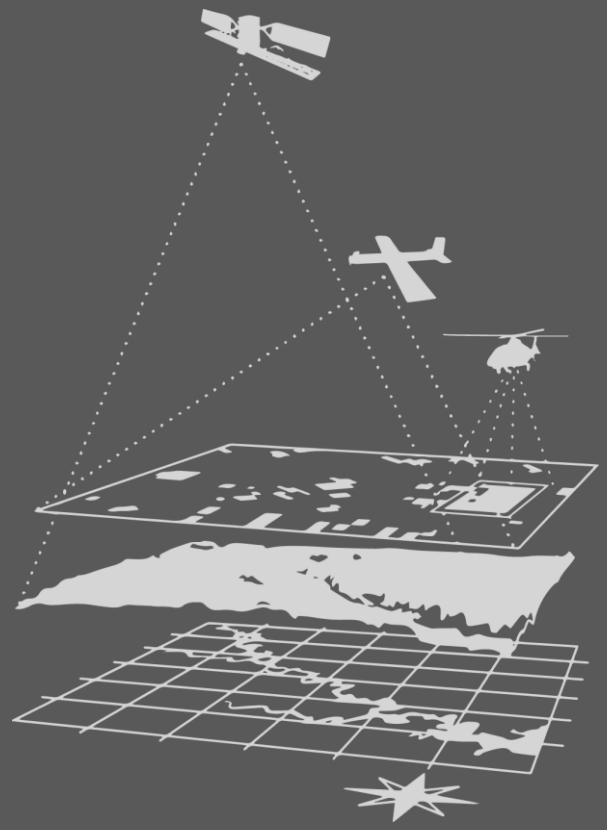
magellium

PUTTING KNOWLEDGE ON THE MAP

Magellium



Domaines d'activité

Observation de la terre

Géo-information

Imagerie & applications

2003
création

150+
employés

2 sites
Paris
Toulouse*

14,2 M€
CA 2016

Intégrée au groupe ARTAL depuis le 1er Septembre 2016

Offres & marchés

Etudes scientifiques
& techniques

Développement logiciel
& intégration de systèmes

Consulting
& assistance technique

30 %
Espace

30 %
Défense
& Sécurité

30 %
Energie &
Transport

10 %
Autres

cnes · DGA · cea · TOTAL · RATP · Rte · RENAULT · AIRBUS · SNCF

magellium

Offres & marchés

Etudes scientifiques
& techniques

Développement logiciel
& intégration de systèmes

Consulting
& assistance technique

Deep Learning

Groupe de travail et R&D interne

4 experts

Plus d'un an d'expérience

Partenariats académiques

INSTITUT DE MATHÉMATIQUES de TOULOUSE

isae SUPAERO

TELECOM ParisTech

irstea

magellium

magellium

PUTTING KNOWLEDGE ON THE MAP

Le Deep Learning appliqué au traitement des images satellite, un retour d'expérience

20/09/2017

Le Deep Learning

Qu’est-ce que c’est ?

- Principes de base
- Concepts théoriques

Comment on fait ?

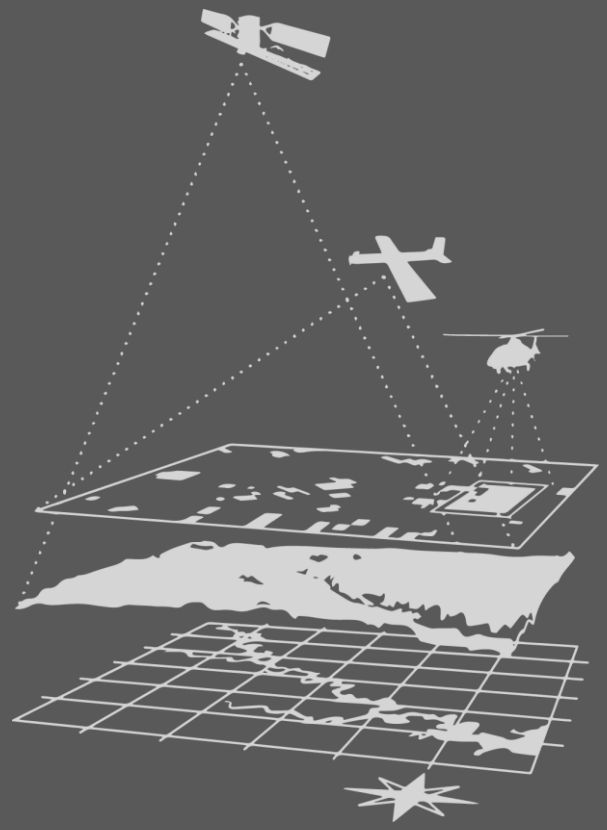
- Cas d’usage
- Points durs

magellium

PUTTING KNOWLEDGE ON THE MAP

Le Deep Learning

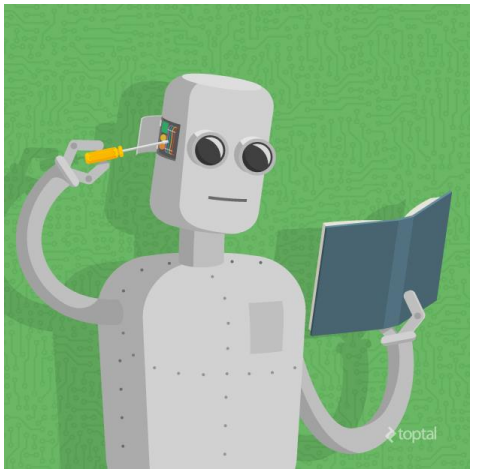
Qu'est-ce que c'est ?

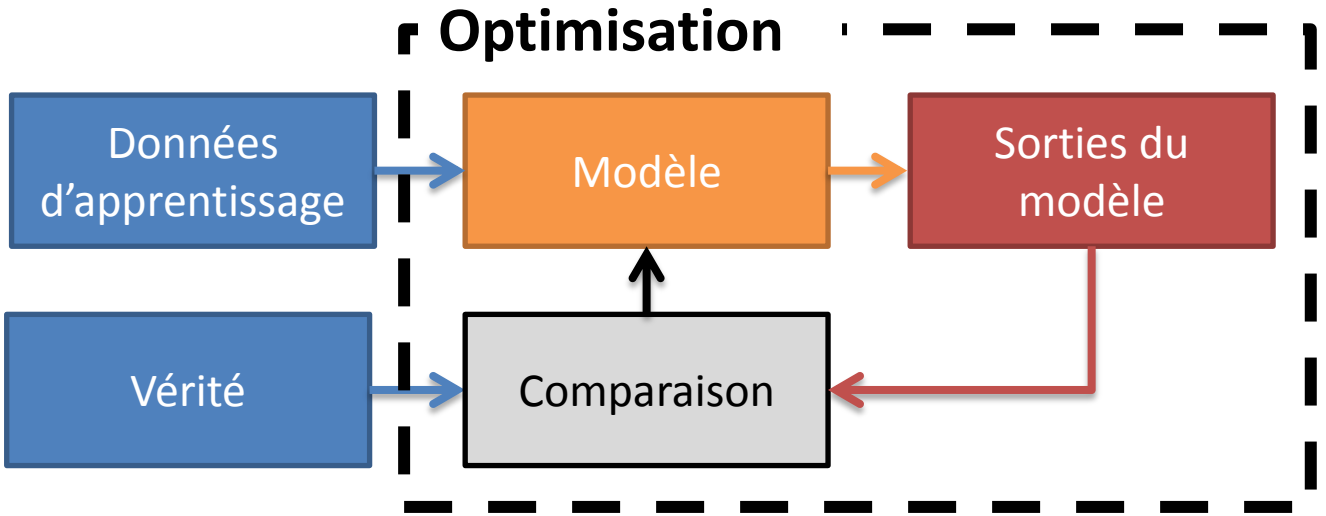


Machine learning

«Apprendre à une machine à reproduire un résultat, un comportement »

- Jouer aux échecs
- Parler
- Lire
- Traduire
- Reconnaitre une personne, un objet
- Se déplacer





Machine learning – une brève histoire

1950 Alan Turing invente le « test de Turing » qui permet de déterminer si une machine est « intelligente »

1952 Arthur Samuel crée le premier programme de Machine Learning pour jouer aux échecs et progresser au fur et à mesure des parties jouées

1957 Invention du perceptron par Frank Rosenblatt (Cornell University). Premier réseau de neurones. Inspiré des théories cognitives.

1997 Deep Blue (IBM) bat Gary Kasparov, alors champion du monde, aux échecs

2006 Apparition du terme « Deep Learning » pour désigner les nouveaux programmes de reconnaissance d'objets dans les images et les vidéos

2011 Amazon et Microsoft lancent leurs outils de Machine Learning

2014 Développement de Google Brain et de réseaux de neurones pour la reconnaissance d'objets

2015 Facebook développe DeepFace, un programme capable de reconnaitre les visages et les personnes dans les photos avec les mêmes performance qu'un être humain

2016 Google crée AlphaGo, un programme capable de battre 5 fois de suite le champion du monde de Go, jeu considéré le plus complexe au monde.

2017 Premières puces de calcul dédiées au calcul neural (Apple A11 bionic neural engine, NPU, …)

Introduction aux réseaux de neurones

Exemple : classification d'image

Features

X

Classification

Cat

Dog

Bird

Umbrella

Car

Plane

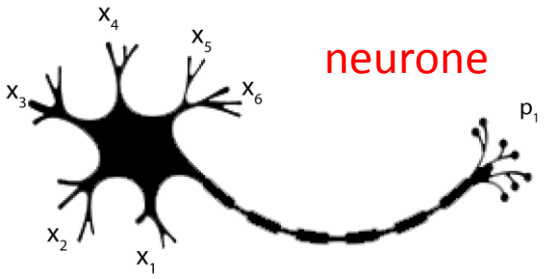
…

Modèle linéaire

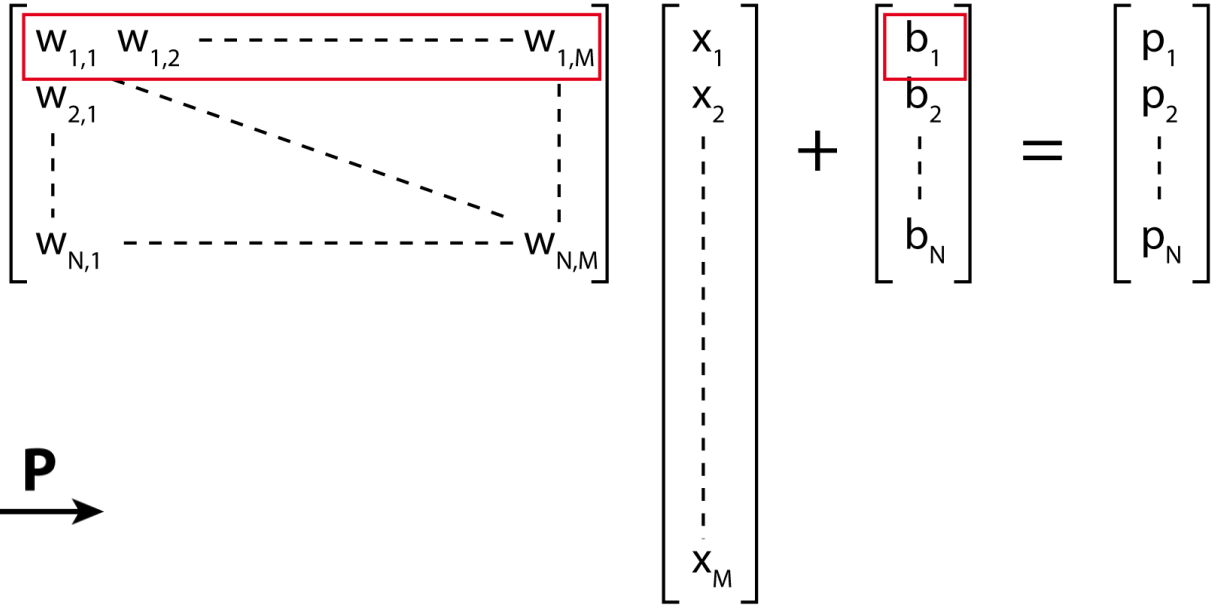
Construction d'un modèle simple – Perceptron / SVM à noyau linéaire

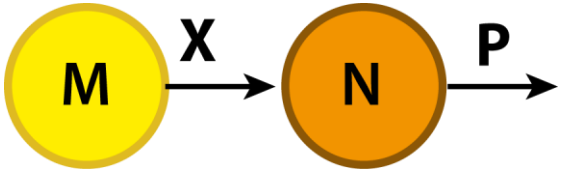
$$W_1 \boldsymbol{X} + B_1 = \boldsymbol{P} \qquad classe = argmax(\boldsymbol{P})$$

Vecteur en entrée	$\boldsymbol{X} \in \mathbb{R}^M$	Poids	$W_1 \in \mathbb{R}^{M \times N}$
Probabilités en sortie	$\boldsymbol{P} \in \mathbb{R}^N$	Biais	$B_1 \in \mathbb{R}^N$



neurone





Input Output

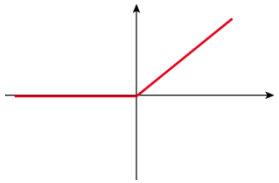
Réseau de neurones

Passage à deux couches

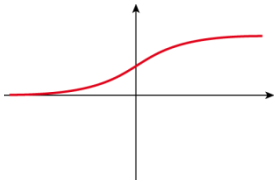
$$relu(\boldsymbol{W_1 X} + B_1) = \boldsymbol{Y}$$

$$W_2 \boldsymbol{Y} + B_2 = \boldsymbol{P}$$

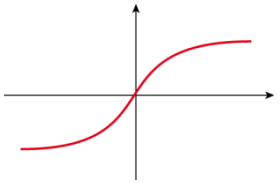
ReLU – Rectified Linear Unit



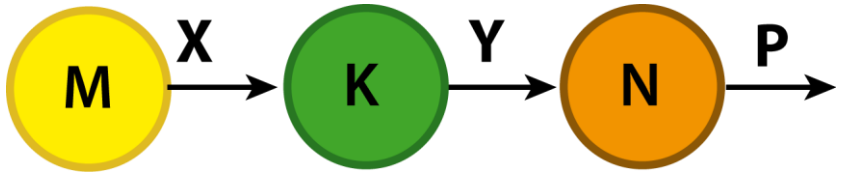
Sigmoïde



Tangente hyperbolique



Vecteur en entrée	$\boldsymbol{X} \in \mathbb{R}^M$	Couche 1	$W_1 \in \mathbb{R}^{M \times K}$	$B_1 \in \mathbb{R}^K$
Vecteur caché	$\boldsymbol{Y} \in \mathbb{R}^K$	Couche 2	$W_2 \in \mathbb{R}^{K \times N}$	$B_2 \in \mathbb{R}^N$
Probabilités en sortie	$\boldsymbol{P} \in \mathbb{R}^N$			



Input Hidden layer Output

Réseau de neurones « Fully Connected »

Apprentissage - Optimisation

Optimisation – trouver les poids et les biais qui permettent de résoudre le problème posé

$$W_2[\, relu(W_1\boldsymbol{X} + B_1)\,] + B_2 = \boldsymbol{P}$$

Couche 1	$W_1 \in \mathbb{R}^{M\times K}$	$B_1 \in \mathbb{R}^K$	$(M+1)\times K$ variables
Couche 2	$W_2 \in \mathbb{R}^{K\times N}$	$B_2 \in \mathbb{R}^N$	$(K+1)\times N$ variables

Exemple : N = 4, K=64, M=32 — 2372 variables à optimiser

Jeu de données d'apprentissage $(\boldsymbol{X_{true}}, \boldsymbol{P_{true}})$

Fonction coût : *Softmax* $\sigma(P)_i = \frac{e^{P_i}}{\sum_{k=1}^{N} e^{P_k}}$

Cross entropy $H(\sigma(\boldsymbol{P}), \boldsymbol{P_{true}}) = -\sum_i \boldsymbol{P_{true,i}} \log(\boldsymbol{\sigma(P)_i})$

Exemple numérique

$$P = \begin{bmatrix} -0.3 \\ 0.2 \\ 2.4 \end{bmatrix} \quad \sigma(P) = \begin{bmatrix} 0.06 \\ 0.09 \\ 0.85 \end{bmatrix}$$

$P_{true} = [0 \quad 0 \quad 1]$ $H(\sigma(P), P_{true}) = 0.16$

$P_{true} = [1 \quad 0 \quad 0]$ $H(\sigma(P), P_{true}) = 2.86$

Réseau de neurones

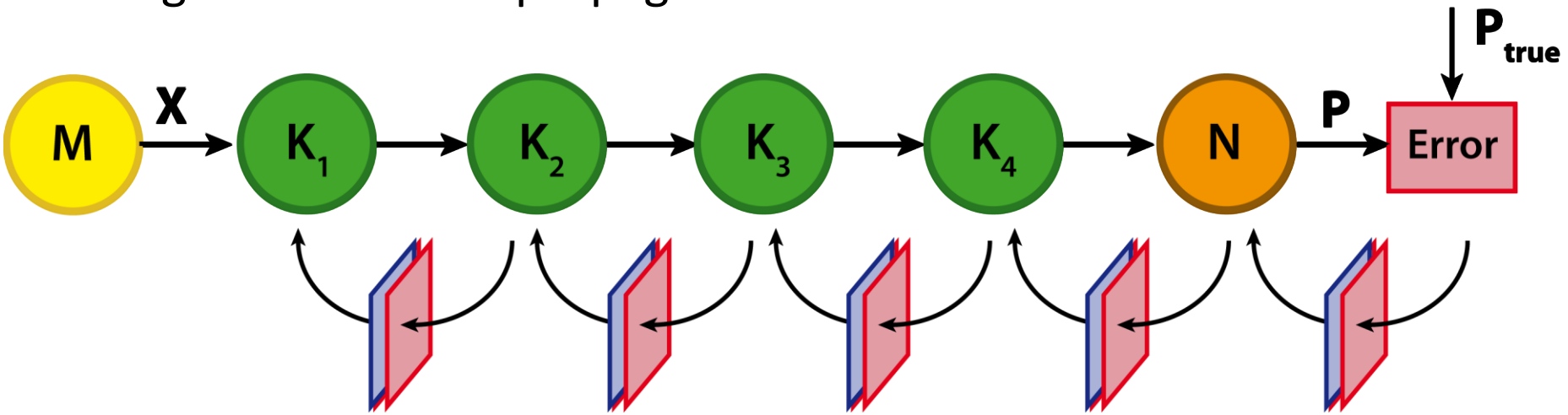
Problèmes

- Convergence de la méthode d'optimisation
- Trouver une « bonne » solution
- Temps de calcul
- Calcul efficace des gradients

Solutions

Stochastic Gradient Descent (SGD)
Gros jeux de données, représentatifs, …
GPU + Calculs par batch
Back-propagation / Chain-rule

Calcul de gradient – Back-propagation



- Calcul d'erreur
- Stockage des valeurs intermédiaires : **cartes d'activation**
- Rétro-propagation de l'erreur et calcul des gradients
- Ajustement des poids

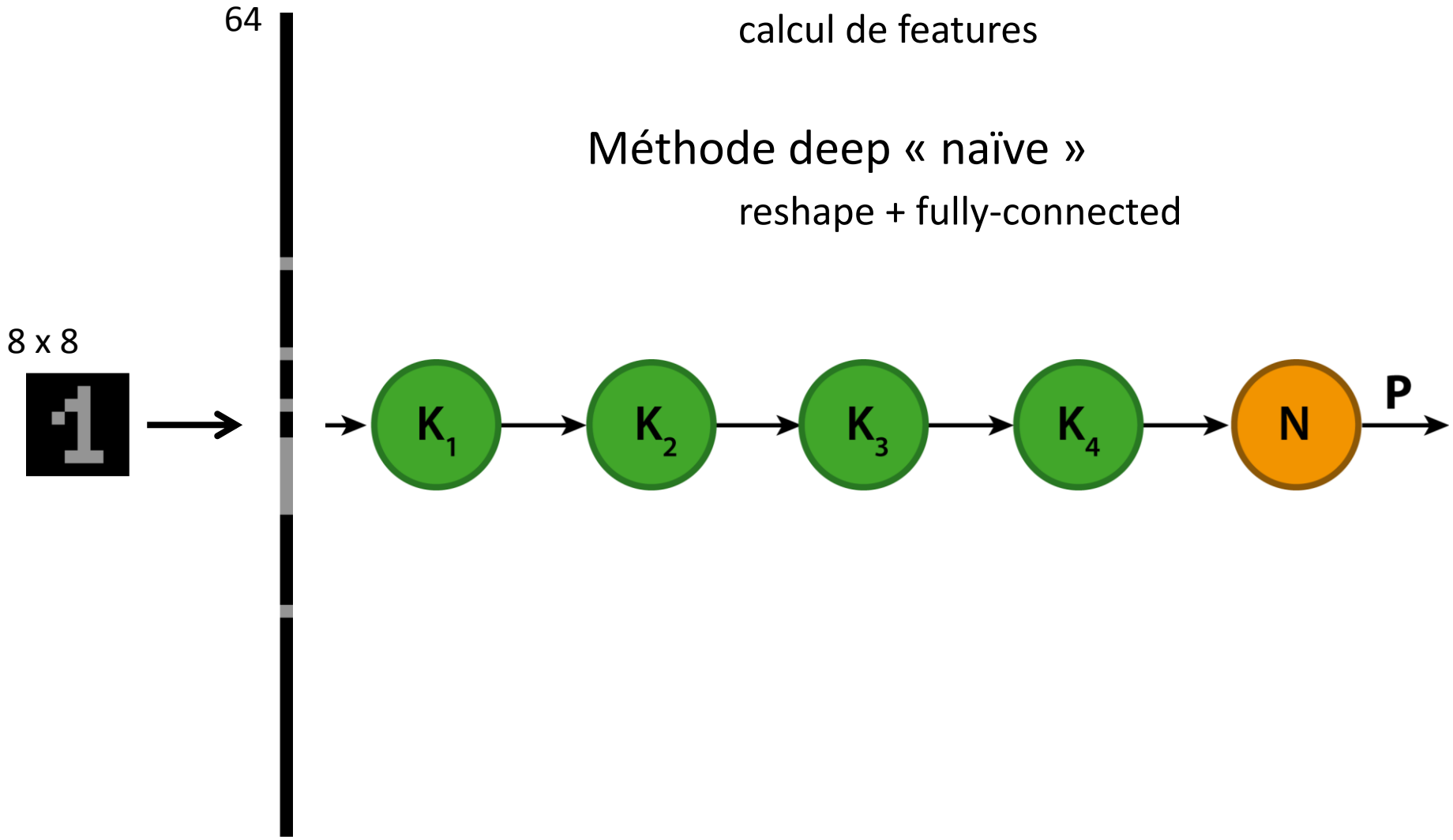
Traitement des images

Méthode « à l'ancienne »

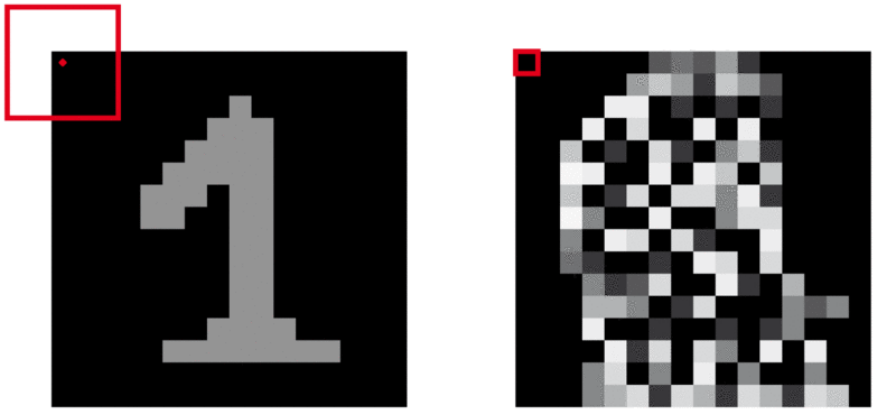
calcul de features

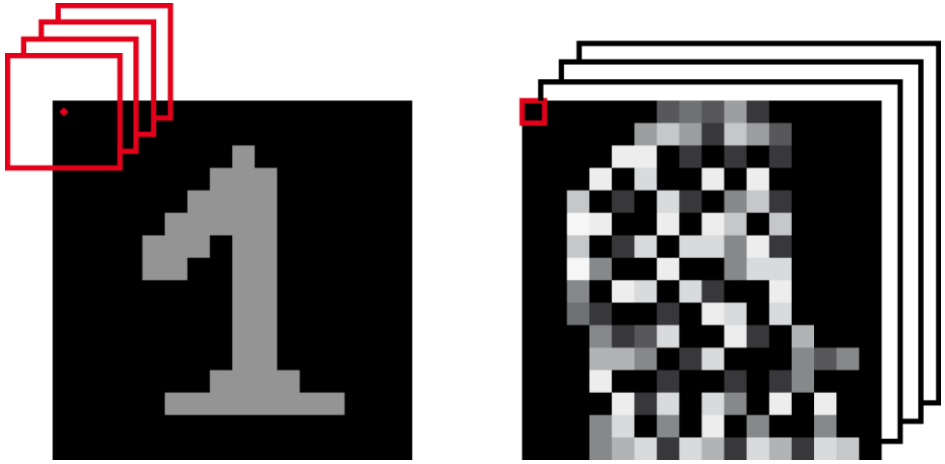
Méthode deep « naïve »

reshape + fully-connected



Couches de convolution

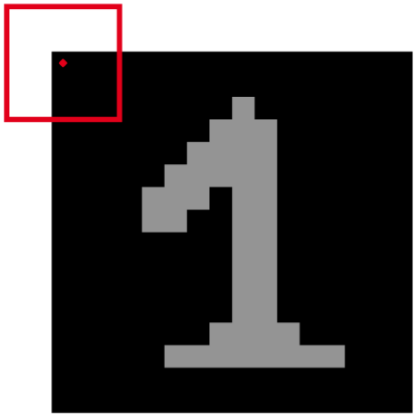


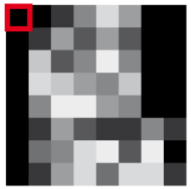


- Utilisation d'un noyau de convolution
- Parcours de l'image
- Extraction d'information avec invariance par translation
- Noyaux de convolution multi-couches

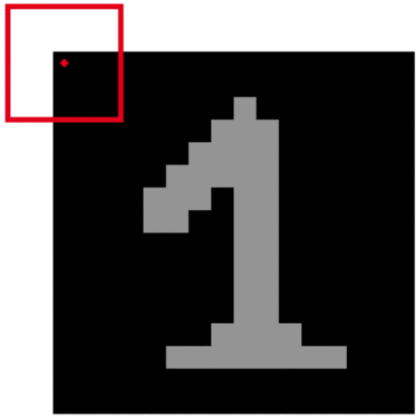
Réseau de neurones convolutifs

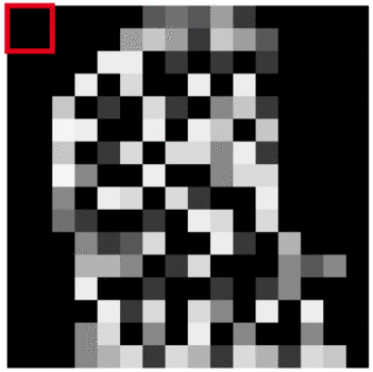
Réduction des dimensions

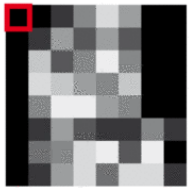




Pas ou « Stride » > 1





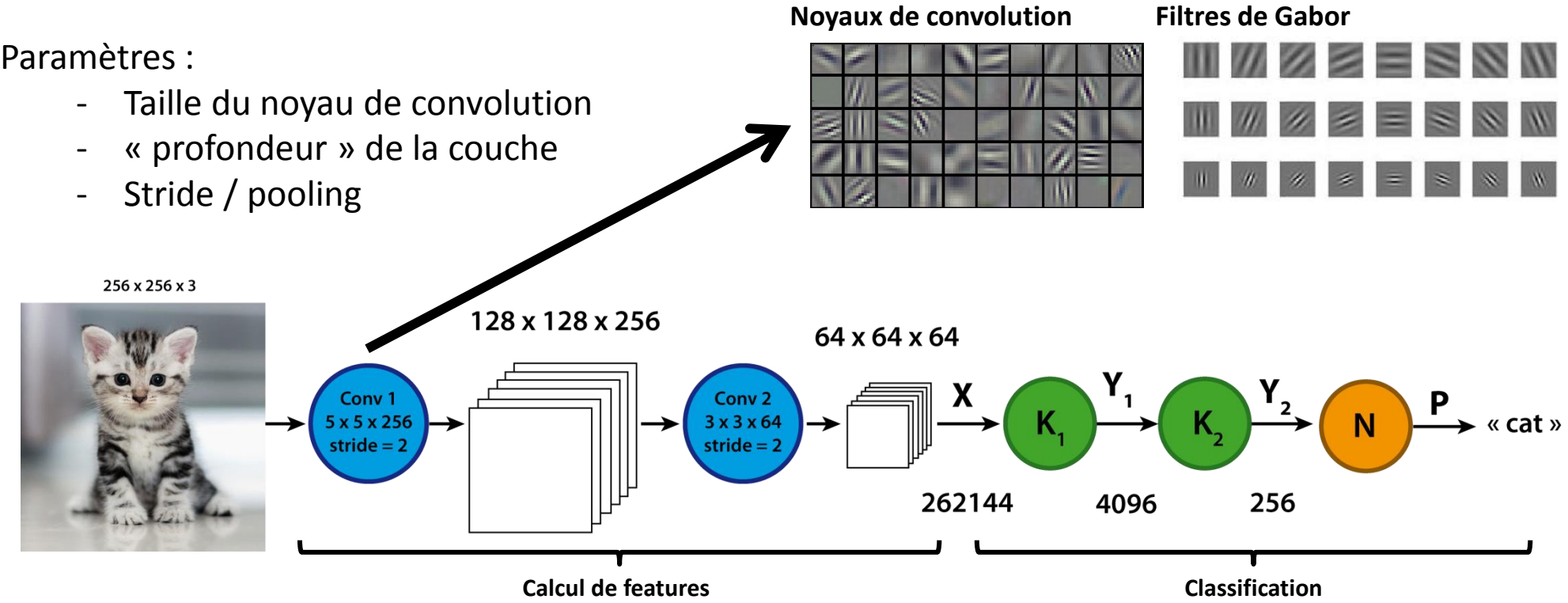


« Pooling »
Max / moyenne / …

Réseau de neurones convolutifs

Paramètres :

- Taille du noyau de convolution
- « profondeur » de la couche
- Stride / pooling



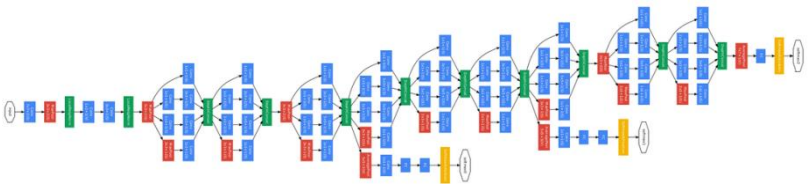
LeNet-5, Yann Le Cun (1990)

Reconnaissance de caractères (32x32)
2 convolutions / 2 fully-connected

Y. Le Cun *et al.*, « Handwritten digit recognition with a back-propagation network », in *Advances in neural information processing systems 2, NIPS 1989*, 1990, p. 396–404.

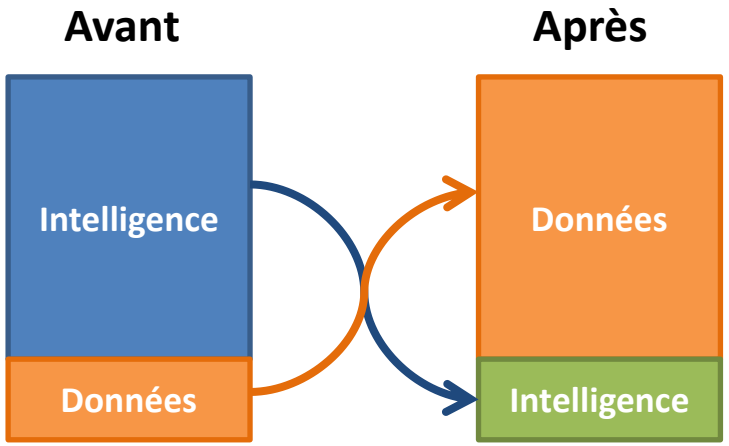
GoogLeNet, Inception, R-CNN (2014)

Reconnaissance d'objets (Imagenet)
59 convolutions / 5 fully-connected / pooling …



La raison

Conséquence

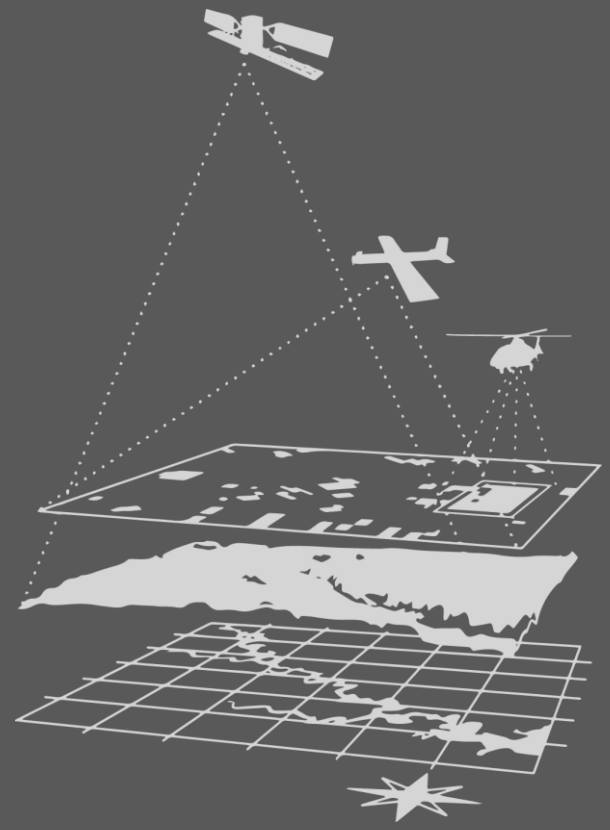


magellium

PUTTING KNOWLEDGE ON THE MAP

Le Deep Learning

Un cas concret



Détection de bâtiments dans des images satellites

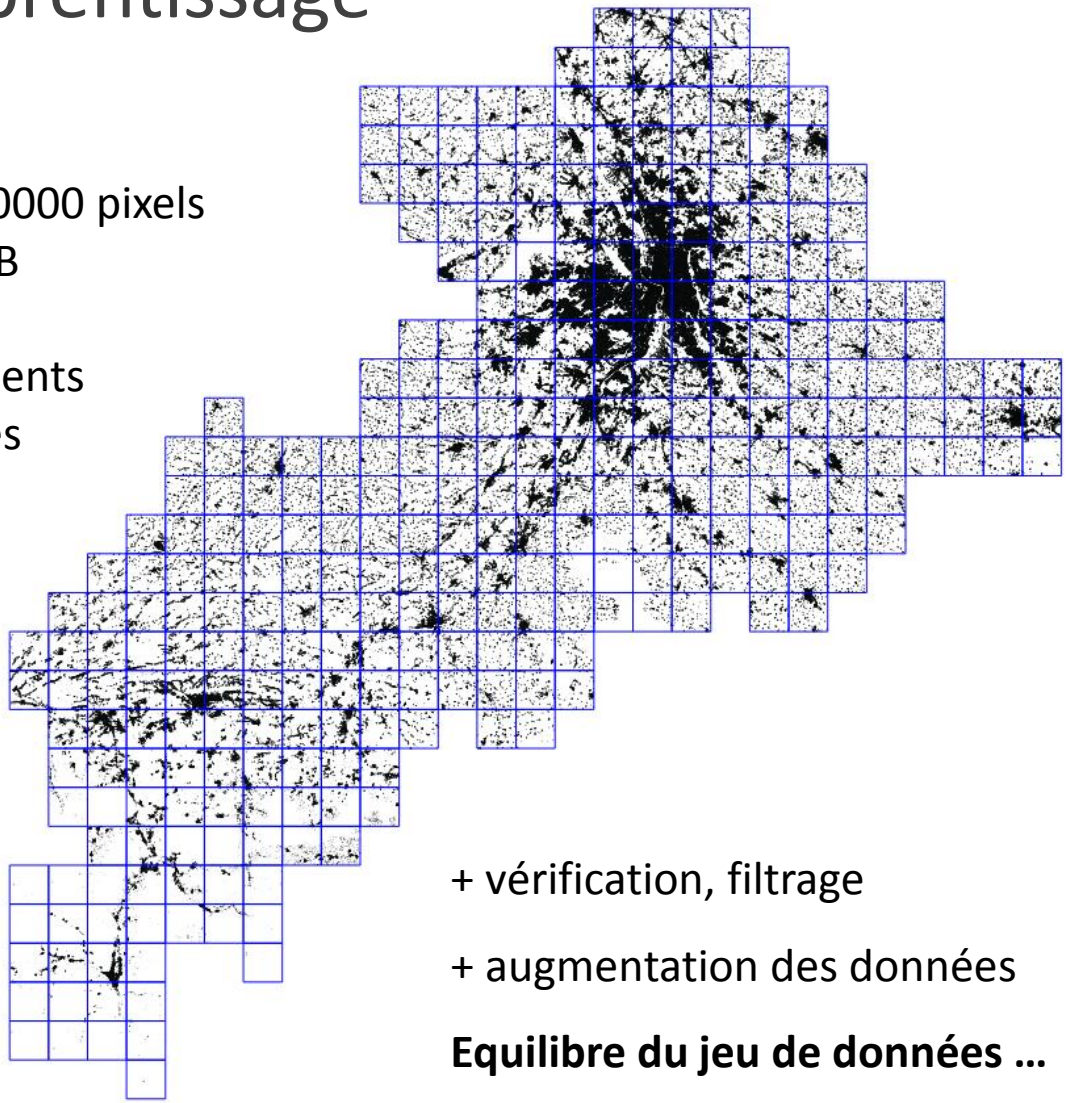
Jeux de données pour l'apprentissage

Images

PHR, 50cm
Haute-Garonne
328 images de 10000x10000 pixels
composition colorée RGB

Vecteur

Open Street Map, Batiments
environ 1M de polygones



Objectif

Produire des couples de vignettes
224x224 (image/masque batiment)

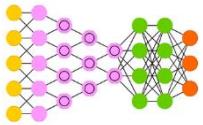
+ vérification, filtrage

+ augmentation des données

Equilibre du jeu de données …

Construction du réseau

Nombreuses structures existantes



http://www.asimovinstitute.org/neural-network-zoo/

La construction du réseau dépend de

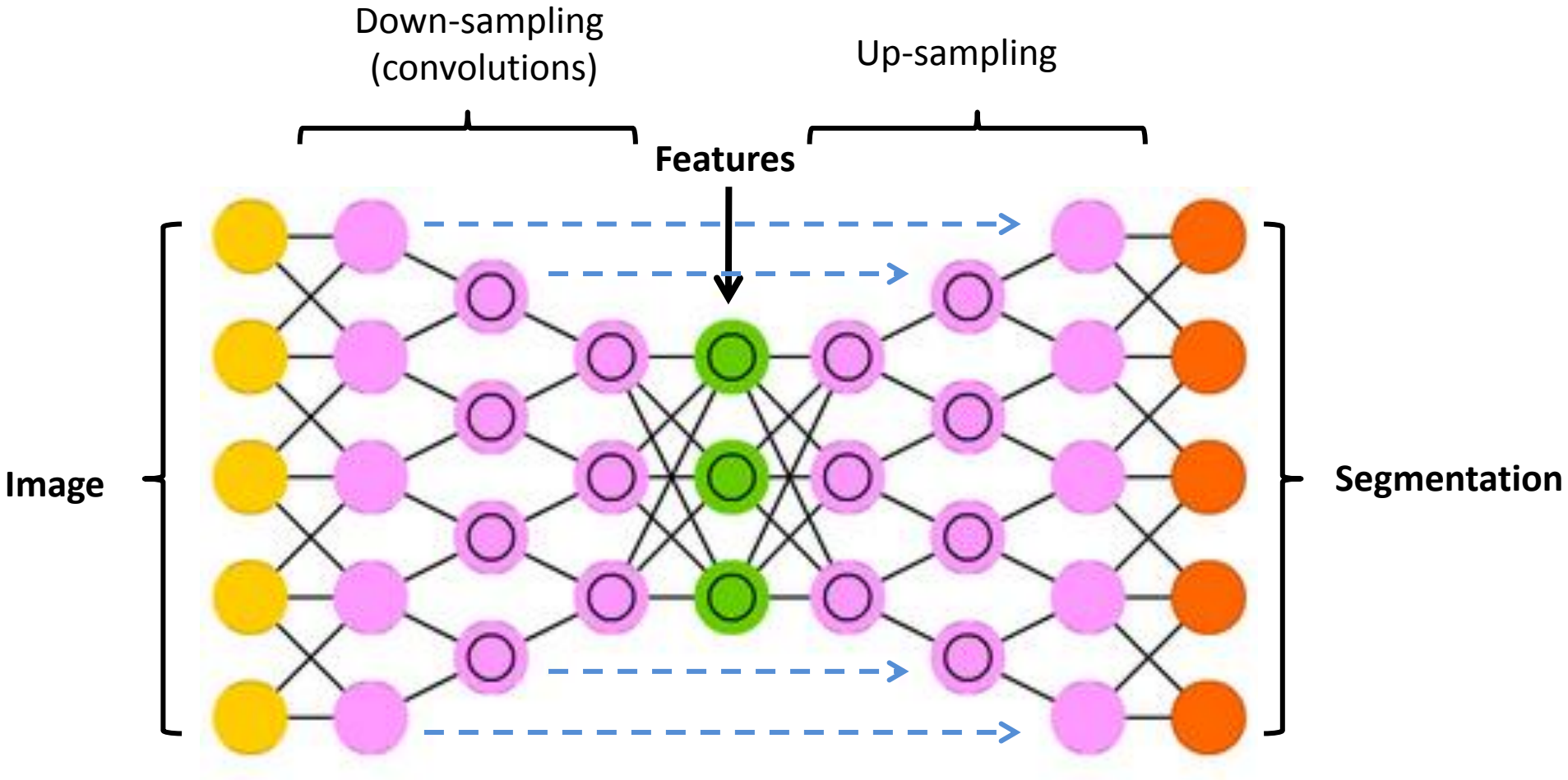
La nature du problème à résoudre
(segmentation, classification, etc…)

La quantité de données disponible

La puissance de calcul disponible (+ mémoire)

magellium

Architecture U-Net



Gestion de l'optimisation

Fonction coût

- Régularisations L1 ou L2
- Complexité ++

Méthode d'optimisation

- Plusieurs méthodes
- Hyper-paramètres

Augmentation de données « in-the-loop »

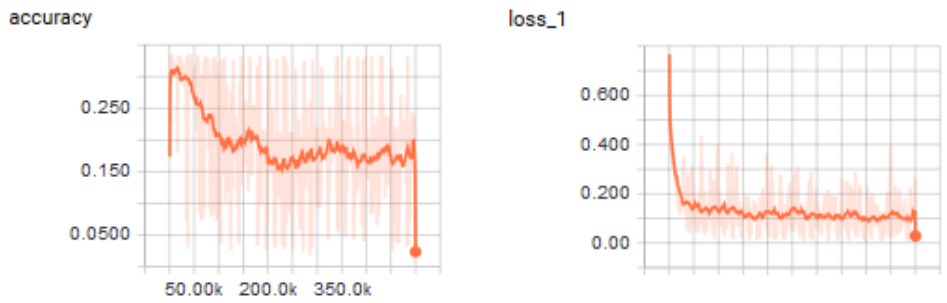
Variabilité ++

Luminosité / contraste / teinte / saturation

Hard-example mining

Présenter plus fréquemment les cas difficiles

Suivi de l'apprentissage



GeForce GTX 1080 – 8GB

100k itérations par batchs de 12 samples

Environ 40h de calculs

- Suivi de la fonction coût et de la précision
- Jeu de validation -> éviter le sur-apprentissage (overfitting)

Implémentation

Nombreuses bibliothèques existantes, la plupart en Python

PYTORCH Caffe2 theano Microsoft CNTK mxnet

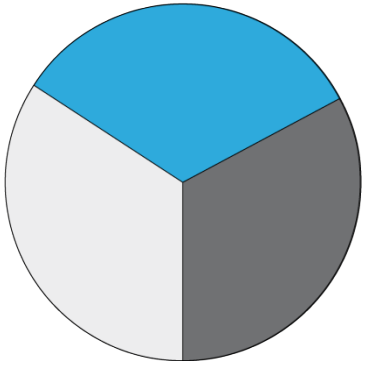
Notre choix

TensorFlow + Keras

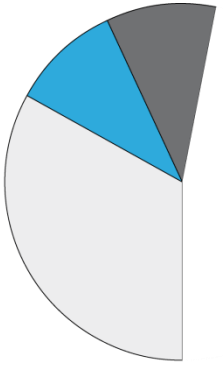
A développer :

- Construction du réseau
- Pipeline d'apprentissage
- Gestion du jeu de données

La 1ère fois

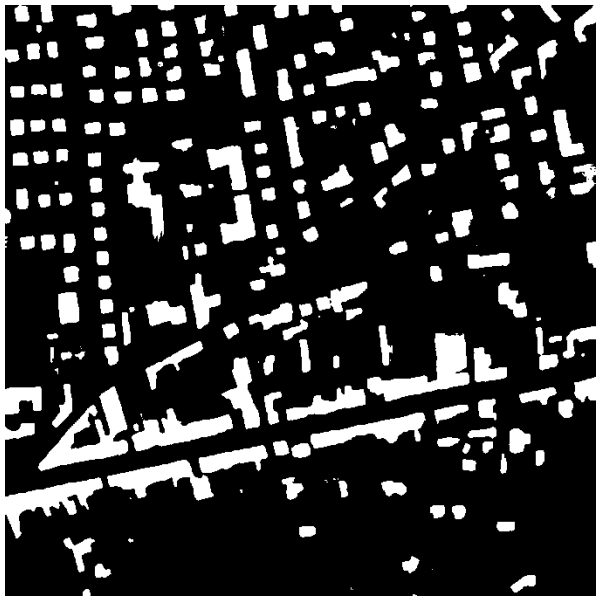
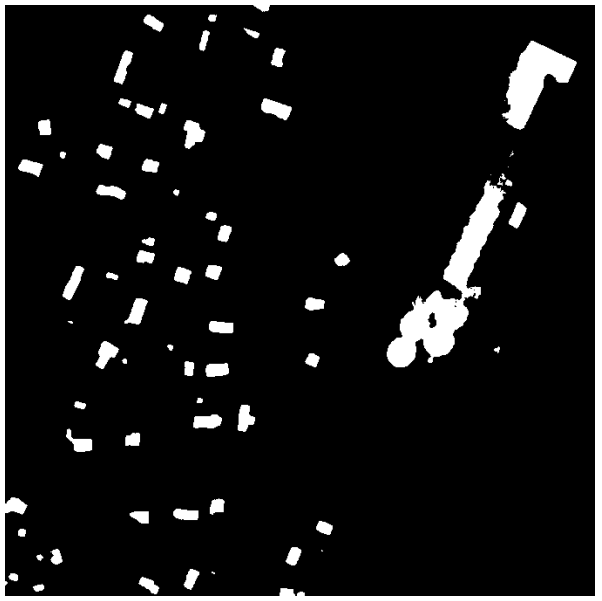
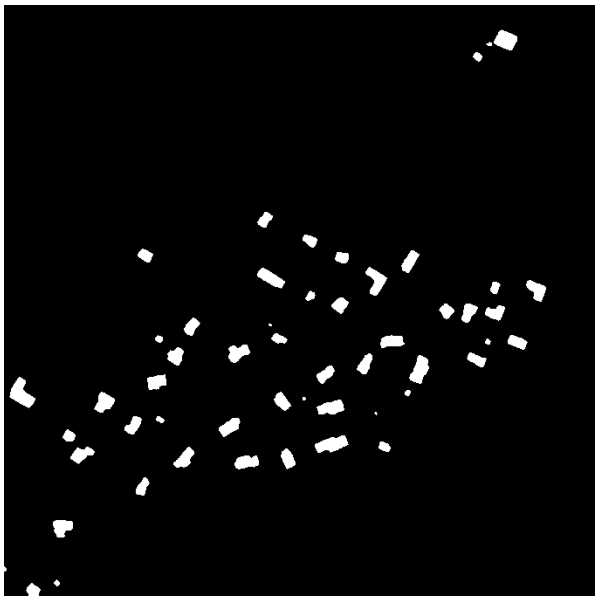


La 2ème fois

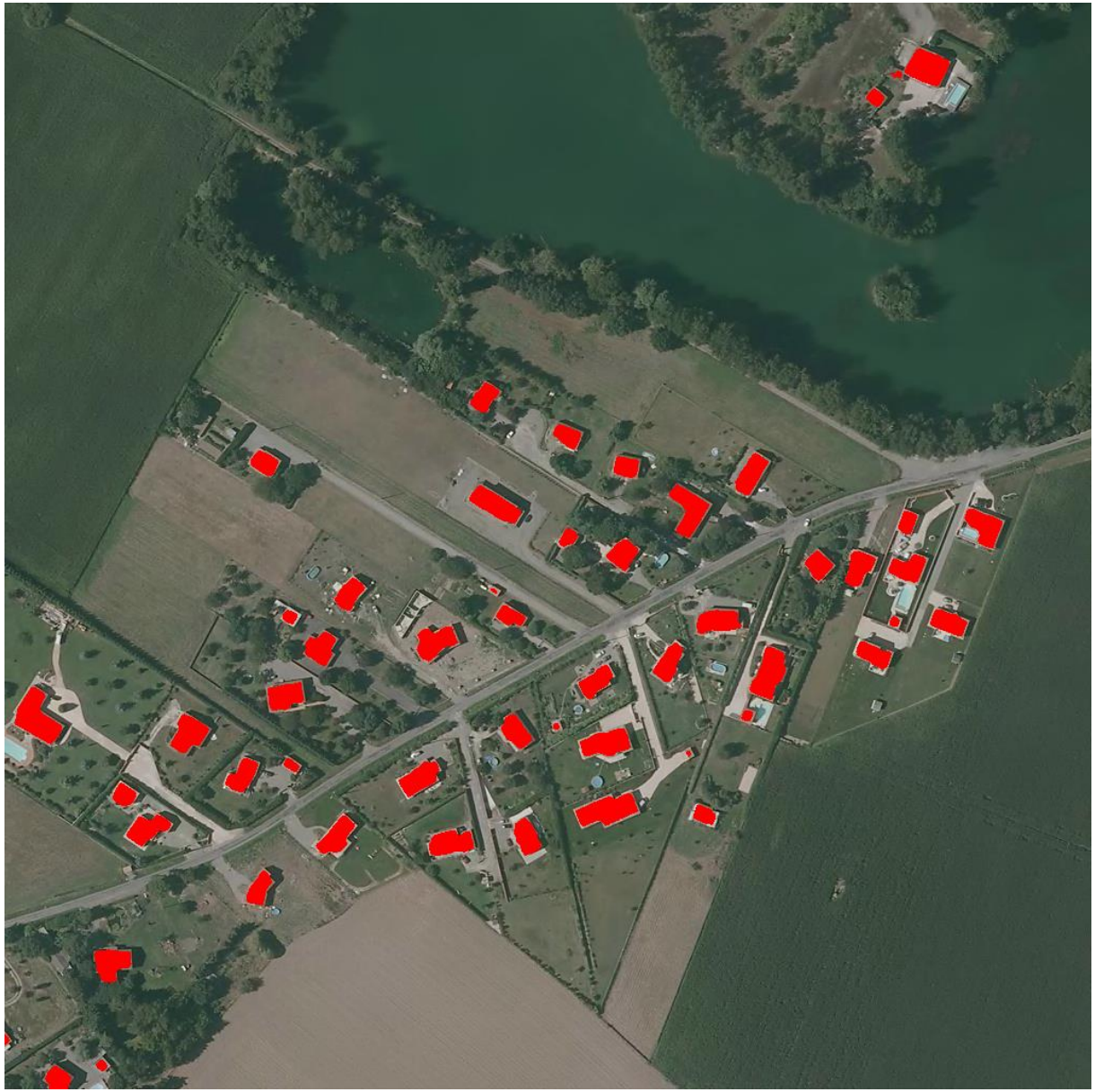


Résultats



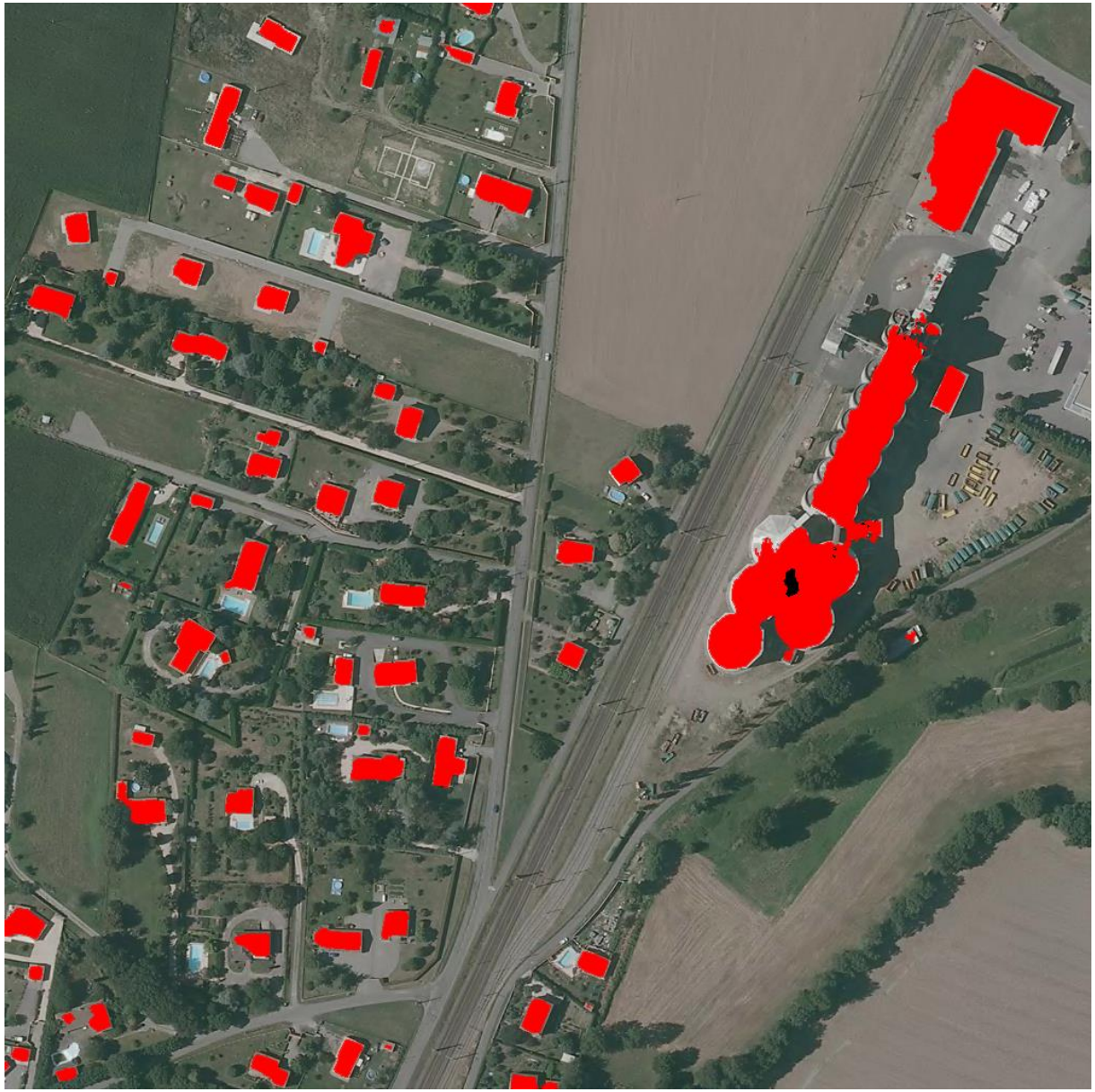





Résultats



Résultats

Résultats



Synthèse des points durs

Construction de réseaux de neurones

Demande un peu de savoir-faire et d'expérience
Lecture bibliographie souvent suffisante

Apprentissage du réseau

Effort d'adaptation aux librairies Deep
Savoir faire non négligeable sur la méthodologie

Jeux de données

Recul **spécifique** indispensable
Notions particulières d'équilibre, de fiabilité des jeux de données
Discipline particulière d'augmentation des données

magellium

PUTTING KNOWLEDGE ON THE MAP

Merci !

Thomas RISTORCELLI
Unité IA
thomas.ristorcelli@magellium.fr

François De Vieilleville
Unité EO
françois.devieilleville@magellium.fr

Sébastien Bosch
Unité GEO
sebastien.bosch@magellium.fr