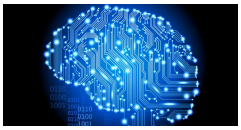
APPRENTISSAGE PROFOND EN IMAGES ET VISION PAR ORDINATEUR

Alexandre Meyer[1]
[1]Equipe SAARA, laboratoire LIRIS

Master ID3D



1

Papiers à lire

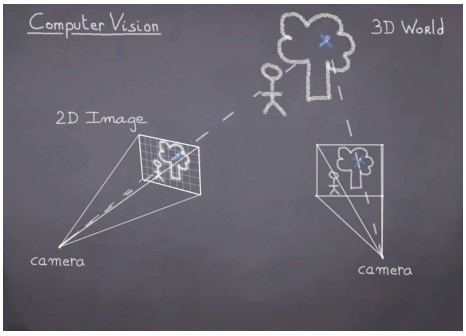
Soccer on Your Tabletop
Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, Steve Seitz
CVPR 2018
https://arxiv.org/abs/1806.00890

FaceNet: A Unified Embedding for Face Recognition and Clustering
Florian Schroff, Dmitry Kalenichenko, James Philbin
CVPR 2015
https://arxiv.org/abs/1503.03832

???
???
???

2

Vision par ordinateur



3

Vision par ordinateur et vision humaine

≠

Nature

inspire l'

Artificiel

≠

4

Vision par ordinateur et vision humaine

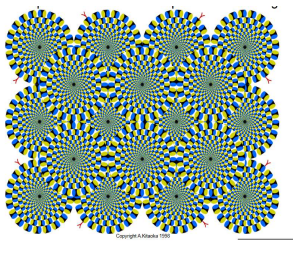
- Vision par ordinateur reste limitée même si d'énormes progrès ont été réalisés …

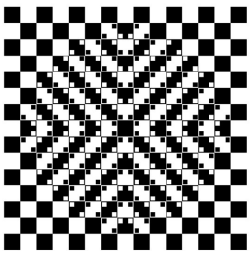
26x27 pixels

5

Vision par ordinateur et vision humaine

- Vision par ordinateur reste limitée
  - Pour l'instant

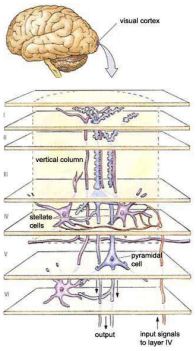




6

Vision par ordinateur et vision humaine

- Vision par ordinateur
  - Traitement d'images
    - Changer la luminosité
    - Mettre en évidence certains aspects
    - …
  - Reconnaissance des formes
    - Retrouver les lignes, les cercles, etc.
    - …
    - Retrouver des visages
  - Vision par ordinateur
    - Identifier les motifs
      - « On voit un visage humain »
      - « Il s'agit de Paul »
    - Identifier des actions
      - « La personne vis un boulon »
    - Analyser une action



7

Vision par ordinateur

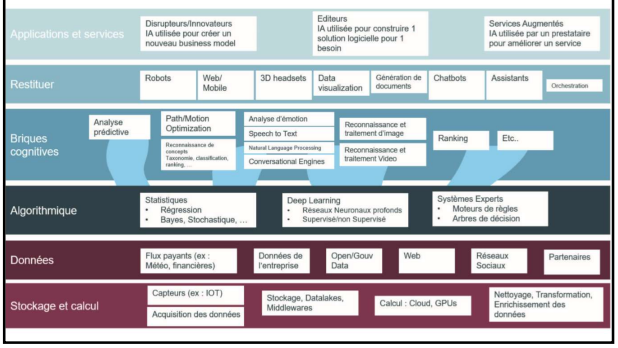
8

RESEAUX DE NEURONES (PROFONDS)

AI
- …
- Machine Learning (apprentissage machine)
  - Random Forest
  - SVM
  - Baysien
  - …
  - Neural Network
  - Deep Learning (dans ce cours voir ca comme un « outils »)
  - Apprentissage par renforcement

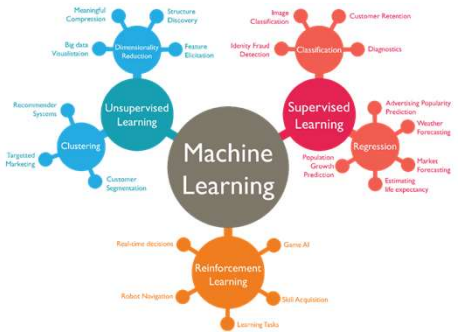
9

IA / ML / DL



10

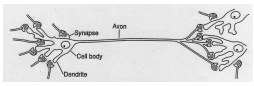
ML : Supervisé / Non supervisé



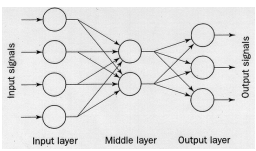
11

Réseau de neurones artificiel (RNA) un modèle de calcul inspiré du cerveau humain

- Cerveau humain :
  - 10 milliards de neurones
  - 60 milliards de connexions (synapses)
  - Un synapse peut être inhibant ou excitant.
- RNA :
  - Un nombre fini de processeurs élémentaires (neurones).
  - Liens pondérés passant un signal d'un neurone vers d'autres.
  - Plusieurs signaux d'entrée par neurone



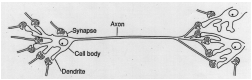
Cerveau	RNA
cellule (soma)	neurone
dendrites	entrées
synapses	poids
axon	sortie



12

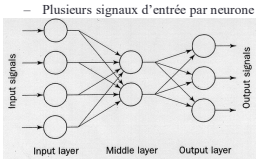
Réseau de neurones artificiel (RNA)
un modèle de calcul inspiré du cerveau humain

- Cerveau humain :
  - 10 milliards de neurones
  - 60 milliards de connexions (synapses)
  - Un synapse peut être inhibant ou excitant.



Cerveau	RNA
cellule (soma)	neurone
dendrites	entrées
synapses	poids
axon	sortie

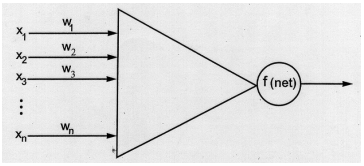
- RNA :
  - Un nombre fini de processeurs élémentaires (neurones).
  - Liens pondérés passant un signal d'un neurone vers d'autres.
  - Plusieurs signaux d'entrée par neurone



Une boite noire qui transforme des nombres en d'autres nombres en « regardant » une base d'apprentissage. Après apprentisage on veut f(input) = ouput

13

Introduction générale aux réseaux de neurones



[McCulloch-Pitts, 1943]

Un type de neurone simple : nD ➔ 1D avec f=sign

- net = Σ $w_i x_i$
- f(net)= +1 si net ≥ 0, -1 sinon.
- C.à-d. ici : 1 neurone = sign(Σ $w_i x_i$)

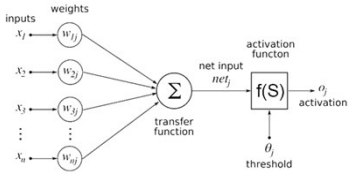
x = données d'entrée, w=les poids
f = Fonction d'activation du neurone

14

Introduction générale aux réseaux de neurones

Apprentissage consiste à trouver les poids w_{ij} par optimisation

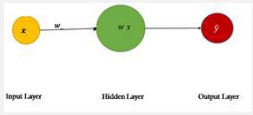
- Base de connaissance, un jeu d'apprentissage
- une série de couple (entrée, sortie) donc de
  Entrée=(x_0, ..., x_i, …x_j), sortie=(y_0, ..., y_i, …y_j)
- Intuition : initialiser w_{ij} au hasard, puis descente de gradient (~)



15

Un exemple très simple

Input	Desired output
0	0
1	2
2	4
3	6
4	8



- 1 neurone ultra simple
  - sans activation
  - ni bias
  - input/output à 1D

w . input = ouput

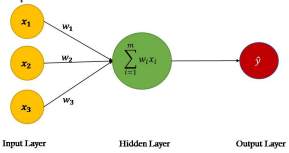
- Optimisation = entrainement = trouver w à partir de la base d'apprentissage (le tableau en haut à gauche)

https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

16

Introduction générale aux réseaux de neurones

- Souvent un neurone à plusieurs entrées et plusieurs sorties
  - En générale problème non linéaire
    ➔ fonction d'activation non linéaire
  - Dimension élevée
  - Nombreux paramètres w



➔ Mais restons dans notre cas très simple

17

Un exemple très simple

- Initialisation au hasard de w à 3

w . input = ouput

Input	Actual output of model 1 (y= 3.x)
0	0
1	3
2	6
3	9
4	12

18

Un exemple très simple

- Initialisation au hasard de w à 3
  w . input = ouput
- Fonction d'erreur : loss

Input	Actual output	Desired output
0	0	0
1	3	2
2	6	4
3	9	6
4	12	8

19

Un exemple très simple

- Initialisation au hasard de w à 3
  w . input = ouput
- Fonction d'erreur : loss

Input	actual	Desired	Absolute Error	Square Error
0	0	0	0	0
1	3	2	1	1
2	6	4	2	4
3	9	6	3	9
4	12	8	4	16
Total:	-	-	**10**	**30**

20

Un exemple très simple

- Initialisation au hasard de w à 3
  w . input = ouput
- Différentiation : dloss/dw = (loss(w) – loss(w+delta))/delta
  - Faire varier un peu w => w=3.0001
  - Calcul du gradient (dérivée partielle en cas de dim > 1)

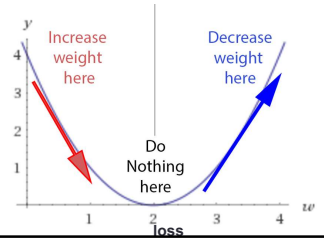
Input	Output	W=3	sq.e(3)	W=3.0001	sq.e
0	0	0	0	0	0
1	2	3	1	3.0001	1.0002
2	4	6	4	6.0002	4.0008
3	6	9	9	9.0003	9.0018
4	8	12	16	12.0004	16.0032
Total:	-	-	**30**	-	**30.006**

➔Descente de gradient
dloss/dw = (30.006-30)/0.0001 > 0 ➔ Augmenter w augmente l'erreur
➔Avancer dans le sens opposé au gradient w_new = w – lambda.gradient

21

Un exemple très simple

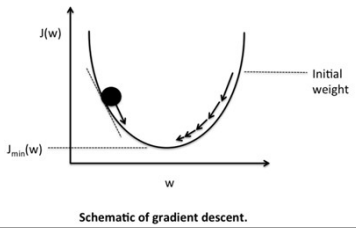
- Initialisation au hasard de w à 3
  w . input = ouput
- Différentiation : df/dw = (f(w) – f(w+delta))/delta
  - Faire varier un peu w => w=3.0001



22

Un exemple très simple

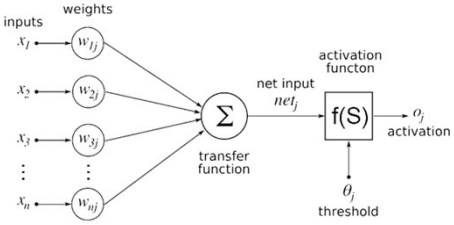
- Initialisation au hasard de w à 3
  w . input = ouput
- Différentiation : df/dw = (f(w) – f(w+delta))/delta
  - Faire varier un peu w => w=3.0001



Schematic of gradient descent.

23

Introduction générale aux réseaux de neurones



Apprentissage consiste à trouver les poids w_{ij} par optimisation

La fonction d'activation (non linéaire) permet d'approximer un problème/une fonction générale non linéaire

24

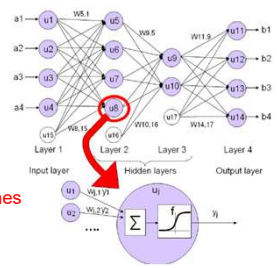
Introduction générale aux réseaux de neurones

Pour chaque neurone
- n Dimensions en entrée ➔ m Dimensions en sortie
- Les poids w_{ij} forment une matrice A

- sortie = A.entrée + B
- $y_n = A \cdot x_m + B$

- La fonction d'erreur « loss » est calculée de manière globale (à la sortie output)

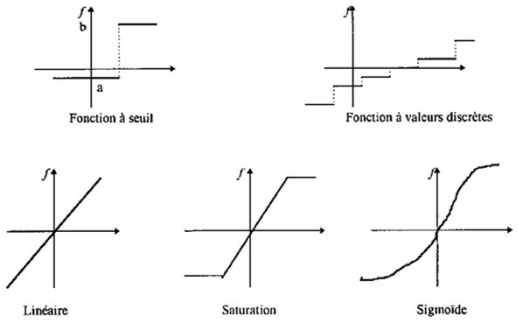
- Optimisation de tous les neurones en même temps



25

Introduction générale aux réseaux de neurones

Les fonctions d'activation f



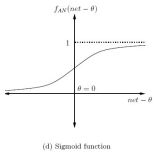
26

Introduction générale aux réseaux de neurones

La fonction sigmoïde

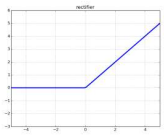
$$f_{AN}(\vec{net}) \in (0,1)$$

$$f_{AN}(net-\theta) = \frac{1}{1+e^{-\lambda(net-\theta)}}$$



RELU (Rectified Linear)

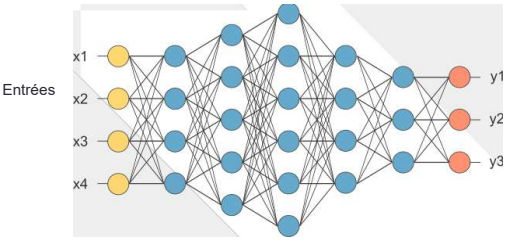
if x<0 then return 0
else return x



27

Réseau de neurones profonds

- Chaque cercle est un neurone f(A.X+B)
  - En bleu les couches cachées



28

La fonction d'erreur (LOSS)

- Entrainement ou apprentissage
  = une optimisation pour trouver les bons poids W qui maximisent les résultats sur une base de connaissance (input X, ouput Y)
- La fonction d'erreur que l'on cherche à minimiser
  - Plusieurs type possible suivant le problème

$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

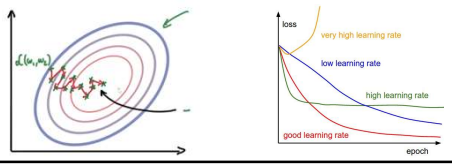
$$L2LossFunction = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

  - Généralement L2-Loss est préférée, mais si vous avez beaucoup d' *outliers* (valeurs aberrantes) dans vos données la L1 peut être meilleurs
  - La fonction d'erreur peut comporter d'autres termes (somme)

29

Réseau de neurones : entrainement

- Stochastic Gradient Descent (SGD)
  - Stochastic car la descente de gradient est calculée sur un sous-échantillonnage des données (epoch)
  - Learning rate = progression ou pas d'apprentissage
  - Moment = comme stochastic le changement de direction se fait avec une inertie
- Besoin des GPU pour la puissance de calcul
  - Par exemple, ImageNET 15 millions d'images avec 20 000 labels

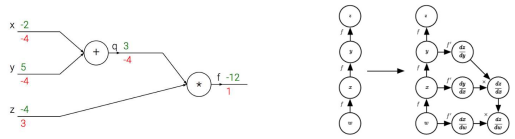


30

Entrainement : backpropagation

- Dans un réseau le calcul « classique » du gradient peut être long, très long à calculer ! Avec N paramètres à optimiser, descente demande N passes.
- Back propagation permet de faire le calcul avec une passe en remontant : en partant de l'erreur et en remontant
  - Au lieu de calculer l'influence que va avoir une variation de w sur l'erreur, calcul de l'influence que va avoir la variation de l'erreur sur les poids
  - Si la fonction inverse de chaque nœud est différentiable

➔voir le cours de Mathieu Lefort (Master IA) …



31

Réseau de neurones : du code

- Plateformes en python
  - TensorFlow (Google)
  - Theano
  - PyTorch (Yann Lecun, maintenant Facebook)
  - CNTK Microsoft Cognitive Toolkit
  - …
- Accélération GPU
- Des exemples de code
  - Keras = une sur-couche facilitant le codage des réseaux
    - Cible : TensorFlow, Theano, CNTK
  - PyTorch
  - TensorFlow 2.0 comportera des fonctions de haut niveau comme Keras/PyTorch

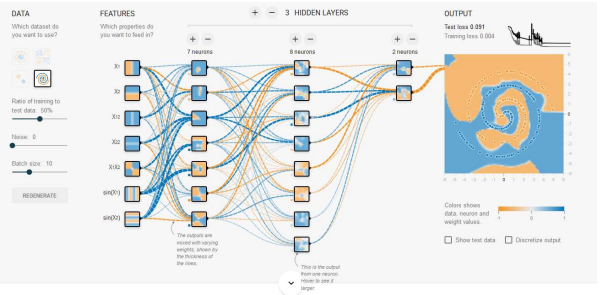
➔ Une communauté immense, des exemples de code (github, etc.) par millier, reproductibilité du code, etc.

32

Réseau de neurones

- Exemple de classification d'un nuage de points

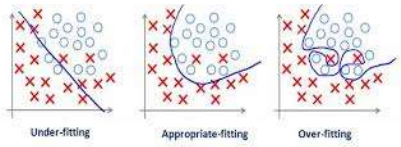
http://playground.tensorflow.org



33

Surapprentissage / Overfitting

- Surapprentissage / Overfitting : c'est quoi ?
  - Quand un modèle apprend trop de détail/bruit sur les données avec pour conséquence d'être mauvais sur la généralisation (ie. Quand il verra des nouvelles données jamais vu avant)
  - Comme un étudiant qui « apprend par cœur » sans « comprendre »



34

Régularisation

Les techniques de **Regularisation** sont les techniques utilisées pour résoudre le surapprentissage (overfitting) en apprentissage machine

Par exemple

- L1 Regularization or Lasso Regularization

$$Min(\sum_{i=1}^{n}(y_i - w_i x_i)^2 + p\sum_{i=1}^{n}|w_i|)$$

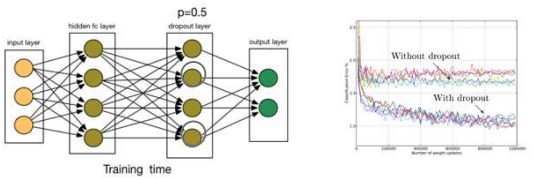
- L2 Regularization or Ridge Regularization

$$Min(\sum_{i=1}^{n}(y_i - w_i x_i)^2 + p\sum_{i=1}^{n}(w_i)^2)$$

35

Drop Out

- Dropout aide pour éviter le surapprentissage (overfitting)
  - force le réseau à apprendre de manière plus robuste
  - technique de régularisation
- Dropout annule temporairement aléatoirement certain neurone avec une proba p

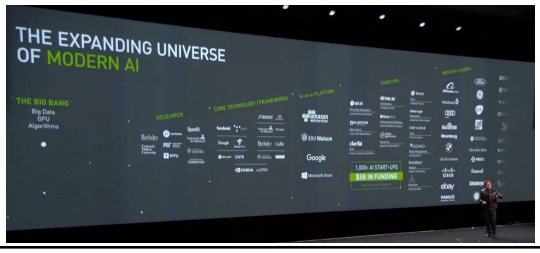


Dropout: A Simple Way to Prevent Neural Networks from Overfitting
Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov
JMLR 2014

36

Explosion du deep learning

- Des données
- Du GPU
- Un algo de back-propagation rapide et facilement codable sur GPU
- Des algo, des algo, des algo …



37

DEEP LEARNING POUR LES BASIQUES DE L'IMAGES ET DE LA VISION

ImageNet Challenge

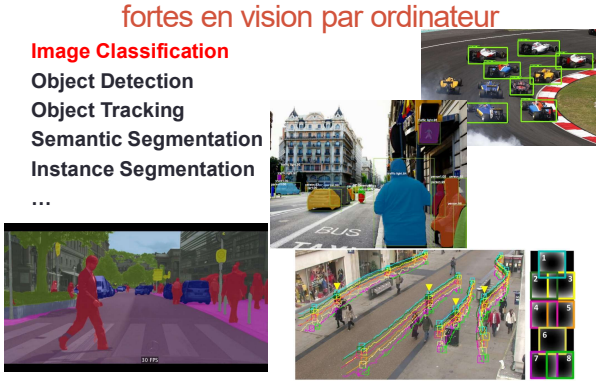
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

38

Deep Learning a permis des avancées fortes en vision par ordinateur

Image Classification
Object Detection
Object Tracking
Semantic Segmentation
Instance Segmentation
…

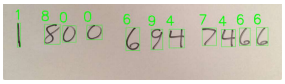


39

Deep Learning et Reconnaissance

Reconnaissance

- d'une famille d'objets : « c'est un chat »
- d'un objet précis : « c'est Paul », « c'est un 8 »
- d'une expression/style : « le visage sourit »
- …



40

Vision de haut niveau : reconnaissance

Qu'est-ce que vous voyez dans cette image ?

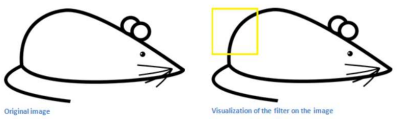
Tâche extrêmement difficile :
le tigre doit être reconnu sous tous les angles, parfois caché, avec des éclairages différents sur chaque photo.

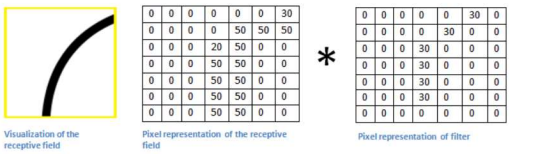
➔ Test de Turing sur l'« intelligence artificielle »

41

Filtres

- Détecteur de courbes



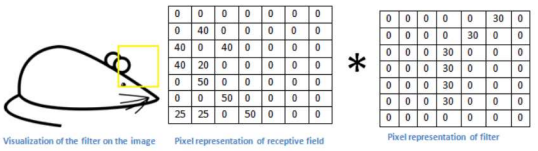


Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

42

Filtre : détecteur de courbes

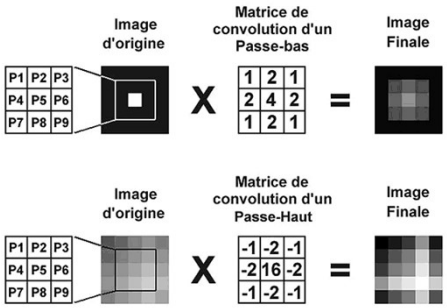
- Le même filtre ailleurs



- On obtient donc une carte d'activation de ce filtre
  - Feature map

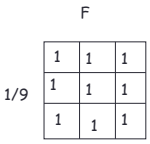
43

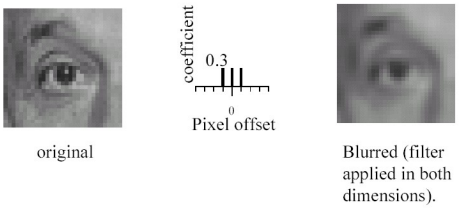
Traitement d'images : filtres



44

Filtre Gaussien : Blurring



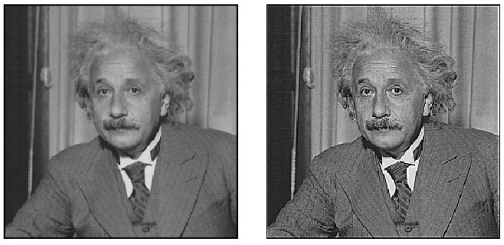


original

Blurred (filter applied in both dimensions).

45

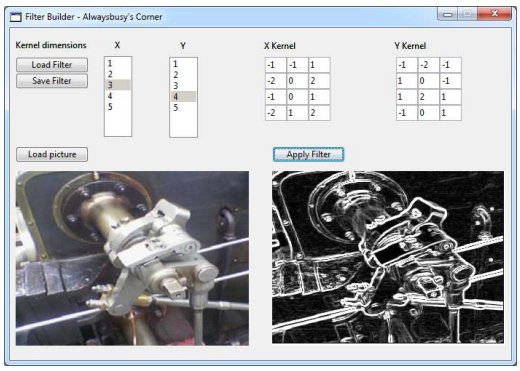
Sharpening



before after

46

Filtre : détecteur d'arêtes



47

Les Difficultés

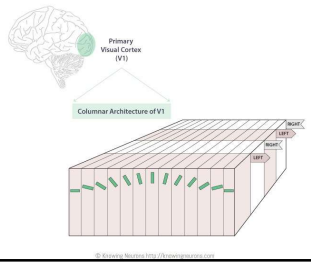
Quel enchainement/combinaison de filtres permettront de bien classifier une image ?

- Avant (le deep learning)
  - un humain (ingénieur/chercheur) proposait des filtres selon son savoir-faire pour produire des descripteurs (quelques centaines de valeurs)
  - Puis classification (SVM, Random Forest, etc.)
- Maintenant, réseau de convolution ConvNeuralNET (CNN)
  - Optimise des poids dans plusieurs filtres pour produire les descripteurs (*feature maps*)
  - Puis les dernières couches complétement connectées font la classification

48

Expérience de Huble

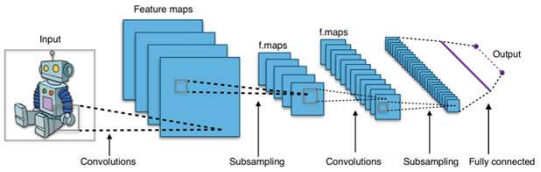
- Huble and Wiesel en1962 montre que dans le cortex visuel les neurones sont organisés pour répondre à des paternes précis : lignes avec différentes inclinaisons, etc.



49

Réseau de convolution ConvNET

- Reconnaitre un objet avec un réseau de convolution

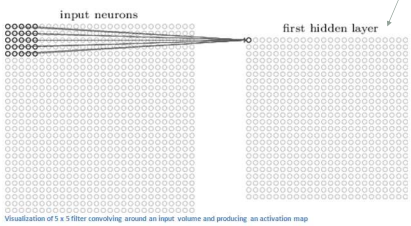


50

Une convolution

Activation map
Feature map
Image de caractéristiques

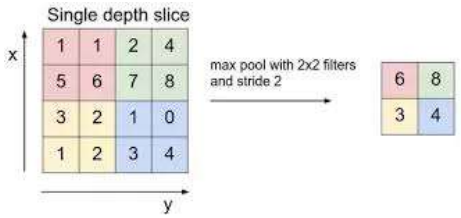
- Input : 32x32x1 ➔ Conv(5,5) ➔ 28x28x1
- Input : 32x32x3 ➔ Conv(5,5,3) ➔ 28x28x1
  - La convolution se fait sur les 3 channels



51

Max Pooling

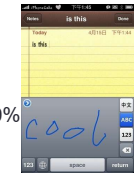
- Reduction de dimensions

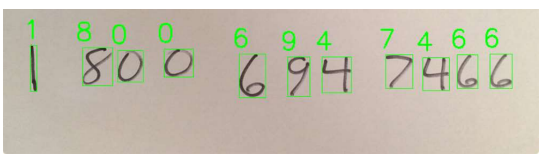


52

Reconnaissance d'écriture

- Reconnaissance d'écriture
  - La Poste : codes postaux sur enveloppe
  - Puis écriture sur tablette
- Avec un ConvNET, taux de bonne reco > 99%

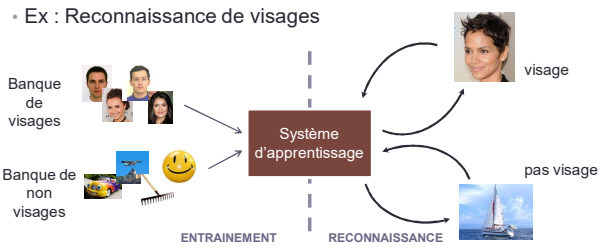




53

Reconnaissance de visages

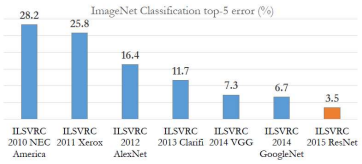
- L'apprentissage automatique
  - A partir d'une banque d'exemple, l'ordinateur apprend à classer différents éléments.
- Ex : Reconnaissance de visages



54

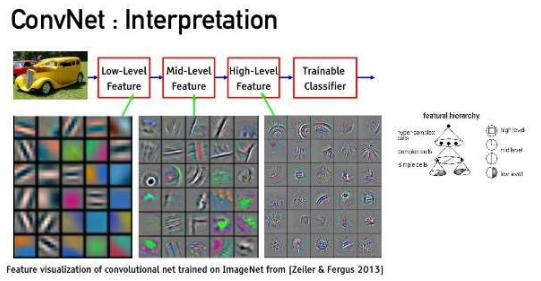
IMAGENET=corpus d'images

- Classification
  - Historique vision humaine : reperer un predateur ou un membre de sa famille rapidement
  - Concours IMAGENET ➔ mettre un label sur une image
    - 14 millions d'images avec 20000 labels



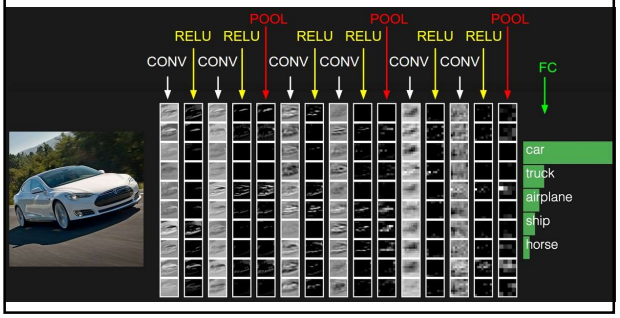
55

Réseau de convolution ConvNET



56

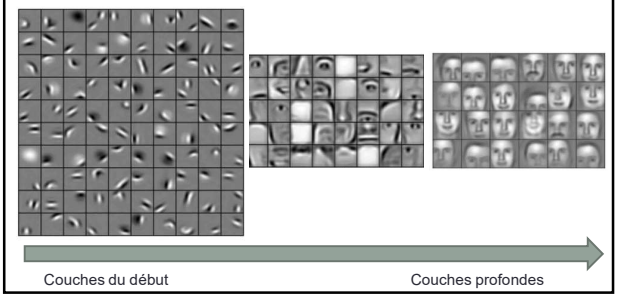
Réseau de convolution ConvNET



57

ConvNET : visage

- Exemple de features (filtres) produit par le réseau dans les couches cachées



58

VGG

K. Simonyan, A. Zisserman
Very Deep Convolutional Networks for Large-Scale Image Recognition
arXiv technical report, 2014

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

59

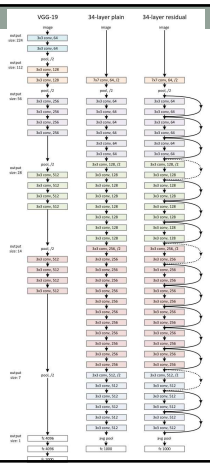
VGG19 : voir TP pour afficher les couches

```
0   Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
1   ReLU(inplace)
2   Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
3   ReLU(inplace)
4   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
5   Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
6   ReLU(inplace)
7   Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
8   ReLU(inplace)
9   MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
10  Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
11  ReLU(inplace)
12  Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
13  ReLU(inplace)
14  Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
15  ReLU(inplace)
16  Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
17  ReLU(inplace)
18  MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
19  Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
20  ReLU(inplace)
21  Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
22  ReLU(inplace)
```

60

ResNET-50

Deep Residual Learning for Image Recognition
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
https://arxiv.org/abs/1512.03385



61

Réseaux de reconnaissance

Souvent utilisés en pré-traitement comme « traitement d'images »

- VGG
  - VGG16, VGG19
  - Série de Conv, Max-pooling, and activation, puis fully-connected (FC)
- ResNet50
  - Grande profondeur
- Inception v3
  - Large
- Xception
  - extreme inception
- MobileNet
  - Optimisé pour mobile

- Dans KERAS / PyTorch ces modèles sont pré-entrainés (ouf)

62

Les données

	VGGNet	DeepVideo	GNMT
Used For	Identifying Image Category	Identifying Video Category	Translation
Input	Image	Video	English Text
Output	1000 Categories	47 Categories	French Text
Parameters	140M	~100M	380M
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

Number of parameters (in millions), for popular neural networks.

63

Problème : manque de données pour DL

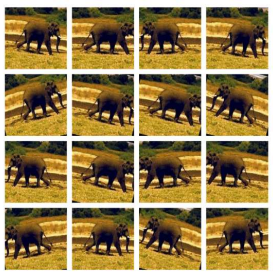
- Taille de certaines base de données
  - MNIST : nombres manuscrits, 70000 images
  - EMNIST : lettres manuscrites
    - Equilibré 134000 images
    - Sinon 814000 images
  - ImageNET : 14 millions d'images avec 20000 labels
  - CelebFaces (CelebA) : 202,599 images de visages de 10,177 célébrités
- Pour de nombreux autres problèmes
  - Cohn Kanade : 486 vidéo de 97 personnes exprimant une expression
  - …
- Branches du machine learning qui cherchent à fonctionner avec moins de données
  - DL plus efficaces : GAN, etc.
  - One-shot learning / Few-shot learning

64

Augmentation de données

La base

- Flip
- Rotation
- Scale
- Crop
- Translation
- Gaussien Noise



65

Augmentation de données

Un peu plus loin

- GAN pour transformer des images (palette de couleur, style, etc.)

- Données issus d'images de synthèses

Do We Really Need to Collect Millions of Faces for Effective Face Recognition?
https://arxiv.org/pdf/1603.07057.pdf

66

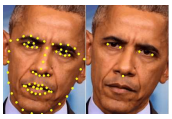
Augmentation de données : aider le réseau avec des données annexes

Par exemple : corpus d'images/vidéo de visages

- DL sur uniquement les images, possible mais …
- Possibilité d'en extraire des points du visage par détection : coins de la bouche, nez, yeux

➔DL sur les points caractéristiques + images

Bien plus efficaces !!!



reconnaissance → Expressions
Nom de la personne
Etc.

67

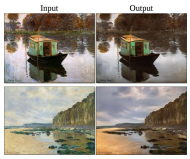
TRANSFERT DE STYLE ENTRE IMAGES

Image Style Transfer Using Convolutional Neural Networks
Gatys etal. CVPR 2016

68

Transfert de style entre images

- Toutes les qualités d'un bon TP
  - Utilisation des frameworks de DL
  - Optimisation
  - Utilisation des réseaux, mais sans avoir besoin de les entrainer (temps de calcul raisonnable pour un TP)



69

Transfert de style entre images

- Différentier
  - Contenu de l'image : objets et leurs places/positions/orientations
  - Style : couleur et textures
- VGG19 pour extraire les caractéristiques
  - Chaque couche de convolutions va produire une carte de caractéristiques (features)
  - Optimisation avec deux termes : Coût_Contenu + Coût_Style

Caractéristiques à différentes échelles

70

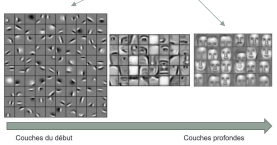
Transfert de style entre images

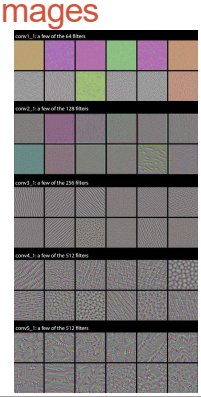
VGG19 pour extraire les caractéristiques

- Chaque couche de convolutions va produire un vecteur de caractéristiques de dimension

  Batch_size x Nfeatures x Height x Weight

  - Certaines couches codent plutôt le contenu (vers le fond du réseau), d'autres plutôt le style (vers le début)



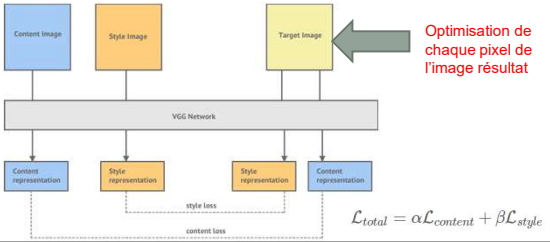


Features visualization of VGG network

71

Transfert de style : optimisation

- Il n'y a pas d'optimisation des neurones d'un réseau.
- Le réseau (VGG) est utilisé pour produire les descripteurs (features)
- Le framework de Deeplearning (pytorch) est utilisé pour optimiser les pixels de l'image



Optimisation de chaque pixel de l'image résultat

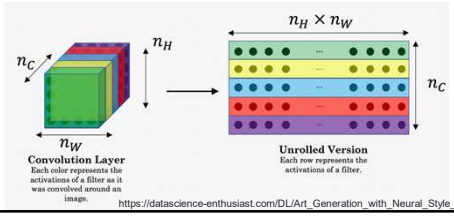
$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$$

72

Transfert de style entre images

VGG19 pour extraire les caractéristiques

- Chaque couche de convolutions va produire un vecteur de caractéristiques de dimension
  Nfeatures(N_c sur la figure) x Height x Weight
  ➔ à aplatir en Nfeatures x Npixels avec Npixels=Height x Weight



https://datascience-enthusiast.com/DL/Art_Generation_with_Neural_Style_Transfer_v2.html

73

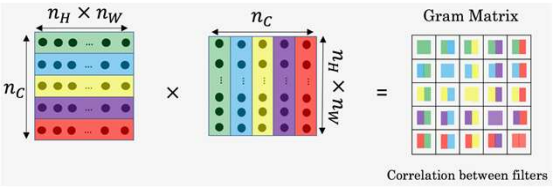
Transfert de style : matrice de Gram

- **Matrice de Gram : M x M^t**
  - Produit scalaire entre toutes les caractéristiques
  - Corrélation entre les features
- Avec une matrice F de caractéristique, une entrée de la matrice de Gram G est le produit scalaire entre 2 caractéristiques

$$G_{ij} = \sum_k F_{ik} F_{jk}$$

74

Transfert de style : matrice de Gram



Si une entrée dans la matrice de Gram a une valeur proche de 0, cela signifie que les 2 *features* ne s'active pas simultanément (non corrélation). Et vice versa, si une entrée a une grande valeur, cela signifie que les 2 *features* s'activent simultanément (corrélation).

Nous allons chercher à créer une image qui réplique un même schémas d'activations des *features* de style.

https://datascience-enthusiast.com/DL/Art_Generation_with_Neural_Style_Transfer_v2.html

75

Transfert de style : coût de contenu

- Si on peut construire une image qui a une carte de caractéristiques équivalentes pour un niveau de convolution donné à une autre image. Ces deux images auront le même contenu (surtout pour les couches profondes) — mais pas nécessairement la même texture ou style.
- Soit une couche de convolution **l** dans VGG, la fonction de coût de contenu est défini comme la moyenne au carré de l'erreur entre la carte de *features* **F** de l'image de contenu **C** et la carte de *features* de l'image générée **Y**.

$$\mathcal{L}_{content} = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

76

Transfert de style : coût de style

- Le calcul de coût du style est similaire au calcul de coût de contenu, mais on le calcule à partir de la matrice de Gram à la place de directement utiliser les features

$$\mathcal{L}_{style} = \frac{1}{2} \sum_{l=0}^{L} (G_{ij}^l - A_{ij}^l)^2$$

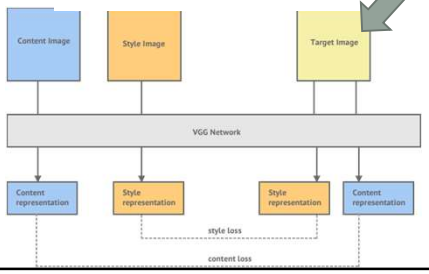
77

Transfert de style : optimisation

- La fonction de coût à optimiser

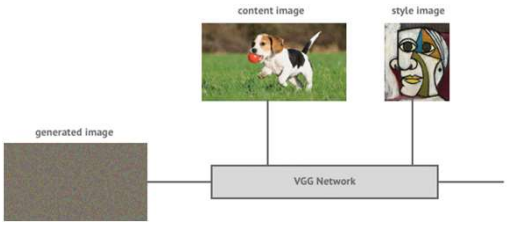
$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

Optimisation de chaque Pixel de l'image résultat



78

Transfert de style



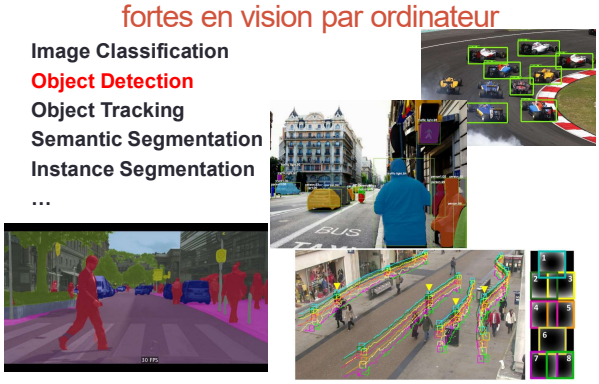
79

➔TP

80

Deep Learning a permis des avancées fortes en vision par ordinateur

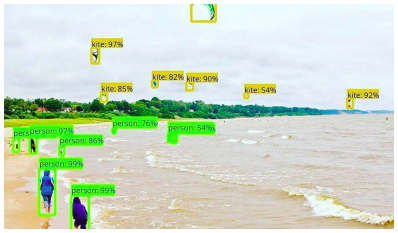
Image Classification
Object Detection
Object Tracking
Semantic Segmentation
Instance Segmentation
…



81

Détection d'objets

- Plus dur que la reconnaissance car les objets peuvent varier
  - Taille, rotation, etc. comme en reconnaissance
  - Position dans l'image : le nombre de rectangle à tester peu être grand



82

Détection d'objets

Region-Based Convolutional Neural Network (R-CNN)

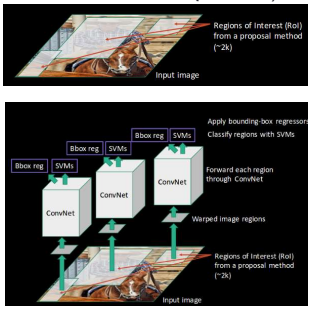
- Segmentation (quelconque)
- Fusion de région : similarité de couleur, texture, forme, …
- Chaque région produit une région d'intérêt : ROI

83

Détection d'objets

Region-Based Convolutional Neural Network (R-CNN)

- ROI
  - CNN➔descripteurs
  - SVM classifie
  - BoundingBox regression pour ajuster la Bbox



Problème avec R-CNN
- 2000 régions
- 2000xNB descripteurs

➔1 minute par image

84

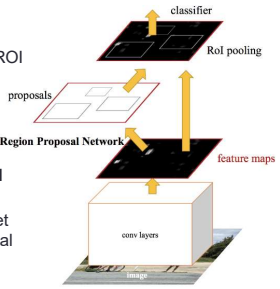
Détection d'objets

Fast Region-Based Convolutional Neural Network

- Replace la phase de segmentation/ROI par un CNN
- 2 secondes par images

Faster Region-Based Convolutional Neural Network

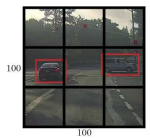
- Replace la phase de segmentation et sélection de ROI par Region Proposal Network (RPN)
- Temps 0.2 seconde par image



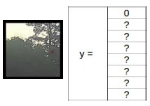
85

Détection d'objets

- R-CNN; Fast R-CNN; Faster R-CNN
- YOLO : rapide, 45 images/seconde
  - Par exemple, détection de 3 classes
    - Pc : objet présent dans la fenêtre
    - bx,by,bh,bw : bounding box
    - c1, c2, c3 : présences des 3 classes



y = pc, bx, by, bh, bw, c1, c2, c3



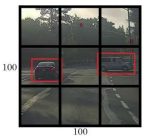
y = 0, ?, ?, ?, ?, ?, ?, ?

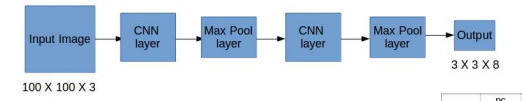
y = 1, bx, by, bh, bw, 0, 1, 0

86

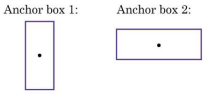
Détection d'objets

- R-CNN; Fast R-CNN; Faster R-CNN
- YOLO : rapide, 45 images/seconde





?



y = pc, bx, by, bh, bw, c1, c2, c3, pc, bx, by, bh, bw, c1, c2, c3

87

Segmentation d'images

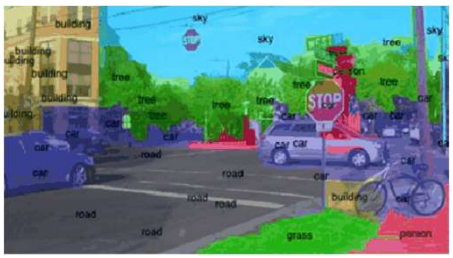
- Le zèbre ☺
  - La nature l'a fait évoluer pour se dissimuler
  - Le plus dur pour les algo de vision

88

Sémantique et image

- Entraine un réseau à regrouper des régions pour segmenter et mettre des labels sur une image

+ VIDEO



89

Deep Learning a permis des avancées fortes en vision par ordinateur

- Souvent des idées simples mais malignes marchent le mieux
  - Viola–Jones object detection 2004
  - …
  - …
  - Yolo 2018
  - Dans les basiques reste à voir : Object Tracking, Semantic Segmentation, Instance Segmentation
- Ces outils de visions peuvent être maintenant
  - Utilisées dans des applications de tous les jours (mobile, jeux vidéo, …)
  - Utilsées en recherche pour résoudre des problèmes d'une autre nature : Reconstruction 3D, MoCap, Extraction de textures,illumination, etc.

90

POUR ALLER PLUS LOIN AVEC LES RÉSEAUX ET L'IMAGES

- Différents types de réseaux
  - RNN, LSTM,
- Pour de l'apprentissage semi ou non supervisée
  - Autoencoder
  - FaceNet (clustering)
  - Apprentissage par transfert
  - GAN
- …

91

Pour aller plus loin avec les images

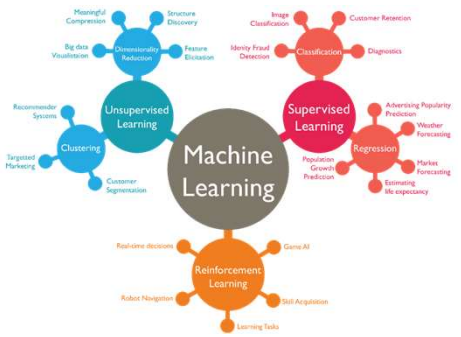
Les réseaux sont un outils pour de nombreux problèmes

On profite

- Framework
  - GPU, Optimiseur, etc.
  - Learning en python mais après utilisation en C++, C#, Java, etc.
  - Format standard de fichiers : Open Neural Network Exchange
- Communauté grande
  - Nombreux tutoriaux et explications
    - medium.com
    - letslearnai.com
    - Etc.
- Recherche reproductible (Github)

92

ML : Supervisé / Non supervisé



93

Différents problèmes / différents réseaux

Images : classification et « traitement d'images »
- CNN à 2 niveaux de convolution (cf. TP)
- CNN à 19..50 niveaux ➔ VGG, ResNET, AlexNet, GoogleNET, etc.

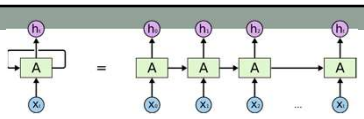
Données temporelles
- RNN
- LSTM

Semi supervisé : par exemple des données mais pas de labels
- Auto-encoder
- Construction de cluster : par exemple FaceNET

Divers problèmes utilisant fortement les réseaux
- Transfert de style sur des images
- Super résolution
- Segmentation
- Générer des données : GAN
- Apprentissage par renforcement : Deep Q-Learning, etc.

94

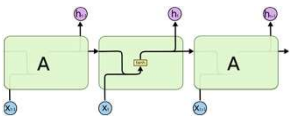
RNN



- Pour des données temporelles
  - Prédire le prochain mot
  - Composer de la musique
  - Reconnaitre le langage parlé
  - Détection d'erreur dans une série d'évenements
  - Prédiction de la bourses

➔Recurrent Neural Network, LSTM, Gated Recurrent Unit, …

RNN simple



95

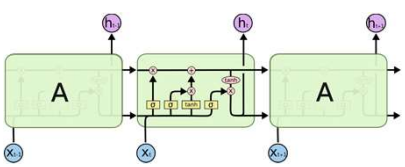
LSTM

- Pour des données temporelles

➔**L**ong **S**hort-**T**erm **M**emory

Par exemple
- 6 -> 7 -> 8 -> ? On voudrait 9
- 2 -> 4 -> 8 -> ? On voudrait 16
- Se baser sur 8 ne suffit pas

- A une mémoire courte et long terme
- **Apprend quand se souvenir et quand oublier**

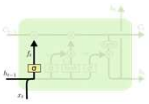


96

LSTM

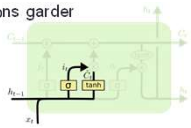
Les couches

- Oublier ou garder: 0..1



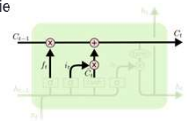
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- Quelles informations garder



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

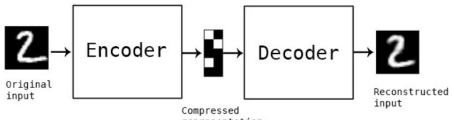
- Produire la sortie

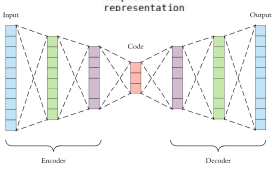


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

97

Auto Encoder (AE) : principe





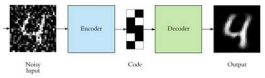
$$Obj = L(x, \hat{x})$$

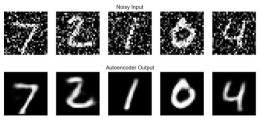
- Il existe de nombreux types d'autoencoder

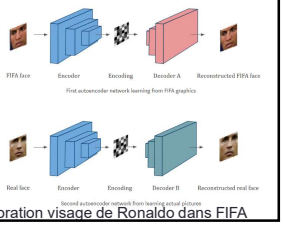
98

Auto Encoder (AE) : principe

- A quoi ca sert ?
  - Débruitage
  - Compression
  - Code compact
    - Possibilité d'appliqué des traitements sur le « code »
    - Un peu le même esprit que la PCA
  - Prémisse du Génératif





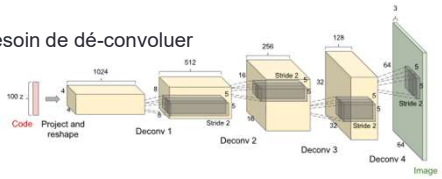


Amélioration visage de Ronaldo dans FIFA

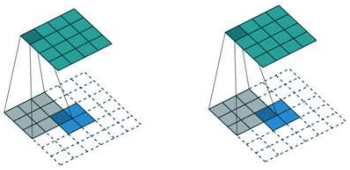
99

Decode

Si image : besoin de dé-convoluer

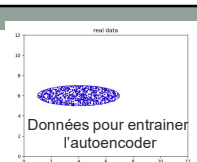


Devrait s'appeler plutôt transposed convolutional layer



100

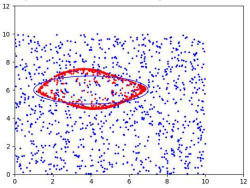
Auto-encoder



Données pour entrainer l'autoencoder

- Démo nuage de points en TP
  - points bleus donnés à l'autoencoder qui donne les points rouges

Après plusieurs passes l'auto-encoder ramène les points bleus qui ne sont pas dans sa représentation apprise à l'intérieur de s

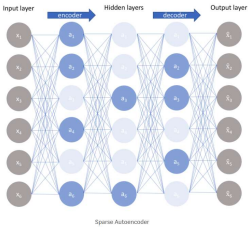


101

Sparse Auto Encoder

Dans la fonction de coût, favorise « sparsity » (clairsemé)
→Améliore les performances

- Intuition : avoir essentiellement des neurones utiles (avec un poids fort) pour avoir une représentation « intelligente »



Distance entre input et output

$$Obj = L(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}|$$

Favorise la sparsity

102

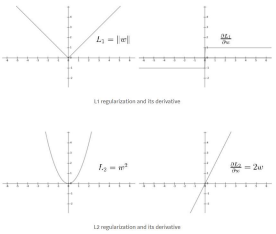
Sparse Auto Encoder : norme L1 / L2

- Utiliser la norme L1 dans la fonction de coût

$$Obj = L(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}|$$

$$L_1 = \|w\|, L_2 = w^2$$

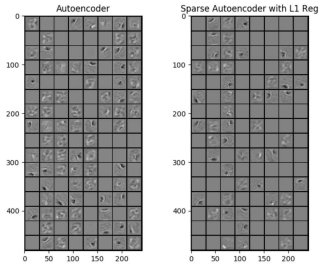
Intuition : gradient vaut -1 ou +1 donc un pas plus grand à chaque itération que avec la norme L2 qui va faire des petits pas



103

Sparse Auto Encoder : norme L1

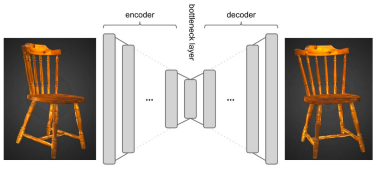
- Moins de neurones activés → les neurones utiles plus efficaces → représentation plus « inteligente »



104

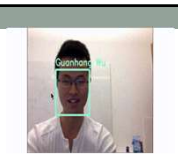
Auto-encoder …

- De nombreux auto encodeur existent
  - Variational AE, …
  - Inclus dans un GAN
  - Possibilité de les entrainer sur autres choses que des images
    - Maillage 3D
    - Animation …
  - Encore un fois un problème vaste … mais un outils puissant



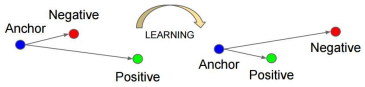
105

FaceNet



Reconnaitre le nom de la personne d'une image de son visage

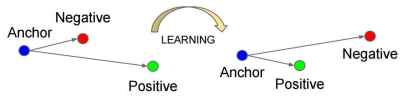
- Plus efficacement que CNN de base
- Apprend une représentation des données pour rapprocher les éléments ayant un rapport entre eux et éloigner les autres ➔ **construire des clusters**



FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR 2015
http://cmusatyalab.github.io/openface/

106

FaceNet



- Utilise un triplet
  - une instance "anchor" x : le visage d'une personne A
  - une instance positive x+ : le visage de la même personne A
  - une instance negative x- : le visage d'une autre personne B
- $f(x)$ la representation de x, la function de coût

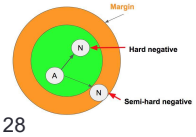
$$\max(0, \quad \|f(x) - f(x+)\|^2 - \|f(x) - f(x-)\|^2 + \alpha)$$

- Ignore le triplet quand x+ est déjà plus proche que x-
- Apprendre $f(x)$ en utilisant la backpropagation

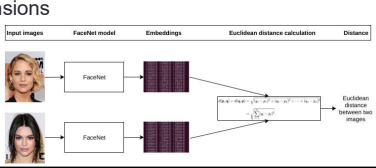
107

FaceNet

- Le nombre de triplet est gigantesque
  - 10^6 de visage => 10^{18}triplets
  - → progression
  - Choisir des triplets semi difficiles
  - Choisir des triplets difficiles

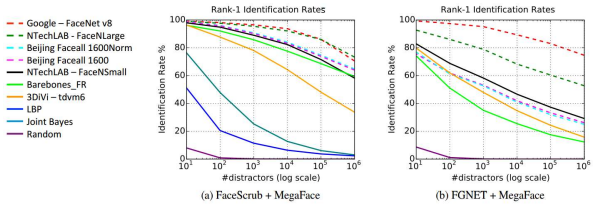


- Produit une description d'un visage en 128 dimensions



108

MegaFace Benchmark



The MegaFace Benchmark: 1 Million Faces for Recognition at Scale, CVPR 2016

109

FaceNet : identifier une personne

- FaceNET les erreurs par paires➔
  - Parmi ces erreurs 13 ont été mal classées dans la base de données
- Un exemple de cluster pour une personne

Figure 7. **Face Clustering.** Shown is an exemplar cluster for one user. All these images in the users personal photo collection were clustered together.

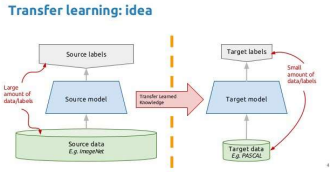
Figure 6. **LFW errors.** This shows all pairs of images that were incorrectly classified on LFW. Only eight of the 13 false rejects shown here are actual errors the other five are mislabeled in LFW.

110

Transfert d'apprentissage avec CNN

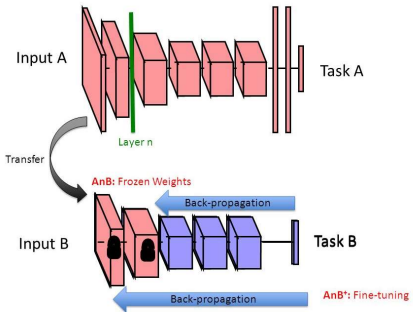
Approche simple pour se spécialiser

- Modèle pré-entrainé sur une grande base de données assez générale
- Gèle certain paramètres (weights) : couches basses de convolution
- Ajoute des couches « classifier » avec ses paramètres à entrainer sur les données spécifiques
- Entraine ce réseau sur les données spécifiques
- Eventuellement dégèle tous les paramètres à la fin



111

Transfert d'apprentissage avec CNN



112

Transfert d'apprentissage avec CNN

- Un exemple avec PyTorch

```
from torchvision import models
model = model.vgg16(pretrained=True)
```

```
# Freeze model weights
for param in model.parameters():
    param.requires_grad = False
```

```
import torch.nn as nn

# Add on classifier
model.classifier[6] = nn.Sequential(
                      nn.Linear(n_inputs, 256),
                      nn.ReLU(),
                      nn.Dropout(0.4),
                      nn.Linear(256, n_classes),
                      nn.LogSoftmax(dim=1))
```

113

Transfert d'apprentissage avec CNN

Transfert learning

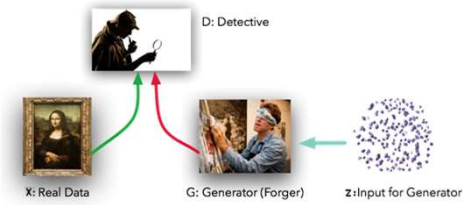
- Un domaine vaste
- Adapter un apprentissage est un des prochains verrous
- …

How transferable are features in deep neural networks?
Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson
NIPS2014

114

Réseau de neurones et génération : GAN

- GAN = Generative Adversarial Networks
  - Proposé en 2016, déjà plusieurs centaines de papiers/variantes
  - + Fonctionne avec une quantité de données moindre car en génère
  - - Instable



115

Réseau de neurones et génération : GAN

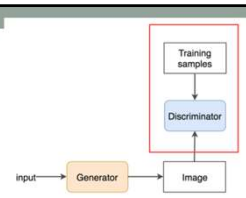
- Generative Adversarial Networks (2014),
- Goodfellow etal.
  - Cité 2000 fois en 3 ans
- Plusieurs centaines de variantes
- Les applications « cools » des GAN
  - Génération de visages
  - Personne avec poses différentes
  - Transfert de texture
  - Super résolution
  - Texte vers images
  - …

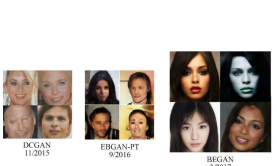
https://medium.com/@jonathan_hui/gan-some-cool-applications-of-gans-4c9ecca35900

116

GAN

- Problème compliqué
  - Nombreuse version avant d'obtenir de bons résultats
- Mais souvent impressionnant

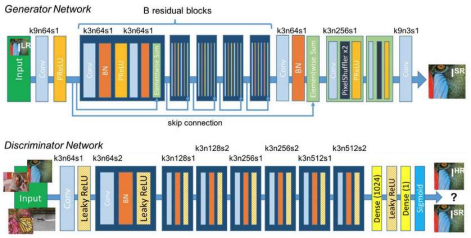




117

Un exemple de GAN : SRGAN

- Super résolution



118

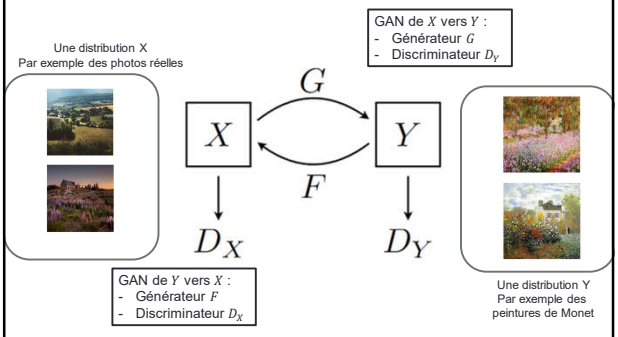
Un exemple de GAN : SRGAN

Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

SRGAN

119

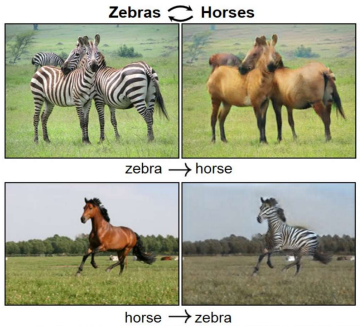
Transfert entre distributions : CycleGAN



120

Cycle GAN

- Distribution
  - Cheval
  - Zèbre



121

GAN : un article à lire

GAN Dissection: Visualizing and Understanding Generative Adversarial Networks

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum1, William T. Freeman, Antonio Torralba

https://gandissect.csail.mit.edu/ (ESSAYER LA DEMO)

122

GAN : les problèmes

Entrainer un GAN n'est pas simple

- **Mode collapse** : le générateur produit un nombre limité d'exemple
- **Diminished gradient** : le discriminateur deviant trop bon, trop rapidement et donc le generator n'apprend rien
- **Non-convergence** : les paramètres du modèle oscillent et ne converge jamais
- Plus généralement, un déséquilibre entre le generateur et le discriminateur provoque du sur apprentissage, hyper sensibilité au paramètres

123

GAN : modification d'un visage

- StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation (2017)

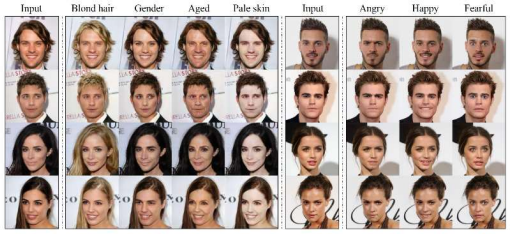


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

124

GAN : modification d'un visage

- StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation (2017)

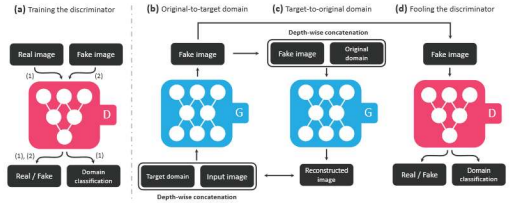


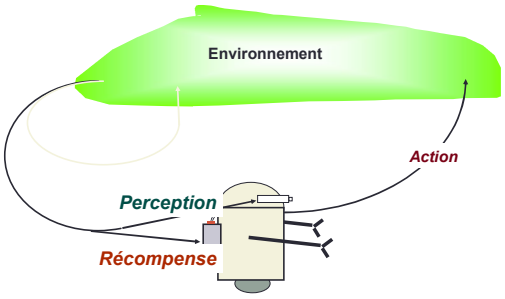
Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G. (a) D learns to distinguish between real and fake images and classify the real images to its corresponding domain. (b) G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. (c) G tries to reconstruct the original image from the fake image given the original domain label. (d) G tries to generate images indistinguishable from real images and classifiable as target domain by D.

125

APRENTISSAGE PAR RENFORCEMENT

126

Apprentissage par renforcement



Problème non uniquement lié à la production d'un geste

127

Apprentissage par renforcement

- *Temps* discret: t
- *États* : $s_t \in S$
- *Actions* : $a_t \in A(s_t)$
- *Récompenses* : $r_t \in R(s_t)$
- L'agent : $s_t \rightarrow a_t$
- L'environnement : $(s_t, a_t) \rightarrow s_{t+1}, r_{t+1}$
- *Politique* : $\pi_t : S \rightarrow A$
  - Avec $\pi_t(s,a)$ = Prob que $a_t = a$ si $s_t = s$
- Les transitions et récompenses ne dépendent que de l'état et de l'action précédents : processus Markovien

T, R

128

Apprentissage par renforcement

- *Politique* :
  ensemble d'associations *situation* → *action* (une application)
  Une simple table ... un algorithme de recherche intensive
  Eventuellement stochastique
- *Fonction de renforcement* :
  - Définit implicitement le but poursuivi
  - Une fonction : (*état*, *action*) → *récompense* $\in \Re$
- *Fonction d'évaluation* $V(s)$ ou $Q(s,a)$:
  - Récompense accumulée sur le long-terme
- *Modèle de l'environnement* :
  - Fonctions T et R : (*état(t)*, *action*) → (*état(t+1)*, *récompense*)

129

Apprentissage par renforcement

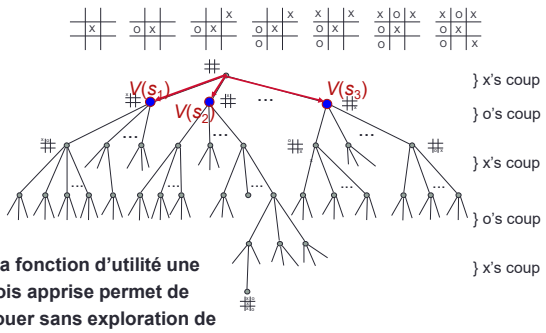
Principe :

- Choisir une action sans avoir besoin de faire une exploration (simulée) en avant

- Il faut donc disposer d'une fonction d'évaluation locale résumant une espérance de gain si l'on choisit cette action : *fonction d'utilité*
- Il faut apprendre cette fonction d'utilité : *apprentissage par renforcement*

130

Apprentissage par renforcement

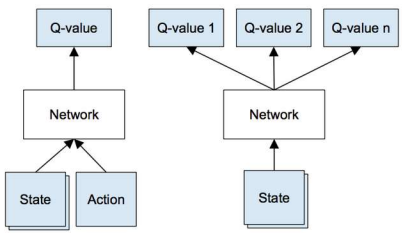


La fonction d'utilité une fois apprise permet de jouer sans exploration de l'arbre de jeu

131

Deep Q Learning

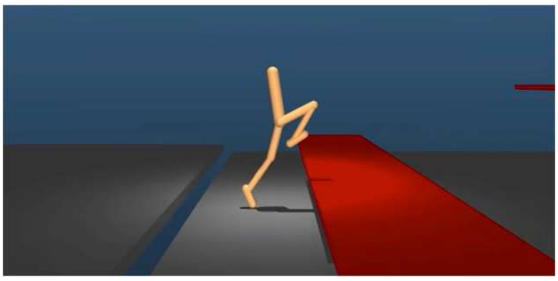
- Réseau de neurones bien adaptés au problème



132

Learning to move

+vidéos



133

… à suivre …

134