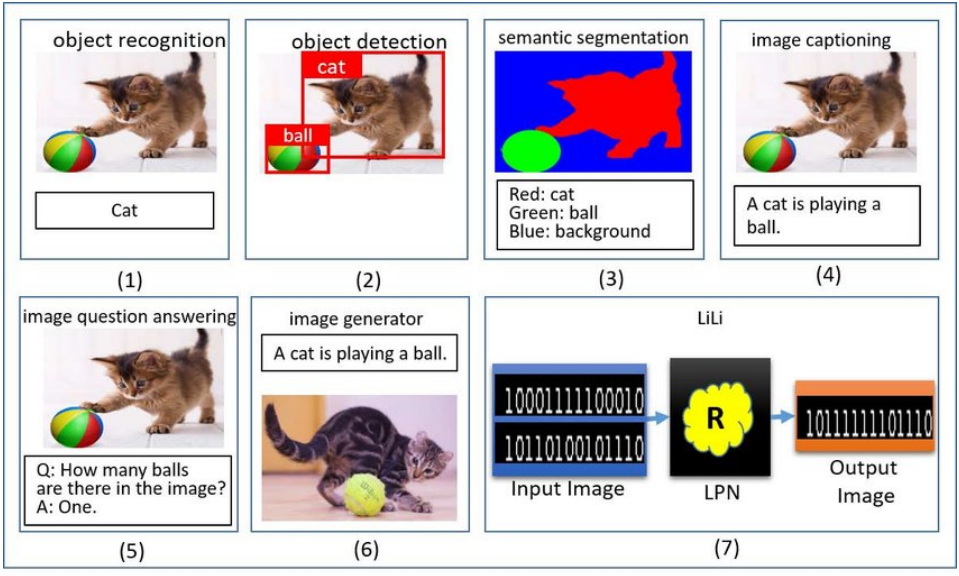
RÉSEAUX CONVOLUTIONNELS POUR L'IMAGE

Vincent Guigue,
inspiré des supports de Nicolas Baskiotis & Benjamin Piwowarski
vincent.guigue@agroparistech.fr

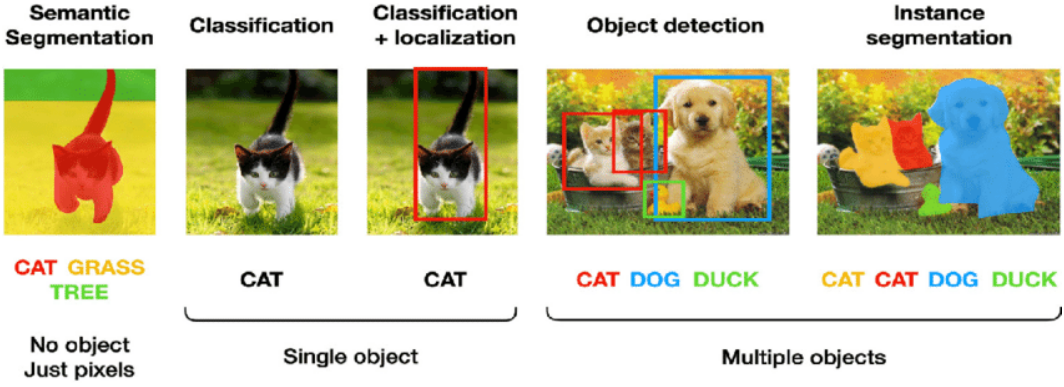
Introduction

Tâches en Computer Vision



Logic could be learned from images, Guo et al. 2019

Tâches en Computer Vision



Application of Deep Learning in Dentistry and Implantology, Kang et al. 2020

Le domaine du *Computer Vision*

Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications, Mueller et al. 2018

Pourquoi les MLP ne sont pas suffisants ?

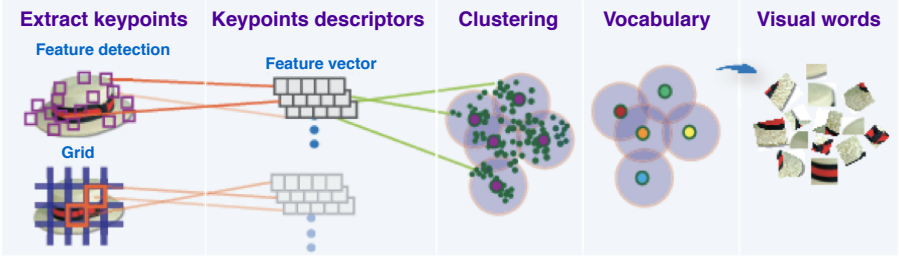
- Image = grille de pixels $\Rightarrow$ sémantique très pauvre
- Fully-connected (MLP) $\Leftrightarrow$ une entrée = un rôle propre
  $\Rightarrow$ dépendance à la position du pixel

$\neq$ Dans une image, l'absolu n'a pas d'importance, le relatif est bien plus important (mais pas seulement)

Les *invariants* sont très importants en image car un objet peut prendre de multiples formes !

Avant le deep

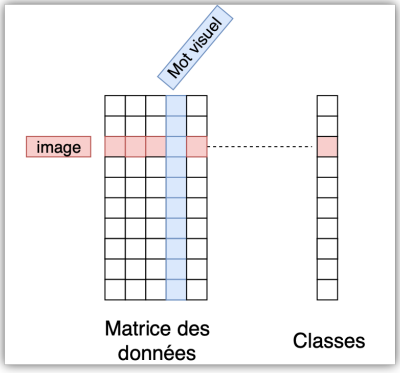
1. extraire un dictionnaire de *features*
2. agréger les features (représentation type *Bag Of Word*)
3. utiliser un classifieur sur cette représentation pour classifier



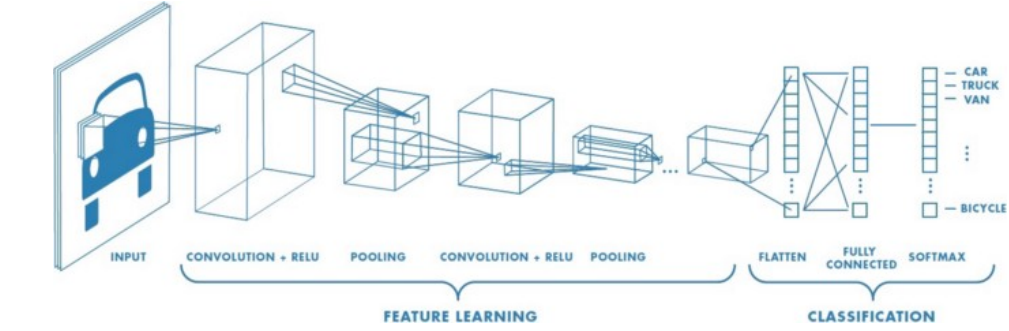
Visual Categorization with Bags of Keypoints, Csurka et al. W-ECCV 2004

Avant le deep

1. extraire un dictionnaire de *features*
2. agréger les features (représentation type *Bag Of Word*)
3. utiliser un classifieur sur cette représentation pour classifier



Réseaux convolutifs

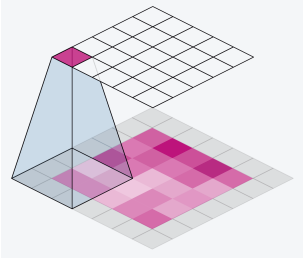


NVidia

Convolution

Filtre d'analyse glissant sur l'image

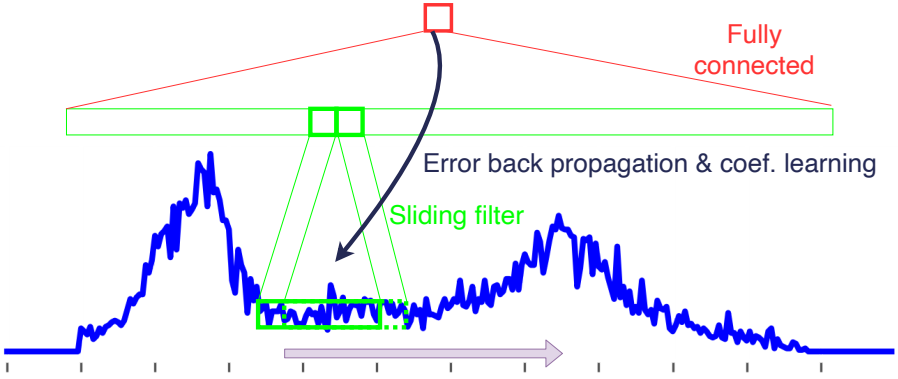
- Peu de paramètres
- Apprentissage des motifs à extraire
- Agrégation progressive des échelles



Convolution

Convolution 1D

Commençons par un exemple 1D pour bien comprendre.

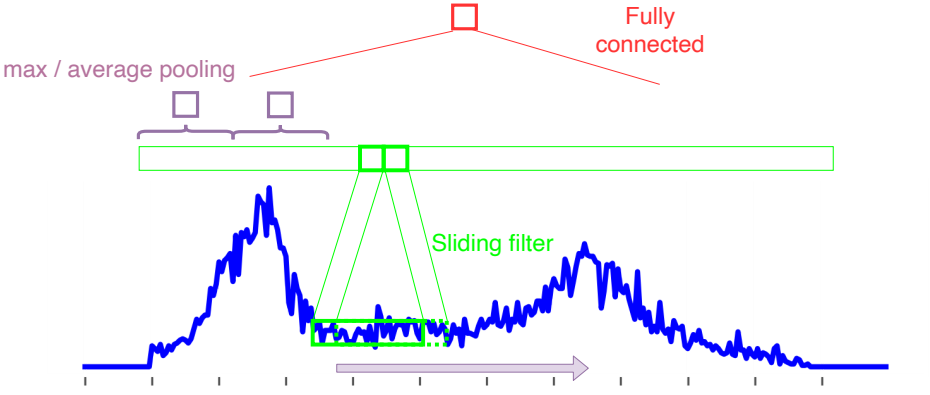


- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 1D

Commençons par un exemple 1D pour bien comprendre.

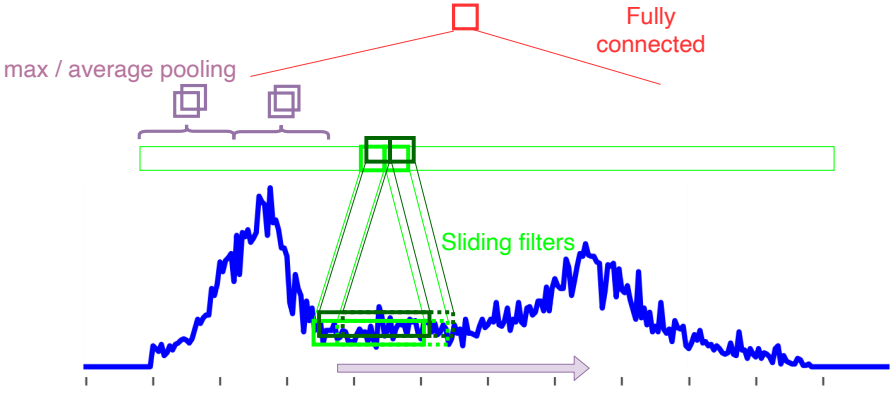


- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 1D

Commençons par un exemple 1D pour bien comprendre.

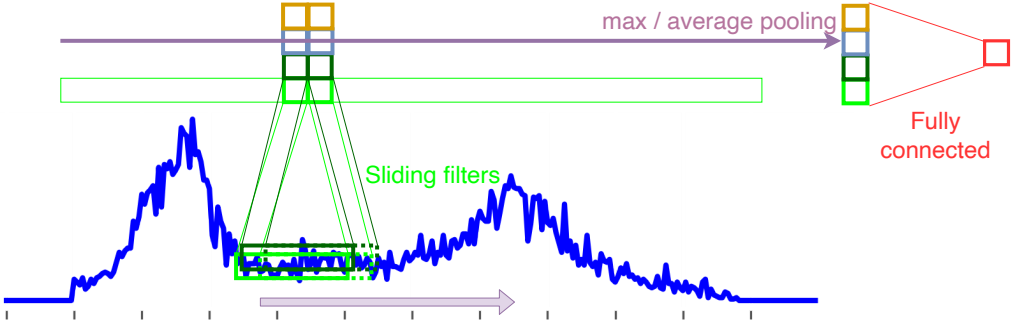


- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 1D

Commençons par un exemple 1D pour bien comprendre.

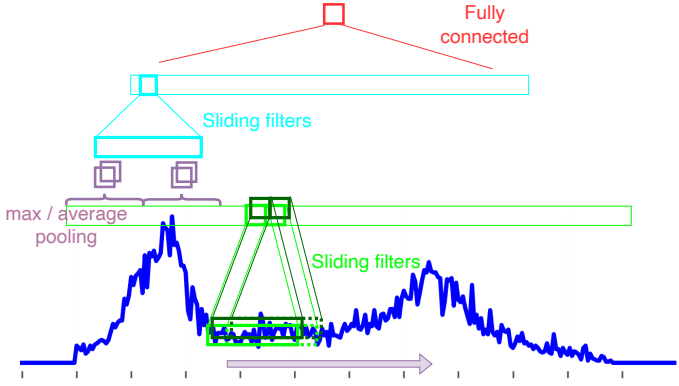


- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 1D

Commençons par un exemple 1D pour bien comprendre.

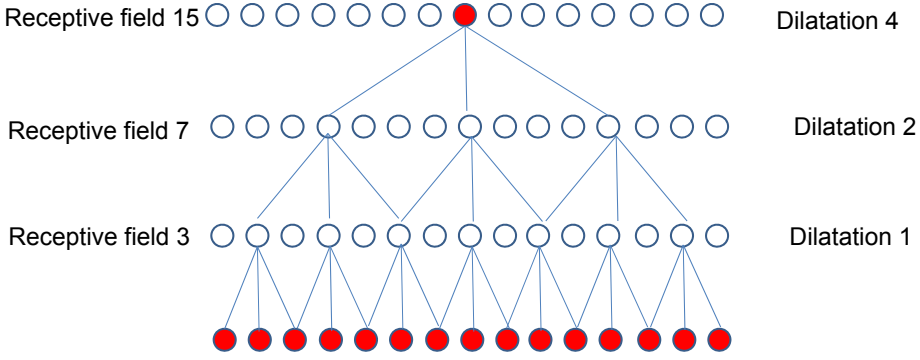


- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 1D

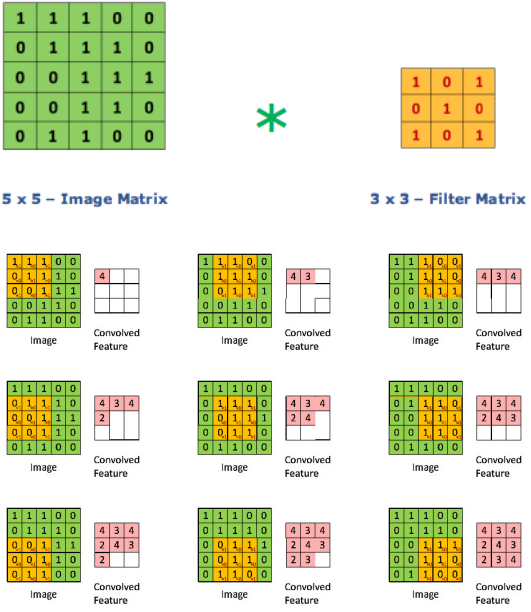
Commençons par un exemple 1D pour bien comprendre.



- Fenêtre glissante
- Peu de paramètres
- Largeur / stride / padding

- Apprentissage des motifs à détecter
- Invariance / dépendance à la localisation

Convolution 2D



Couche de *Pooling*

Pooling (ou subsampling) : Réduire la dimensionalité de sortie

- Max Pooling : on prend le max sur une fenêtre
- Average Pooling : on fait la moyenne
- Sum Pooling : la somme ...

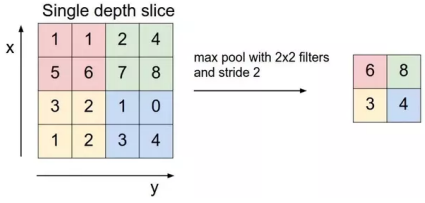
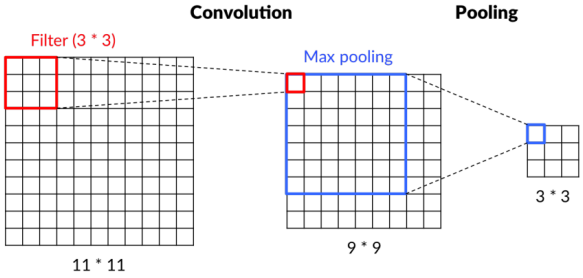
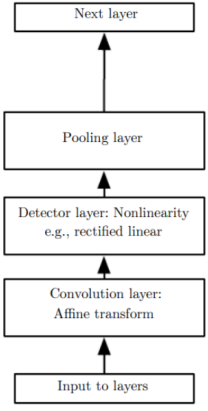


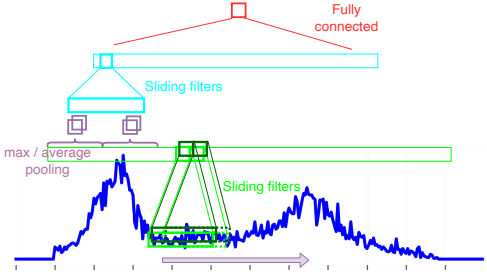
illustration : https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

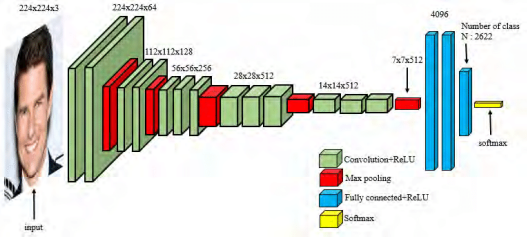
Couche convolutionnelle usuelle

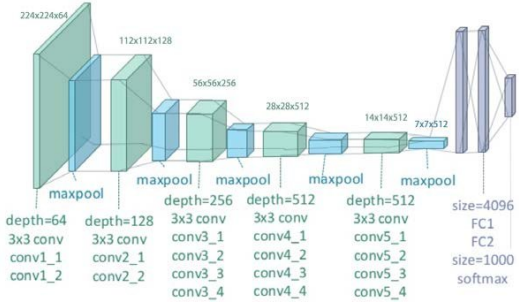




Couche convolutionnelle usuelle







Exemple

Reconnaissance de caractères [Duda et al 00]

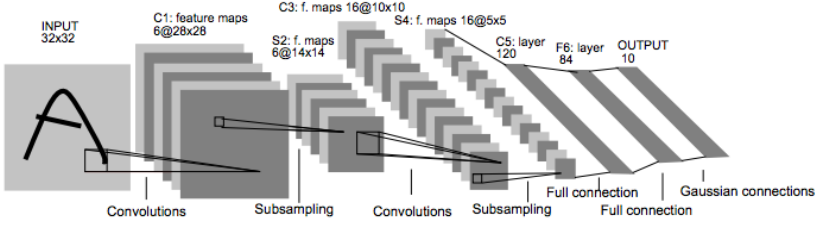
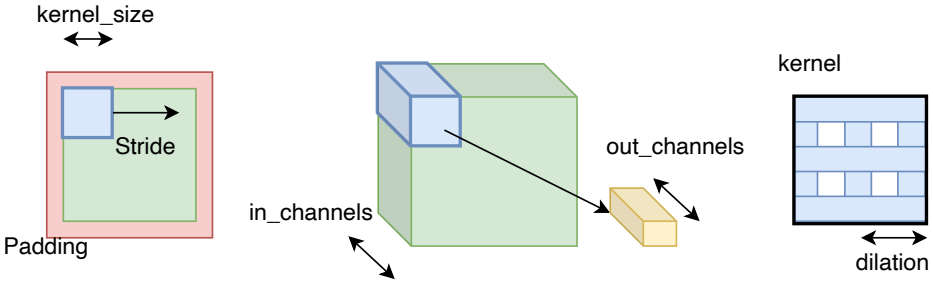


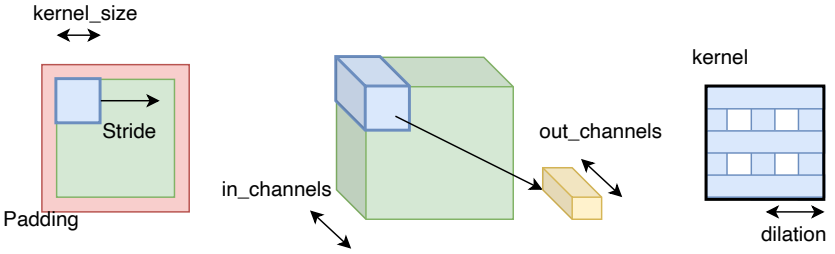
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Récapitulatif des paramètres




```
1     class ConvNet(nn.Module):
2     # argument sur le choix du pooling
3     def __init__(self,pooling = nn.MaxPool2d):
4         super().__init__()
5         # Convolution 5*5, 16 filtres
6         self.conv1 = nn.Conv2d(in_channels=1, out_channels=16,
7             kernel_size=5, stride=1, padding=0, dilation=1)
8         # Max pooling 3x3
9         self.pool1 = pooling(kernel_size=3, stride=1)
10        ...
```

Input/Output dimensions



This formula should give you the output size per channel

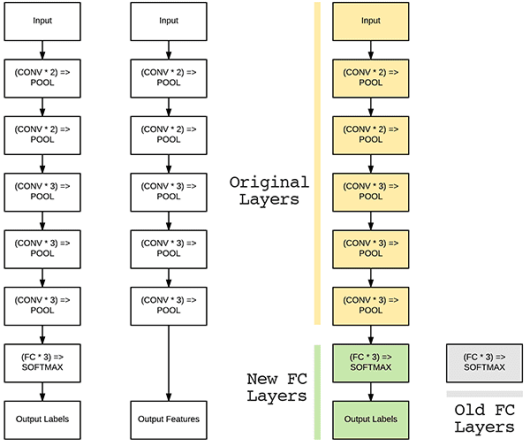
$$\frac{(W - K + 2P)}{S} + 1$$

- W is the input volume
- K is the Kernel size
- P is the padding
- S is the stride

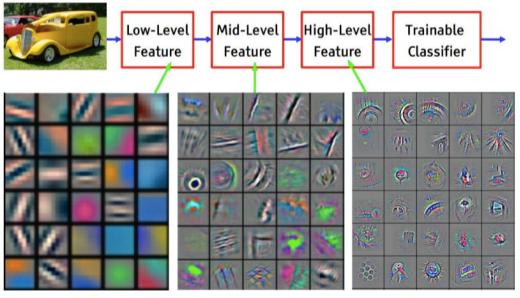
Fine-tuning approach

- Learning a network from scratch ⇒ OK, good descriptors...
  - Require a lots of data
  - (Often) not optimal performance

⇒ Fine-tune a network



- How many layers to keep / re-train?

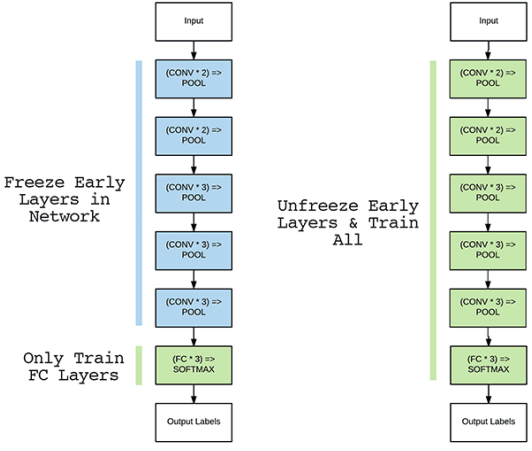


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

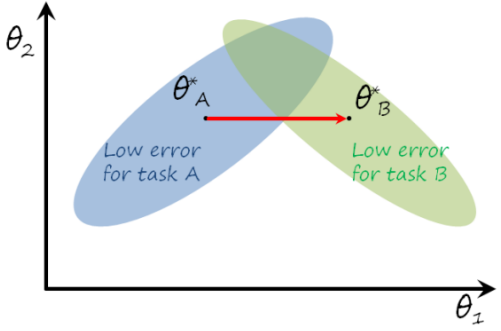
Fine-tuning approach

- Learning a network from scratch ⇒ OK, good descriptors...
  - Require a lots of data
  - (Often) not optimal performance

⇒ Fine-tune a network

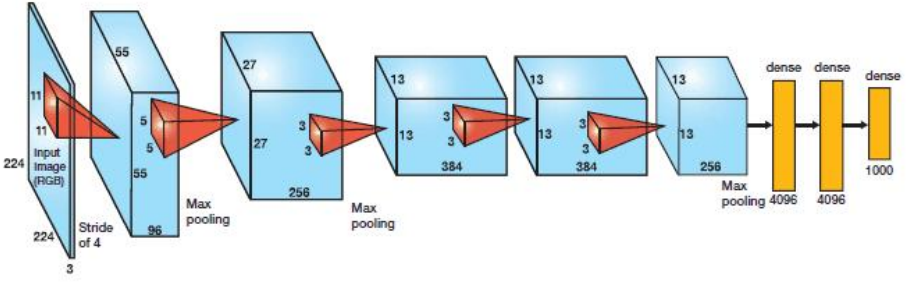


- Do we need to fine-tune all?
- Catastrophic forgetting



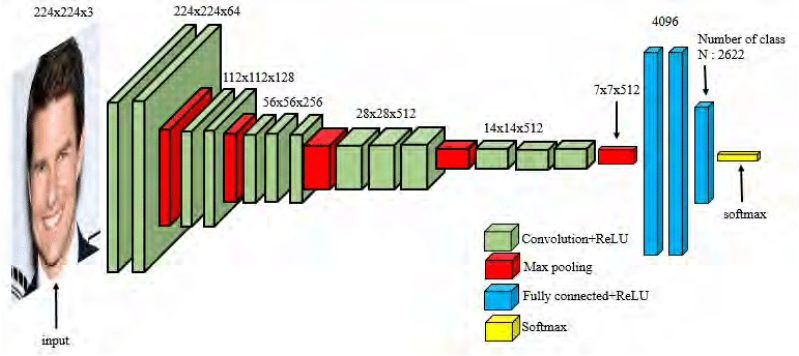
Architectures en image (vision)

AlexNet (2012)

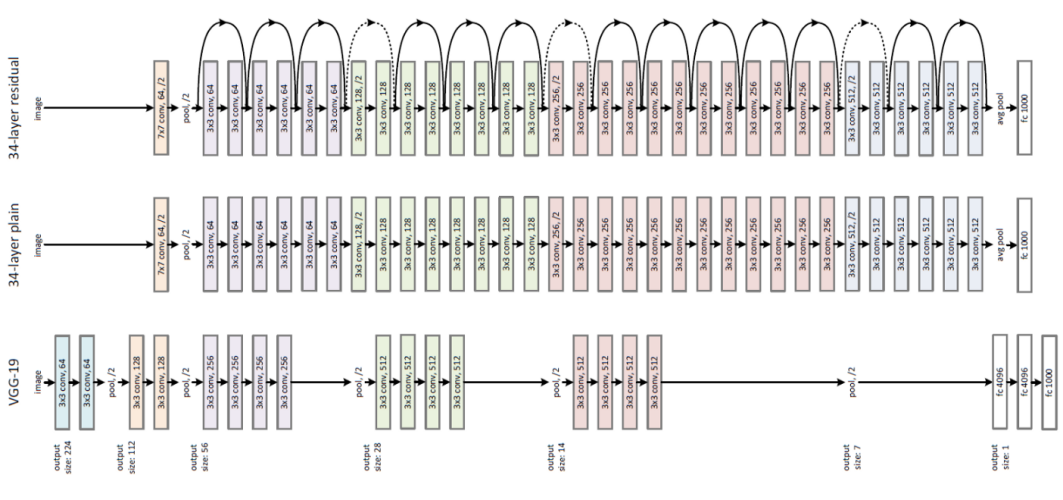


- 11×11, 5×5, 3×3 convolutions
- Max-pooling, ReLU activations
- Dropout et Data-augmentation

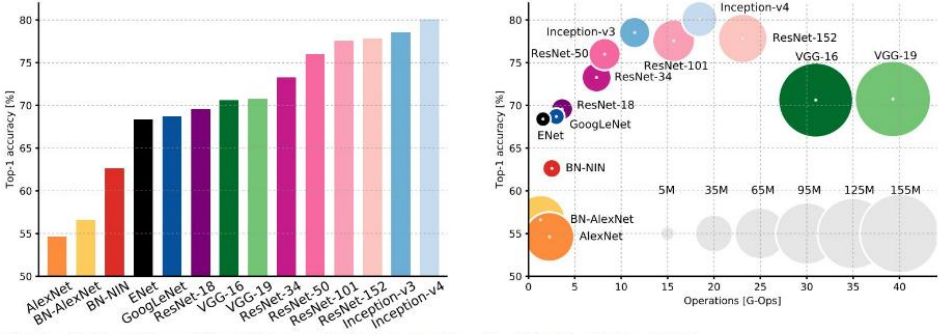
Deep CNN / VGG / ResNet



Deep CNN / VGG / ResNet



Un grand nombre d'architectures



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Image Reconstruction

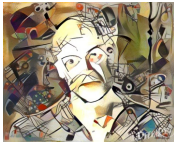
Generate images by combining content and style

Makes use of a discriminatively trained CNN

Image generation

- inverse problem on the CNN

https://deepart.io



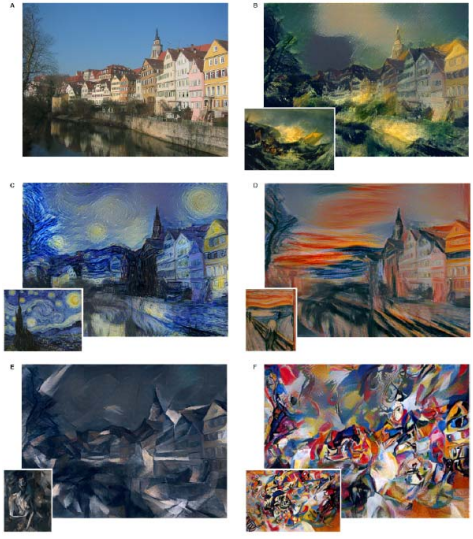
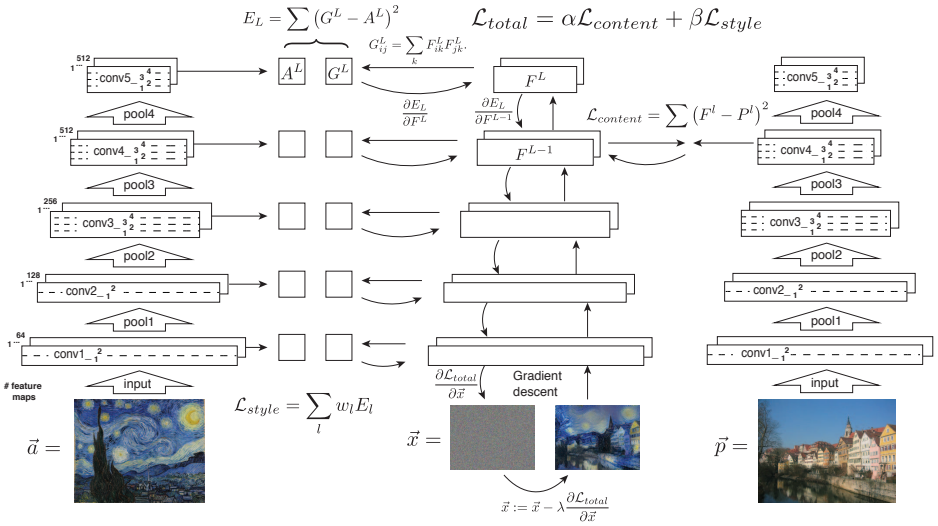


Image Reconstruction



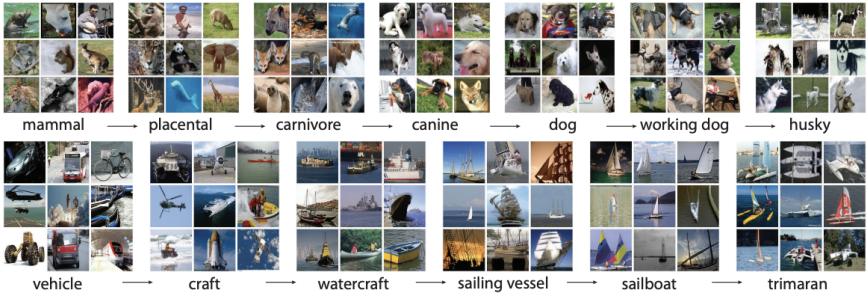
Different problems, models & metrics

$$\begin{array}{ccc} \mathcal{I} & & \mathcal{C} \\ X & \mapsto & f(X) \end{array}$$

`VGG, ResNet, etc`

Classification problem: $\mathcal{I} = RGB^{d \times d}$
1000 classes (ImageNet)
Metrics: accuracy, precision, recall

How to switch to multi-class?



$\Rightarrow$ Think about transfer learning!

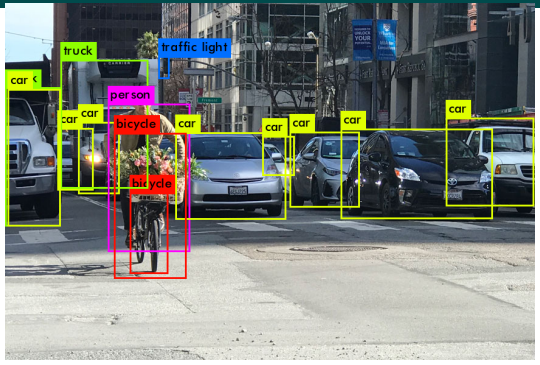
Different problems, models & metrics

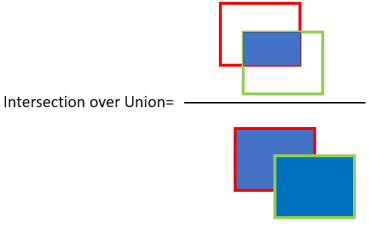
$$\begin{array}{ccc} \mathcal{I} & & \mathcal{C} \times \mathcal{B} \\ X & \mapsto & f(X) \end{array}$$

$$\mathcal{B} = (pos_x, pos_y, width, height)$$

$\Rightarrow$ A regression problem (!)

`Yolo, RCNN, fast-RCNN`





Metrics: Geometric area intersection

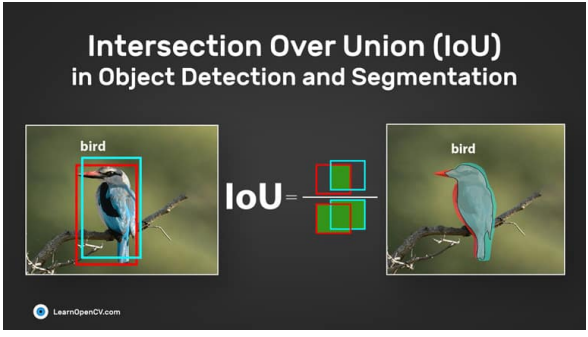
Different problems, models & metrics

$$\begin{array}{ccc} \mathcal{I} & & \mathcal{I}' \\ X & \mapsto & f(X) \end{array}$$

Pixel level annotation

⇒ How to output an image?

U-Net



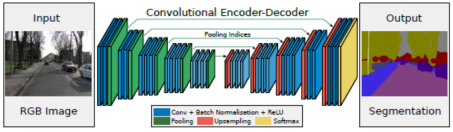
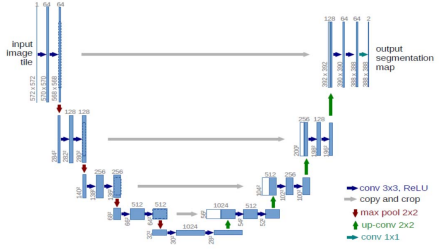


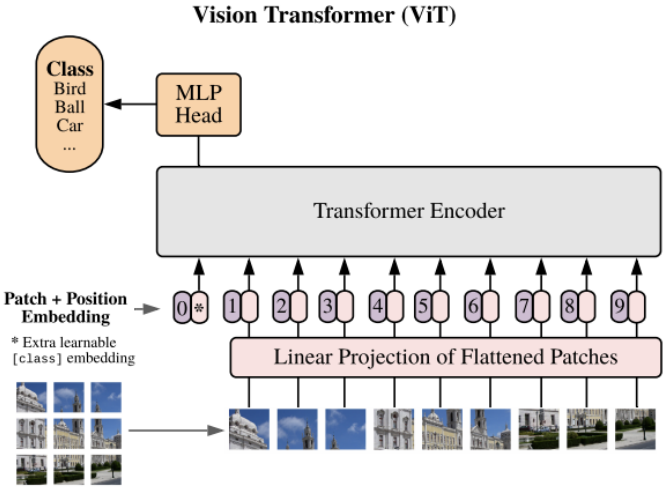
Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.



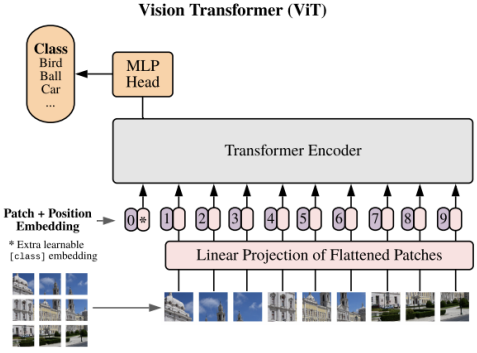
⇒Specificities wrt auto-encoders?

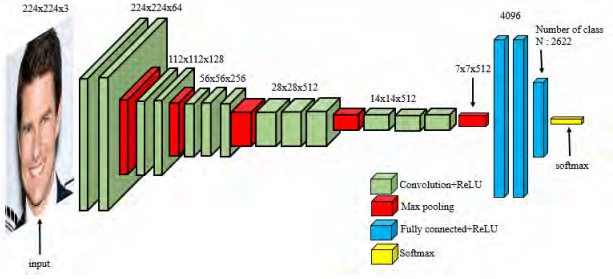
The Transformer Wave

Using transformer architecture for image processing



The Transformer Wave



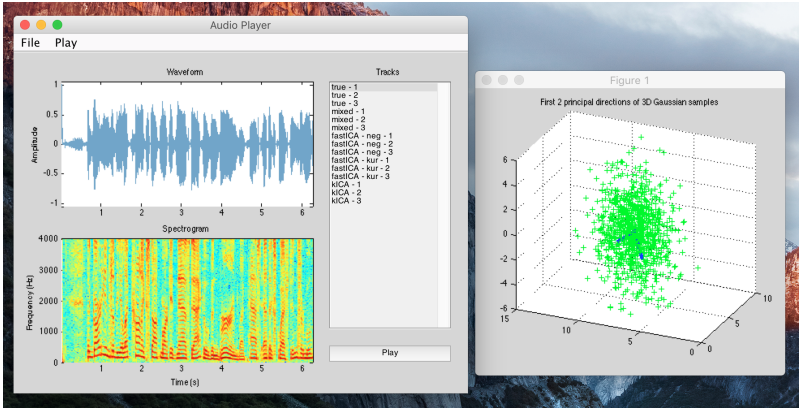


⇒ Which differences / complementarities with CNN?

CNN & Time-Frequency representation

The example of source separation (that makes great progress over the last 5 years)

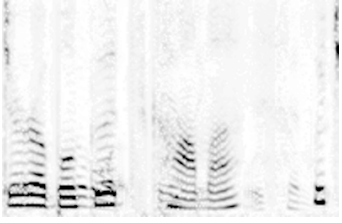
Original problem: ICA (independant componant analysis)
SVD algorithm (unsupervised) in time or time frequency domain:

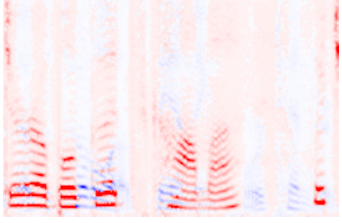


CNN & Time-Frequency representation

The example of source separation (that makes great progress over the last 5 years)

New Problem:
A supervised classification problem in the time frequency domain

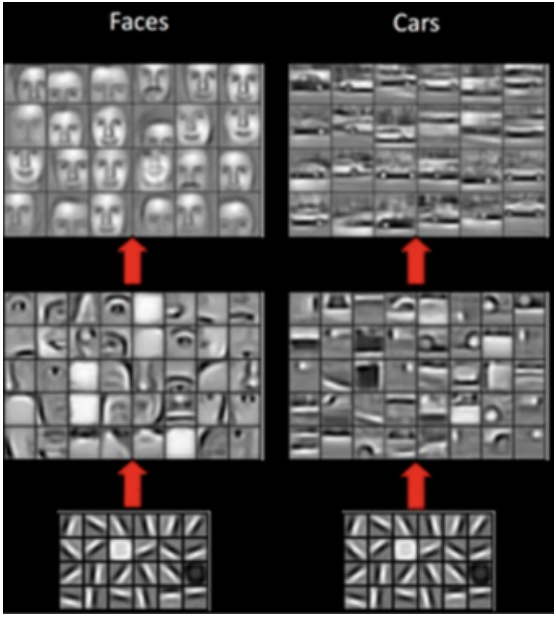




Introspection

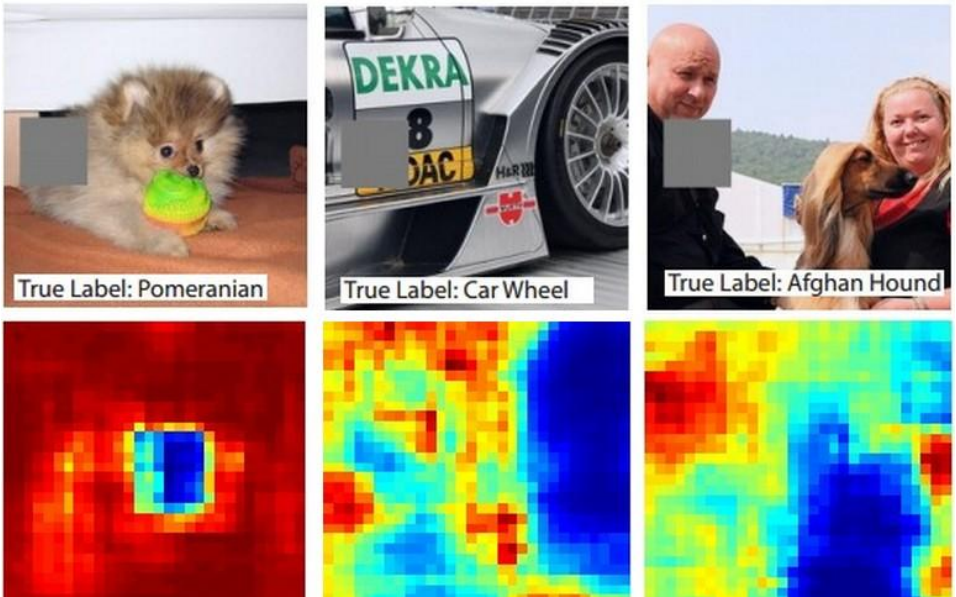
Introspection d'un CNN

- Comprendre la classification : quelle région de l'image active la classification, quels filtres sont les plus importants ...
- Les filtres sont de plus en plus abstraits, produisant des mots élémentaires visuels
- Les couches conservent les informations topologiques

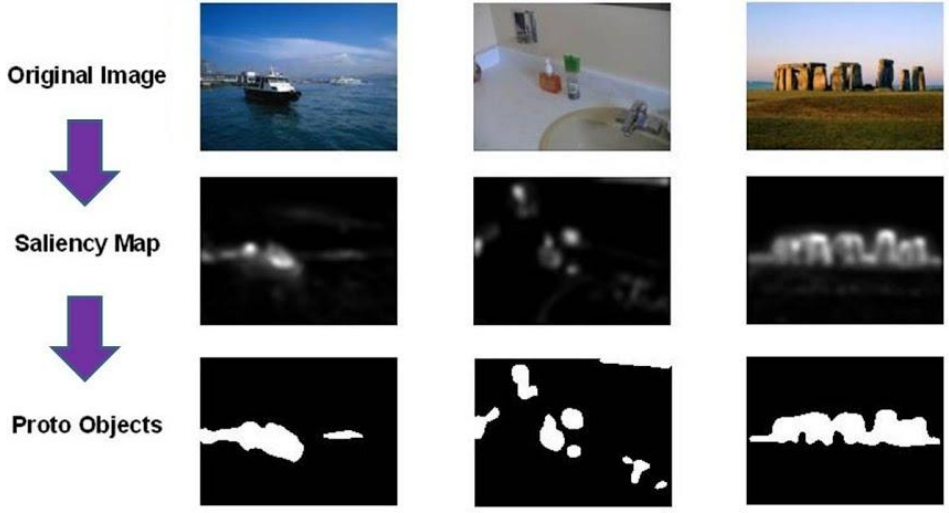


Occlusion

Sensibilité de la classe détectée aux occlusion:

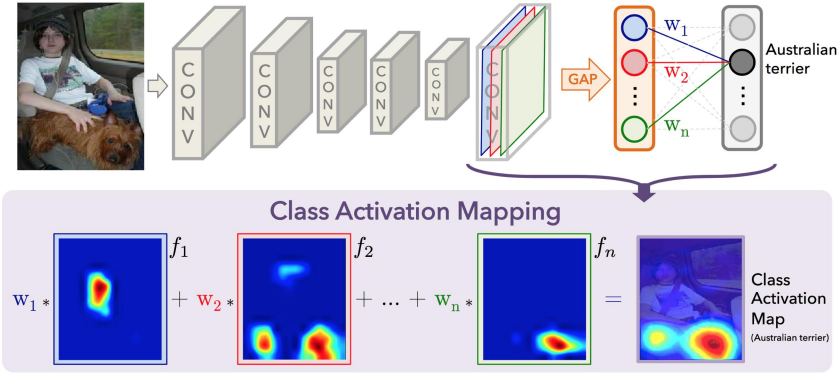


Carte de saillance



Elles sont calculées en prenant le gradient de la sortie par rapport à l'image d'entrée. Elles permettent de mettre en avant les pixels auxquels la classification est la plus sensible.

Class Activation Map

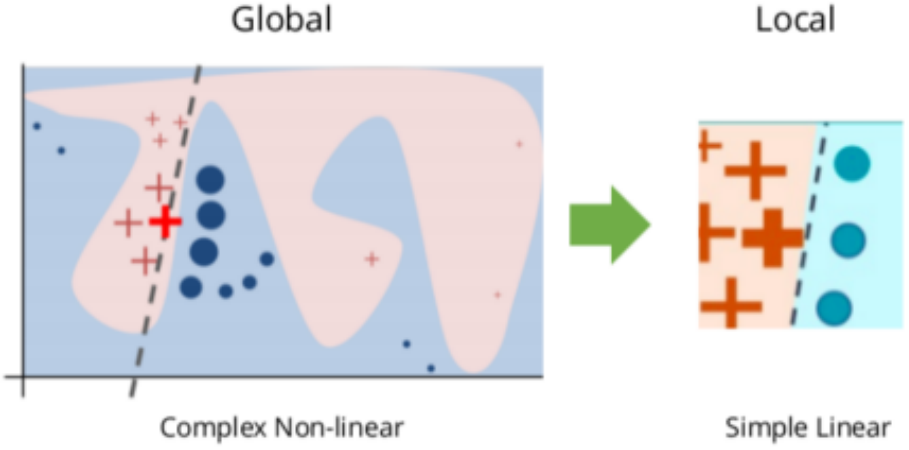


Class Activation Maps :

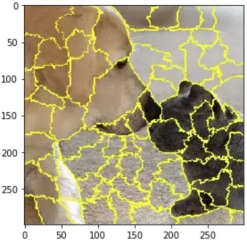
- Global Average Pooling (1 filtre = 1 feature) + pondération
- Combinaison linéaire
- Agrégation pondérée des filtres pour indiquer les régions d'intérêts.

Local Interpretable Model-Agnostic Expl. (LIME)

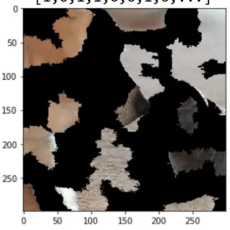
- Sampling ($P \approx \text{sim}$)
- Local linear model (on weighted data)
- Linear coefficients interpretation



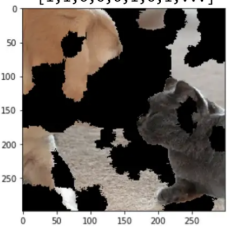
Local Interpretable Model-Agnostic Expl. (LIME)



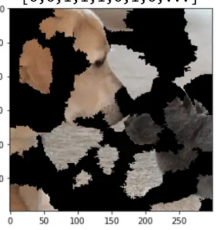
perturbation1=
[1,0,1,1,0,0,1,0,...]

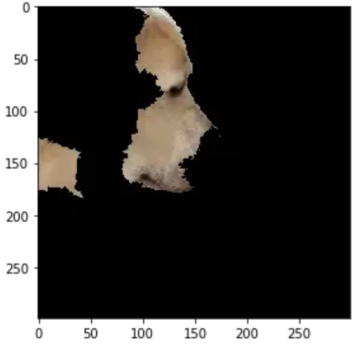


perturbation2=
[1,1,0,0,0,1,0,1,...]



perturbation3=
[0,0,1,1,1,0,1,0,...]

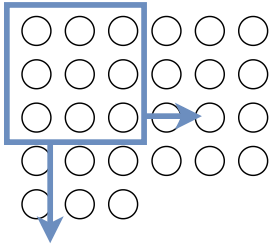


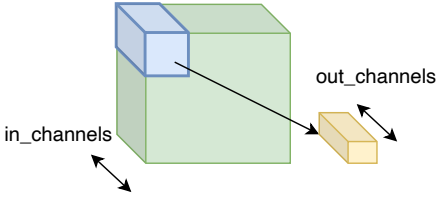


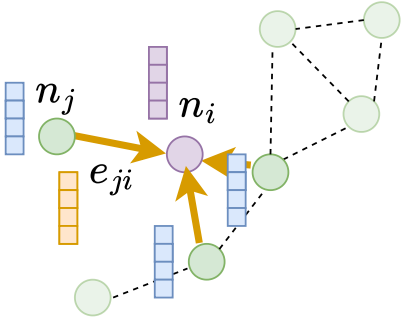
Graph Neural Networks

Passer des convolutions à une logique de graphe

Convolution = outils sur une grille régulière ⇒ passage à des liens quelconques



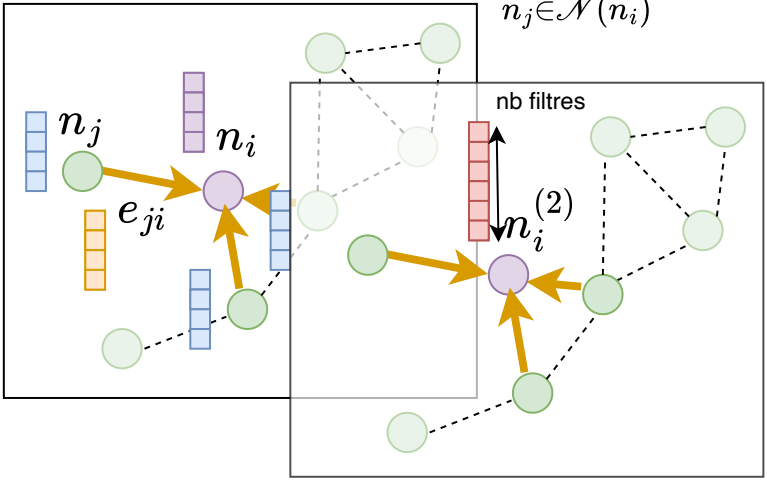




Passer des convolutions à une logique de graphe

Convolution = outils sur une grille régulière ⇒ passage à des liens quelconques

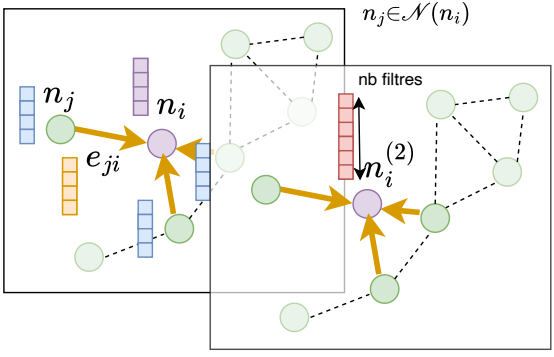
$$n_i^{(2)} = \sum_{n_j \in \mathcal{N}(n_i)} \psi(n_i, n_j, e_{ji})$$



Passer des convolutions à une logique de graphe

Convolution = outils sur une grille régulière ⇒ passage à des liens quelconques

$$n_i^{(2)} = \sum_{n_j \in \mathcal{N}(n_i)} \psi(n_i, n_j, e_{ji})$$



Message Passing

- ψ Réseau de neurones (e.g. PMC)
- Nb couches ++ ⇒ ↗ receptive field
- Possibilité de brancher des *losses* sur
  - nœuds/nodes
  - arcs/edges
  - graphes entiers

[trouver un agrégateur]