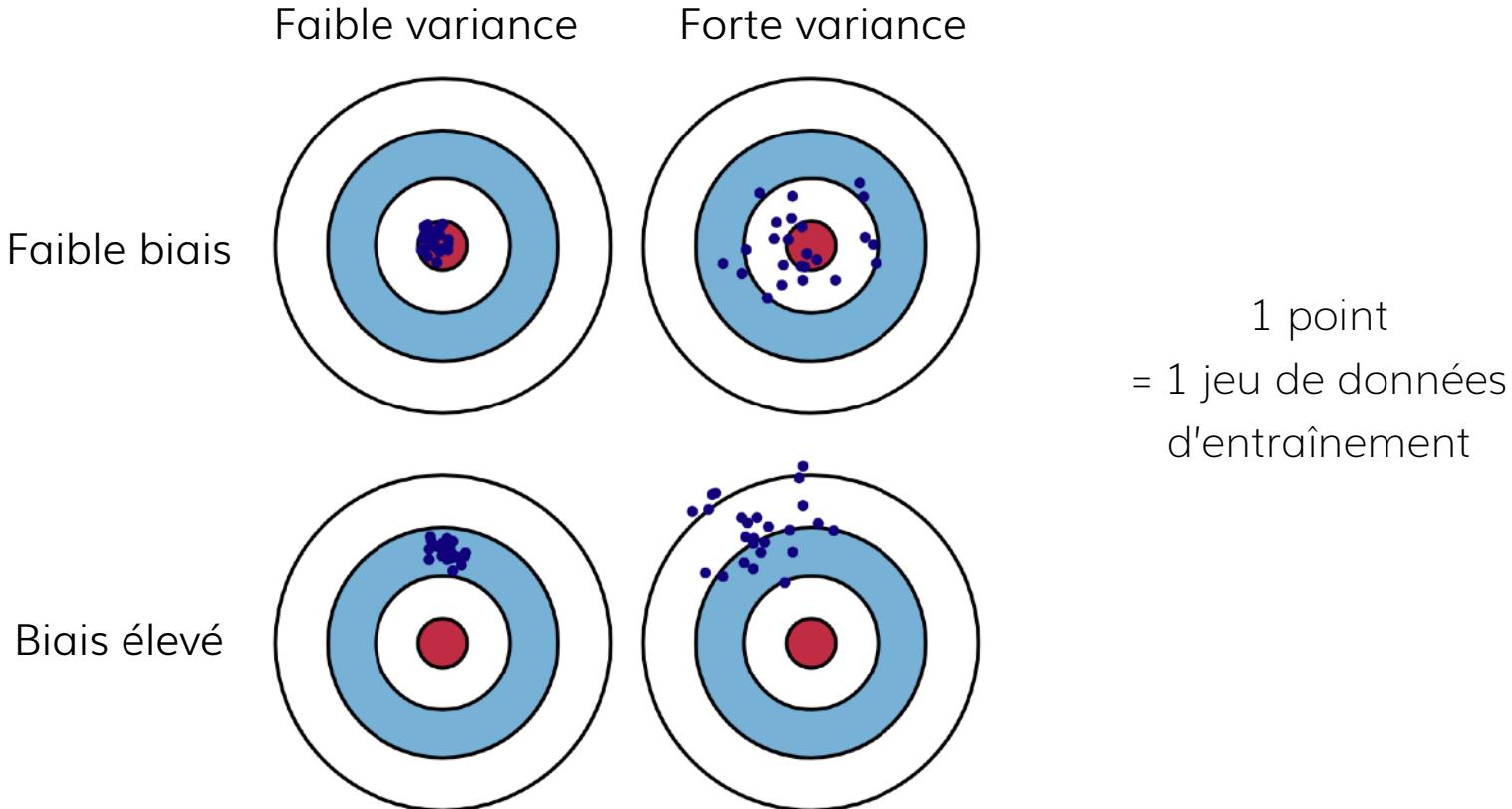
Fonction de coût (erreur totale)



# Une explication théorique : le compromis biais-variance.



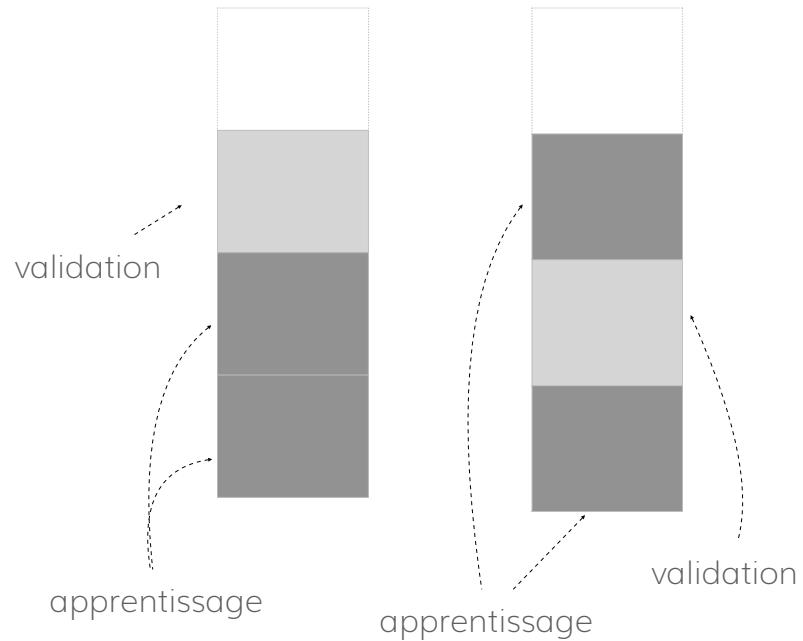
# Le compromis biais-variance.



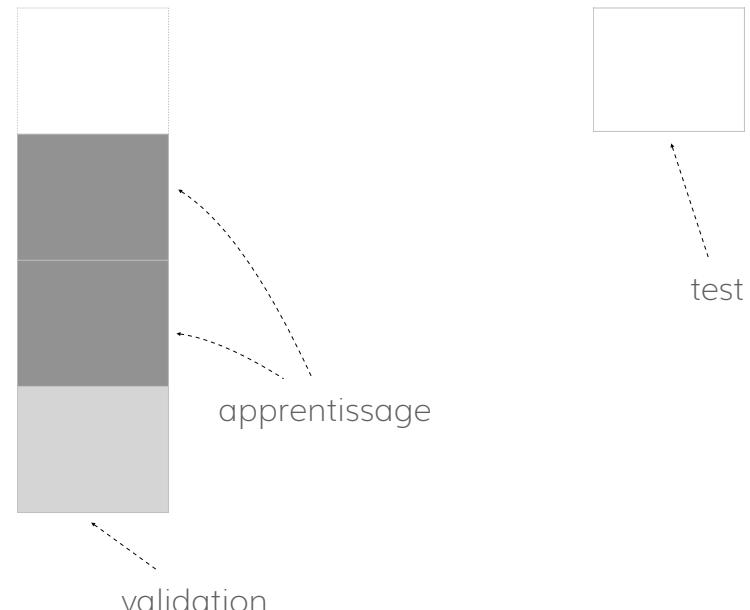
# Toujours évaluer sur un jeu de données à part.

Mettre de côté un jeu de *test* et ne pas y toucher jusqu'à l'évaluation finale.

Ajustement du modèle et validation croisée



Évaluation finale



# Qu'est-ce qu'un bon modèle (supervisé) ?

Comment évaluer la qualité d'un modèle ?

# 1<sup>er</sup> cas : la régression — l'étiquette est un nombre quelconque.

Erreur moyenne absolue

$$|y_1 - \hat{y}_1| + \dots$$

A diagram illustrating the absolute error. A dashed horizontal line segment connects two points. The left point is labeled "Vraie étiquette" and the right point is labeled "Étiquette prédite". Arrows point from the text labels to their respective points on the line.

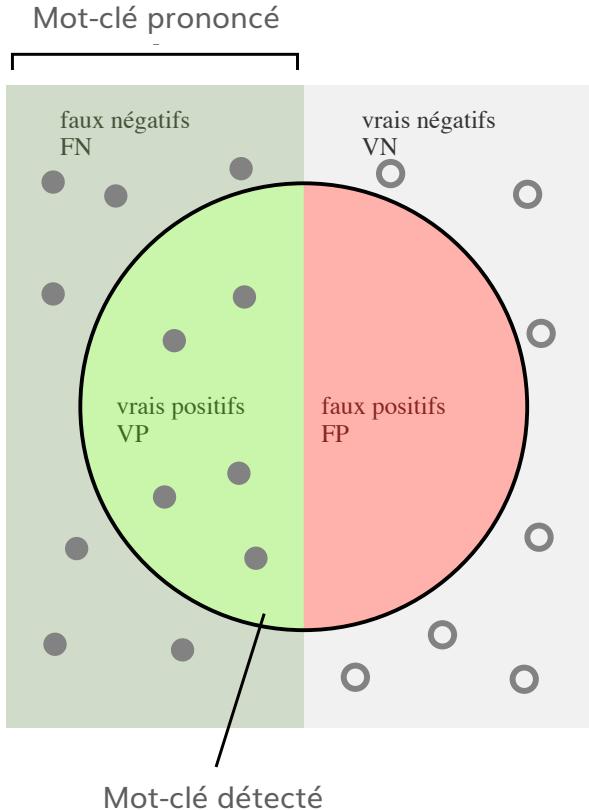
Erreur moyenne relative

$$\left| \frac{y_1 - \hat{y}_1}{y_1} \right| + \dots$$

A diagram illustrating the relative error. A dashed horizontal line segment connects two points. The left point is labeled "Vraie étiquette" and the right point is labeled "Étiquette prédite". Arrows point from the text labels to their respective points on the line. Below the line, the text "On divise par la vraie étiquette" is written, explaining the division step.

## 2<sup>e</sup> cas : la classification — l'étiquette est 0 ou 1.

Ex : détection de mot-clé. Une erreur d'en vaut pas une autre.



Vraie étiquette

Étiquette prédite

	0 (mot-clé non-détecté)	1 (mot-clé détecté)
0 (mot-clé non-prononcé)	Vrais Négatifs	Faux Positifs
1 (mot-clé prononcé)	Faux Négatifs	Vrais Positifs

Parmi les sons sans mot-clé,  
combien ne génèrent effectivement  
pas de détection ?

$$\text{Spécificité} = \frac{\text{VN}}{\text{VN} + \text{FP}}$$

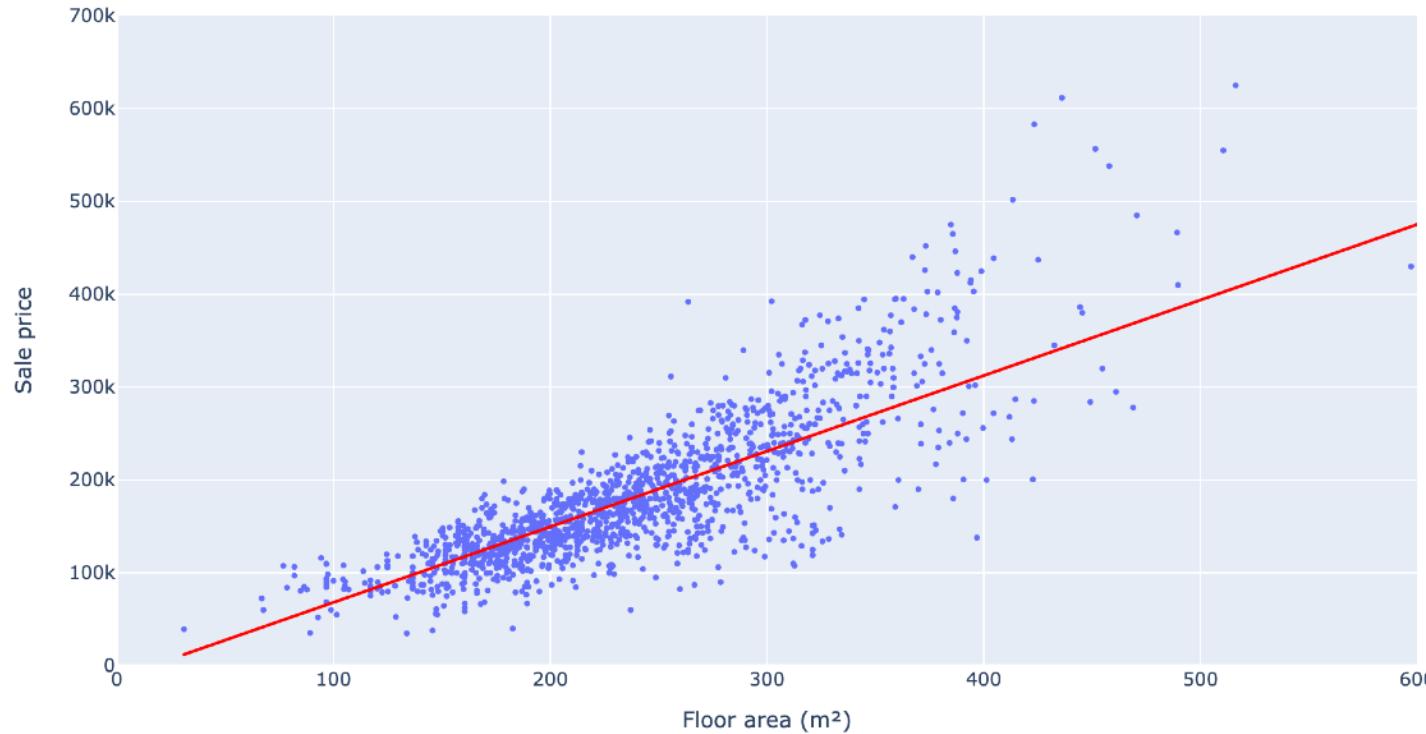
Parmi les mots-clés prononcés,  
combien sont détectés ?

$$\text{Sensibilité} = \frac{\text{VP}}{\text{FN} + \text{VP}}$$

Qu'est-ce qu'un  
réseau de neurones (artificiel) ?

# Régression linéaire — Et si les étiquettes sont seulement 0 ou 1 ?

Houses sale price as a function of floor area



Prix

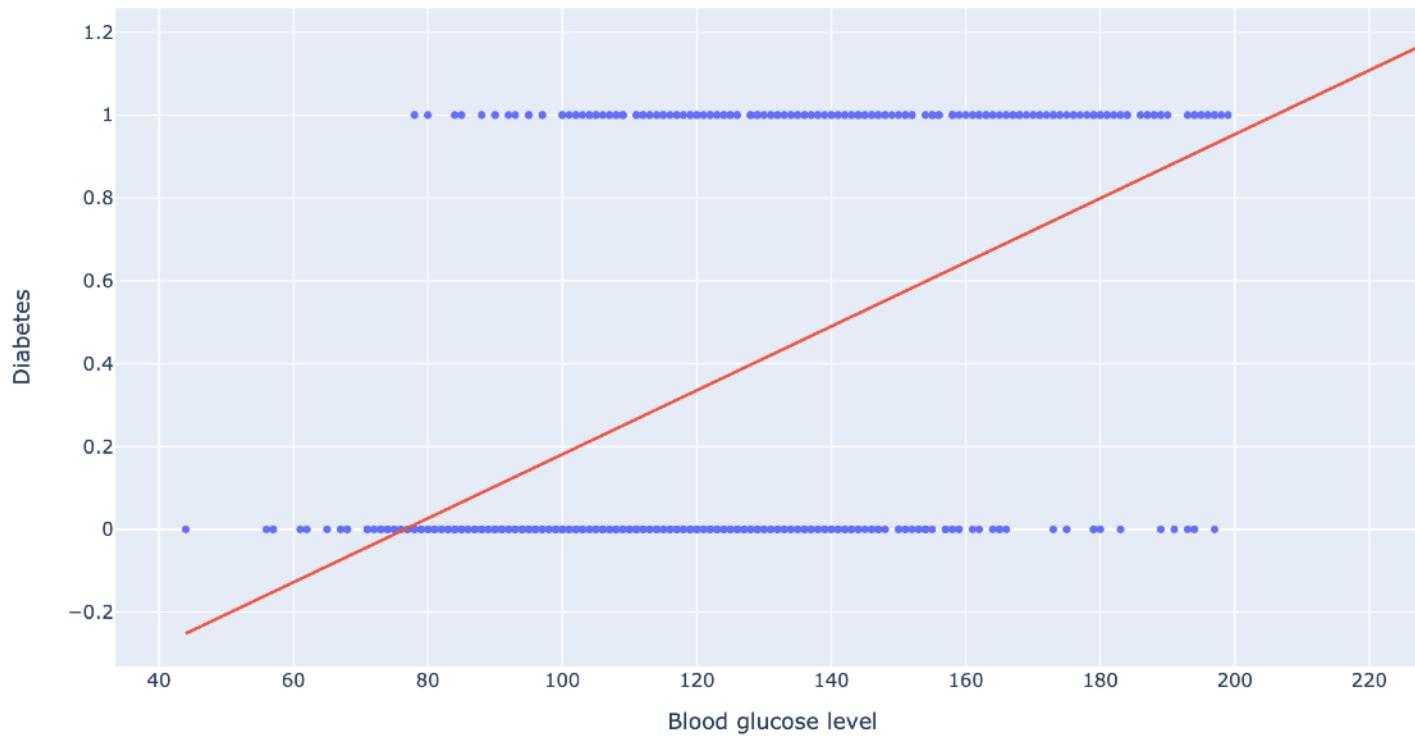
Surface habitable

$$y = ax + b$$

Paramètres

# Régression linéaire — Et si les étiquettes sont seulement 0 ou 1 ?

Diabetes condition as a function of blood glucose level



Score de diabète

Score de diabète

Taux de glucose sanguin

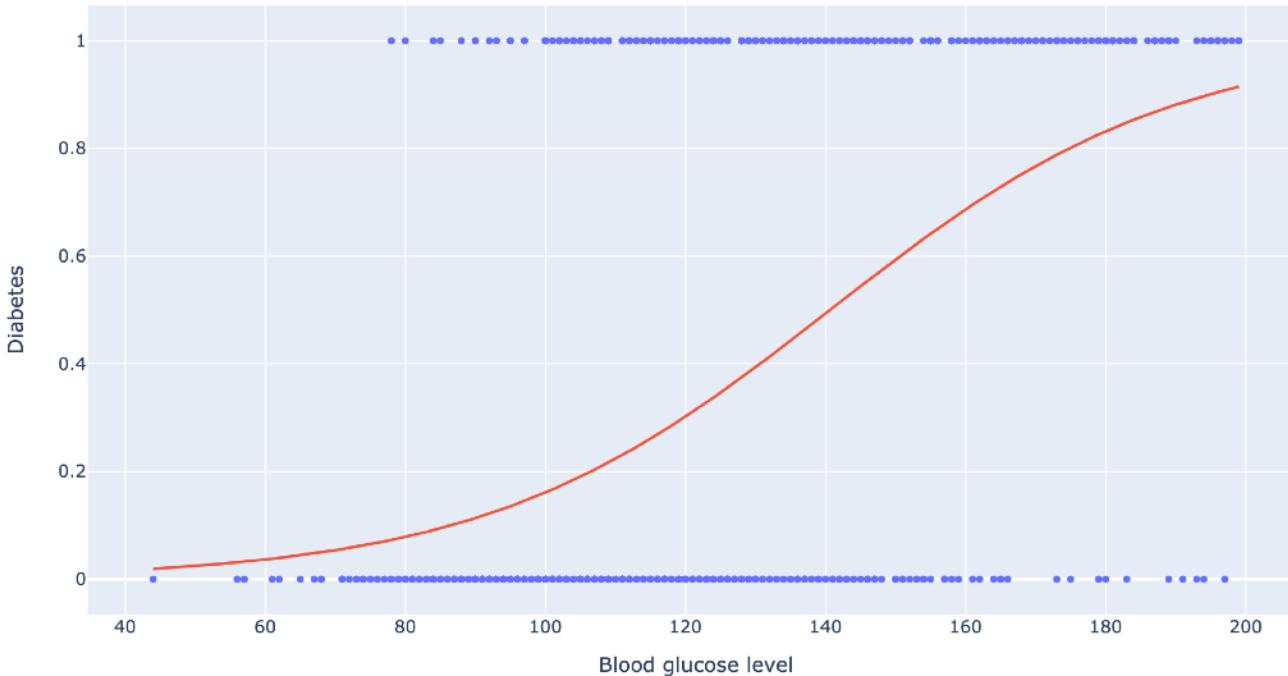
$y = ax + b$

Paramètres

# La régression linéaire devient régression logistique.

On applique un "seuillage doux" pour produire un score de "décision".

Diabetes condition as a function of blood glucose level



Score de risque de diabète

Taux de glucose sanguin

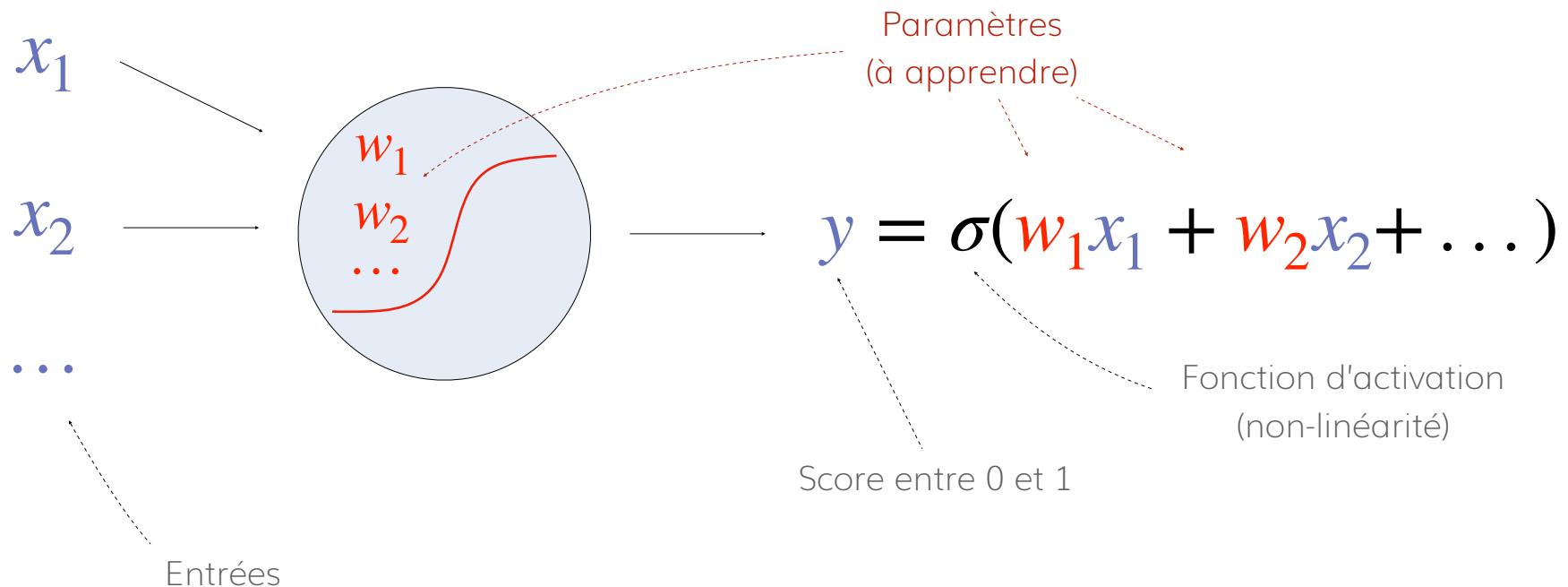
$$y = \sigma(ax + b)$$

Paramètres

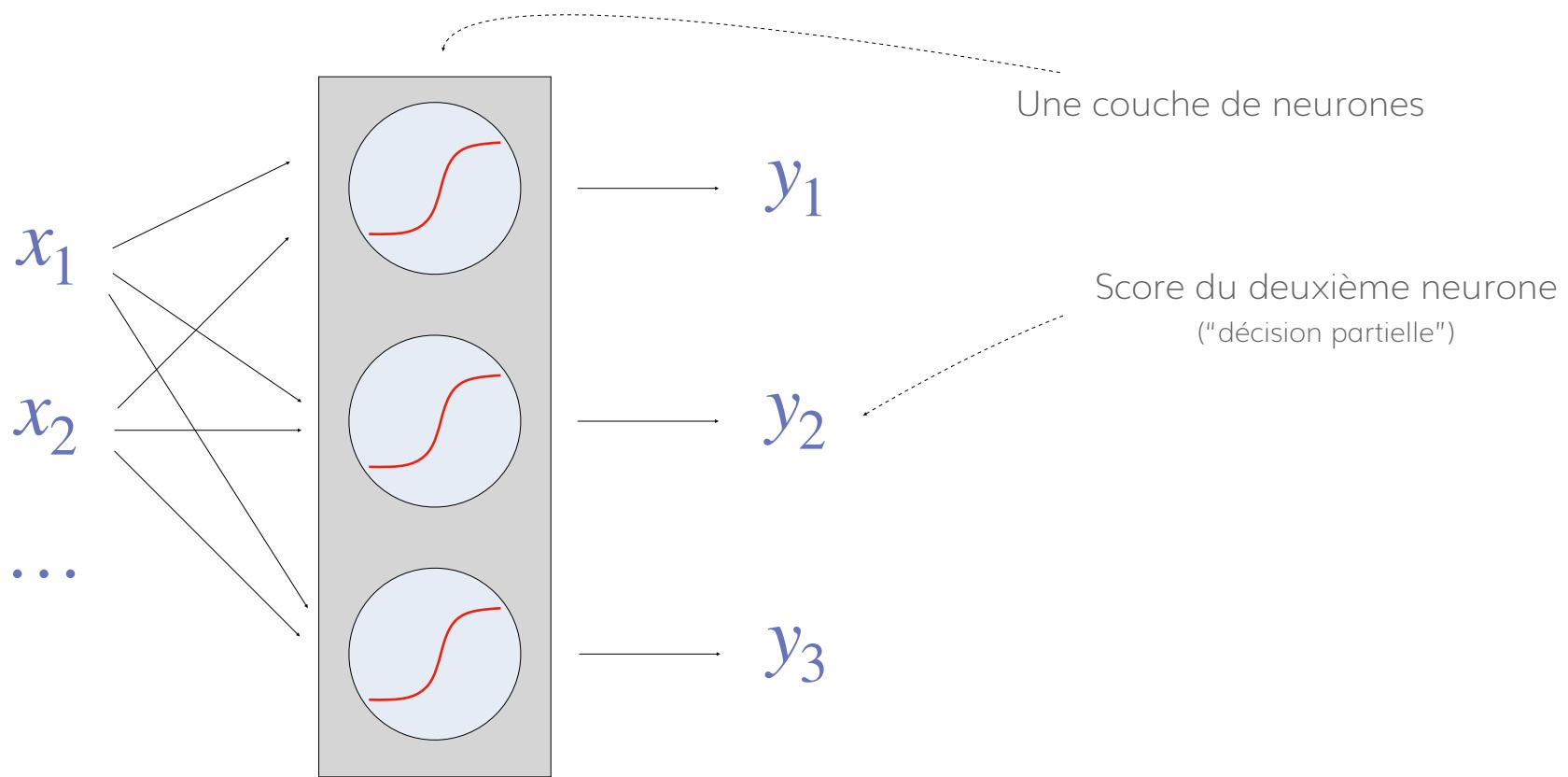
Fonction d'activation (non-linéarité)

The diagram illustrates the logistic regression model. It shows a blue dashed arrow pointing from the text "Taux de glucose sanguin" to the input variable "x" in the equation. A red dashed arrow points from the text "Paramètres" to the parameters "a" and "b". A vertical dotted line connects the text "Fonction d'activation (non-linéarité)" to the sigmoid curve in the plot.

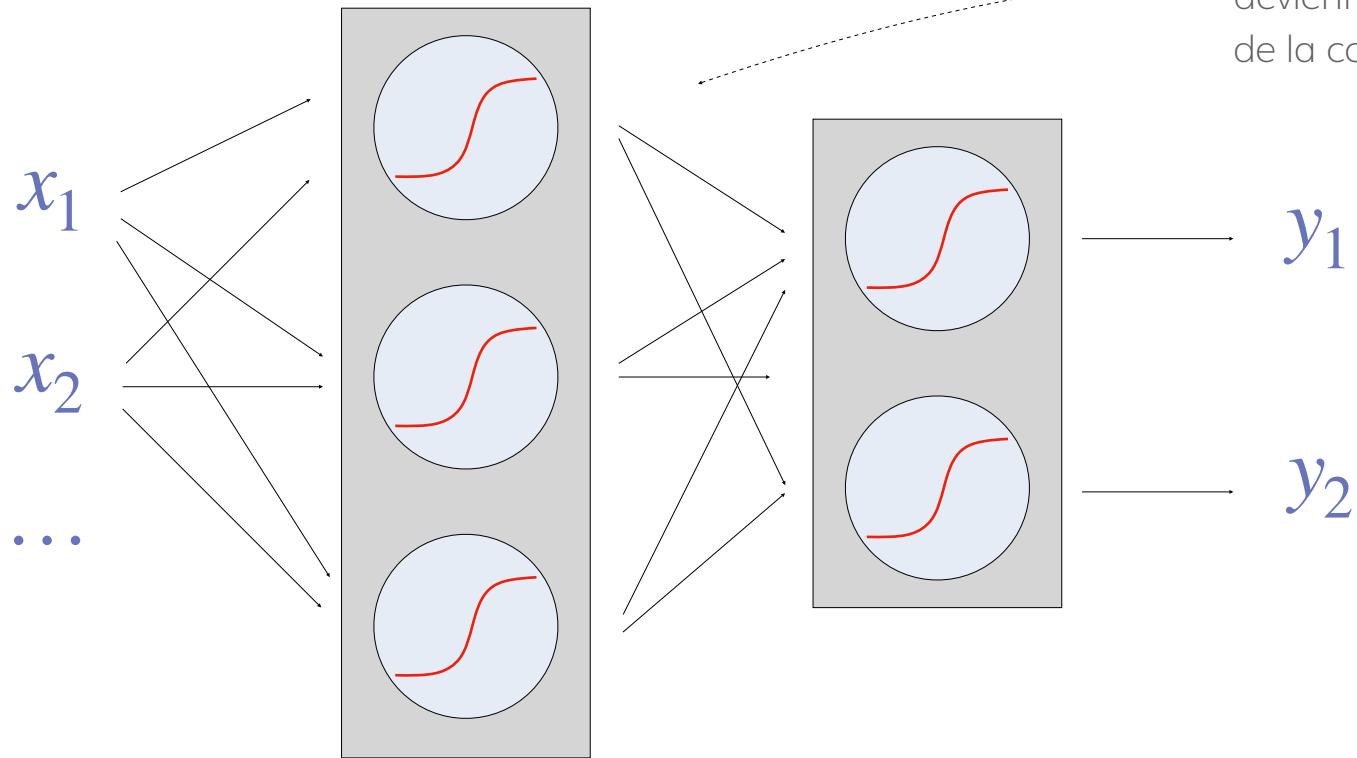
# Un neurone artificiel “prend une décision”.



Chaque neurone artificiel “prend une décision partielle”.

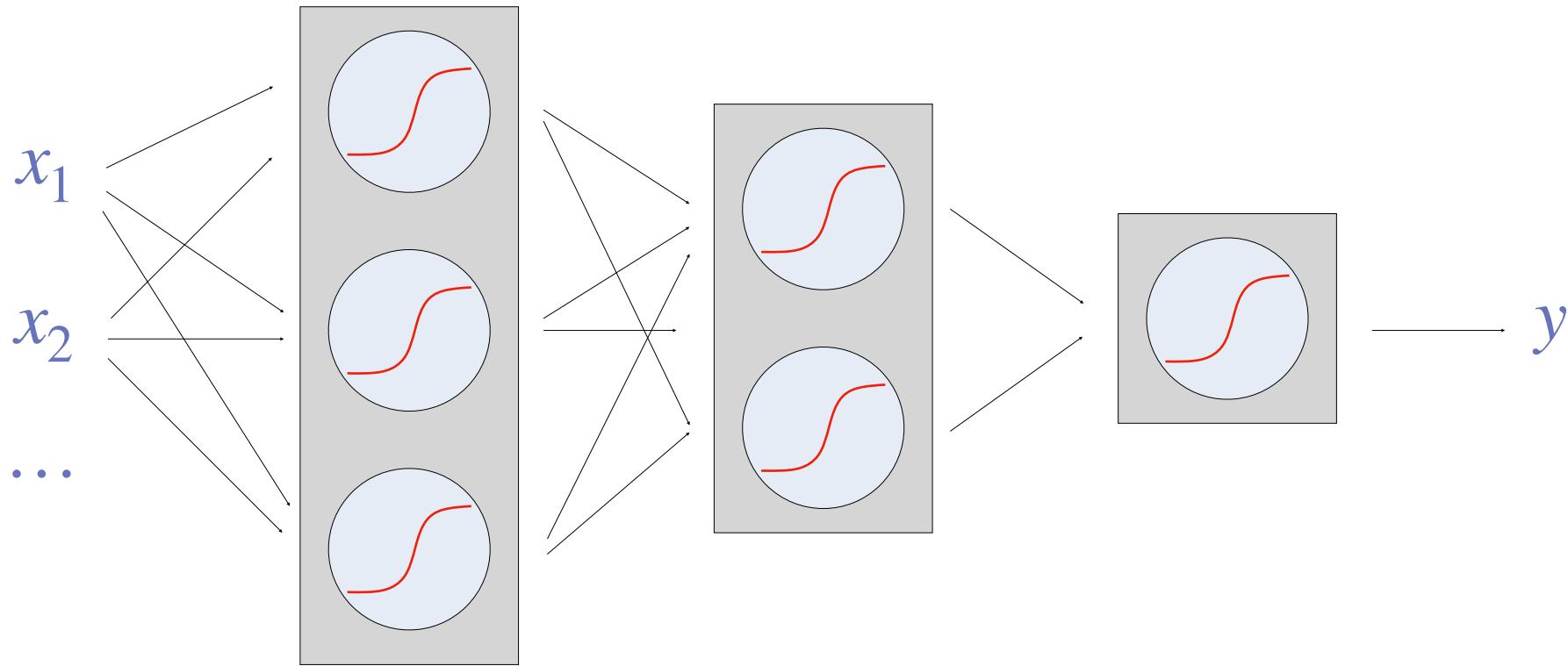


# On superpose des couches de neurones.

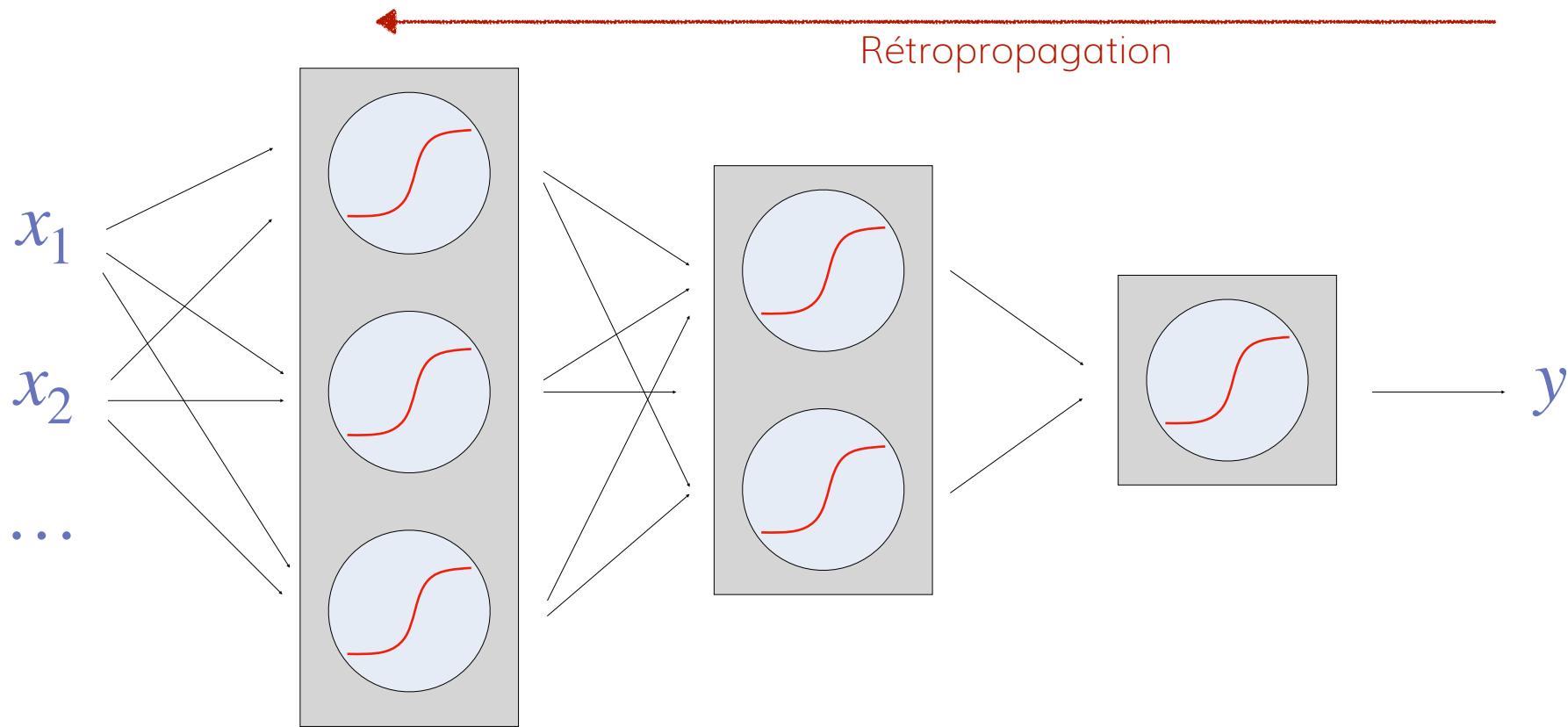


Les sorties de d'une couche deviennent les entrées de la couche suivante.

Le dernier neurone donne le résultat final.

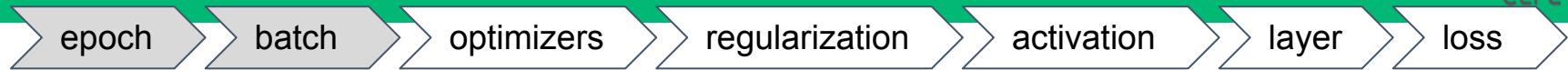


Les (nombreux) poids sont appris, pour minimiser l'erreur totale.

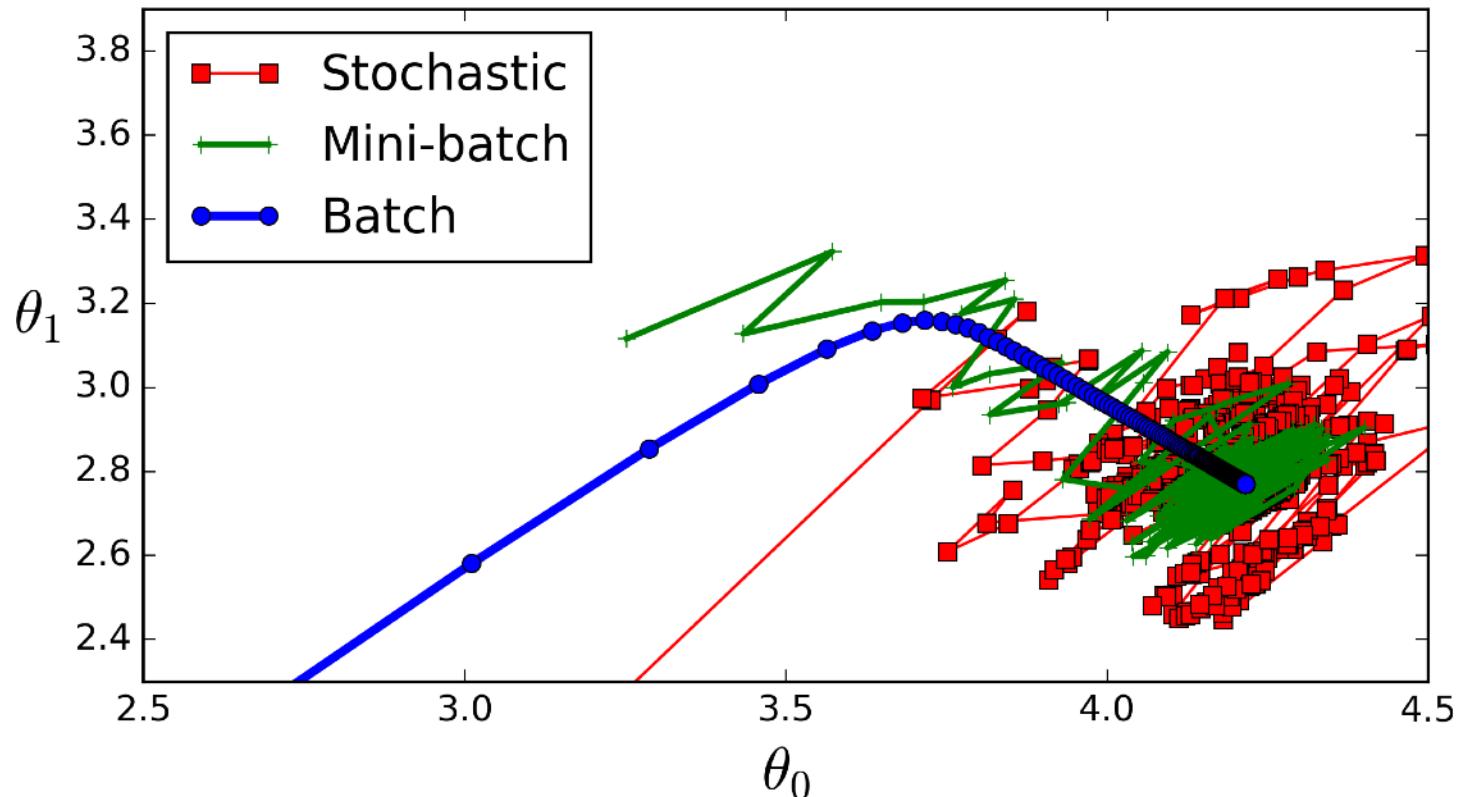
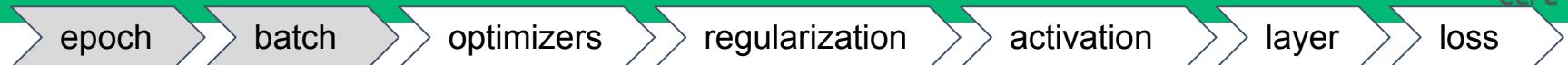


# Training a neural network

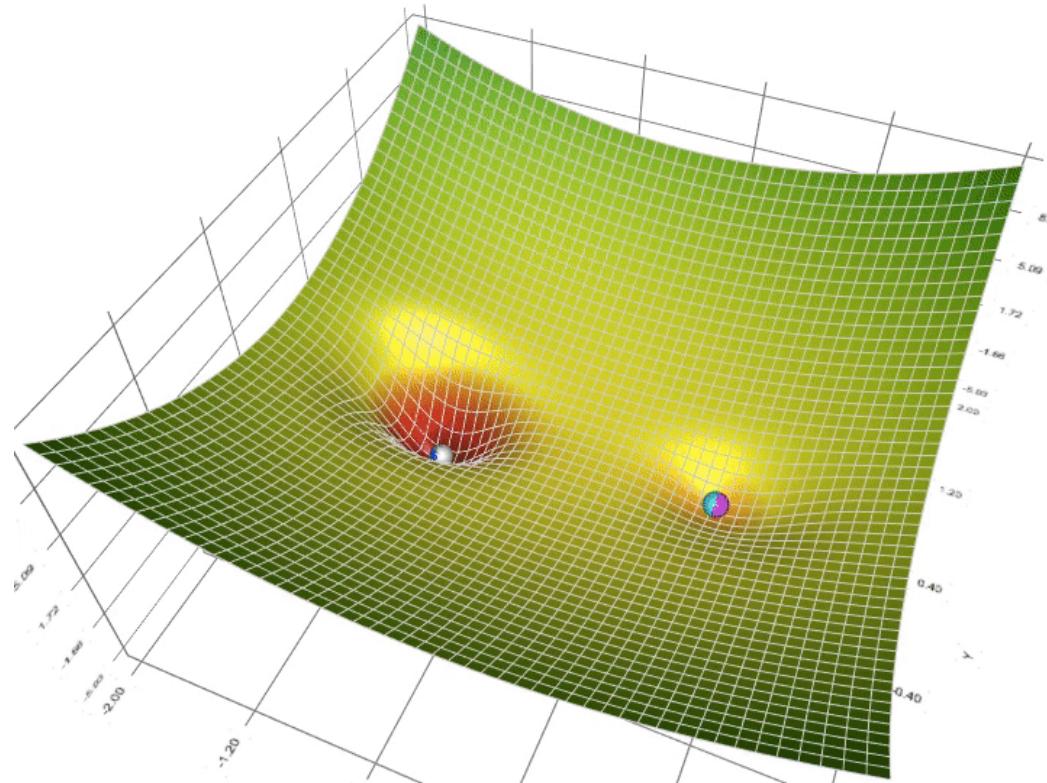
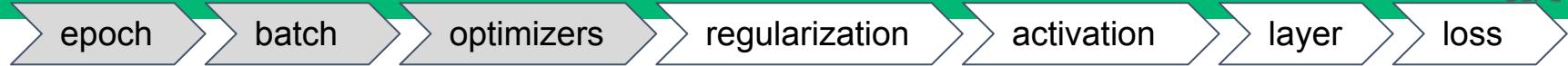
# Batch size - analogy



## Batch size

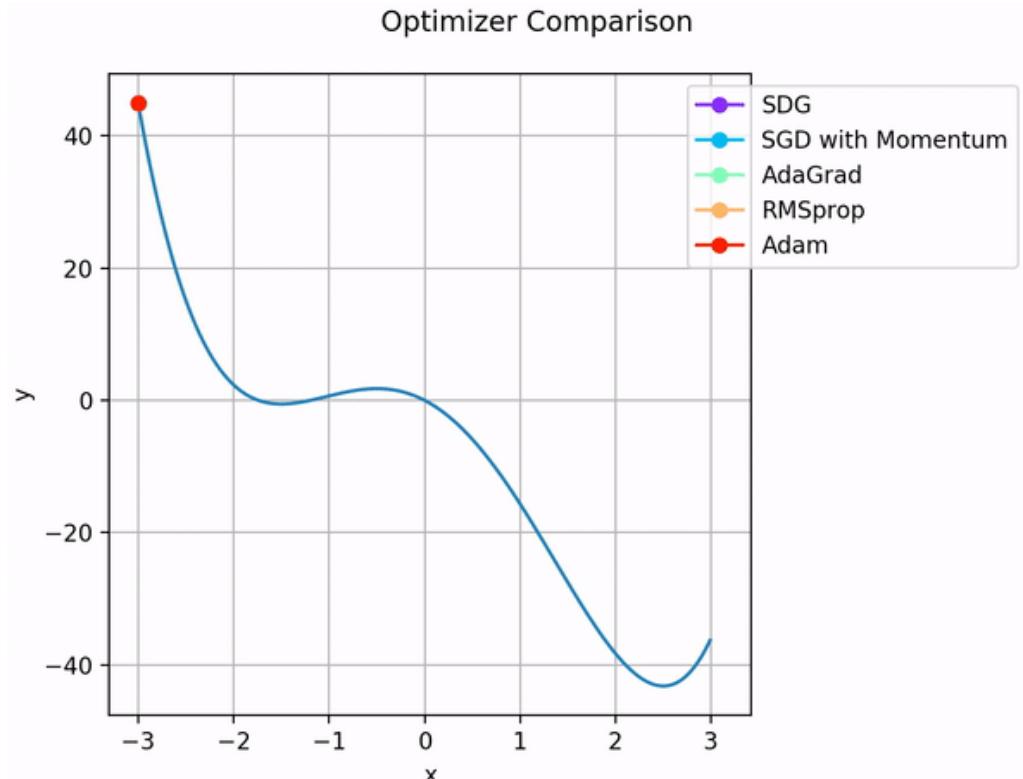
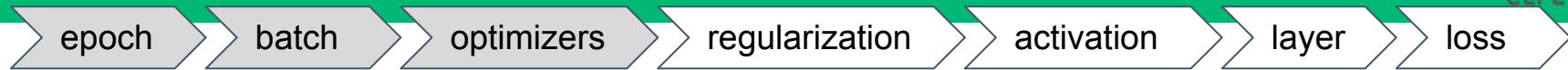


# Optimizers



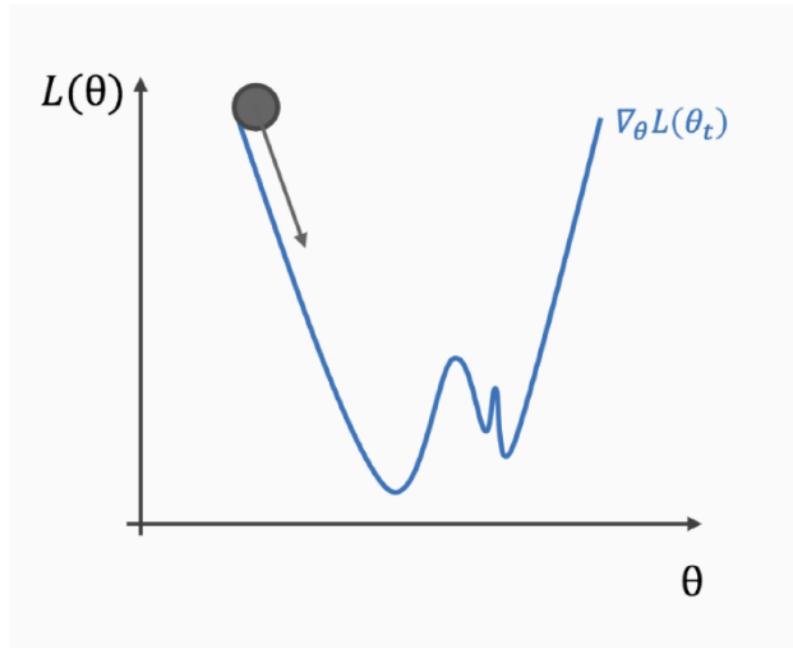
<http://ruder.io/optimizing-gradient-descent/>

# Optimizers



# SGD: stochastic gradient descent

epoch    batch    optimizers    regularization    activation    layer    loss



$$w_{t+1} = w_t - \lambda \cdot \nabla_w \mathcal{L}(w_t),$$

where  $\lambda$  : learning rate  
and  $\nabla_w \mathcal{L}(w_t)$  : gradient of  $\mathcal{L}(w)$  w.r.t.  $w$

# Momentum

epoch

batch

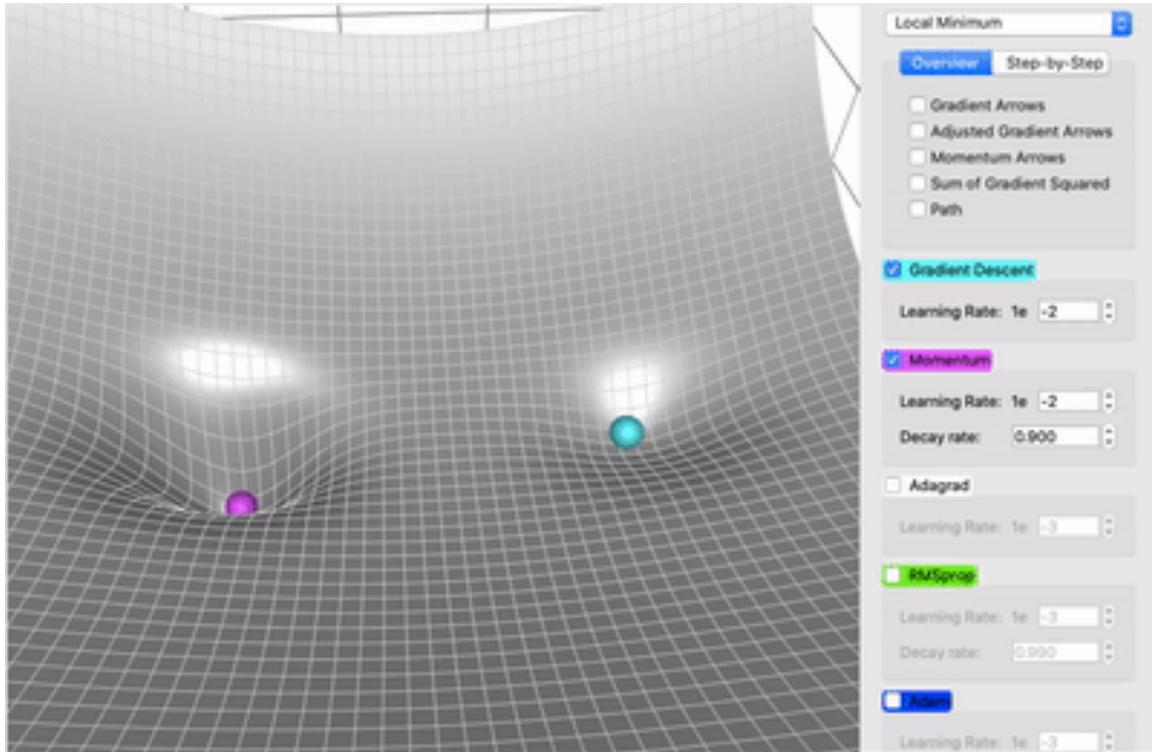
optimizers

regularization

activation

layer

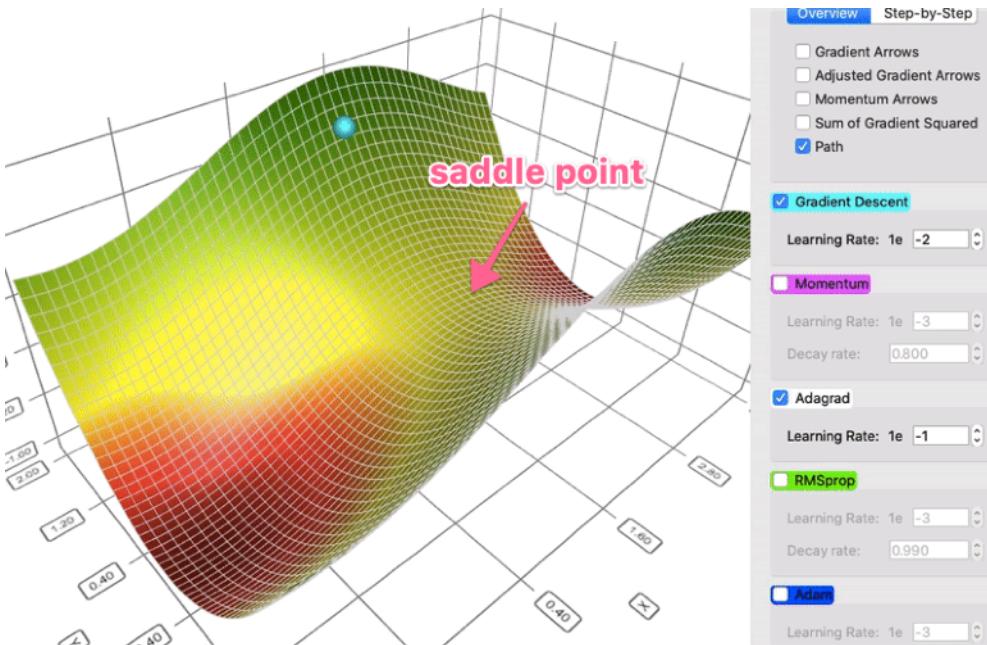
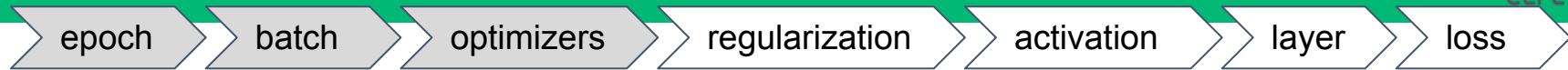
loss



Momentum (magenta) vs. Gradient Descent (cyan) on a surface with a global minimum (the left well) and local minimum (the right well)

= like a real ball, accumulate energy and speed when going down, and slow down when going up.

# Adagrad and RMSProp



In ML, some features are very sparse. AdaGrad addresses this problem using this idea: the more you have updated a feature already, the less you will update it in the future, thus giving a chance for the other features (for example, the sparse features) to catch up.

**It adapts the learning rate to the parameters.**

RMSProp : same as AdaGrad but "forgets" about old updates (exponentially-weighted sum instead of sum of past updates).

## Adam (Adaptive Moment Estimation)



Best of the two worlds (momentum + learning rate adapted to the parameters)

**Adam** empirically works well, and thus in recent years,  
it is commonly the go-to choice of deep learning problems.

# Backpropagation

**Goal:** compute partial derivatives for each weight!

$$w_{i,j} \leftarrow w_{i,j} - a \frac{\partial E(D, \mathbf{w})}{\partial w_{i,j}}$$

This is complex!

Let's try to write it as a function of  $X$ ,  $y$  and other parameters of the next layers!

<https://www.nature.com/articles/323533a0>

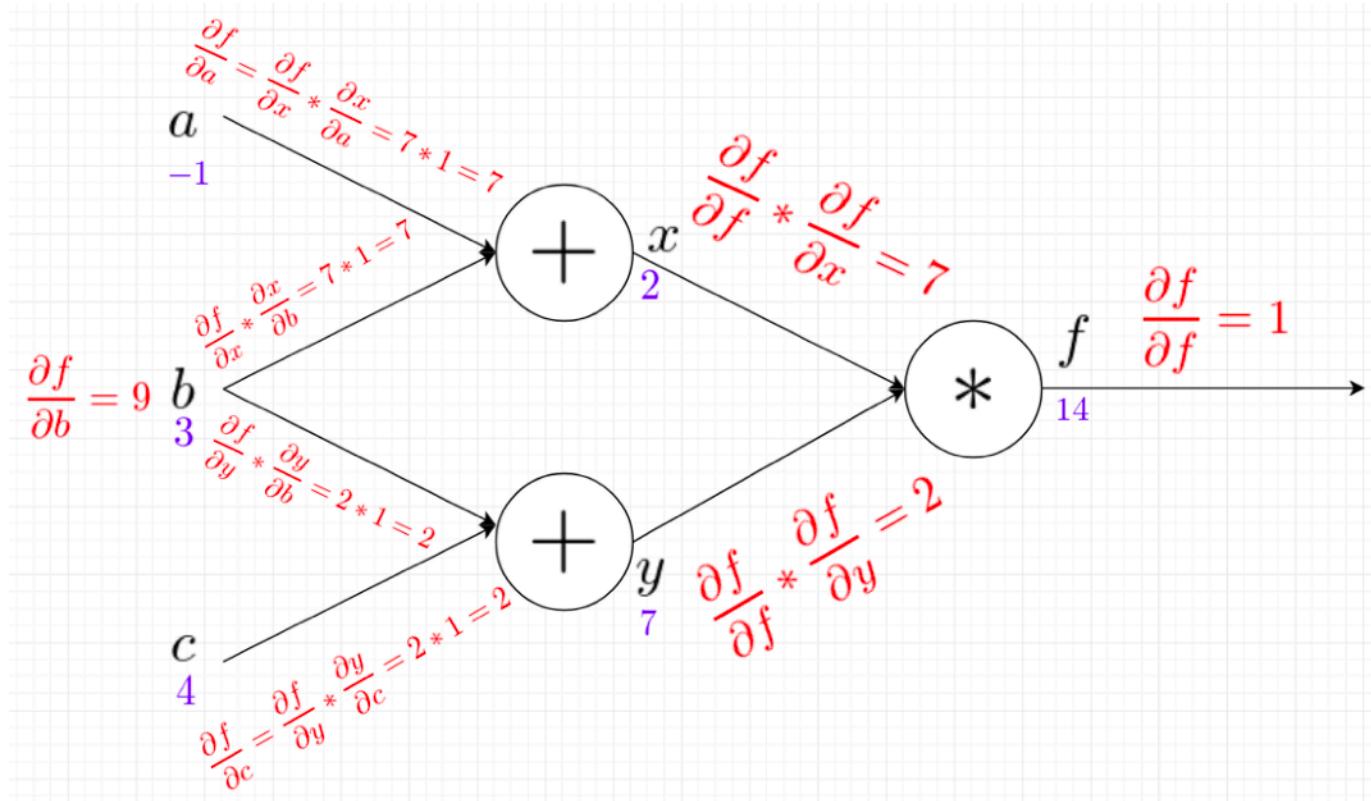


# Backpropagation

**Use of the chain rule!**

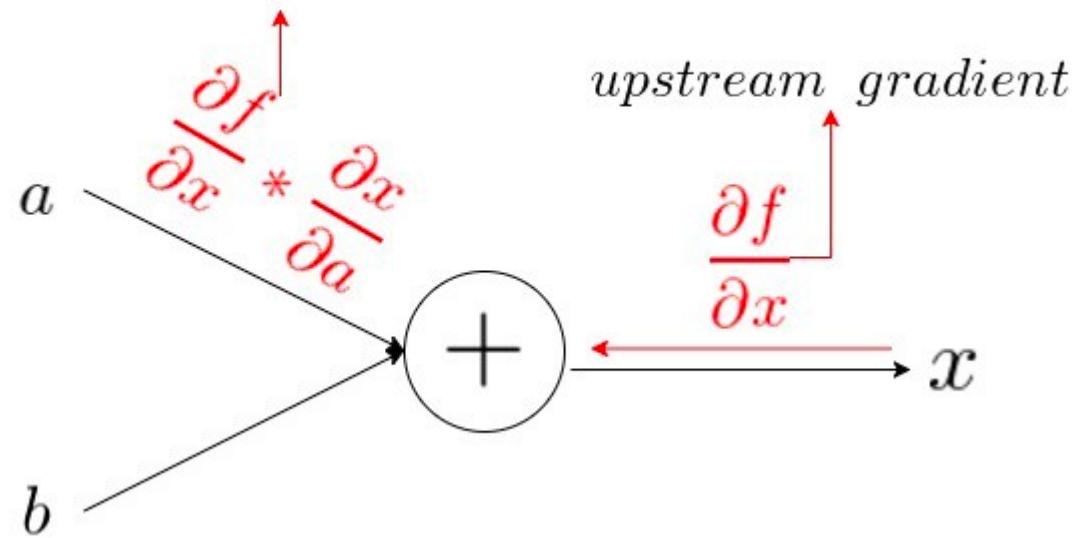
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

# Backpropagation



# Backpropagation

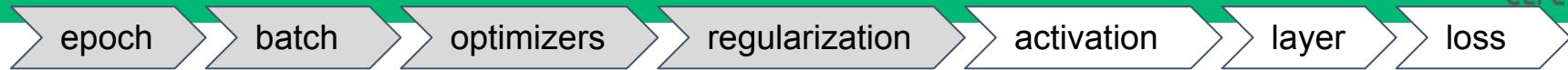
*upstream gradient \* local gradient*





# Regularization for neural networks

# Weights regularization



Don't forget the norm regularizers we saw this morning !

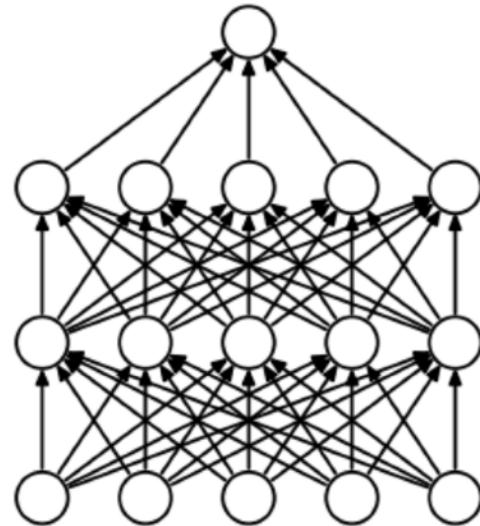
Smaller weights are, in some sense, lower complexity, and so provide a simpler and more powerful explanation for the data, and should thus be preferred.

$$\mathcal{L}(w) = \text{MSE}(y, \hat{y}(x)) + R(w)$$

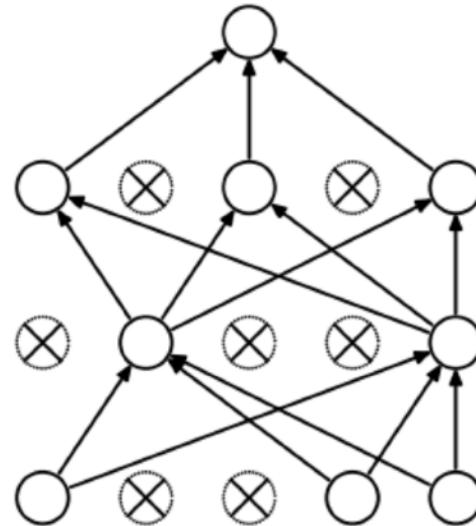
$$\left\{ \begin{array}{l} R(w) = \lambda \cdot \underline{\|w\|} \\ \text{MSE}(y, w) = \sum (y - Xw)^2 \end{array} \right.$$

$\min_w \mathcal{L}(w)$  **we prevent weights from going too big (using N1, N2 ...)**

# Dropout during training



(a) Standard Neural Net

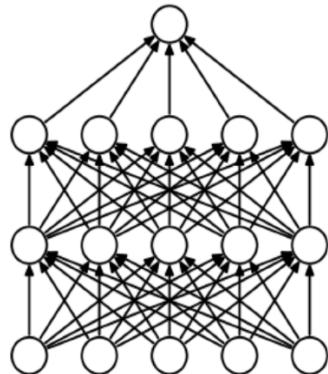


(b) After applying dropout.

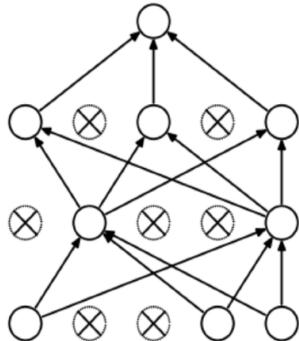
*"This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons."*

*\*ImageNet Classification with Deep Convolutional Neural Networks, by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012).*

# Dropout: some more information



(a) Standard Neural Net

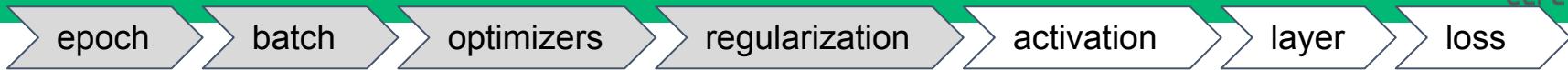


(b) After applying dropout.

There are several ideas proposed as to why implementing dropout seems to be a good idea:

1. With dropout, the weights of the nodes learned through backpropagation become somewhat more insensitive to the weights of the other nodes and learn to decide the outcome independent of the other neurons.
2. Dropout can also be thought of as an ensemble of models that share parameters. When we drop a neuron, it has no effect on the loss function and thus the *gradient* that flows through it during backpropagation is effectively zero and so its weights will not get updated. This means that we are basically *subsampling* a part of the neural network and we are training it on a single example.

# Dataset Augmentation



François Chollet ✅  
@fchollet

...

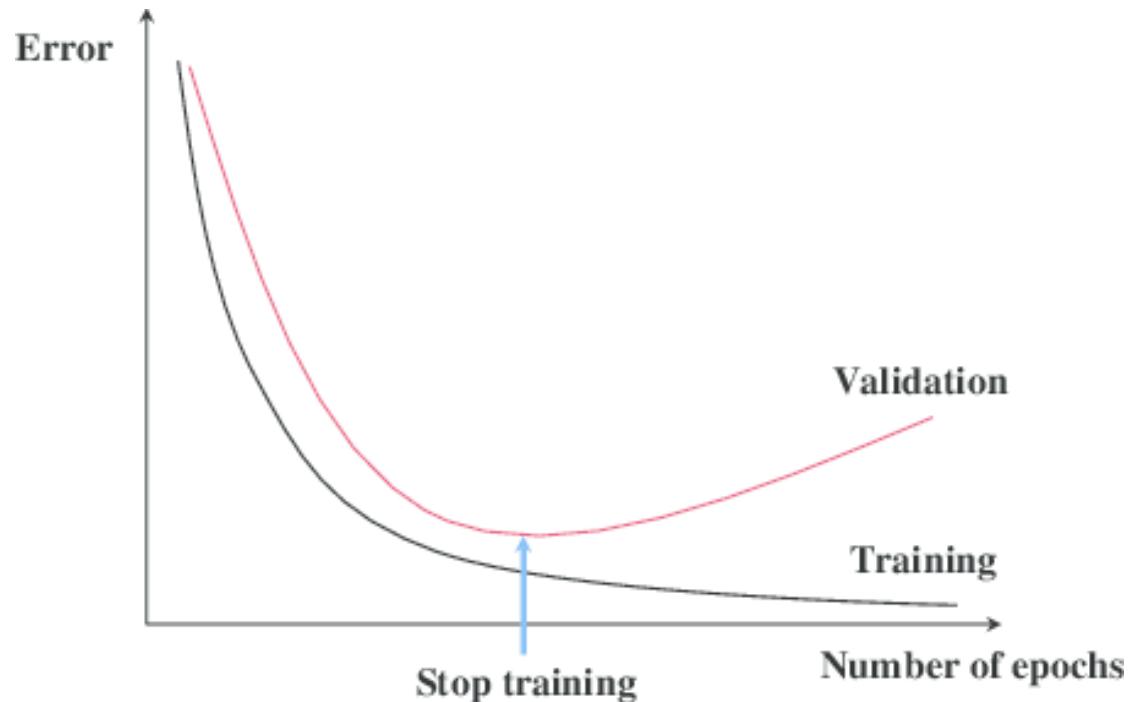
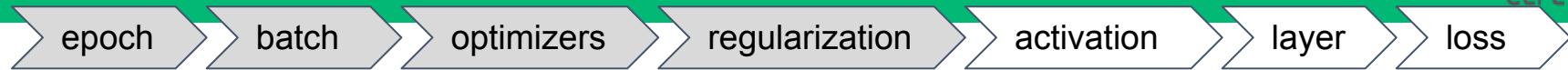
"If you get an extra 50 hours to spend on a [ML] project, chances are that the most effective way to allocate them is to collect more data rather than search for incremental modeling improvements."

For images...

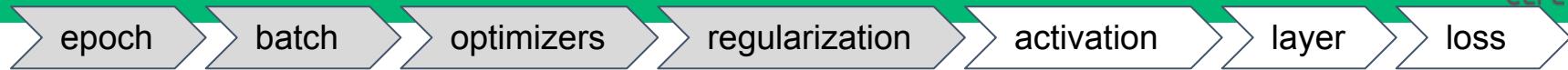
- Zoom
- Symmetry
- Rotation
- Adding some noise
- Cast to grayscale
- ...
- ?



# Early stopping



## Play with hyper-parameters through grid search



- Learning rate
- Batch size
- Number of epochs
- Network architecture (number of neurons, layers, ...)
- ...

-> search smartly through a grid in hyper-parameter space.

[Random search for hyper-parameter optimization, by James Bergstra and Yoshua Bengio \(2012\).](#)

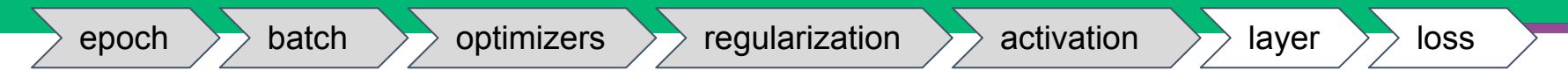
Can also use **Bayesian Optimization**.

But hyperparameters optimization is very time consuming !...



# Activation functions

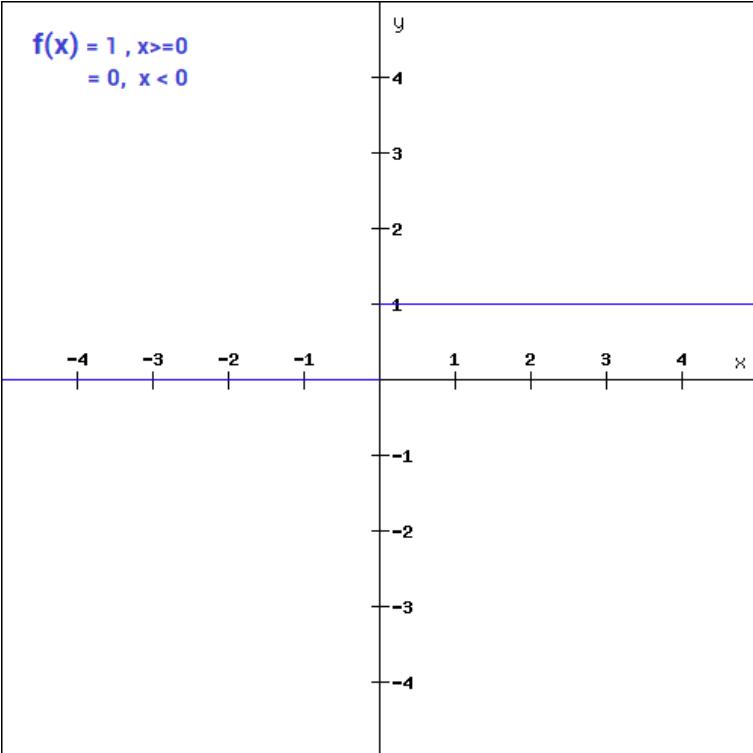
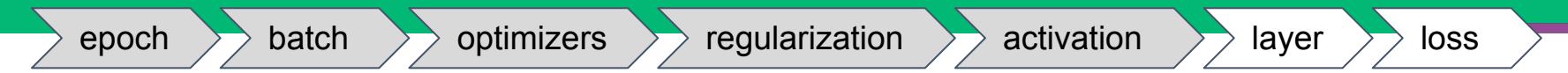
# Activation functions



*It is used to determine the output of neural network, like “yes” or “no”.*

*It can map the resulting values in between 0 to 1, or between -1 to 1, or... etc. (depending upon the function).*

# Binary step

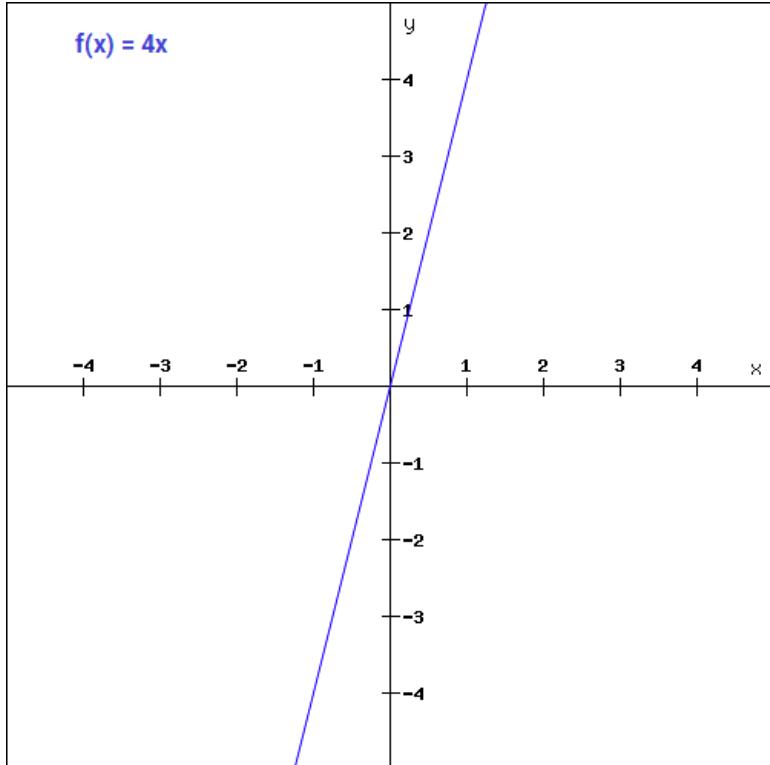
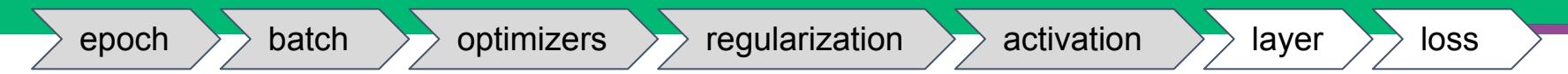


## Binary step

Cons:

- Cannot learn: the gradient is always zero.

# Linear



## Linear

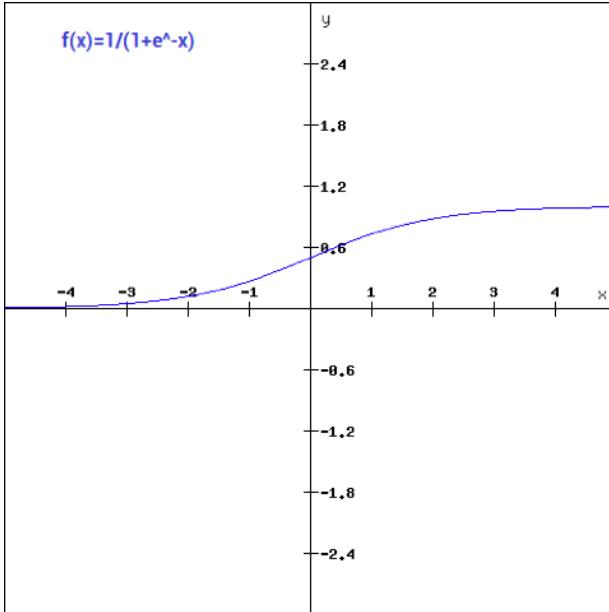
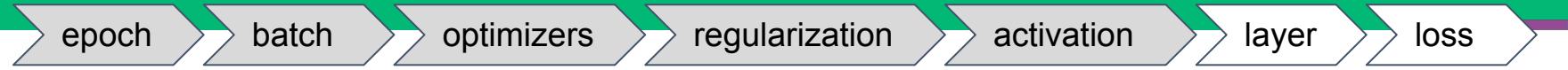
Pros:

- Easy function, does not overfit

Cons:

- Doesn't catch a high complexity

# Sigmoid



Often used for binary classification problems

## Sigmoid

### PROS:

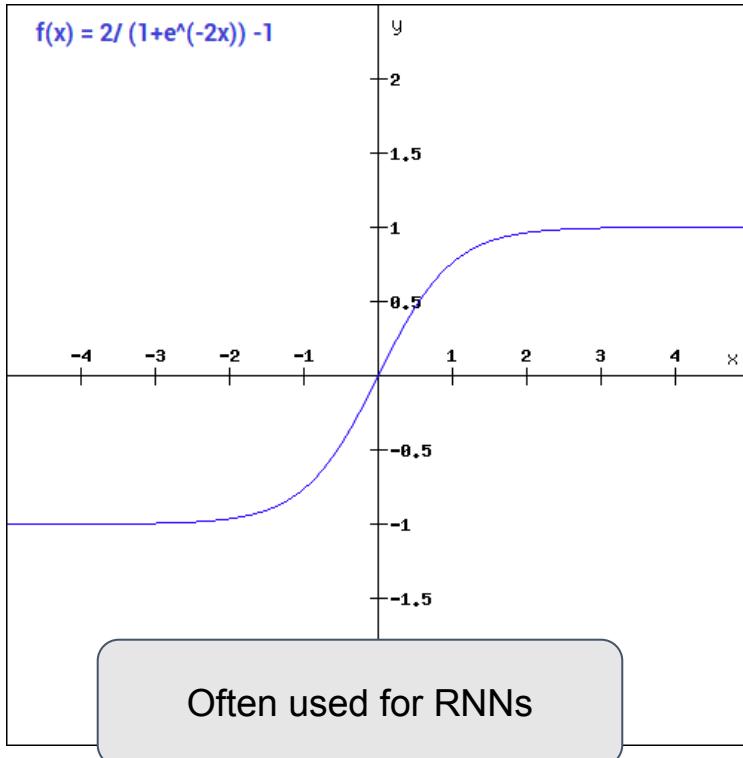
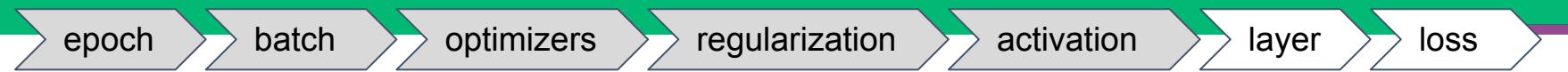
- Non-linear
- Smooth, continuous and differentiable
- Useful for predicting a probability
- Efficiently calculating gradients ((keeps in memory the feed-forward activations)):

$$\frac{d}{dx} S(x) = S(x)(1 - S(x))$$

### CONS:

- The sigmoid can cause neural network to get **stuck** at the training time (0 if very negative values).
- Sensible to the vanishing gradient problem outside [-3,3].
- Always positive

# Activation: tanh



**Tanh:** similar to sigmoid but 0-centered

#### PROS:

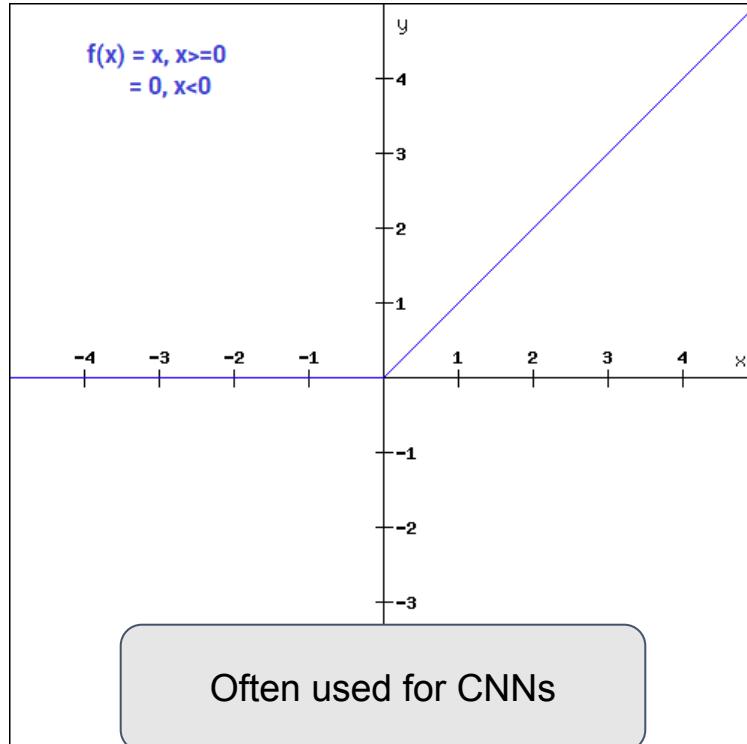
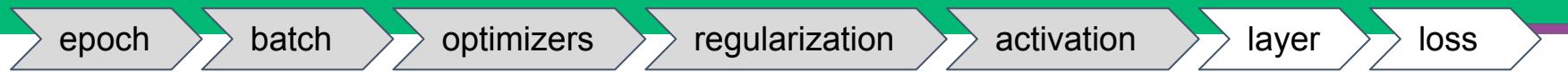
- Also “sigmoidal”
- Smooth, continuous and differentiable
- Centered around 0 (less likely to get stuck during training)
- Efficiently calculating gradients too

#### CONS:

- Sensible to the vanishing gradient problem

The tanh function is mainly used for classification between two classes, and for recurrent neural networks.

# ReLU



## Relu (rectified linear unit)

### PROS

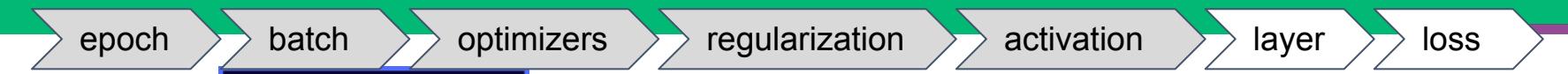
- Doesn't activate all neurons at the same time
- Far more computationally efficient (compared to tanh or sigmoid)
- Avoid vanishing gradient

### CONS:

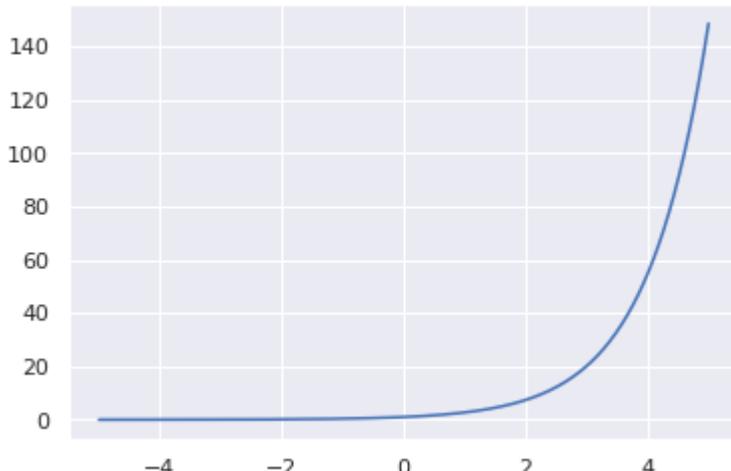
- This can create dead neurons which never get activated. This is taken care of by the 'Leaky' ReLU function.
- Sensible to exploding gradient

Often used for CNNs

# Softmax (for a last layer)



$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}}$$

Plot of  $\exp(x)$ 

Often used for multi-class classification problems

## PROS:

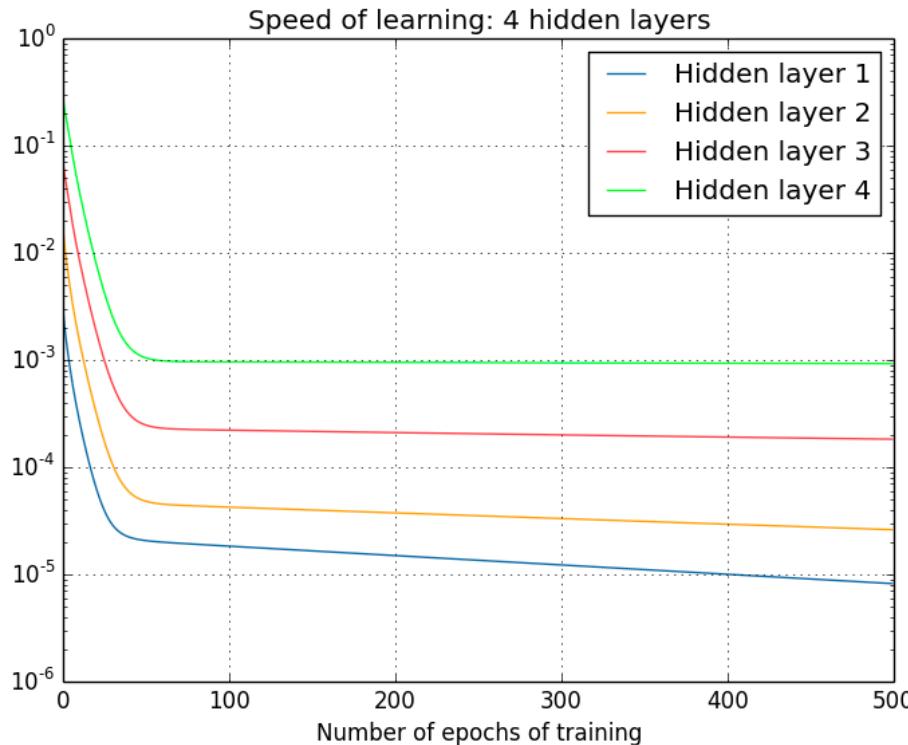
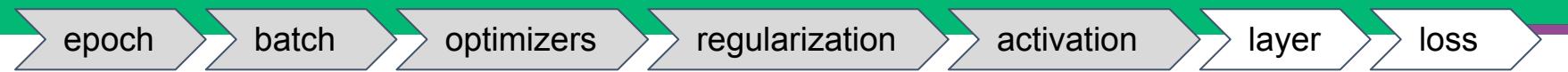
- always positive
- differentiable

*"The term softmax is used because this activation function represents a smooth version of the winner-takes-all activation model in which the unit with the largest input has output +1 while all other units have output 0."*

Neural Networks for Pattern Recognition, 1995.

*"Any time we wish to represent a probability distribution over a discrete variable with n possible values, we may use the softmax function. This can be seen as a generalization of the sigmoid function which was used to represent a probability distribution over a binary variable."*

# Vanishing gradient problem

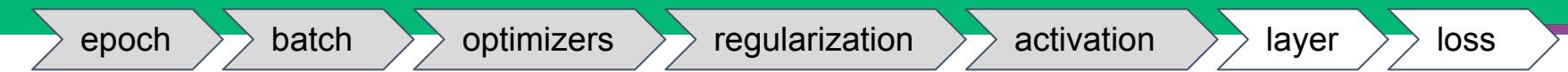


Early hidden layers learn much more slowly than later hidden layers.

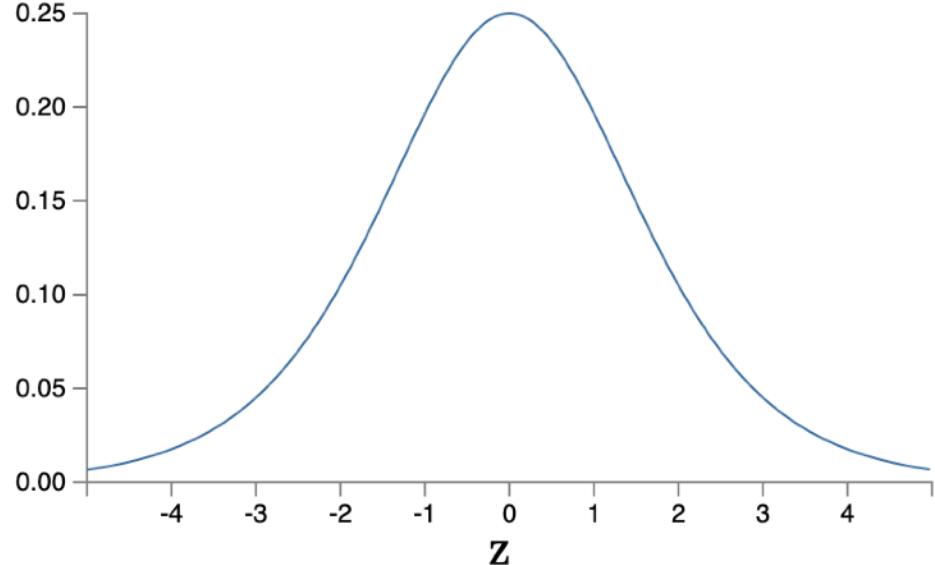
**Why?**

The fundamental problem here is that the gradient in early layers is the product of terms from all the later layers.

# Vanishing gradient problem



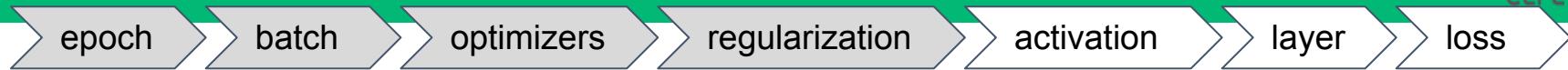
Derivative of sigmoid function



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \underbrace{w_2 \sigma'(z_2)}_{< \frac{1}{4}} \underbrace{w_3 \sigma'(z_3)}_{< \frac{1}{4}} w_4 \sigma'(z_4) \underbrace{\frac{\partial C}{\partial a_4}}_{\text{common terms}}$$

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3) \underbrace{w_4 \sigma'(z_4)}_{\text{common terms}} \frac{\partial C}{\partial a_4}$$

## Batch normalization



To facilitate learning, we typically **normalize** the initial values of our parameters by initializing them with **zero mean and unit variance (gaussian)**.

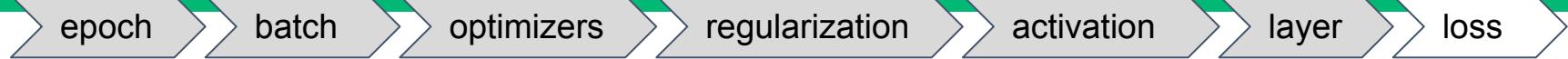
As training progresses and we update parameters to different extents, **we lose this normalization**, which slows down training (vanishing gradient, see later) and amplifies changes as the network becomes deeper.

**Batch normalization** reestablishes these normalizations for every mini-batch and changes are back-propagated through the operation as well.



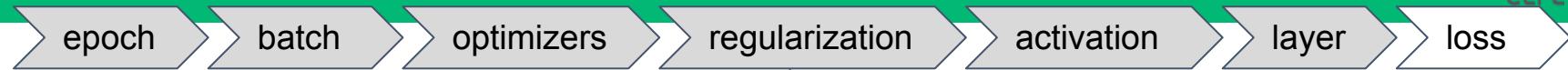
# Layers

## Layers



- Dense
- Convolution (1D, 2D, 3D)
- Dropout
- Pooling
- MaxPooling
- BatchNormalization
- Flatten
- Concat
- ...

Layers...



## Flatten

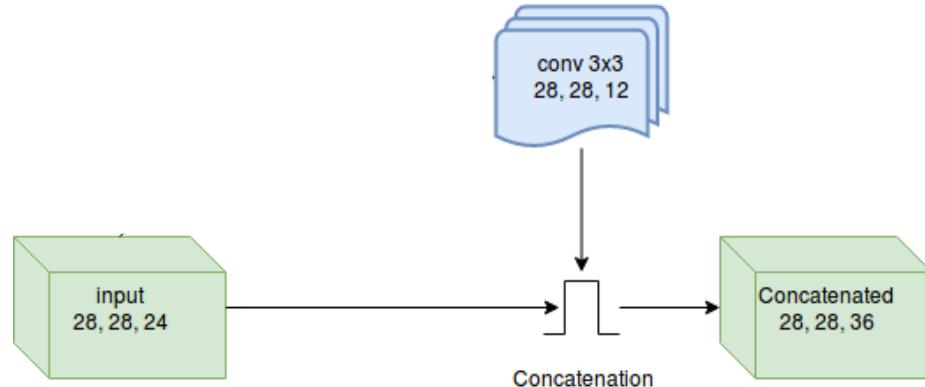
1	1	0
4	2	1
0	2	1

Flattening

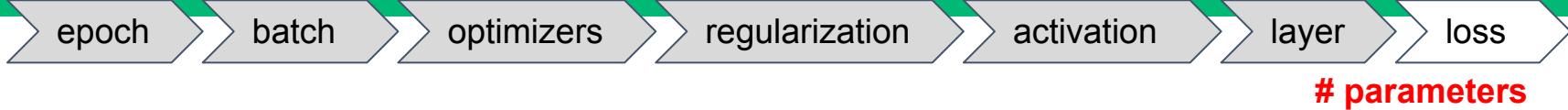
1
1
0
4
2
1
0
2
1

Pooled Feature Map

## Concat



## Layers

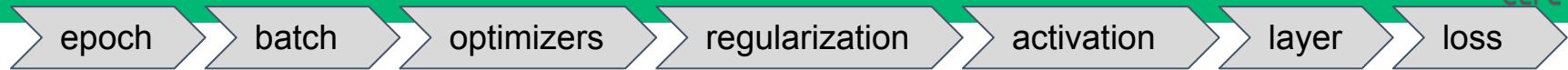


layer	parameters	# parameters
conv2d_42 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_43 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_44 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_5 (Flatten)	(None, 25088)	0
dense_19 (Dense)	(None, 4096)	102764544
dense_20 (Dense)	(None, 4096)	16781312
dense_21 (Dense)	(None, 1000)	4097000



# LOSS

# Regression loss functions: MSE



## Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

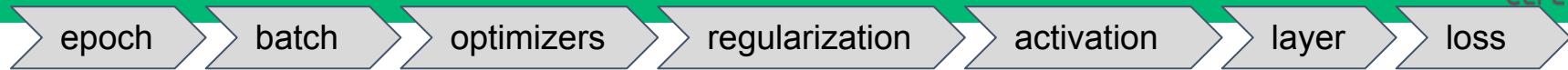
### PRO

- Convex function, suitable for gradient descent
- Doesn't care if value is below or above

### CONS

- Very sensitive to outliers

# Regression loss functions: MAE



## Mean Absolute Error (MAE)

$$MAE = \frac{1}{n} \sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}|$$

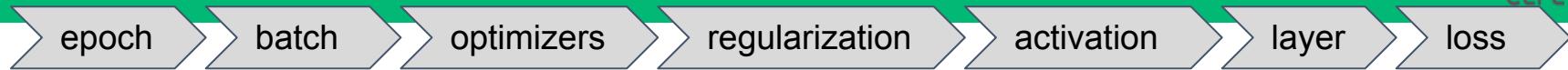
### PRO

- Less sensible to outliers. Will be used when the training data has a large number of outliers (less overfitting).

### CONS

- As the average distance approaches 0, gradient descent optimization will not work, as the function's derivative at 0 is undefined.

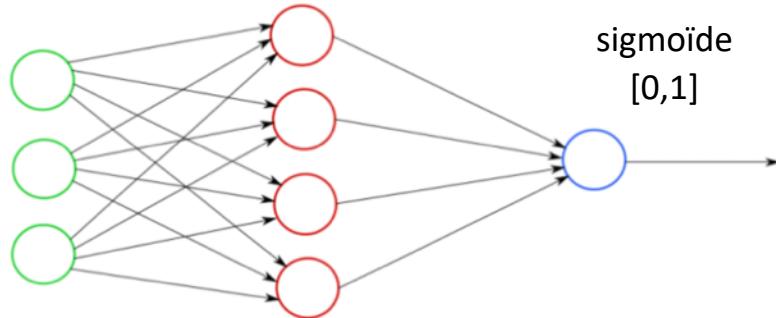
# Classification loss functions: Binary Cross-Entropy/Log Loss



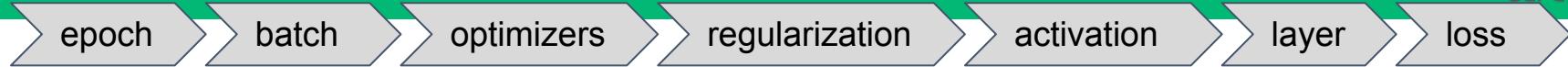
## Binary Cross-Entropy/Log Loss

$$CE\ Loss = \frac{1}{n} \sum_{i=1}^N - (y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i))$$

Used when the modelisation outputs one between 0 and 1 (sigmoïd activation): [binary classification](#)



# Classification loss functions: Cross-Entropy Loss

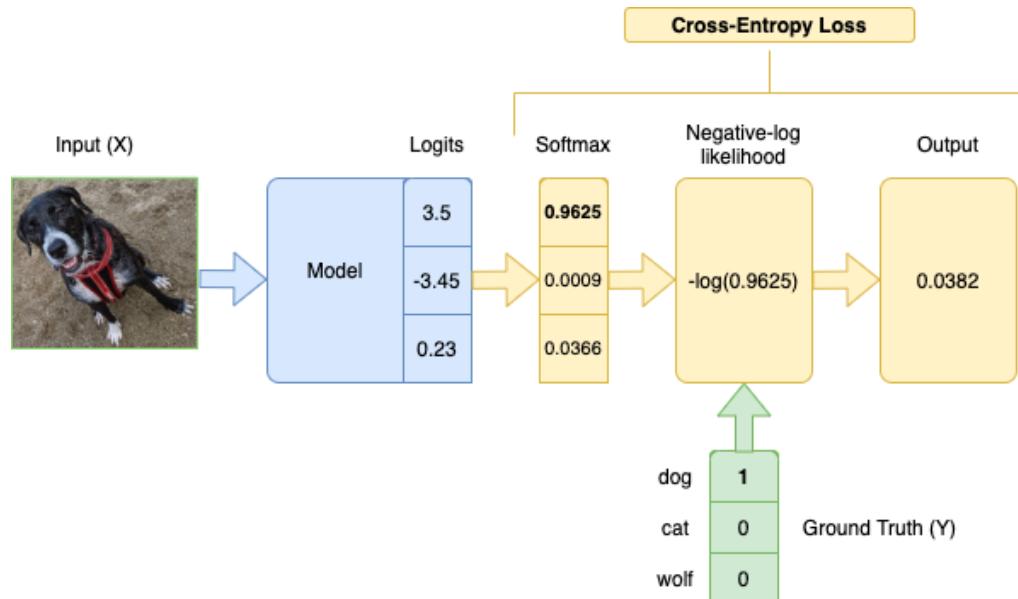


## Cross-Entropy Loss

$$CE\ Loss = -\frac{1}{n} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \cdot \log(p_{ij})$$

Used when the modelisation outputs several signals (logits or softmax):

### multi-class classification





# QUIZZ TIME On ML and DL!



CEPE

ENSAE-ENSAI  
Formation continue

# PRACTICAL SESSION: Getting familiar

## Play: get familiar

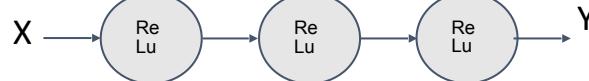
<https://playground.tensorflow.org>

# Learnings of the workshop “Get familiar”

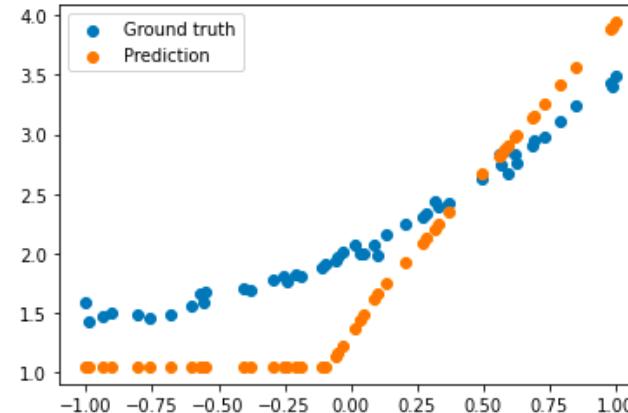
## Objective 1: get an intuition on the number of layers VS neurons

Several **layers** = composition of functions :

$$f(X) = \text{ReLU}(\text{ReLU}(\text{ReLU}(X))) = \text{ReLU}!$$



3 layers ReLU of 1 neuron each

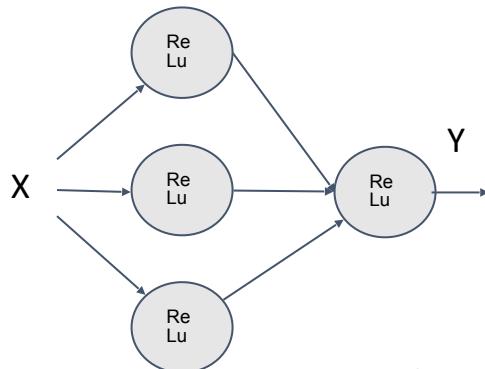


# Learnings of the workshop “Get familiar”

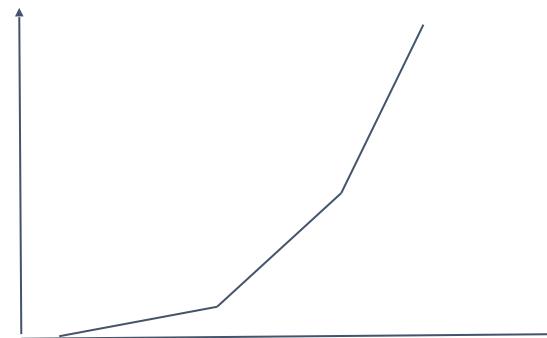
## Objective 1: get an intuition on the number of layers VS neurons

Several **neurons** = sum of functions :

$$f(X) = \text{ReLU}(X) + \text{ReLU}(X) + \text{ReLU}(X) = \text{fonction linéaire par morceaux}$$



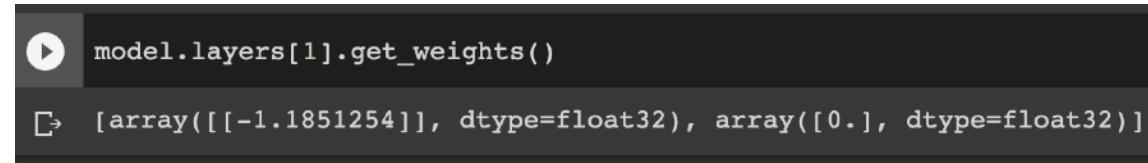
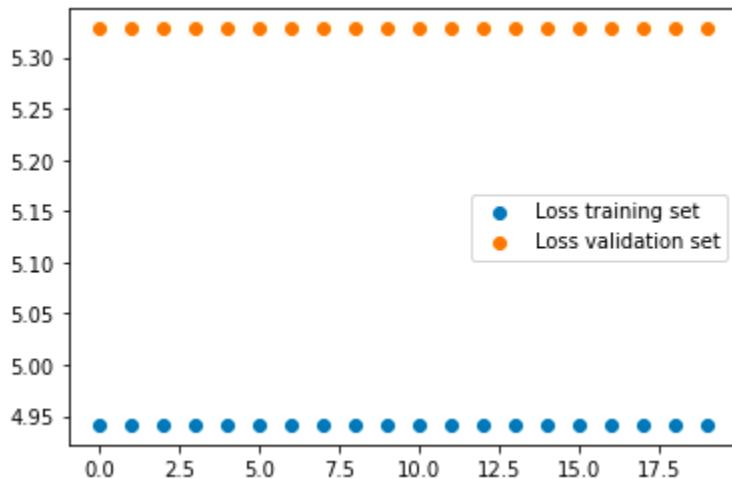
1 layer ReLu of 3 neurons



# Learnings of the workshop “Get familiar”

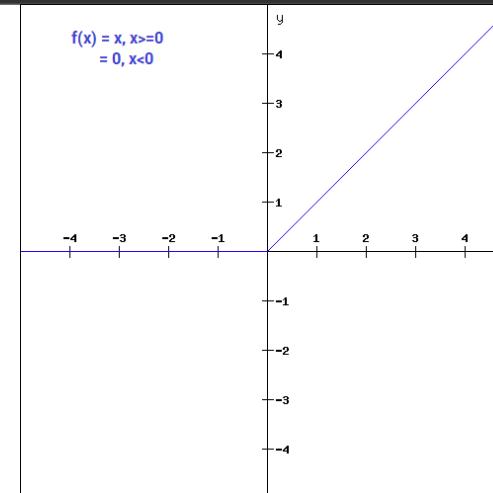
## Objective 2: get an intuition of the risks of using some activations

Some of you got dead neurons!



```
model.layers[1].get_weights()
```

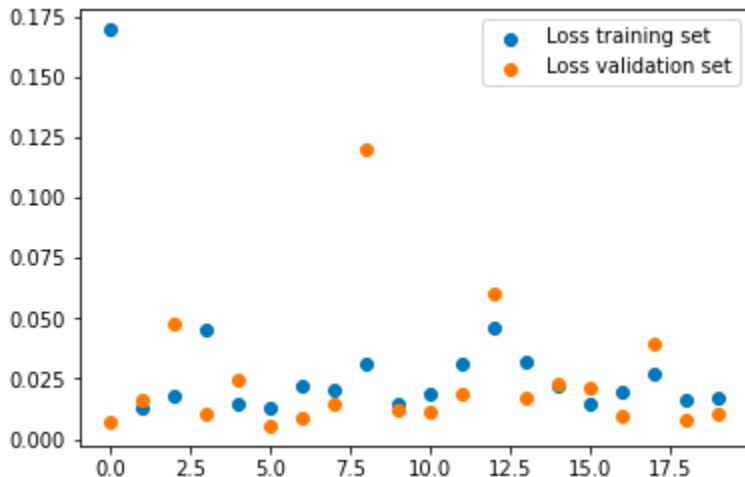
```
[array([[-1.1851254]], dtype=float32), array([0.], dtype=float32)]
```



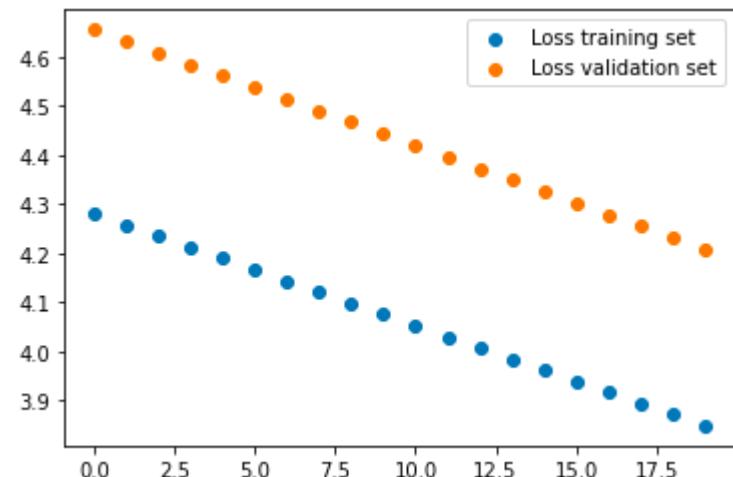
# Learnings of the workshop “Get familiar”

## Objective 3: see the impact on the learning rate

High learning rate



Low learning rate



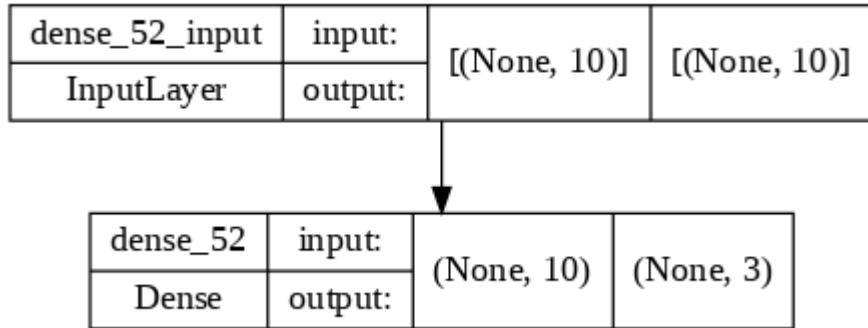
# Learnings of the workshop “Get familiar”

## Objective 4: Play with dimensions!

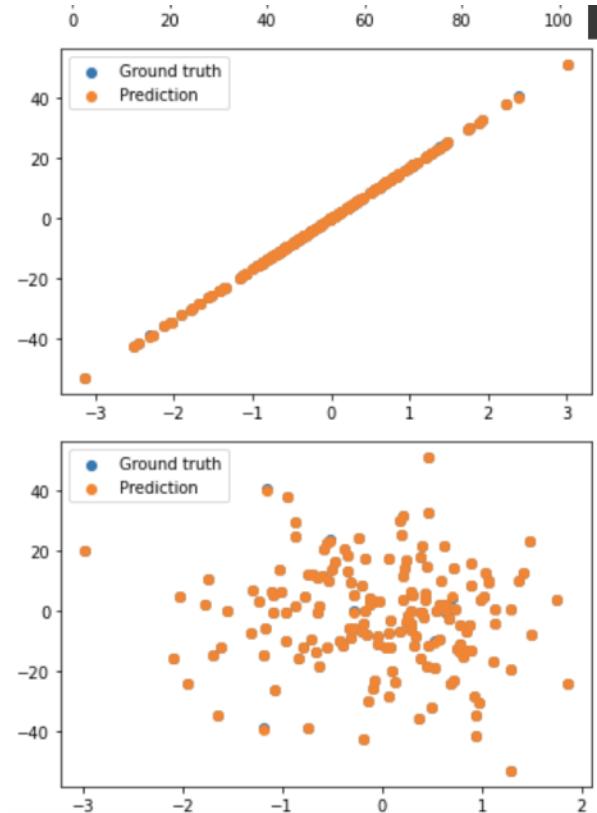


# Learnings of the workshop “Get familiar”

## Objective 5: Rasoir d'Ockham, “simpler is better”



Goal: set some weights to zero => L1-norm!  
(MAE, regularization)





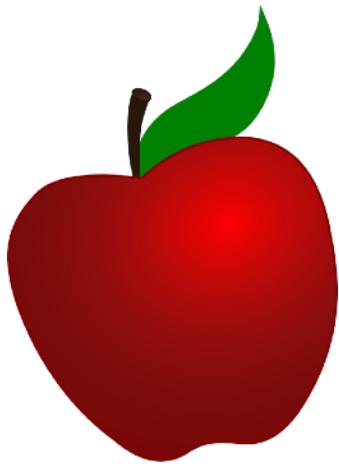
# DEEP LEARNING

## Convolutional Neural Networks

# Convolutional neural networks

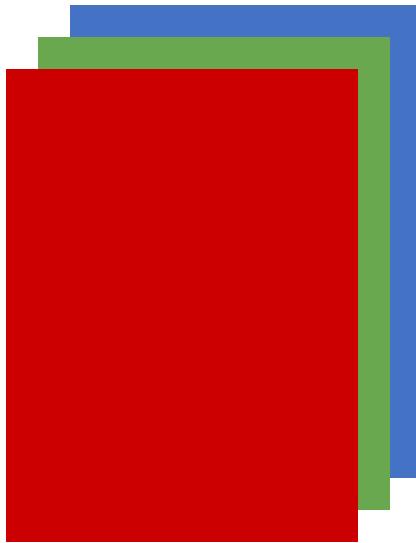
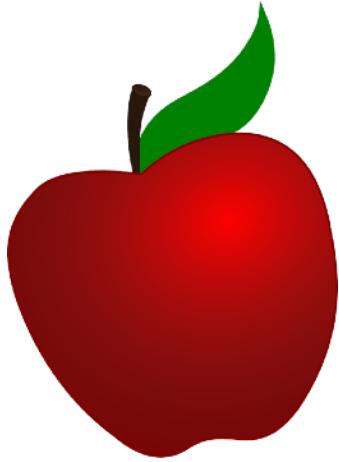
*“CNNs works under the assumption that points close to each other in the data share some correlations/relationship whist points further apart don't share as much information.”*

# Image classification

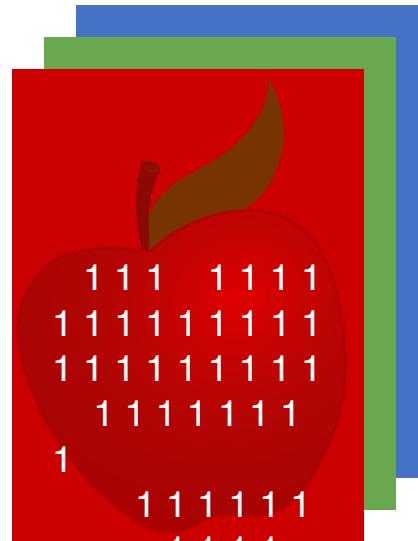
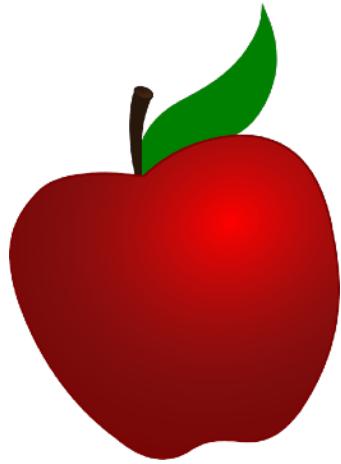


<b>apple</b>	<b>0.8</b>
pear	0.11
cherry	0.04
strawberry	0.05

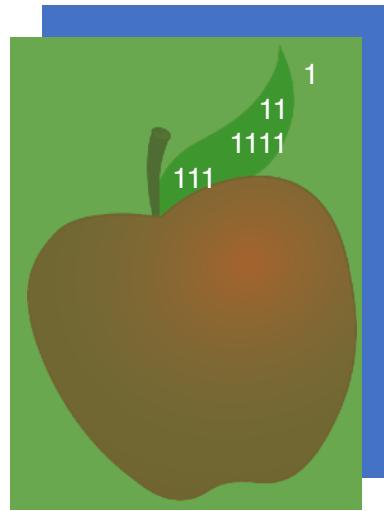
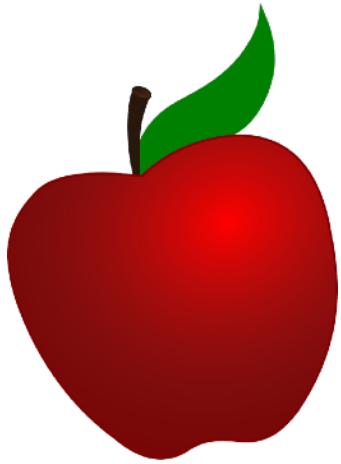
# What's an image ?



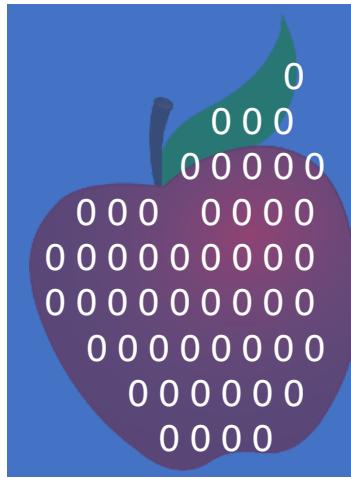
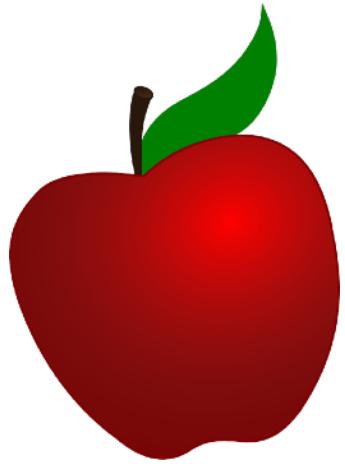
# What's an image ?



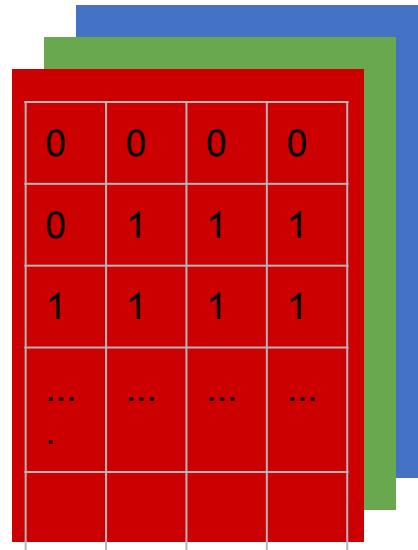
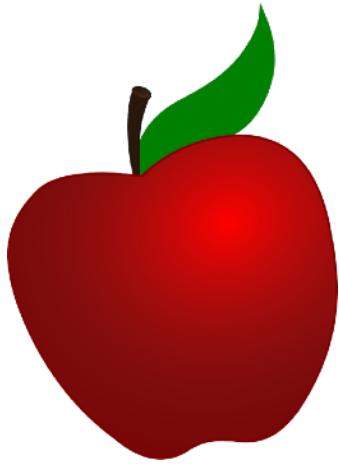
# What's an image ?



# What's an image ?



# What's an image ?



RGB = 3 matrices

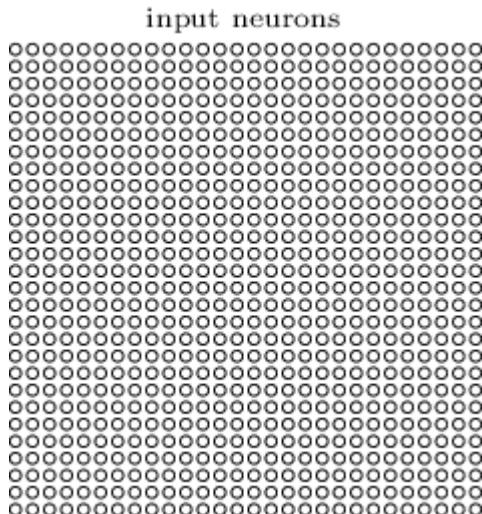
# What's an image ?



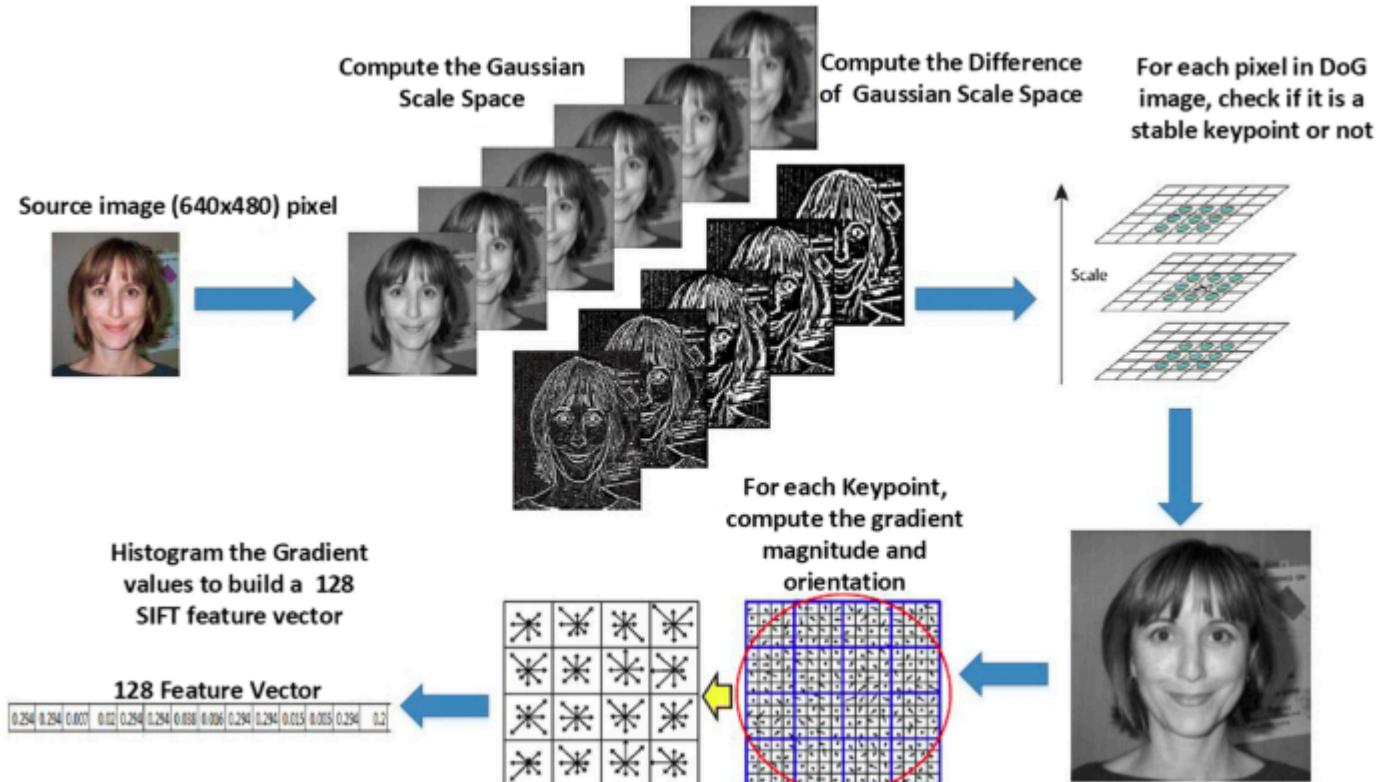
0	0	0	0
0	1	1	1
1	1	1	1
...	...	...	...
.			

Grayscale = 1  
matrix

# What's an image ?



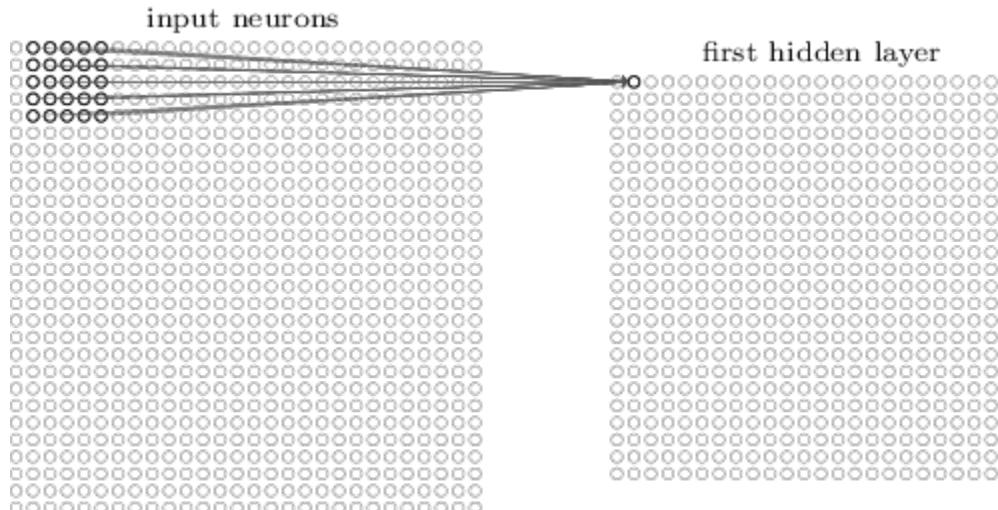
# With classic machine learning: SIFT features



# What's a convolution?

We only make connections in small, localized regions of the input image.  
=

We apply a **filter**.



# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution

$$\sigma \left( b + \sum_{l=0}^4 \sum_{m=0}^4 w_{l,m} a_{j+l, k+m} \right)$$

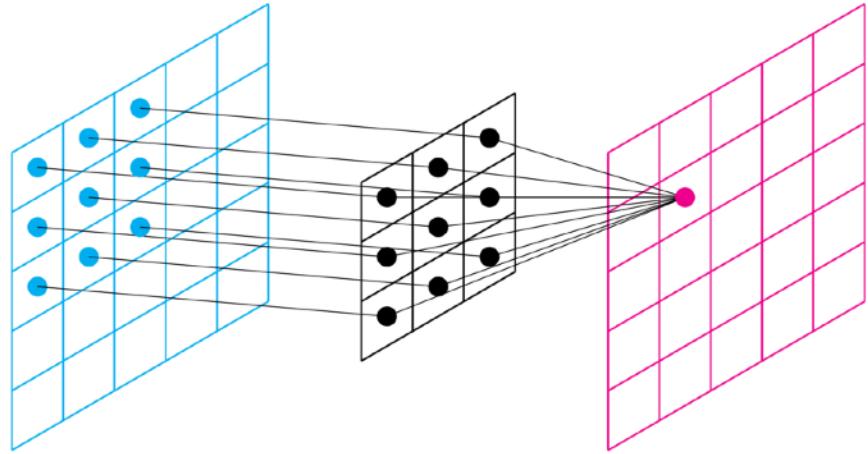


Image ou carte d'entrée

Masque convolutionnel

Carte de caractéristiques de sortie

# Filtering an image

## Choose a kernel / filter

= define a pattern to look for

1	0	-1
1	0	-1
1	0	-1

## And apply convolution

= look for this pattern  
everywhere on the image

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

input image

filter →

24 0	118	9	12	19
65	57	75	19 0	17 2
0	12 4	89	31	112
51	51	12 2	41	39

filtered image

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

1*1	0*0	-1*0
1*3	0*0	-1*2
1*10	0*2	-1*0

$$\begin{aligned}
 &= 1 + 0 - 0 \\
 &+ 3 + 0 - 2 \\
 &+ 10 + 0 - 0 \\
 &= 12
 \end{aligned}$$

And apply convolution

1	0	0	0	0
3	0	2	0	0
-10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter



			12	

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

1*3	0*0	-1*2
1*10	0*2	-1*0
1*0	0*1	-1*0

$$\begin{aligned}
 &= 3 + 0 - 2 \\
 &+ 10 + 0 - 0 \\
 &+ 0 + 0 - 0 \\
 &= 11
 \end{aligned}$$

And apply convolution

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter




12

11

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter



12				
11				
10				

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter →

12				
11				
10				
0				

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter



-2	2	0	2	0
-2	12	1	2	1
-3	11	-18	2	21
-3	10	-18	0	21
-1	0	-19	0	20

# Filtering an image

Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution

1	0	0	0	0
3	0	2	0	0
10	2	0	1	0
0	1	0	20	0
0	0	0	0	0

filter →

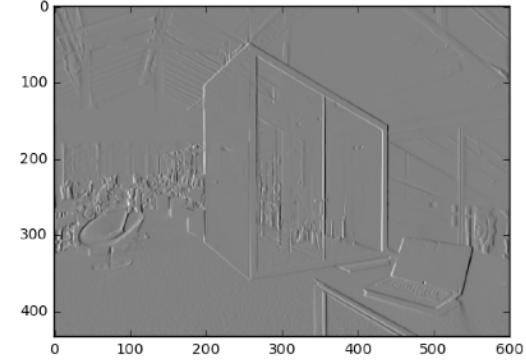
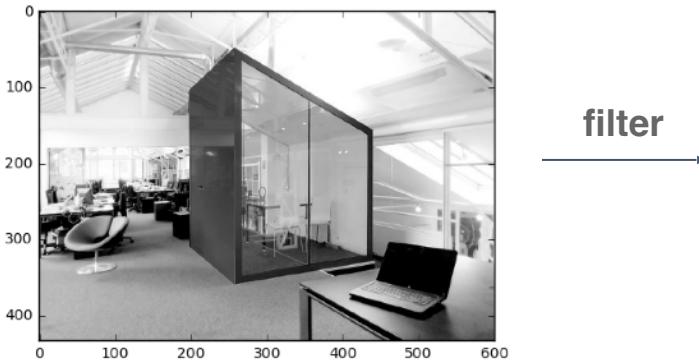
-2	2	0	2	0
-2	12	1	2	1
-3	11	-18	2	21
-3	10	-18	0	21
-1	0	-19	0	20

# Filtering an image

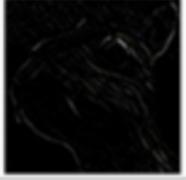
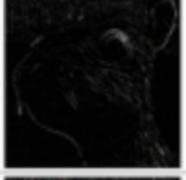
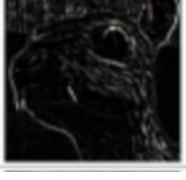
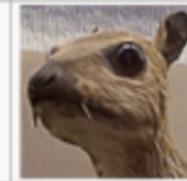
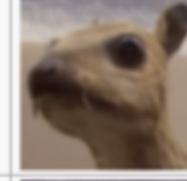
Choose a kernel / filter

1	0	-1
1	0	-1
1	0	-1

And apply convolution



Kernels are often used in photoediting software to apply blurring, edge detection, sharpening, etc.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen		
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

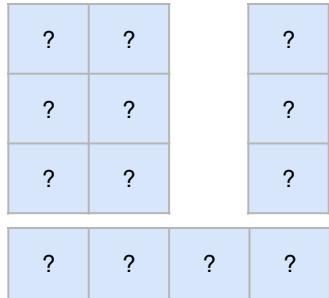
Play with <http://setosa.io/ev/image-kernels/>

# So many kernels / filters

Values can vary

?	?	?
?	?	?
?	?	?

Sizes can vary



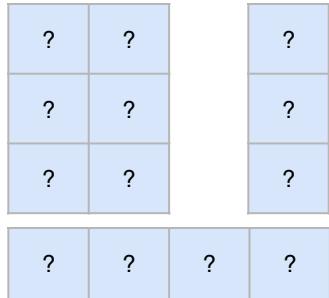
There exists an **infinite number** of ways to extract information out of an image

# So many kernels

Values can vary

?	?	?
?	?	?
?	?	?

Sizes can vary



There exists an **infinite number** of ways to extract information out of an image

# Machine Learning vs Deep Learning

## Machine Learning Approach

- Figure out **manually** which kernels make sense regarding the task we're trying to automatize. Then train the model.
- Example : Suppose we're looking to classify **buildings**, we might want to build features that highlight the presence of **vertical** edges
  - We'll thus use the previous kernel.

## Deep Learning Approach

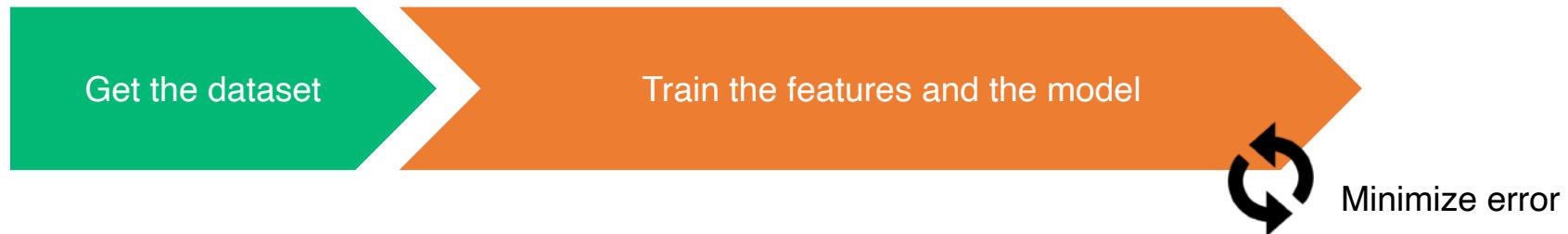
- Kernels coefficients are **parameters (weights)** of our model, and we train it to learn by itself the best kernels to use.
- You end up training **both** feature extraction and model.

# Machine Learning vs Deep Learning

## Machine Learning approach



## Deep Learning approach



# Some vocabulary

Kernel / Filter

1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	0	0
0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1	0
0 <small><math>\times 1</math></small>	0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

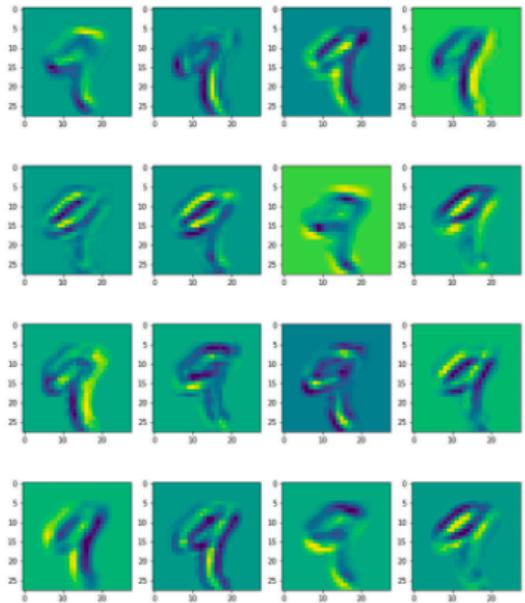
Convolved  
Feature

= feature map

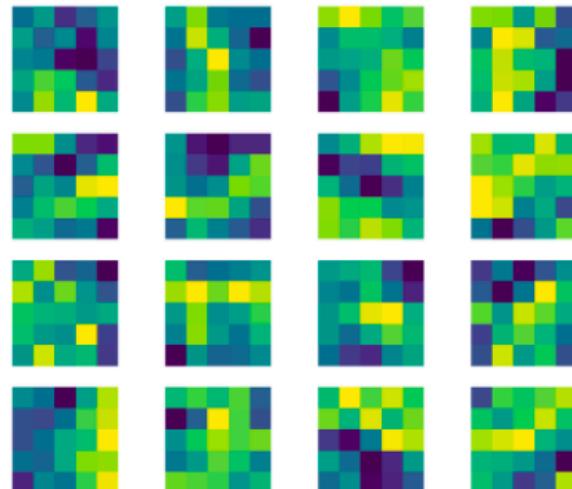


# Advantage 1: a bit of interpretability

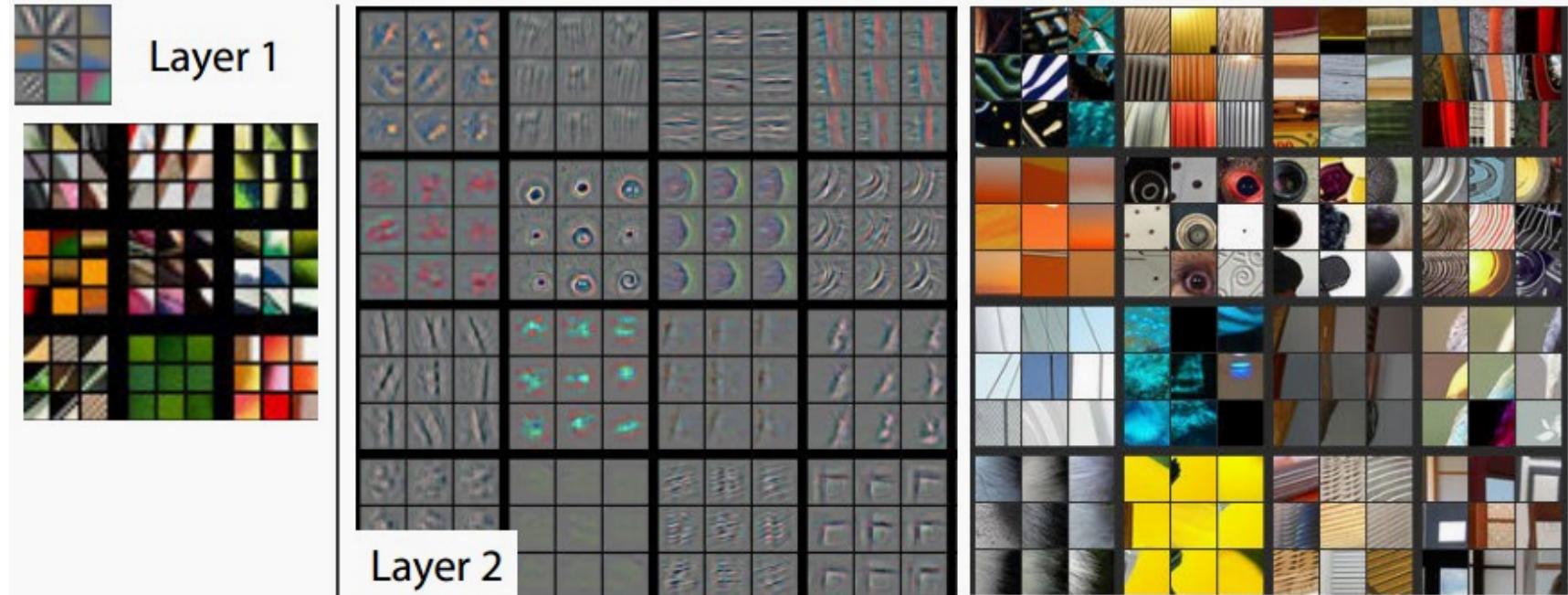
Feature map



Filters

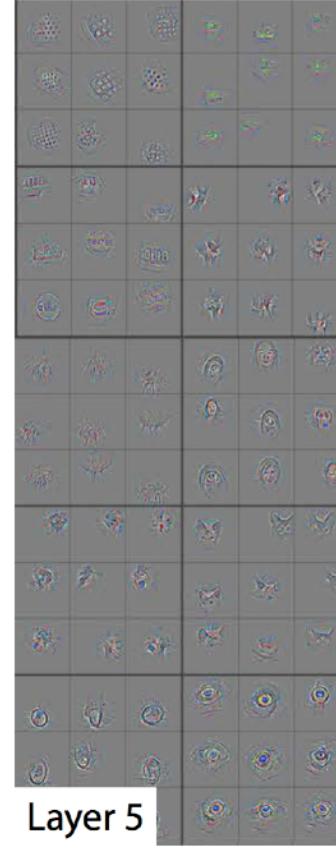
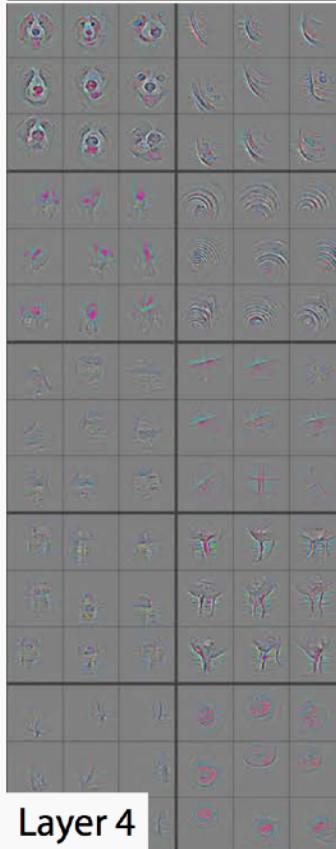


# Advantage 1: a bit of interpretability

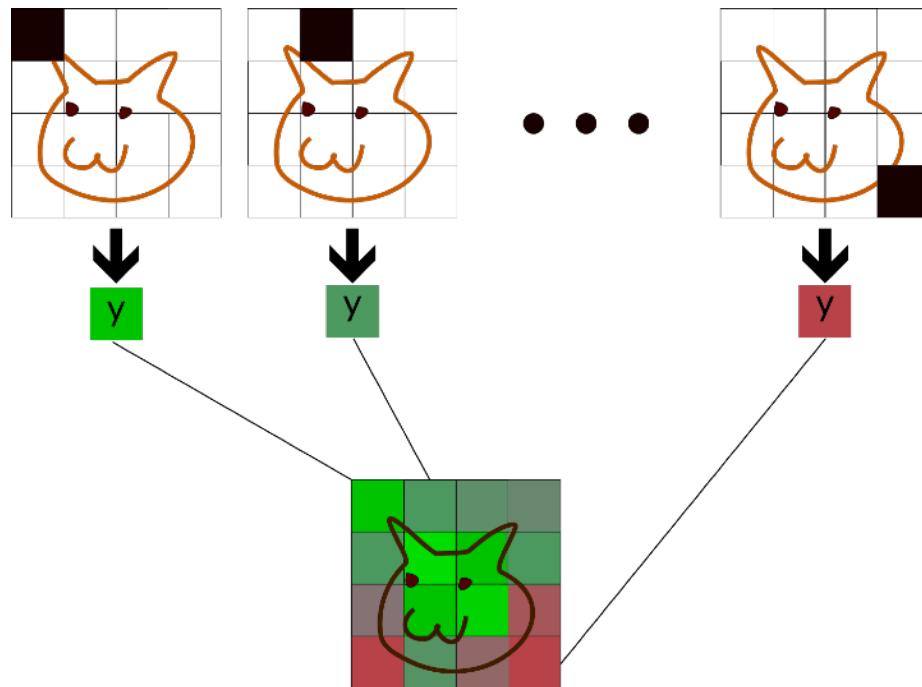


[Visualizing and Understanding Convolutional Networks](#), Matthew D. Zeiler and Rob Fergus  
(NYU)

# Advantage 1: a bit of interpretability

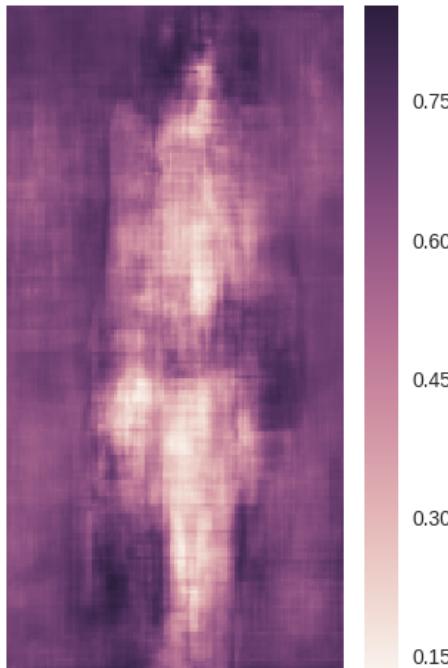
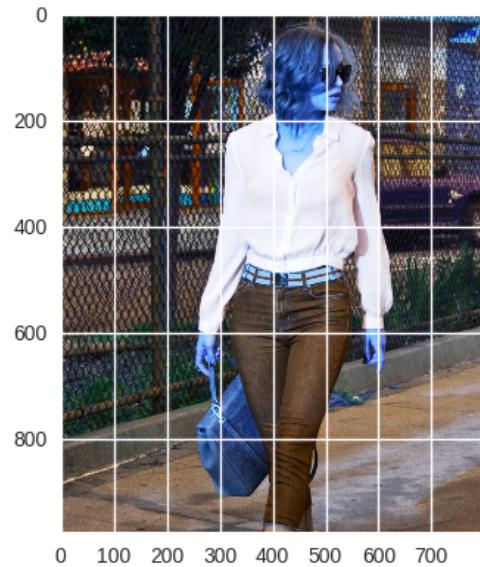


# Advantage 1: a bit of interpretability (occlusion)



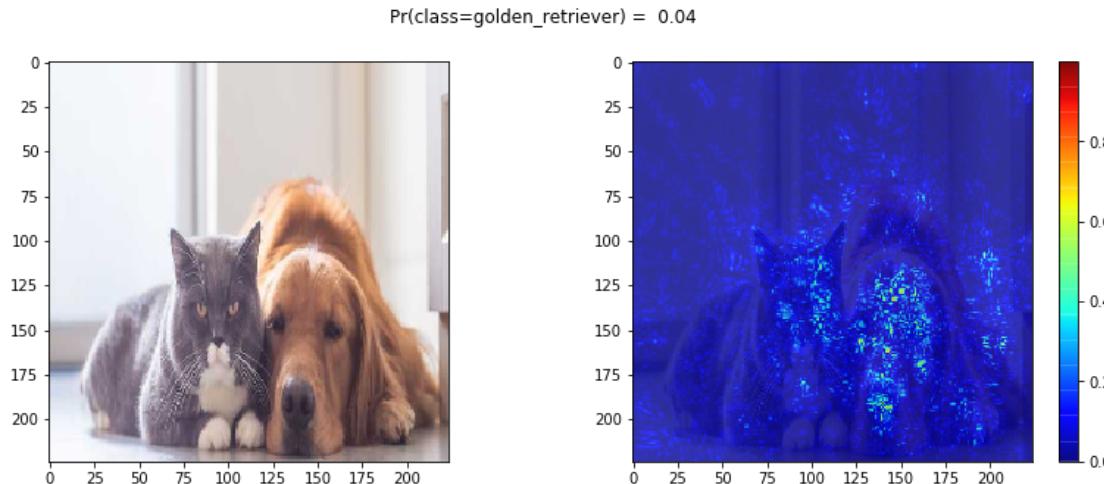
Occlusion analysis calculates the importance of every patch by observing the change of model output  $y$  when the patch is removed. The individual results can be assembled to an attribution map. Image by author

# Advantage 1: a bit of interpretability (occlusion)



# Advantage 1: a bit of interpretability (saliency maps)

Saliency shows the magnitude of the derivative indicating which pixels need to be changed the least to affect the class score the most. One can expect that such pixels correspond to the object location in the image.





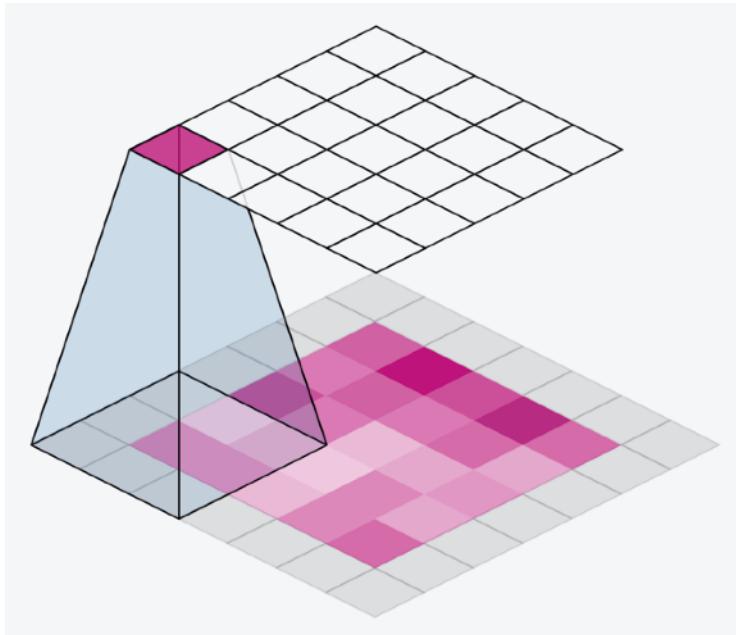
## Advantage 2: much less parameters than a fully-connected NN

For an image of size 28X28:

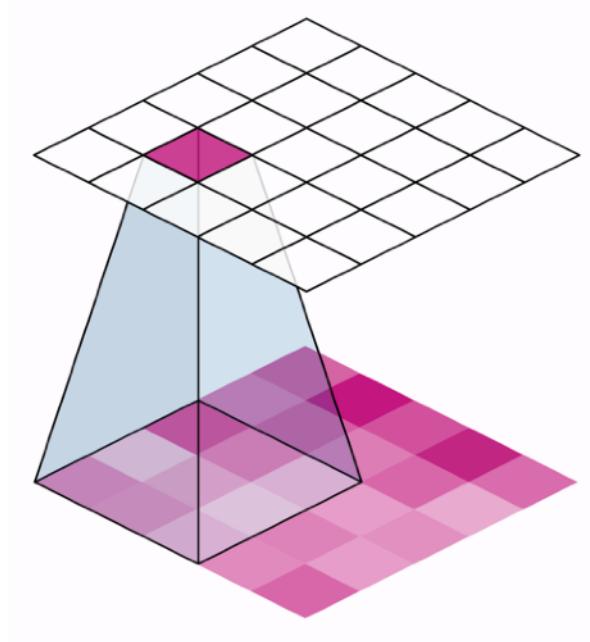
- Feedforward NN with 30 neurons:
  - Flatten:  $784 = 28 \times 28$  input neurons
  - $784 \times 30$  weights + 30 bias
  - Total of **23,550 parameters**
- CNN with 20 feature maps (ie, 20 different filters of size 5x5):
  - For each feature map,  $25 = 5 \times 5$  shared weights + 1 bias
  - **520 = 20x26 parameters**

# To go further with convolution...

Very [nice paper](#) from MILA that describes all cases of strides, paddings etc. <https://arxiv.org/pdf/1603.07285.pdf>



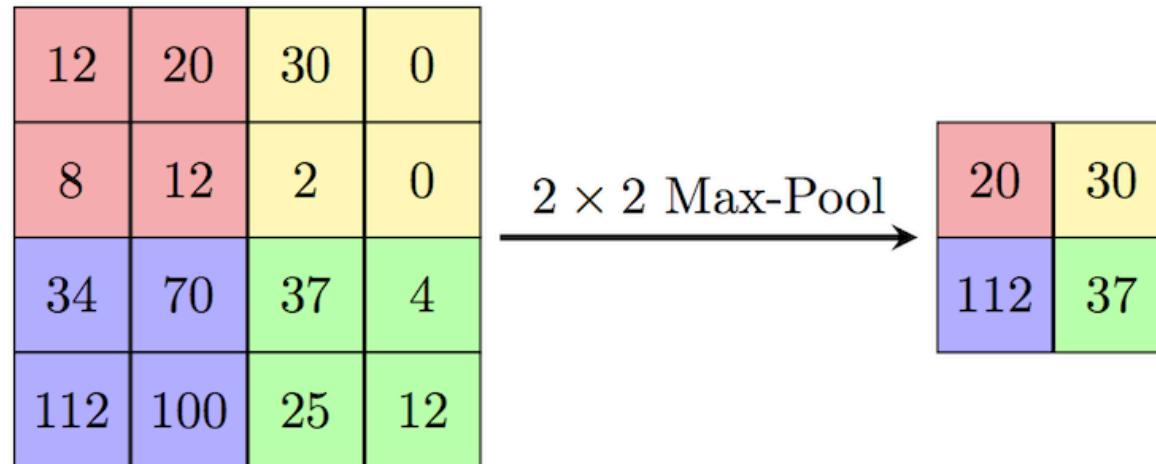
Padding



Stride

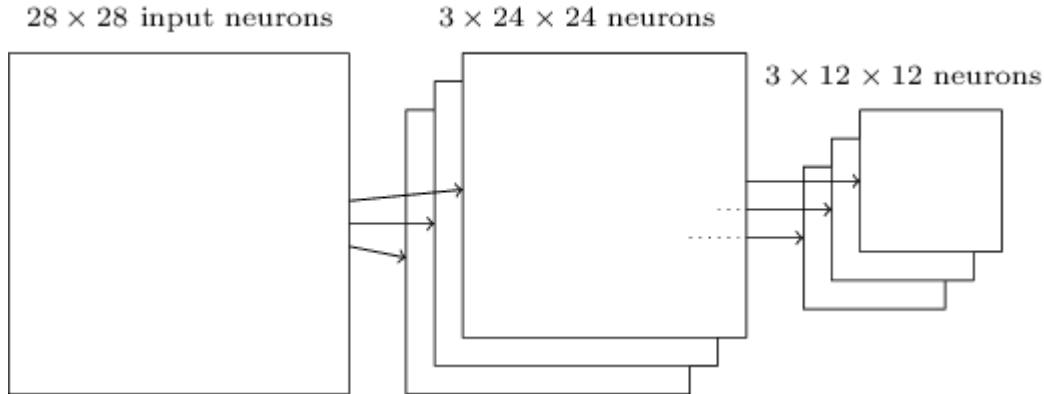
# Pooling / subsampling

Combined with a system called pooling (ex: max-pooling), the non-highlighted elements are discarded from each feature map, leaving only the features of interest, **reducing** the number of learned parameters, and **decreasing** the computational cost (e.g. system memory).



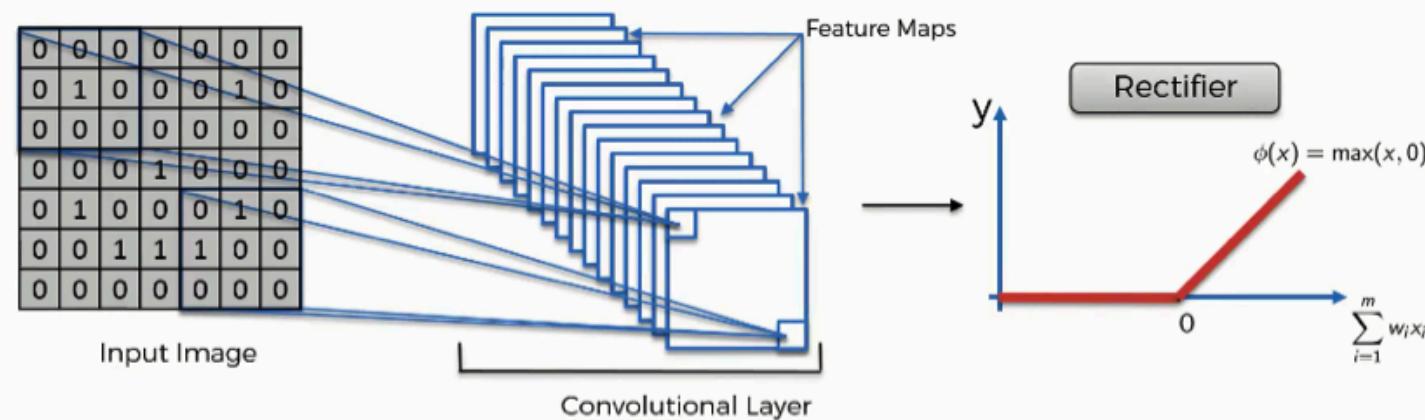
Max pooling, min pooling, average pooling ...

# Pooling / subsampling

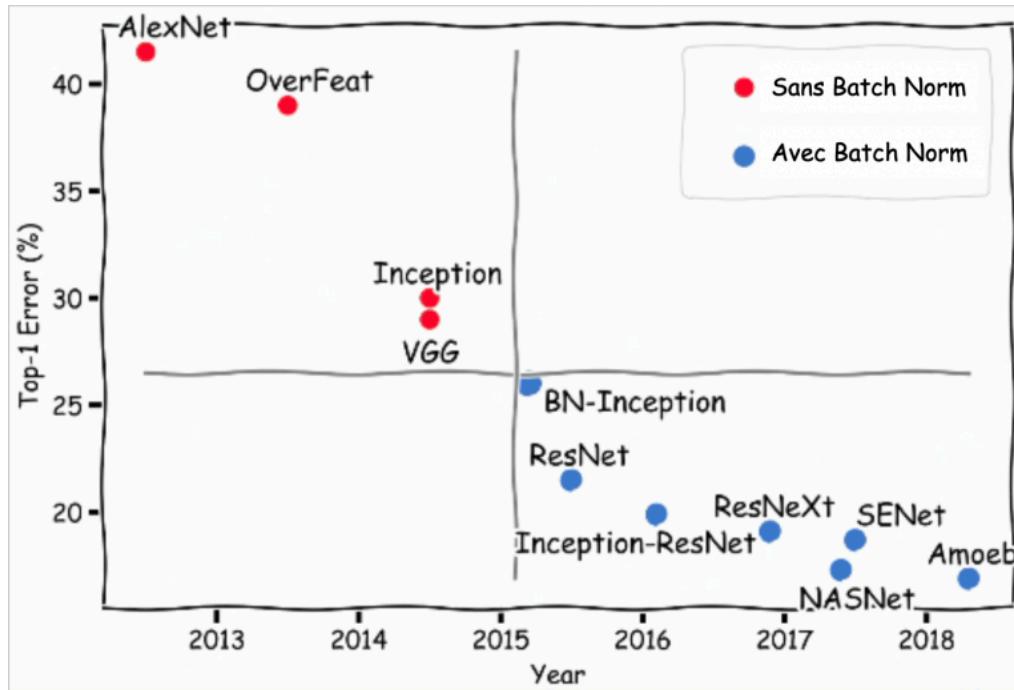


# Activation - ReLu

*“What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors”*



# Batch normalization



[Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift](#)

# Batch normalization - What is it?

A layer inserted right before (or after) a non-linear layer, in order to normalize the output of the layer, as you do some normalization directly on the input.

The normalization (mean/standard deviation) is computed on **one entire batch at each iteration** during **training**.

$$BN(x_i) = \gamma \left( \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \beta$$

# Batch normalization - What is it?

For **inference**, we use the **estimated mean** and **estimated standard-deviation** as the mean of all values met during training (=on all batches!)

# Batch normalization - Why?

## Why batch normalization ?

- enables the use of higher learning rates
- acts as a regularizer
- can speed up training a lot!



# Batch normalization - When?

## C.2.6) Before or after the nonlinearity ?

Historically, BN layer is positioned **right before** the nonlinear function, which was consistent with the authors objectives and hypothesis at that time :

In their article, they stated :

*“We would like to ensure that, for any parameter values, the network always produces activations with the desired distribution.”*

*– Sergey Ioffe & Christian Szegedy (source : [1])*

# Batch normalization - When?

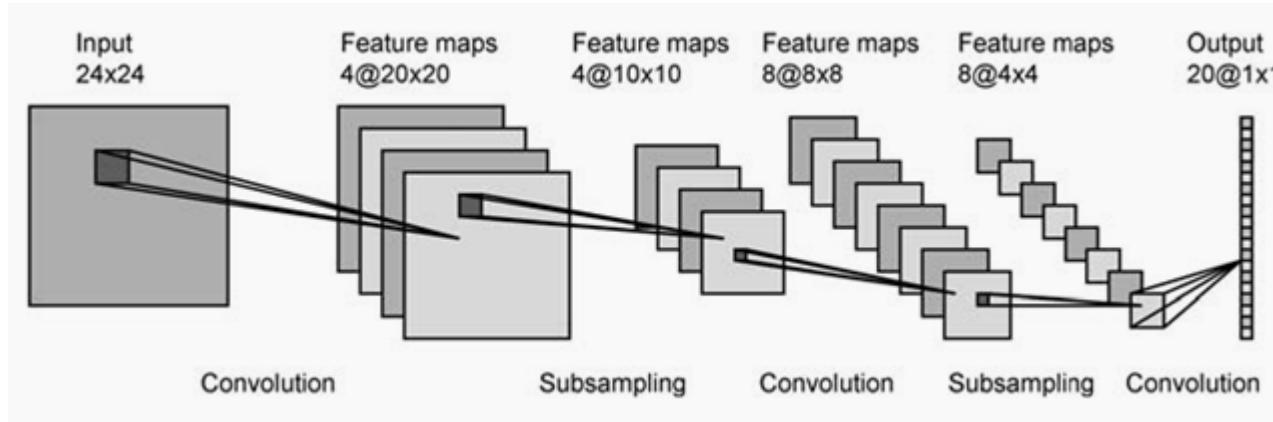
Some experiments showed that positioning BN layers **right after** the nonlinear function leads to **better results**. Here is an [example](#).

François Chollet, creator of Keras and currently engineer at Google, stated that :

*“I haven’t gone back to check what they are suggesting in their original paper, but I can guarantee that recent code written by Christian [Szegedy] applies relu before BN. It is still occasionally a topic of debate, though.”*

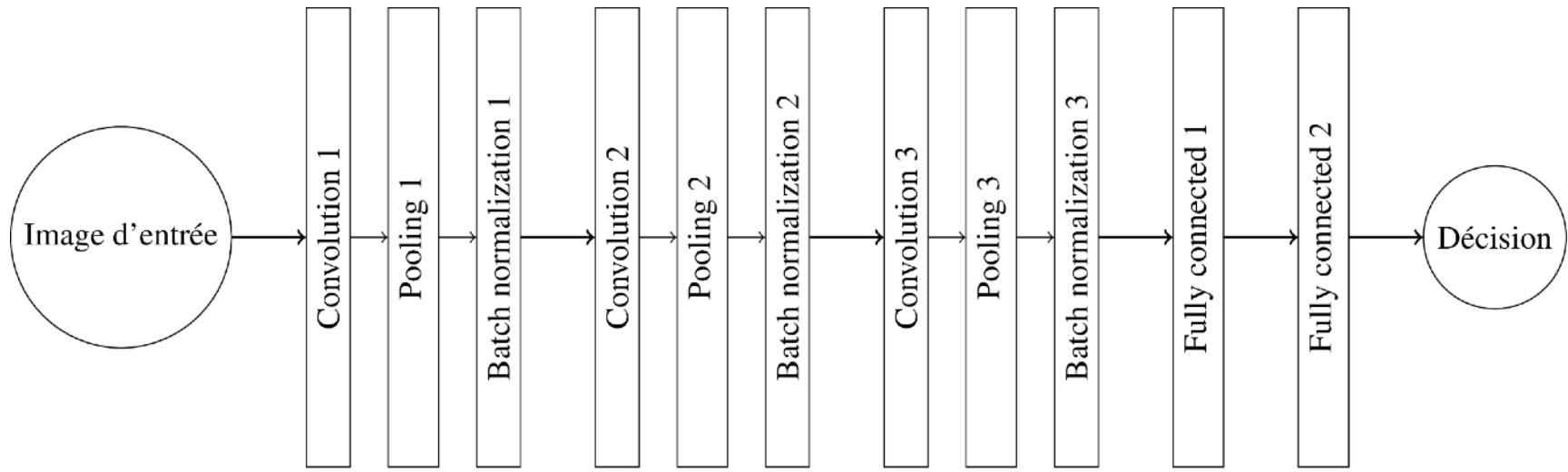
— François Chollet ([source](#))

# A complex CNN architecture for image classification

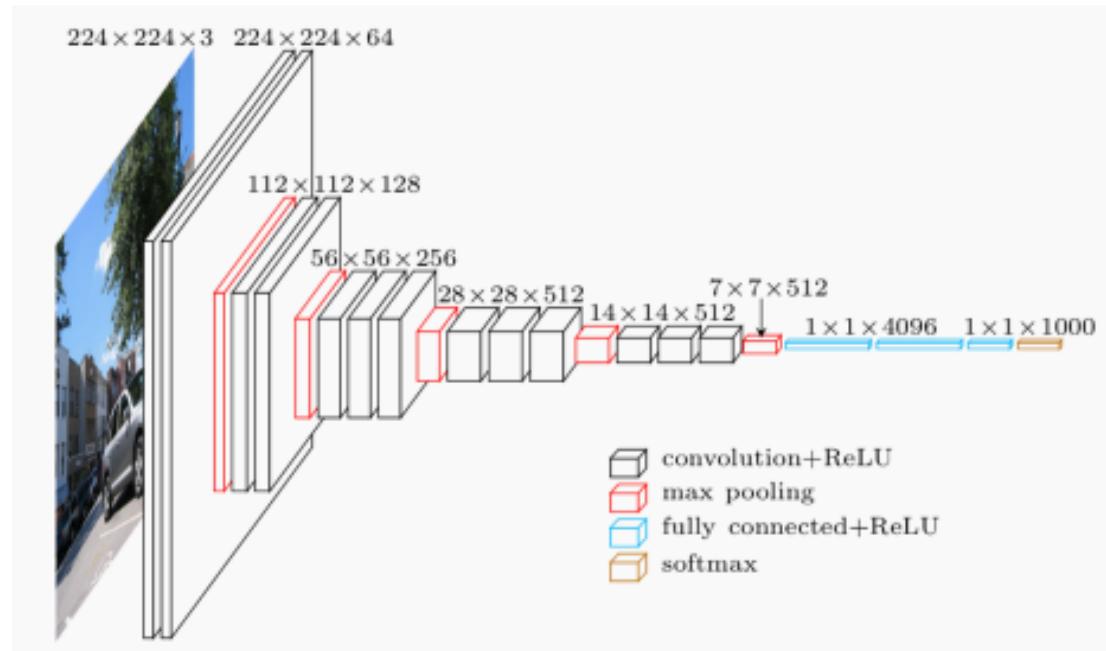


Convolution of convolution of convolution...

# A complex CNN architecture for image classification



# Machine vs. deep learning: features extraction



Take your data, train  
the feature extraction  
process + the model  
and optimize your  
error.



# What remains to be done ?

It looks all automatic, but it's not. A lot still has to be chosen :

- Network architecture
  - Number of layers
  - Kernel size
  - Stride
  - Max-pooling ? Dropout ?
  - Number of kernels used per layer
  - Optimization parameters (which optimizer ? which learning rate ? ...)
  - Which error measure ?
  - Which activation functions and models ?
  - ...

# *ImageNet* competition

- Held every year
- Benchmark for image classification
- Competition : 1 000 classes
- Full dataset : over 14 M images of 5 000 classes
  - Animals
    - Golden
    - Labrador
  - Cars
  - Boats
  - Trees
  - ...

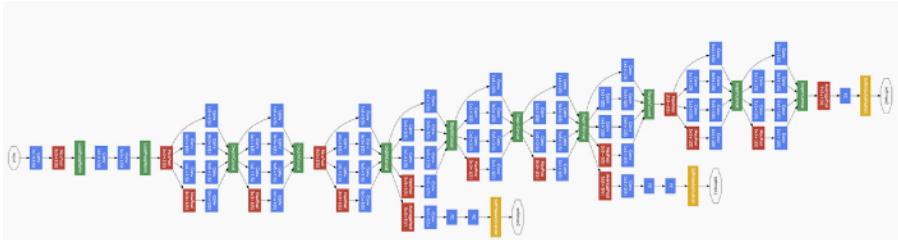
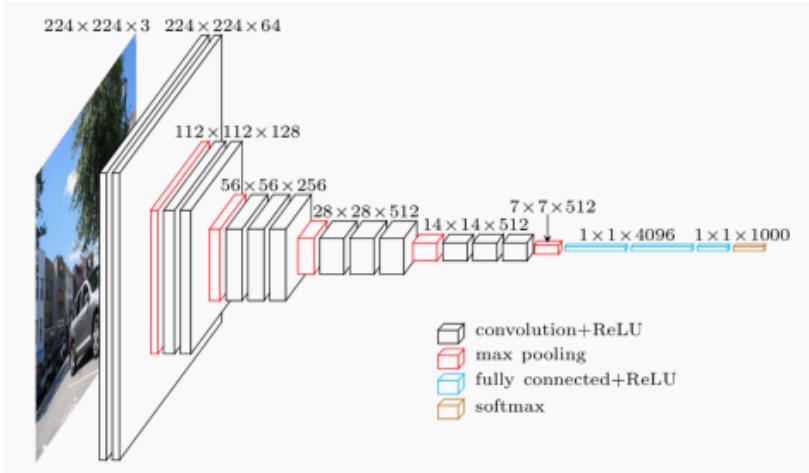


[Leaderboard](#)

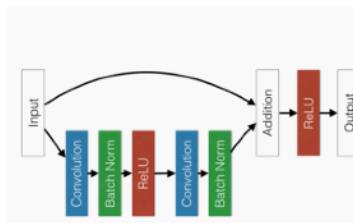
# ImageNet winners

## Inception, Google's CNN (2015)

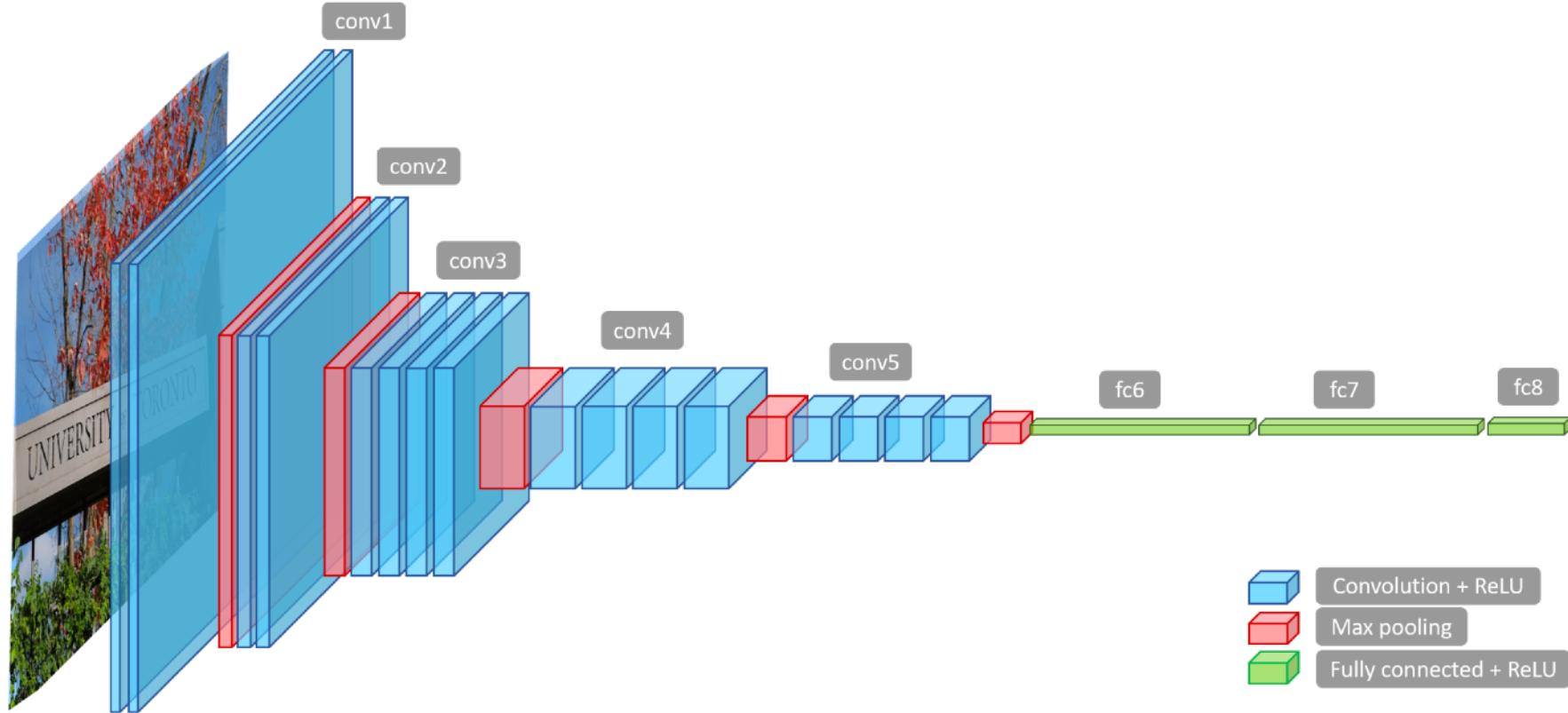
### VGG - winner in 2014



## ResNet main block (+1000 layers) (2018)



# VGG19



## Other applications of CNNs?

*“CNNs works under the assumption that points close to each other in the data share some correlations/relationship whist points further apart don't share as much information.”*

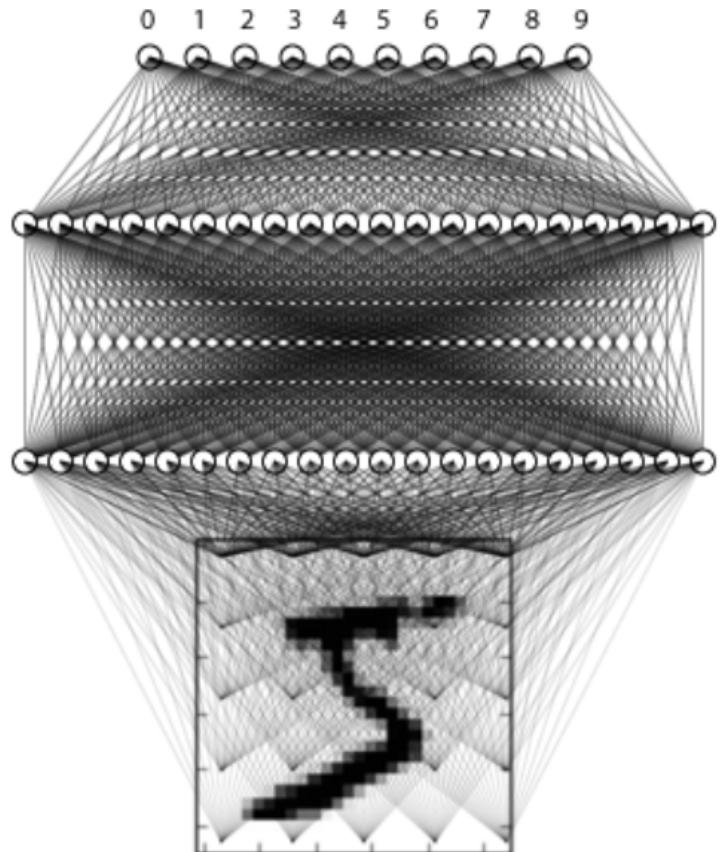
- Speech recognition
- [Character-level Convolutional Networks for Text Classification, LeCun 2016](#)
- Music classification
- ...



# PRACTICAL SESSION: Image classification (MNIST)

# MNIST Classification

000000000000000  
111111111111111  
222222222222222  
333333333333333  
444444444444444  
555555555555555  
666666666666666  
777777777777777  
888888888888888  
999999999999999





# DEEP LEARNING

## Transfer learning

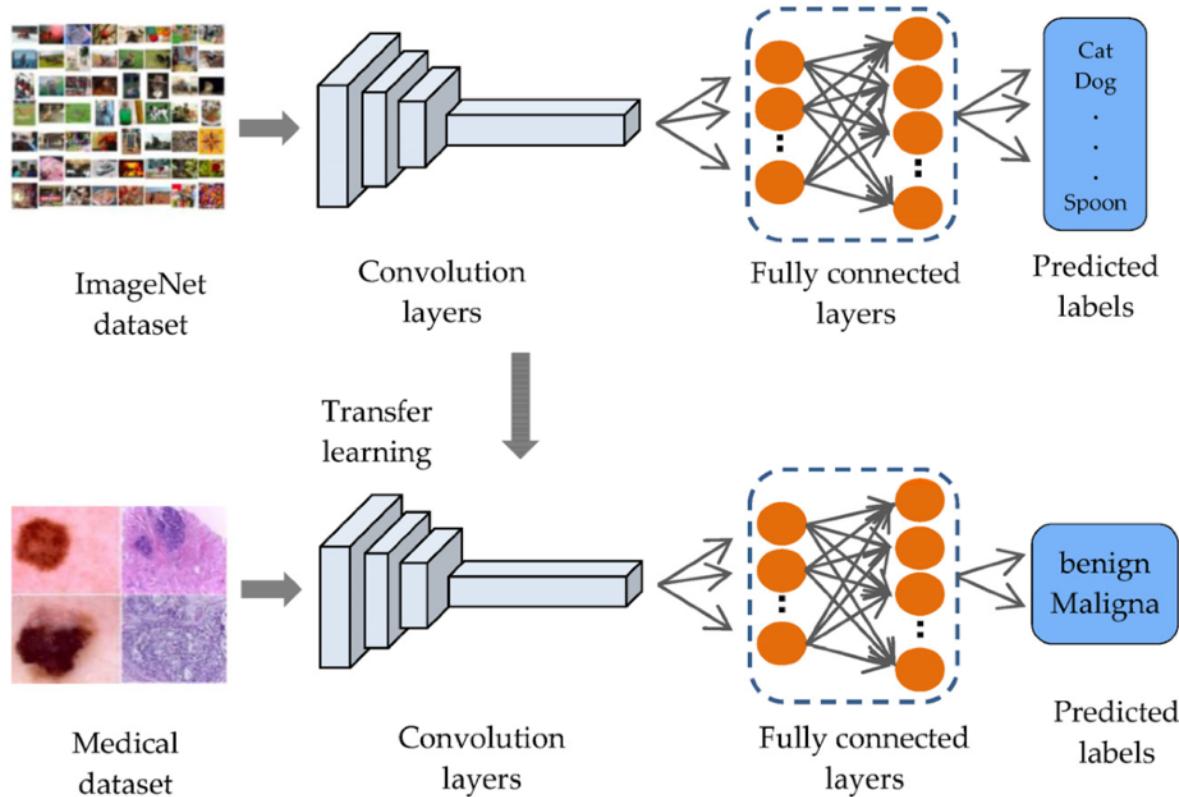
If you can move  
mountains you can  
move molehills

Every year, models architectures and weights are **released**, making it possible to use them and only adapt them to our personal use cases.

The *ImageNet* challenge is very complicated, the winner models are very complex and strong. They can generalize very well to our (probably simpler) tasks.

That's called **transfer learning / fine tuning**.

# What is transfer learning?



# Transfer learning



Train on ImageNet full dataset. Takes days to weeks.

Usually weights of best models are open-source



Have a similar task, small dataset ?  
(image classification)

Use the extracted features. Only train again the

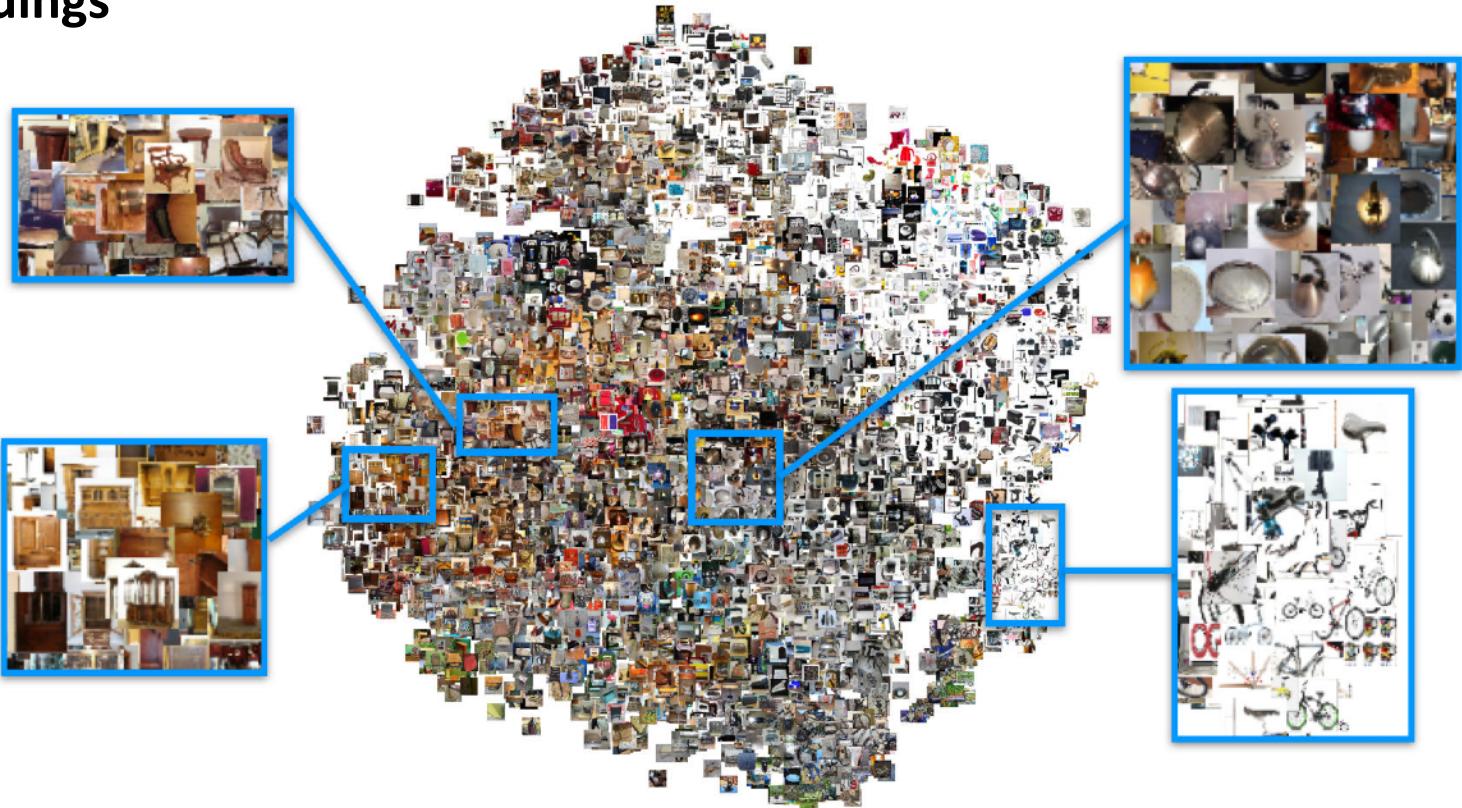


Medium sized dataset ?

*Finetune* : use old weights as initialization & train some of the

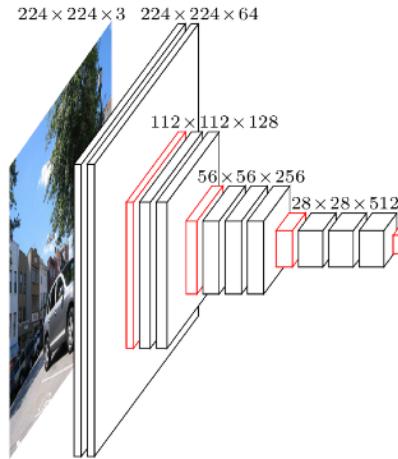
# Transfer learning for image processing

## Image embeddings



# Transfer learning for image processing

My images ->



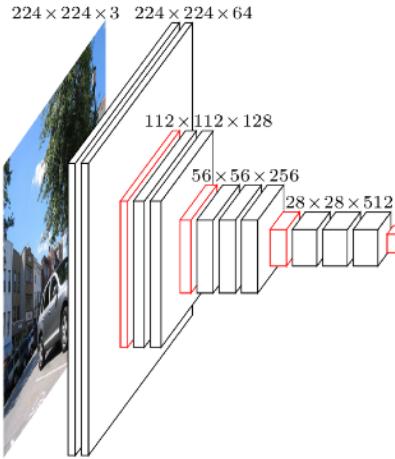
**Frozen** part of  
VGG-16, I don't learn  
anything. Just use the  
weights they found.

-> Learn a new  
model.

NN, SVM, ...

# Transfer learning for image processing

My images ->



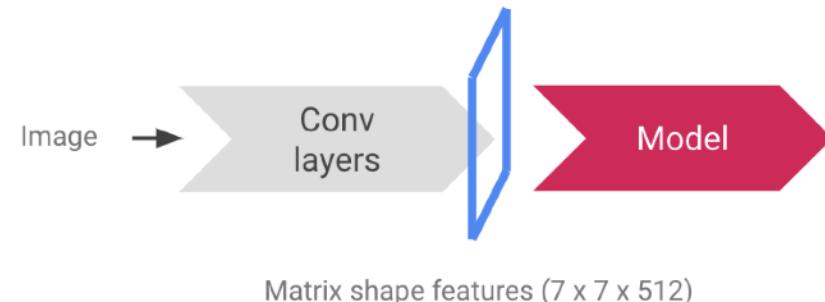
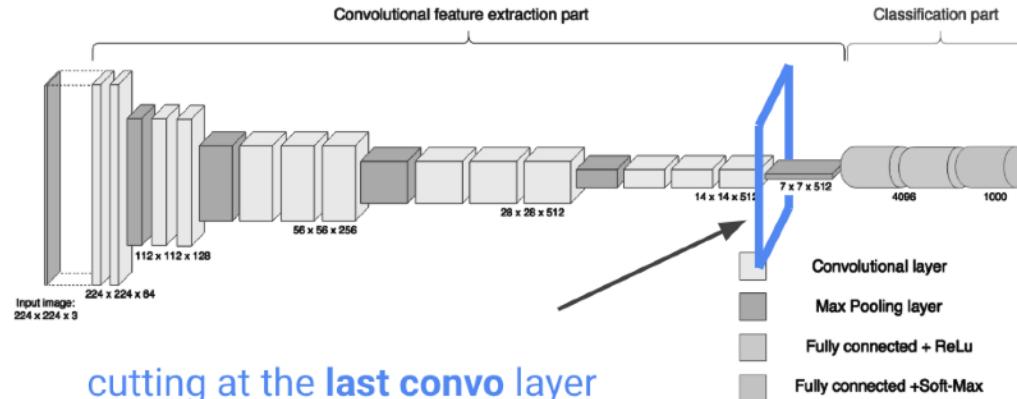
**Frozen** part of  
VGG-16, I don't learn  
anything. Just use the  
weights they found.

## Feature extractor !

-> Learn a new  
model.

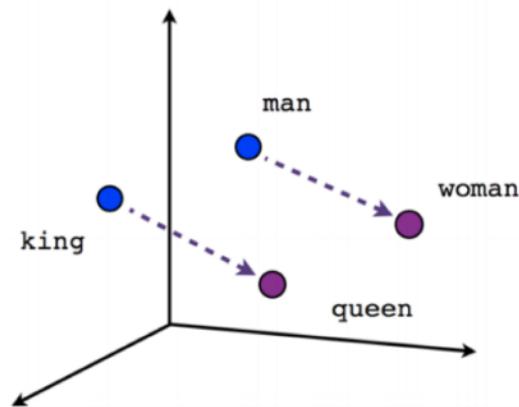
NN, SVM, ...

# Transfer learning for image processing

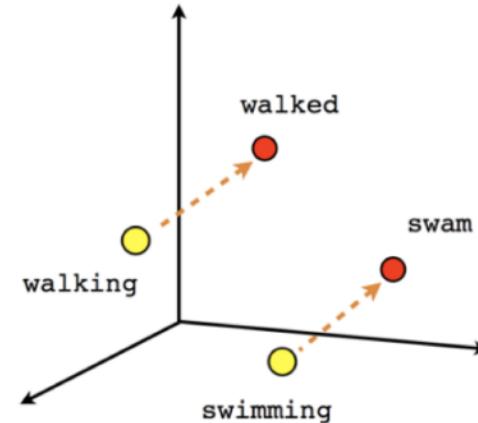


# Transfer learning for natural language processing

## Word2vec



Male-Female



Verb tense

# Transfer learning for natural language processing

## Word2vec : word embeddings

{

“cat” : [.4, .52, .346, .11, .... 0.34]

“eat” : [.1, .36, .13, .65, .... .13]

...

}

# Transfer learning for natural language processing

## Word2vec : word embeddings

{

“cat” : [.4, .52, .346, .11, .... 0.34]

“eat” : [.1, .36, .13, .65, .... .13]

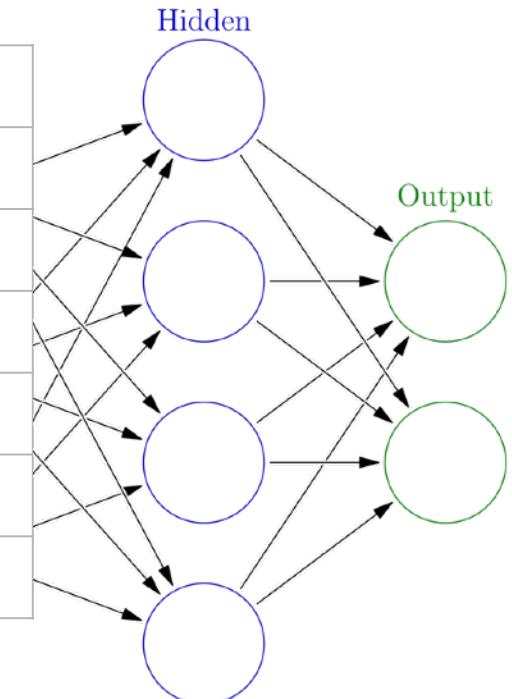
...

}

The number of existing words is **smaller** than that of images... Weights are already online for your words ! No need to run the model again.

# Transfer learning for natural language processing

<b>word</b>	<b>word embedding</b>
It	[ 0.91547946, 0.7544022 , 0.67863324, 0.43281506, 0.73449711]
has	[ 0.46900183, 0.45051458, 0.65645181, 0.5646351 , 0.46867353]
been	[ 0.53305252, 0.20943627, 0.48401917, 0.78969676, 0.37212058]
a	[ 0.44045638, 0.62744136, 0.24633633, 0.27124499, 0.54295044]
wonderful	[ 0.51238418, 0.83086943, 0.24552175, 0.240159 , 0.50736234]
course	[ 0.64031771, 0.84796496, 0.21361924, 0.23347336, 0.43963726]



Or any model !



# PRACTICAL SESSION: Transfer learning



# QUIZZ TIME On CNNs!



CEPE

ENSAE-ENSAI  
Formation continue

# DEEP LEARNING

## Recurrent neural networks

## Recurrent neural networks

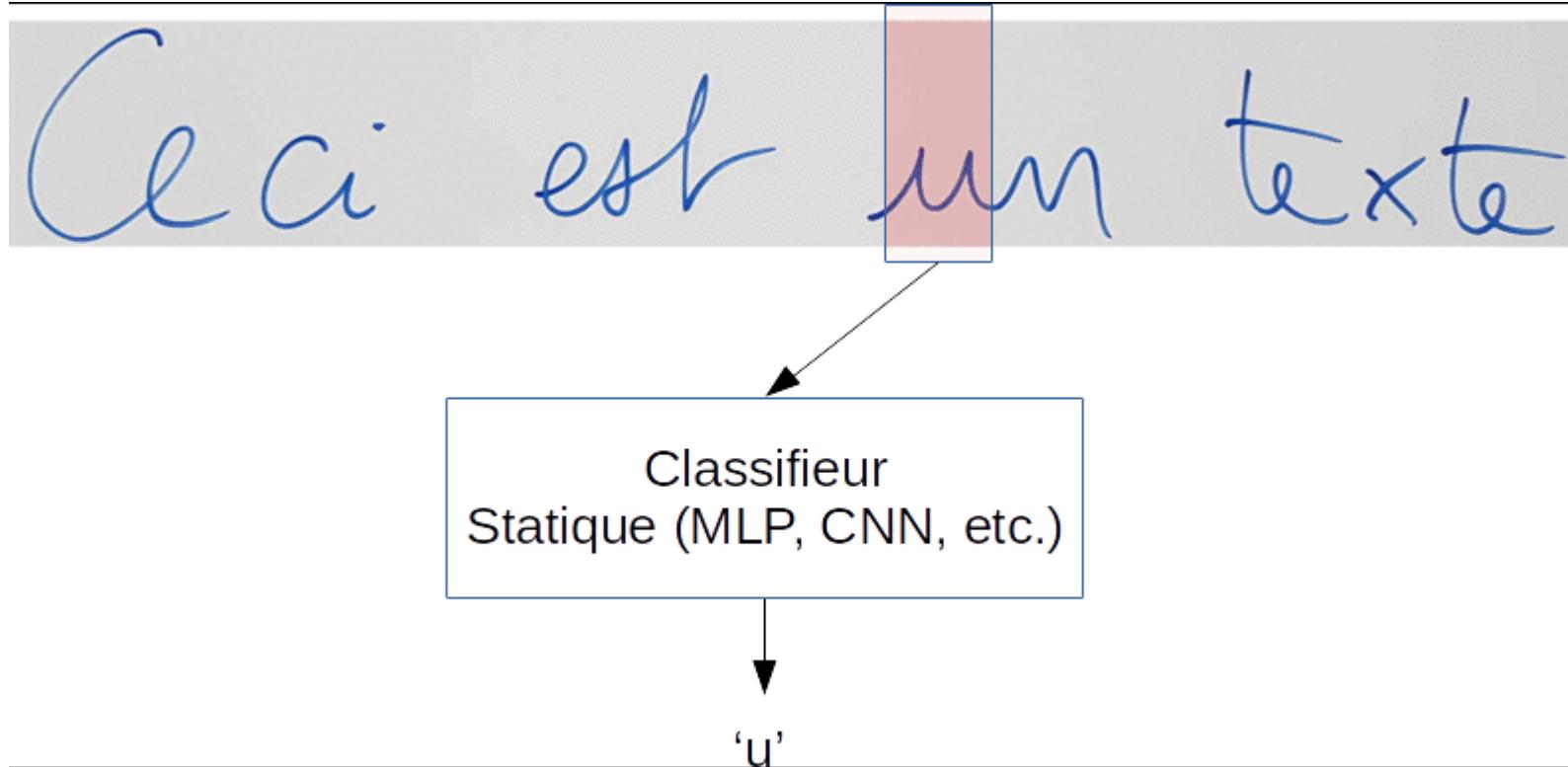
Unfortunately, feedforward neural networks have a very **strict constraint**: the inputs and outputs should all be the **same size**.

What about:

- Texts or speech that are not all the same size,
- Automatic translation: the input and output sentences do not have the same size
- ...

## How can we deal with sequences?

# CNN or MLP for sequences...



... But sometimes, we need the context!

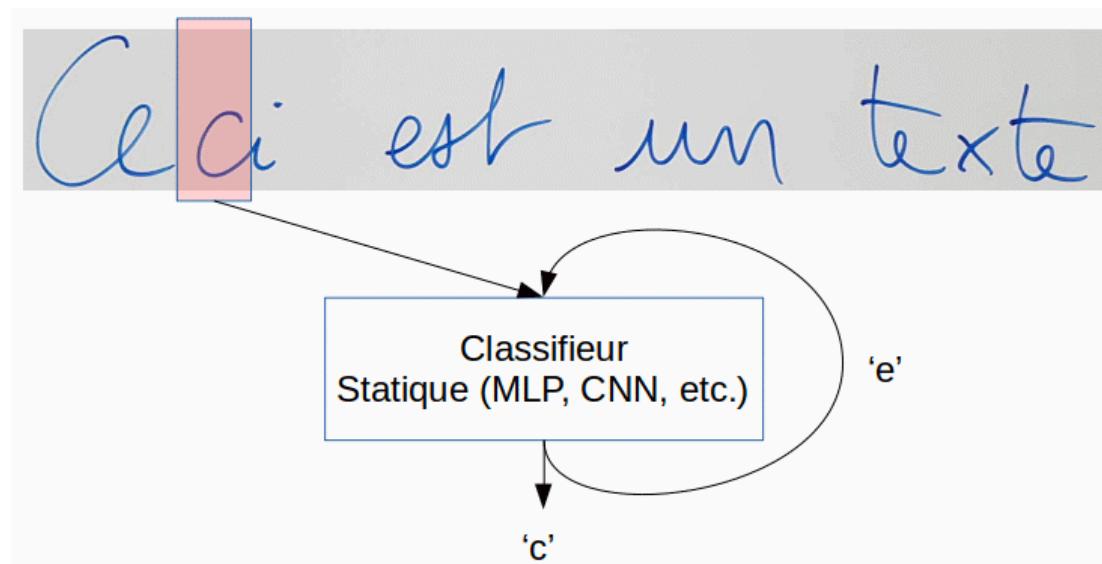


un deus ~~tre~~ quatre cinq

"En me rasant ce matin, je me suis \_\_\_\_"

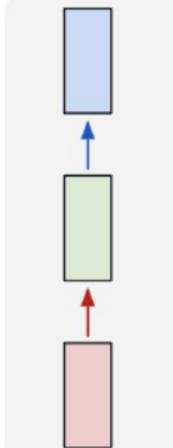
## RNN : two principles

- **Sliding window** to handle inputs from different size
- Recurrent connexions to the **past signal**

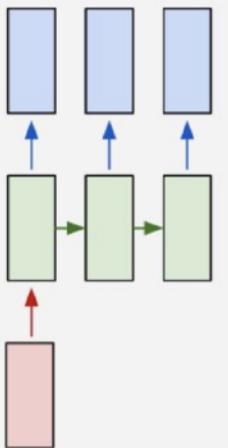


# Recurrent neural networks

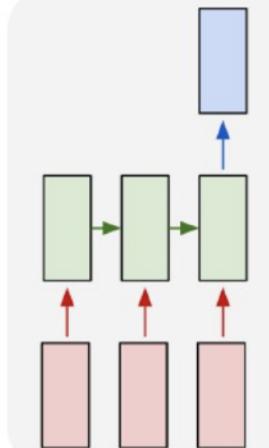
one to one



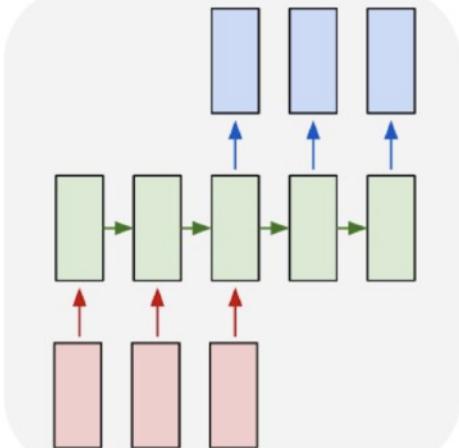
one to many



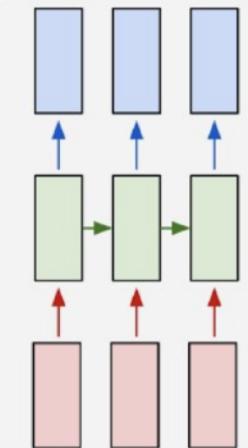
many to one



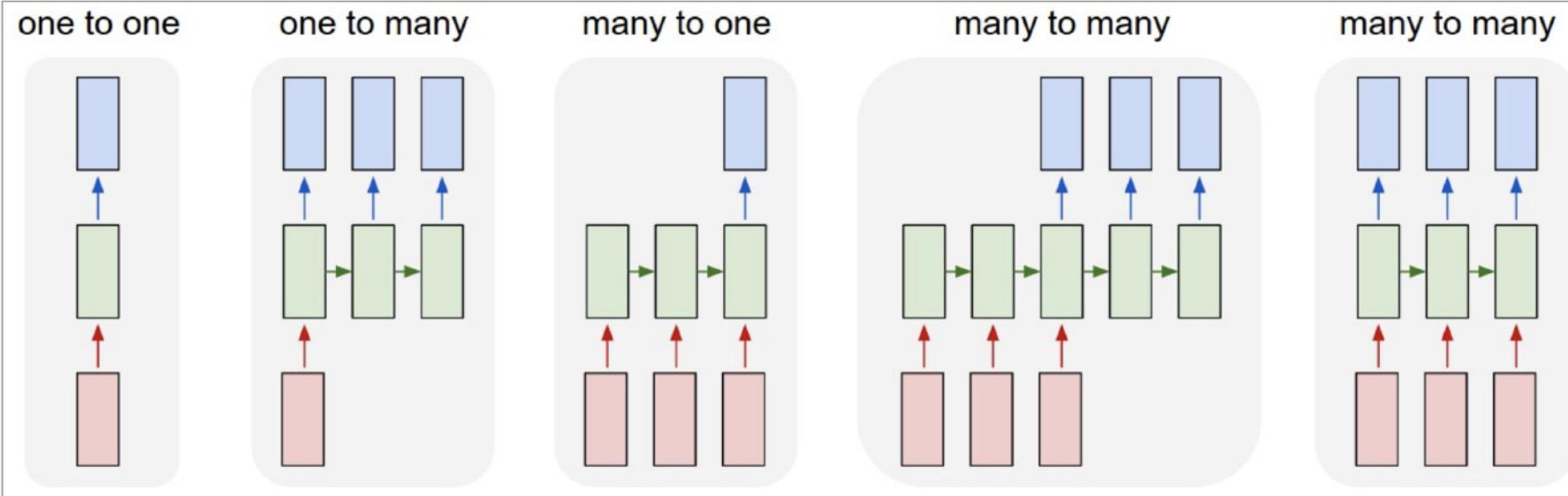
many to many



many to many

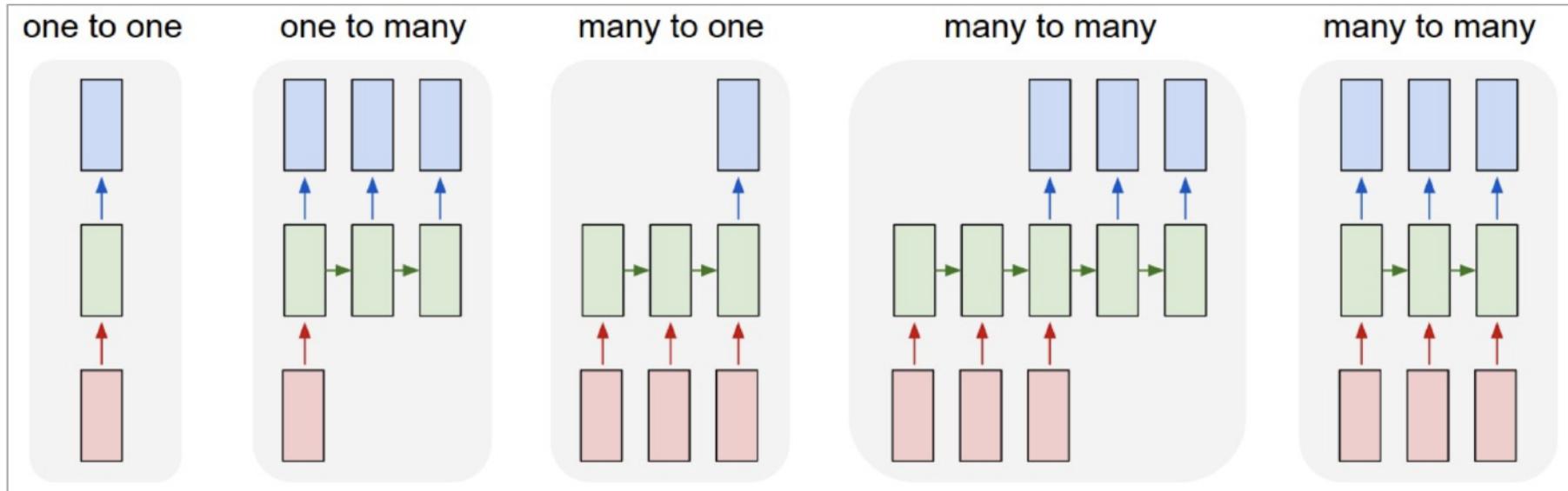


# Recurrent neural networks



Feed Forward  
NNs,  
CNNS

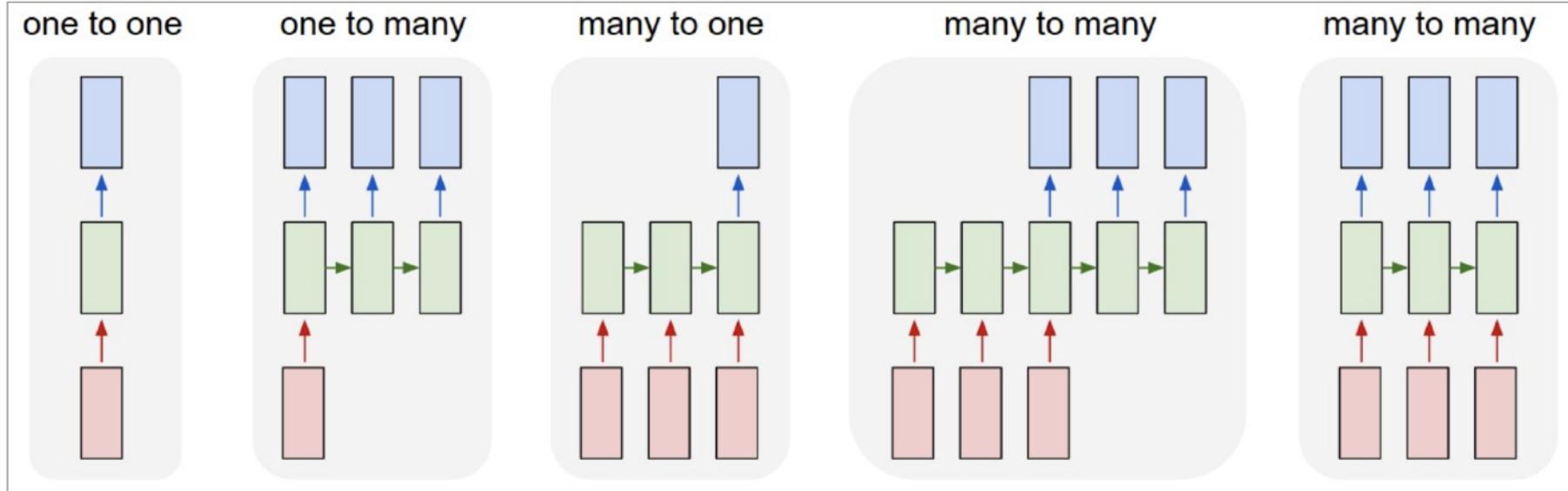
# Recurrent neural networks



Feed Forward  
NNs,  
CNNS

Image captioning  
(img  $\rightarrow$  sequence  
of words)

# Recurrent neural networks

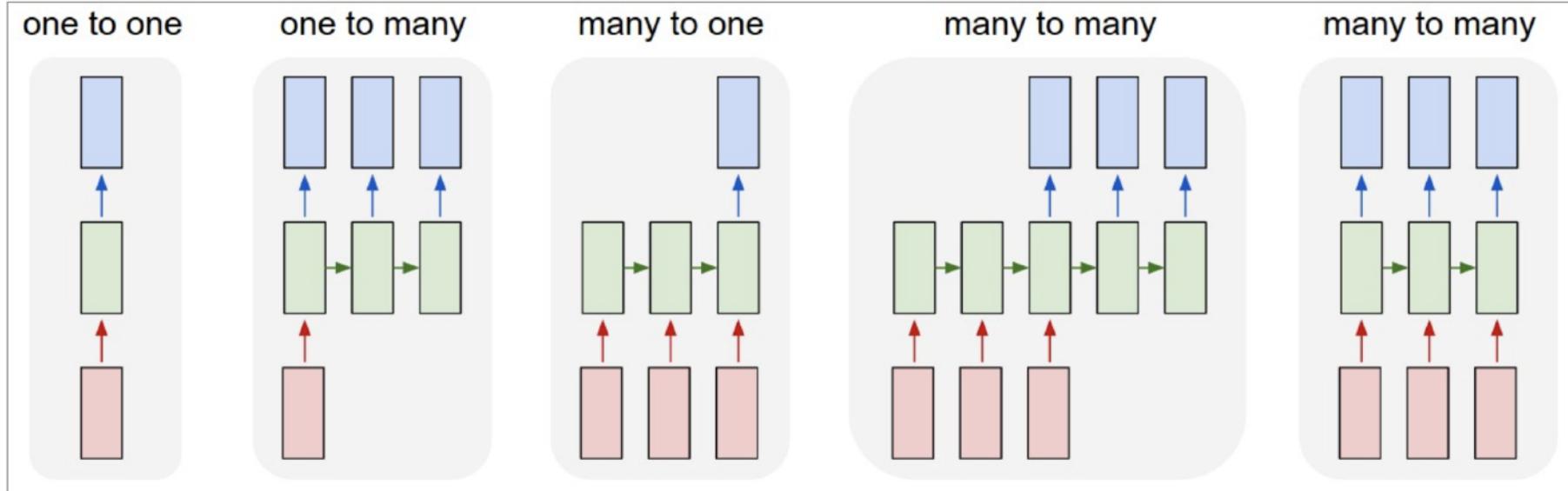


Feed Forward  
NNs,  
CNNs

Image captioning  
(img  $\rightarrow$  sequence  
of words)

Sentence classification  
(sentiment analysis)

# Recurrent neural networks



Feed Forward  
NNs,  
CNNs

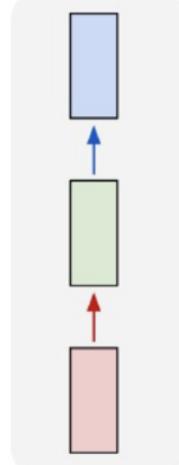
Image captioning  
(img  $\rightarrow$  sequence  
of words)

Sentence classification  
(sentiment analysis)

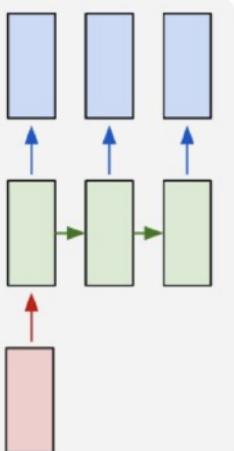
Translation

# Recurrent neural networks

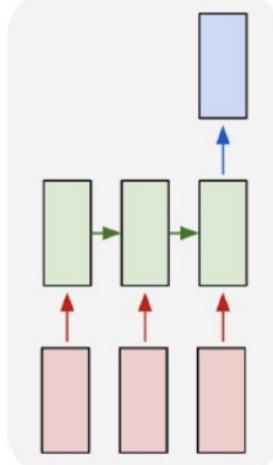
one to one



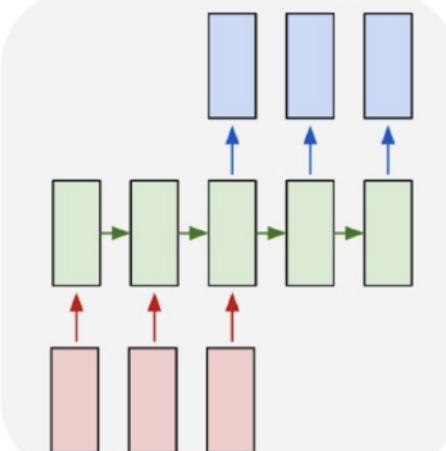
one to many



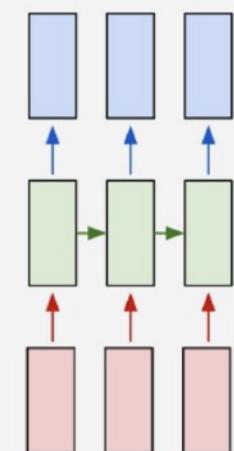
many to one



many to many



many to many



Feed Forward  
NNs,  
CNNs

Image captioning  
(img  $\rightarrow$  sequence  
of words)

Sentence classification  
(sentiment analysis)

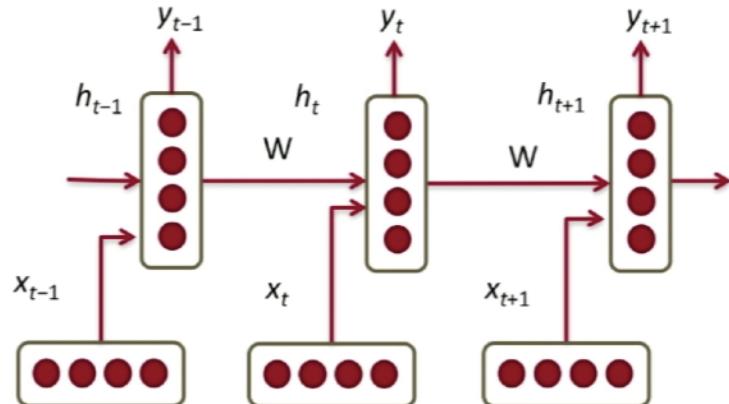
Translation

Video frame  
classification

# Recurrent neural networks

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$

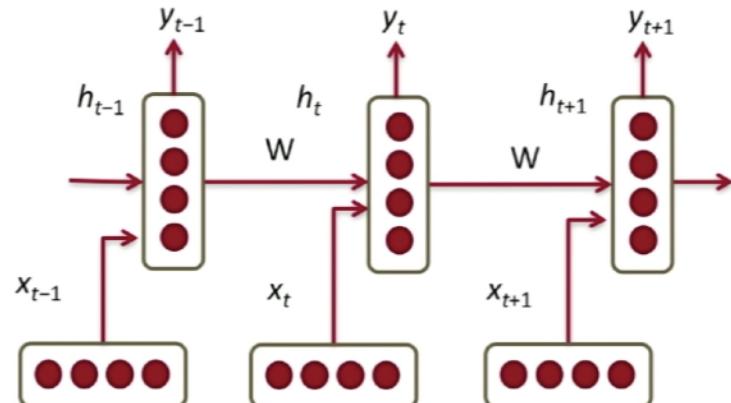


# Recurrent neural networks

past inputs dependency

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$

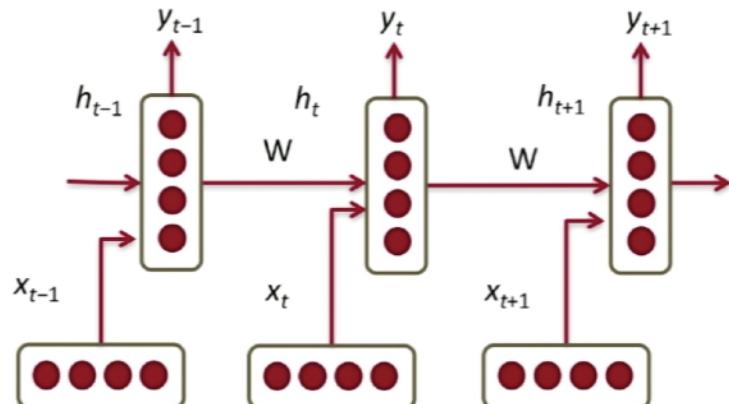


# Recurrent neural networks

current input dependency

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$

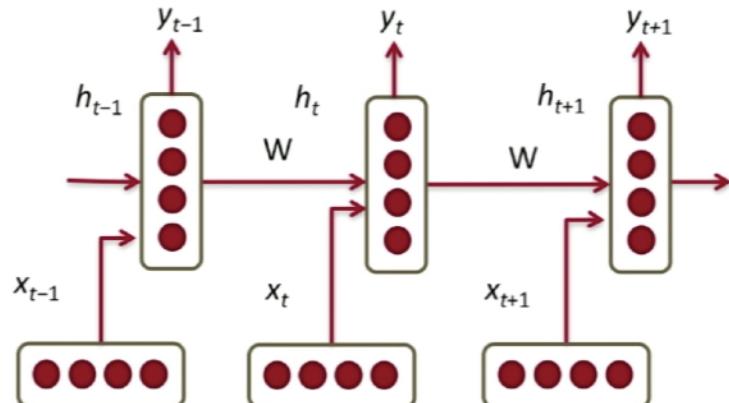


# Recurrent neural networks

$$h_t = \boxed{\sigma}(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

activation function

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$



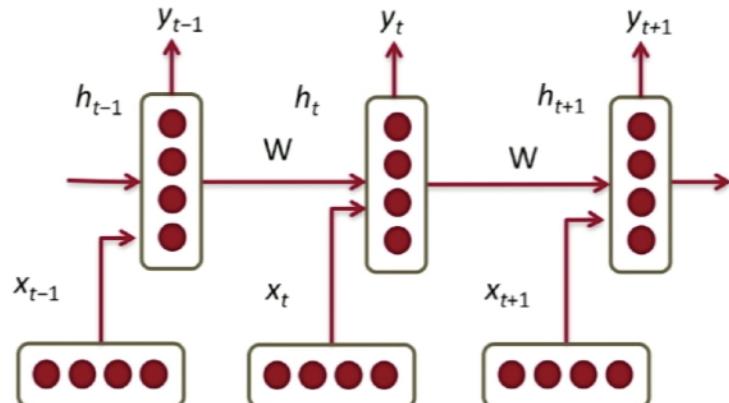
# Recurrent neural networks

current input dependency past inputs dependency

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

activation function

$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$

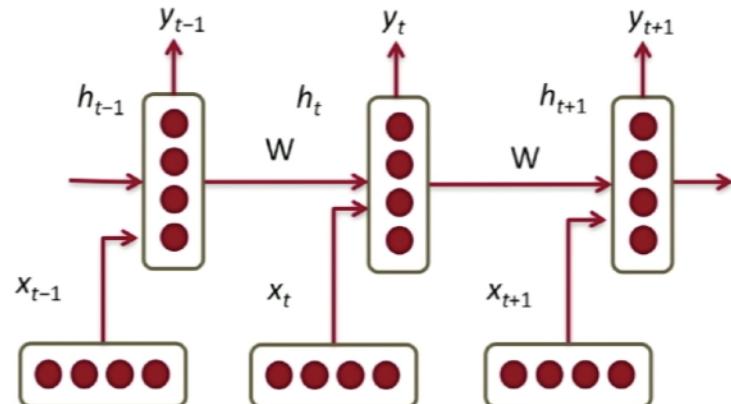


# Recurrent neural networks

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

output at time t

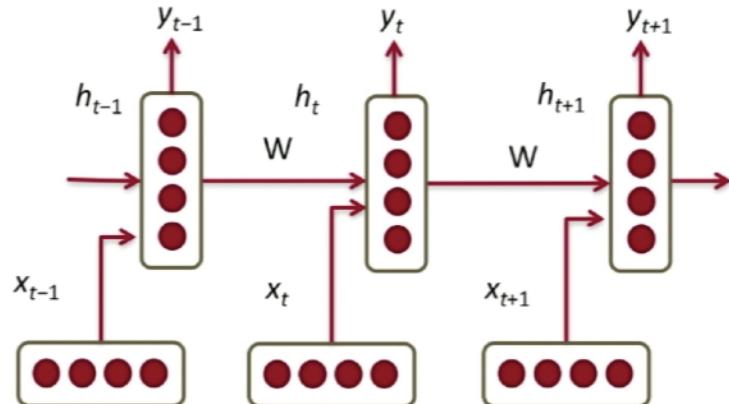
$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$



# Recurrent neural networks

$$h_t = \sigma(W^{(hx)}x_{[t]} + W^{(hh)}h_{t-1})$$

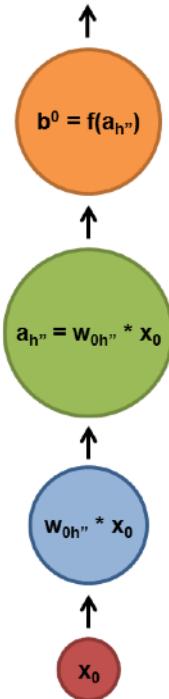
$$\hat{y}_t = \text{softmax}(W^{(S)}h_t)$$



# Recurrent neural networks

Great animated GIF see <https://i.imgur.com/kpZBDfV.gif>

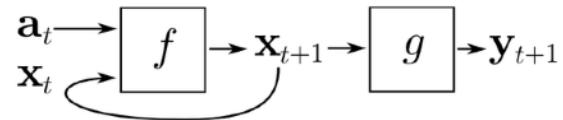
$b^0$  is fed to next layer



# Training recurrent neural networks

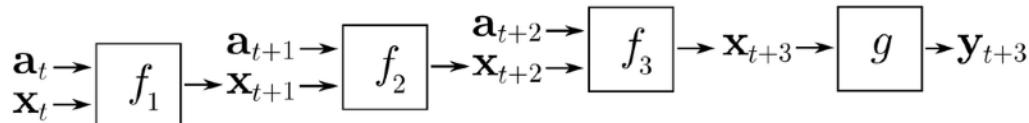
## Back Propagation Through Time (BPTT)

Neural networks, whether they are recurrent or not, are simply nested composite functions like  $f(g(h(x)))$ . Adding a time element only extends the series of functions for which we calculate derivatives with the chain rule.



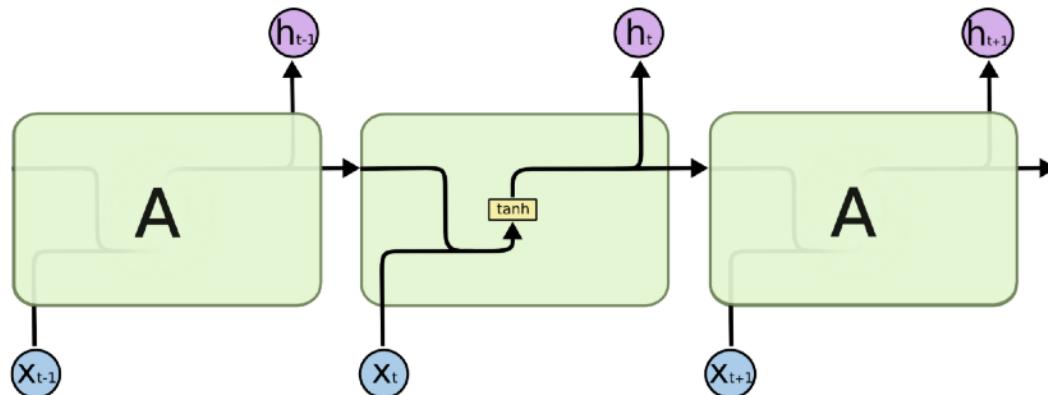
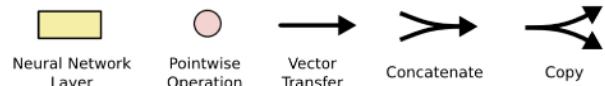
↙ unfold through time ↘

& apply usual back propagation !



# Recurrent neural networks

## RNNs



The repeating module in a standard RNN contains a single layer.

/!\ A single layer, but several neurons!

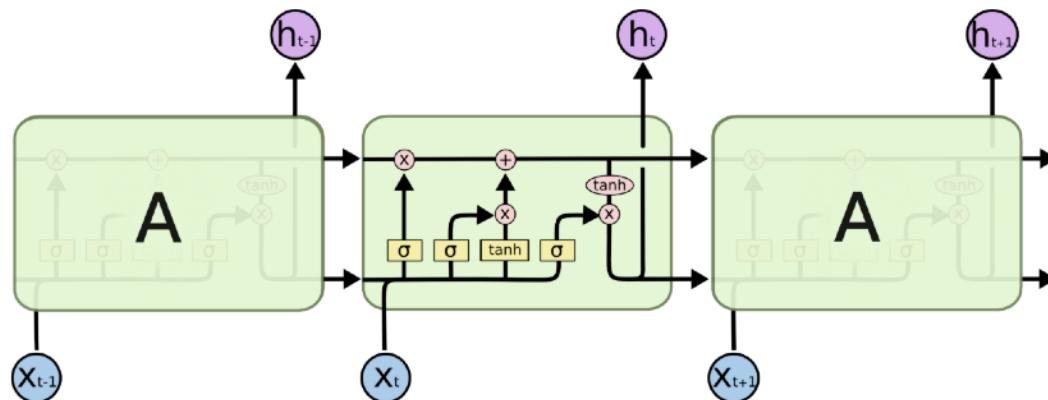
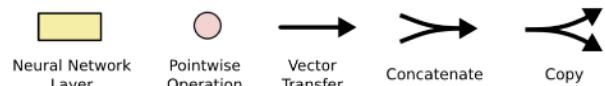
# Recurrent neural networks

**Problem** with RNNs: an "old" observation from the past will have a very small impact on the current state. The network has “forgotten” the previous states.

**Solution:** give network the possibility to control an internal memory depending on the past context.

# Recurrent neural networks

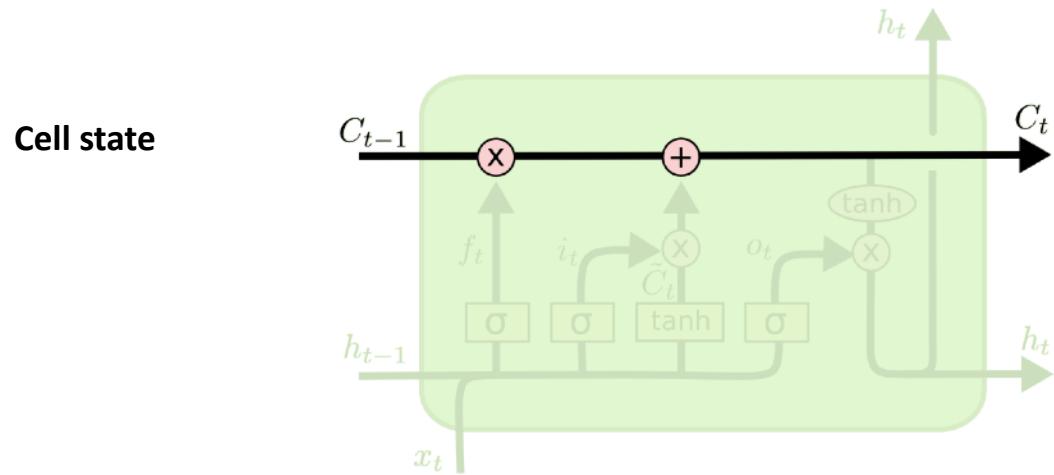
## LSTMs



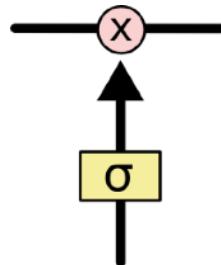
The repeating module in an LSTM contains four interacting layers.

See awesome [blog article](#) on RNNs & LSTMs

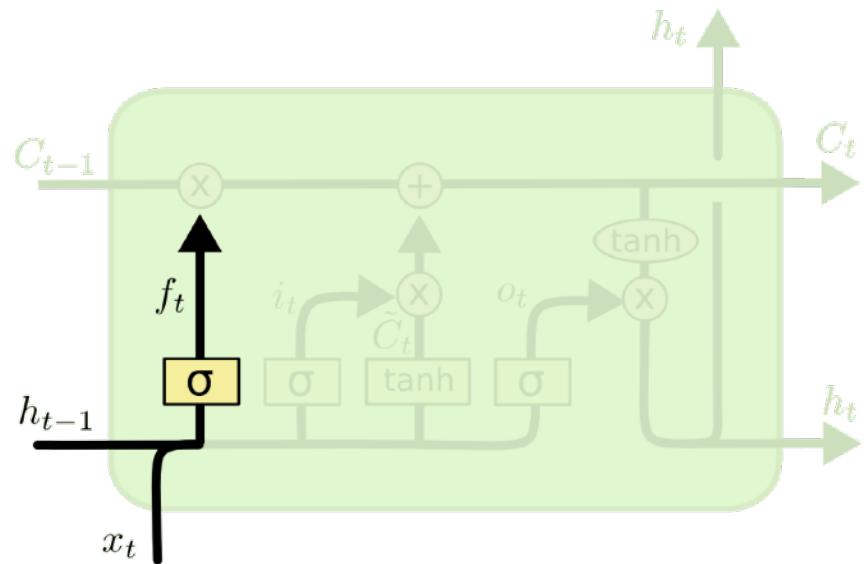
# Long Short Term Memory



**Gates:** optionally let information through



# Long Short Term Memory: forget gate

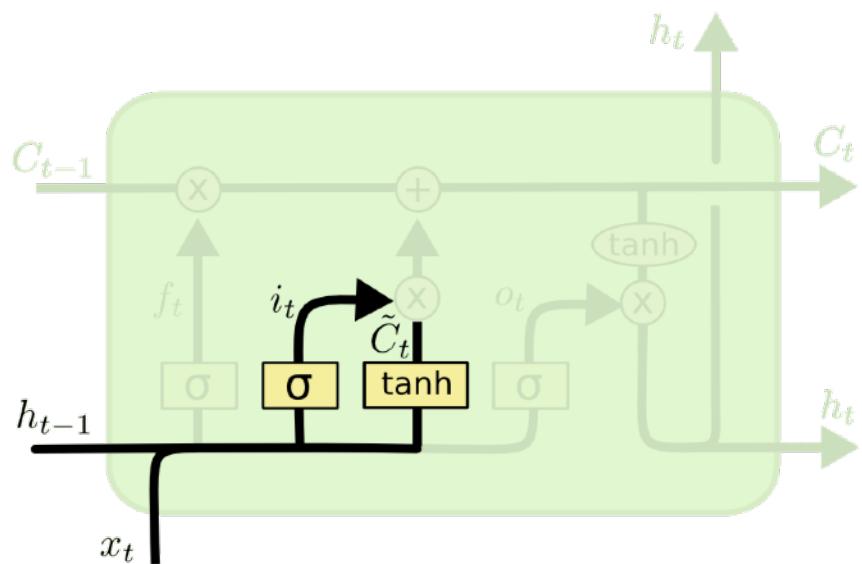


**Forget gate layer:** decide if we need to reset the information from the cell state.

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

*Intuition... if my new word  $x$  is a "", then it means that we are handling a new sentence and probably can forget the past.*

# Long Short Term Memory: input gate



**Input gate layer:** decide what information we're going to store in the cell state

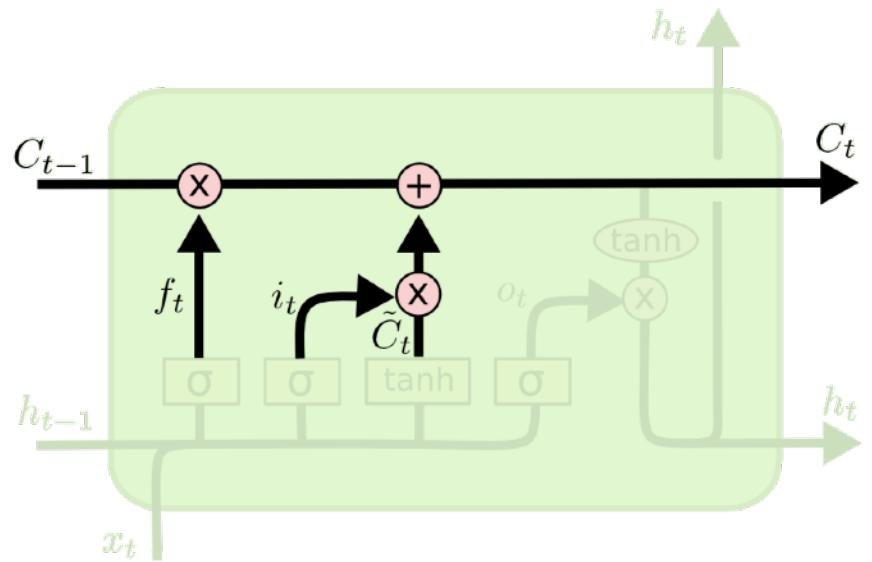
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

(Candidate values)

*Intuition... on sentiment classification task, the input gate will learn to “let get in” all signals related to sentiments (“nice”, “amazing”, “awful”, ...). Useless words like “house” will not enter the input gate.*

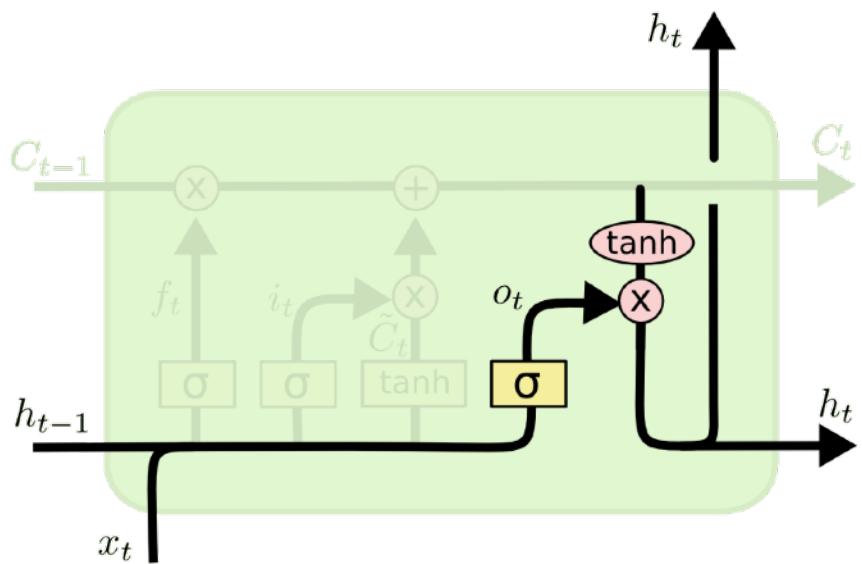
# Long Short Term Memory: cell state update



We update the cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# Long Short Term Memory: output gate!



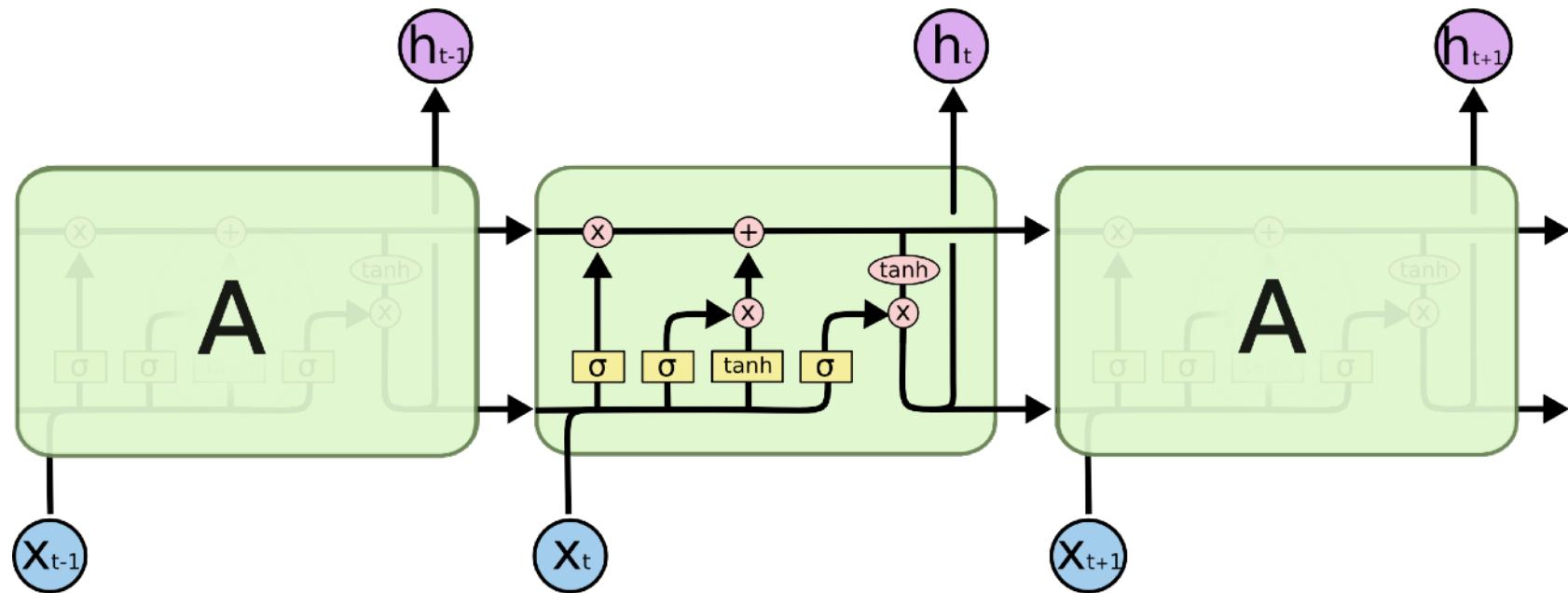
## Output:

- Decide which part of new cell state to keep
- Combine with current & past info

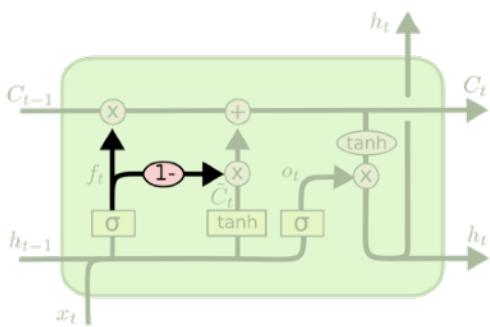
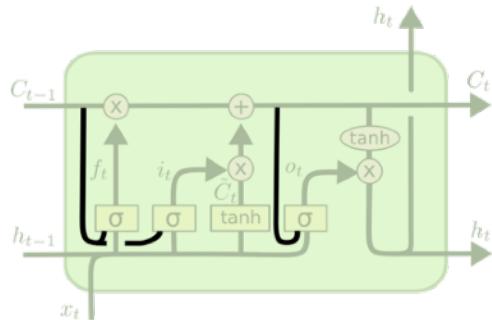
$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

# Long Short Term Memory



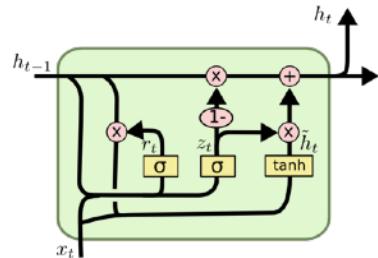
# Others



$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

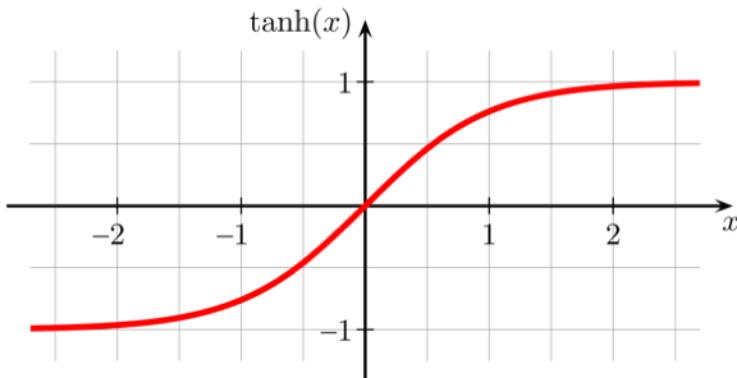
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

**GRU**  
(gated recurrent unit)

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

# Recurrent neural networks usual activation function



$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} = \frac{1 - e^{-2z}}{1 + e^{-2z}}.$$

## Why tanh?

LSTMs manage an internal state vector whose values should be able to increase or decrease when we add the output of some function.

Sigmoid output is always non-negative; values in the state would only increase. The **output from tanh can be positive or negative**, allowing for **increases and decreases** in the state.

Remember... vanishing & exploding gradients gradients on RNNs

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left( \prod_{t=2}^k \sigma'(W_{rec} \cdot c_{t-1} + W_{in} \cdot x_t) \cdot W_{rec} \right) \frac{\partial c_1}{\partial W}$$

---

**As timesteps grow, chances are that our gradients vanish / explode.**

# Vanishing gradient: LSTMs are more **robust**!

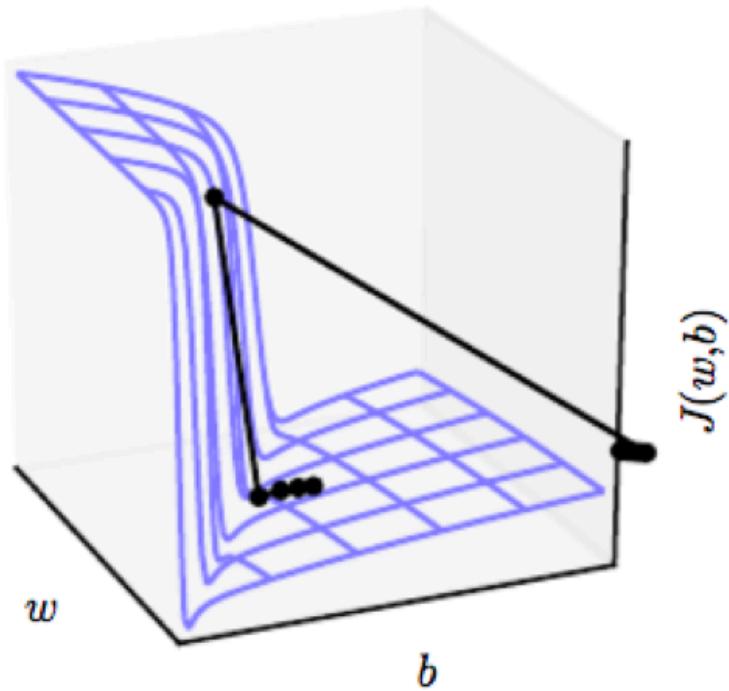
$$h_t = o_t * \tanh(C_t)$$

has a gradient less  
dependent to the  
derivative of activation  
functions

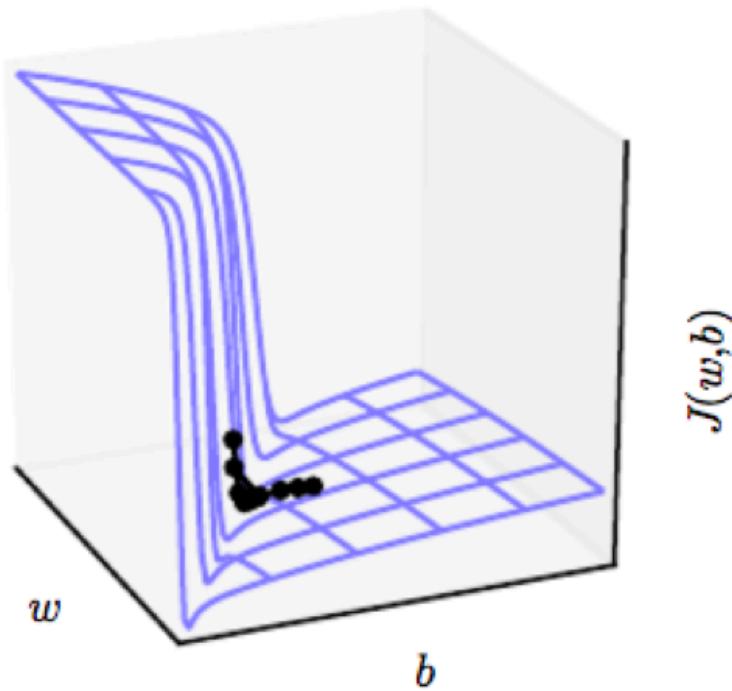
**At time step, we may find an appropriate parameter update of the cell state such that the gradient does not disappear.**

# Exploding gradient: gradient clipping technique

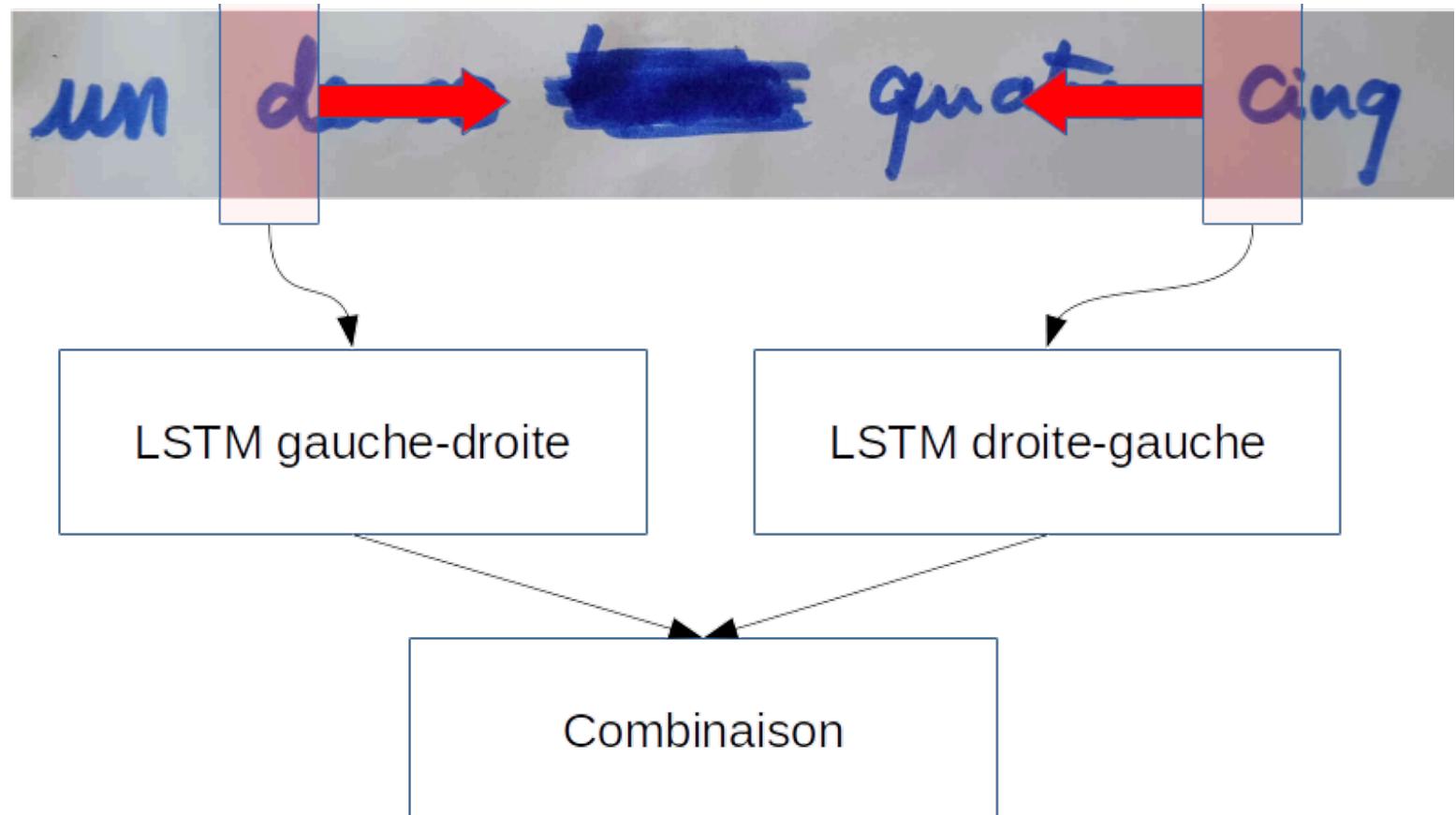
Without clipping



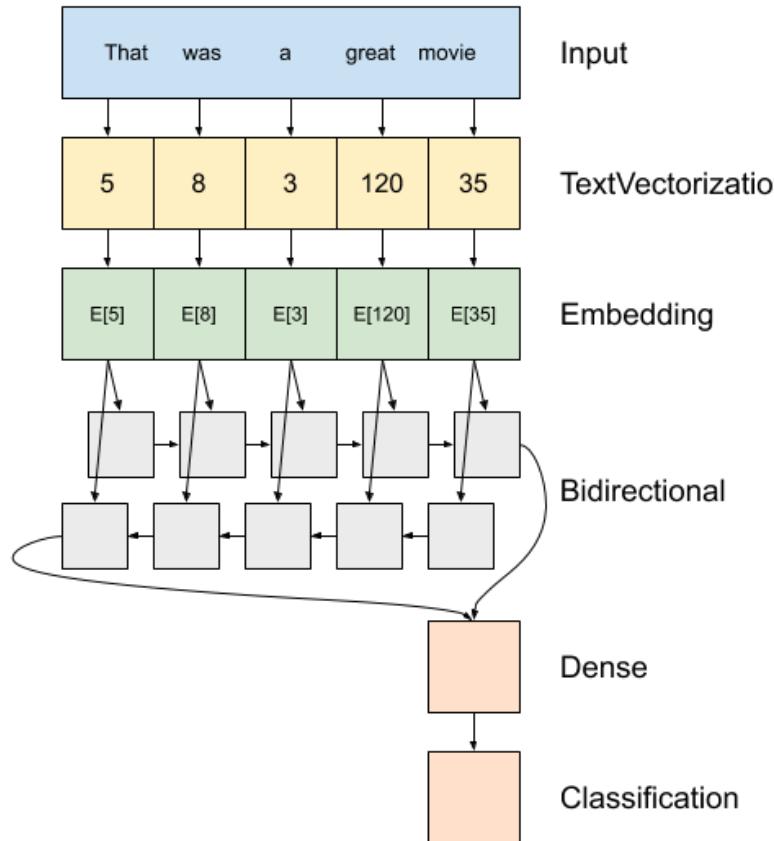
With clipping



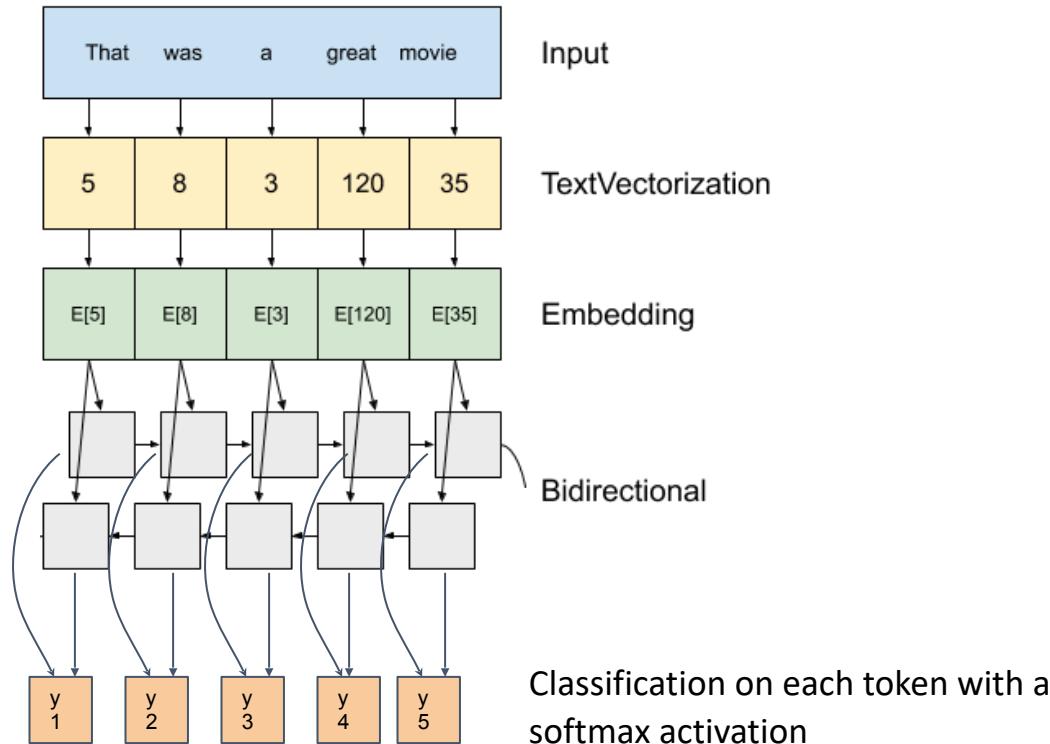
# Bidirectionality: bi-LSTM



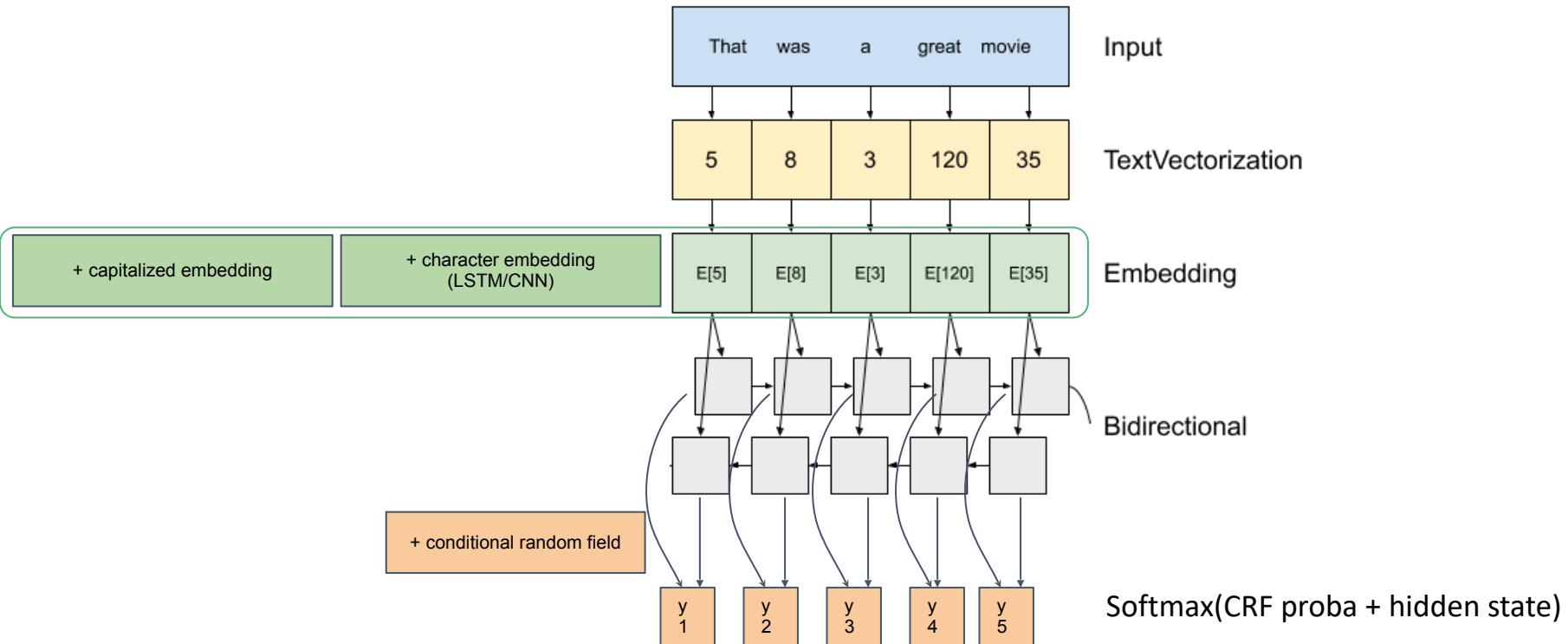
# Bidirectionality: bi-LSTM for sequence classification



# Bidirectionality: bi-LSTM for token classification (NER)



You can concatenate your RNN features with other layers of features!





# Natural language processing

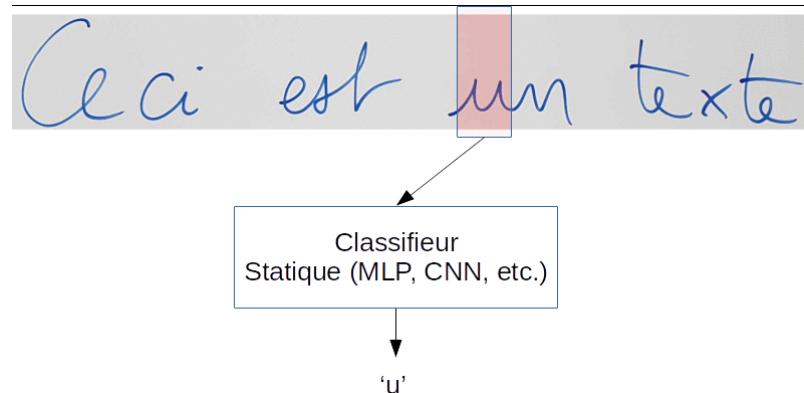
# A text is a sequence.

Text  
=

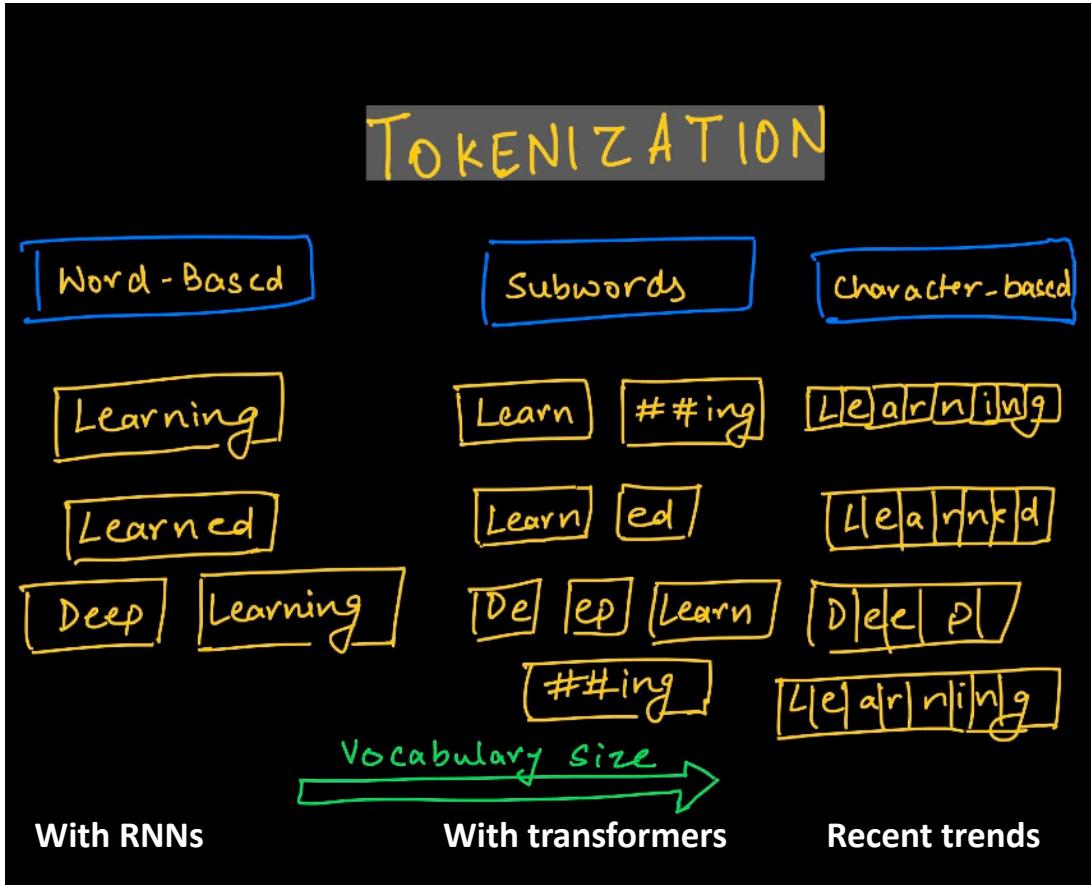
sequence of words?  
sequence of letters?

...

## What is our sliding window?



# Tokenization

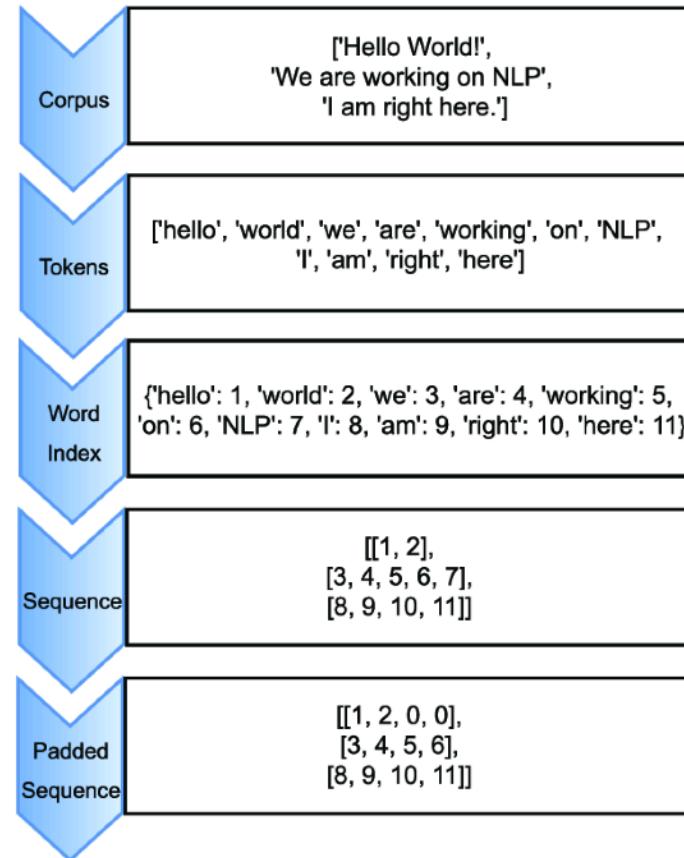


- Subword tokenization has been shown to perform poorly on noisy input (misspelling, ...)
- In addition, it imposes a dependence on the tokenization, which can lead to a mismatch when adapting a model to new data.

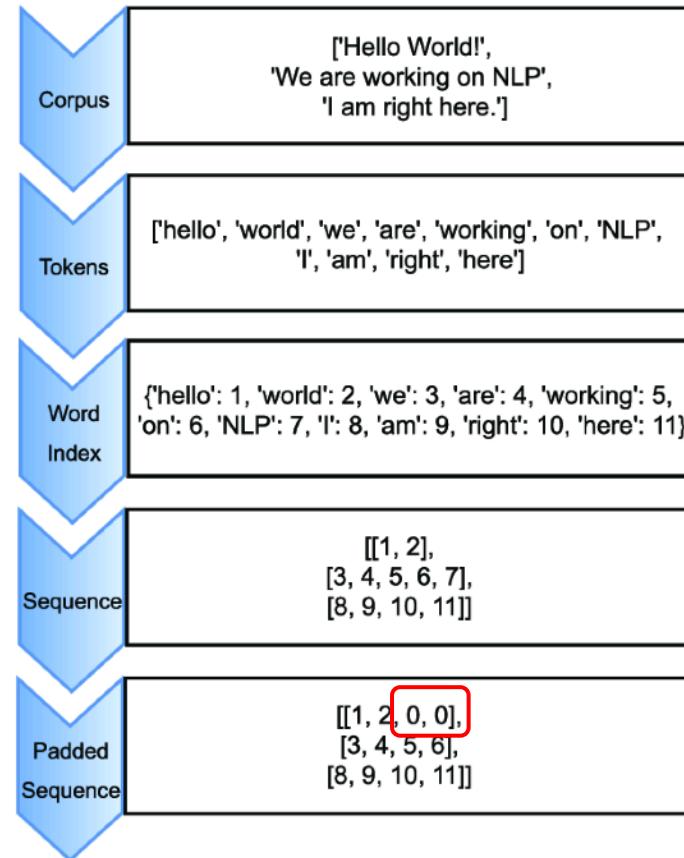
-> 2021 saw the emergence of new token-free methods that directly consume a sequence of characters.

<https://ruder.io/ml-highlights-2021/index.html#12tokenfreemodels>

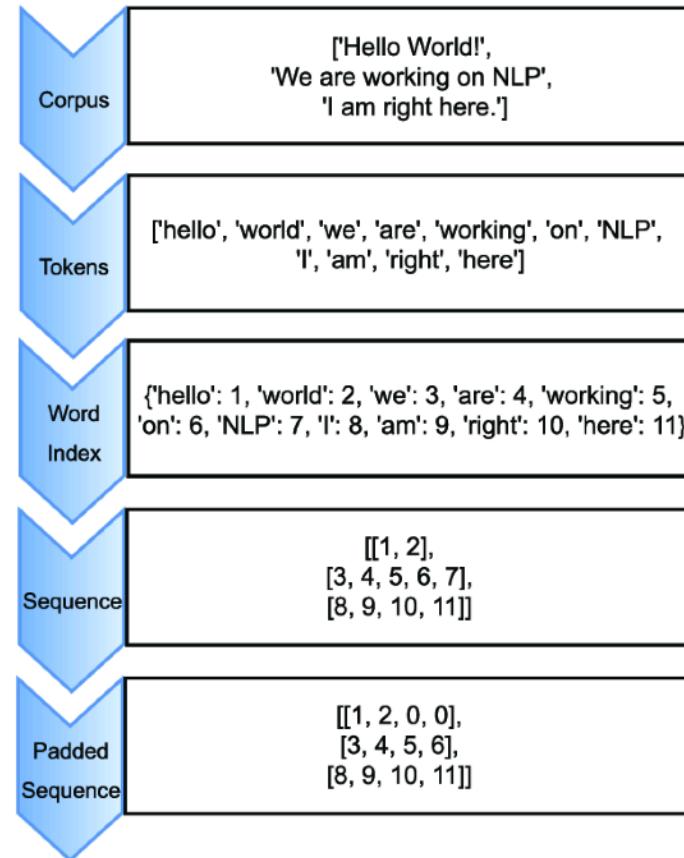
# Padding (for batch creation)



# Padding (for batch creation)



# Padding (for batch creation)



Same shape as images!

# Text normalization & preprocessing?

- Lowercase... or not.
- Stem, lemmatize... or not.
- Remove stopwords... or not.
- Remove punctuation... or not.

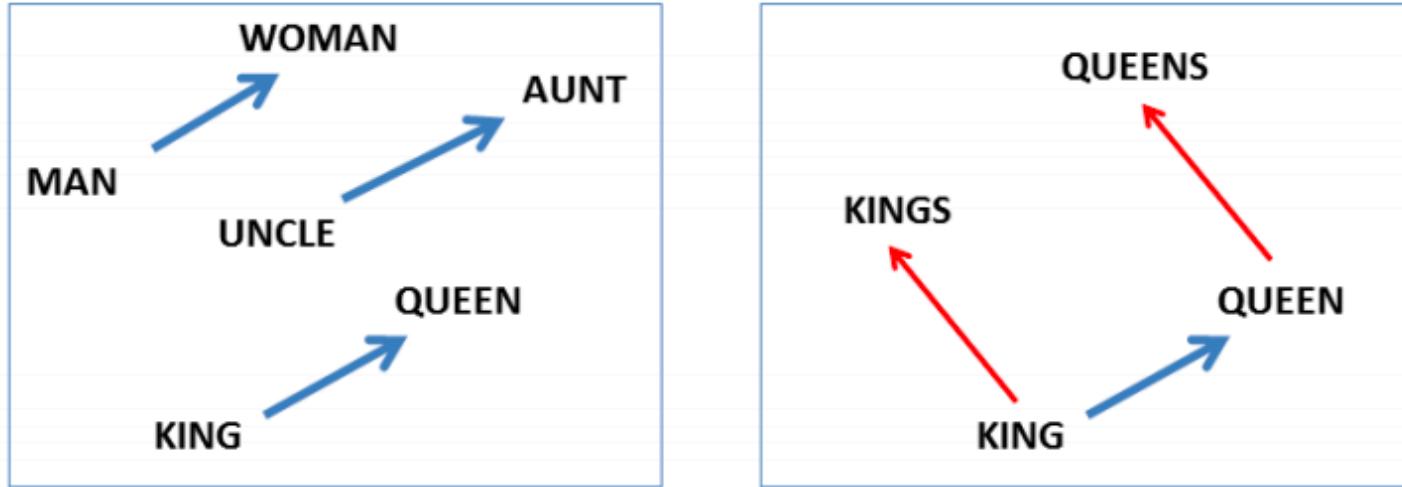
# Text normalization & preprocessing?

- Lowercase... or not.
- Stem, lemmatize... or not.
- Remove stopwords... or not.
- Remove punctuation... or not.

Can depend on your problematic. Sometimes it can help (remove sparsity in the data for example), but it can also remove interesting data (a “but”, “however”, “??”, “!!”).

But keep in mind that Transformers have their own preprocessing (WordPiece), and that you have to use the same one if using a pre-trained model.

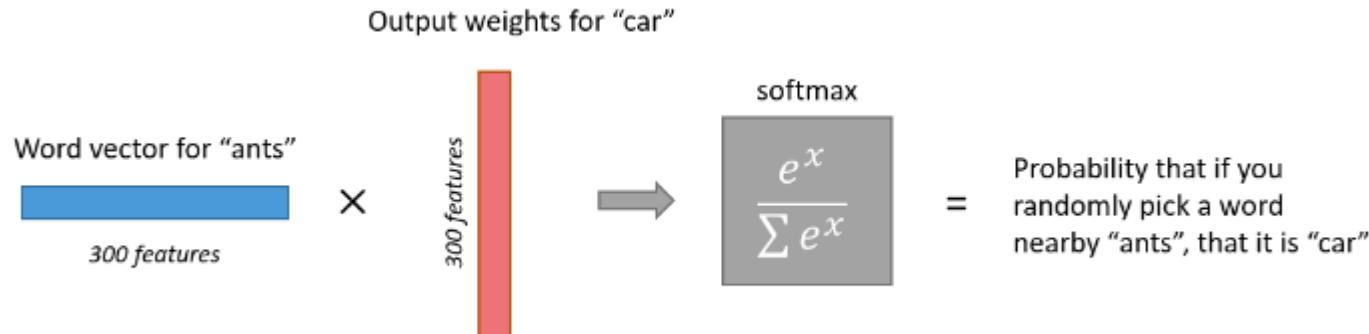
# Word embeddings



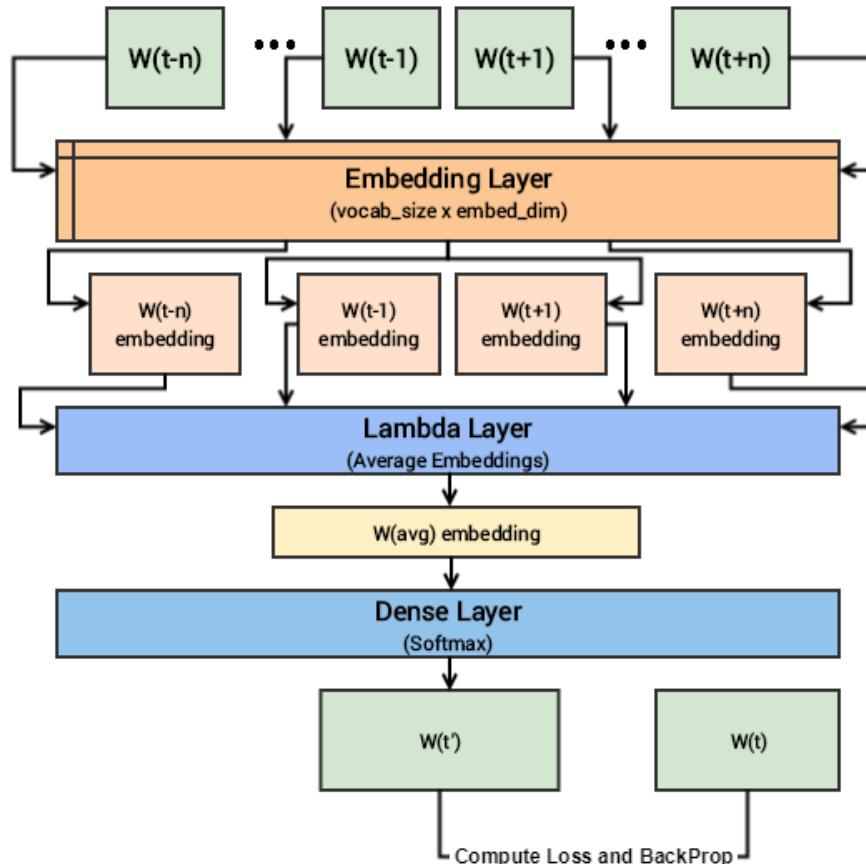
(Mikolov et al., NAACL HLT, 2013)

# Word embeddings: doc2vec (CBOW)

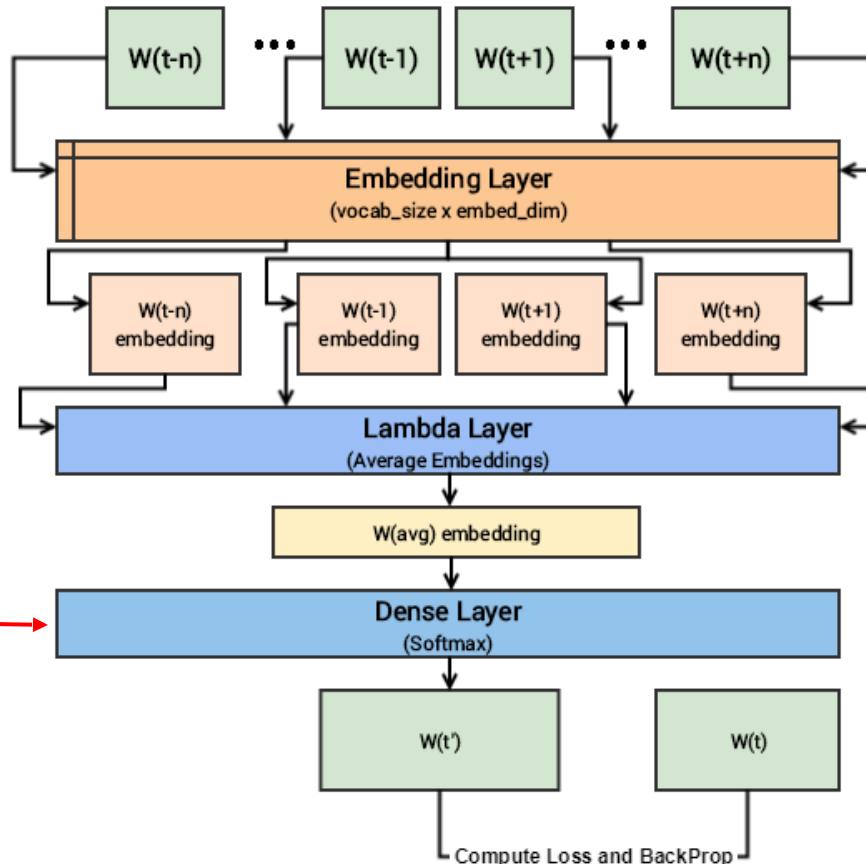
Word2vec = simple linear perceptrons with 1 hidden layer



# Word embeddings: doc2vec (CBOW)



# Word embeddings: doc2vec (CBOW)

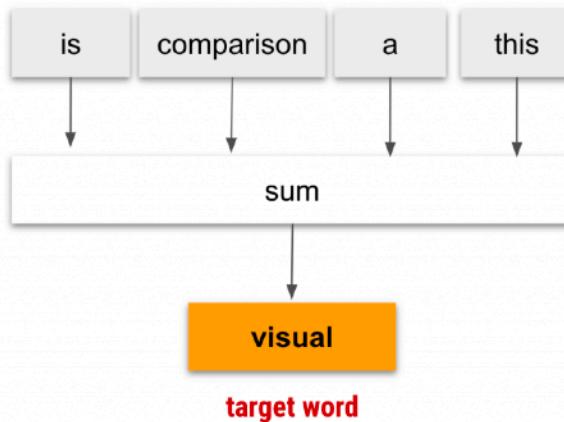


Softmax because this is  
a classification task!

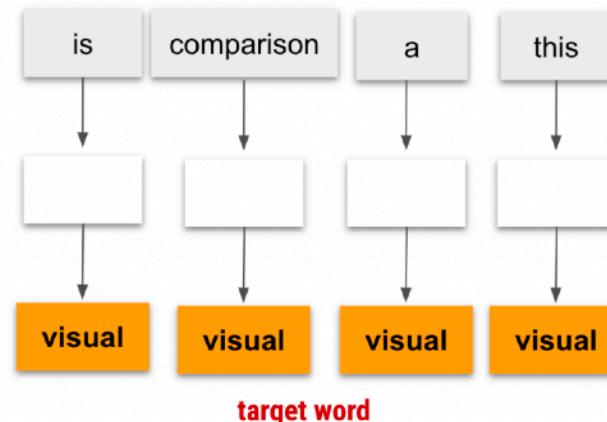
# Word embeddings: doc2vec (SkipGram)

Also word2vec but trained on a different task: given a word, predict a context.

CBOW



SkipGram

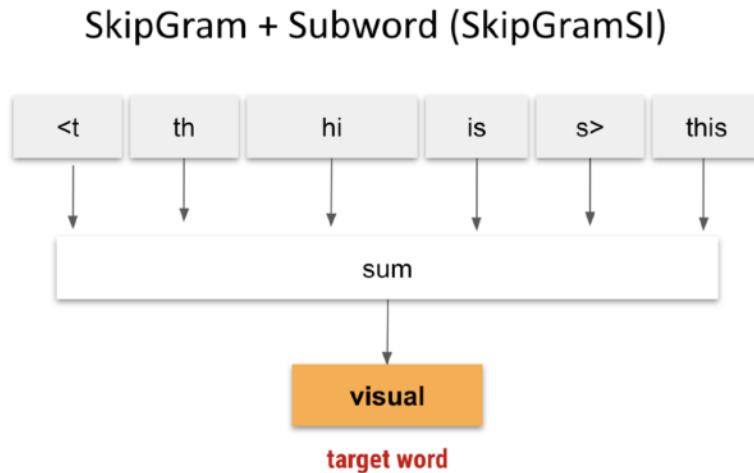


By: Kavita Ganesan

This is a visual comparison

# Word embeddings: fastText

Unlike Word2Vec, which under the hood uses words to predict words, fastText operates at a more granular level with **character n-grams**. Where words are represented by the **sum of the character n-gram vectors**.



This is a visual comparison

Skip-Gram with character n-gram information (character n-gram size=2).

# Word embeddings: fastText

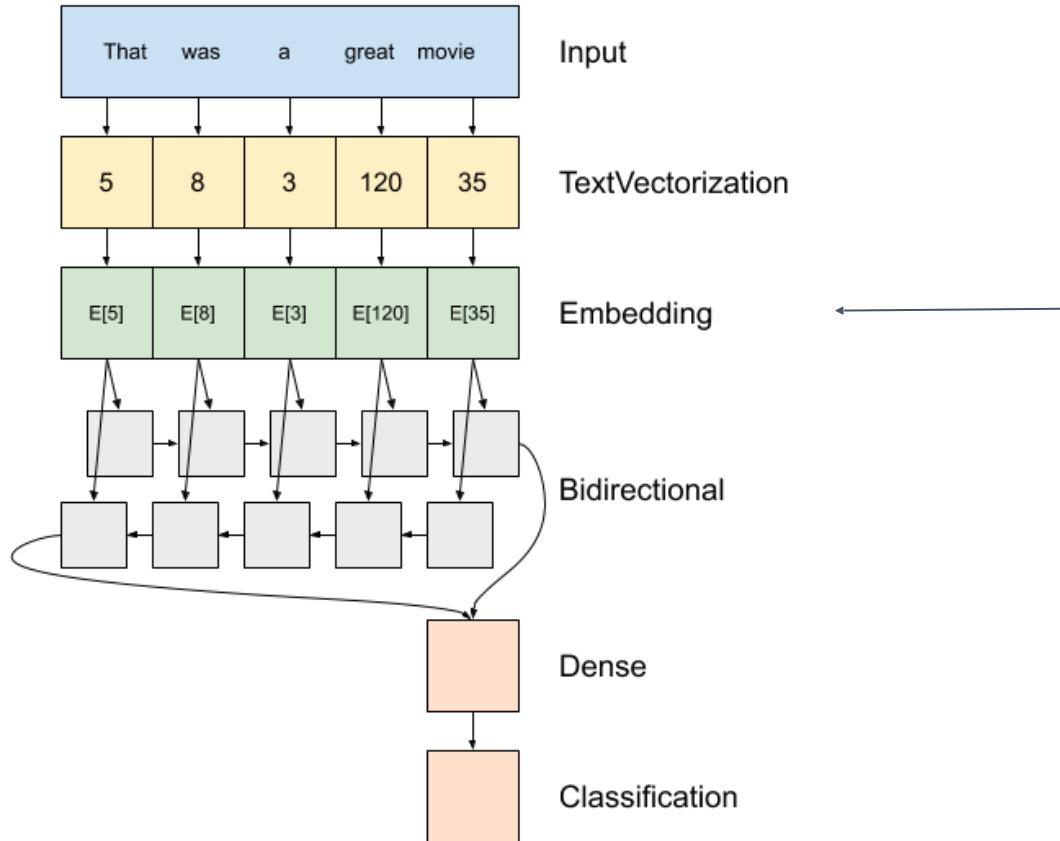
*Intuition... works better for morphologically rich languages, like german.*

Tennis

Tischtennis

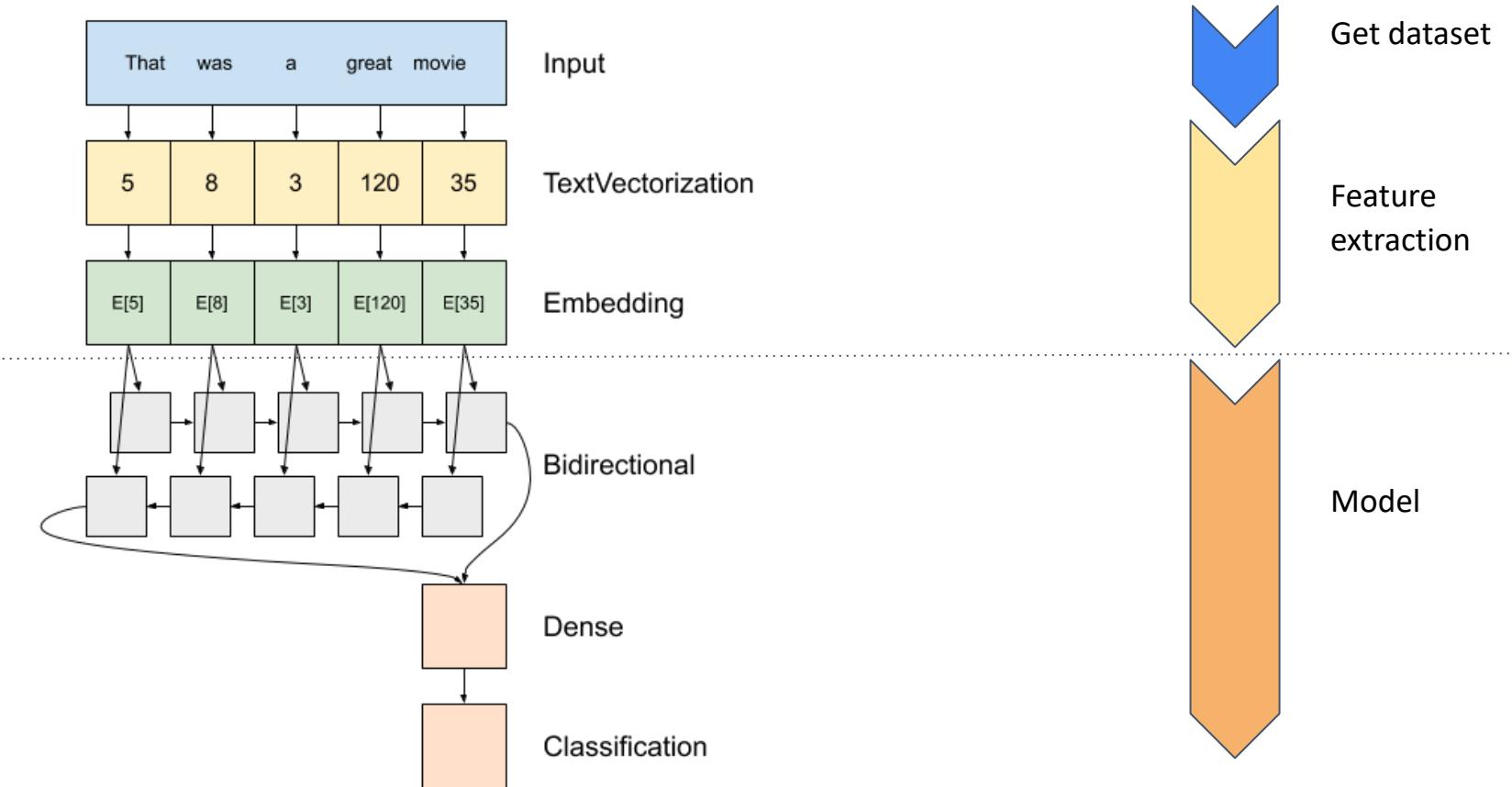
*-> n-grams overlapping, making them closer in vector space*

# Transfer learning on texts!



We can use pre-trained word embeddings on very large datasets like Wikipedia to initialize the weights of this layer!

# Transfer learning on texts!



# Non-supervised algorithms on word embeddings!



(Mikolov et al., NAACL HLT, 2013)

# Non-supervised techniques with word embeddings!



From pixels to image...

What if you take the average?

# Non-supervised techniques with word embeddings!

You lose a lot of information!!

Yes, but sometimes it can be sufficient...

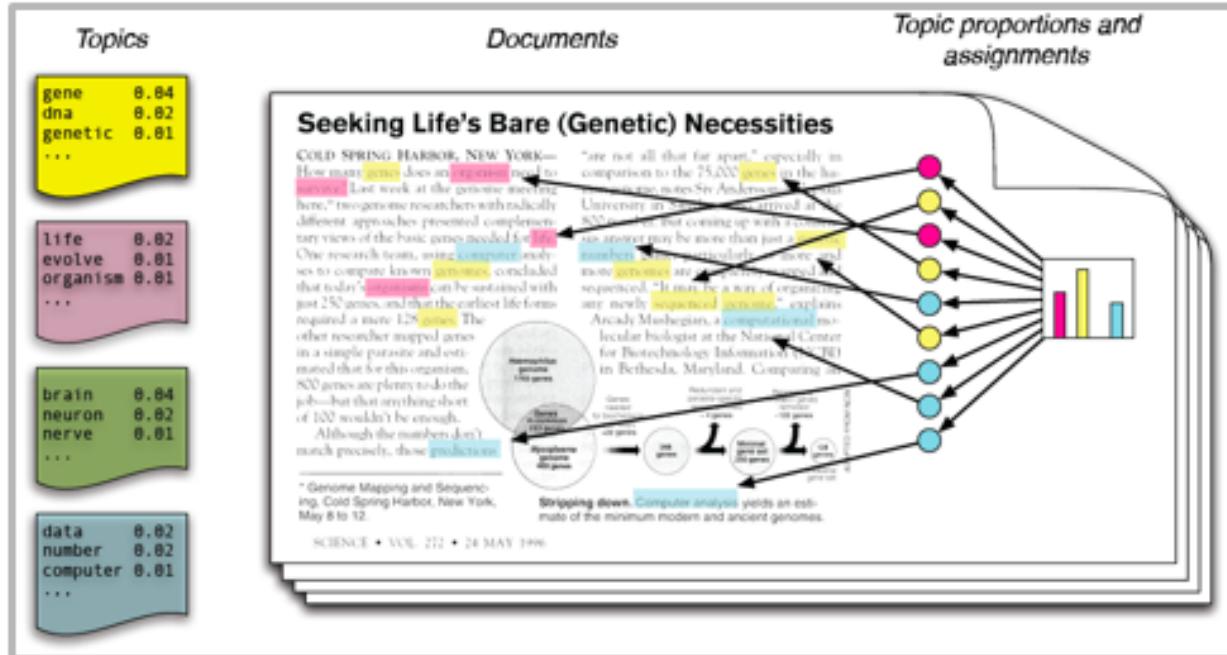
## Non-supervised techniques: bag of words (no order)

Le chat attrape la souris

≠

La souris attrape le chat

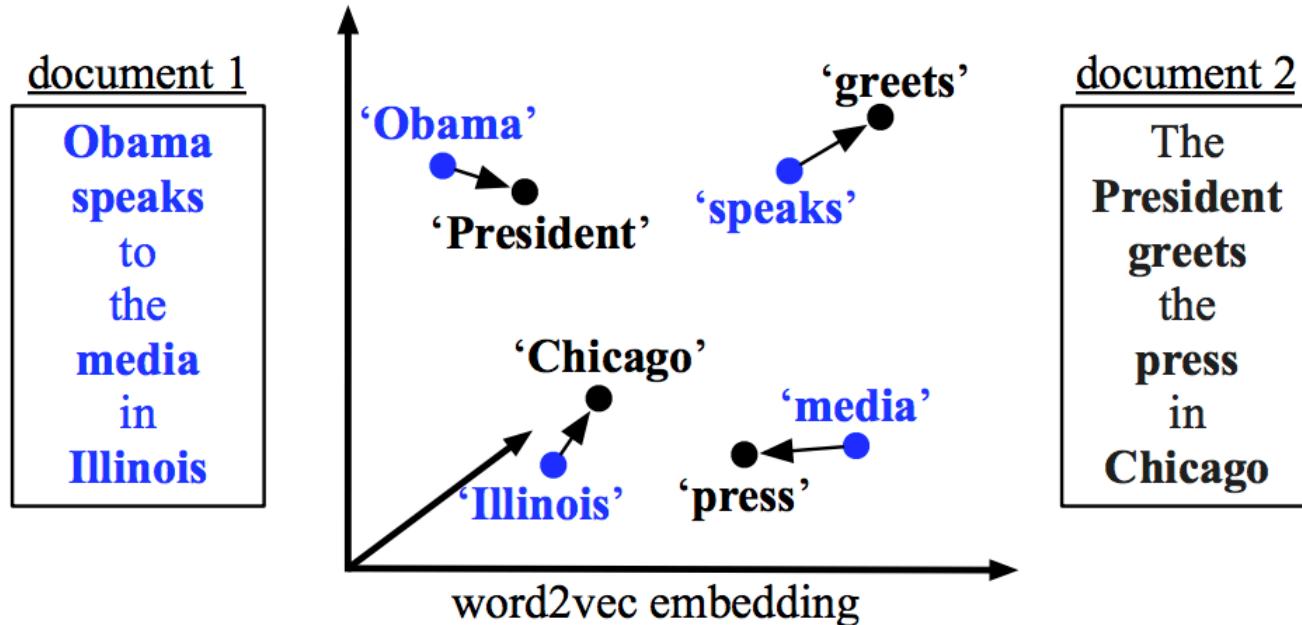
# Non-supervised techniques: topic modelling



# Non-supervised techniques: text classification



# Non-supervised techniques: text similarities (recommendation, ...)



# Language models

*A statistical language model is a probability distribution over sequences of words.*

A language model is learnt on an auto-supervised way with:

- Next word prediction task (=causal language model)
- Word prediction in a context (=masked language model)
- Next sentence prediction task
- ...

# RNNs & NLP: take away messages

- RNN versus LSTM
- Backpropagation through time
- Vanishing & exploding gradients
- Bidirectionality
- Tokenization
- Padding
- Word embeddings (word2vec, fastText)
- Language models



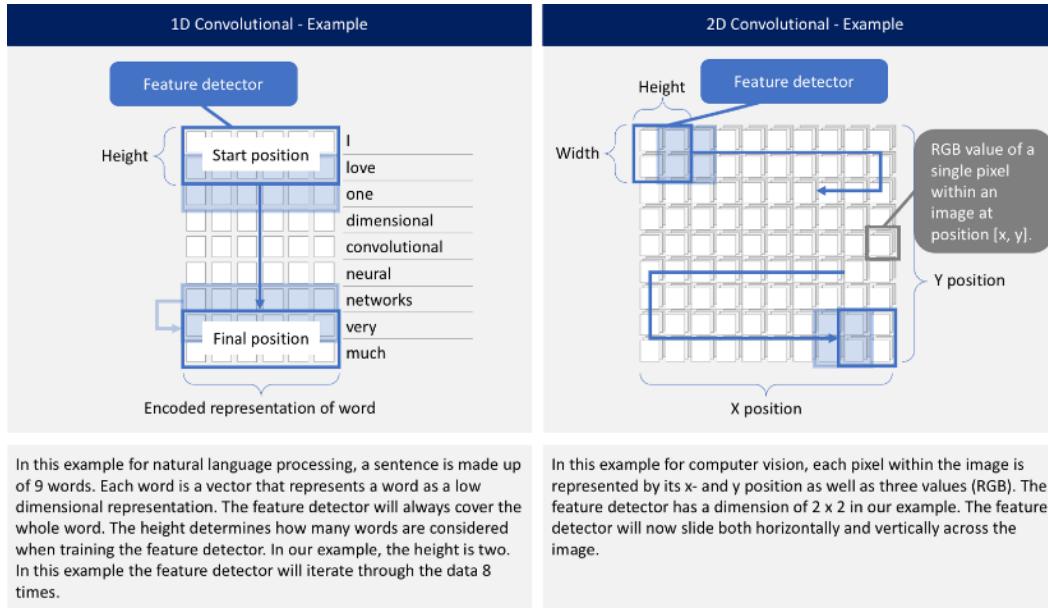
# SENTIMENT ANALYSIS USING RNNs

# Sentiment analysis using RNNs

[“Play with RNNs” notebook](#)

# About 1D convolutions

Great [blog post](#) about time series

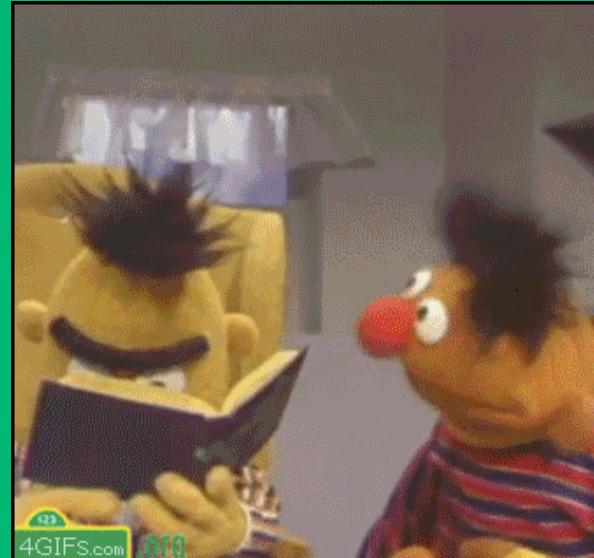


# GOING DEEPER

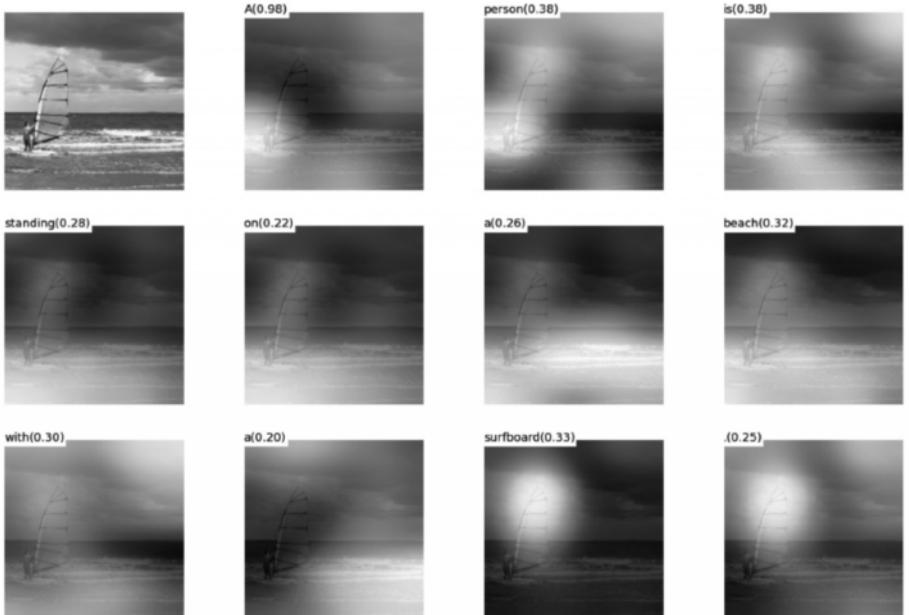
## Transformers



# But first, some other concepts



# Attention mechanisms

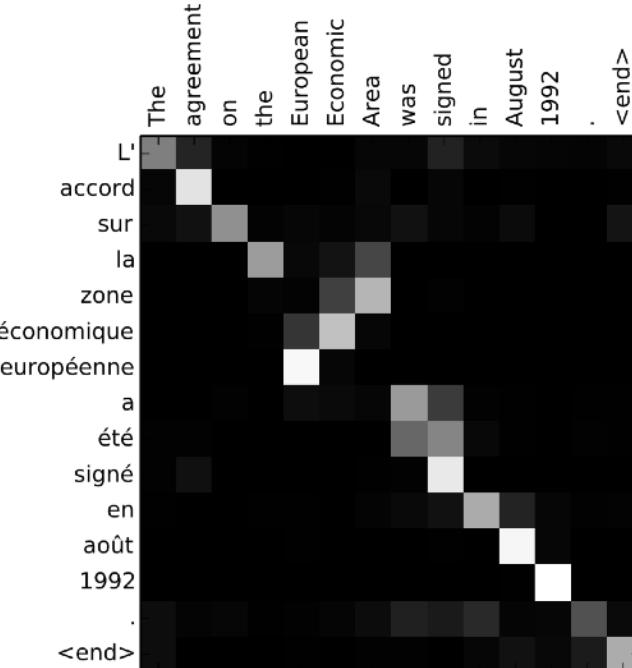


(b) A person is standing on a beach with a surfboard.

# Attention mechanism

First paper introducing the concept:

[Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau, Cho & Bengio, sept 2014](#)



**Classic translation system:** map the meaning of a sentence into a fixed-length vector representation and then generate a translation based on that vector.

**Problem:** for very long sentences, not all can be encoded. Long-range dependencies are still problematic for LSTM.

# Attention mechanism

*“The basic problem that the attention mechanism solves is that it allows the network to refer back to the input sequence, instead of forcing it to encode all information into one fixed-length vector.*

*Interpreted another way, the attention mechanism is simply giving the network access to its internal memory, which is the hidden state of the encoder.”*

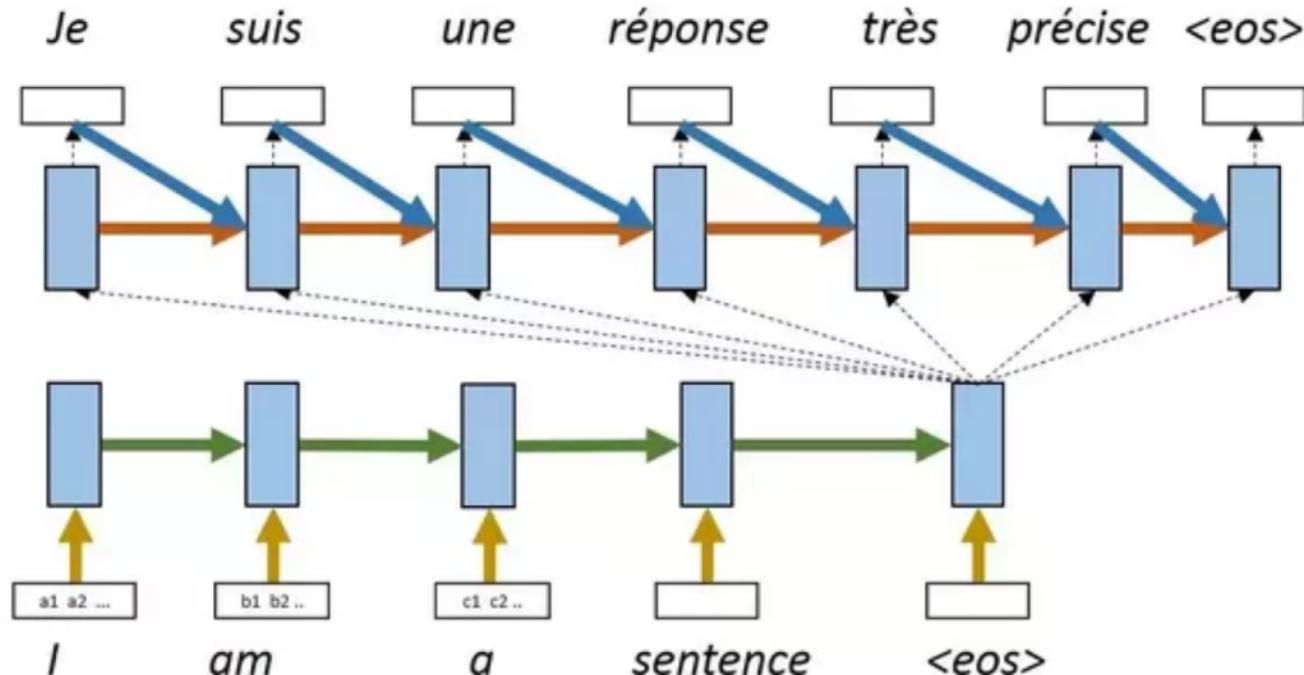
**“You can’t cram the meaning of a whole %&!\$ing sentence into a single \$&!\*ing vector!”**

**Ray Mooney**

ACL-14 Workshop on Semantic Parsing talk

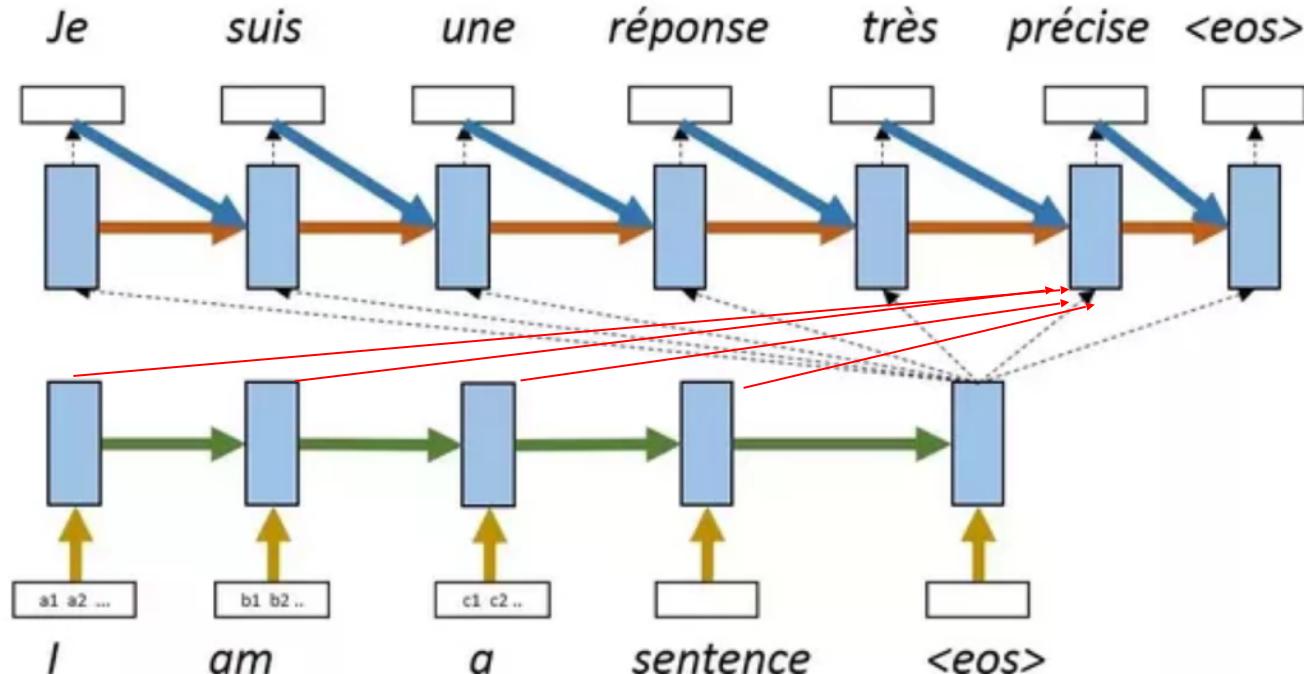
First paper introducing the concept: [Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau, Cho & Bengio, sept 2014](#)

# Attention mechanism



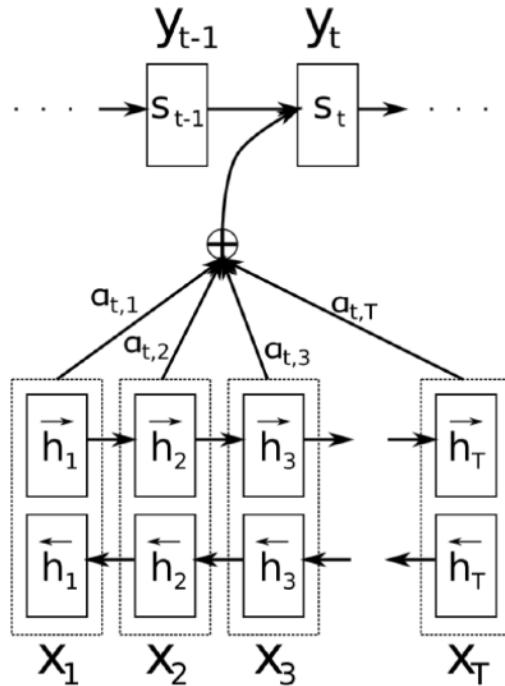
First paper introducing the concept: [Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau, Cho & Bengio, sept 2014](#)

# Attention mechanism



First paper introducing the concept: [Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau, Cho & Bengio, sept 2014](#)

# Attention mechanism



for a model generating a state  $h_t$  at each timestep, compute a context vector  $c_t$  as the weighted mean of the state sequence  $h$ :

$$c_t = \sum_{j=1}^T \alpha_{tj} h_j$$

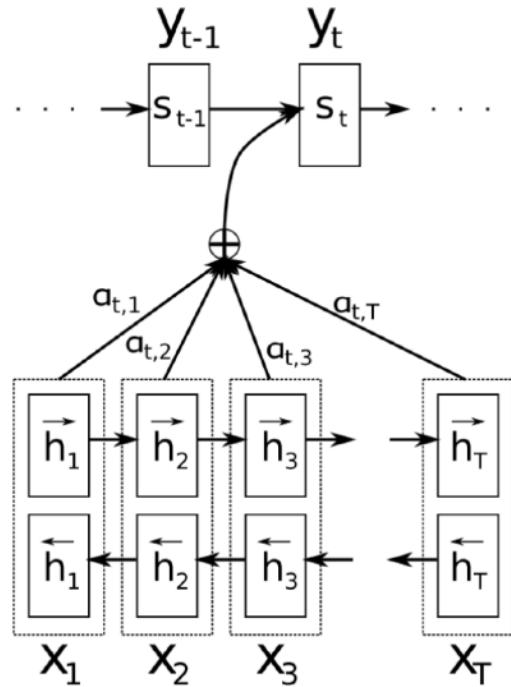
these context vectors are then used to compute a new state sequence  $s_t$ , depending on  $s_{t-1}$  and  $c_t$ .

softmax (=weighted sum)

$$e_{tj} = a(s_{t-1}, h_j), \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^T \exp(e_{tk})}$$

$a(s,h)$  is a learnable feedforward neural network  $\Rightarrow$  learnt during backprop.

# Attention mechanism



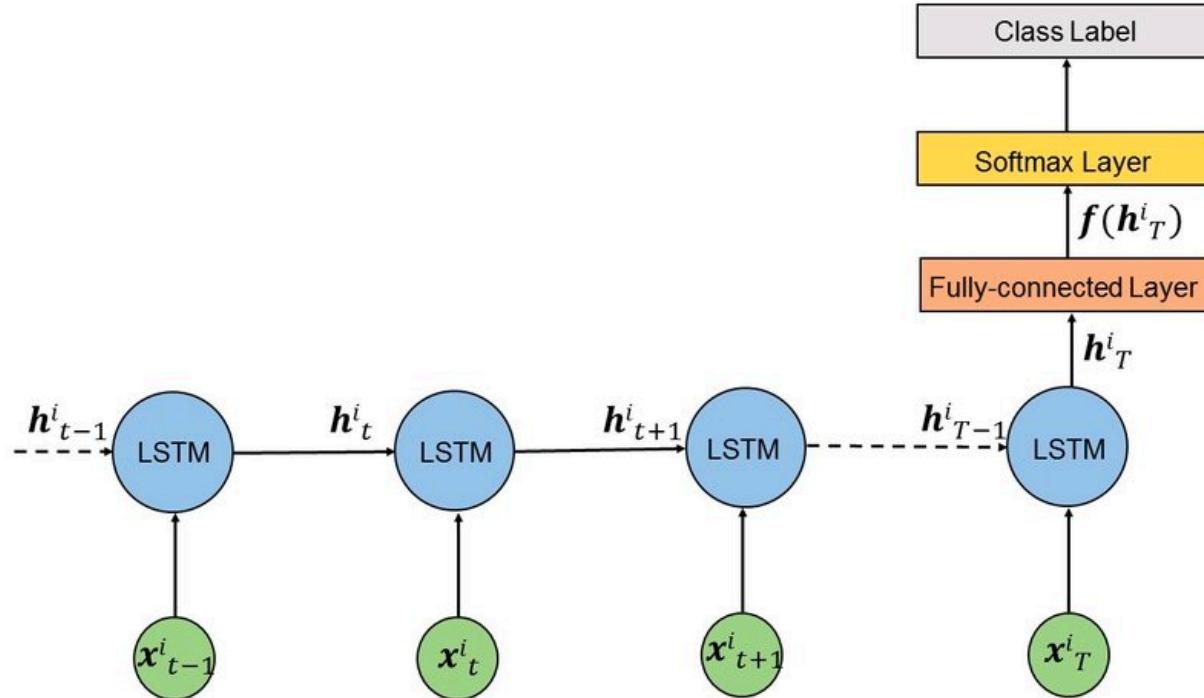
See it as a simple **scalar product**!

If the generated sequence is  
 "Du kannst ein"

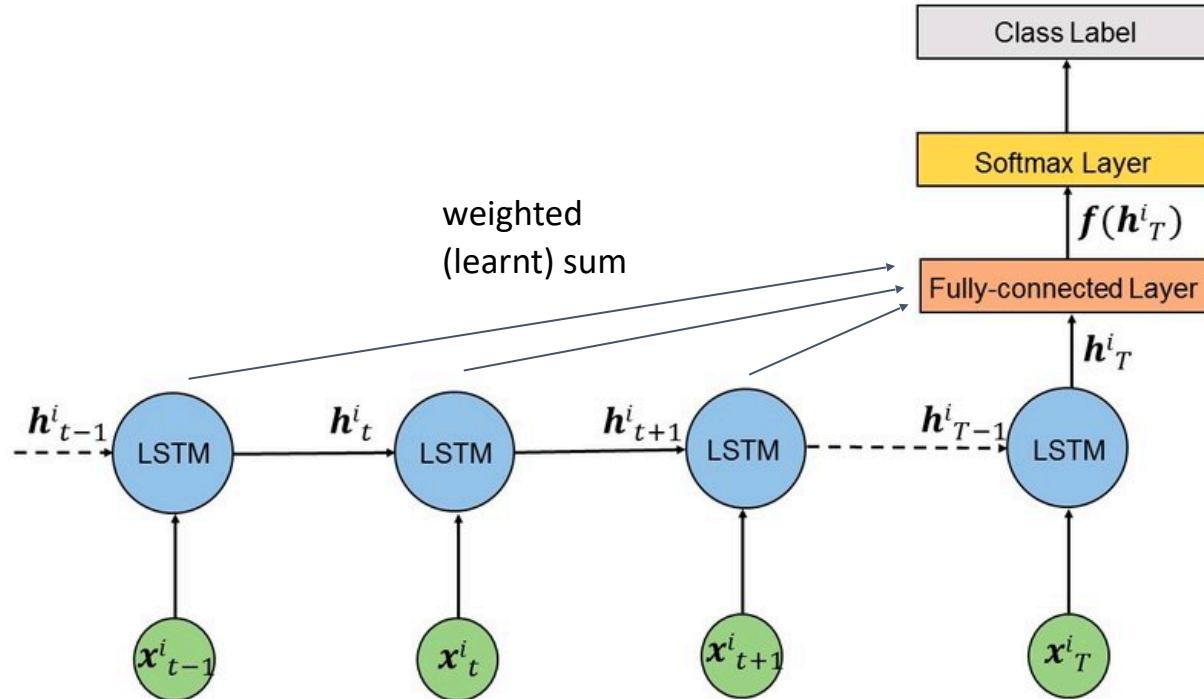
and the input sequence is  
 "Tu peux manger une pomme"

then the network will learn that after a "ein", it has to look at a name and not a verb.

# Attention mechanism for text classification (self-attention)



# Attention mechanism for text classification (self-attention)



# Attention mechanism for text classification (self-attention)

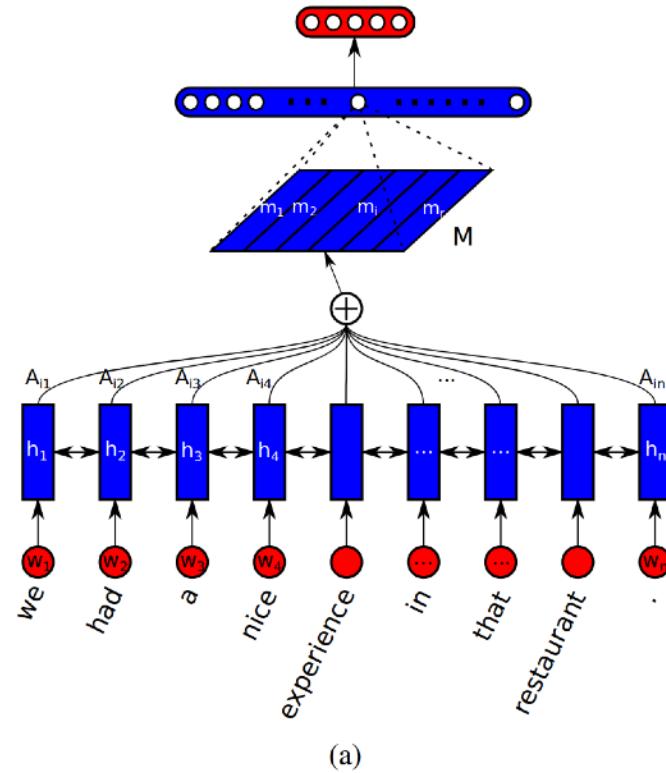
- if I can give this restaurant a 0 will we be just ask our waitress leave because someone with a reservation be wait for our table my father and father-in-law be still finish up their coffee and we have not yet finish our dessert I have never be so humiliated do not go to this restaurant their food be mediocre at best if you want excellent Italian in a small intimate restaurant go to dish on the South Side I will not be go back
- this place suck the food be gross and taste like grease I will never go here again ever sure the entrance look cool and the waiter can be very nice but the food simply be gross taste like cheap 99cent food do not go here the food shot out of me quick then it go in
- everything be pre cook and dry its crazy most Filipino people be used to very cheap ingredient and they do not know quality the food be disgusting I have eat at least 20 different Filipino family home this not even mediocre
- seriously f \*\*\* this place disgust food and shitty service ambience be great if you like dine in a hot cellar engulf in stagnate air truly it be over rate over price and they just under deliver forget try order a drink here it will take forever get and when it finally do arrive you will be ready pass out from heat exhaustion and lack of oxygen how be that a head change you do not even have pay for it I will not disgust you with the detailed review of everything I have try here but make it simple it all suck and after you get the bill you will be walk out with a sore ass save your money and spare your self the disappointment
- i be so angry about my horrible experience at Medusa today my previous visit be amaze 5/5 however my go to out of town and I land an appointment with Stephanie I go in with a picture of roughly what I want and come out look absolutely nothing like it my hair be a horrible ashy blonde not anywhere close to the platinum blonde I request she will not do any of the pop of colour I want and even after specifically tell her I do not like blunt cut my hair have lot of straight edge she do not listen to a single thing I want and when I tell her I be unhappy with the colour she basically tell me I be wrong and I have do it this way no no I do not if I can go from Little Mermaid red to golden blonde in 1 sitting that leave my hair fine I shall be able go from golden blonde to a shade of platinum blonde in 1 sitting thanks for ruin my New Year's with 1 the bad hair job I have ever have

(a) 1 star reviews

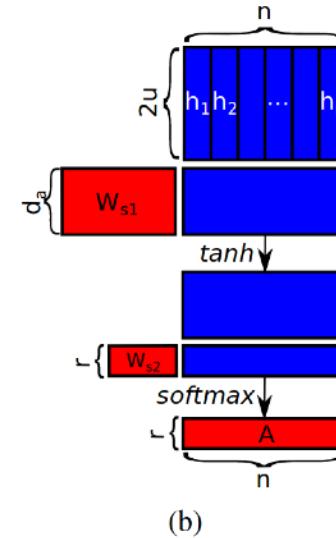
# Attention mechanism for text classification (self-attention)

“Scalar product” between each word of the sequence and a learnt vector that has encoded “sentiment” for sentiment classification task.

awsome



(a)



(b)

# Different types of attention

“Additive attention”

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a^\top \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{s}_j)$$

Seq2seq models

“Multiplicative attention”

$$f_{att}(h_i, s_j) = h_i^\top \mathbf{W}_a s_j$$

Seq2seq models

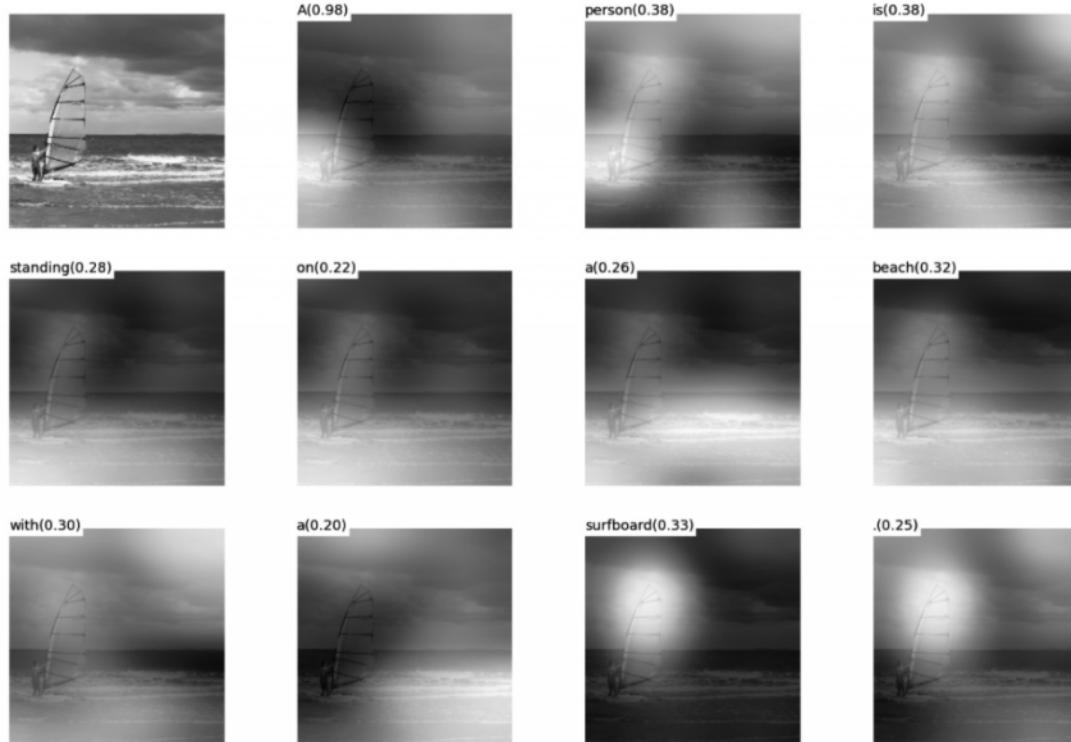
“Self -attention”

$$f_{att}(\mathbf{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{h}_i)$$

Seq2one models

...

# Attention mechanism for image caption generation



(b) A person is standing on a beach with a surfboard.

# Attention mechanism for question answering

by *ent423* ,*ent261* correspondent updated 9:49 pm et ,thu march 19 ,2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* ,*ent85* ,near *ent312* ,a *ent119* official told *ent261* on wednesday .he was identified thursday as special warfare operator 3rd class *ent23* ,29 ,of *ent187* ,*ent265* .`` *ent23* distinguished himself consistently throughout his career .he was the epitome of the quiet professional in all facets of his life ,and he leaves an inspiring legacy of natural tenacity and focused

...

*ent119* identifies deceased sailor as **X** ,who leaves behind a wife

by *ent270* ,*ent223* updated 9:35 am et ,mon march 2 ,2015 (*ent223*) *ent63* went familial for fall at its fashion show in *ent231* on sunday ,dedicating its collection to `` mamma '' with nary a pair of `` mom jeans '' in sight .*ent164* and *ent21* ,who are behind the *ent196* brand ,sent models down the runway in decidedly feminine dresses and skirts adorned with roses ,lace and even embroidered doodles by the designers ' own nieces and nephews .many of the looks featured saccharine needlework phrases like `` i love you ,

...

**X** dedicated their fall fashion show to moms

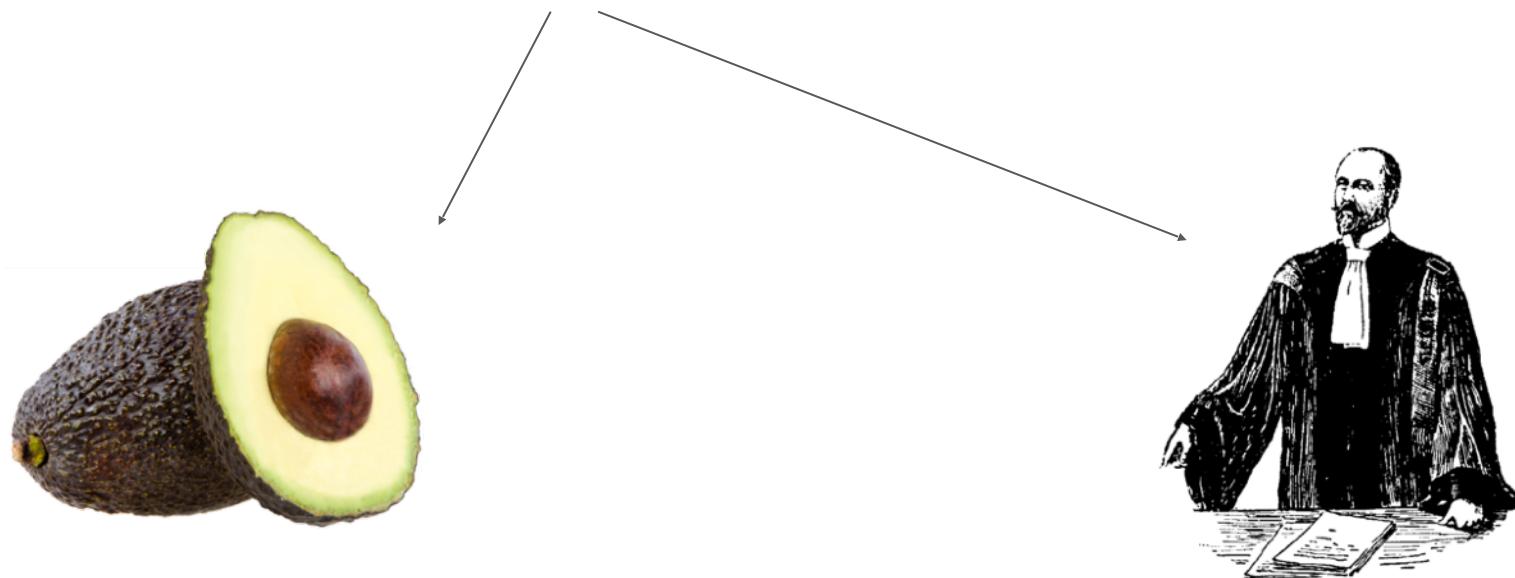
# Contextualized Word Representations



# ELMo - Deep Contextualized Word Representations (2018)

Initial word embedding models suffers from a disadvantage: **context insensitivity**. They behave like hash tables and are not able to perform **word disambiguation**.

## L'avocat de la défense

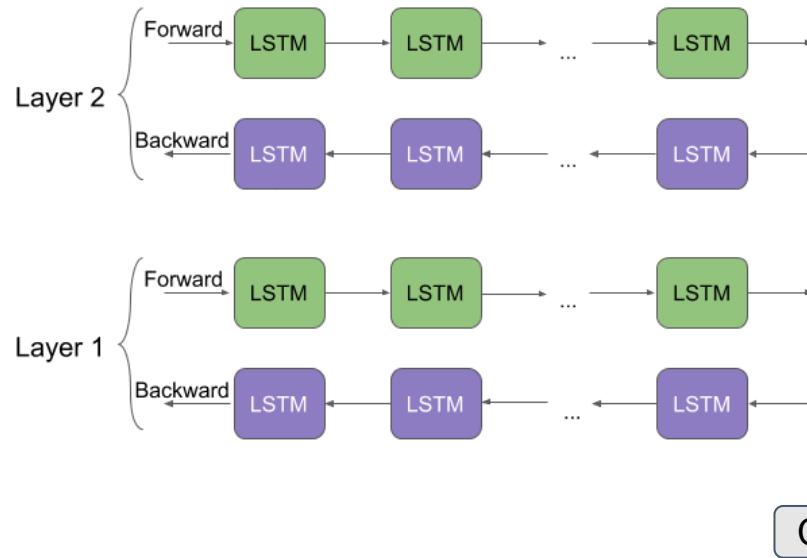


# ELMo - Deep Contextualized Word Representations (2018)

Initial word embedding models suffers from a disadvantage: **context insensitivity**. They behave like hash tables and are not able to perform **word disambiguation**.



# ELMo - Deep Contextualized Word Representations (2018)



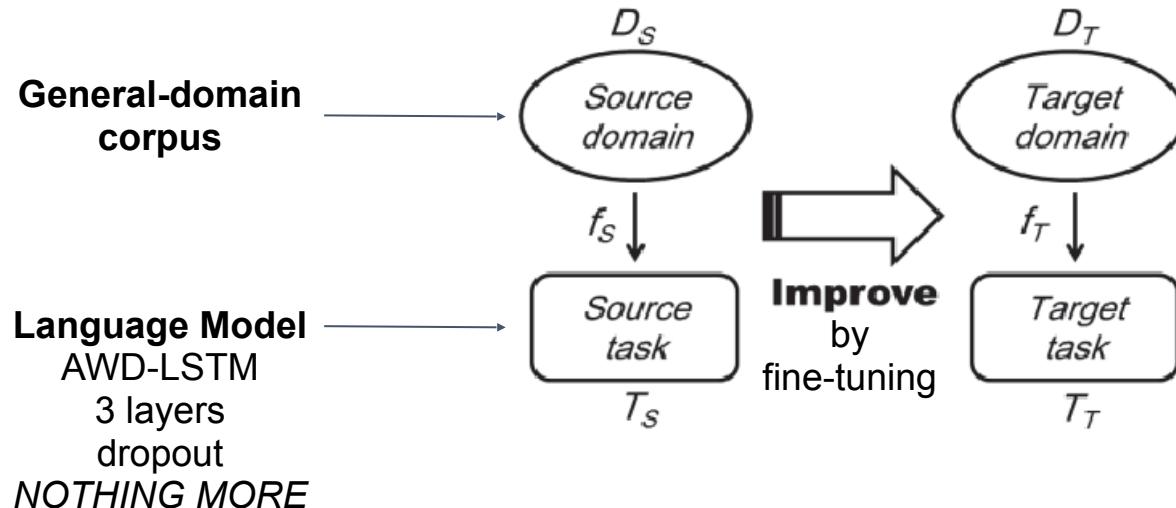
*Exactly what we did  
with the hidden layers  
of VGG!*

CNN

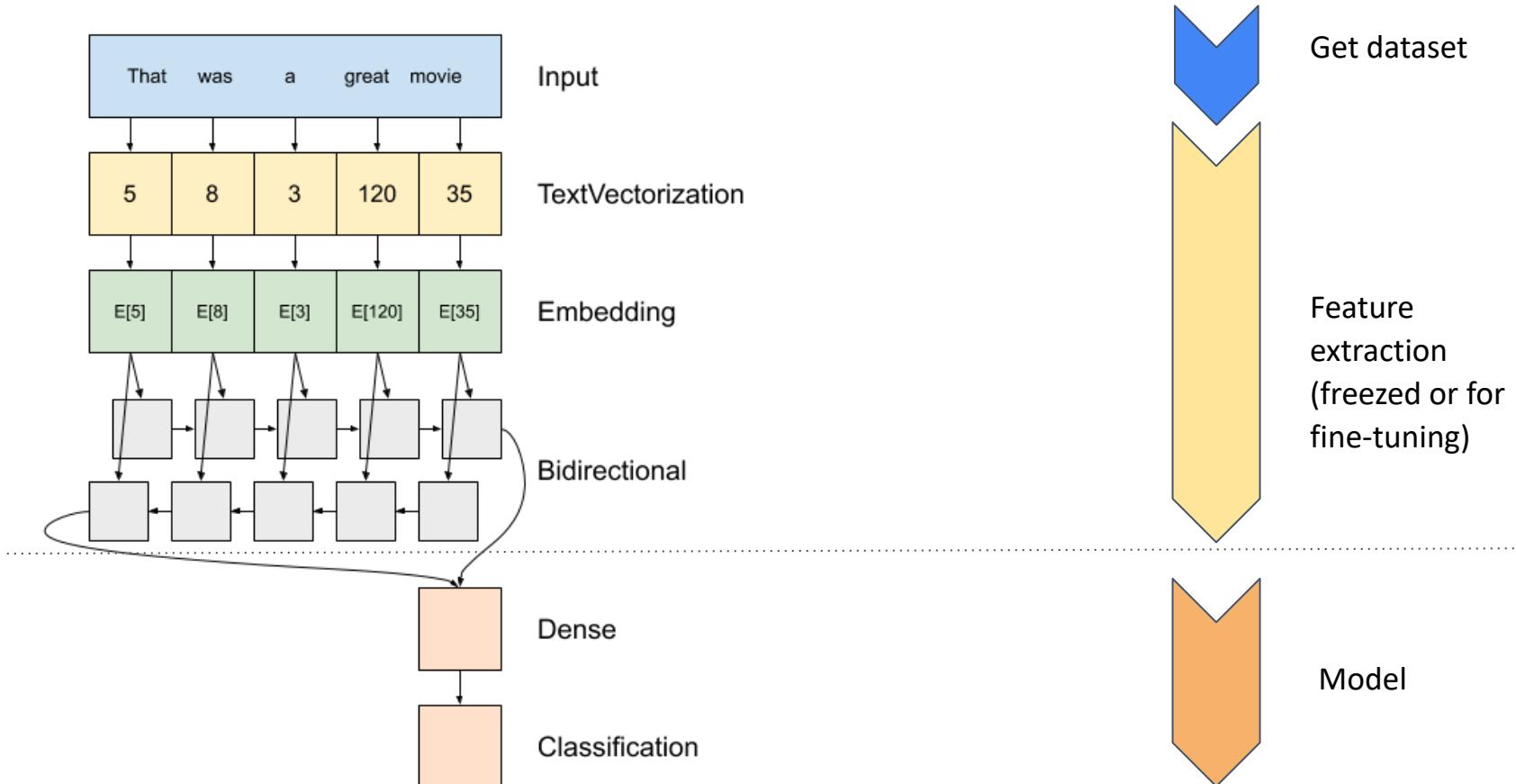
# Transfer learning In NLP



## Universal Language Model Fine-Tuning



# ULMFiT - Universal Language Model Fine-Tuning (2018)





# Transformers



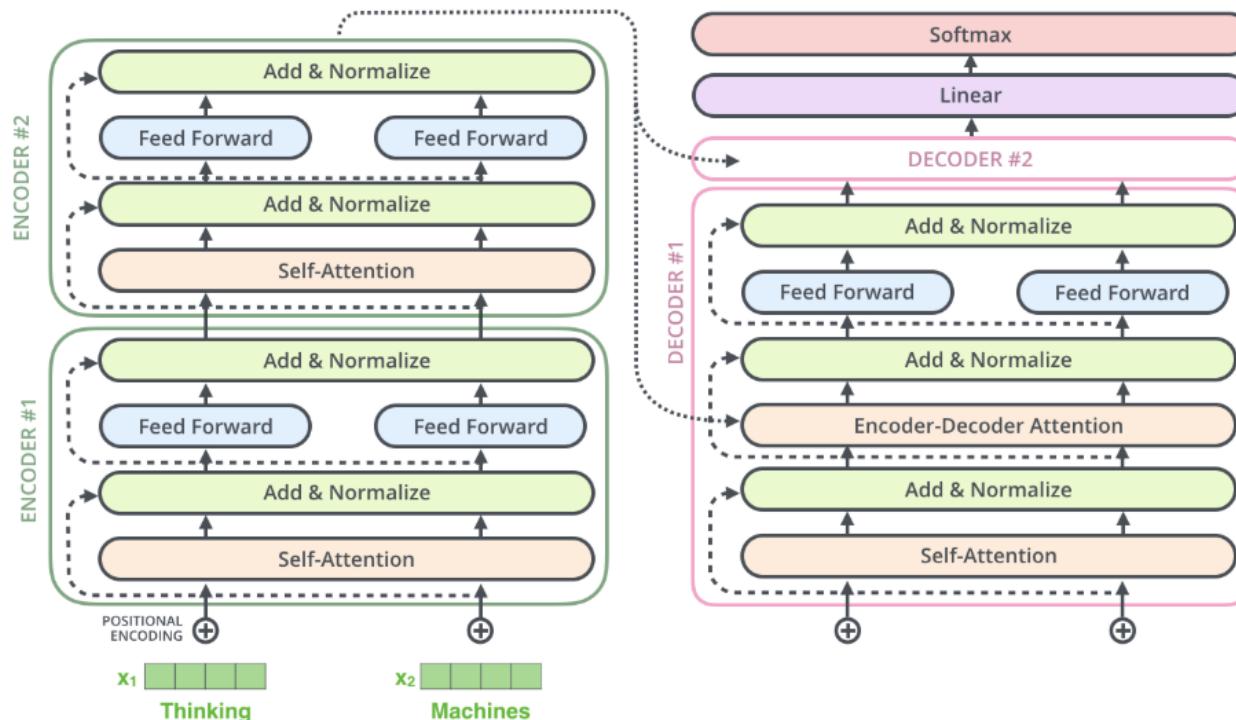
# Why Transformers?

---

- Used for **machine translation** first
- **Lack of observability** within the full processed sequence implying a performance ceiling for the machine translation context
- This drawback has been explored and the best solution retained in the academic community was **bi-contextual and attention** models
- Recurrent architectures face **poor training speed** because of their sequential processing

-> All these drawbacks motivated the development of “**The Transformer**” architecture by **Google**

# The Transformers architecture



# The Transformers architecture

A lot of tricks...

- Layer normalization ([Ba et al, 2016](#))
- “Add” layer : add some residual (introduced on [ResNet](#) for vanishing gradient problem)

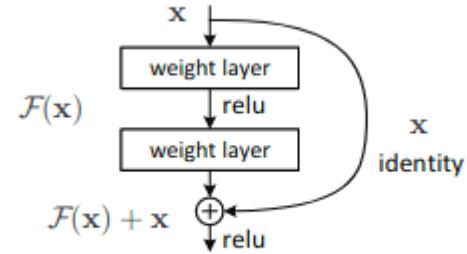
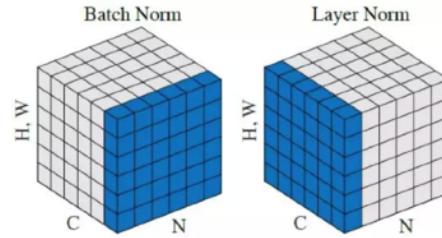
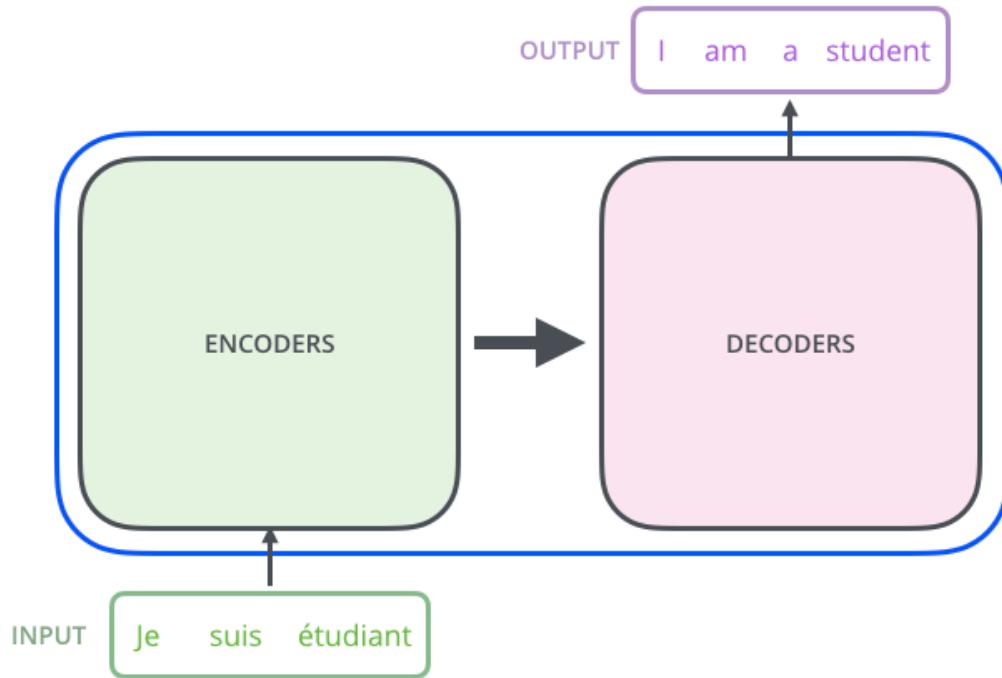


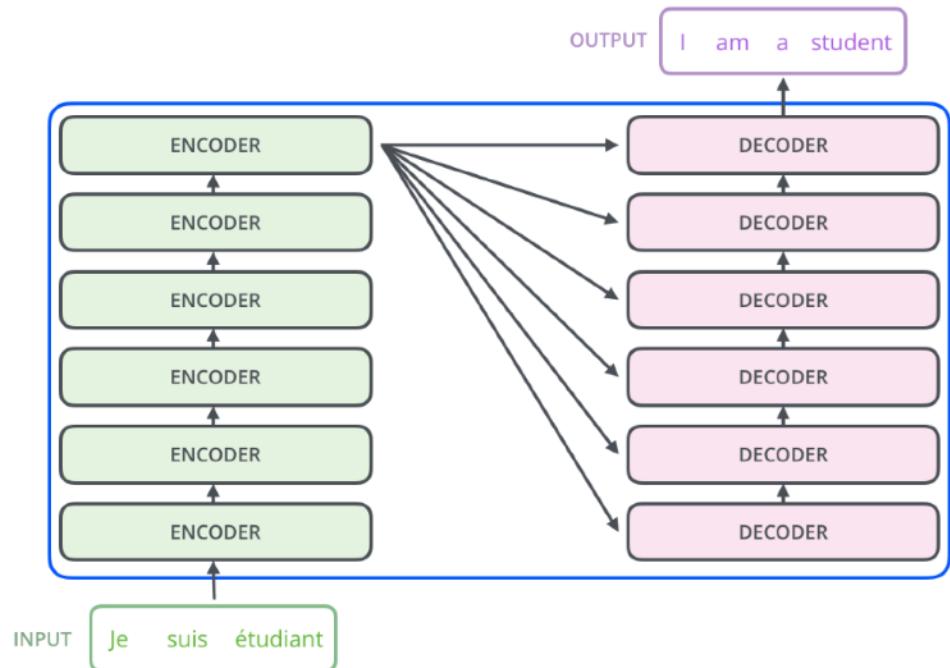
Figure 2. Residual learning: a building block.

# A classical seq2seq model...



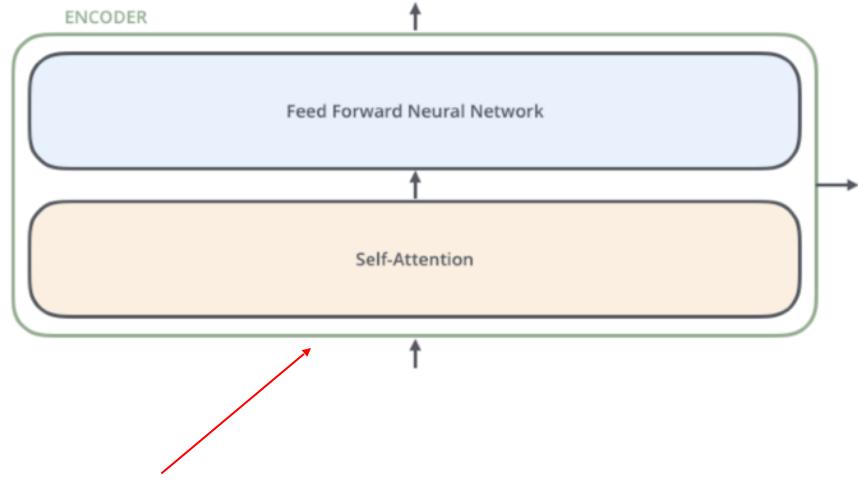
# ... which is stacked

- In their paper, Google uses **2 stacks** (encoder/decoder) of **6 layers**
- The architecture is not recurrent so **full input/output sequences** are sent into the layers
- The encoders have all the **same structure** but different **trained weights**
- The **outputs** of the **last encoder layer** are kept constant and fed to **all decoder layers**



# Autoencoder layer abstraction

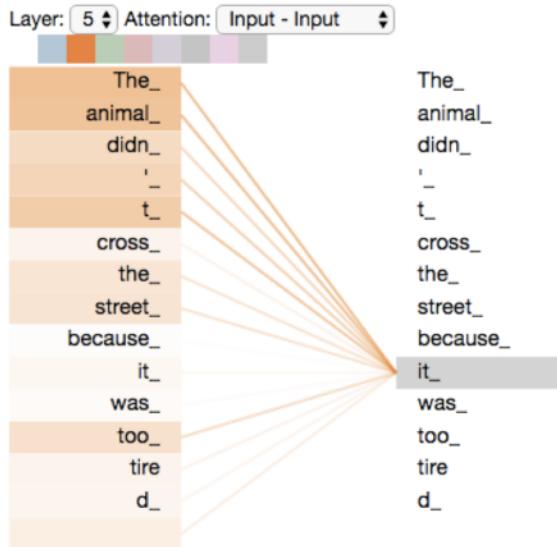
- An **encoder** layer is composed of **2 sub-layers**
- The **self-attention layer** is responsible for **connecting input sequence tokens** semantically linked. This is the **core** building **block** replacing the **internal memory** stored in a **recurrent architecture**



replace the recurrence of RNNs

# Self-attention zoom

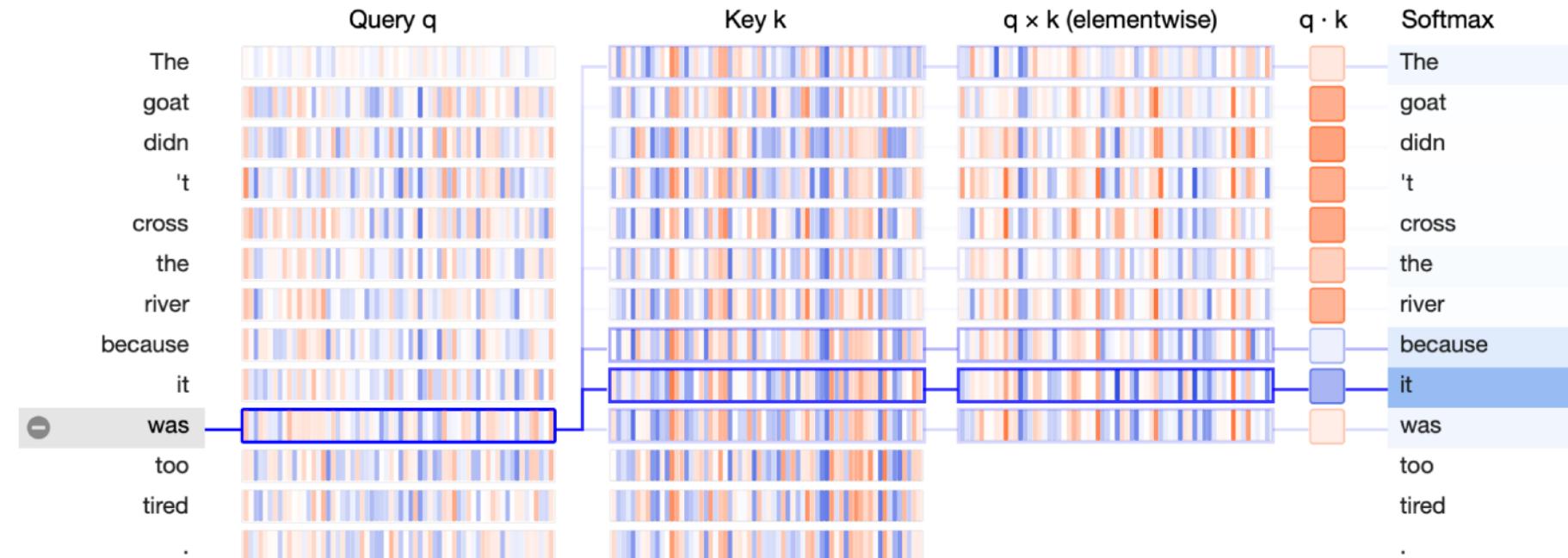
The **encoder** has the capability to **focus on others tokens** when it is **encoding** a specific **one**



As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".

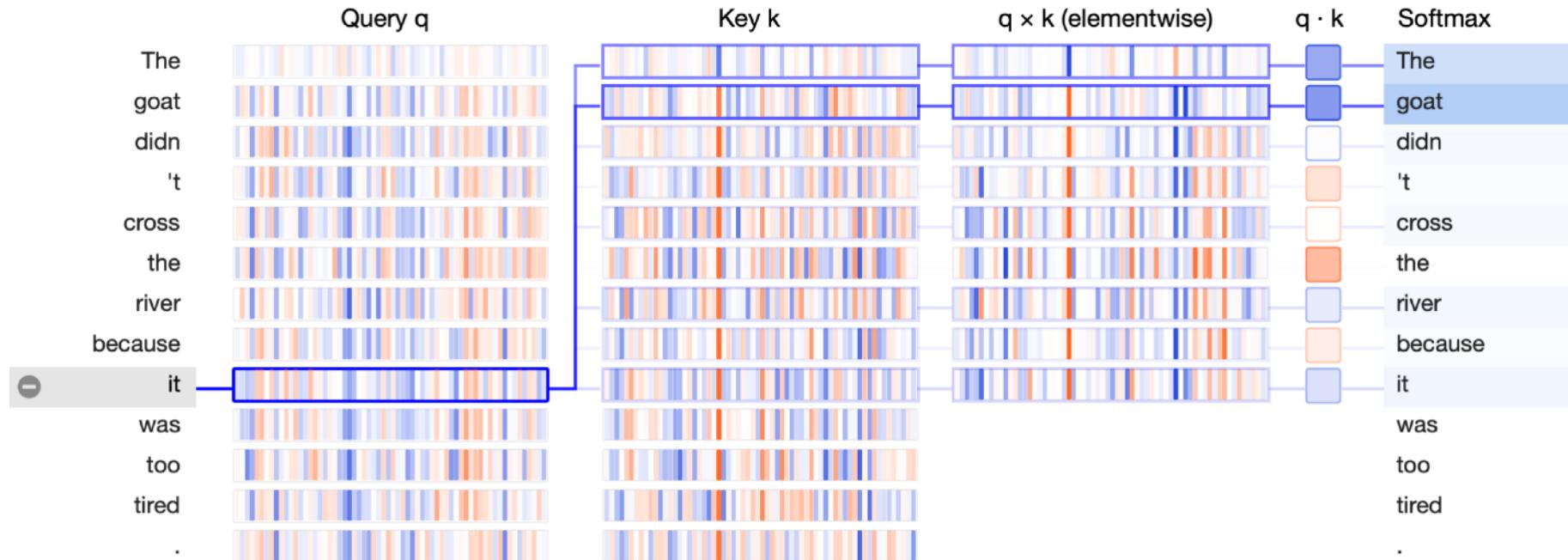
# Self-attention zoom

Layer: 1 Head: 0



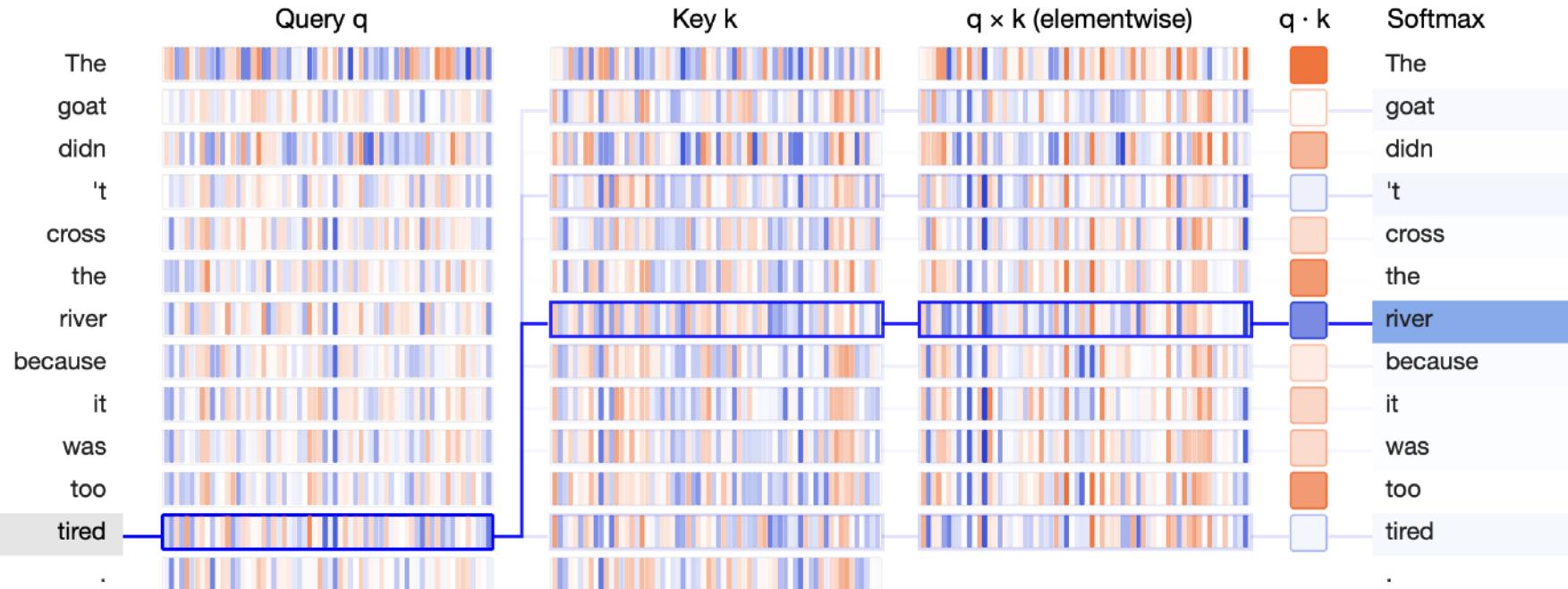
# Self-attention zoom

Layer: 9 Head: 0



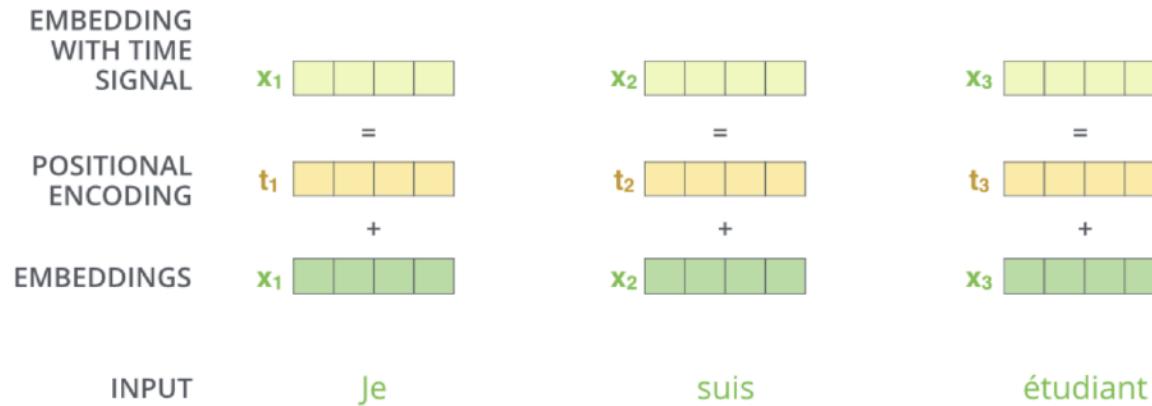
# Self-attention zoom

Layer: 11 Head: 8



# Positional encoding

- You may have noticed that the **encoder** has **no information** about the **order** in which the tokens appeared in the **input sequence**
- So if we do nothing, our model play in a **Bag of Word** context
- We need to **give** the input **sequence order** to the **model** if we want a superior **predictive power**
- **Google** used a **technique** name **positional encoding** which is simply **adding** an **ordering signal**



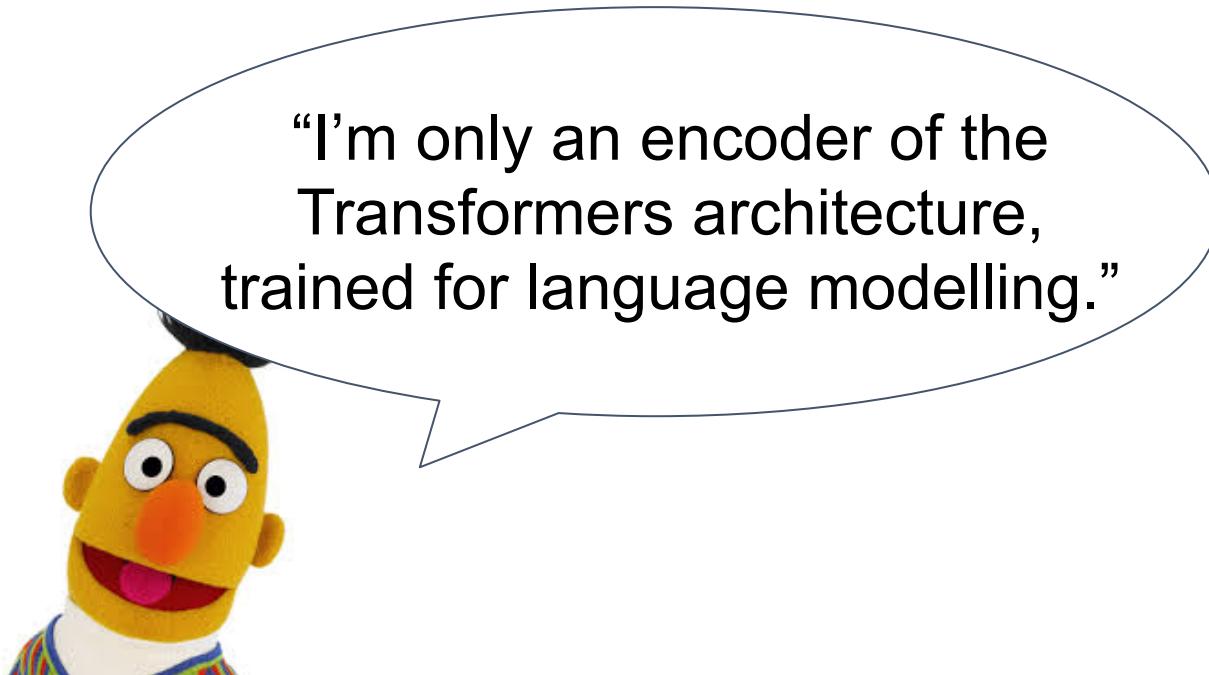
# BERT

Bidirectional Encoder Representations  
from Transformers



[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)

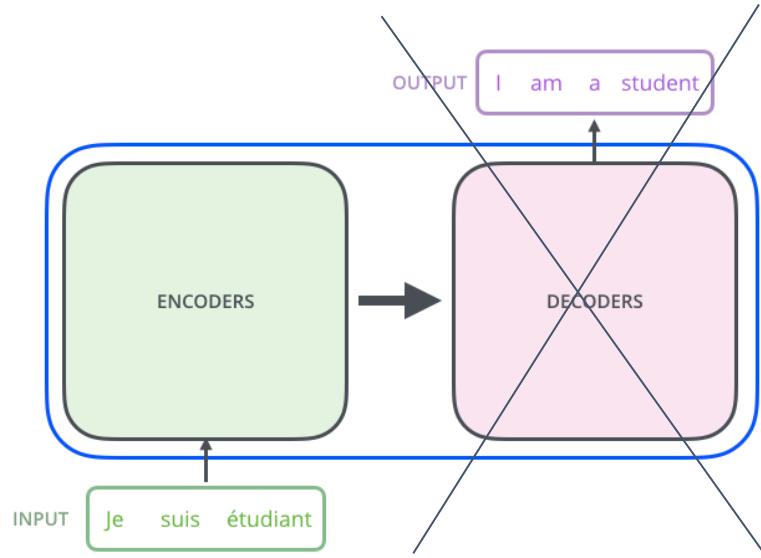
# What's new with BERT ?



“I’m only an encoder of the  
Transformers architecture,  
trained for language modelling.”

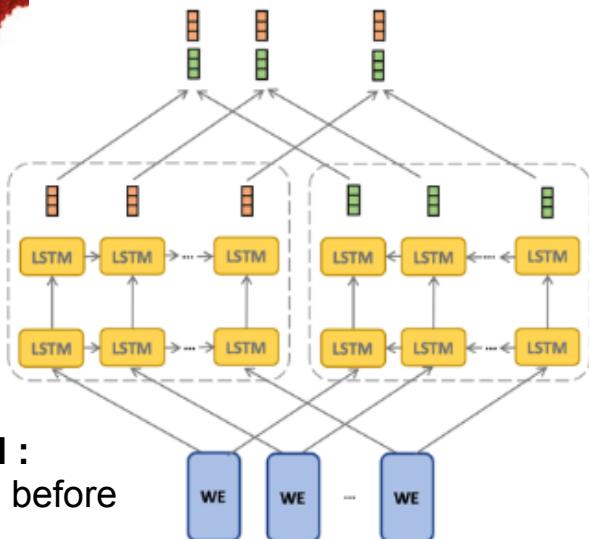
# What's new with BERT ?

- BERT is a **language model**
- BERT is an **encoder only**, trained on an auto-supervised way



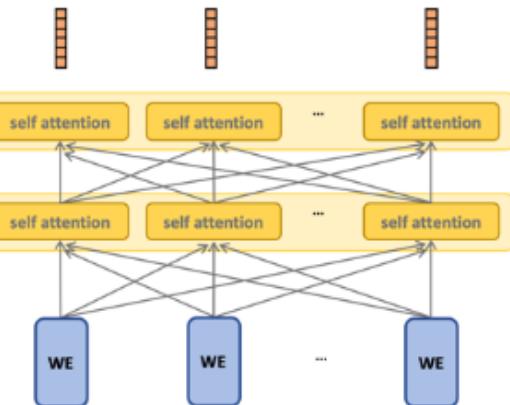
# BERT - bi-directionality

Right and Left contexts  
taken **independently**



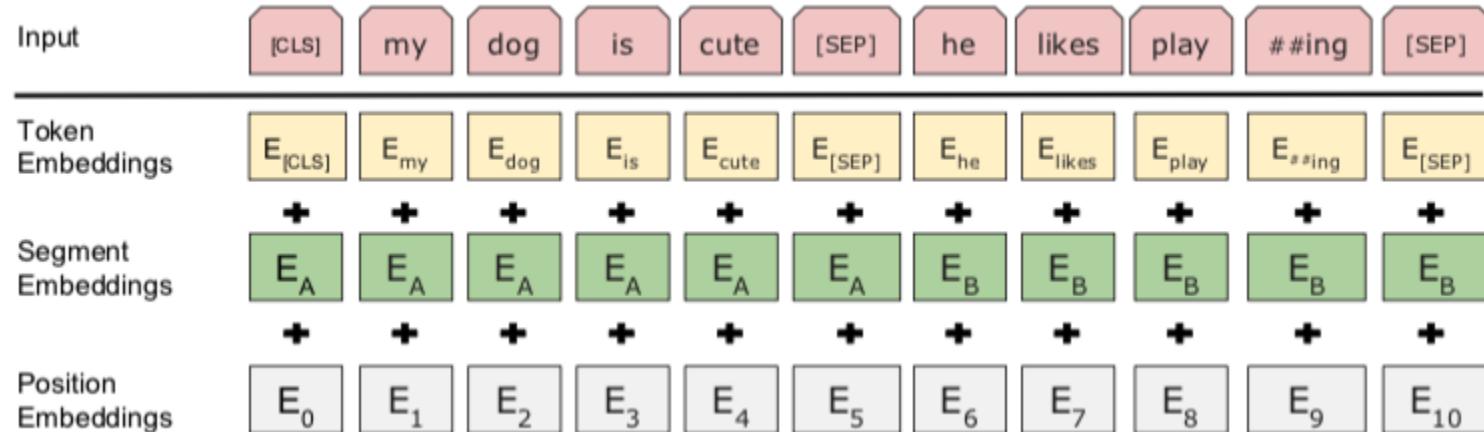
**Bi-LSTM :**  
The reference before

**Deep bidirectional model**



**Transformers with self-attention :**  
The new reference

# BERT - inputs



- **Word embeddings**
- **Position embedding**
- **Sentence** embedding in order to treat sentence-level and token-level task ([CLS])
- **Segment** embedding for the next sentence prediction task (autosupervised, see next slide)

# BERT - Pre-training

- **Masked LM (MLM)**

15% of the words are masked

« Un exemple de **texte** partiellement masqué où il s'agit de **deviner** les termes manquants. »

« Un exemple de **[MASK]** partiellement masqué où il s'agit de **[MASK]** les termes manquants. »

- **Next sentence prediction**

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

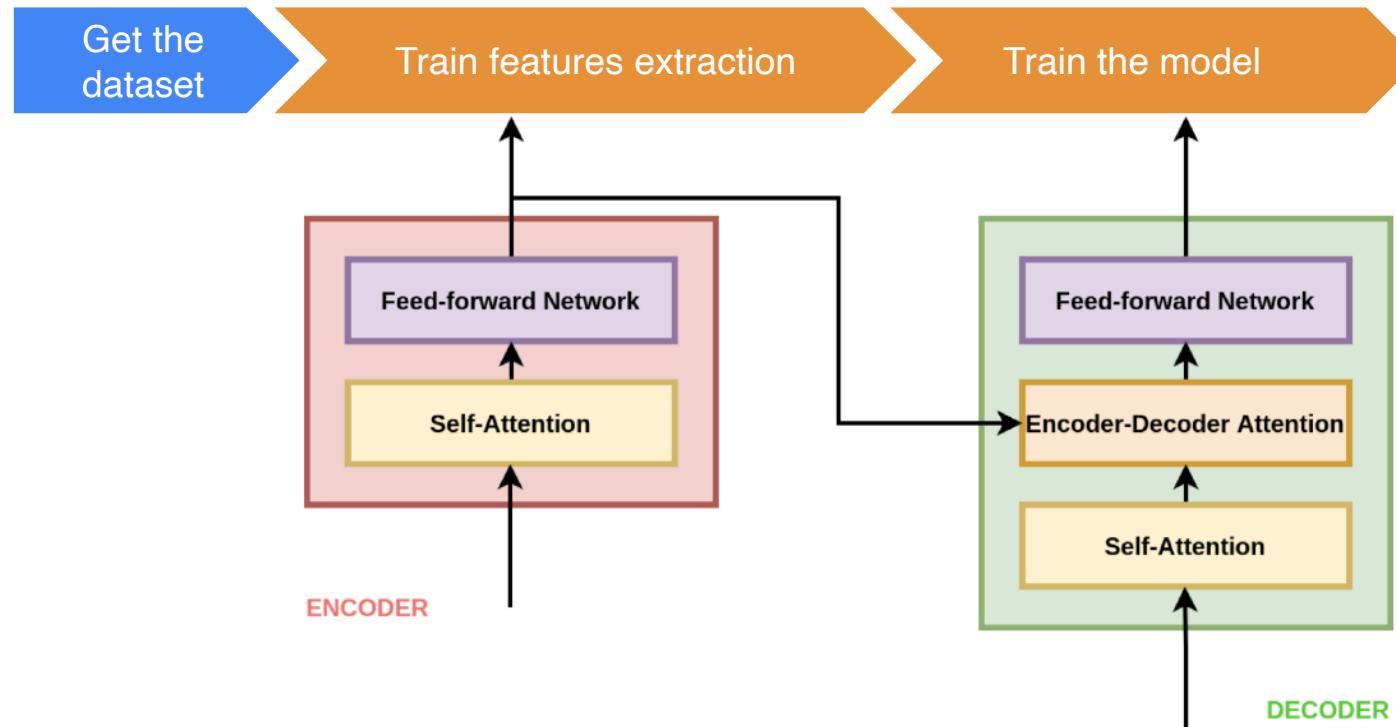
Label = **IsNext**

Input = [CLS] the man [MASK] to the store [SEP]

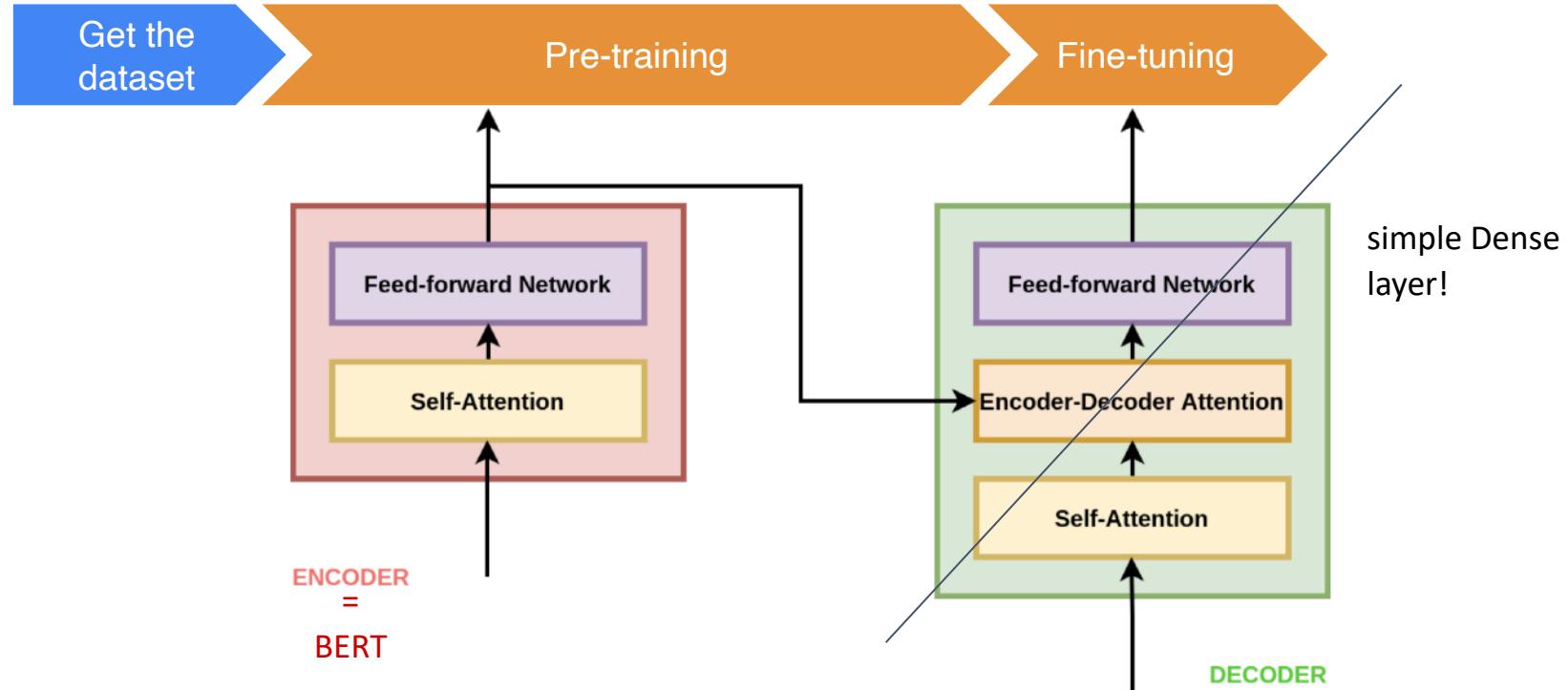
penguin [MASK] are flight ##less birds [SEP]

Label = **NotNext**

# BERT - encoder only, then plug your decoder!

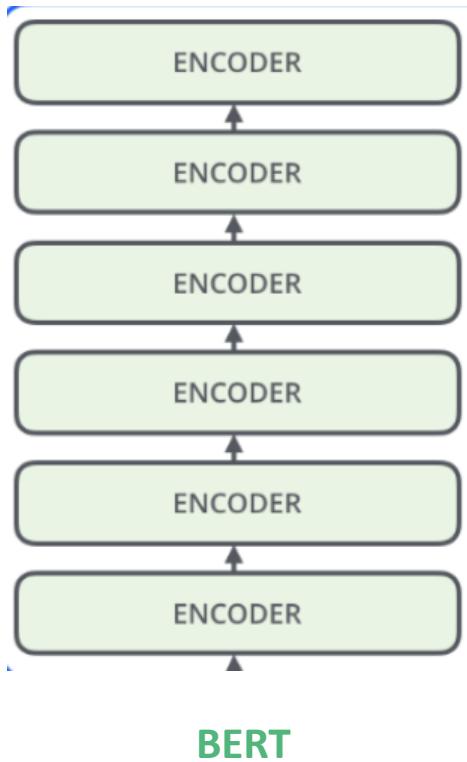


# BERT - encoder only, then plug your decoder!



# BERT - Fine-tuning

For fine-tuning, you can  
freeze the encoder, or  
unfreeze only the last hidden  
states.



# Transformers tutorial

---

[https://colab.research.google.com/github/pytorch/pytorch.github.io/blob/master/assets/hub/huggingface\\_pytorch-transformers.ipynb](https://colab.research.google.com/github/pytorch/pytorch.github.io/blob/master/assets/hub/huggingface_pytorch-transformers.ipynb)

Let's try camemBERT for French! <https://huggingface.co/camembert-base>

# Transformers for ImageNet too!

---

## Scaling Vision Transformers

CVPR 2022 · Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, Lucas Beyer ·  Edit social preview

Attention-based neural networks such as the Vision Transformer (ViT) have recently attained state-of-the-art results on many computer vision benchmarks. Scale is a primary ingredient in attaining excellent results, therefore, understanding a model's scaling properties is a key to designing future generations effectively. While the laws for scaling Transformer language models have been studied, it is unknown how Vision Transformers scale. To address this, we scale ViT models and data, both up and down, and characterize the relationships between error rate, data, and compute. Along the way, we refine the architecture and training of ViT, reducing memory consumption and increasing accuracy of the resulting models. As a result, we successfully train a ViT model with two billion parameters, which attains a new state-of-the-art on ImageNet of 90.45% top-1 accuracy. The model also performs well for few-shot transfer, for example, reaching 84.86% top-1 accuracy on ImageNet with only 10 examples per class.



# QUIZZ TIME

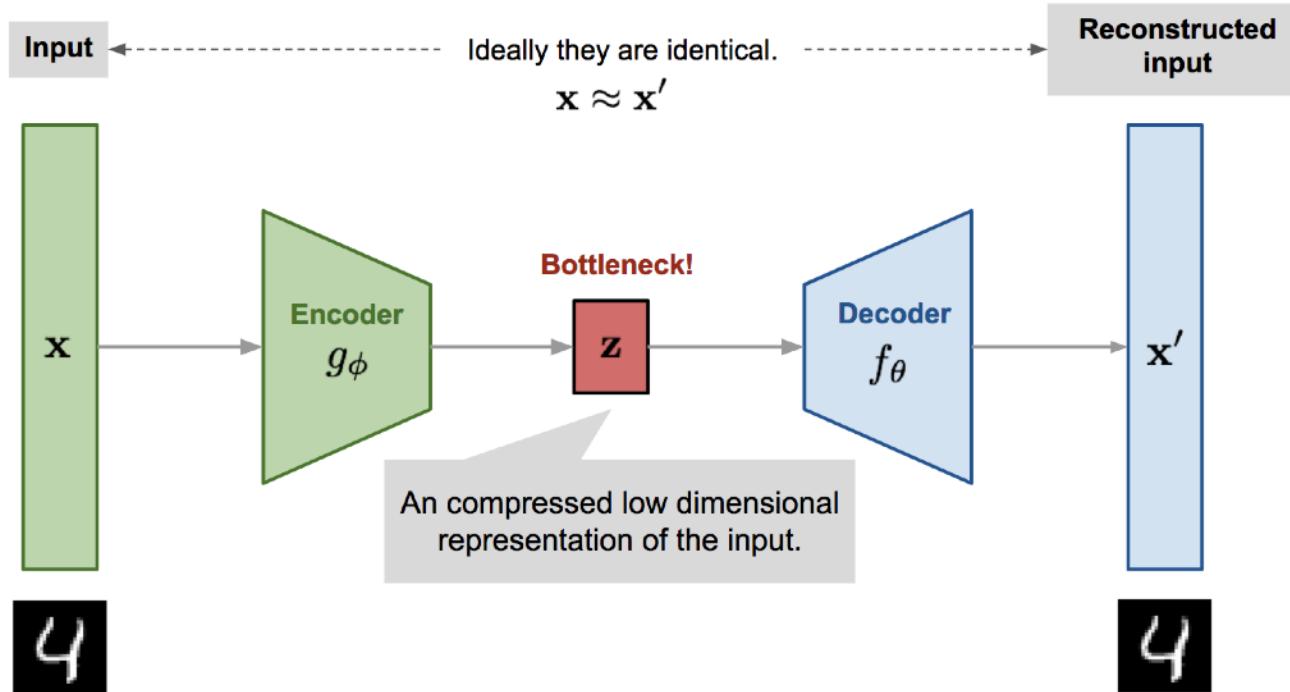
## On RNNs!



# DEEP LEARNING

## Autoencoders

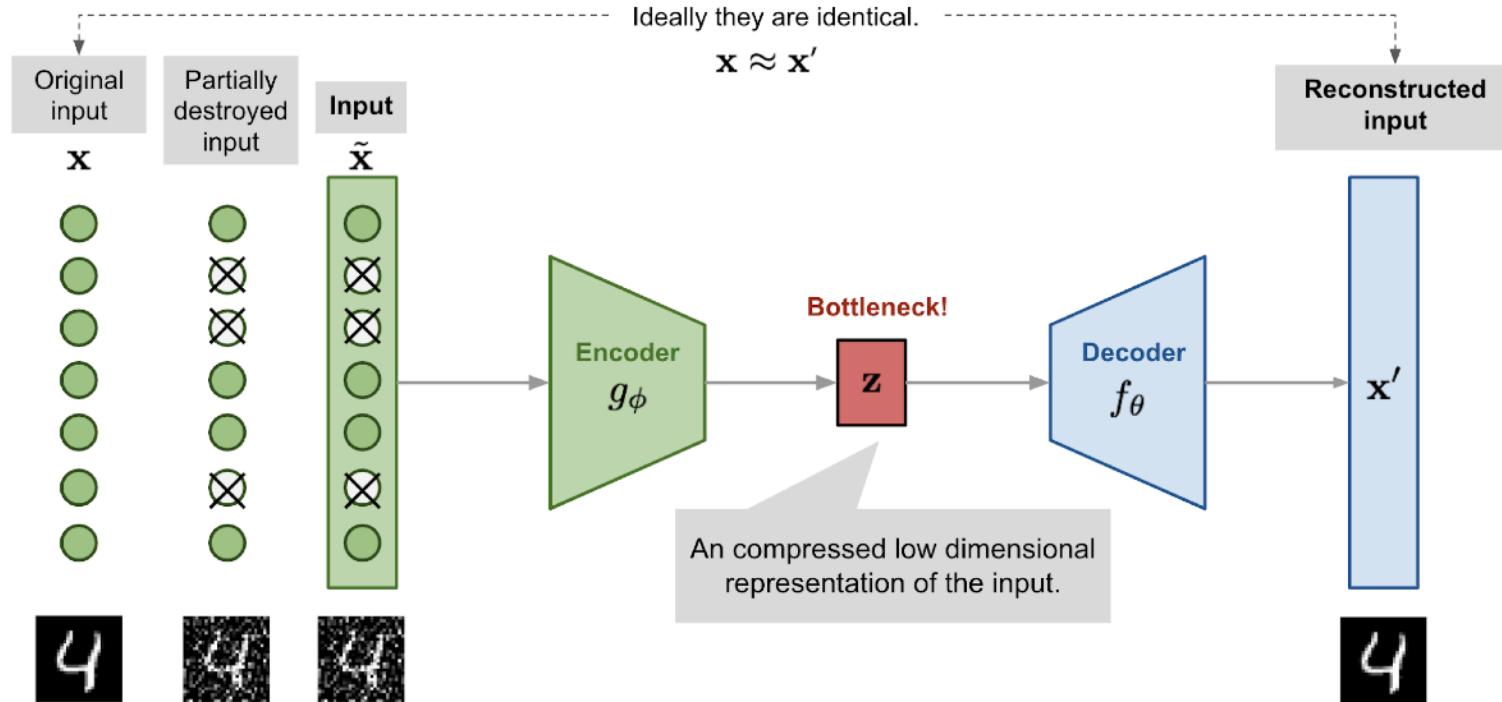
# Autoencoders are unsupervised – learn "compressed" representation



Loss: mean squared error

<https://lilianweng.github.io/posts/2018-08-12-vae/>

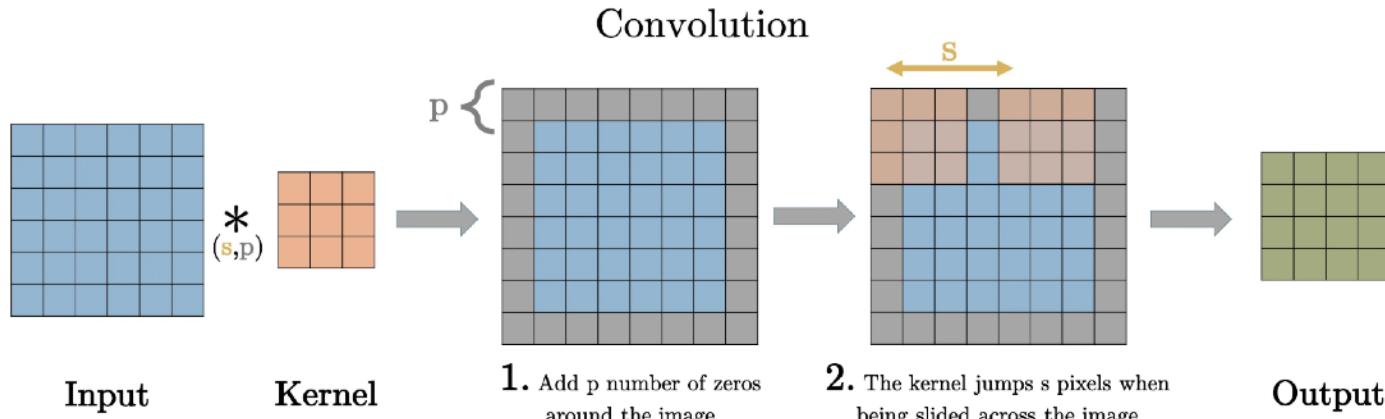
# Denoising autoencoders – learn to denoise



Prefigures the diffusion models (StableDiffusion, Dall-E, etc.)

<https://lilianweng.github.io/posts/2018-08-12-vae/>

# Convolutional autoencoders – recall what a convolution is

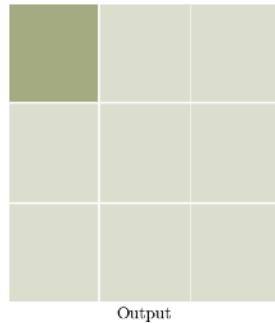
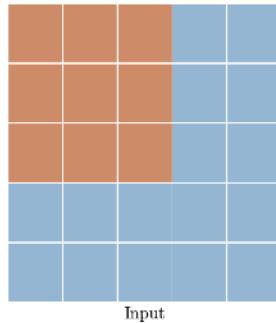


$$o = \frac{i + 2p - k}{s} + 1$$

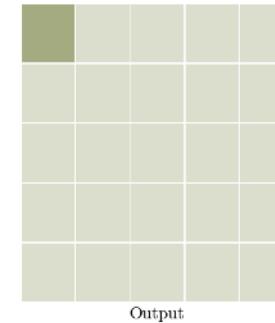
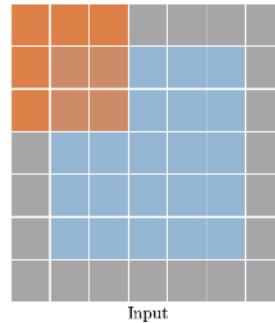
size of the input (**i**)  
 kernel (**k**)  
 padding (**p**)  
 stride (**s**)

# Convolutional autoencoders – recall what a convolution is

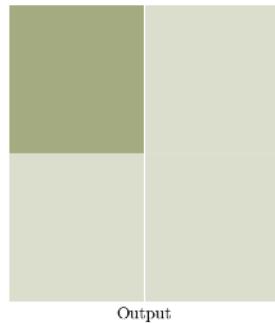
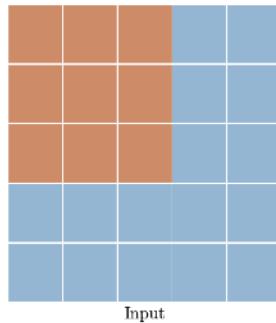
Type: conv - Stride: 1 Padding: 0



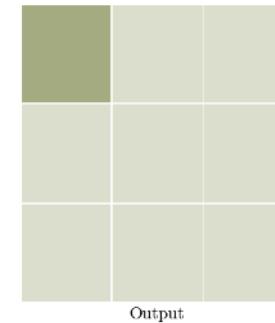
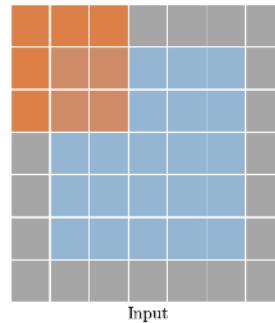
Type: conv - Stride: 1 Padding: 1



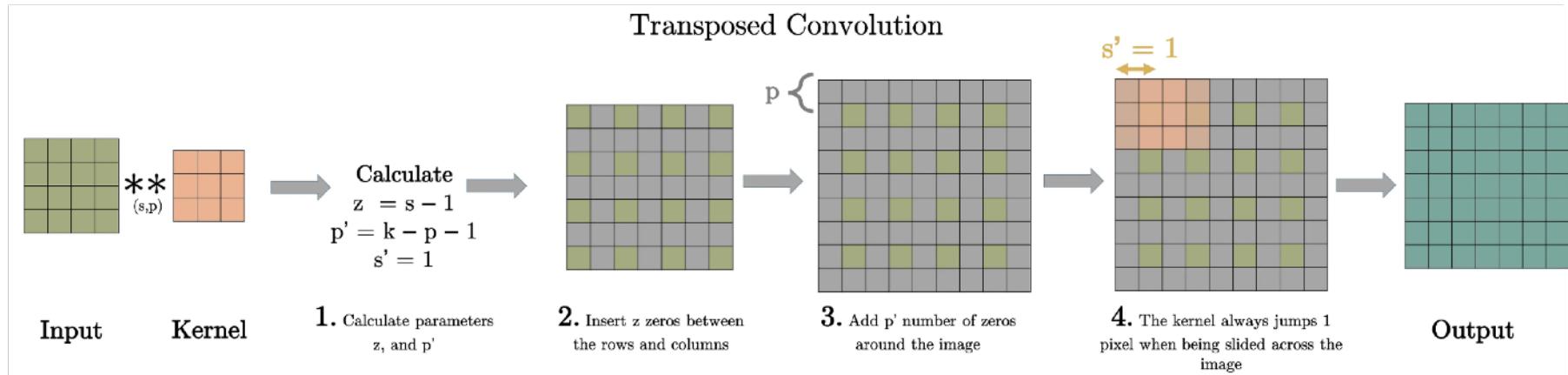
Type: conv - Stride: 2 Padding: 0



Type: conv - Stride: 2 Padding: 1



# Convolutional autoencoders: how to reconstruct the image?

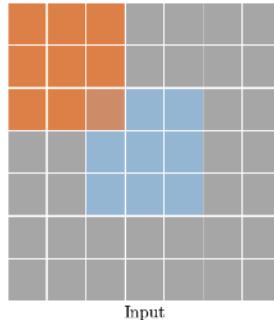


$$o = (i - 1) \times s + k - 2p$$

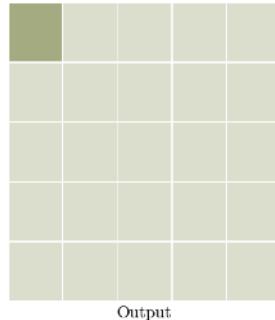
size of the input (**i**)  
 kernel (**k**)  
 padding (**p**)  
 stride (**s**)

# Convolutional autoencoders – transposed convolution

Type: transposed'conv - Stride: 1 Padding: 0

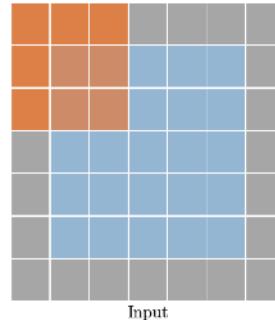


Input

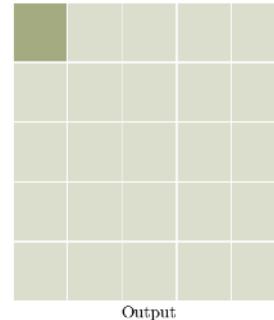


Output

Type: transposed'conv - Stride: 1 Padding: 1

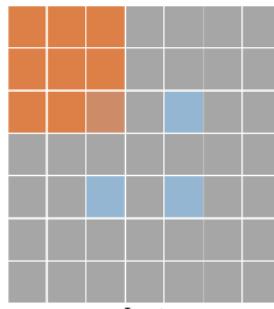


Input

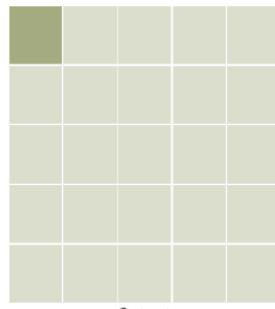


Output

Type: transposed'conv - Stride: 2 Padding: 0

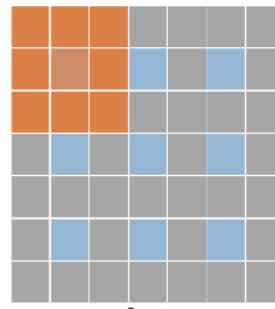


Input

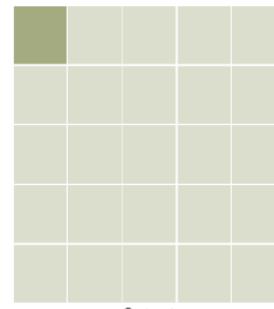


Output

Type: transposed'conv - Stride: 2 Padding: 1



Input



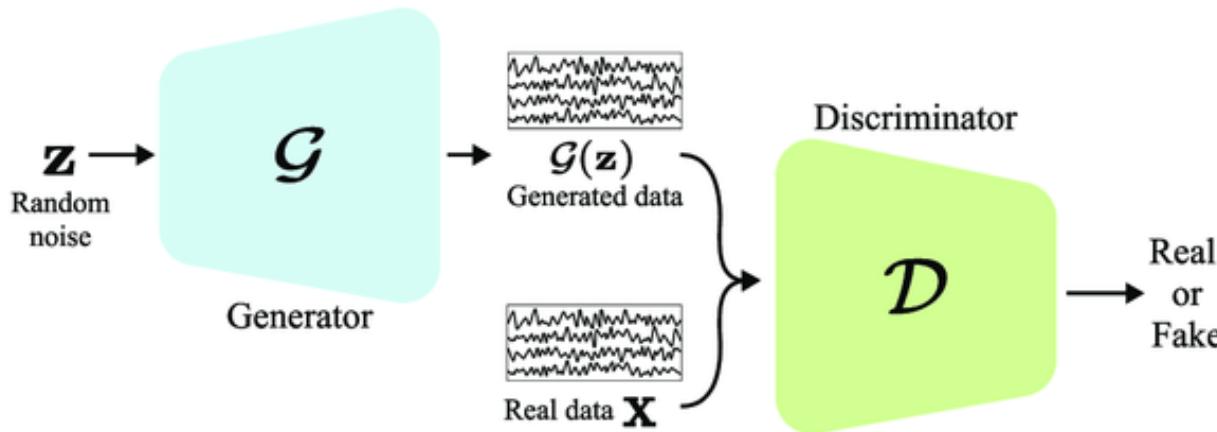
Output

# DEEP LEARNING

## Generative Adversarial Networks (GAN)

# Generative Adversarial Networks – two competing networks.

Goodfellow et al., 2014  
<https://arxiv.org/pdf/1406.2661.pdf>



A good generator: generates fakes images indistinguishable from real ones.

A good discriminator: distinguishes between real and fake images.

# Generative Adversarial Networks – two competing networks.

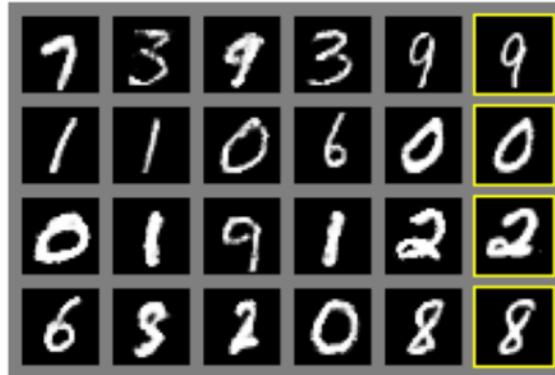
Goodfellow et al., 2014  
<https://arxiv.org/pdf/1406.2661.pdf>

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

A good generator: generates fakes images indistinguishable from real ones.

A good discriminator: distinguishes between real and fake images.

# Generative Adversarial Networks – two competing networks.



a)



b)



c)

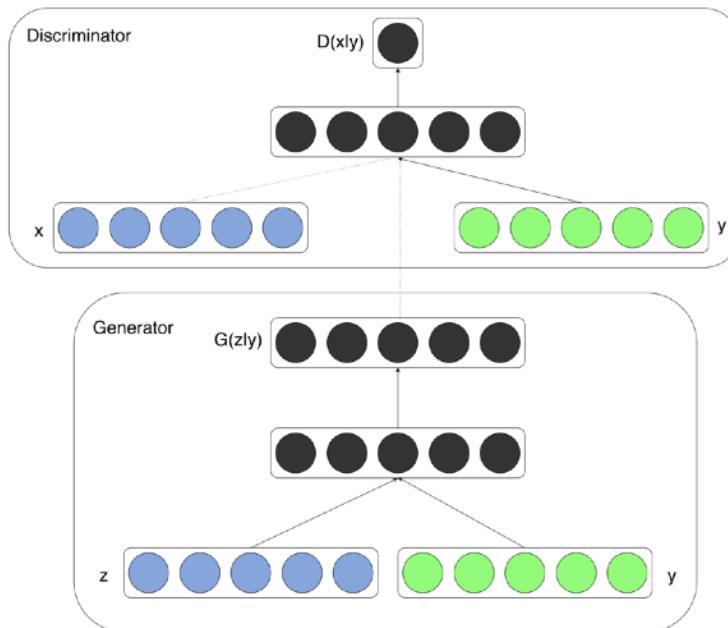


d)

Goodfellow et al., 2014  
<https://arxiv.org/pdf/1406.2661.pdf>

# Conditional Generative Adversarial Networks (CGAN)

Same idea, but  $x$  and  $z$  are conditioned on some label  $y$ .



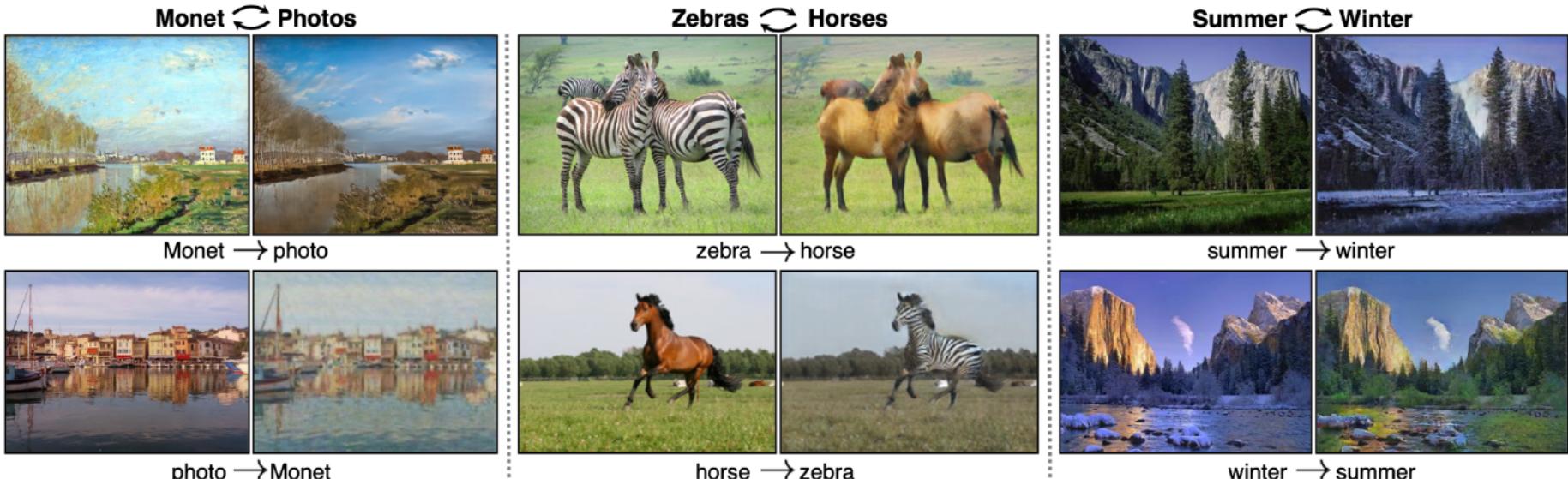
A lot of applications  
(generate realistic objects):

- Face generation from labels.
- Text-to-image
- Inpainting
- Text-to-speech

# Cycle Generative Adversarial Networks (CycleGAN)

<https://arxiv.org/pdf/1703.10593.pdf>

Objective: domain adaptation

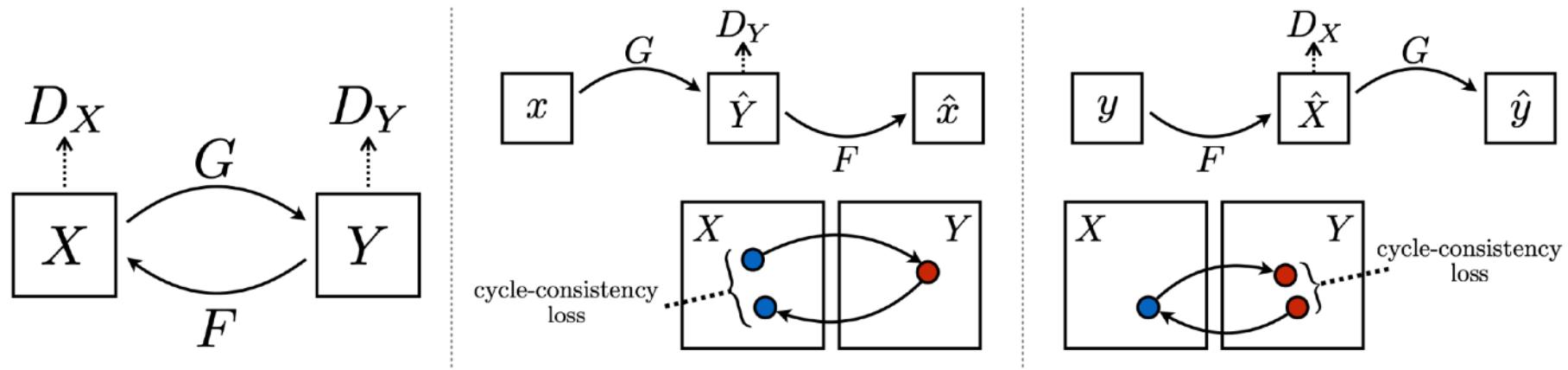


One example: image style transfer

Can also be used to improve transfer learning when labels come from a different domain

# Cycle Generative Adversarial Networks (CycleGAN)

Same idea as GAN, but replace the noise  $z$  by another "domain".



A good generator  $D_X$ : generates images that looks like images from domain X.

A good generator  $D_Y$ : generates images that looks like images from domain Y.

A good discriminator: distinguishes between domains X and Y.

# Cycle Generative Adversarial Networks (CycleGAN)

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = & \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] \\ & + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = & \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] \\ & + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1].\end{aligned}$$

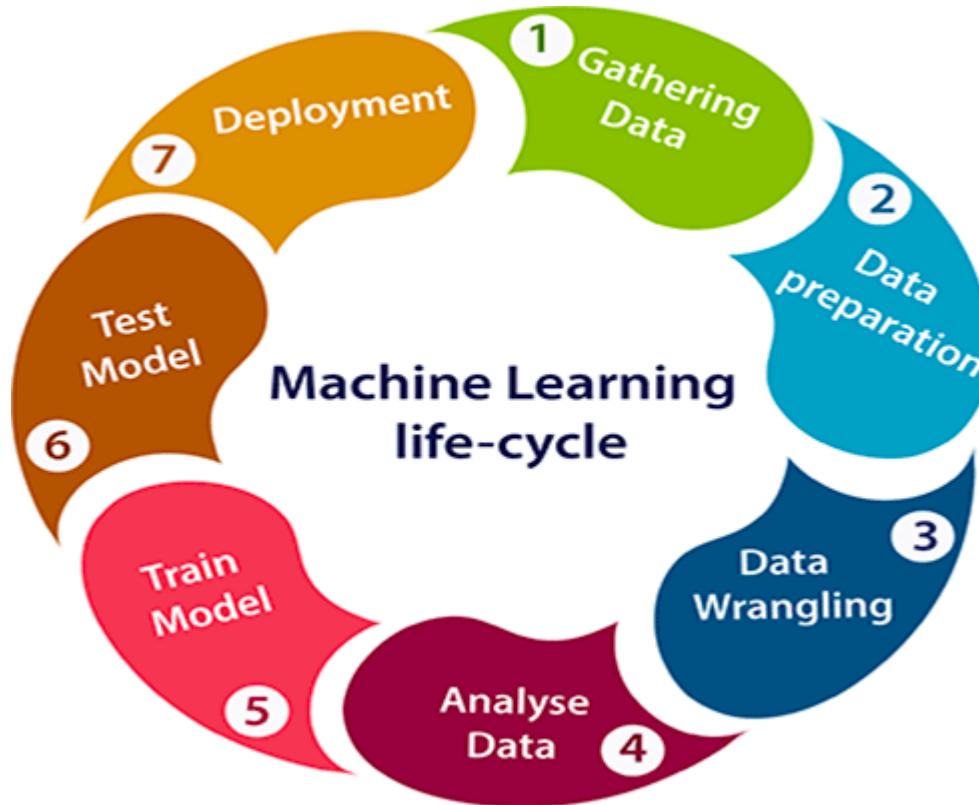
$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$



# ML Ops

## Models life-cycle

# ML/DL life-cycle



## Cloud actors



# all-in-one

## Amazon SageMaker overview

### Amazon SageMaker

#### PREPARE

**SageMaker Ground Truth**  
Label training data for machine learning

**SageMaker Data Wrangler NEW**  
Aggregate and prepare data for machine learning

**SageMaker Processing**  
Built-in Python, BYO R/Spark

**SageMaker Feature Store NEW**  
Store, update, retrieve, and share features

**SageMaker Clarify NEW**  
Detect bias and understand model predictions

#### BUILD

**SageMaker Studio Notebooks**  
Jupyter notebooks with elastic compute and sharing

**Built-in and Bring your-own Algorithms**  
Dozens of optimized algorithms or bring your own

**Local Mode**  
Test and prototype on your local machine

**SageMaker Autopilot**  
Automatically create machine learning models with full visibility

**SageMaker JumpStart NEW**  
Pre-built solutions for common use cases

#### TRAIN & TUNE

**Managed Training**  
Distributed infrastructure management

**SageMaker Experiments**  
Capture, organize, and compare every step

**Automatic Model Tuning**  
Hyperparameter optimization

**Distributed Training NEW**  
Training for large datasets and models

**SageMaker Debugger NEW**  
Debug and profile training runs

**Managed Spot Training**  
Reduce training cost by 90%

#### DEPLOY & MANAGE

**Managed Deployment**  
Fully managed, ultra low latency, high throughput

**Kubernetes & Kubeflow Integration**  
Simplify Kubernetes-based machine learning

**Multi-Model Endpoints**  
Reduce cost by hosting multiple models per instance

**SageMaker Model Monitor**  
Maintain accuracy of deployed models

**SageMaker Edge Manager NEW**  
Manage and monitor models on edge devices

**SageMaker Pipelines NEW**  
Workflow orchestration and automation

#### SageMaker Studio

Integrated development environment (IDE) for ML

# Data annotation

Hardly customizable

# prodigy

Radically efficient machine teaching.



French startup, still growing  
(and expensive)



KILI TECHNOLOGY

Quite expensive

# Label Studio



Free (open-source),  
so self-hosted...

# doccan

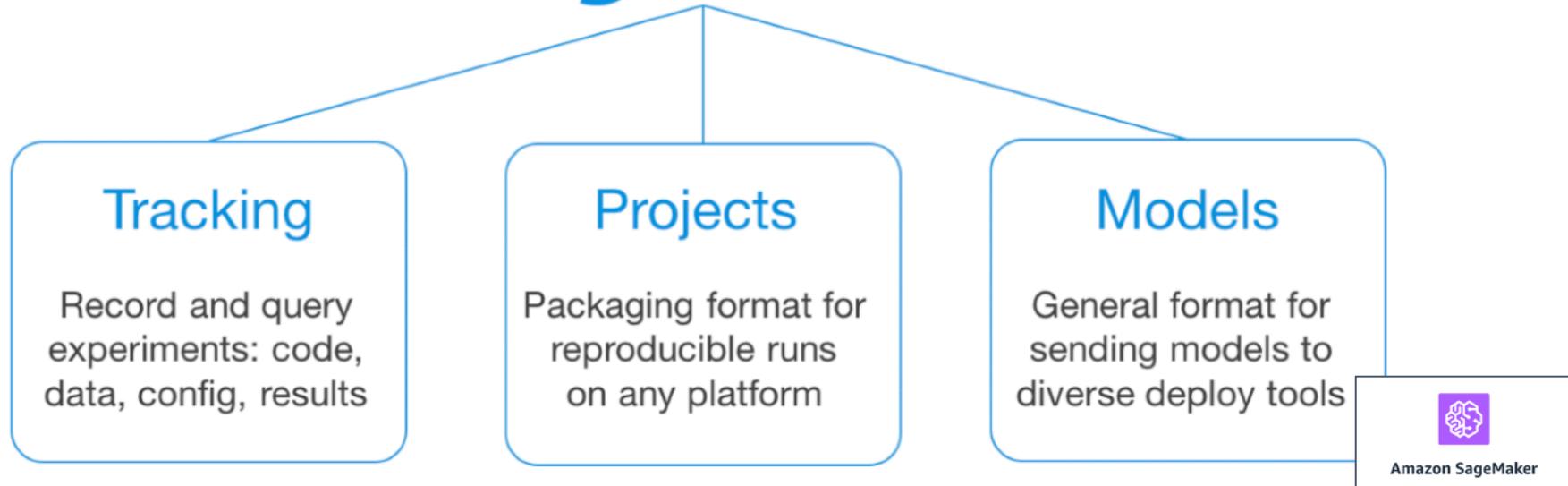


all-in-one



# amazon web services

# Experimentation & training



# Experimentation & training



# Experimentation & training

## Vertex AI

Créez, déployez et faites évoluer des modèles de ML plus rapidement avec des outils pré-entraînés et personnalisés dans une plate-forme d'intelligence artificielle unifiée.

[Essayer gratuitement Vertex AI](#)[Contacter le service commercial](#)

- ✓ Tirez parti des outils de ML innovants qui alimentent Google, développés par Google Research
- ✓ Déployez des modèles plus rapidement, avec 80 % de lignes de code en moins requis pour la modélisation personnalisée
- ✓ Utilisez les outils MLOps pour gérer facilement vos données et vos modèles en toute confiance, et les réutiliser à grande échelle

# Production deployment



Ready to use Machine Learning models



Cloud  
Vision API



Cloud  
Speech API



Cloud  
Jobs API



Cloud  
Translation  
API



Cloud Natural  
Language API



Cloud Video  
Intelligence API



Coming  
soon

# Google Vision Cloud API

Image	LABEL_DETECTION	LOGO_DETECTION	TEXT_DETECTION
	bottle, liqueur, wine bottle, distilled beverage, glass bottle	Becherovka	BECHEROVK ORIGINAL cl CARL SBA
	green, yellow, texture, pattern, wallpaper, computer wallpaper	/	/
	event, television crew, profession	/	Pecnod Ripard
	landmark, black and white, building, monochrome photography, structure, architecture, arch, urban area, metropolis, monochrome	/	/
	product, product, distilled beverage, liquid, lotion	Ballantine's	alain alanune BLENDED SCOTCH WHISKY SIGNATURE DISTILLERY AGID 17 YIAKS AGED YEARS

# Production deployment

A strong collaboration to make NLP easy & accessible for all



**Hugging Face**



Hugging Face is the most popular Open Source company providing state of the art NLP technology

**AWS**



Amazon SageMaker offers high performance resources to train and use NLP Models

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved | 2

One command is all you need

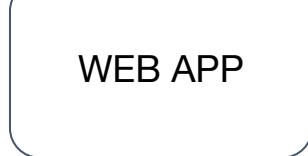
Accelerate machine learning from science to production

Built-in performance

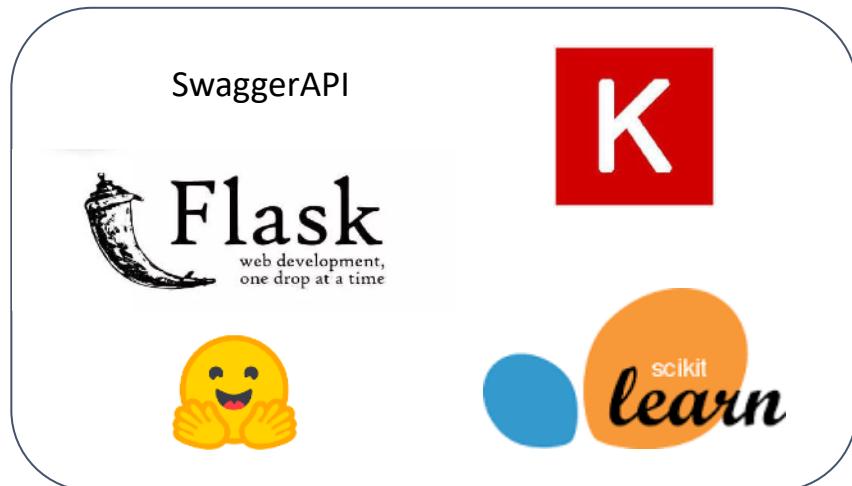
<https://huggingface.co/docs/sagemaker/getting-started>

# Models in production ?

- Using APIs (Application Programming Interfaces)



HTTP GET / POST  
↔



<https://huggingface.co/inference-endpoints>

# Showcase a model

Build a webapp in python, in order to show your model to your team without any software engineering!

<https://bert-keyword-extractor.streamlitapp.com/>



# Conclusion

# What you've learnt (big picture!)

- Deep learning is complex, but you can have some intuitions sometimes, if you think about the maths behind it.
- Simpler is better.
- Use transfer learning as often as you can. Careful with fine-tuning of the hidden states. Sometimes, you don't even fine-tuning.
- Your network doesn't learn anything? Think about dead neurons (activation), vanishing gradient (batch & layer norm), learning rates.
- Your network learns too much (overfitting)? Think about dropout, regularization, normalization, data augmentation.
- Data augmentation is really key (image & text). Remember François Chollet's tweet...

# References

## Cours de deep learning :

- [fastai : Cours en ligne orienté pratique.](#) Très adapté pour des profils plutôt développeurs, même si J. Howard donne aussi des intuitions (et fait des convolutions sur Excel).
- [CS231n](#) : Cours en ligne de Computer Vision donné par l'université Stanford. Laboratoire de Fei-Fei Li.
- [EE-59 : Deep Learning](#). Cours donné à l'EPFL - contenu très claire, et des exercices associés intéressants (de ce qu'on m'en a dit)
- [deeplearning.ai](#) : Andrew Ng's course on deep learning

## Blogs intéressants :

- [http://neuralnetworksanddeeplearning.com/](#) : blog très clair sur les intuitions du deep learning
- [Blog de Christopher Olah \(colah\)](#). Blog général sur la compréhension de certains points clés, notamment des réseaux de neurones récurrents.
- [Distill](#). Publications très pédagogues, articles avec beaucoup de visualisations.
- [Blog de Sebastian Ruder](#). Blog sur des notions / actualités en NLP. Sa newsletter aussi.
- [Machine Learning Mastery](#), notamment [un article sur les time series](#)
- [Playground tensorflow](#) pour jouer avec un réseau

## A suivre pour l'actualité du deep learning :

- Le groupe [Apprentissage Profond](#) sur Facebook : chouette communauté française autour du deep learning
- Sur Twitter, suivre [Jeremy Howard](#) (founder of fastai & ex director of Kaggle), [hardmaru](#), [François Chollet](#) (founder of Keras), [Thomas Wolf](#) (founder of HuggingFace, startup NLP française). Aussi [Andrej Karpathy](#) ([ex blog](#), [new blog](#)), [Ian Goodfellow](#) (fondateur des réseaux GANs), [Yann LeCun](#), [Andrew Ng](#)...
- [arxiv-sanity](#) : Feed de papers extraits d'arxiv, fait par Andrej Karpathy avec de la recommandation
- [Papers With Code : the latest in machine learning](#) : Papiers importants qui ont en plus le code disponible

# References

Des articles de blog de Doctrine pour des applications NLP :

- [Structuring legal documents with Deep Learning](#)
- [A single legal text representation at Doctrine: the legal camemBERT](#)
- Architecture “état de l’art” de 2016 dont on se sert à Doctrine en production pour de la reconnaissance d’entités nommées :  
[Neural Architectures for Named Entity Recognition, Lample et al, 2016](#)

Pour aller plus loin sur le DL...

- [La descente de gradient et les optimizers](#)
- [Les détails des équations de la backpropagation](#)
- [Explication plus granulaire du vanishing gradient](#)
- [Pour aller plus loin sur les convolutions](#)
- [Le papier qui introduit la batch normalization \(et un article de blog de vulgarisation\)](#)
- [Très bon article de blog sur les RNNs et LSTMs](#)
- [Word embeddings \(word2vec\)](#)
- Le papier qui introduit pour la première fois l’architecture Transformers: [Attention is all you need, 2017](#)
- [Tuto Hugging Face pour manipuler des Transformers](#), ou tout simplement jouer avec leur API démo

Quelques leaderboards!

- [Classification d’images sur ImageNet](#)
- [Classification de sentiments, IMDB reviews](#)

# References

Autre :

- [Quiver](#), librairie pour analyser les features trouvées par vos CNNs, [Lucid](#) & [Lime](#) pertinents aussi
- [Tensorboard](#), pour faire des callbacks depuis Keras / TensorFlow et observer votre loss directement sur un dashboard
- [Hundreds-Page Machine Learning... by Andriy Burkov PDF/iPad/Kindle](#)
- [Pense-bête](#) pour les réseaux de neurones très propre, par deux français chez Uber (voir la [version web aussi](#))
- [Pense-bête](#) pour savoir comment améliorer son modèle par Machine Learning Mastery (blog intéressant par ailleurs)
- <https://github.com/eugeneyan/applied-ml> référence à des papiers de recherche en NLP
- <https://github.com/daviddao/awful-ai> des applications peu éthiques en AI référencées sur ce github



# Thank you !