

Deep learning

Applications du deep learning et Deep reinforcement learning

Adrien CHAN-HON-TONG

ONERA/DTIS département traitement de l'information et système

Rappel

Ce qui est très important à retenir

- ▶ Qu'est ce que l'apprentissage par ordinateur ?
- ▶ Qu'est ce qu'un réseau de neurones ?

Rappel

Pour votre culture

- ▶ Comment on optimise les poids ?
- ▶ Qu'est ce qu'on peut faire avec ?

Aujourd'hui

Pour votre culture

Deep learning et IA

Rappel

classification supervisée de vecteurs

- ▶ base d'apprentissage
 $(x_1, y(x_1)), \dots, (x_N, y(x_N)) \in \mathbb{R}^D \times \{-1, 1\}$
- ▶ à partir de la base on crée un modèle f
- ▶ **on ne peut pas être sûr que $f = y$** (mais on peut être très bon et il n'y a pas d'autre façon d'attaquer le problème)
- ▶ on prend un base de test
 $(\chi_1, y(\chi_1)), \dots, (\chi_N, y(\chi_N)) \in \mathbb{R}^D \times \{-1, 1\}$
- ▶ on évalue le modèle : $\frac{1}{N} \sum_{n=1}^N |f(\chi_n) - y(\chi_n)|$

Rappel

un réseau de neurones

- ▶ 1 neurone : $\text{neurone}_{w,b}(x) = wx + b$
- ▶ activations $\max(0, \text{neurone}_{w,b}(x))$ (pas sur la dernière couche)
- ▶ couches puis réseau
- ▶ pour l'image, neurone convolutif : pareil mais local
- ▶ et pooling (qu'on peut voir comme une activation)

Rappel

comment on optimise

- ▶ La descente de gradient : $F(x - \lambda \nabla_x F) < F(x)$
- ▶ **Stochatique** pour ne pas à avoir à calculer F !
- ▶ Le gradient : la backpropagation

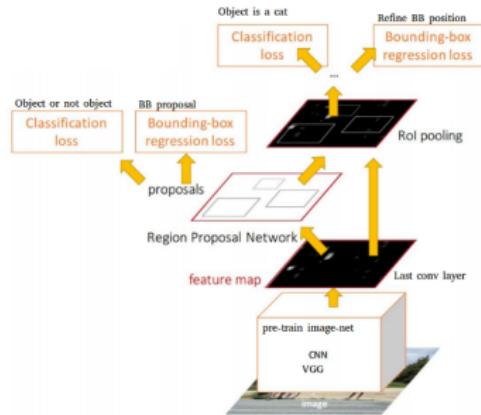
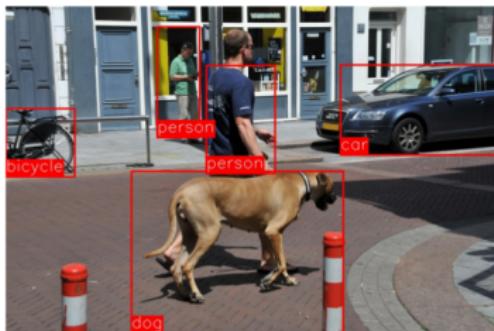
$$f(x) = g(u(x), v(x))$$

$$\Rightarrow \frac{\partial f}{\partial x_i} = \frac{\partial g}{\partial u} \frac{\partial u}{\partial x_i} + \frac{\partial g}{\partial v} \frac{\partial v}{\partial x_i}$$

ATTENTION : RIEN À VOIR AVEC UN NEURONE
c'est général à la dérivation

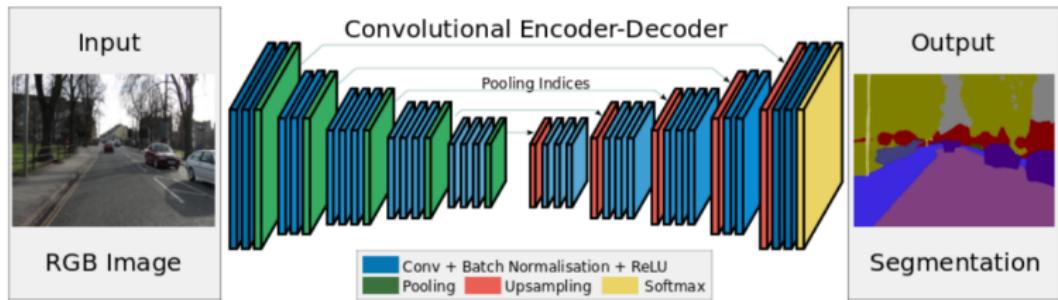
Rappel

qu'est ce qu'on peut faire avec



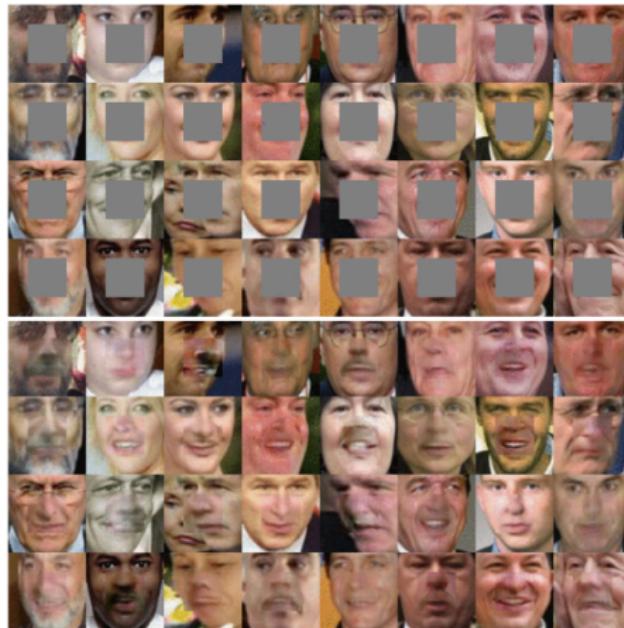
Rappel

qu'est ce qu'on peut faire avec



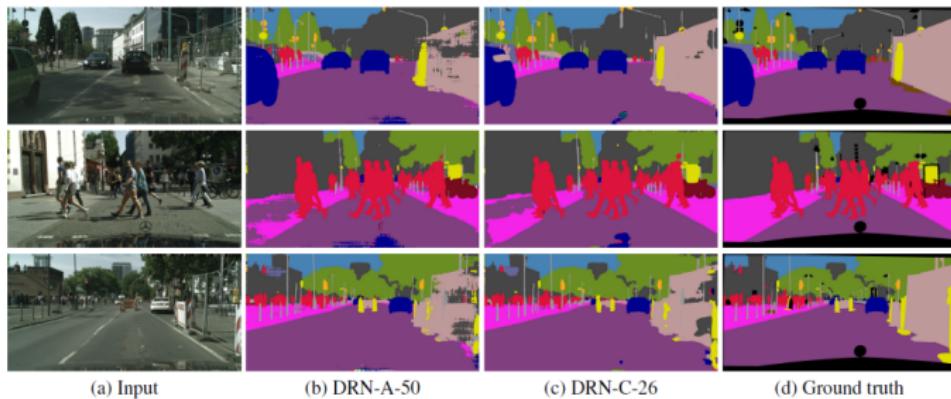
Rappel

qu'est ce qu'on peut faire avec



En pratique

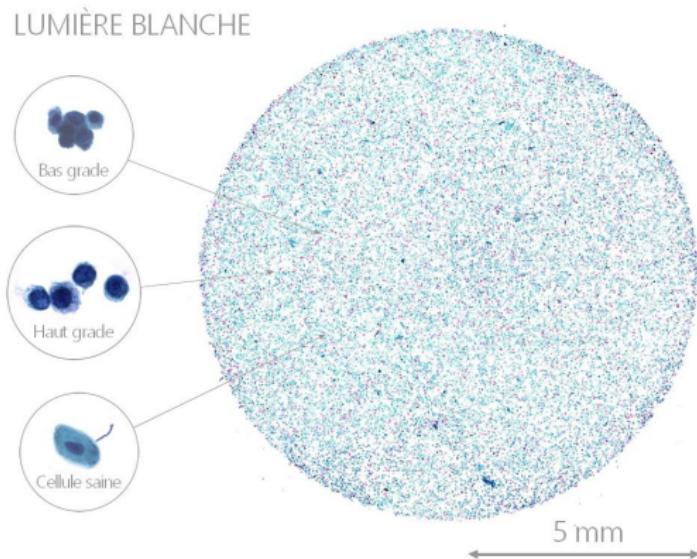
Autonomous driving



En pratique

Diagnosis

<https://vitadx.com/>



:-)

Mais, non, ça ne sert pas à sauver des vies...

Ça sert à quoi ?

2 forces principales :

- ▶ le renseignement massif (EU)
- ▶ la vidéo surveillance massive (Chine)

EU

En pratique



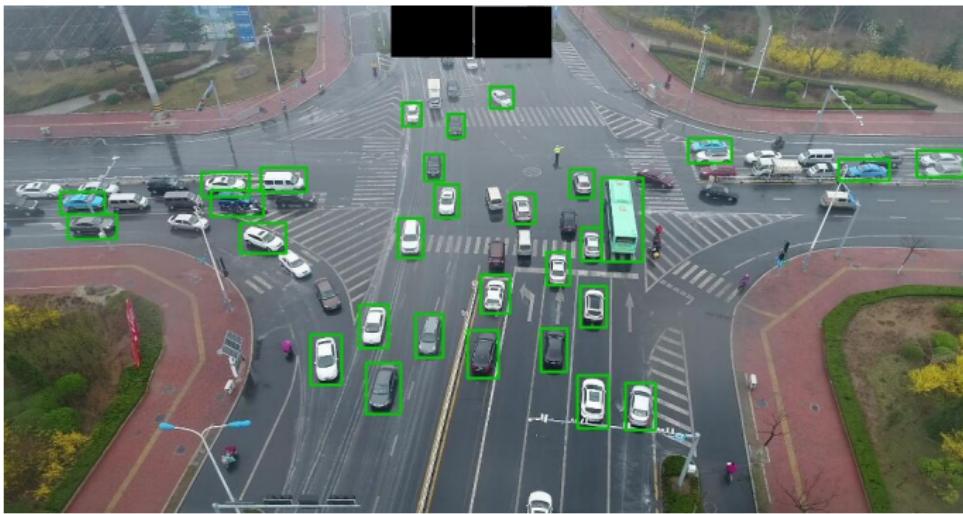
EU

En pratique



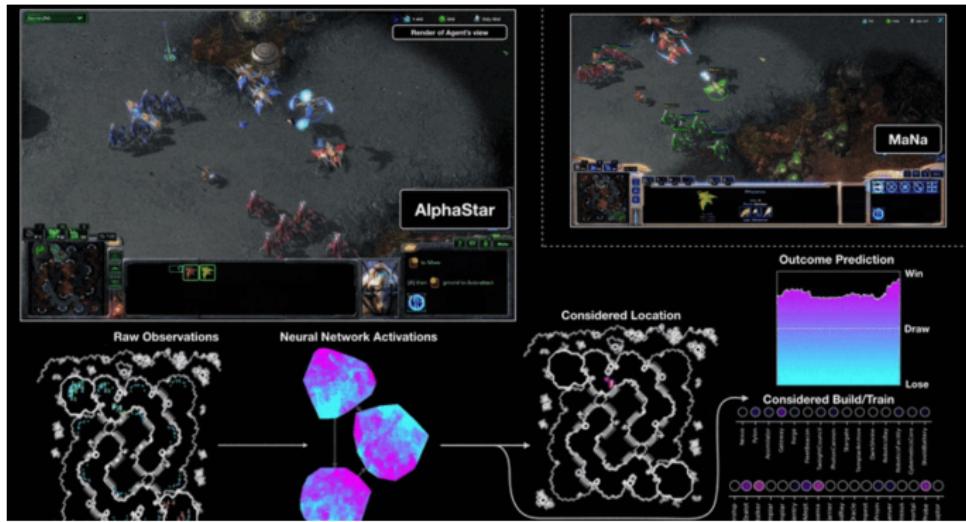
Chine

En pratique



Aujourd'hui

deep reinforcement learning



Aujourd'hui

deep reinforcement learning

Certains considèrent que le machine learning n'a rien à voir avec l'IA - il s'agit surtout d'apprendre des modèles

Mais quelque soit la définition qu'on donne à IA, personne ne peut argumenter qu'alphastar n'est pas de l'IA !

Reinforcement learning

Agent

observe son environnement
peut effectuer une action
l'action modifie l'environnement

Exemple Atari

on voit ce qu'il se passe à l'écran
on peut appuyer sur A, B, haut, bas, droite, gauche ou ne rien faire
le jeu se déroule

Reinforcement learning

Apprentissage

Durant l'apprentissage, on donne à l'agent une information concernant la qualité de ses actions

Ce score peut être éparse :

- ▶ gagner = +100
- ▶ game over = -100
- ▶ sinon 0

ou dense :

- ▶ gagner = +100000
- ▶ game over = -100000
- ▶ sinon le score courant du jeu

Reinforcement learning

Le but de l'apprentissage est de calculer une politique dans le but de récolter un score maximal.

Une fois calculée, cette politique est utilisée pour interagir avec l'environnement.

s_t l'état à l'instant t

a_t l'action à l'instant t

une politique c'est par exemple $Q(s_t, a_t)$ qui estime $Q^*(s_t, a_t)$ le score total qu'on peut obtenir en commençant par faire a_t depuis s_t

Reinforcement learning

$Q^*(s_t, a_t)$ le score total qu'on peut obtenir en commençant par faire a_t depuis s_t

c'est le score élémentaire reçu par le mouvement s_t vers s_{t+1} qu'on note $r(s_t, a_t, s_{t+1})$
plus le score total qu'on peut obtenir depuis s_{t+1}

Équation fondamentale de Q^*

$$Q^*(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma \max_a Q^*(s_{t+1}, a)$$

Reinforcement learning

ex : le plus court chemin vu comme du renforcement

la plus petite distance entre A et C est plus petite
que la distance élémentaire entre A et B
plus la plus petite distance entre B et C

$$D(A, C) = \min_{B \text{ voisin de } A} d(A, B) + D(B, C)$$

Reinforcement learning

Équation fondamentale de Q^*

$$Q^*(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma \max_a Q^*(s_{t+1}, a)$$

Cas fini

résoudre ce programme linéaire

$$\min_Q \sum_{s,a} Q(s, a)$$

$$\forall s, a, s', a', \quad Q(s, a) \geq r(s, a, s') + \gamma Q(s', a')$$

permet de trouver Q^*

Reinforcement learning quand l'espace est trop grand

Équation fondamentale de Q^*

$$Q^*(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma \max_a Q^*(s_{t+1}, a)$$

Cas trop grand voire infini

choisir une structure $Q(s, a, \theta)$ paramétré par θ
optimiser θ de sorte que

$$\forall s, a, s', a', \quad Q(s, a, \theta) \approx r(s_t, a_t, s_{t+1}) + \gamma \max_{a'} Q(s', a', \theta)$$

donne une approximation de Q^* par la famille de fonction $Q(., ., \theta)$

Reinforcement learning quand l'espace est trop grand

Est ce que ça marche ?

Le reinforcement learning marche depuis longtemps sur des petits problèmes.

Mais depuis 2012 cette technologie a permis des exploits (AlphaGo Zero, AlphaStar) par sa rencontre avec le deep learning !

Deep reinforcement learning

Renforcement et deep learning

choisir un réseau de neurone $Q(s, a, \theta)$
optimiser θ par SGD avec l'objectif

$$(r(s, a, s') + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta))^2$$

tout en explorant pour trouver des quadruplets s, a, s', a'

Deep reinforcement learning

AlphaStar



Deep reinforcement learning

Renforcement et deep learning

choisir un réseau de neurone $Q(s, a, \theta)$

optimiser θ par SGD avec l'objectif

$$(r(s, a, s') + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta))^2$$

tout en explorant pour trouver des quadruplets s, a, s', a'

C'est tout ??

oui en théorie mais en pratique...

Difficulté de mise en oeuvre

- ▶ SVM : l'optimisation marche à tous les coups avec peu de puissance
 - pas très expressif mais quand le problème est à peu près séparable ça marche très bien avec peu de données
- ▶ réseau de neurones : l'optimisation est complexe et va converger 3 fois sur 10 - très/trop expressif mais c'est ce qui marche le mieux quand il y a beaucoup de données et des GPU
- ▶ GAN (2 réseaux) : l'optimisation est très complexe et va converger 1/10 - permet de faire des choses jusque là inenvisageable
- ▶ deep reinforcement : l'optimisation est extrêmement complexe et instable ! Elle nécessite énormément de données et de puissance !
aujourd'hui seule quelques équipes dans le monde sont capables de mettre en oeuvre alphago zero - et seule deep mind semble être en mesure de mettre en oeuvre alphastar !

Shapping

Densifier le score est indispensable

Ajouter de l'information via des scores intermédiaires permet d'accélérer la convergence.

choisir un réseau de neurone $Q(s, a, \theta)$, optimiser θ par SGD avec l'objectif

$$(r(s, a, s') + F(s, a, s') + \gamma \max_{a'} Q(s', a', \theta) - Q(s, a, \theta))^2$$

tout en explorant pour trouver des quadruplets s, a, s', a' .

Si $F(s, a, s') = \gamma\phi(s') - \phi(s)$ alors la politique optimal est la même

Et aussi

Tout un tas de trucs

- ▶ replay buffer
- ▶ entropy or ε greedy policy pour ne pas bloquer l'exploration
- ▶ model swapping
- ▶ optimizer, loss
- ▶ ...