



Introduction to Deep Learning



Romain Raveaux

romain.raveaux@univ-tours.fr

Maître de conférences

Université de Tours

Laboratoire d'informatique (LIFAT)

Equipe RFAI



LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS



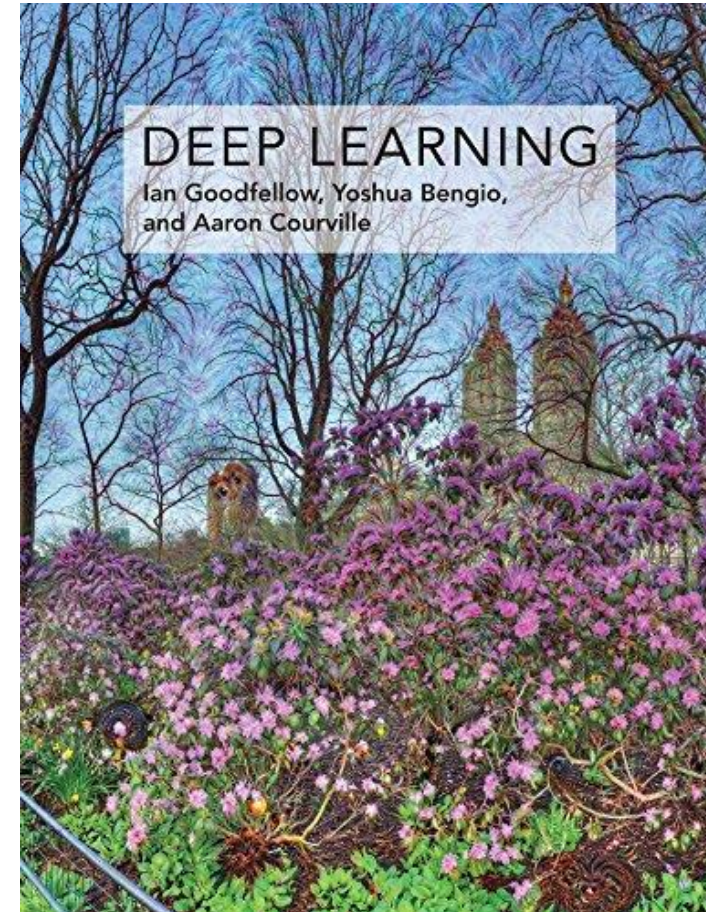
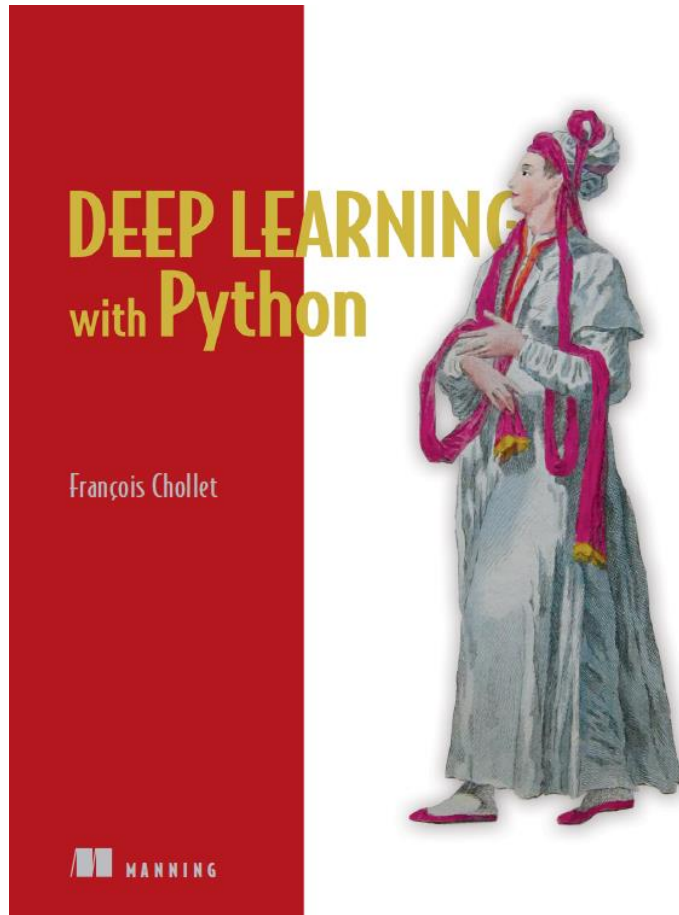
Content

- Introduction
 - Definition
 - Why Deep Learning now ?
 - Deep Learning in Vision
 - Deep Learning in speech processing
 - Deep Learning in games
- Data and representation
- Model
- Supervised learning of a deep model
- Good stuff in deep learning
- Health-care Applications
- Conclusions

Goal of this presentation

- Provide the key elements inside the deep learning paradigm
- Provide the key elements to understand the successes of deep learning in many applications (Game of Go, speech recognition, image classification, ...).

Books



“I strongly believe that there are no difficult ideas in deep learning.” François Chollet

The main references for this course are :

- Ian Goodfellow, Yoshua Bengio and Aaron Courville :
<http://www.deeplearningbook.org/>
- Bishop (1995) : Neural networks for pattern recognition, Oxford University Press.
- Hugo Larochelle (Sherbrooke): <http://www.dmi.usherb.ca/larocheh/>
- Deep learning course, Charles Ollion et Olivier Grisel :
<https://github.com/m2dsupsdclass/lectures-labs>
- François Chollet : Deep Learning with Python:
<http://faculty.neu.edu.cn/yury/AI/Textbook/Deep%20Learning%20with%20Python.pdf>
- Lecture of Julien Mille . Introduction to Deep Learning. Maître de conférences au LIFAT
- Andrew Ng : <https://fr.coursera.org/specializations/deep-learning>

Introduction

Deep Learning : Introduction

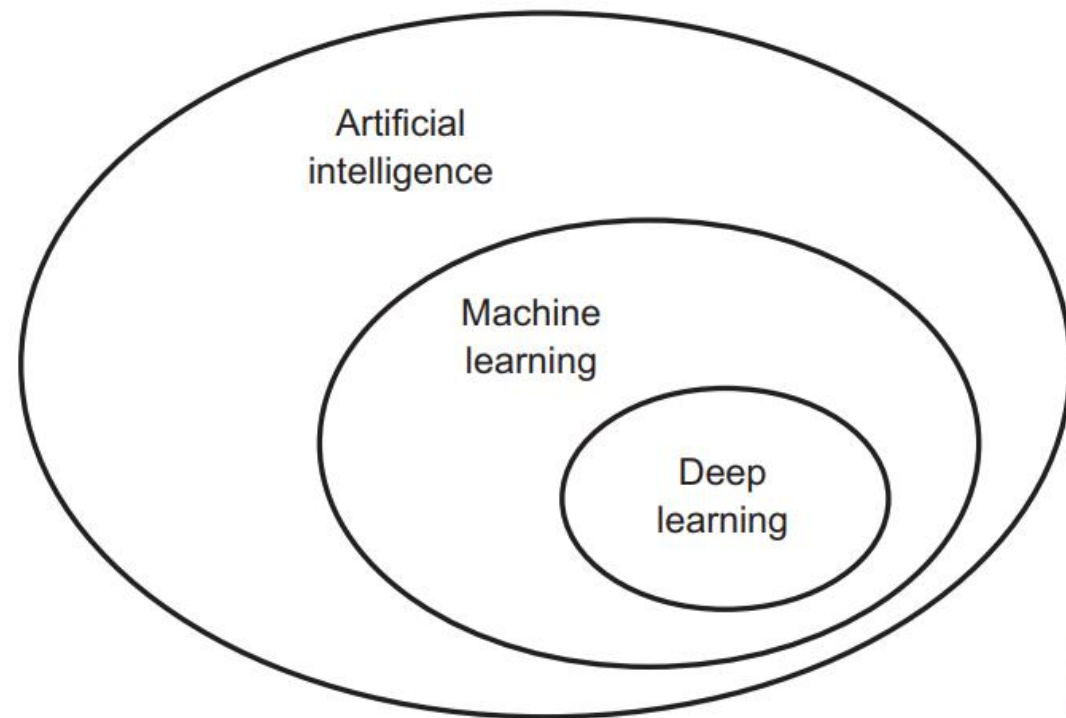


Figure by François Chollet : Deep Learning with Python

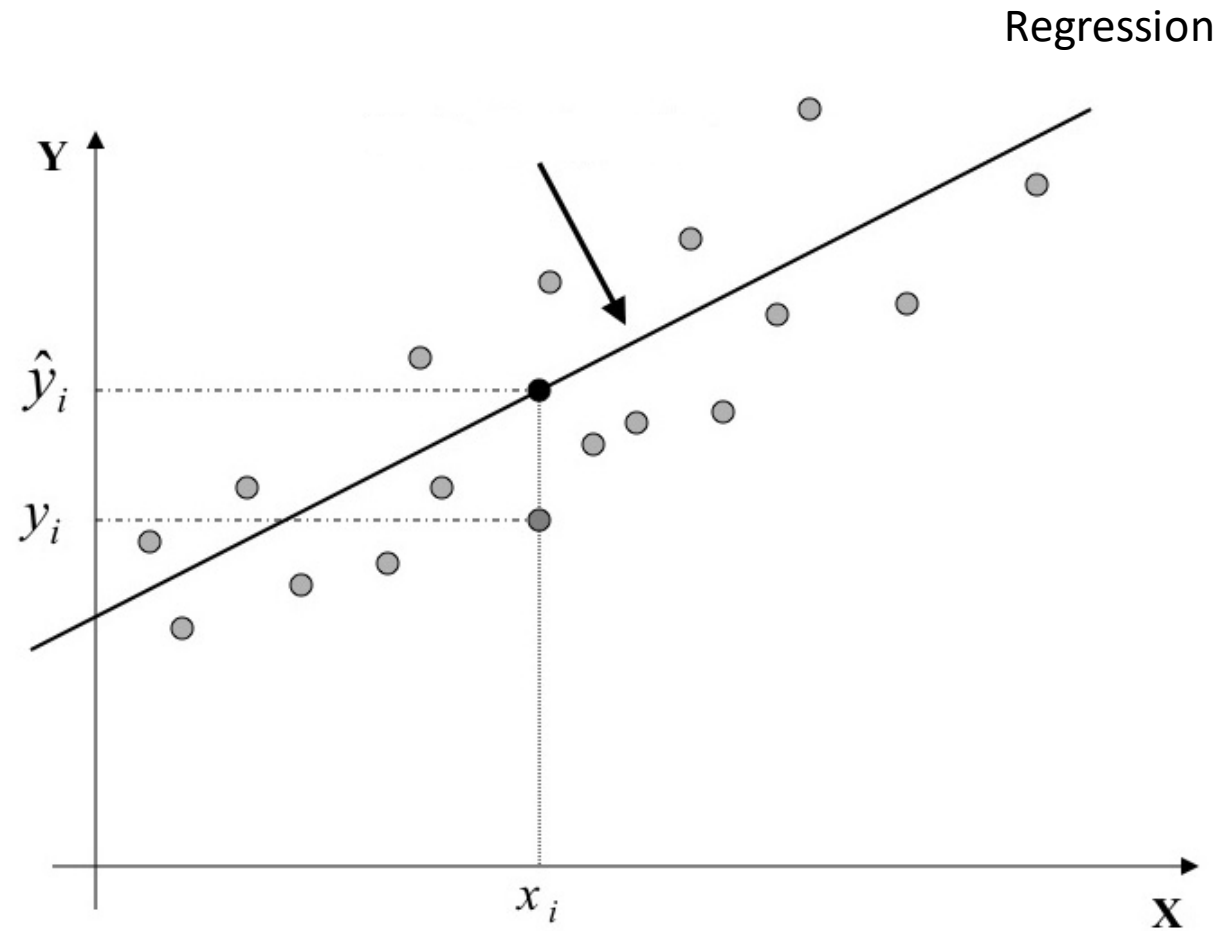
Learning paradigms

- Supervised learning
 - task of learning a function: f
 - that maps an input to an output : $f: X \rightarrow Y$
 - based on example input-output pairs : $\{x_i, y_i\}_{i=1}^N$; $x_i \in X$ et $y_i \in Y$
 - Reinforcement learning $f = f_1 \circ f_{\dots} \circ f_k: X \rightarrow \bar{Y}$
 - differs from supervised learning
 - f is composed of sub functions (in X) to output \bar{y}_i ,
 - Incomplete feedback (\bar{y}_i is not completely given $\bar{y}_i \subset y_i \in Y$)
- Unsupervised learning :
 - task of learning a function: f
 - that maps an input to an output : $f: X \rightarrow Z$
 - based on input examples : $\{x_i\}_{i=1}^N$; $x_i \in X$
 - No labels (y_i) are given
- Self-supervised learning
 - is autonomous supervised learning.
 - eliminates the pre-requisite requiring humans to label data.

Recall : Supervised learning

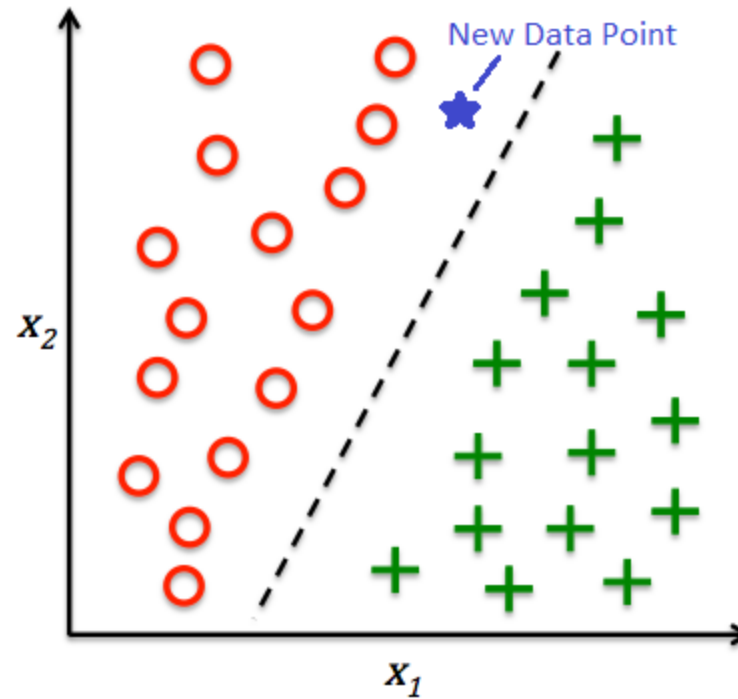
- Regression : $f: X \rightarrow Y$ and $y \in \mathbb{R}$
- Classification : $f: X \rightarrow Y$ and $y \in \{a, b, c, d\}$
- Do not read the next lines
 - Structured classification : $f: X \rightarrow Y$ and $y \in \{a, b, c, d\}^{N \times M}$
 - Structured Regression : $f: X \rightarrow Y$ and $y \in \mathbb{R}^{N \times M}$
 - Structured Regression : $f: X \rightarrow Y$ and $y \in \mathbf{G}$ (*graph space*)
 - Structured learning when the output is complex:
 - Segmentation mask for instance

Recall : Supervised learning



Recall : Supervised learning

Classification



Deep Learning : Introduction

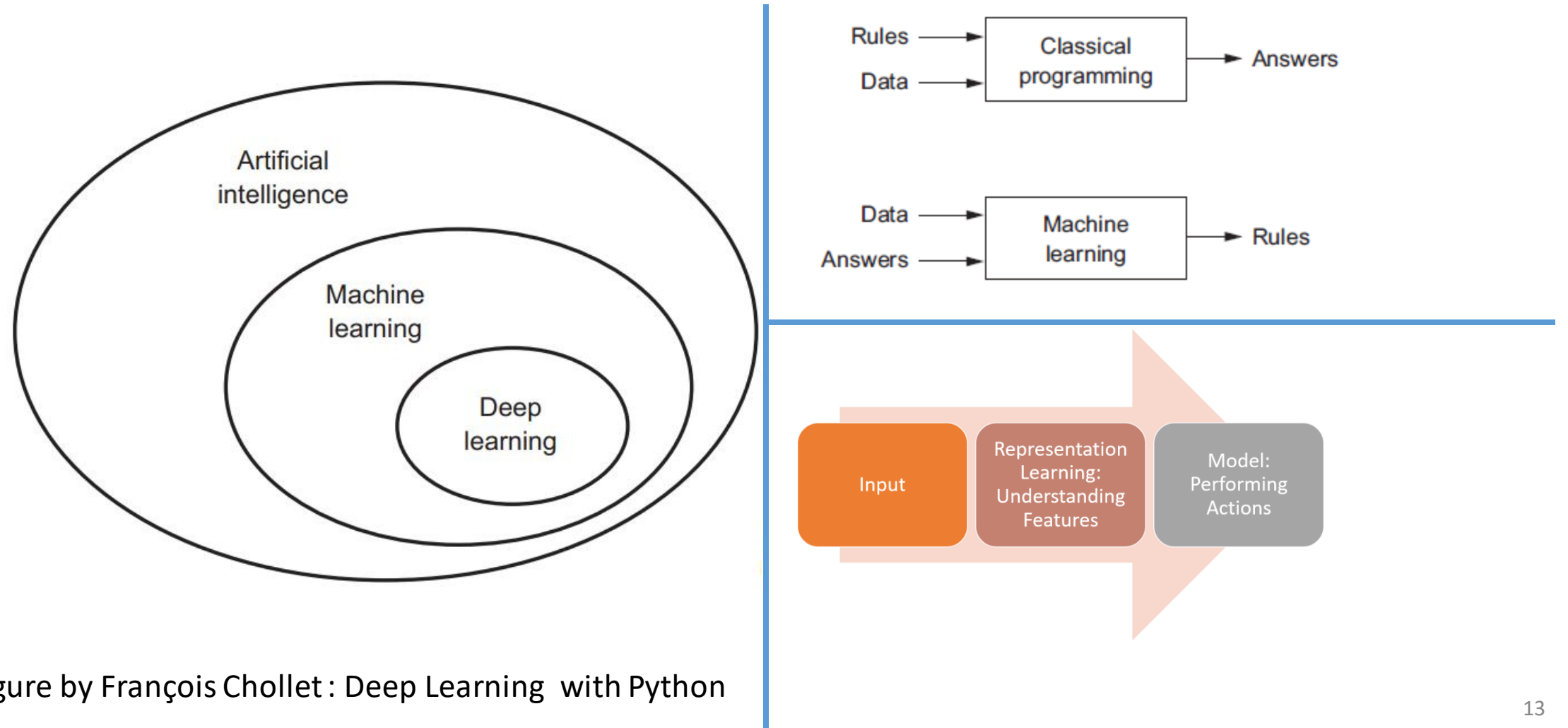


Figure by François Chollet : Deep Learning with Python

Deep Learning : Introduction

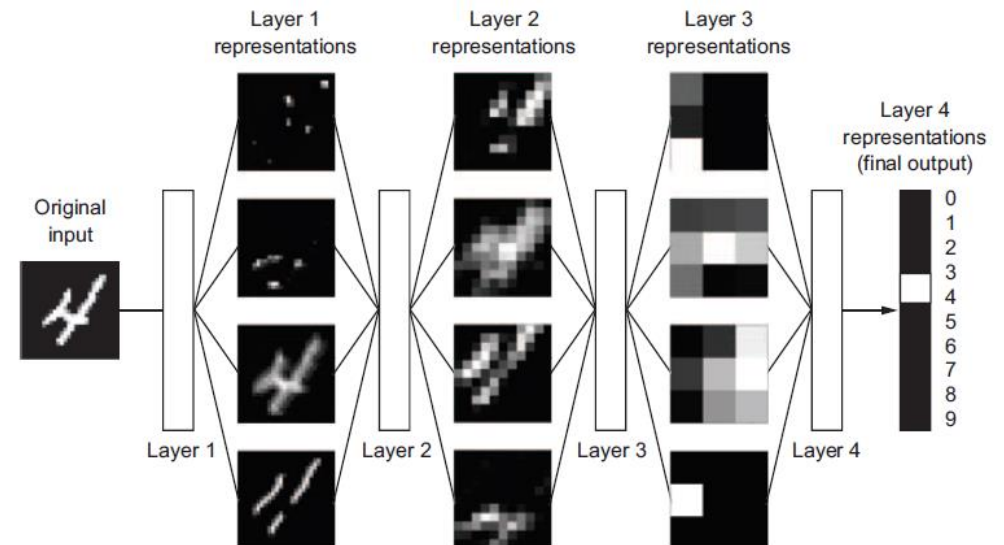
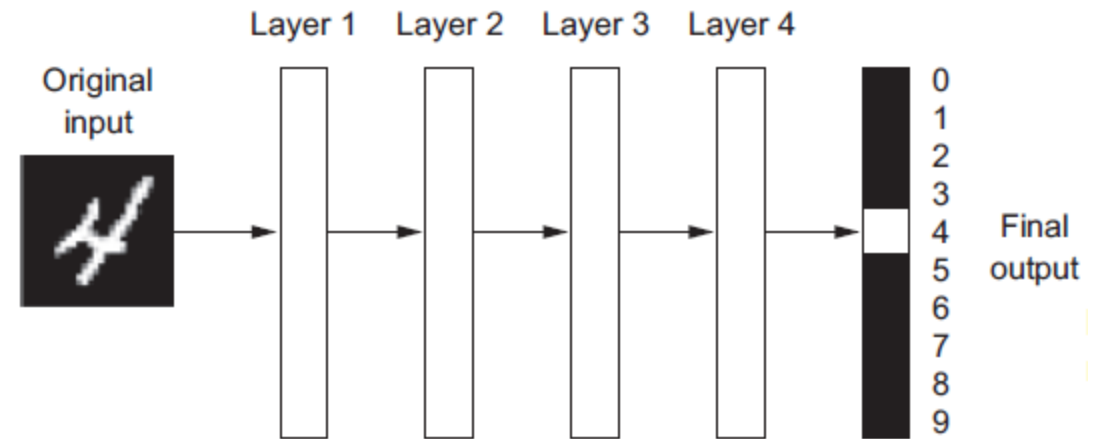
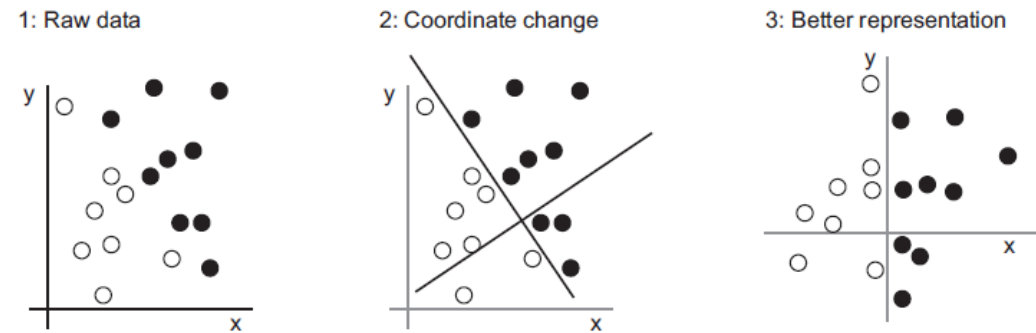
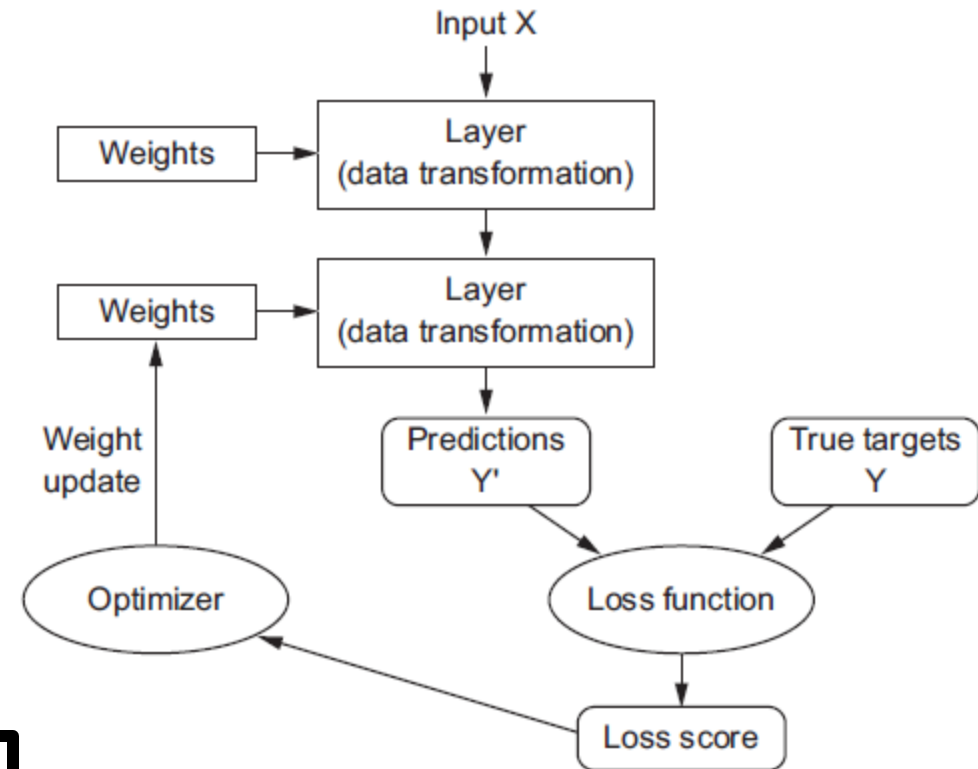
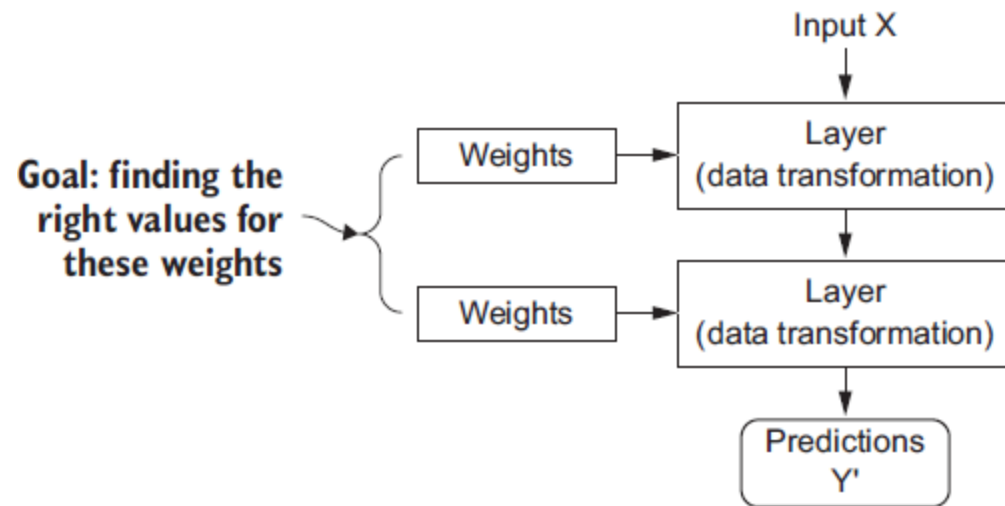


Figure by François Chollet : Deep Learning with Python

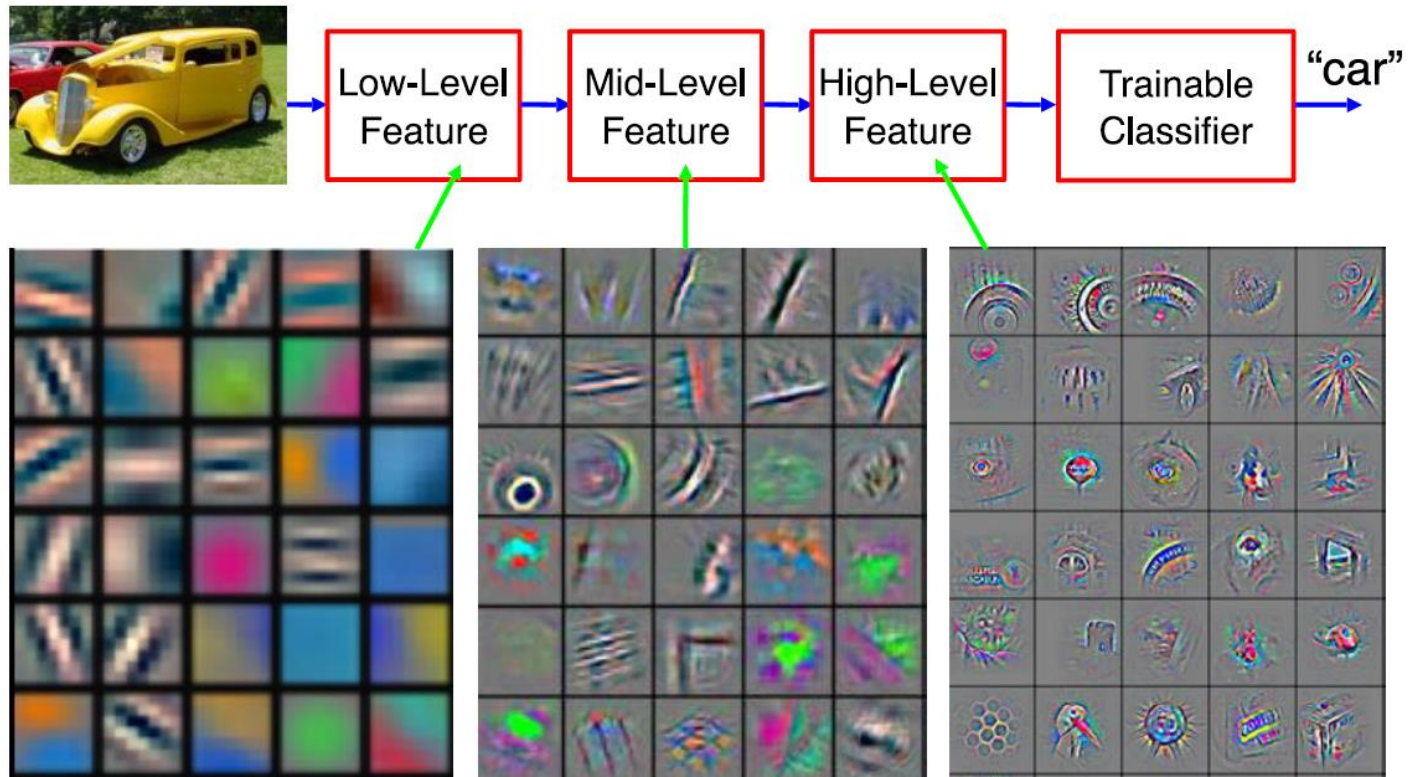
Deep Learning : Introduction



Deep Learning is about finding a good data representation with respect to an objective thanks to a composition of transformations (functions/layers)

Deep Learning : Introduction

- ▶ Learning a hierarchy of increasingly abstract representations



Deep Learning : Introduction

Deep Learning → End-to-End learning

- ▶ A hierarchy of trainable feature transforms.
- ▶ Each module transforms its input representation into a higher-level one.
- ▶ Low-level features are shared among categories.
- ▶ As the level increases, features are increasingly global and invariant.

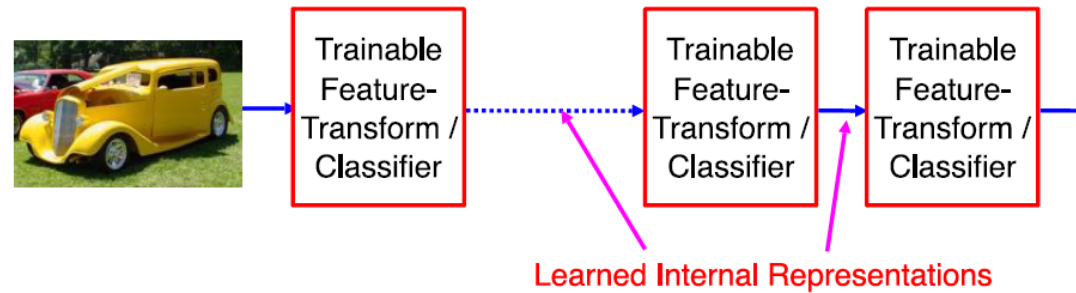
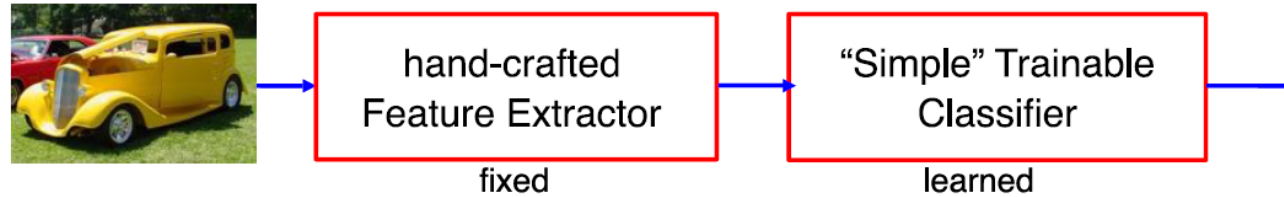


Figure by Y.Lecun and M.A. Ranzato

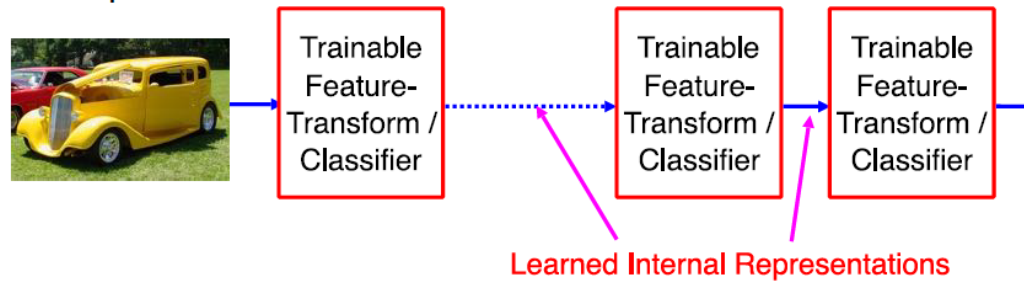
Deep Learning : Introduction

"Shallow" vs Deep Learning

▶ "Shallow" models



▶ "Deep" models



Figures by Y.Lecun and M.A. Ranzato

Deep Learning heroes : turing award



Deep Learning : Definitions

- “Deep Learning is learning good representations by composition of learned functions.” (From Yoshua Bengio)
 - Composition = Complicated functions are combinations of smaller, simpler functions.
- “Deep Learning is building a system by assembling parameterized modules into a computation graph, and training it to perform a task by optimizing the parameters using a gradient-based method.” (From Yann Lecun)
- So deep learning is (From Yoshua Bengio)
 - (1) an approach to machine learning
 - (2) usually based on artificial neural networks but not only
 - (3) based on learning multiple levels of representations or based on composing learned functions to learn good representations
 - (4) in order to achieve the learning of such good representations and perform well in machine learning tasks.

What is the left to the Engineer

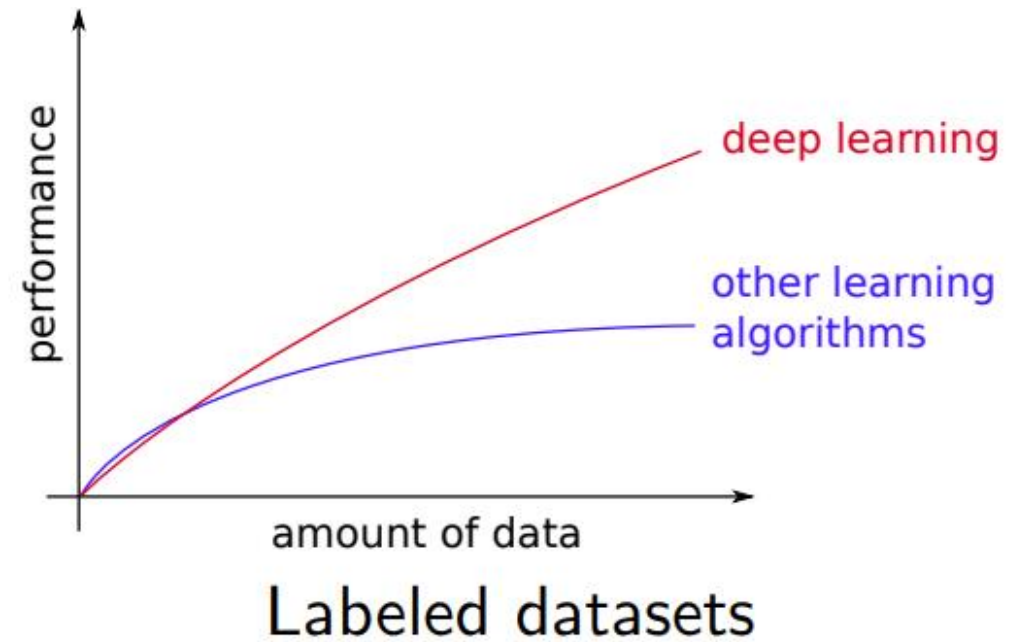
- Learning paradigms : supervised, reinforced, self-supervised/unsupervised
- Objective functions : how to measure the closeness to the goal
- The architecture : the layout of the modules

Why Deep Learning now ?

1. More Data
2. More Computing Power



Computing power (GPUs)



Deep learning : some successes

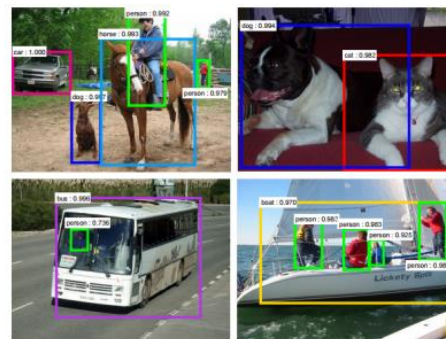
► Deep Learning in Vision



[Krizhevsky 2012]



[Ciresan et al. 2013]



[Faster R-CNN - Ren 2015]

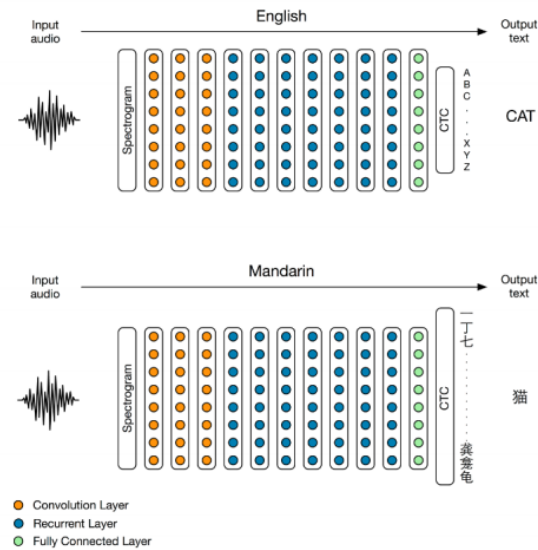


[NVIDIA dev blog]

Figures by O. Grisel and C. Ollion, <https://github.com/m2dsupsd1class>

Deep learning : some successes

► Deep Learning in speech processing

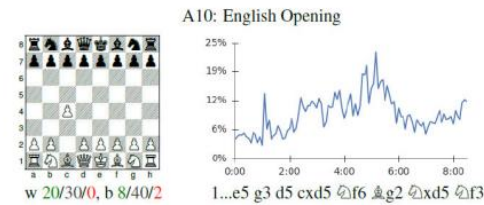


[Baidu 2014]

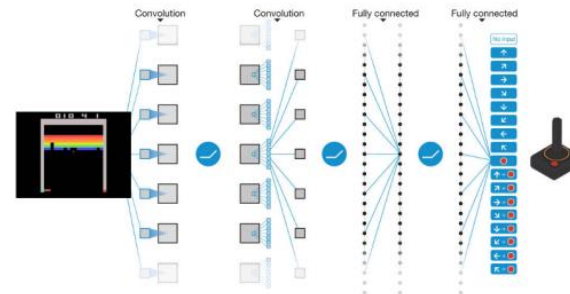
Figures by O. Grisel and C. Ollion, <https://github.com/m2dsupsdclass>

Deep learning : some successes

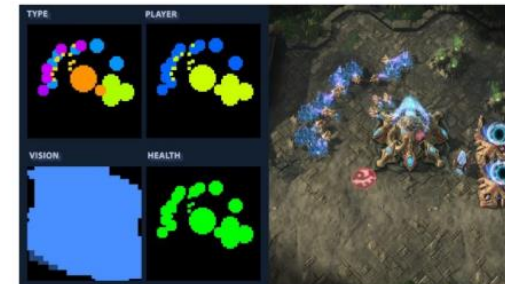
► Deep Learning in games



[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]



[Starcraft 2 for AI research]

Figures by O. Grisel and C. Ollion, <https://github.com/m2dsupsd1class>

Deep learning : some successes

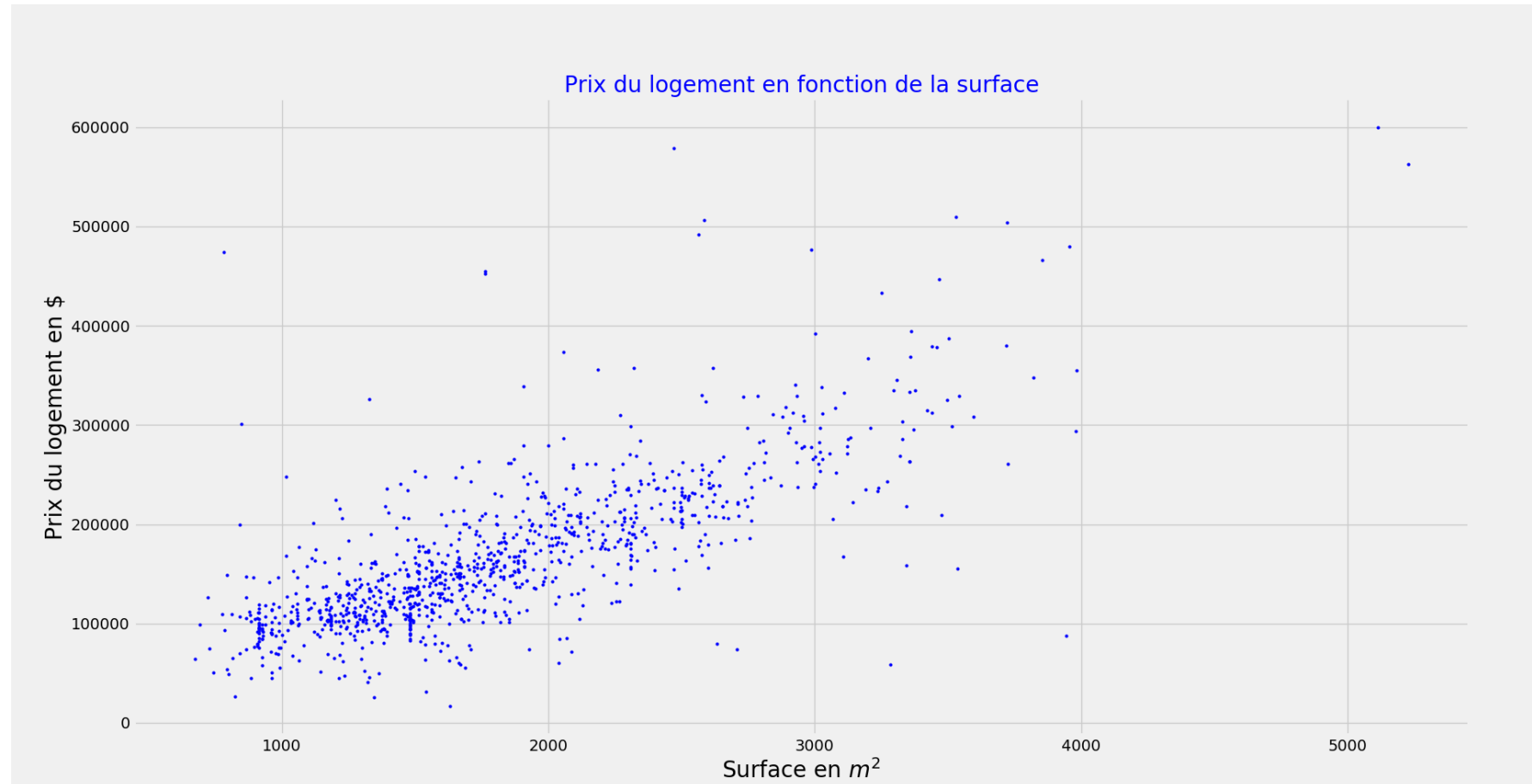
- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Improved machine translation
- Improved text-to-speech conversion
- Digital assistants such as Google Now and Amazon Alexa
- Near-human-level autonomous driving
- Improved ad targeting, as used by Google, Baidu, and Bing
- Improved search results on the web
- Ability to answer natural-language questions
- Superhuman Go playing

Data and representation

It starts with data !!!!

- What is a datum ?????
 - A world measurement from a sensor : image, sound, ...
 - A synthetic data : digital born information. This text !!!
- Type of discrete data :
 - Nominal data : « cat », « dog », « horse »
 - Ordinal data : « small », « medium », « large »
 - Numerical data : Something measurable
 - 1D : 10 kg, 25 kg, 30 kg, ...
 - 1D circular data : 10 °, 350 ° ,
 - 2D : [5 cm; 10kg]

House pricing : money(\$) = f(surface)



Iris flower data set

The data set consists of 150 samples

The length and the width of the [sepals](#) and [petals](#), in centimeters.

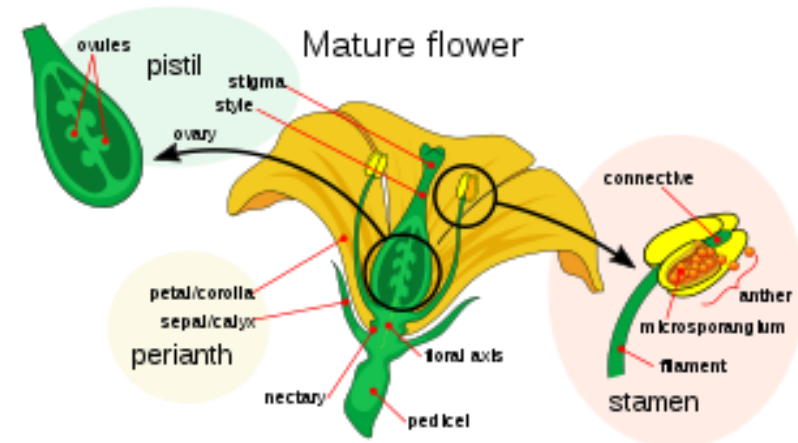
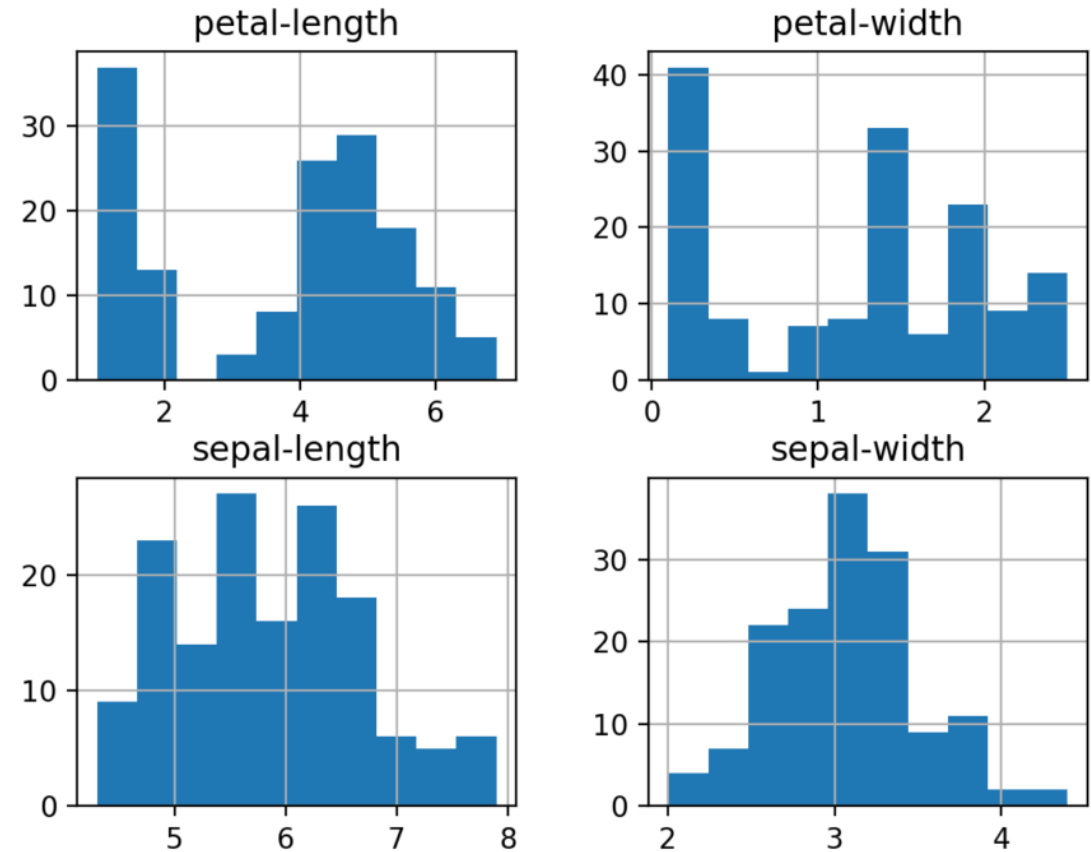
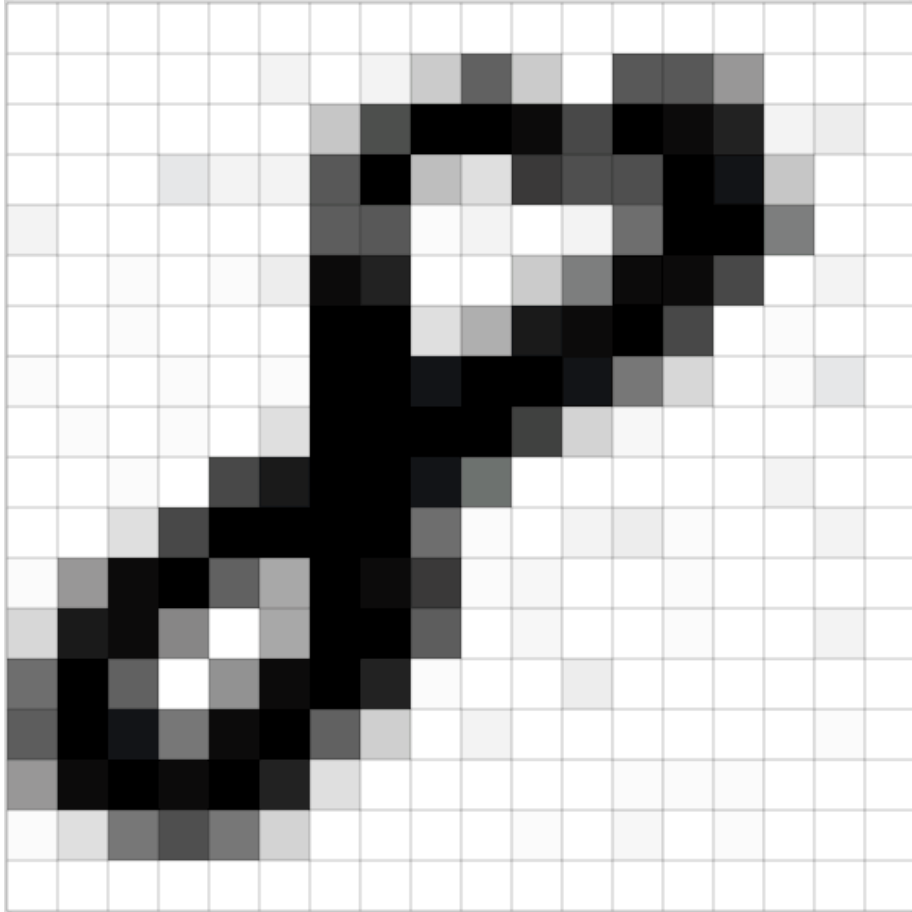


Image in numbers



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 12 0 11 39 137 37 0 152 147 84 0 0 0 0
0 0 1 0 0 0 41 160 250 255 235 162 255 238 206 11 13 0
0 0 0 16 9 9 150 251 45 21 184 159 154 255 233 40 0 0
10 0 0 0 0 0 145 146 3 10 0 11 124 253 255 107 0 0
0 0 3 0 4 15 236 216 0 0 38 109 247 240 169 0 11 0
1 0 2 0 0 0 253 253 23 62 224 241 255 164 0 5 0 0
6 0 0 4 0 3 252 250 228 255 255 234 112 28 0 2 17 0
0 2 1 4 0 21 255 253 251 255 172 31 8 0 1 0 0 0
0 0 4 0 163 225 251 255 229 120 0 0 0 0 0 11 0 0
0 0 21 162 255 255 254 255 126 6 0 10 14 6 0 0 9 0
3 79 242 255 141 66 255 245 189 7 8 0 0 5 0 0 0 0
26 221 237 98 0 67 251 255 144 0 8 0 0 7 0 0 11 0
125 255 141 0 87 244 255 208 3 0 0 13 0 1 0 1 0 0
145 248 228 116 235 255 141 34 0 11 0 1 0 0 0 1 3 0
85 237 253 246 255 210 21 1 0 1 0 0 6 2 4 0 0 0
6 23 112 157 114 32 0 0 0 0 2 0 8 0 7 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

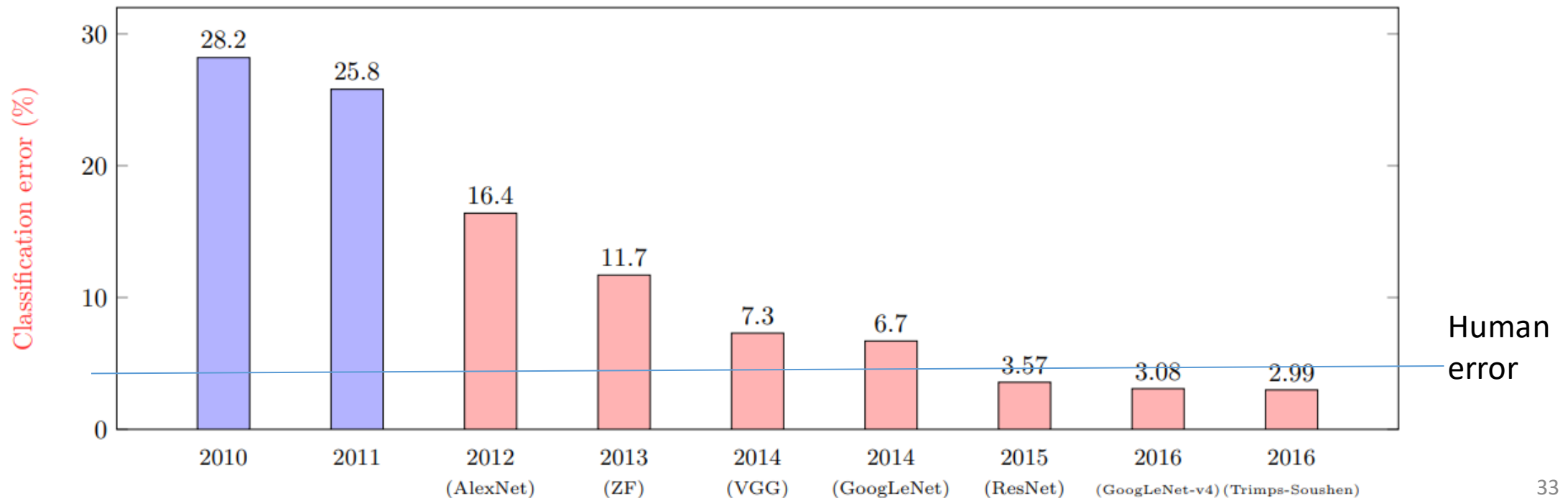
MNIST database



- 70 000 samples
- 1 sample = one image
- One image of size 28x28

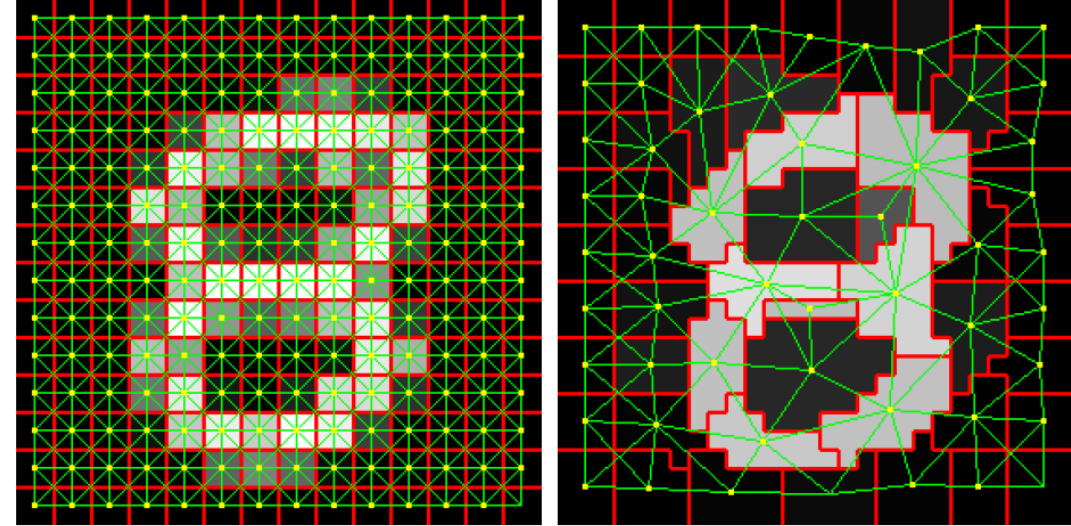
ImageNet Large Scale Visual Recognition Challenge

- 1000 object classes, 1.4 million labelled training images
- Pink indicates a deep learning based solution.
- **Deep learning based solutions exceeded human ability at the narrow ILSVRC tasks**

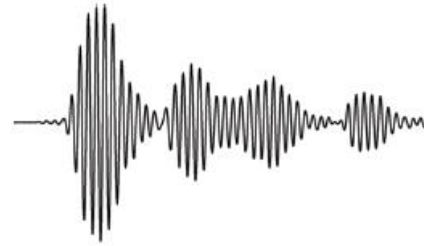


Data representation

- Data is
 - discrete (in representation)
 - structured / unstructured (e.g. grid / cloud)
 - of a specific dimension D (e.g. 2D / 3D)
- Usually categorised into specific types:
 - scalar (1D)
 - vector (commonly 2D or 3D; ND in \mathbb{R}^N)
 - tensor (N dimensional array)
 - Rank 0 is a scalar — Rank 1 is a vector — Rank 2 is a matrix



Numerical data



Audio signals



Images

And beyond : Euclidean and non Euclidean domains

Doubt thou the stars are fire,
Doubt that the sun doth move,
Doubt truth to be a liar,
But never doubt I love...

Text



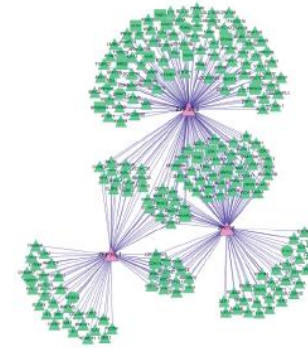
Audio signals



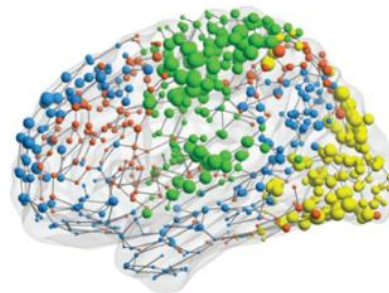
Images



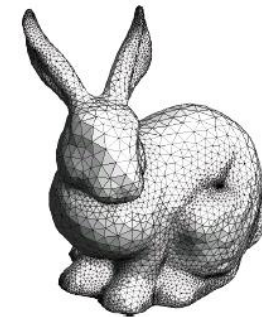
Social networks



Regulatory networks



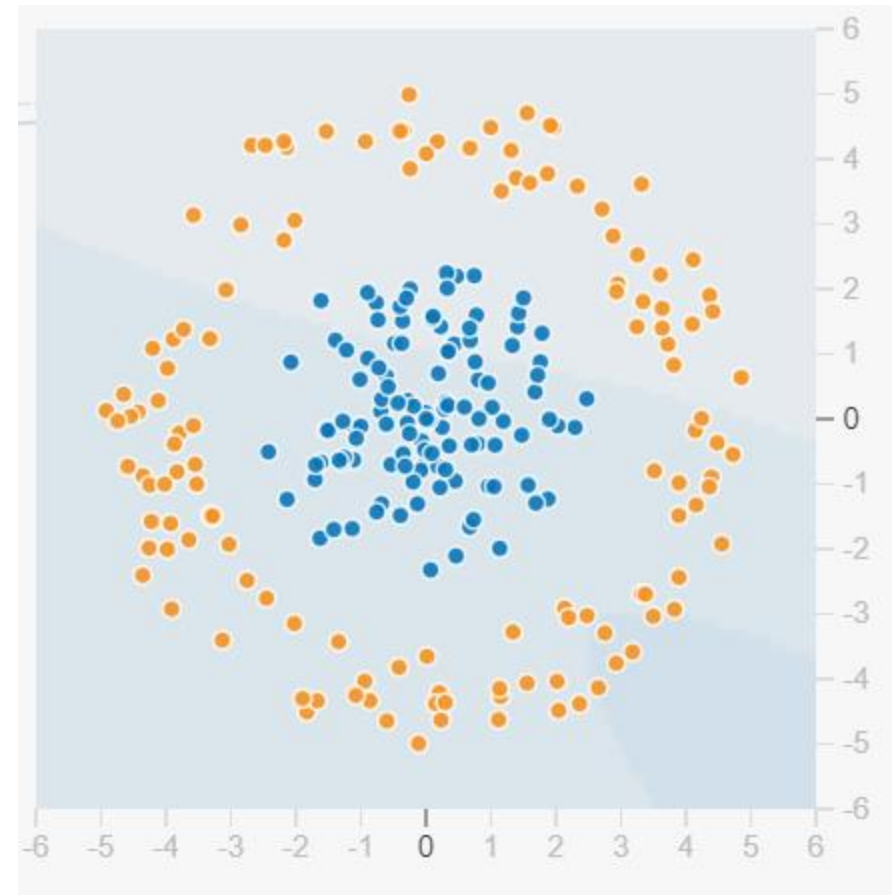
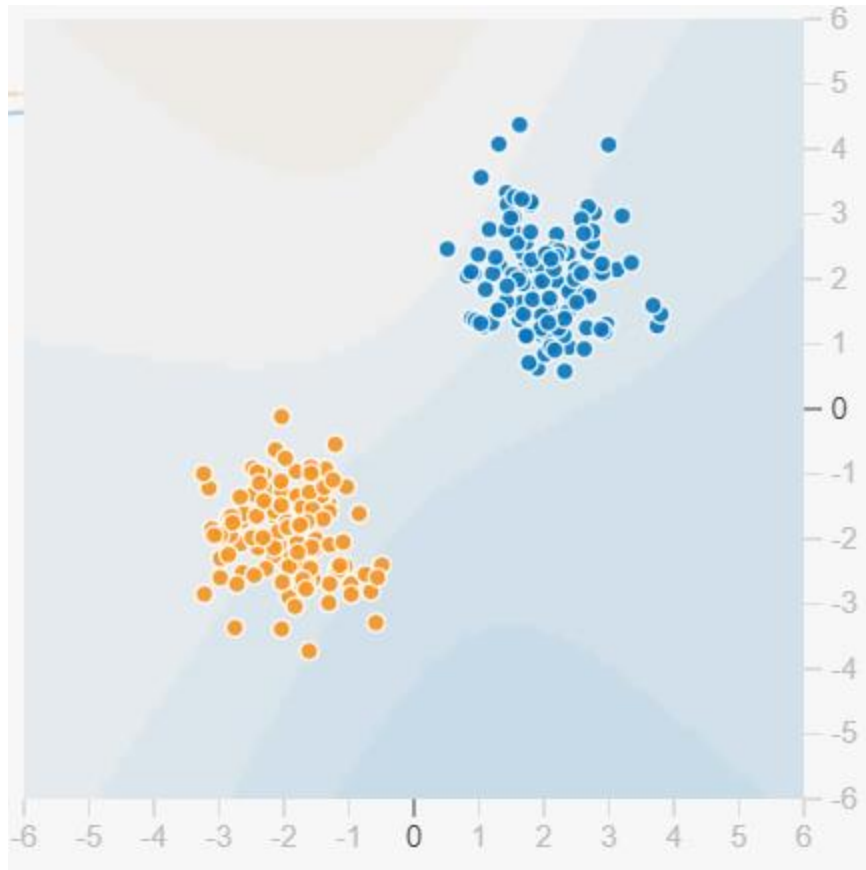
Functional networks



3D shapes

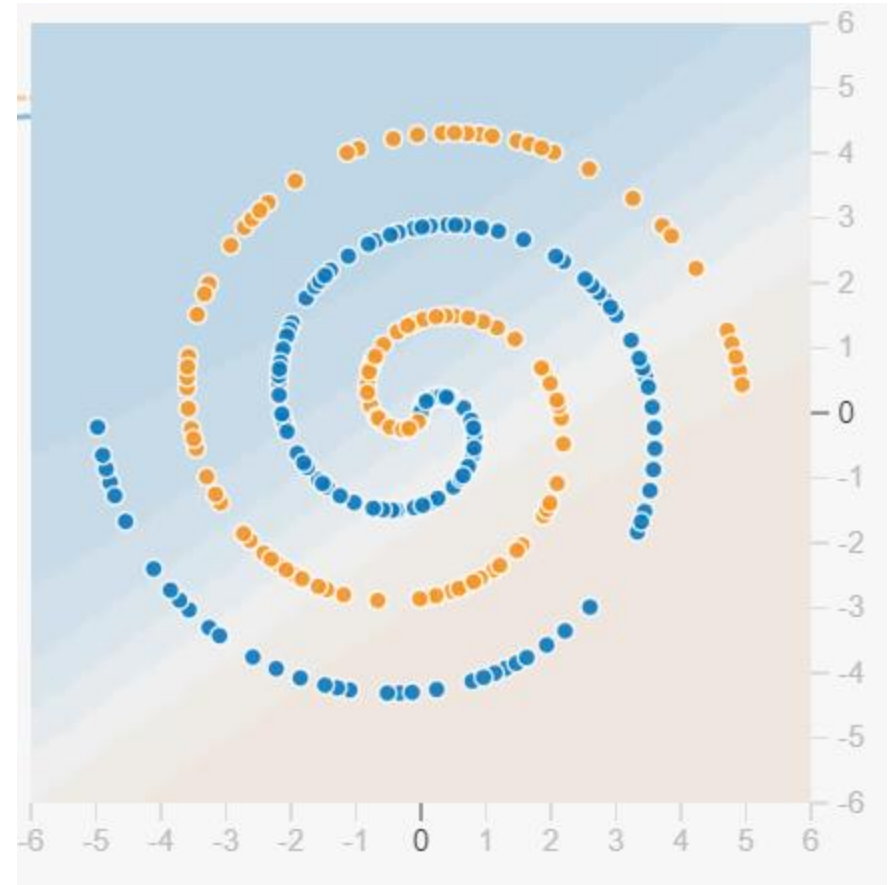
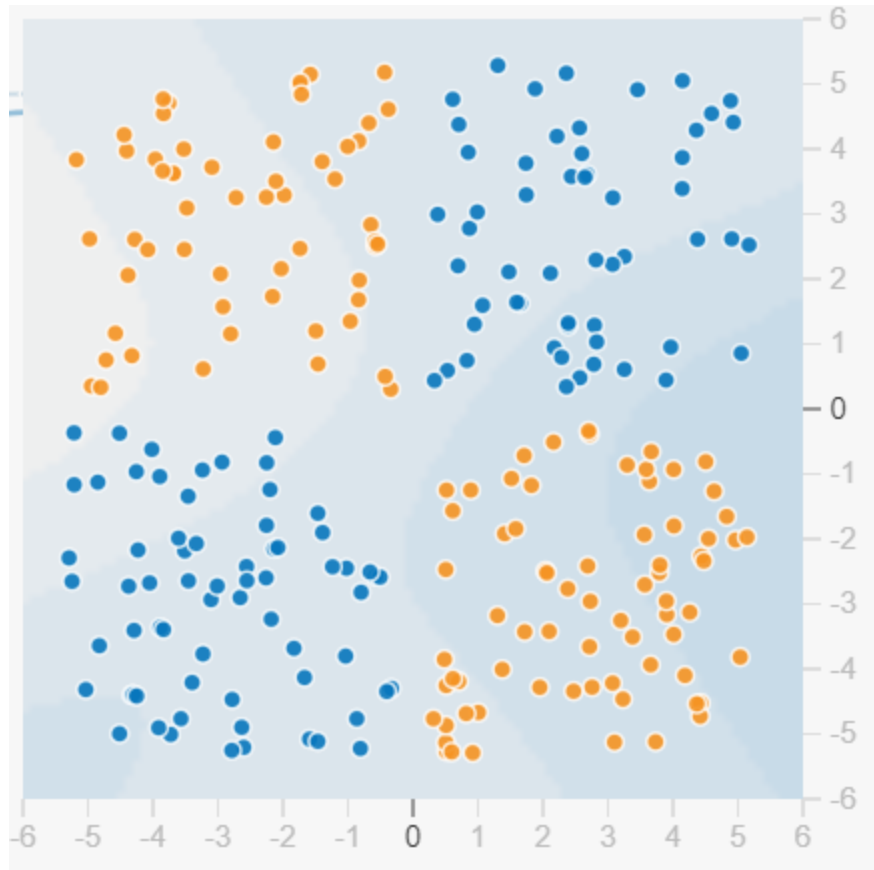
→ Information associated with data topology

Data representation : Features are important to take actions



<https://playground.tensorflow.org/>

Data representation : Features are important to take actions



Feature transformation : Kernel trick

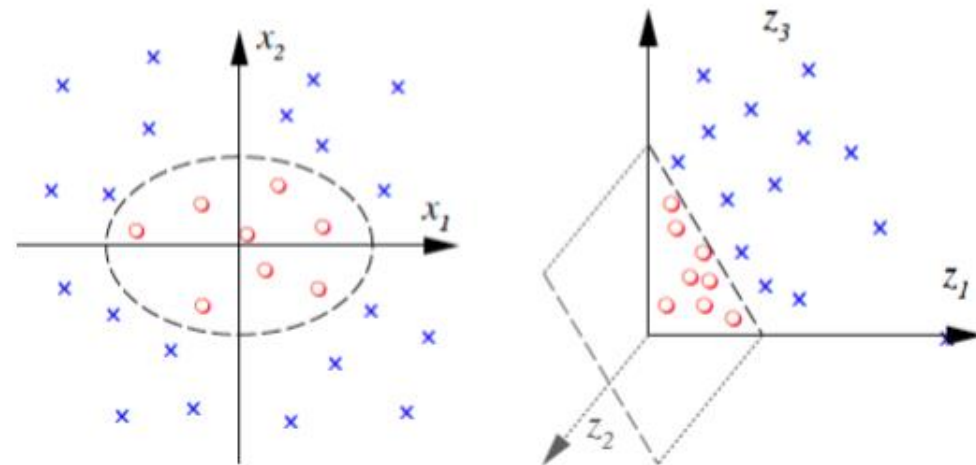
Example:

- Let \vec{x} and \vec{y} be two vectors in \mathbb{R}^2
- Let $\vec{x} = (x_1, x_2)$ and $\vec{y} = (y_1, y_2)$
- Let $\varphi(\vec{x})$ and $\varphi(\vec{y})$ be two functions projecting \vec{x} and \vec{y} into \mathbb{R}^3 .
- $k(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2$
- $k(\vec{x}, \vec{y}) = x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + y_2^2 y_2^2$
- $k(\vec{x}, \vec{y}) = \langle (x_1^2, \sqrt{2}x_1 x_2, x_2^2), (y_1^2, \sqrt{2}y_1 y_2, y_2^2) \rangle$
- $k(\vec{x}, \vec{y}) = \langle \varphi(\vec{x}), \varphi(\vec{y}) \rangle$
- $\varphi(\vec{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$

Example:

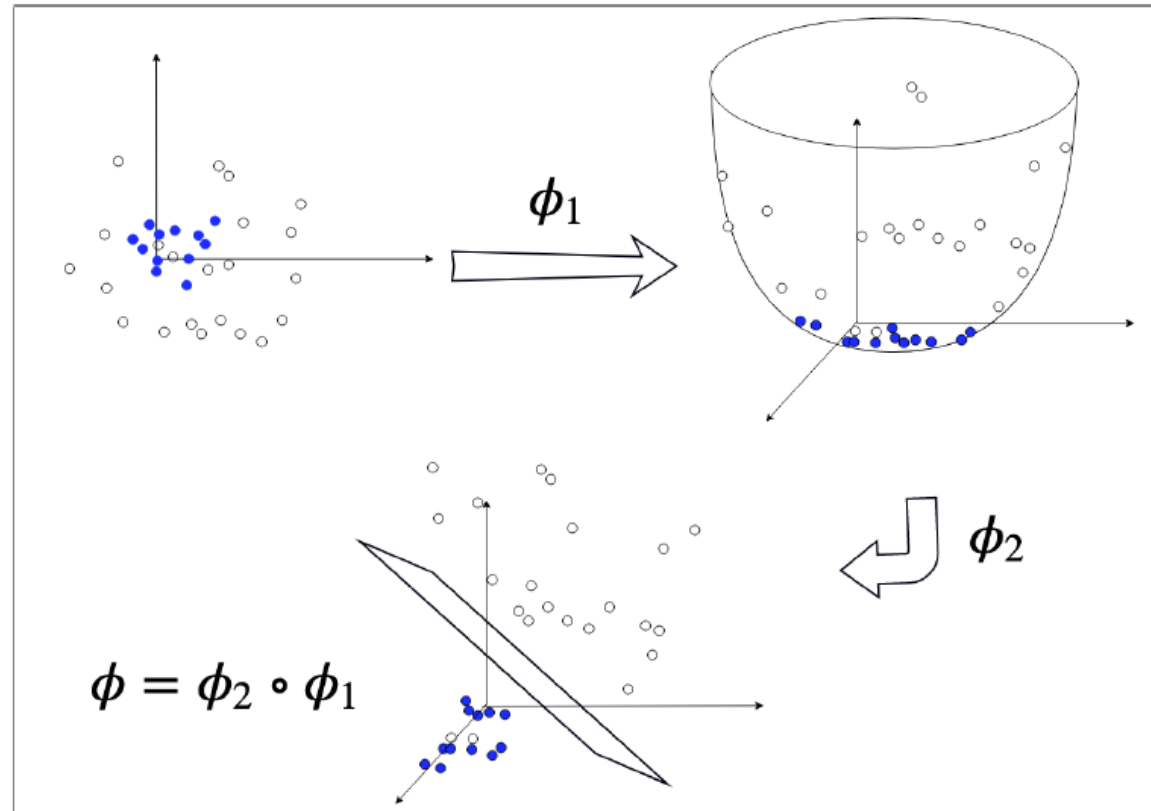
$$\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$$



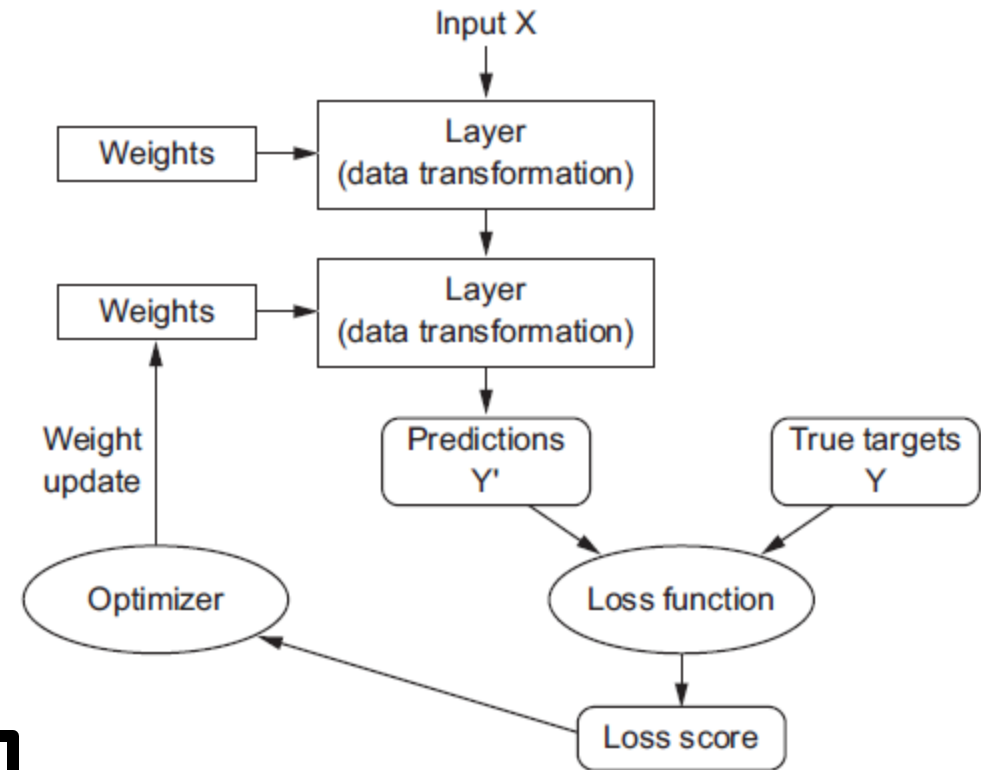
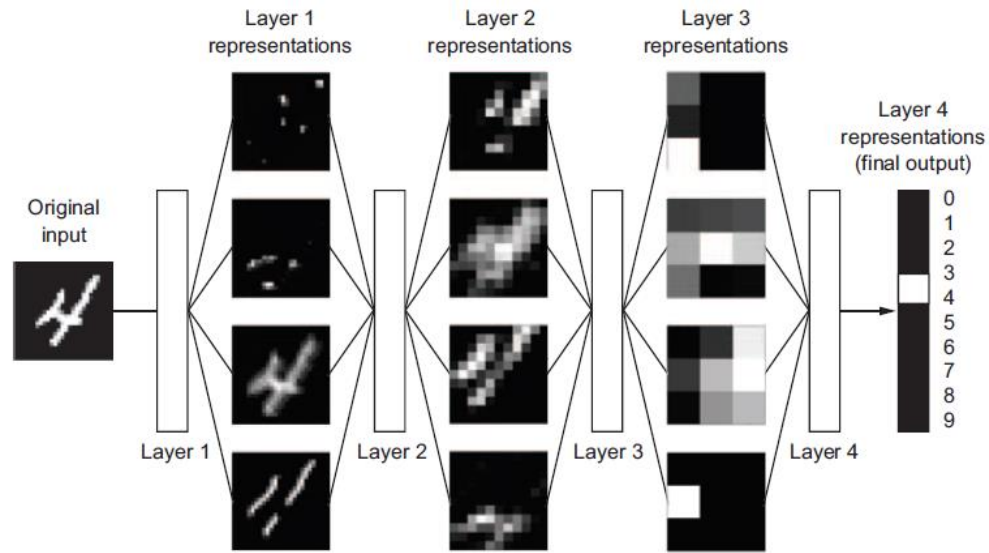
- **Change the representation (the feature space)**
- **A new representation where it is easier to take actions (classification, regression)**

Feature transformation : Composition



Transforming inputs \mathbf{x}_i 's using a certain (parameterized) function ϕ .

Data representation : a summary

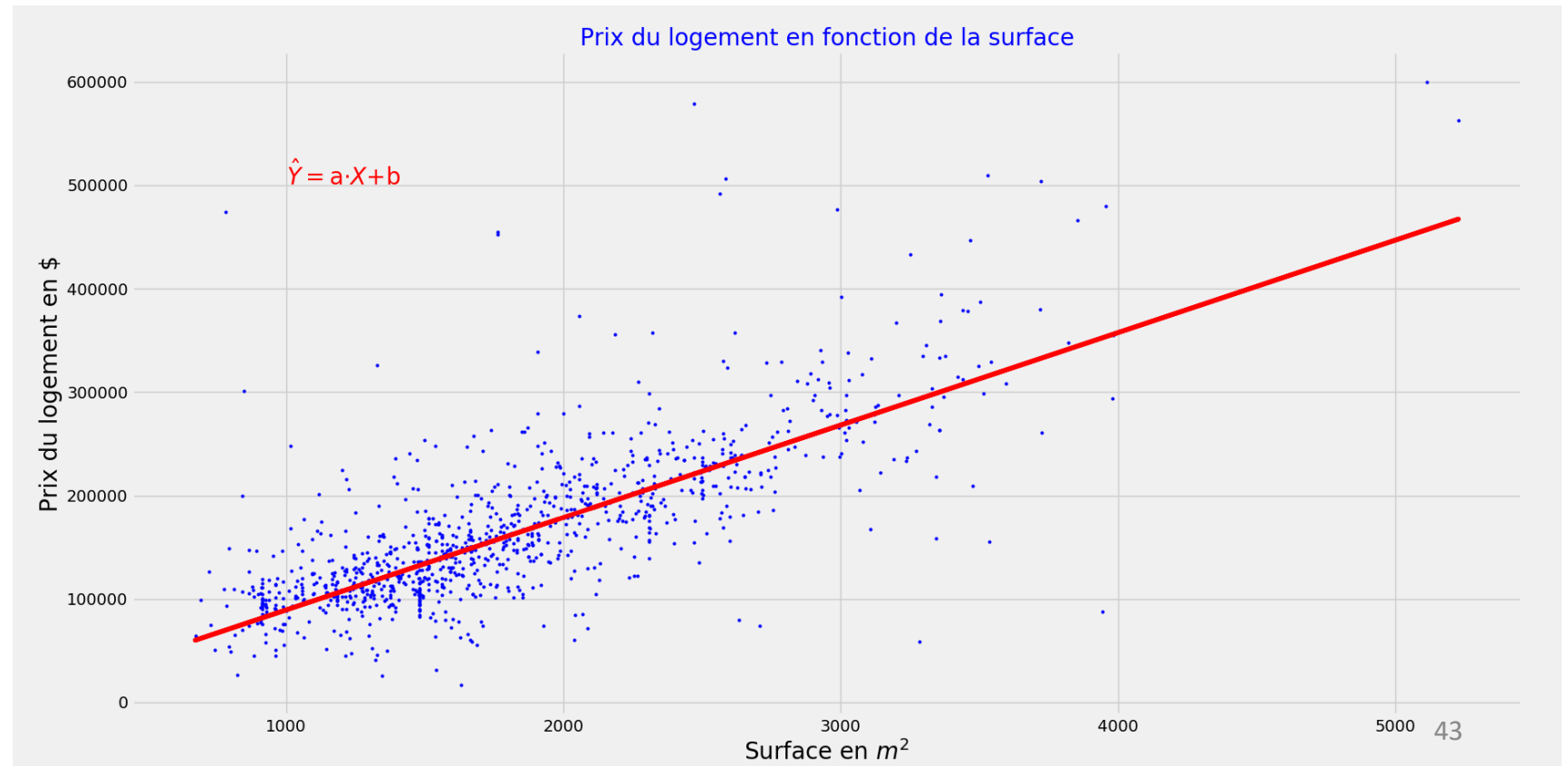


Deep Learning is about finding a good data representation with respect to an objective thanks to a composition of transformations (functions/layers)

Model

Data and Model

- What is a model ?
 - A machine learning model can be a mathematical representation of a real-world process.
 - $X \in \mathbb{R}^{m \times 1}$
 - $Y \in \mathbb{R}^{m \times 1}$
 - $\hat{Y} \in \mathbb{R}^{m \times 1}$



A Linear model

- $X \in \mathbb{R}^{m \times 1}$: input
- $W \in \mathbb{R}$
- $B \in \mathbb{R}$
- $\hat{Y} \in \mathbb{R}^{m \times 1}$: output (prediction)
- $\hat{Y} = XW + B$

$$[w_1] = W$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \begin{bmatrix} x_1 w_1 \\ x_2 w_2 \\ x_3 w_3 \end{bmatrix} = XW$$

A Linear model : for multi-dimensional input

- $X \in \mathbb{R}^{m \times d}$: input
- $W \in \mathbb{R}^{d \times 1}$
- $B \in \mathbb{R}$
- $\hat{Y} \in \mathbb{R}^{m \times 1}$: output (prediction)
- $\hat{Y} = XW + B$

$$\begin{bmatrix} w_a \\ w_b \end{bmatrix} = W$$

$$X = \begin{bmatrix} x_{1a} & x_{1b} \\ x_{2a} & x_{2b} \\ x_{3a} & x_{3b} \end{bmatrix} \begin{bmatrix} x_{1a}w_a + x_{1b}w_b \\ x_{2a}w_a + x_{2b}w_b \\ x_{3a}w_a + x_{3b}w_b \end{bmatrix} = XW$$

A Linear model : for multi-dimensional input and output

- $X \in \mathbb{R}^{m \times d}$: input
- $W \in \mathbb{R}^{d \times n}$
- $B \in \mathbb{R}^n$
- $\hat{Y} \in \mathbb{R}^{m \times n}$: output (prediction)
- $\hat{Y} = XW + B$

$$\begin{bmatrix} w_{1a} & w_{2a} \\ w_{1b} & w_{2b} \end{bmatrix} = W$$

$$X = \begin{bmatrix} x_{1a} & x_{1b} \\ x_{2a} & x_{2b} \\ x_{3a} & x_{3b} \end{bmatrix} \begin{bmatrix} x_{1a}w_{1a} + x_{1b}w_{1b} & x_{1a}w_{2a} + x_{1b}w_{2b} \\ x_{2a}w_{1a} + x_{2b}w_{1b} & x_{2a}w_{2a} + x_{2b}w_{2b} \\ x_{3a}w_{1a} + x_{3b}w_{1b} & x_{3a}w_{2a} + x_{3b}w_{2b} \end{bmatrix} = XW$$

A non-linear model : for multi-dimensional input/output

- $X \in \mathbb{R}^{m \times d}$: input
- $W \in \mathbb{R}^{d \times n}$
- $B \in \mathbb{R}^n$
- $R(a), \sigma(a)$: a non-linear function
- $\hat{Y} \in \mathbb{R}^{m \times n}$: output (prediction)
- $\hat{Y} = \sigma(XW + B)$

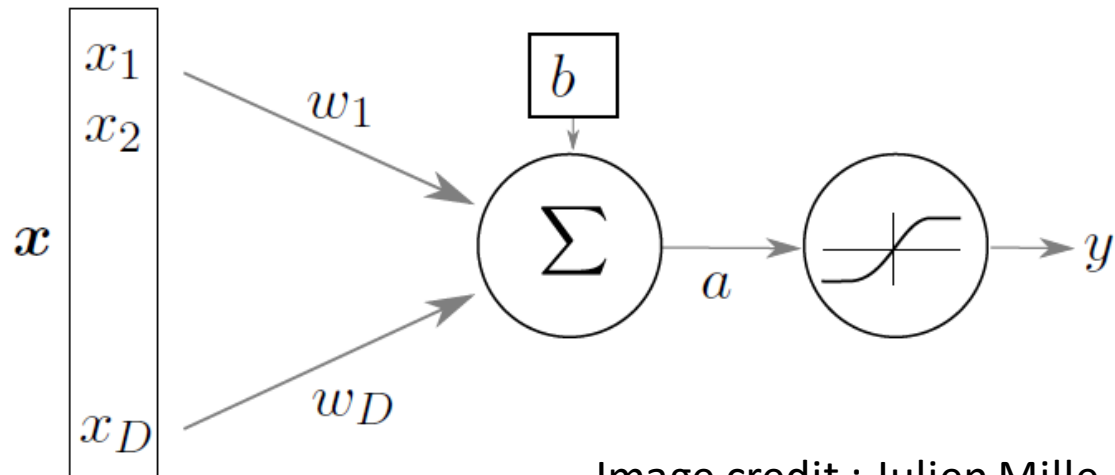
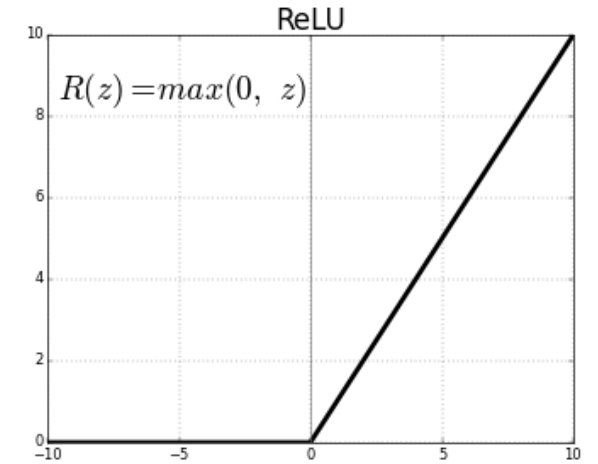
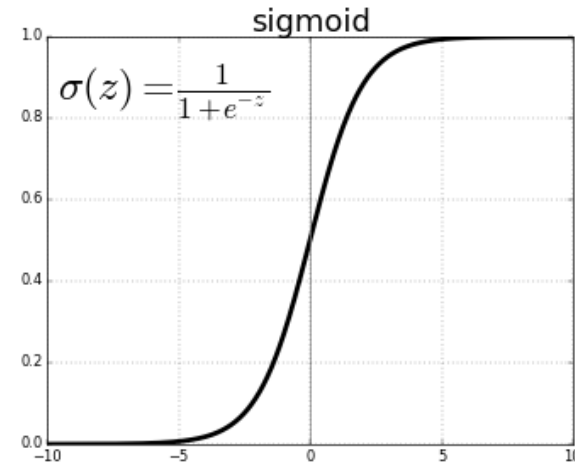
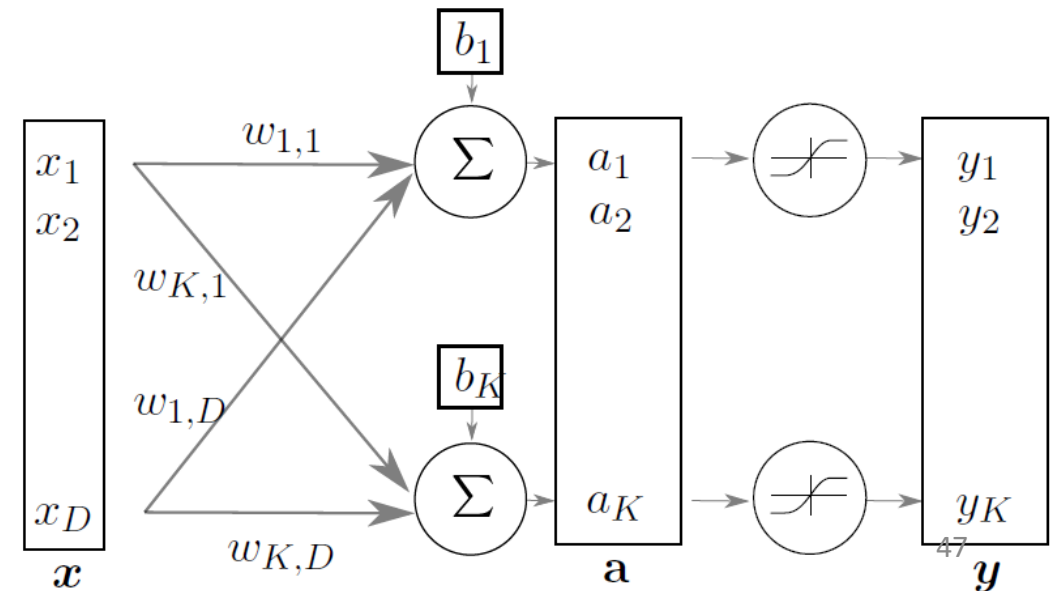


Image credit : Julien Mille



Can we stack the model ? Endomorphism property

- $X \in \mathbb{R}^{m \times d}$: input
- $W^{(1)} \in \mathbb{R}^{d \times n}$
- $B^{(1)} \in \mathbb{R}^n$
- $\widehat{Y}^{(1)} \in \mathbb{R}^{m \times n}$: intermediate results
- $W^{(2)} \in \mathbb{R}^{n \times o}$
- $B^{(2)} \in \mathbb{R}^o$
- $\widehat{Y}^{(2)} \in \mathbb{R}^{m \times o}$: output (prediction)
- $\widehat{Y}^{(1)} = \sigma(X W^{(1)} + B^{(1)})$
- $\widehat{Y}^{(2)} = \sigma(\widehat{Y}^{(1)} W^{(2)} + B^{(2)})$
- $\widehat{Y}^{(2)} = \sigma(\sigma(X W^{(1)} + B^{(1)}) W^{(2)} + B^{(2)})$
- $\widehat{Y}^{(l)} = f(\widehat{Y}^{(l-1)}, W^{(l)}, B^{(l)}) = f(\widehat{Y}^{(l-1)}, \theta^{(l)})$
- $\widehat{Y}^{(l)} = f(f(X, \theta^{(l-1)}), \theta^{(l)})$: *Composition of functions*
- $F(X, \Theta) = f_{\theta^{(l)}} \circ f_{\theta^{(l-1)}}$: *Composition of functions*

Multi Layer Perceptron
: MLP
A sequence of linear
operations and non
linear operations

Why these non-linear operations?

Universal Approximation Theorem

Neural nets (MLP) as universal approximators

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0, 1]^p \rightarrow \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ s.t.

$$\left| \sum_{i=1}^n a_i \xi(\mathbf{w}_i^\top \mathbf{f} + b_i) - y(\mathbf{f}) \right| < \epsilon \quad \forall \mathbf{f} \in [0, 1]^p$$

- ☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer
- ☹ How many neurons?
- ☹ How to find the parameters?
- ☹ Does it generalize well / overfit?

MLP and images

- $X \in \mathbb{R}^{m \times d}$: input
- $X_i \in \mathbb{R}^{1 \times d}$
- d : 640x480 pixels = 307200
- $W \in \mathbb{R}^{d \times n}$
- $B \in \mathbb{R}^n$; $\sigma(a)$; $\hat{Y} \in \mathbb{R}^{m \times n}$: output (prediction)
- $\hat{Y} = \sigma(XW + B)$
- Issues :
 - W : is huge (one parameter by pixel)
 - An image is reduced to a vector :
 - Spatiality is lost

Convolutional Neural Networks : CNN

- CNN to the rescue

CNN : Prior knowledge

- Integrate « Prior knowledge » into a specific (handmade) architecture
- Prior knowledge :
 - Self similarity : Data (image) are self similar across domain.
 - Locality : Pixel position and neighborhood have semantic meanings. Nearby pixels are more related than distant ones.
 - Objects are built up out of smaller parts : Compositionality
 - Translation invariance (image classification tasks)
- CNN are well suited for N dimensional arrays (Tensors)
 - Successful to deal with images, videos, sounds, LIDAR...

Self similarity



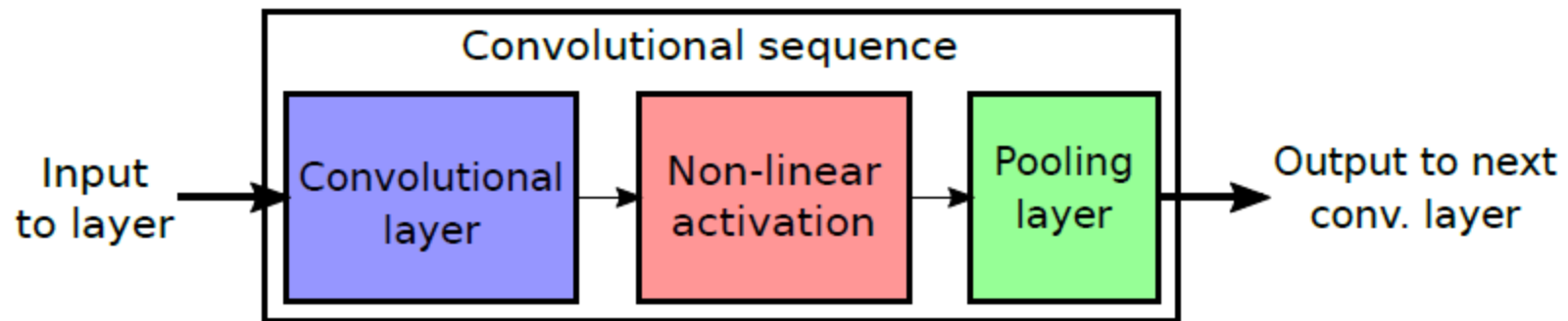
Data is self-similar across the domain

Image credit : <http://geometricdeeplearning.com/>

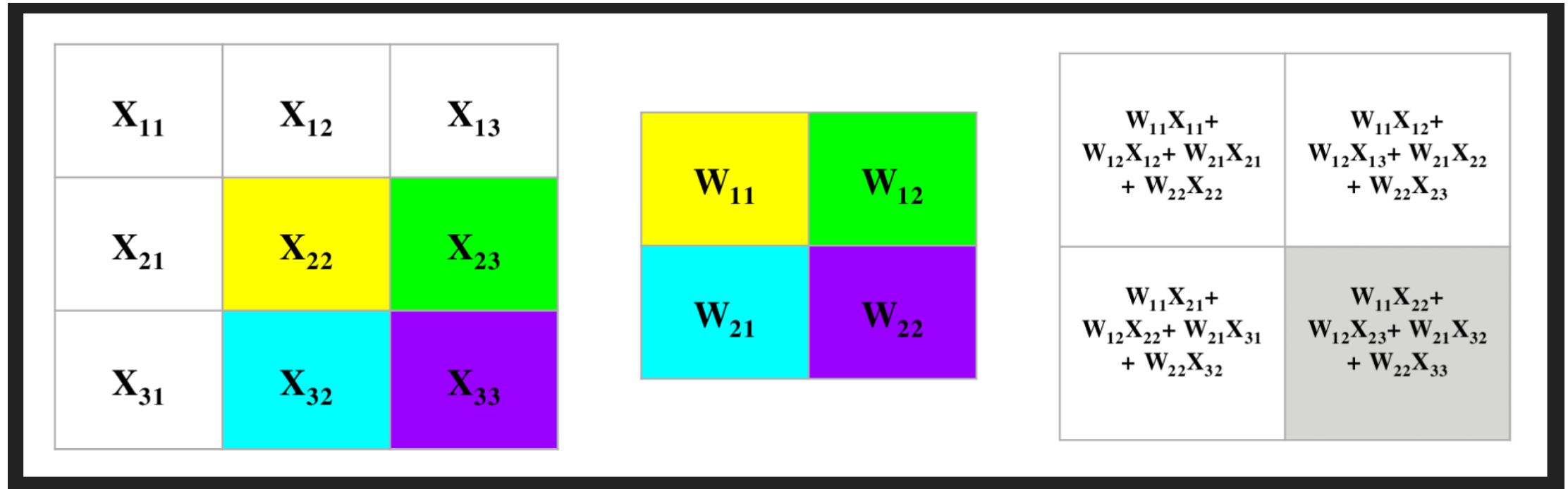
Translation invariance (image classification tasks)



CNN : How it works



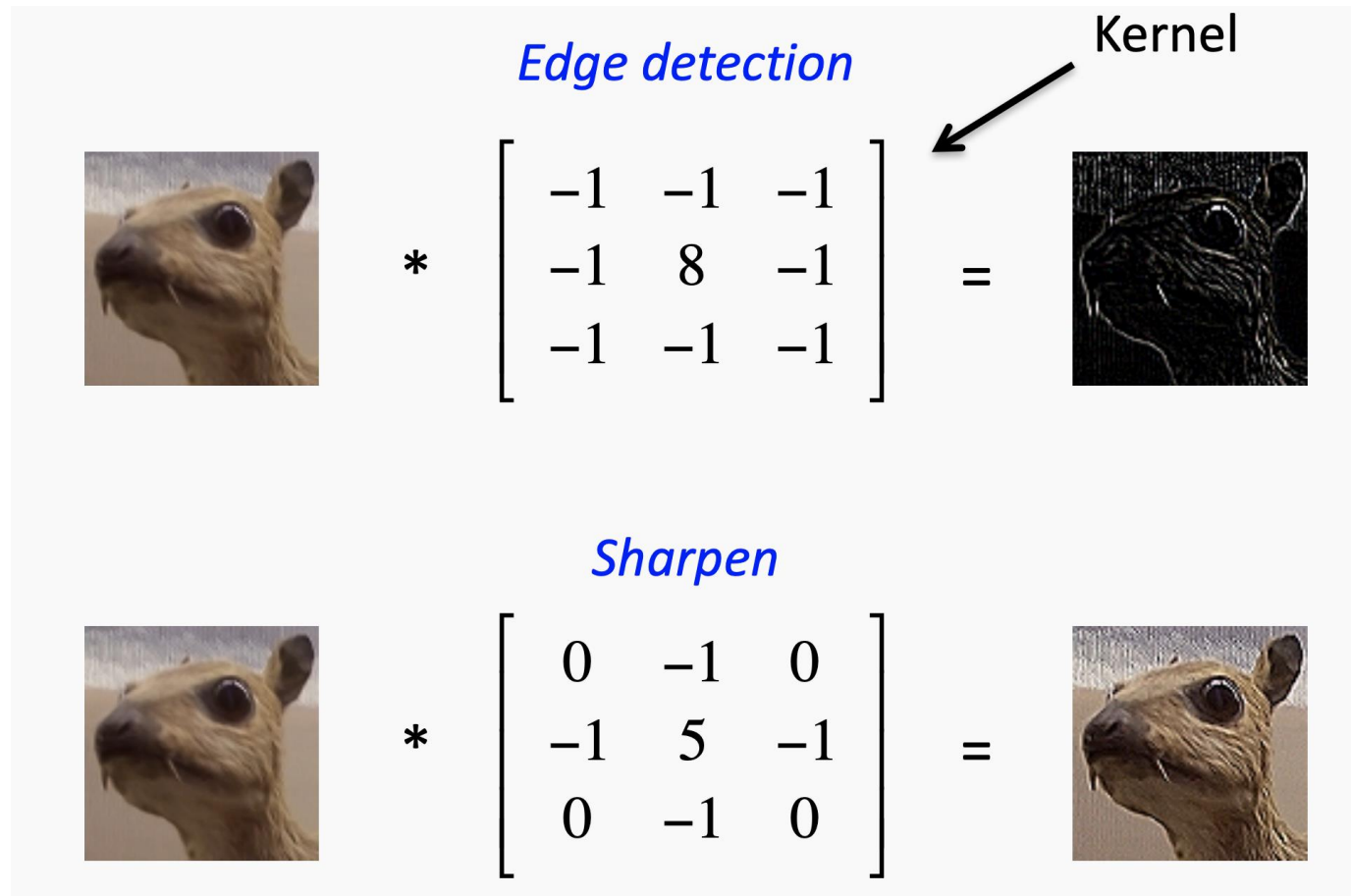
CNN : Convolutional Layer



Convolution is a linear operation.

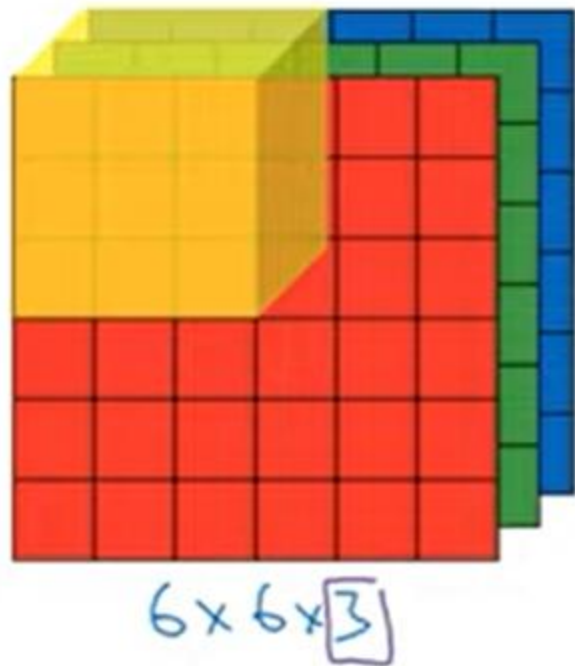
<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

CNN : Convolution with fixed parameters values

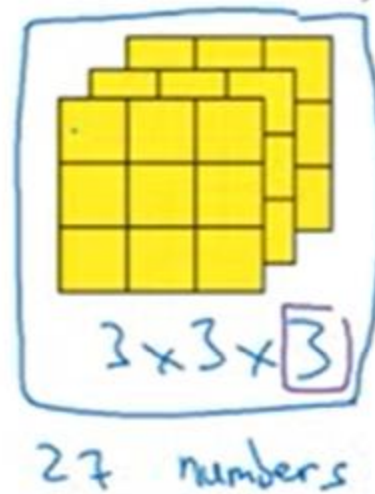


Taken from deeplearning.ai

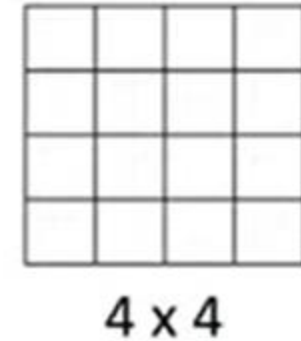
Convolutions on RGB image



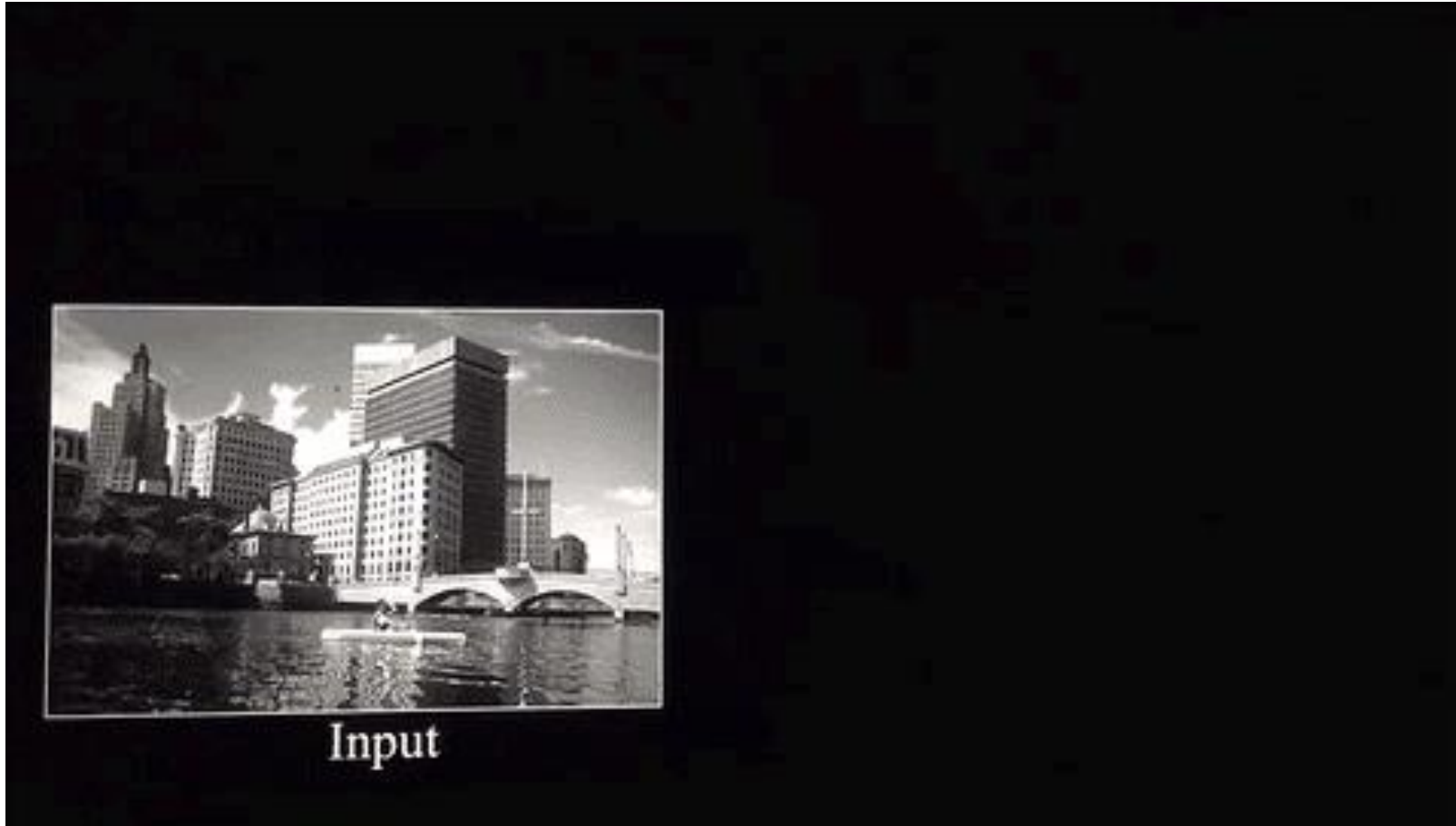
*



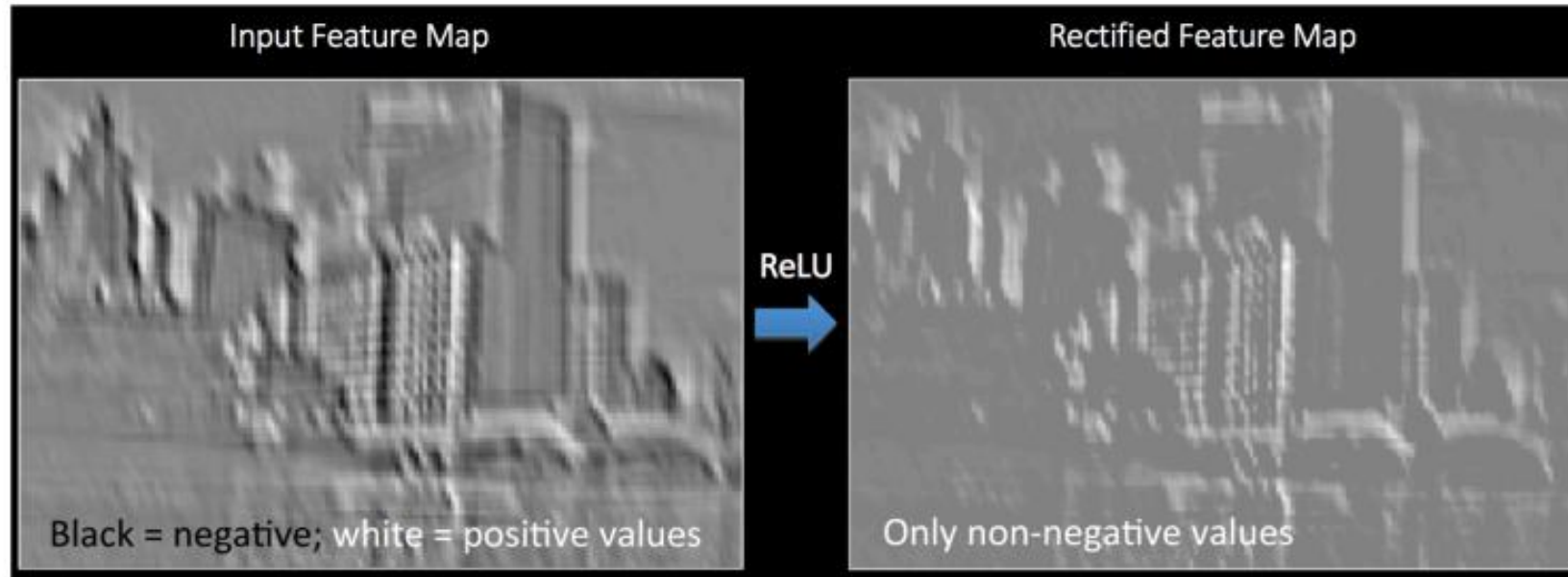
=



CNN : Convolution

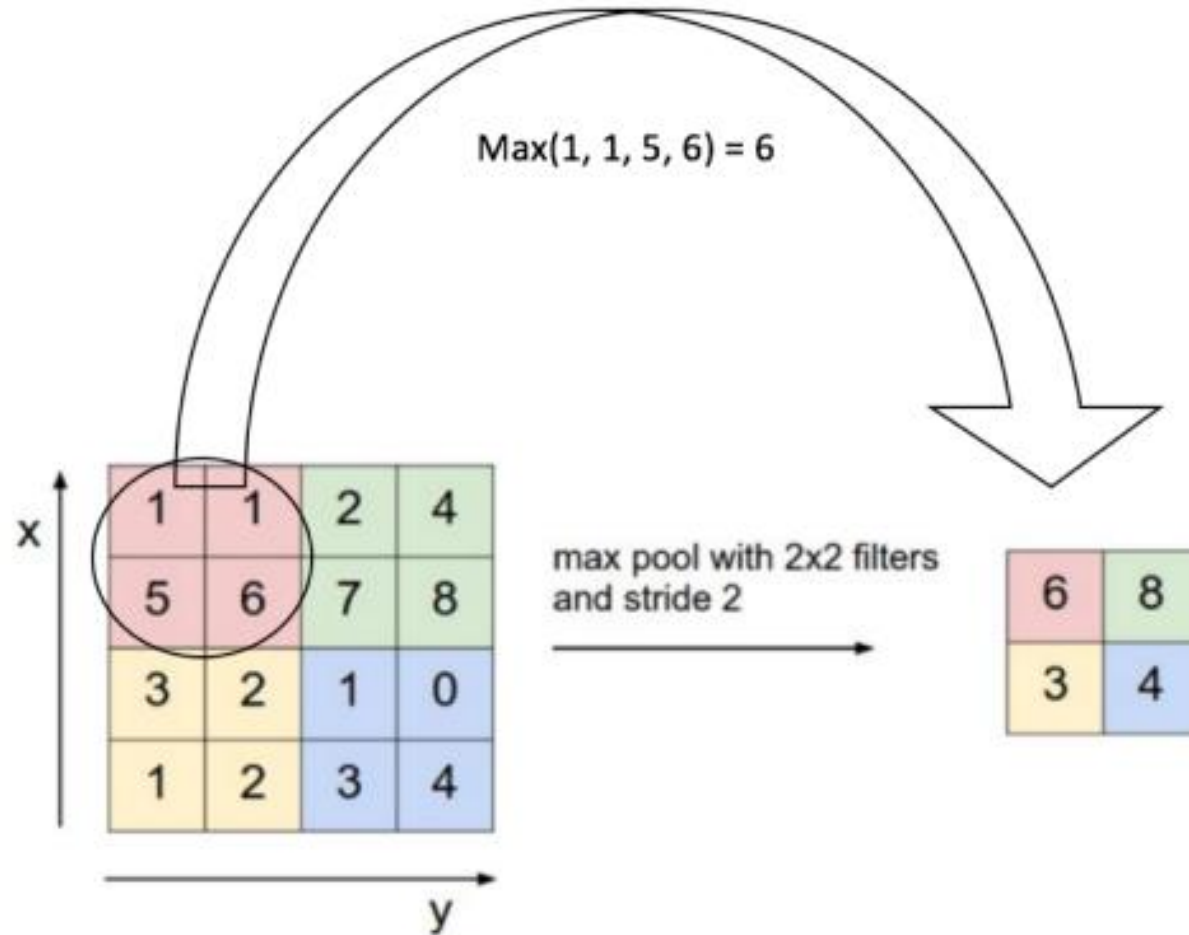


CNN : non-linear activation



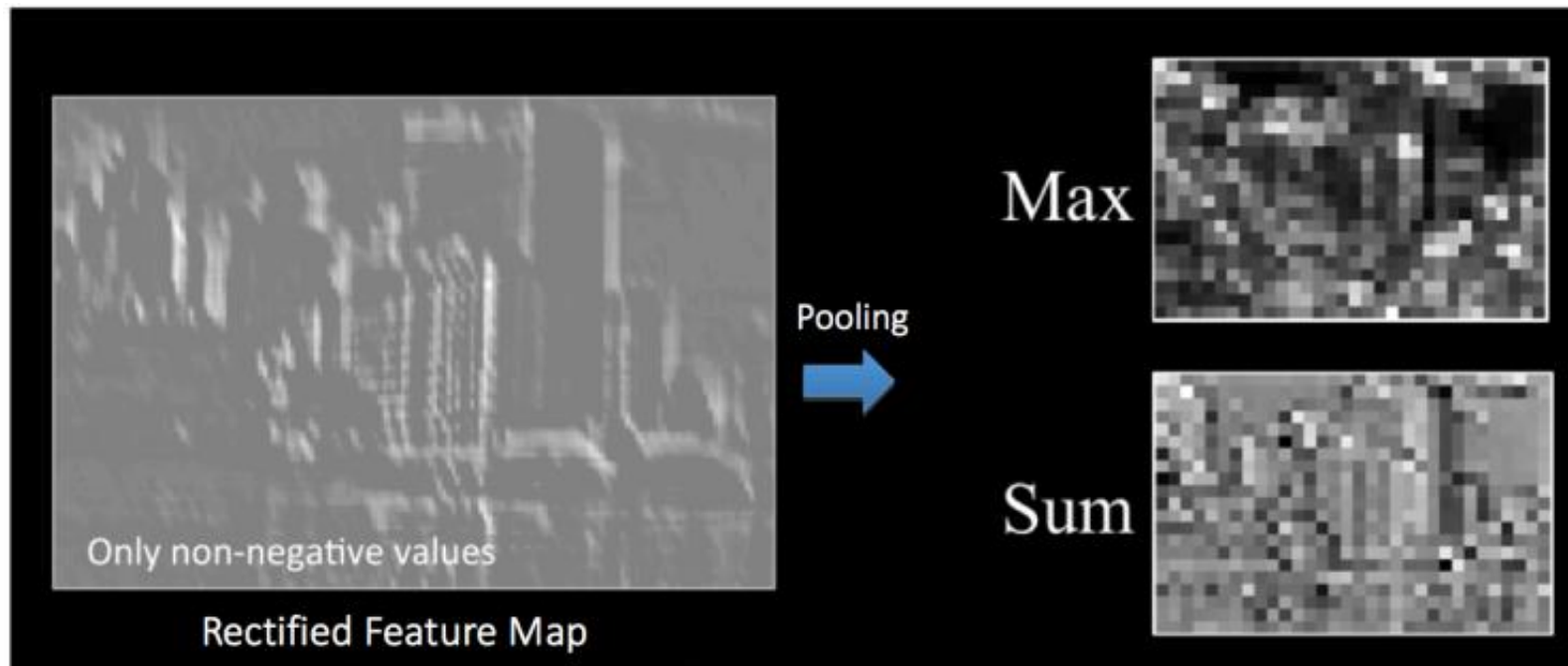
<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

CNN : pooling



Rectified Feature Map

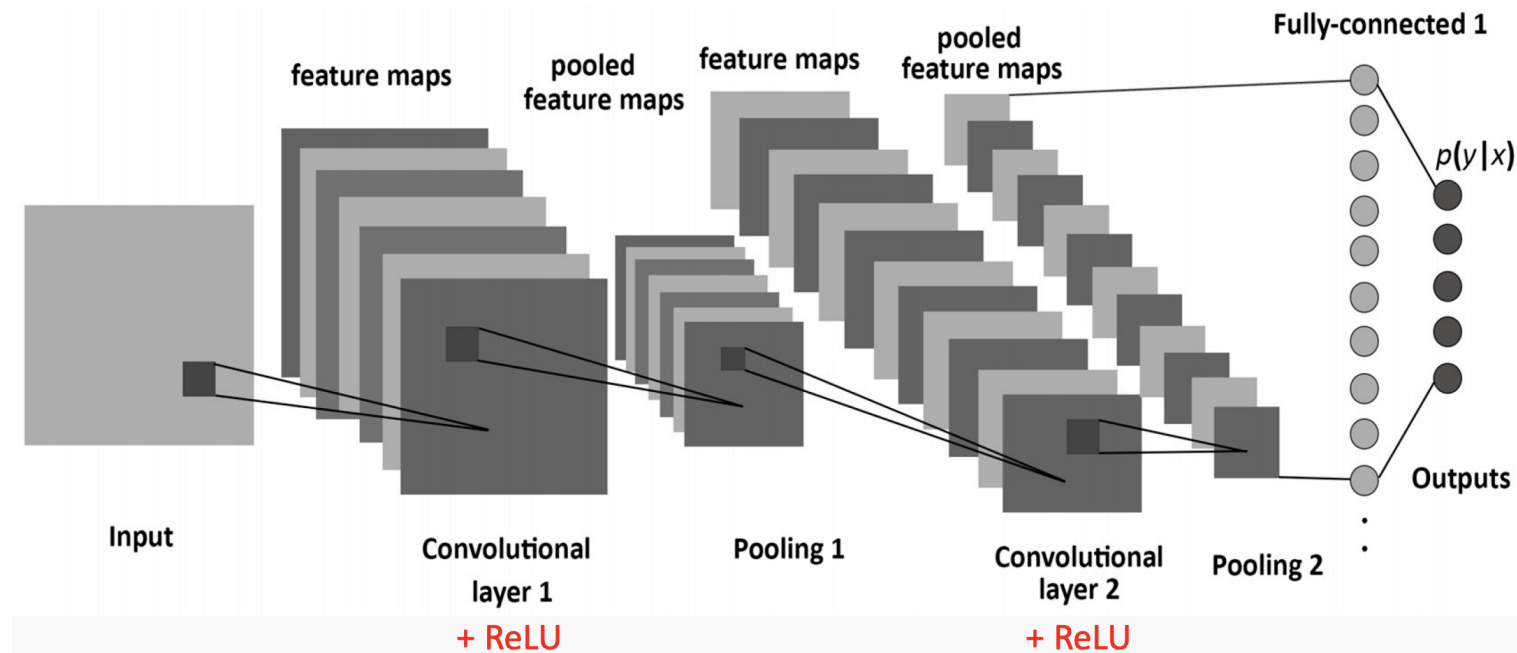
CNN : pooling



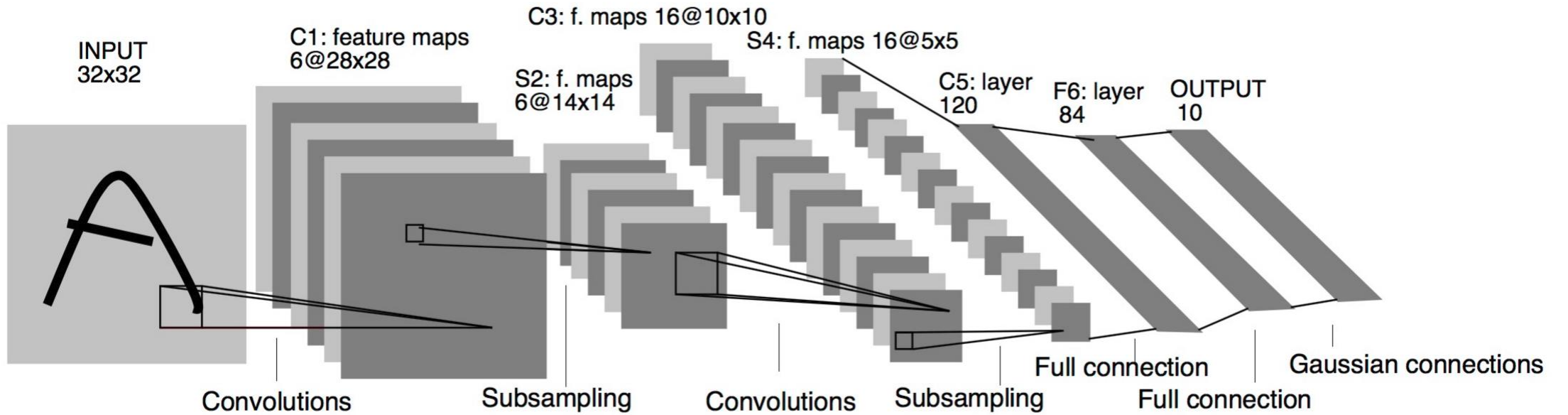
<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

CNN : Architecture

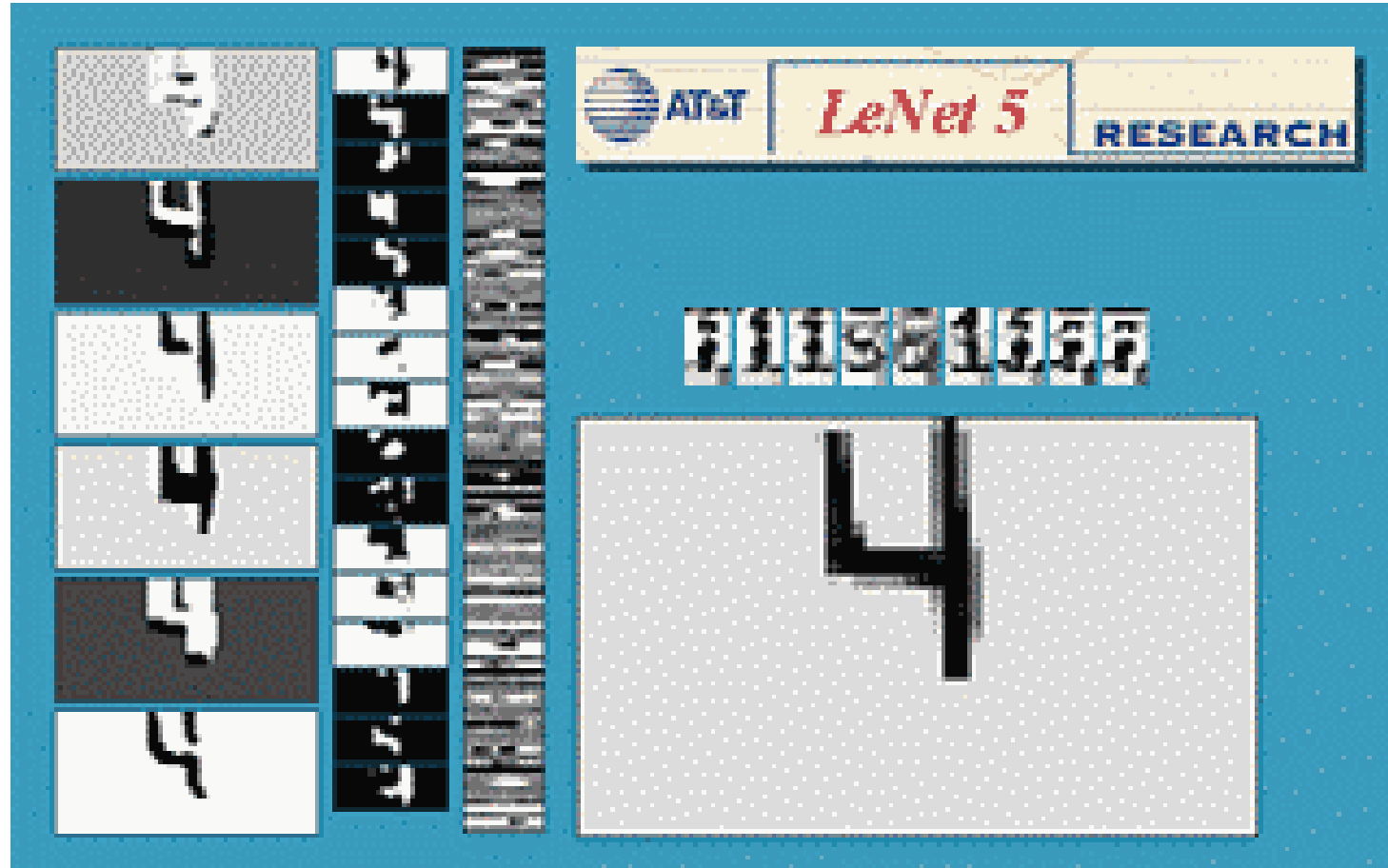
- Stack-up with convolution layers
- Stack-up with fully connected layers (Multi Layer Perceptron : MLP)



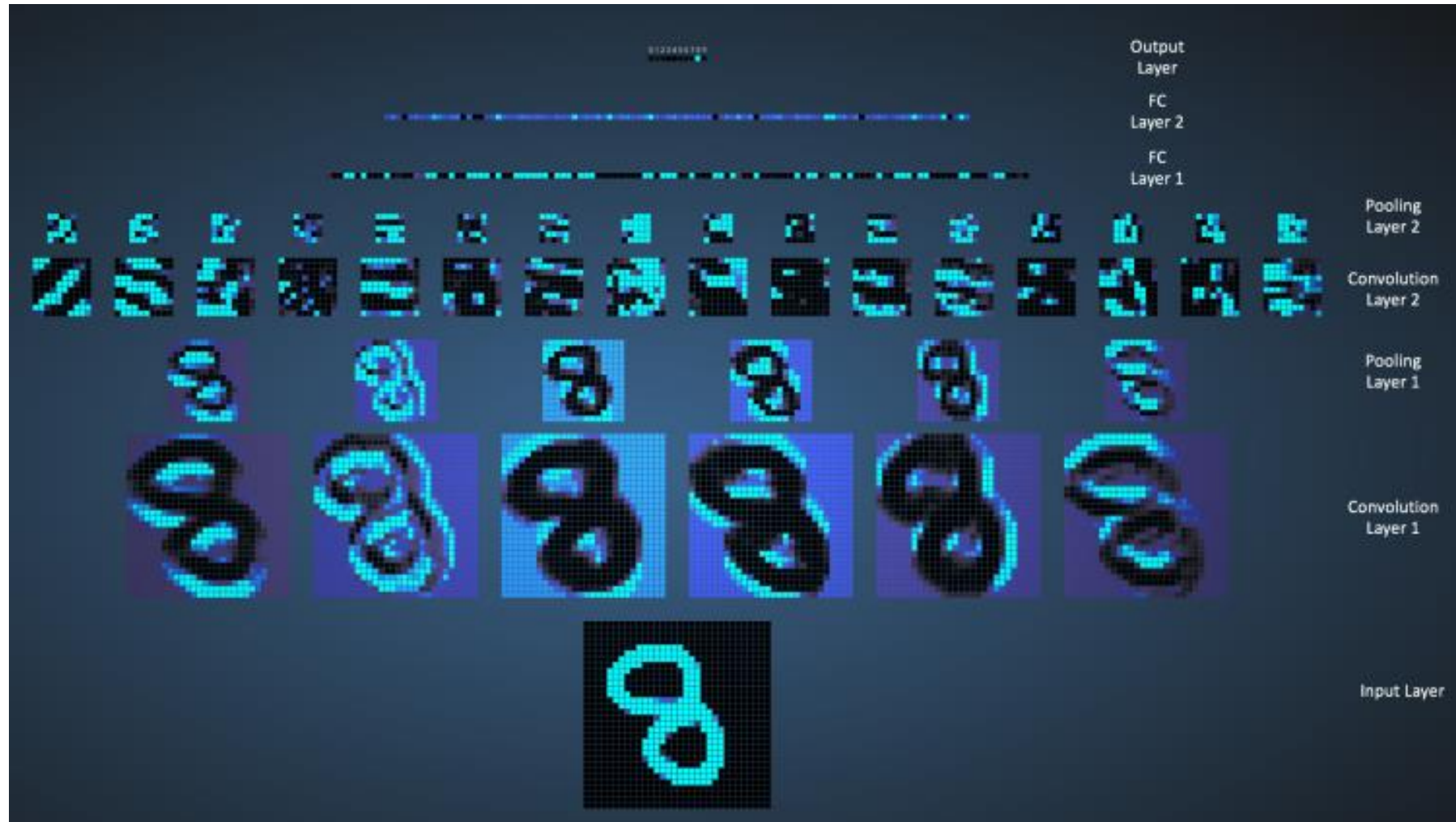
CNN : Architecture : LeNet5



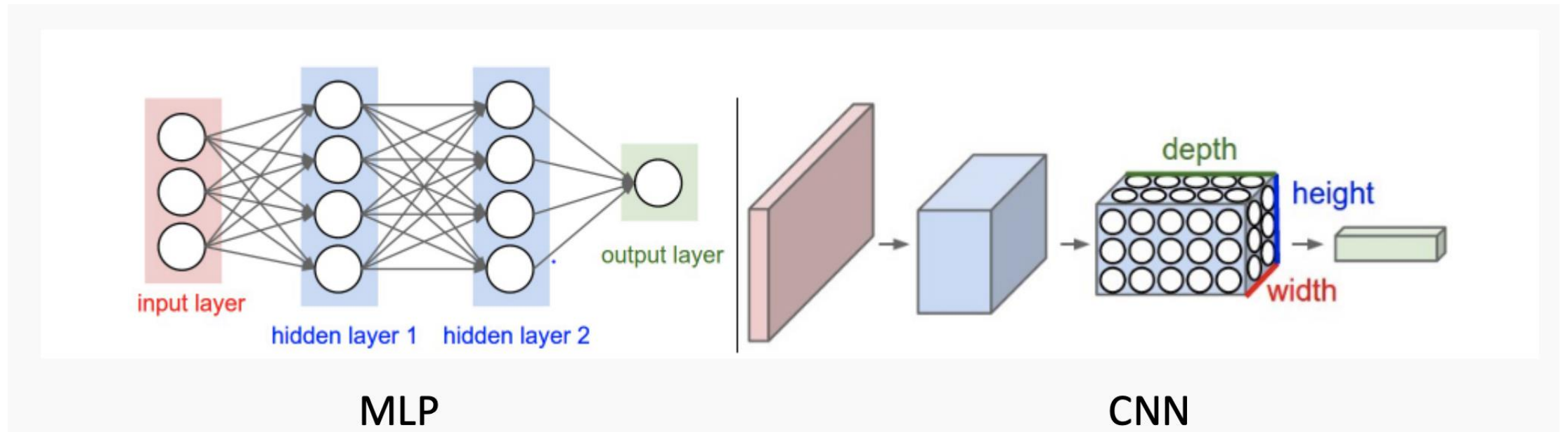
CNN : Architecture : LeNet5



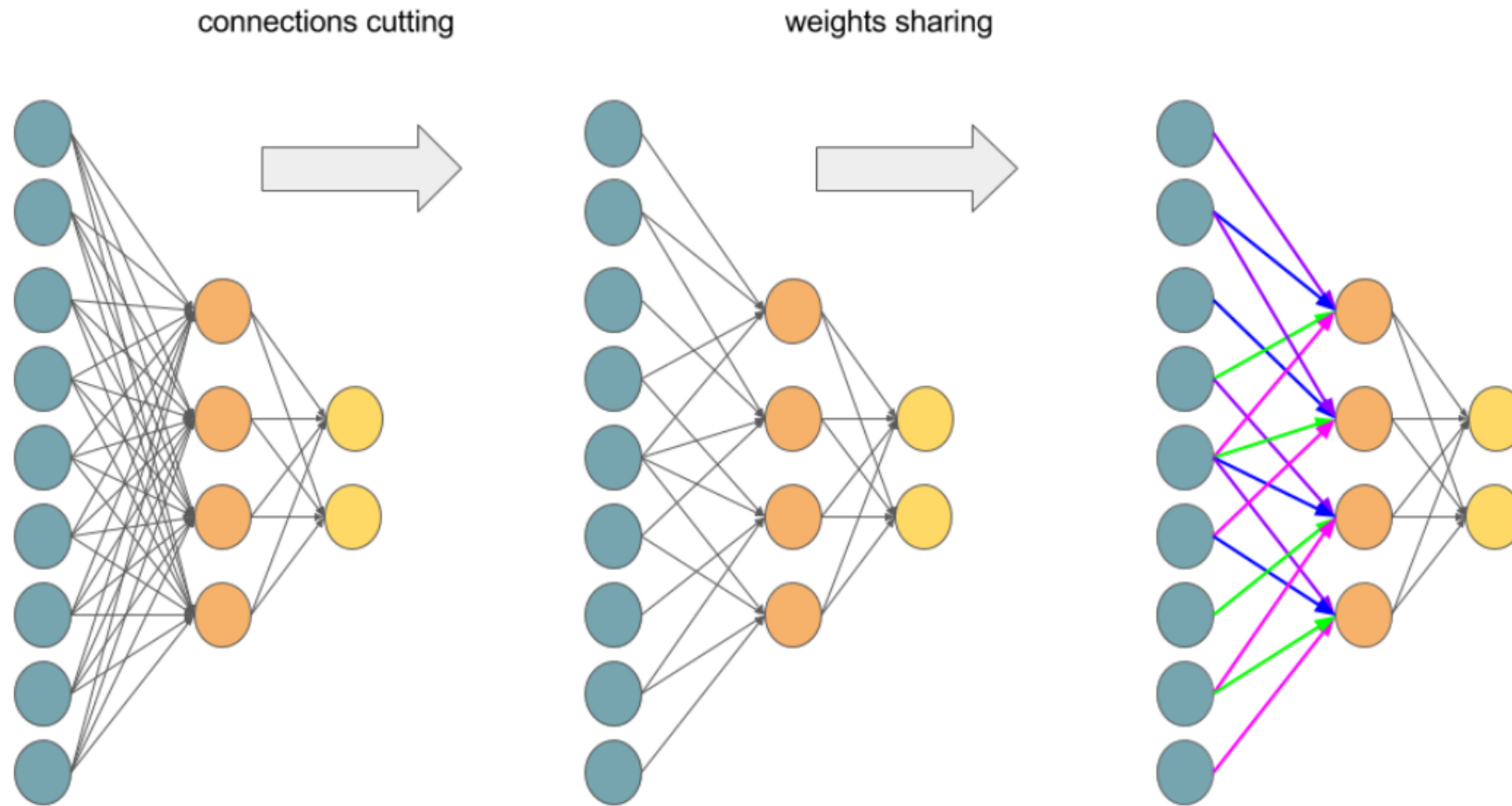
CNN : Architecture



MLP and CNN : Any link ?



From MLP to CNN



Back to CNN and prior knowledge

Prior knowledge	CNN operation
Self similarity (Stationarity)	Shared parameters
Locality	Local connection
Translation invariance	Convolution
Compositionality	Stackable

Other models : recurrent neural networks

- To model sequences :

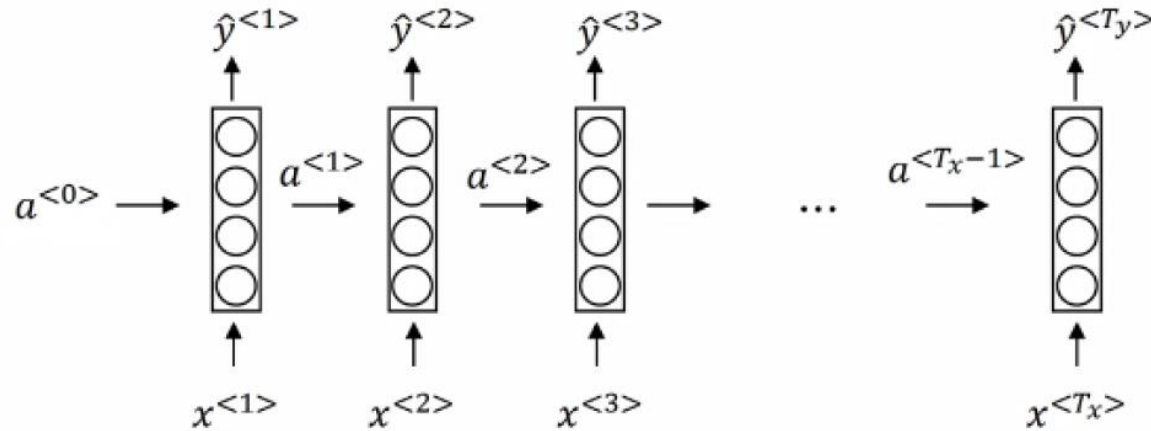


Image from : <https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/12/06022525/bppt.png>

- Any many others :
 - <https://www.asimovinstitute.org/neural-network-zoo/>

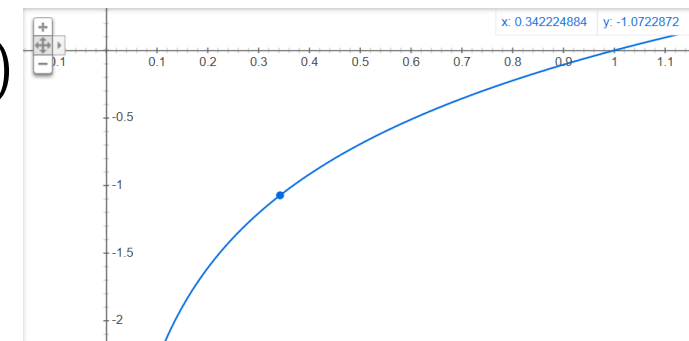
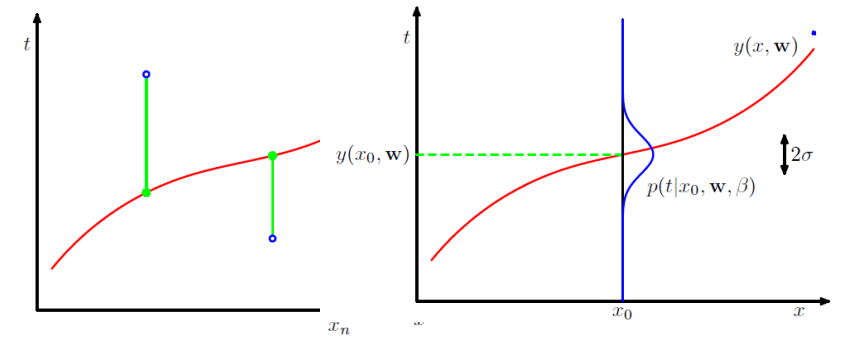
Supervised learning of a model

Learn a model : Supervised learning

- Learn a model :
 - Find the parameters of a model : Θ
- To achieve a task :
 - Classification :
 - To minimize an error of classification
 - Detection :
 - To minimize an error of detection
- Trainingset :
 - Pairs of input and output : X and Y :
 - $X \in \mathbb{R}^{(m \times d)}$: input
 - $\hat{Y} \in \mathbb{R}^{(m \times n)}$: output (prediction)
 - $Y \in \mathbb{R}^{(m \times n)}$: The ground truth
- Achieve a task = optimizing a function
 - To find the minimum of a function for instance: $\min_{\Theta} L(X, Y, \Theta)$
 - L is called the loss function or cost function
- Learn a model = an optimization problem
 - $\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} L(X, Y, \Theta)$

Loss functions

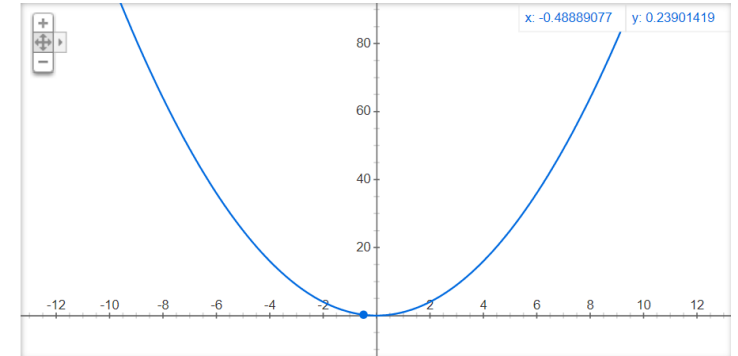
- There are a lot of them
- They are important, they provide the « sens » of the learning
- Let us see two of them : $\hat{Y} = F(X, \Theta)$
 - Least square loss : Good for regression problems
 - $L(X, Y, \Theta) = ||Y - F(X, \Theta)||_2^2$
 - Cross entropy loss : Good for classification problems
 - 2 classes : Cats vs Dogs
 - $L(X, Y, \Theta) = \sum_{i=1}^m -Y_i \log(F(X_i, \Theta)) + (1 - Y_i) \log(1 - F(X_i, \Theta))$
 - Example : $Y_0=1$ et $F(X_0, \Theta) = 0.1$



In the context of a discriminative model for probabilistic classification, minimizing the cross entropy is equivalent to maximize the likelihood

Minimizing the loss function

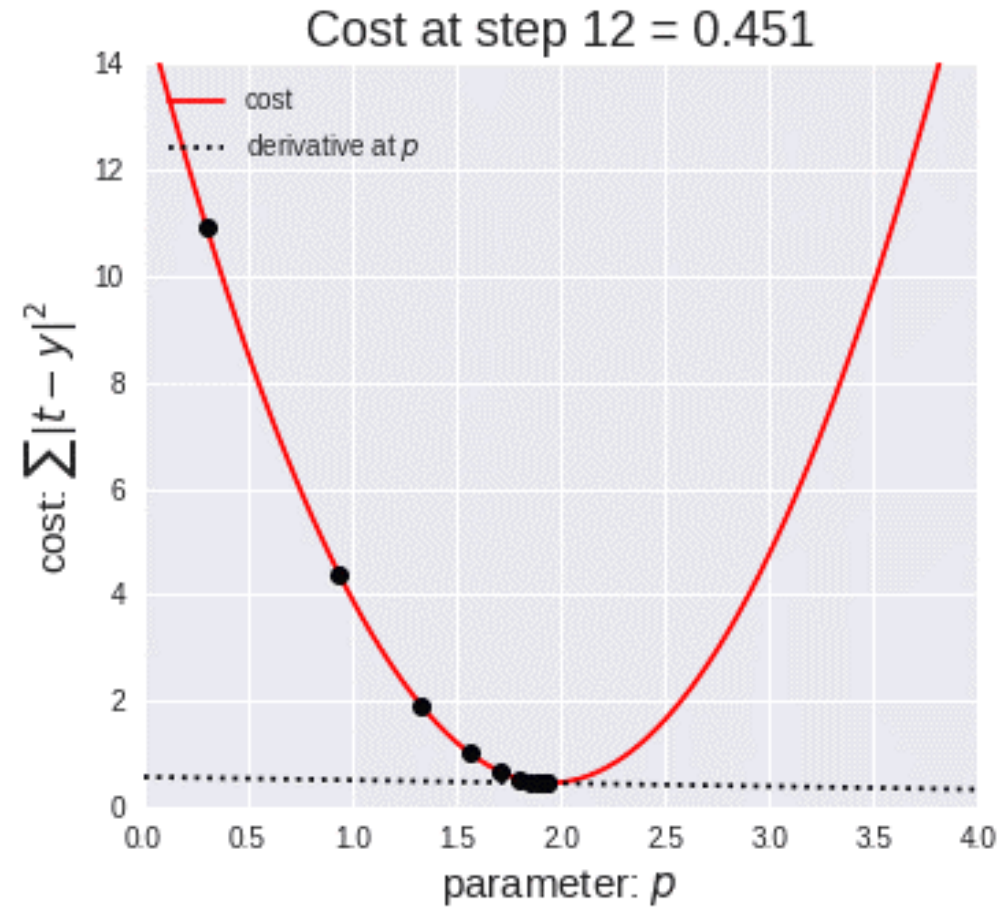
- $\min_{\Theta} L(X, Y, \Theta)$
- How ?
 - Find where the derivative equal to zero with respect to Θ
 - $\frac{d L(X, Y, \Theta)}{d \Theta} = 0$
 - If L is convex then we find the global minimum
 - If L is not convex then we find a local minimum
- Convexity depends on the model and the loss function
 - Deep Learning models are not convex



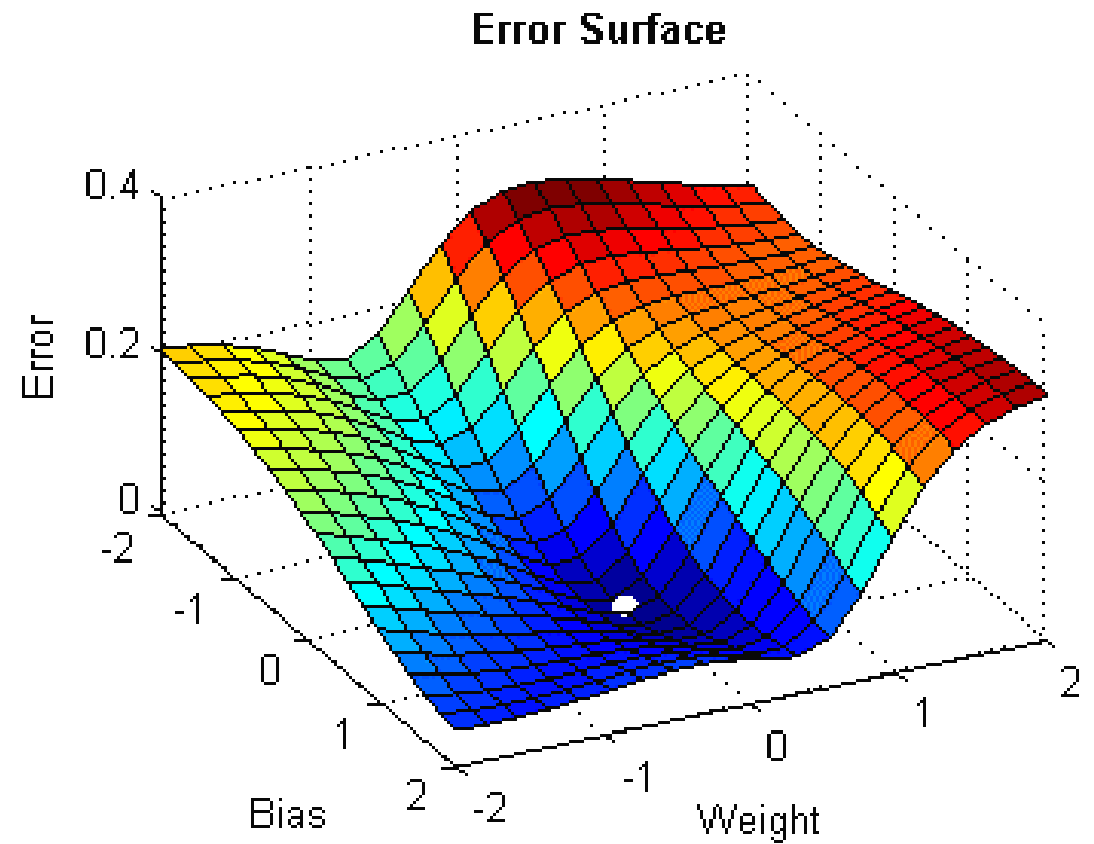
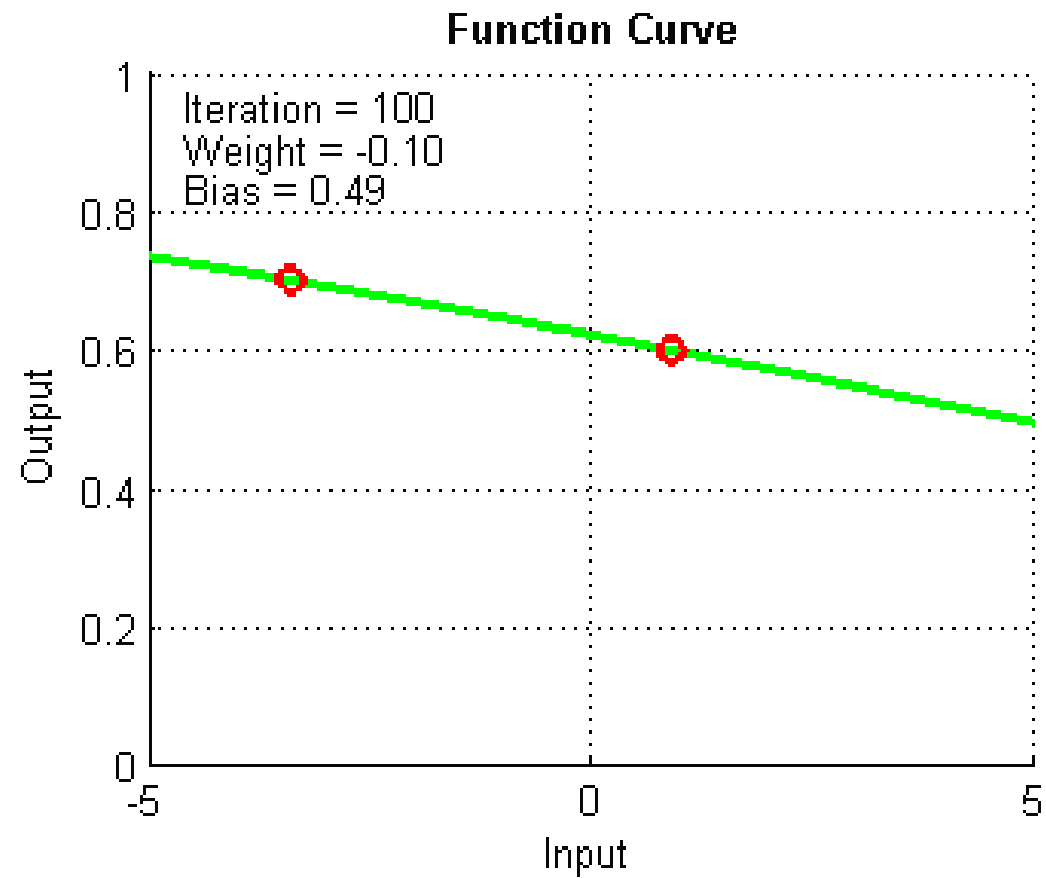
How to find where derivative equal to 0

- How to find : $\frac{d L(X,Y,\Theta)}{d \Theta} = 0$
- By an iterative algorithm called : Gradient Descent
 - $\Theta^{t+1} = \Theta^t - \alpha \frac{d L(X,Y,\Theta^t)}{d \Theta^t}$
 - « t » indicates a given iteration
 - where α is the learning rate
 - Stop to iterate when a criterion is met :
 - A number of iterations for instance

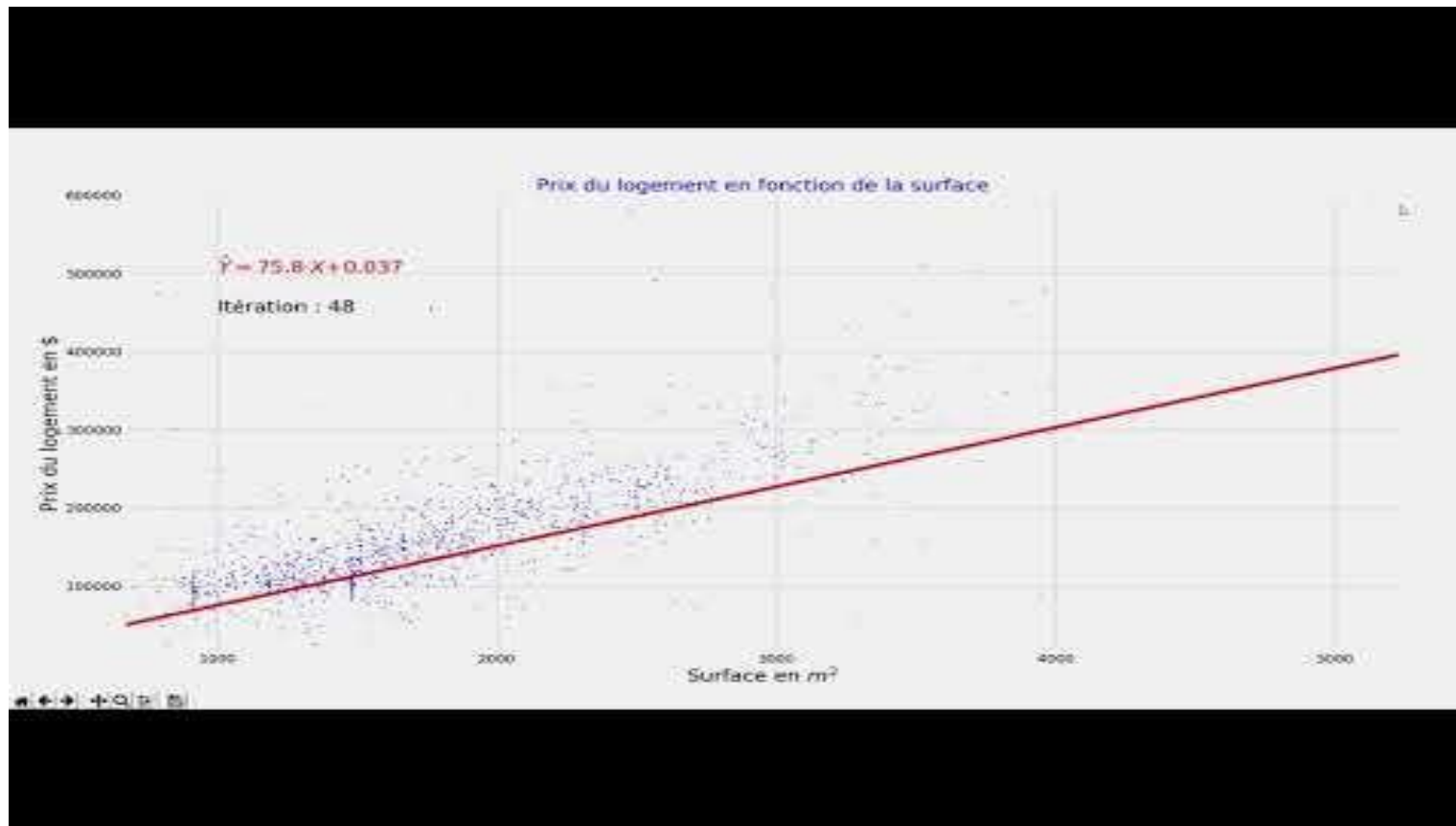
Gradient Descent



Gradient Descent



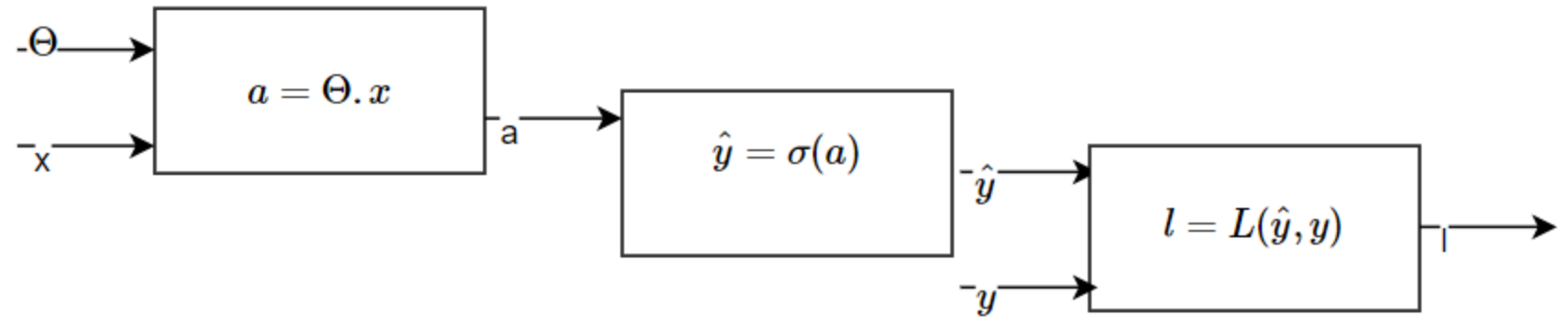
Gradient Descent applied to a Linear Model



How to compute the derivative

- How to compute $\frac{d L(X,Y,\Theta)}{d \Theta}$
- The function L is a composition of functions
- The derivative of composed functions can be achieved by the chain rule also called back propagation
- Example : $p(q(x))$ (*another notation* : $p \circ q$).
 - Let us note : $z = q(x)$
 - $\frac{dp(q(x))}{dx} = \frac{dp(z)}{dz} \cdot \frac{dq(x)}{dx}$

How to compute the derivative : Computation graph



$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$L(\hat{y}, y) = -y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})$$

$$\frac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$$

$$\frac{d \log(\hat{y})}{d\hat{y}} = \frac{1}{\hat{y}}$$

$$\frac{dL}{d\Theta} = \frac{dL}{d\hat{y}} \cdot \frac{d\sigma}{da} \cdot \frac{d \Theta \cdot x}{d\Theta}$$

$$\frac{dL}{d\Theta} = \left[\frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right] \cdot [\sigma(a)(1 - \sigma(a))] \cdot [x]$$

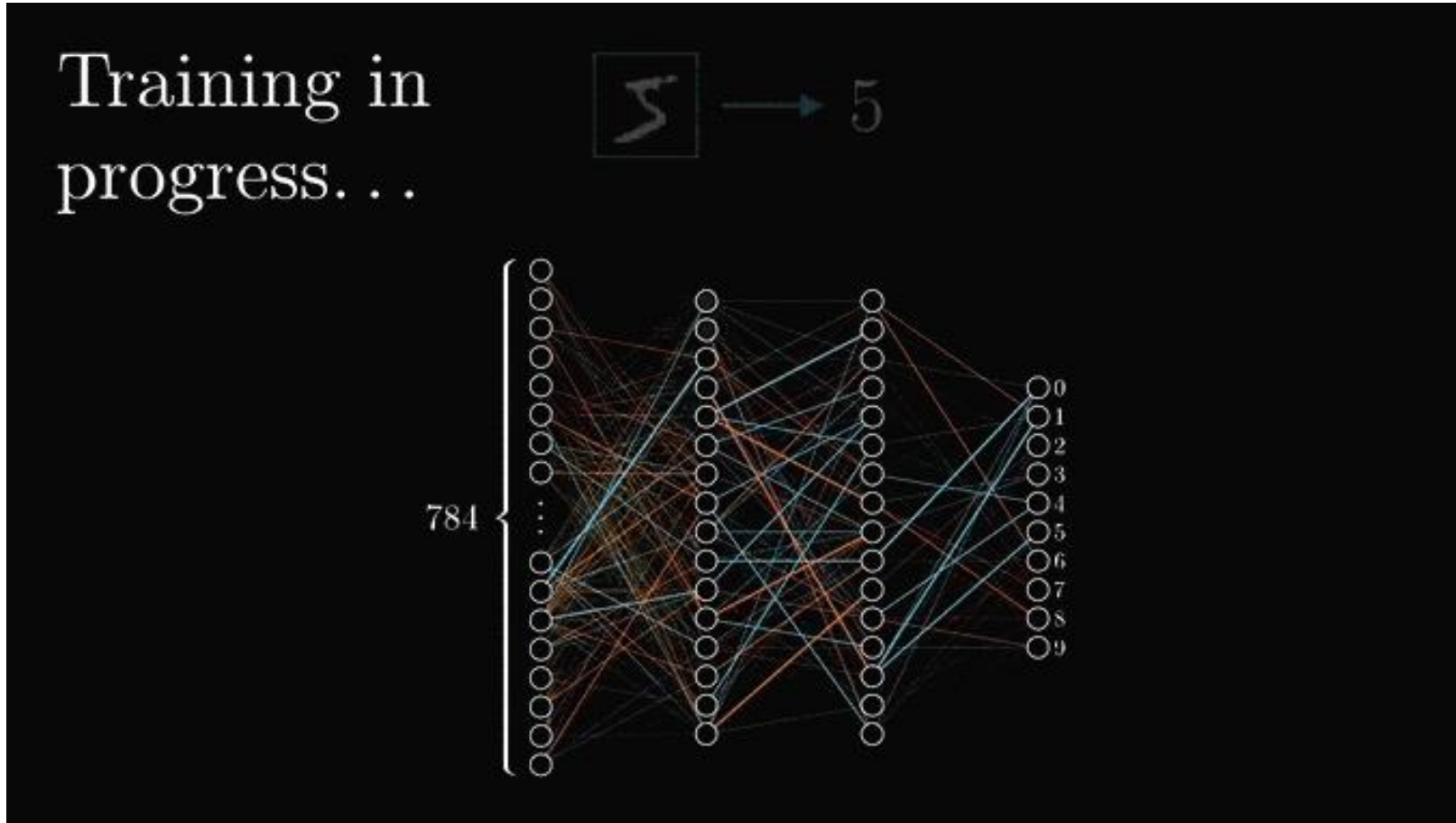
$$\frac{dL}{d\Theta} = \left[\frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right] \cdot [\hat{y}(1 - \hat{y})] \cdot [x]$$

$$\frac{dL}{d\Theta} = (\hat{y} - y) \cdot x$$

$$\frac{dL}{d\hat{y}} = \frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{d \Theta \cdot x}{d\Theta} = x$$

Training a neural net : Forward and Backward

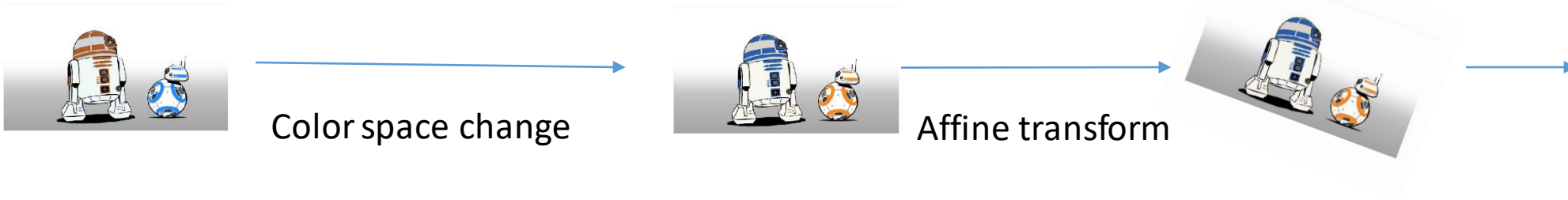


Gradient descent and back propagation

- Any model can be learned by Gradient Descent :
 - Each function of the network should be differentiable
 - Automatic Differentiation can help (AutoGrad)
 - The computation graph should not hold loops
- Many variations :
 - Batch gradient descent, Stochastic gradient descent, Stochastic minibatch gradient descent <http://romain.raveaux.free.fr/document/GradientDescent.html>
 - Adam, RMSprop → see

Many libraries

- <https://www.tensorflow.org/> : Google
- <https://pytorch.org/> : FaceBook
 - <https://kornia.github.io/> : Differentiable computer vision

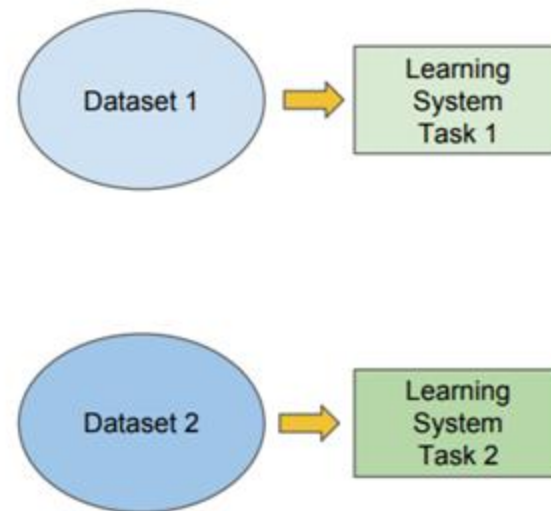


Good stuff in deep learning

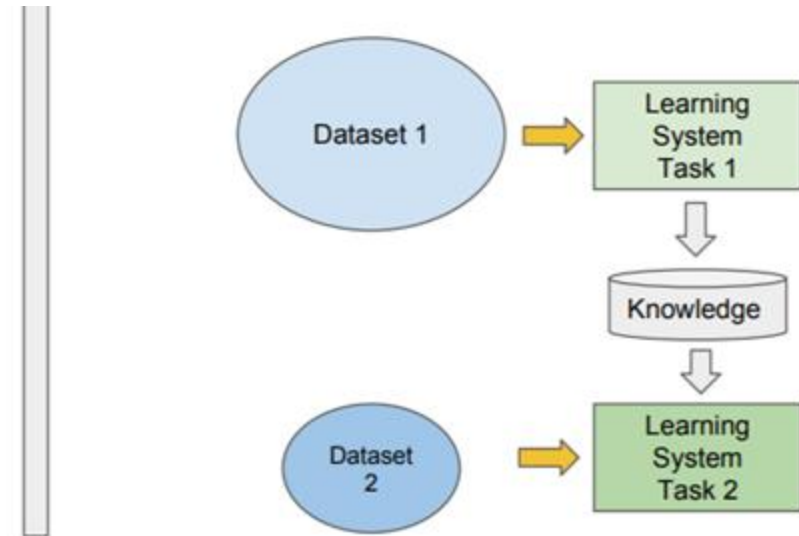
- Shared Models
- Competition of models
- Self supervised learning

Share your models

- Transfer learning
 - Take a model trained on a given problem
 - Adapt it to a particular problem
- Possible because : Deep Neural Networks (learned) parameters are generics

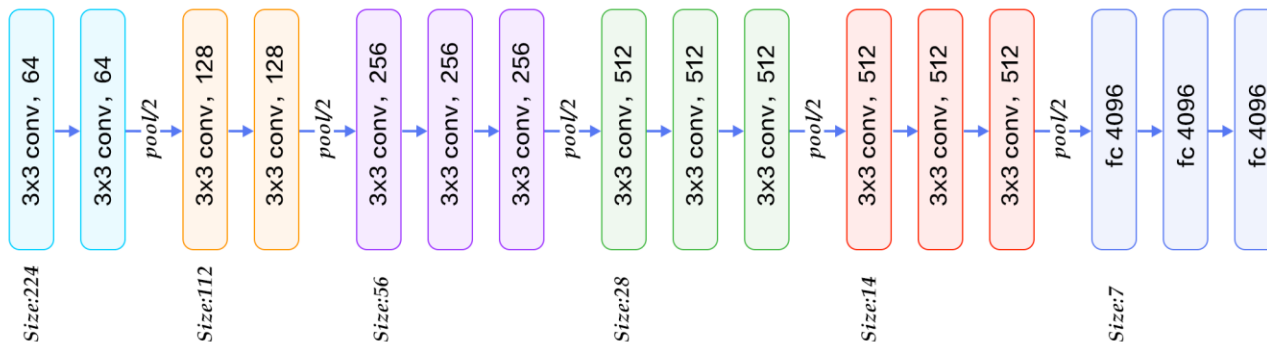
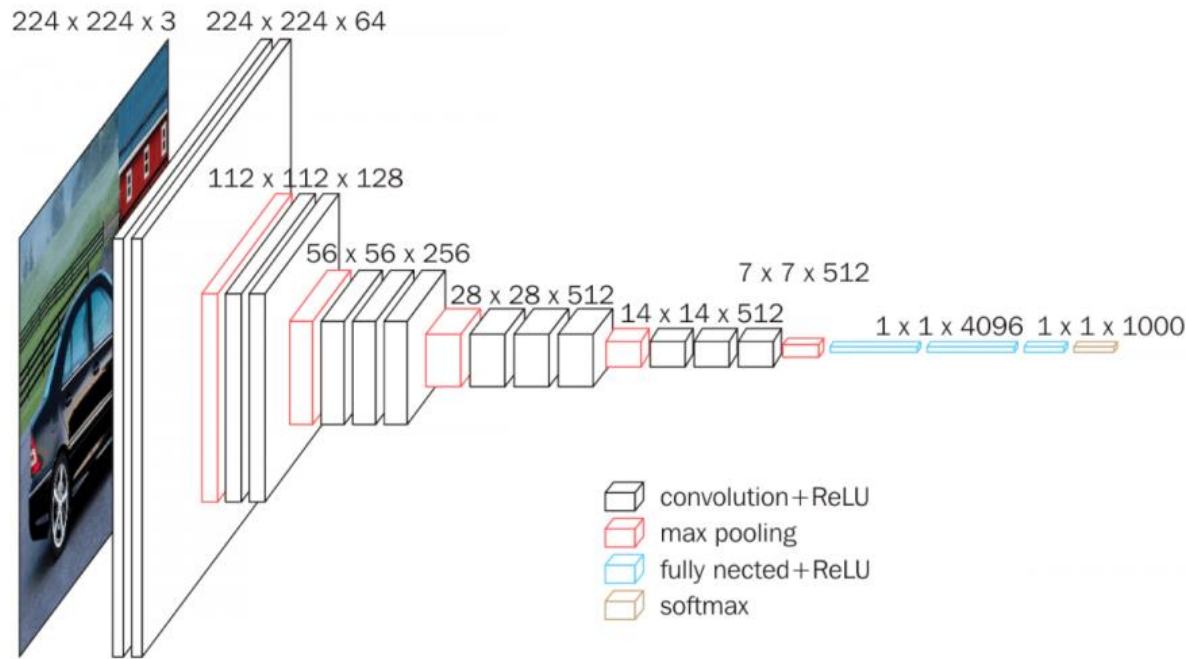


Traditional ML



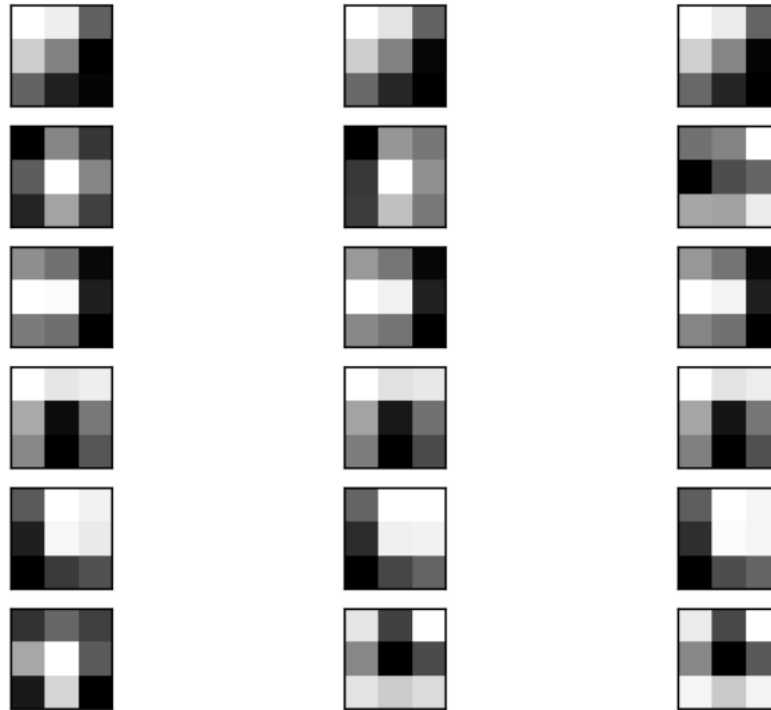
Transfer learning thanks to DL

Share your models (VGG16 for instance)



	Layer	Feature Map	Size	
	Input	Image	1	224 x 224 x 3
1	2 X Convolution	64	224 x 224 x 64	
	Max Pooling	64	112 x 112 x 64	
3	2 X Convolution	128	112 x 112 x 128	
	Max Pooling	128	56 x 56 x 128	
5	2 X Convolution	256	56 x 56 x 256	
	Max Pooling	256	28 x 28 x 256	
7	3 X Convolution	512	28 x 28 x 512	
	Max Pooling	512	14 x 14 x 512	
10	3 X Convolution	512	14 x 14 x 512	
	Max Pooling	512	7 x 7 x 512	
13	FC	-	25088	
14	FC	-	4096	
15	FC	-	4096	
Output	FC	-	1000	

Plot of the First 6 Filters From VGG16 With One Subplot per Channel



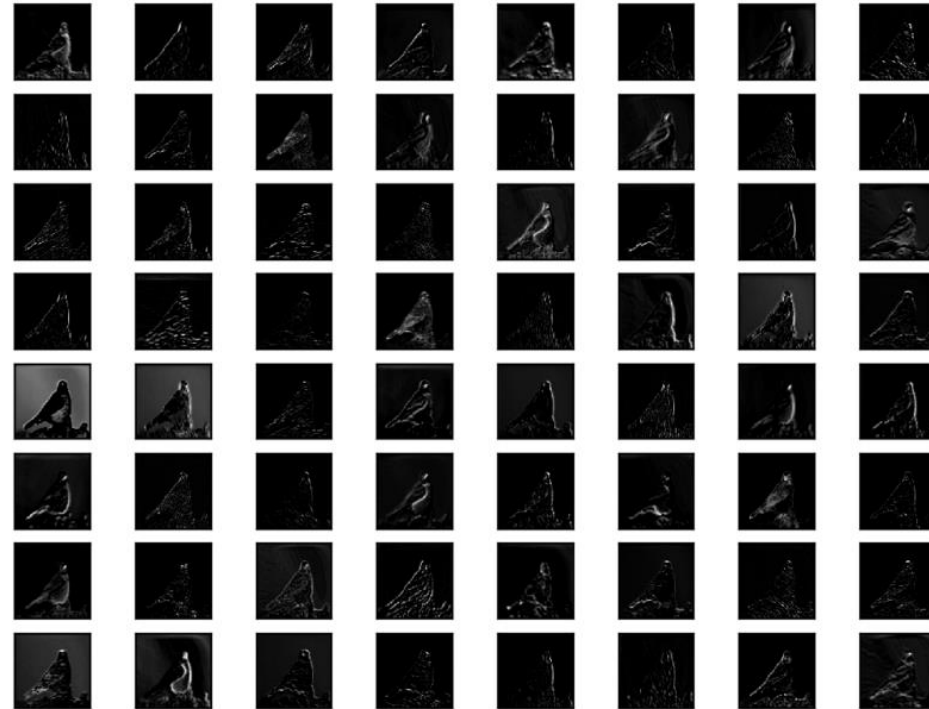
Robin, by Chris Heald



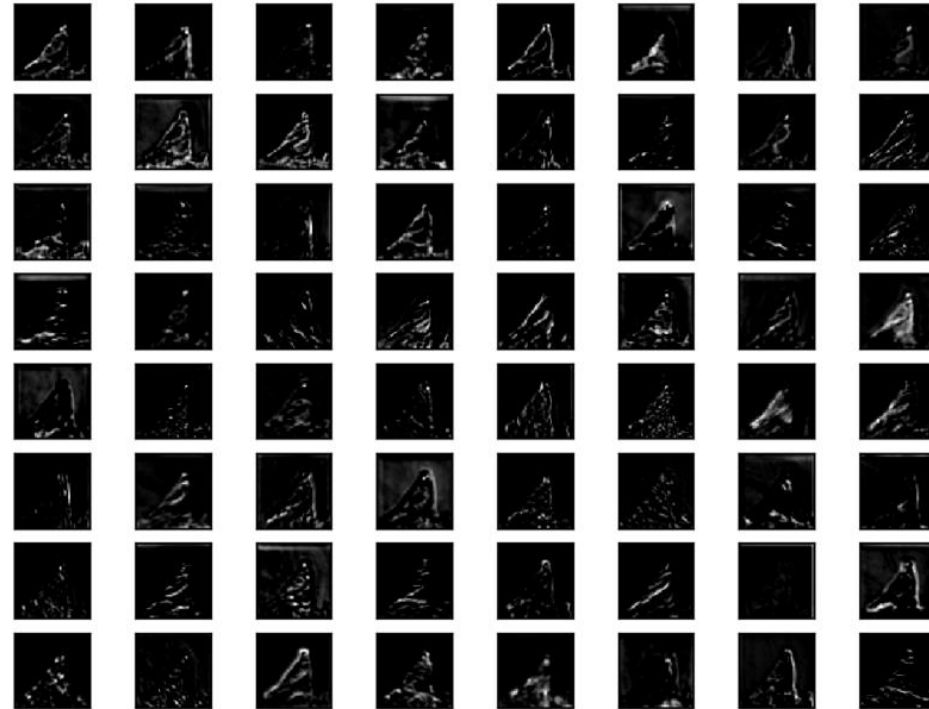
Visualization of the Feature Maps Extracted From Block 1 in the VGG16 Model



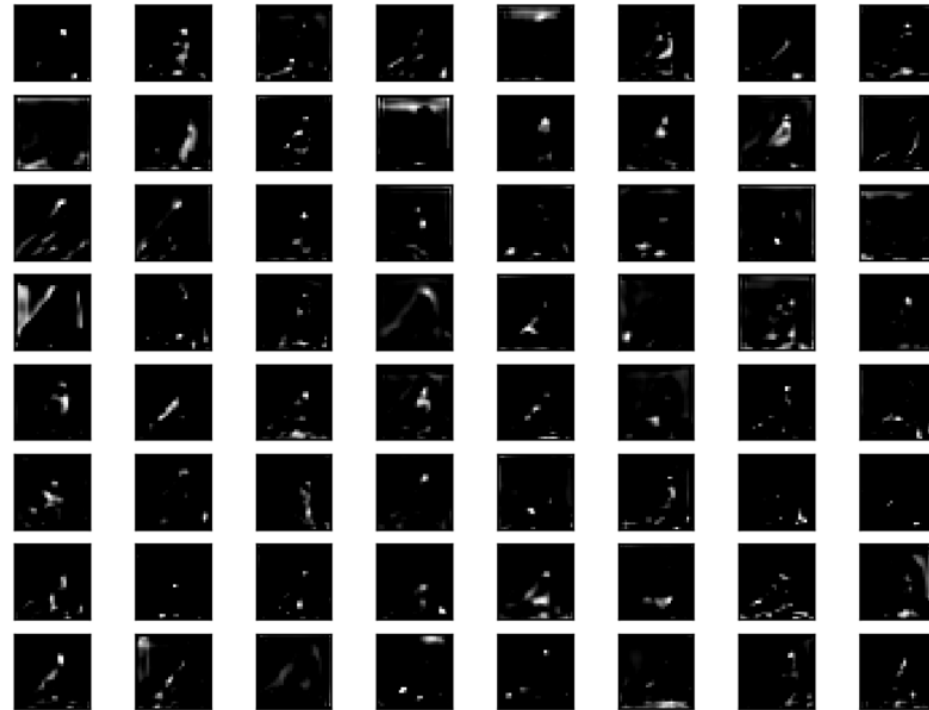
Visualization of the Feature Maps Extracted From Block 2 in the VGG16 Model



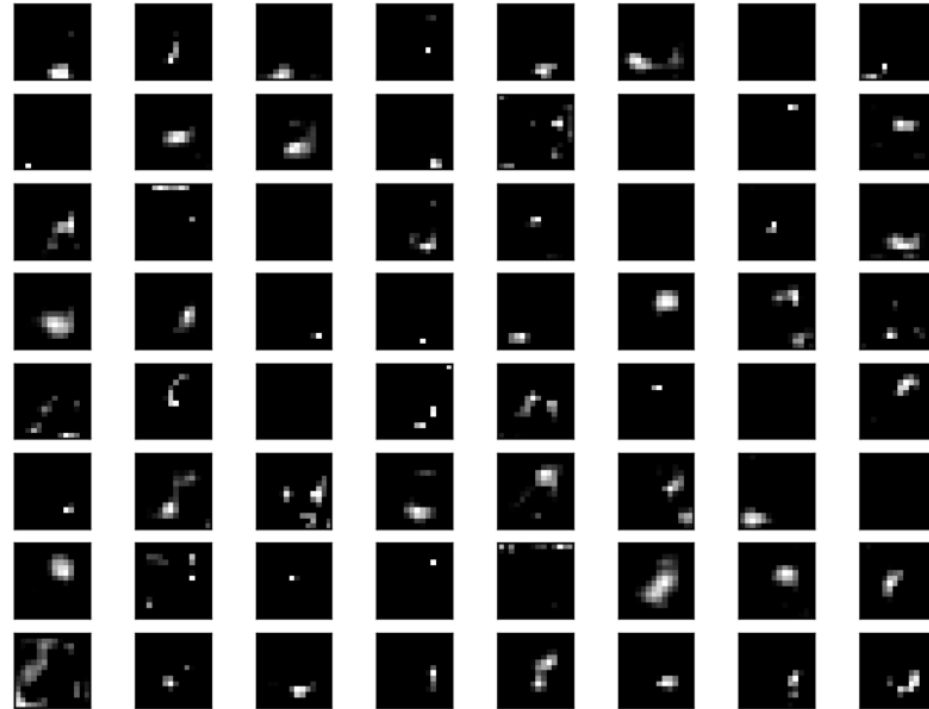
Visualization of the Feature Maps Extracted From Block 3 in the VGG16 Model



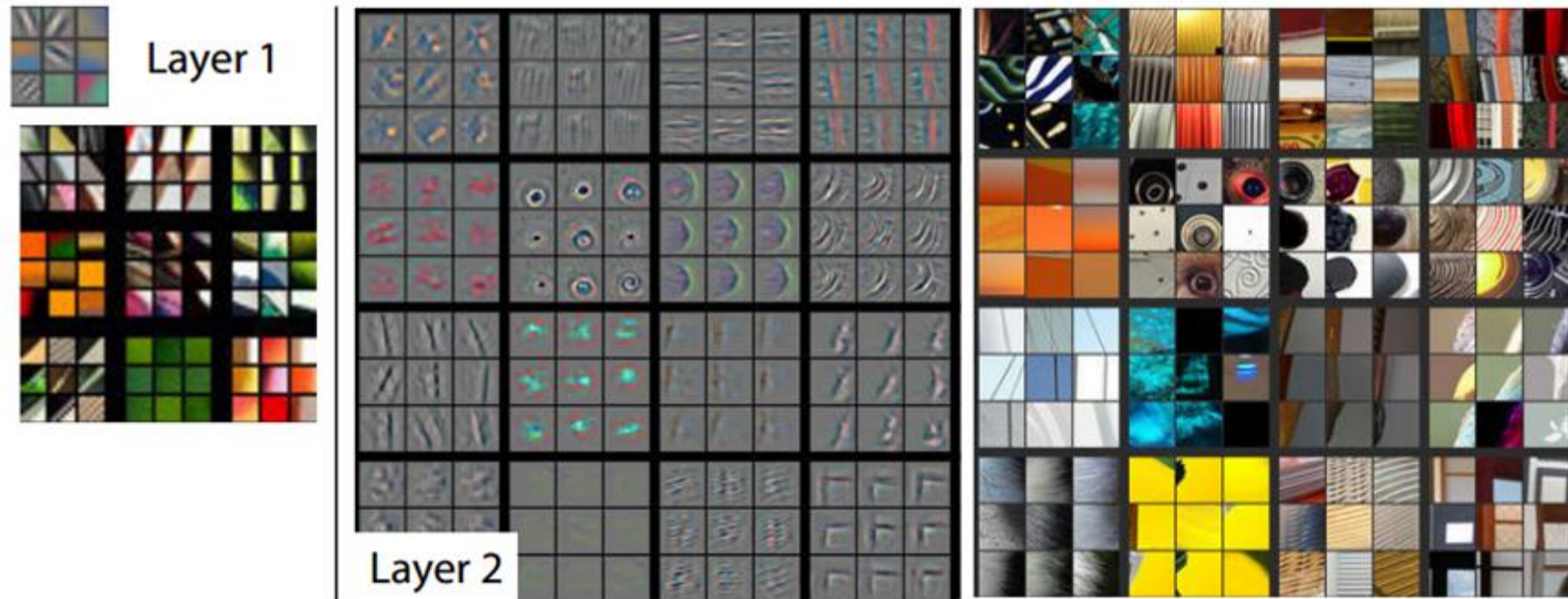
Visualization of the Feature Maps Extracted From Block 4 in the VGG16 Model



Visualization of the Feature Maps Extracted From Block 5 in the VGG16 Mode

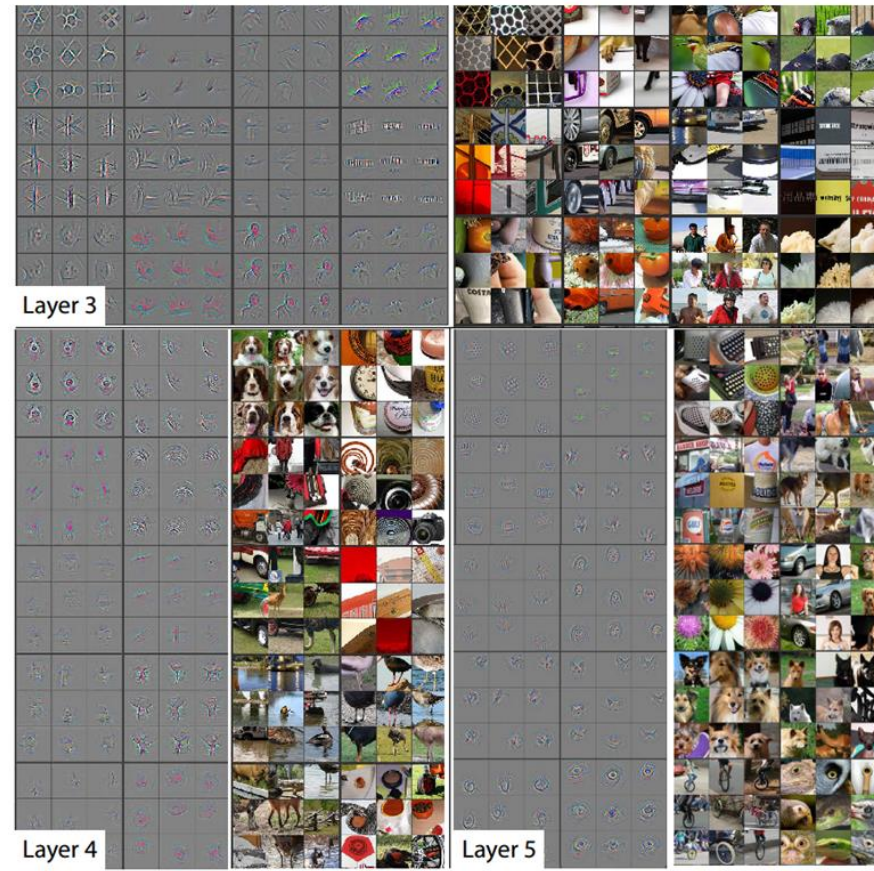


Share your models (VGG16 for instance)



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labeled Layer 2, we have representations of the 16 different filters (on the left)

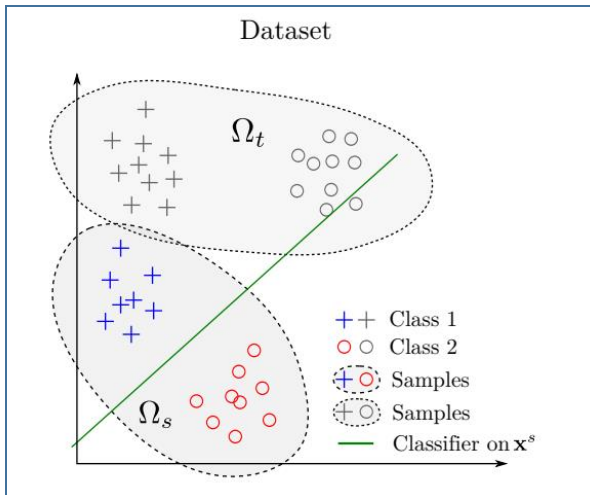
Share your models (VGG16 for instance)



Visualizations of Layers 3, 4, and 5

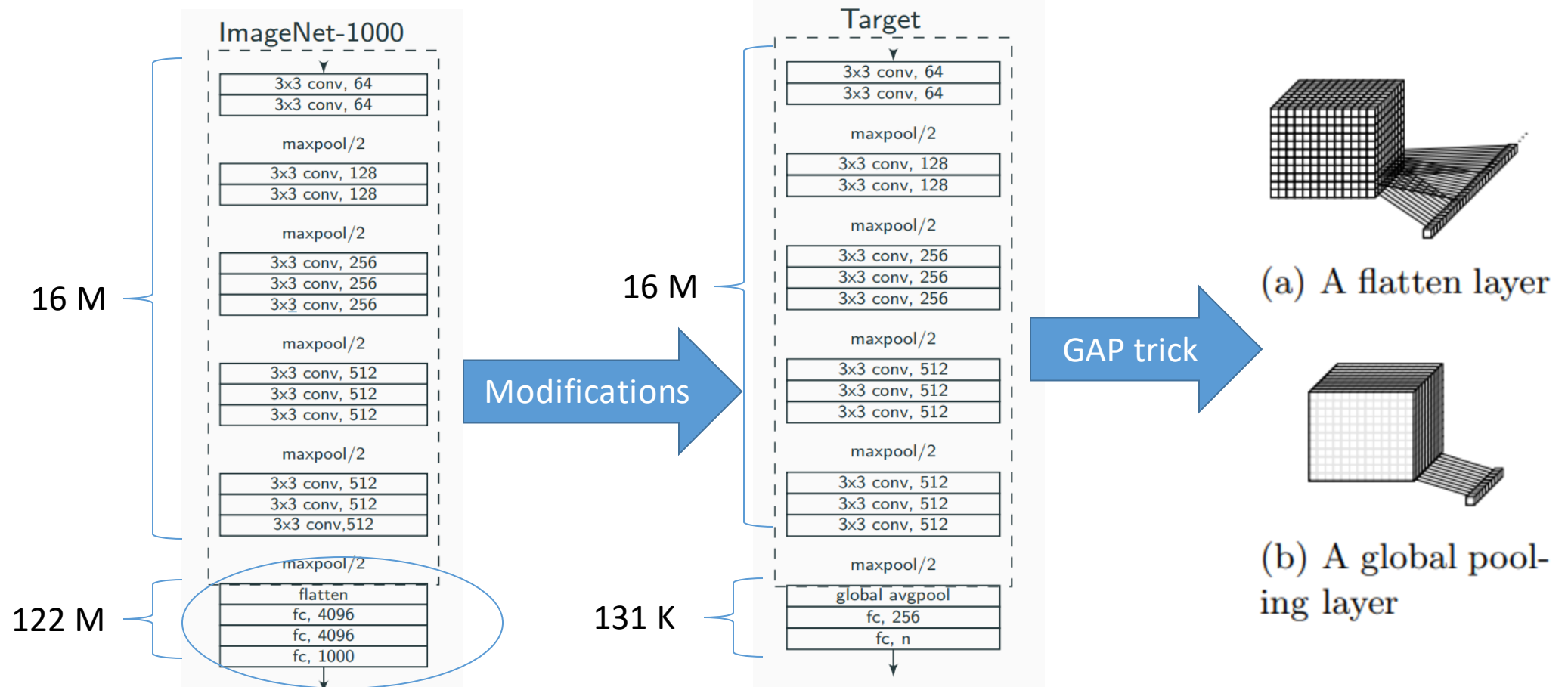
Transfer learning

- A new classification task?
 - Insect image classification
 - **Few data**: hard to label and hard to get images
- How to solve it?
 - Take a pre-trained model on ImageNet
 - Simplify the model
 - Fine tune the model on insects



Maxime Martineau, **Romain Raveaux**, Clément Chatelain, Donatello Conte, Gilles Venturini. Effective Training of Convolutional Neural Networks for Insect Image Recognition. *19th International Conference on Advanced Concepts for Intelligent Vision Systems*, Sep 2018

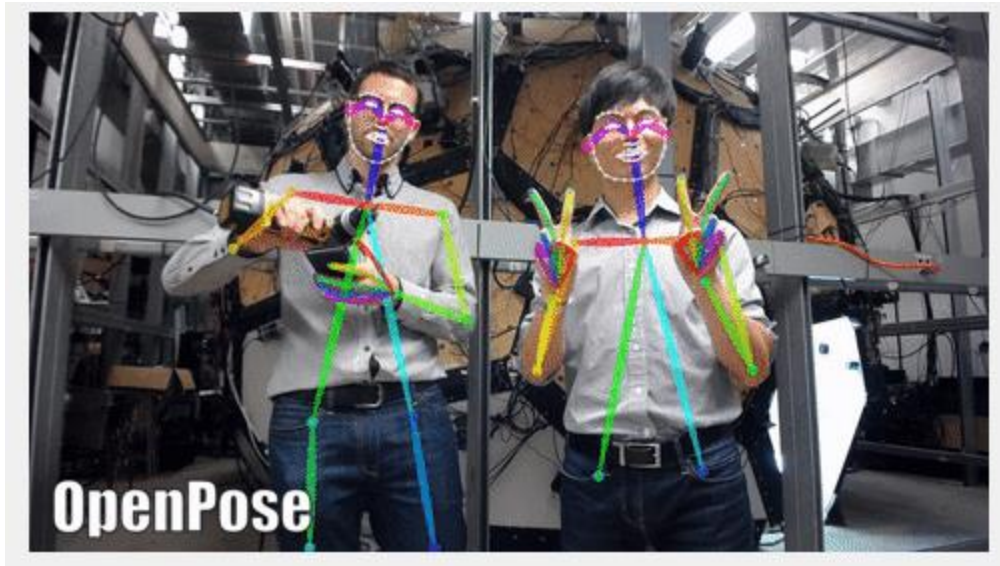
The VGG16 model: 138 millions parameters



Decision parameters are not « generic »
Parameters to be retrained

Share your models

- <https://modelzoo.co/>
- **Model Zoo** : Discover open source deep learning code and pretrained models.

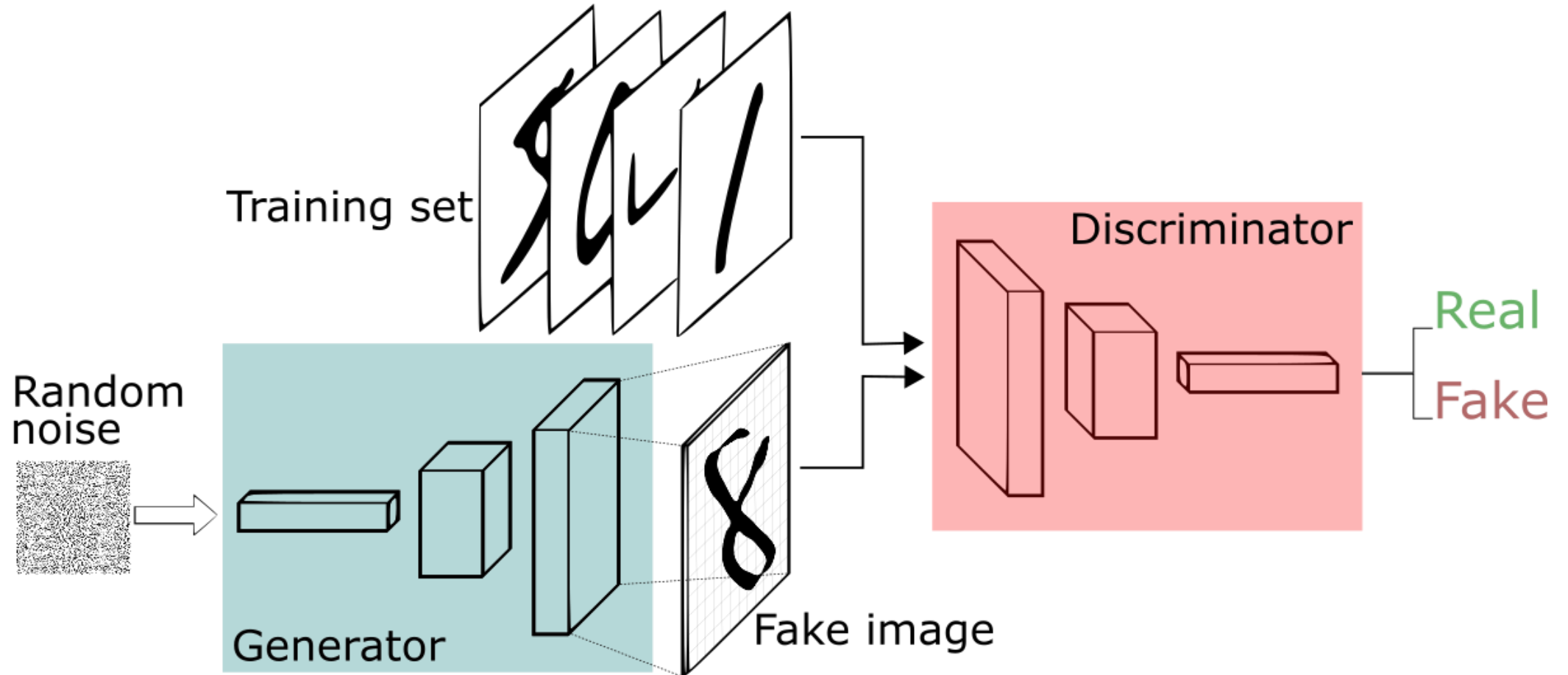


DeepOSM

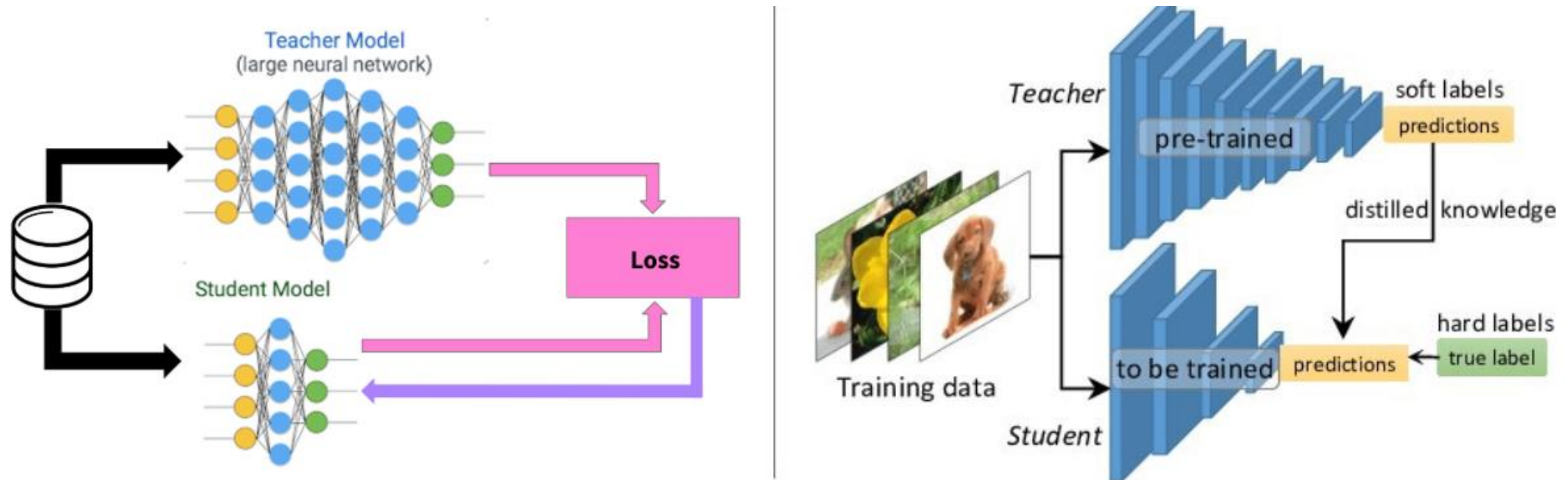
Competition of models

- Auto Encoders
- Generative Adversarial Networks
- Co-learning

Generative Adversarial Networks (GAN)

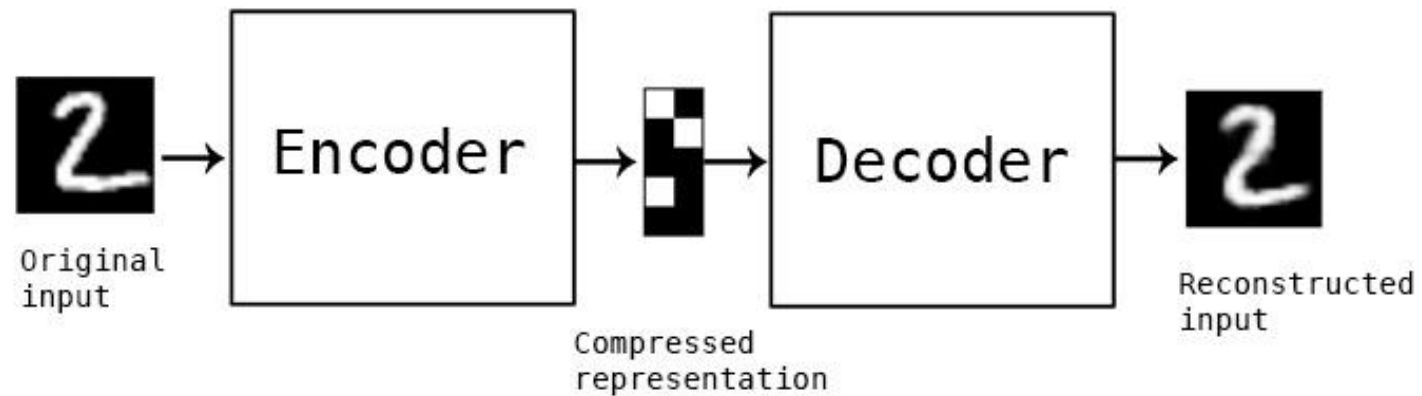
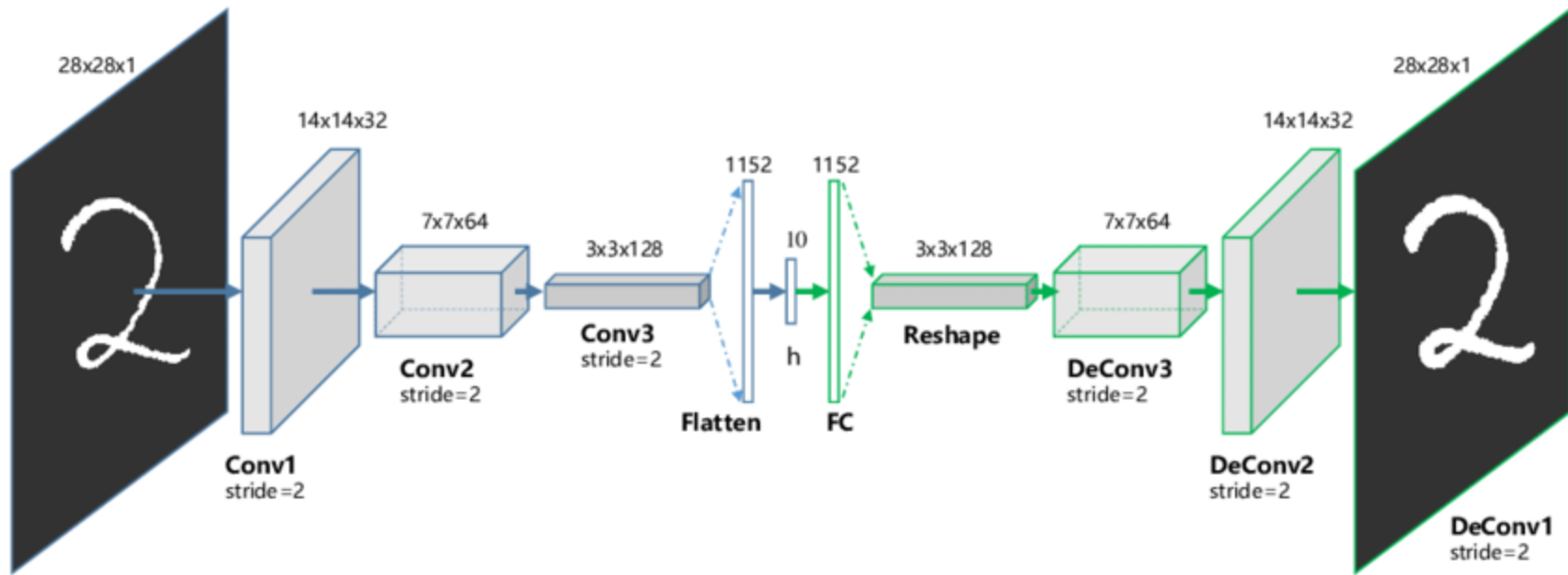


Teacher and student models : Knowledge Distillation



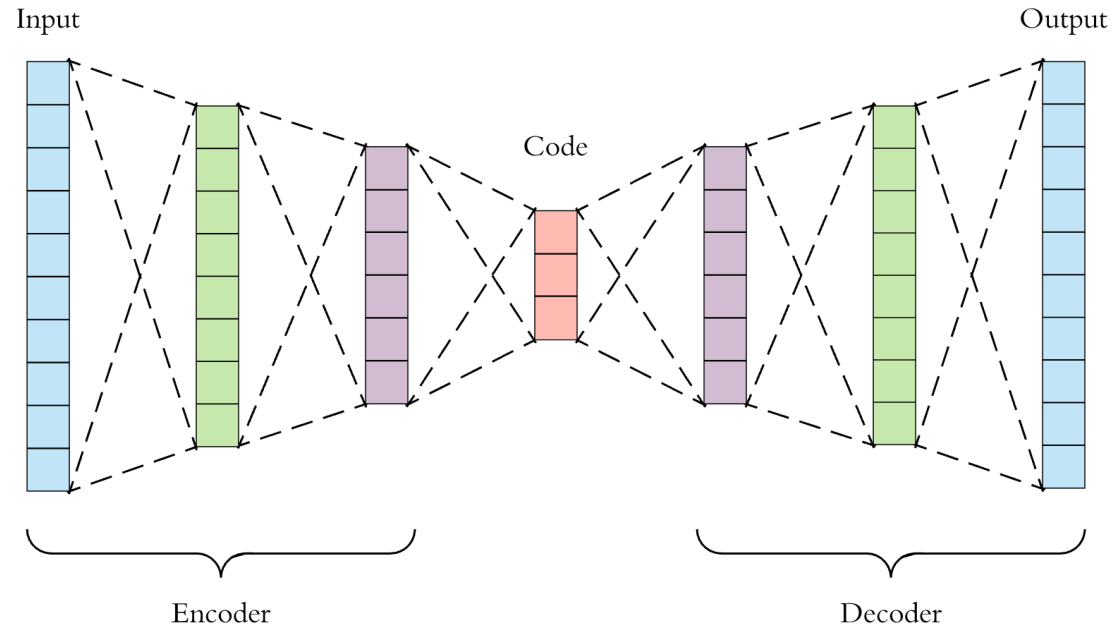
Knowledge distillation refers to the idea of model compression by teaching a smaller network, step by step, exactly what to do using a bigger already trained network.

Auto Encoder



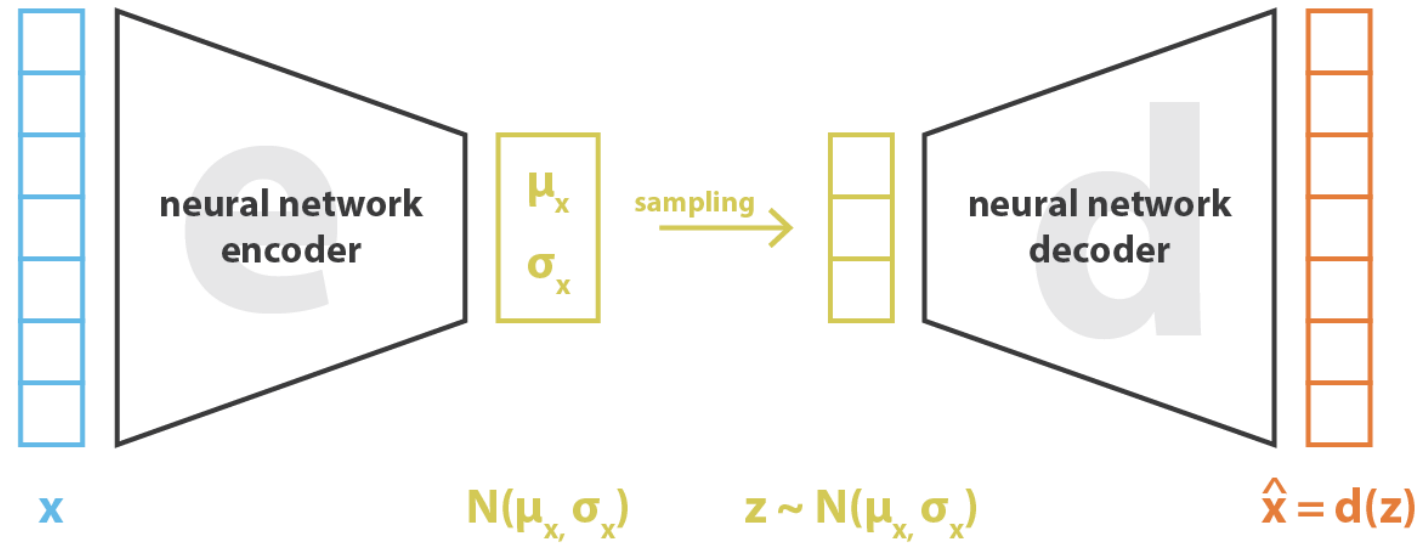
Auto Encoder : Self supervised learning

- The output (\hat{x}) can be compared to the input (x)
- The input data contains the annotation
- Self-supervised learning is autonomous supervised learning.
- It eliminates the pre-requisite requiring humans to label data



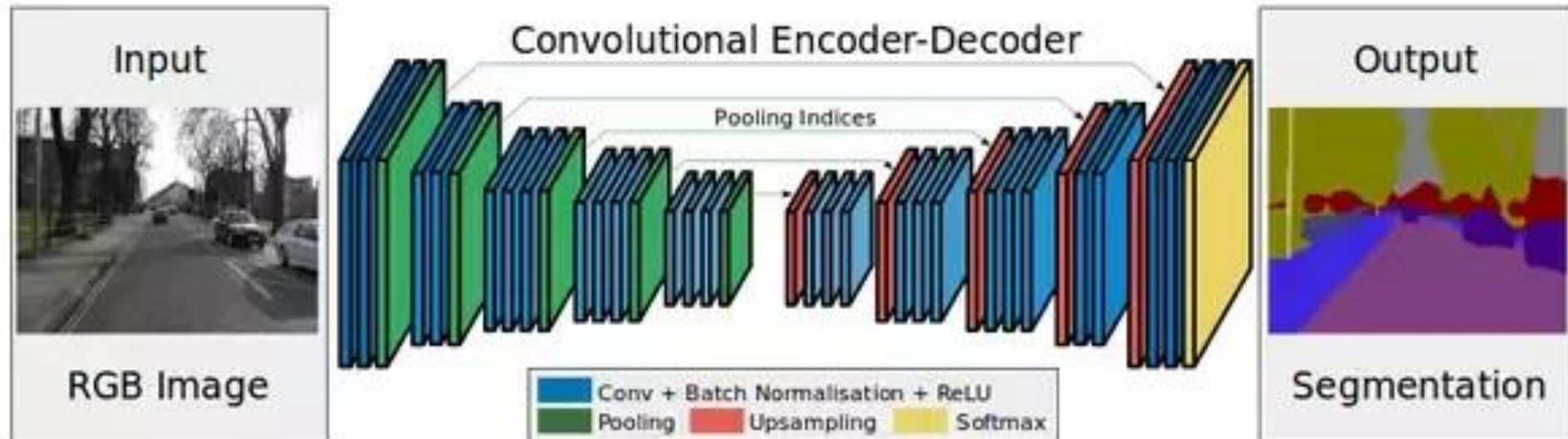
$$\text{loss} = \|x - \hat{x}\|^2$$

Auto Encoder : A competition ?



$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

Supervised Learning : Encoder – Decoder : Semantic segmentation



Blue : Pavement

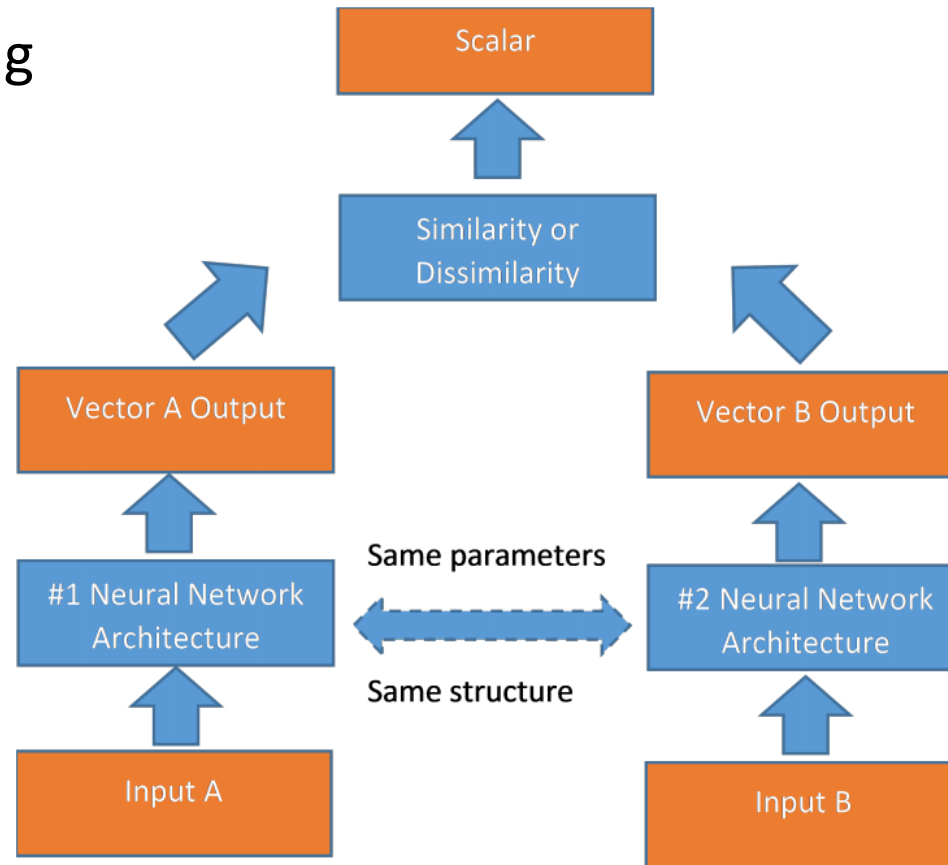
Green : Tree

Purple : Cars

....

Co-operation : Siamese architecture

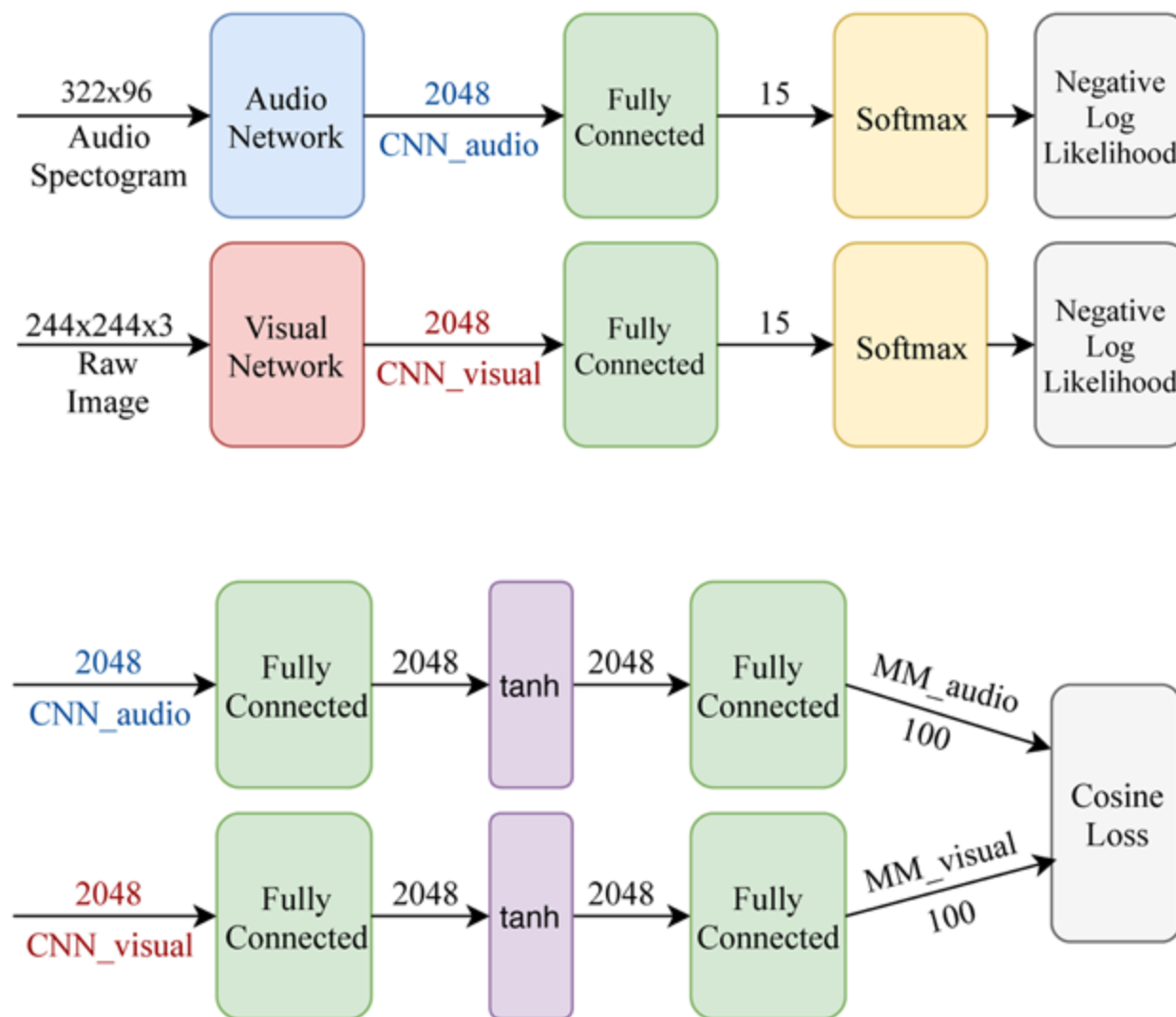
(Dis)Similarity Learning



Co-Learning

Co-Learning : A challenge is to associate knowledge between modalities with different representations.

- Joint reasoning



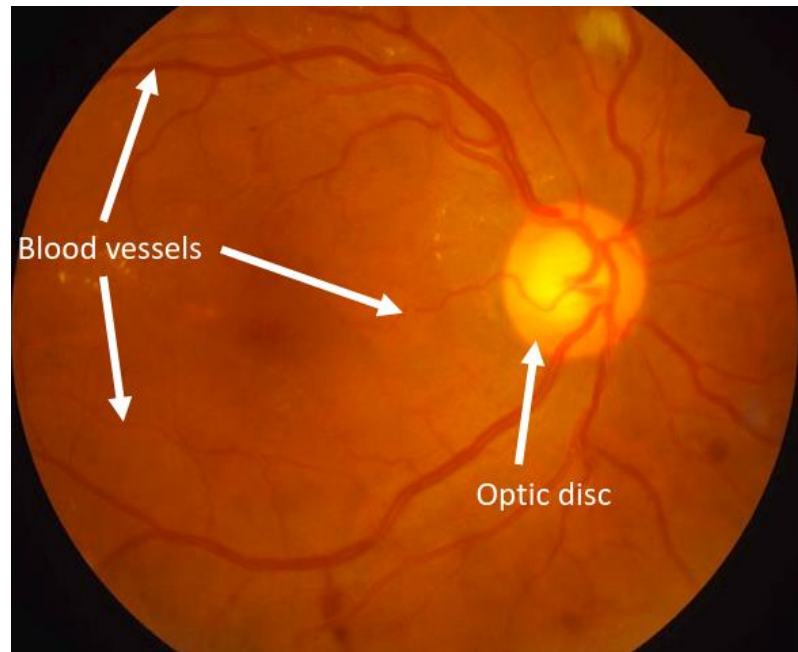
Application : Healthcare

Application : healthcare

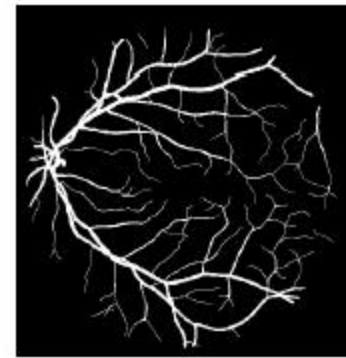
PhD of Taibou BIRGUI SEKOU

Title : Dictionary Learning and Convolutional Neural Networks
for Retinal Image Segmentation

Lab : LIFAT (Tours)



(A) DRIVE image



(B) DRIVE annotation



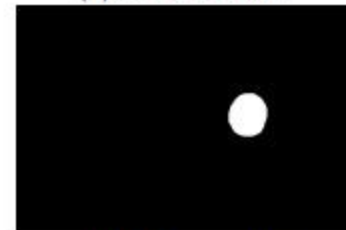
(C) STARE image



(D) STARE annotation



(E) IDRiD image



(F) IDRiD annotation



(G) CHASE_DB1 image



(H) CHASE_DB1 annotation

The present and future of deep learning in radiology

- [European Journal of Radiology](#)
- [Volume 114](#), May 2019, Pages 14-24
- <https://www.sciencedirect.com/science/article/pii/S0720048X19300919>
- PDF in CELENE

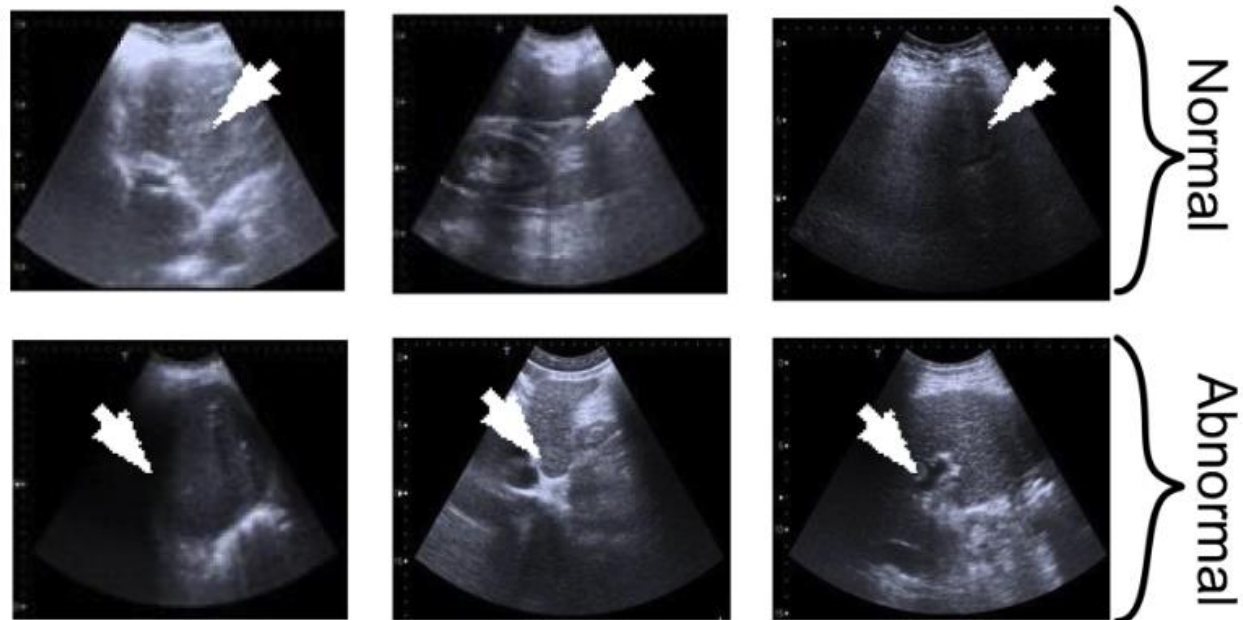


Fig. 5. Fatty Liver Disease US images (Normal Liver: Upper Row, Abnormal: Lower Row (Image courtesy: AtheroPoint™)) (Courtesy of AtheroPoint, CA, USA).

Conclusion

<https://threader.app/thread/1209497021398343680>(taken from)

- Don't say "Deep Learning can't do X" when what you really mean is "supervised learning needs too much data to do X" ..
- Don't say " Deep Learning is biased" when what you really mean "plain supervised learning reproduces the biases in the training data"
- In all of this, the questions are:
 - (1) what architecture? (it needs to be compatible with backprop)
 - (2) what learning paradigm and objective function? (Reinforcement Learning, Self-supervised Learning, Supervised Learning, ...)
 - (3) what training data and training protocol?

But trainable networks of differentiable modules are here to stay

Forthcoming challenges

- Explainability of learnt features :
 - Algorithms should not decide alone.
 - I trust my doctor. Doctors already use automatic systems.
 - Beyond reliability measure
- Spurious correlation : correlation vs causality
 - Simpson's paradox : https://en.wikipedia.org/wiki/Simpson's_paradox
 - Criminal never smile
 - <https://arxiv.org/pdf/1611.04135.pdf>

<https://sciencetonante.wordpress.com/2013/04/29/le-paradoxe-de-simpson/>



	Traitement A	Traitement B
Petits calculs (<2cm)	81/87 93%	234/270 87%
Gros calculs (>2cm)	192/263 73%	55/80 69%
Total	273/350 78%	289/350 83%



(a) Three samples in criminal ID photo set S_c .



(b) Three samples in non-criminal ID photo set S_n .

Long term challenges

- First challenge : The next revolution will not be supervised
 - How is it that many people learn to drive a car fairly safely in 20 hours of practice, while current imitation learning algorithms take hundreds of thousands of hours, and reinforcement learning algorithms take millions of hours? Clearly we're missing something big.
 - “Labeled samples are the opium of the AI researcher” Jitendra Malik
 - **A model of the world is missing**
 - How to learn it ? Animal-like self-supervised learning : But how ?
 - How to integrate it in the training protocol?
- The second challenge : is to get machines to learn to reason and plan,
 - not just to perceive and to act reactively.
 - The challenge here is to get learning machines to be able to perform long chains of reasoning.
 - Multihead-self-attention based methods and memory-augmented networks, transformer networks...

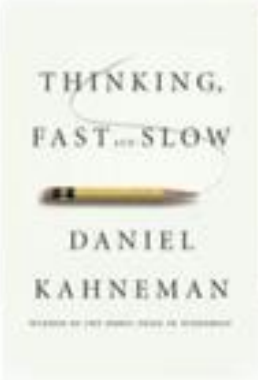
From AAAI-20* conference slide of Yoshua Bengio

SYSTEM 1 VS. SYSTEM 2 COGNITION

2 systems (and categories of cognitive tasks):

System 1

- Intuitive, fast, **UNCONSCIOUS**, non-linguistic, habitual
- Current DL





THINKING,
FAST... SLOW
DANIEL
KAHNEMAN

System 2

- Slow, logical, sequential, **CONSCIOUS**, linguistic, algorithmic, planning, reasoning
- Future DL

Manipulates high-level / semantic concepts, which can be recombined combinatorially





Mila

Deep Learning heroes : turing award

Y. Lecun : CNN +
BackProp

G. Hinton : Self
Supervised learning +
BackProp

Y. Bengio : GAN + RNN



Believe or not : Bell Lab 1995

1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand quantitatively why big neural nets working on large databases are not so bad. (Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong



V. Vapnik 3/14/95



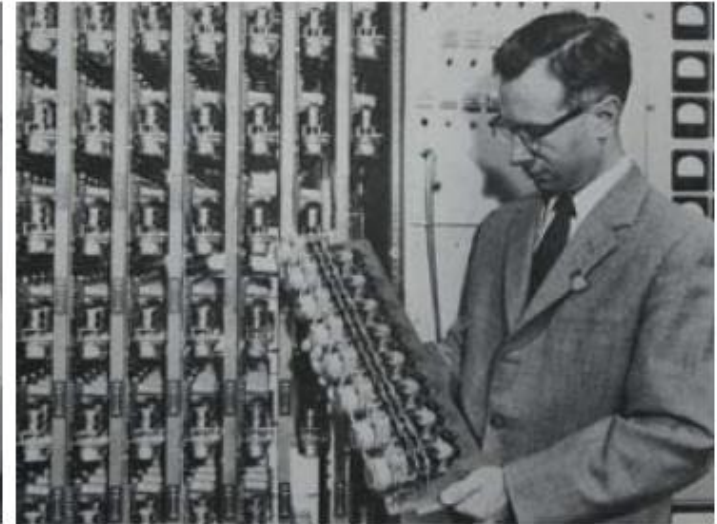
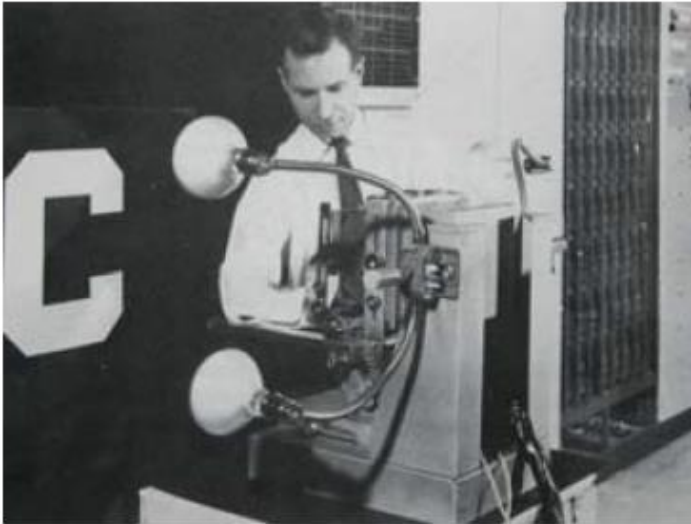
L. Jackel 3/14/95



Witnessed by Y. LeCun 3/14/95



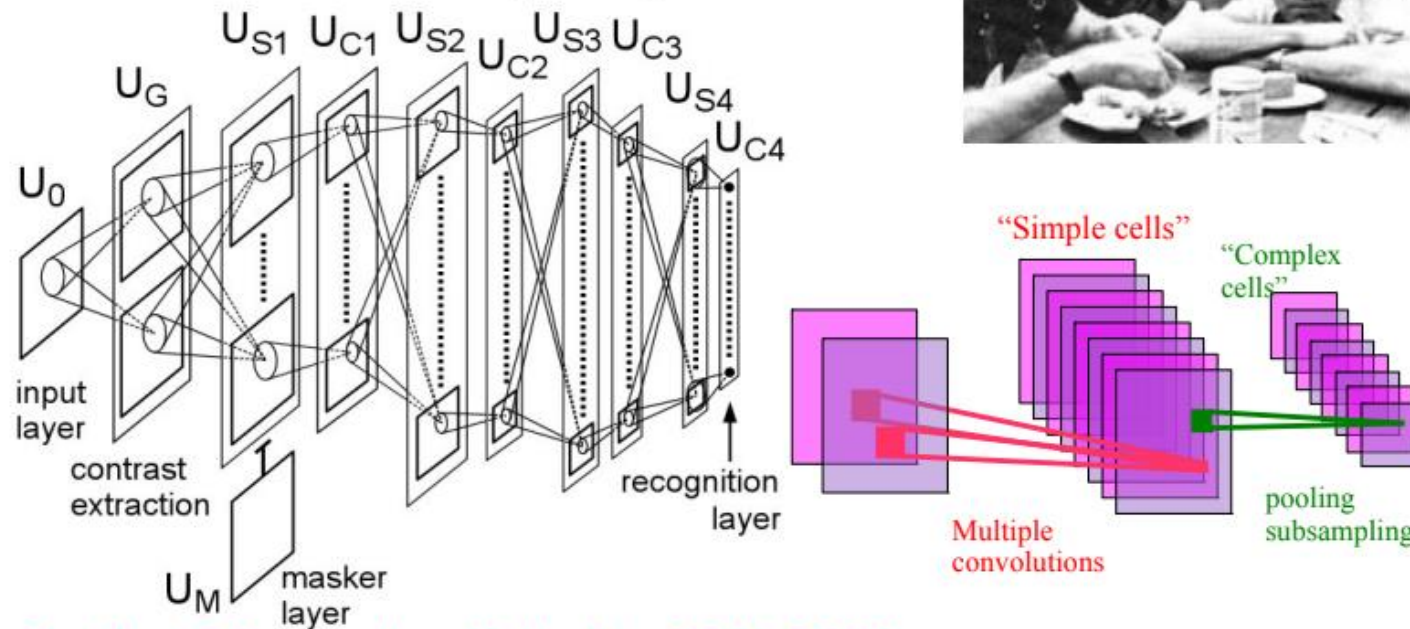
“Perceptrons and the Theory of Brain Mechanisms” published in 1962. Frank Rosenblatt



History : Epistemology

■ [Hubel & Wiesel 1962]:

- ▶ **simple cells** detect local features
- ▶ **complex cells** “pool” the outputs of simple cells within a retinotopic neighborhood.



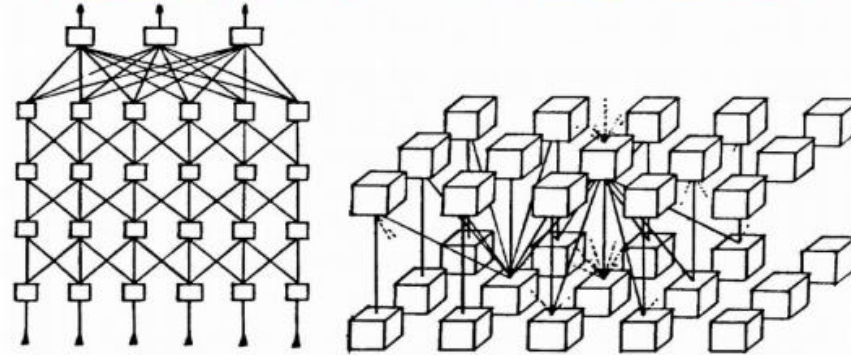
Cognitron & Neocognitron [Fukushima 1974-1982]

History : Epistemology

Early Networks [LeCun 85, 86] Y LeCun

Binary threshold units
trained supervised
with "target prop"

Hidden units compute a
virtual target

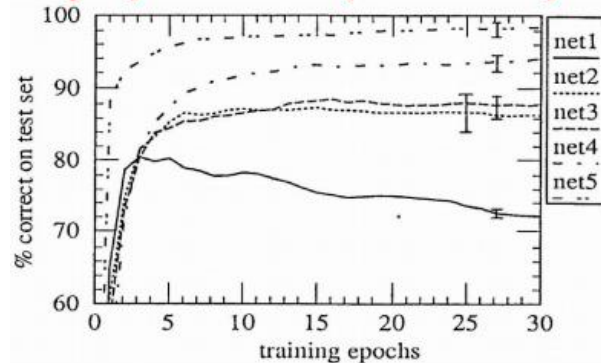
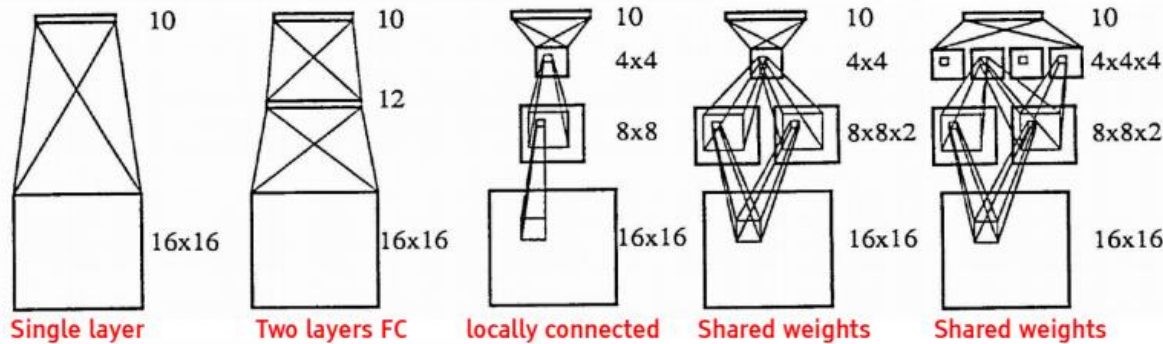


<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											
0	0	0	0	0	0	0	0																																																																																											

History : Epistemology



Trained with Backprop. 320 examples.



- Convolutions with stride (subsampling)
- No separate pooling layers

network architecture	links	weights	performance
single layer network	2570	2570	80 %
two layer network	3240	3240	87 %
locally connected	1226	1226	88.5 %
constrained network	2266	1132	94 %
constrained network 2	5194	1060	98.4 %

Thank you

- That's it or maybe not ...
- Supervised Machine Learning for structured input/output: Polytech, Tours
 - 1. Introduction to supervised Machine Learning: A probabilistic introduction [PDF](#)
 - 2. Connecting local models : The case of chains [PDF slides](#)
 - 3. Connecting local models : Beyond chains and trees. [PDF slides](#)
 - 4. Machine Learning and Graphs : Introduction and problems [PDF slides](#)
 - 5. Graph Neural Networks. [PDF slides](#)
 - 6. Graph Kernels. [PDF slides](#)
 - 7. Appendix : An introduction to deep learning. [PDF slides](#)

- Supervised Machine Learning for structured input/output: Polytech, Tours
 - 1. Introduction to supervised Machine Learning: A probabilistic introduction [PDF](#)
 - 2. Connecting local models : The case of chains [PDF slides](#)
 - 3. Connecting local models : Beyond chains and trees. [PDF slides](#)
 - 4. Machine Learning and Graphs : Introduction and problems [PDF slides](#)
 - 5. Graph Neural Networks. [PDF slides](#)
 - 6. Graph Kernels. [PDF slides](#)
 - 7. Appendix : An introduction to deep learning. [PDF slides](#)