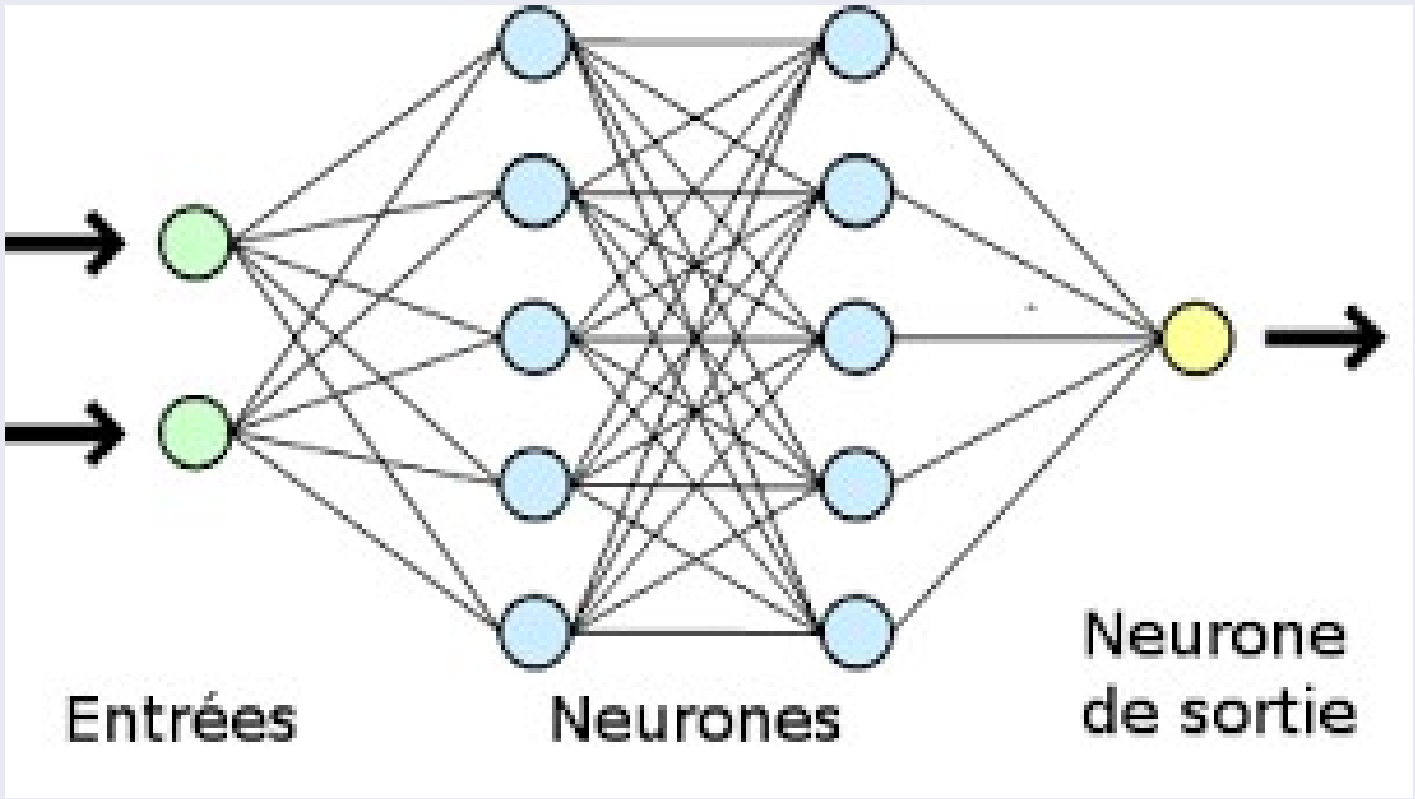
Cours de reconnaissance des formes

cours 5 : deep learning

Adrien CHAN-HON-TONG
ONERA/DTIS département traitement de l'information et système

la dernière fois



Aujourd'hui


Render of Agent's view
AlphaStar
MaNa
Outcome Prediction
Win
Draw
Lose
Considered Location
Raw Observations
Neural Network Activations
Considered Build/Train

Plan

- classification d'images
  - convolution
  - pooling
  - optimisation
  - batch norm, régularisation
- de la classification à un peu tout
  - segmentation
  - detection
  - auto encoding
  - génération
- mais aussi
  - réseau récurrent
  - deep reinforcement learning

Rappel

classification supervisée de vecteurs

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n) \in \mathbb{R}^D \times \{-1, 1\}$
base de test $(x'_1, y'_1), ..., (x'_n, y'_n) \in \mathbb{R}^D \times \{-1, 1\}$

on choisit une structure de réseau $DL(x, \theta) = K_1$ filtres + relu + K_2 filtre + relu + ... + K_r filtre + relu + 1 filtre appliqué à x

on optimise θ par SGD avec l'objectif que $sign(DL(x_i, \theta)) \approx y_i$

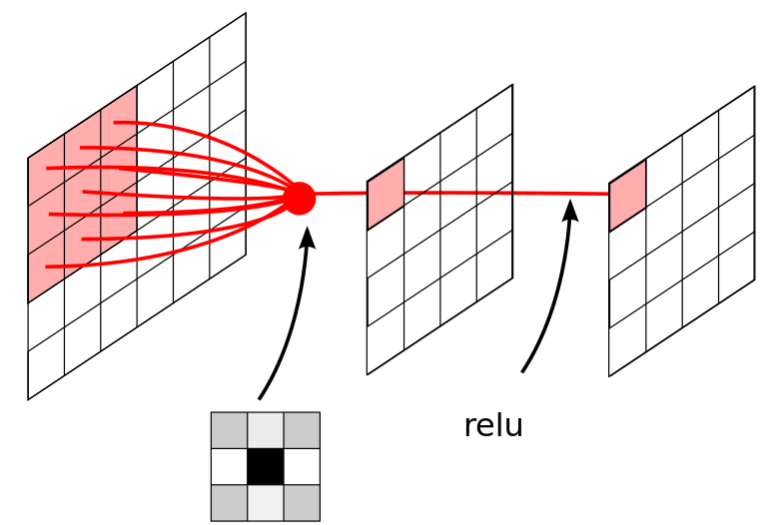
dans l'espoir que $\forall i \in \{1, ..., n\}, sign(DL(x'_i, \theta)) \approx y'_i$

Classification d'images

1 image RGB de 480x340 = 1 vecteur dans $[0, 255]^{163200}$

Il est impossible de faire de la classification d'images sans tenir compte des spécificitées des images

Convolution



l'image $x \in \mathbb{R}^{H \times W \times Ch}$

$+$ le filtre de convolution $w \in \mathbb{R}^{\Delta H \times \Delta W \times Ch}$

$=$ une *image* en sortie $y \in \mathbb{R}^{H-\Delta H \times W-\Delta W}$

$$y_{h,w} = \sum_{dh,dw,ch \in [0,\Delta H] \times [0,\Delta W] \times [0,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$

Convolution

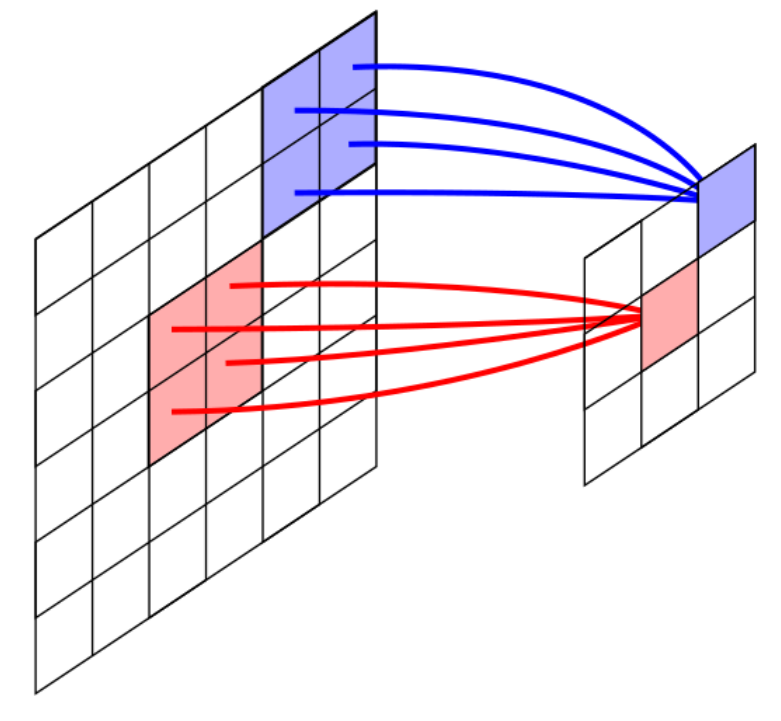
$$y_{h,w} = \sum_{dh,dw,ch \in [0,\Delta H] \times [0,\Delta W] \times [0,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$

comme un filtre linéaire mais prenant en compte l'aspect spatial des images

permet d'extraire de l'image des informations locales (contrastes, texture, bords...) avec un nombre de poids très inférieur

à compléter avec un non linéarité (ex : relu)

Pooling



l'image en entrée $x \in \mathbb{R}^{H \times W \times Ch}$

l'image en sortie $y \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times Ch}$

$$y_{h,w,ch} = \max_{dh,dw \in [0,1]} x_{2h+dh,2w+dw,ch}$$

Pooling

$$y_{h,w,ch} = \max_{dh,dw \in [0,1]} x_{2h+dh,2w+dw,ch}$$

augmentation du champs récepteur de la convolution
invariance à de petites transformations spatiales

Convolution et pooling

Aller voir l'excellente illustration animée
http ://cs231n.github.io/convolutional-networks/

(convolution 3x3 mélangée à un pooling sur une image 7x7 avec un padding de 1)

Optimisation

Prétraitement

Les réseaux s'optimisent mieux sur des données centrées réduites : $\sum_n x_n = \mathbf{0}$ et $\sum_n x_n | x_n = 1$

Data augmentation

si le problème possède des invariants, on peut essayer de forcer le réseau à les respecter ou juste donner au réseau des entrées transformer selon ces invariants

batch

calcul de $\nabla_\theta l$

tous les exemples : descente de gradient - trop long

1 exemple : pure SGD - trop instable

$\rightarrow 1 < k \ll$ tous - k est souvent déterminé par la taille du GPU

Optimisation

Lissage du gradient

SGD : $\theta_{t+1} = \theta_t - \lambda \nabla_\theta l$

adagrad : $\theta_{t+1} = \theta_t - \lambda \frac{\nabla_\theta l}{||\nabla_\theta l|| + \epsilon}$

momentum : $\theta_{t+1} = \theta_t - \lambda \omega_{t+1}$ avec $\omega_{t+1} = \gamma \omega_t + (1 - \gamma) \nabla_\theta l$

adam, ...

Initialisation des poids

uniforme, gaussien, xavier, ...

Batch normalisation

idée : centrer réduire à chaque couche : $\sum_n x_n = \mathbf{0}$ et $\sum_n x_n | x_n = 1$

impossible de faire passer tous les exemples à chaque pas de gradient

$\rightarrow$ le faire sur le batch et maintenir une approximation de la moyenne variance !

- : besoin de gros batch

+ : même dynamique à chaque couche, le gradient peut se propager sur 100 couches !

Mais aussi

- fonction de cout
- dropout
- gradient clipping
- norme spectrale
- ...

Bref

classification d'images

- architecture : convolution, pooling, batch norm
- optimisation
- fonction de cout
- et bien d'autres choses

ATTENTION

Il y a des tendances
sgd + momentum > sgd
gros batch > petit batch
batch normalisation > pas de batch normalisation
Mais on trouvera toujours des contre exemples

Bref

classification supervisée d'images

base d'apprentissage $(x_1, y_1), ..., (x_n, y_n)$ et base de test $(x'_1, y'_1), ..., (x'_n, y'_n)$

on choisit une structure de réseau $DL(x, \theta)$: convolution / pooling / filtre / batch norm / ...

on optimise θ avec SGD / Adam / ... avec l'objectif que $sign(DL(x_i, \theta)) \approx y_i$

dans l'espoir que $\forall i \in \{1, ..., n\}, sign(DL(x'_i, \theta)) \approx y'_i$

ATTENTION

Il y a des tendances sur ces choix

mais on trouvera toujours des contre exemples

Je vous avais prévenu...

C'est très expérimental !

Plan

- classification d'images
  - convolution
  - pooling
  - optimisation
  - batch norm, régularisation
- de la classification à un peu tout
  - segmentation
  - detection
  - auto encoding
  - génération
- mais aussi
  - réseau récurrent
  - deep reinforcement learning

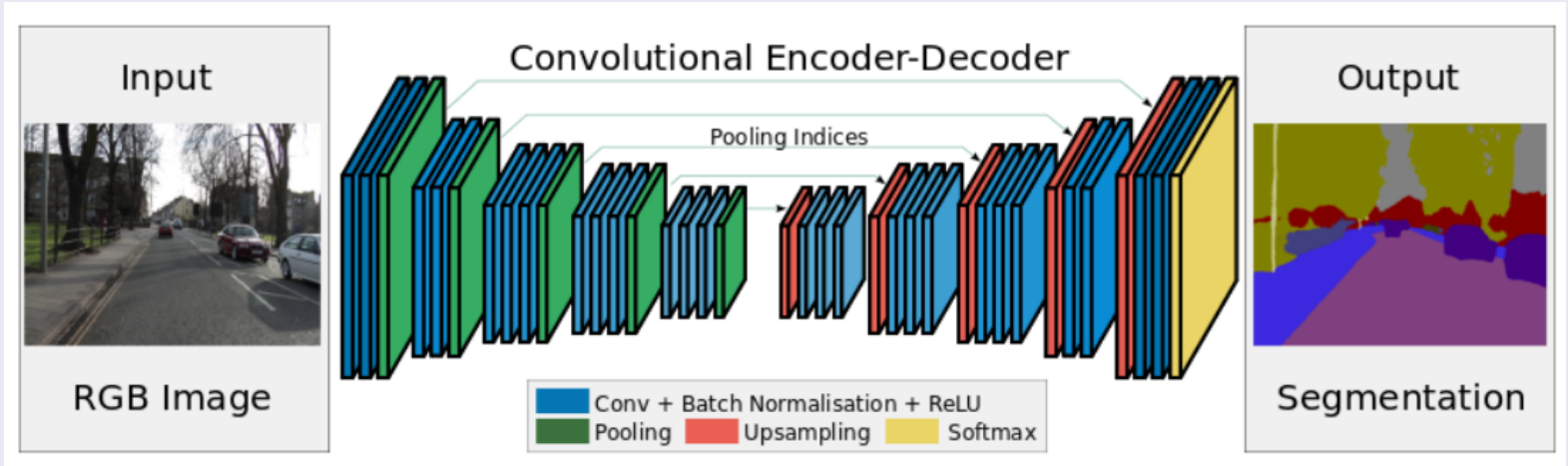
Classification $\rightarrow$ segmentation

l'image $x \in \mathbb{R}^{H \times W \times Ch}$

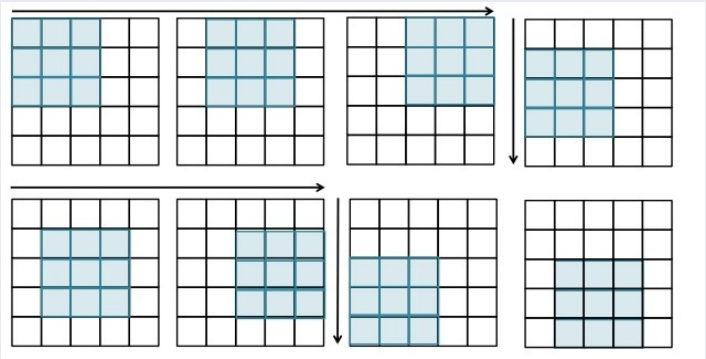
$+$ le filtre de convolution $w \in \mathbb{R}^{\Delta H \times \Delta W \times Ch}$

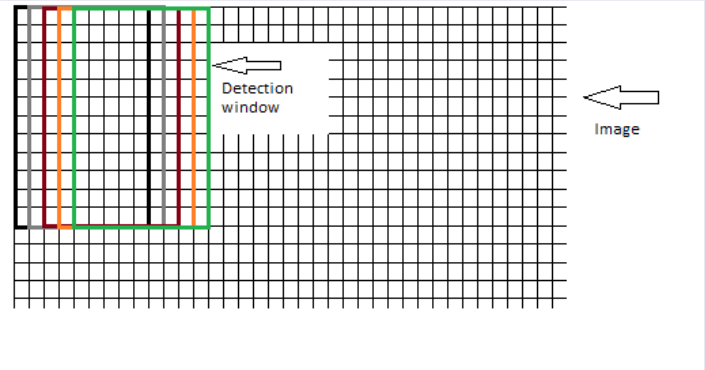
$=$ une *image* en sortie $y \in \mathbb{R}^{H-\Delta H \times W-\Delta W}$

$$y_{h,w} = \sum_{dh,dw,ch \in [0,\Delta H] \times [0,\Delta W] \times [0,Ch]} x_{h+dh,w+dw,ch} \times w_{dh,dw,ch}$$



Classification $\rightarrow$ détection

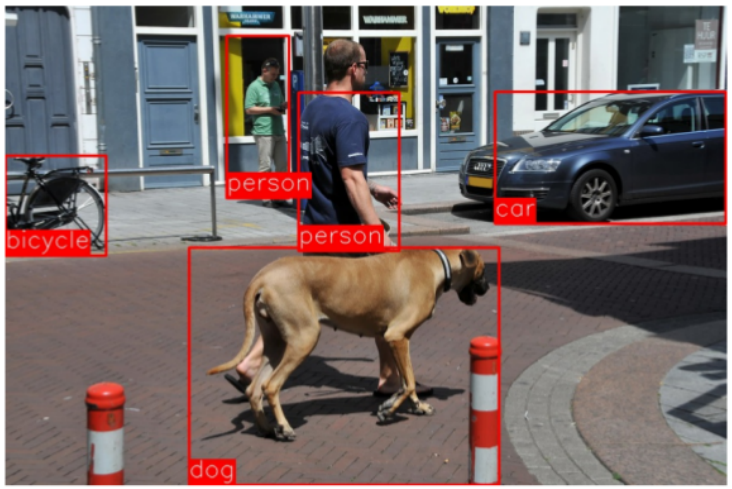


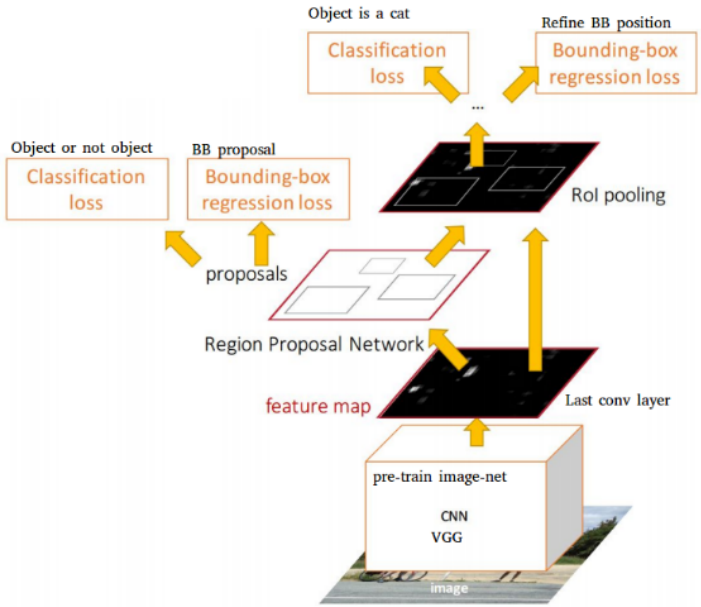


si on sait classer, on sait +/- détecter

Classification → détection

brutalement, il y a trop de boites

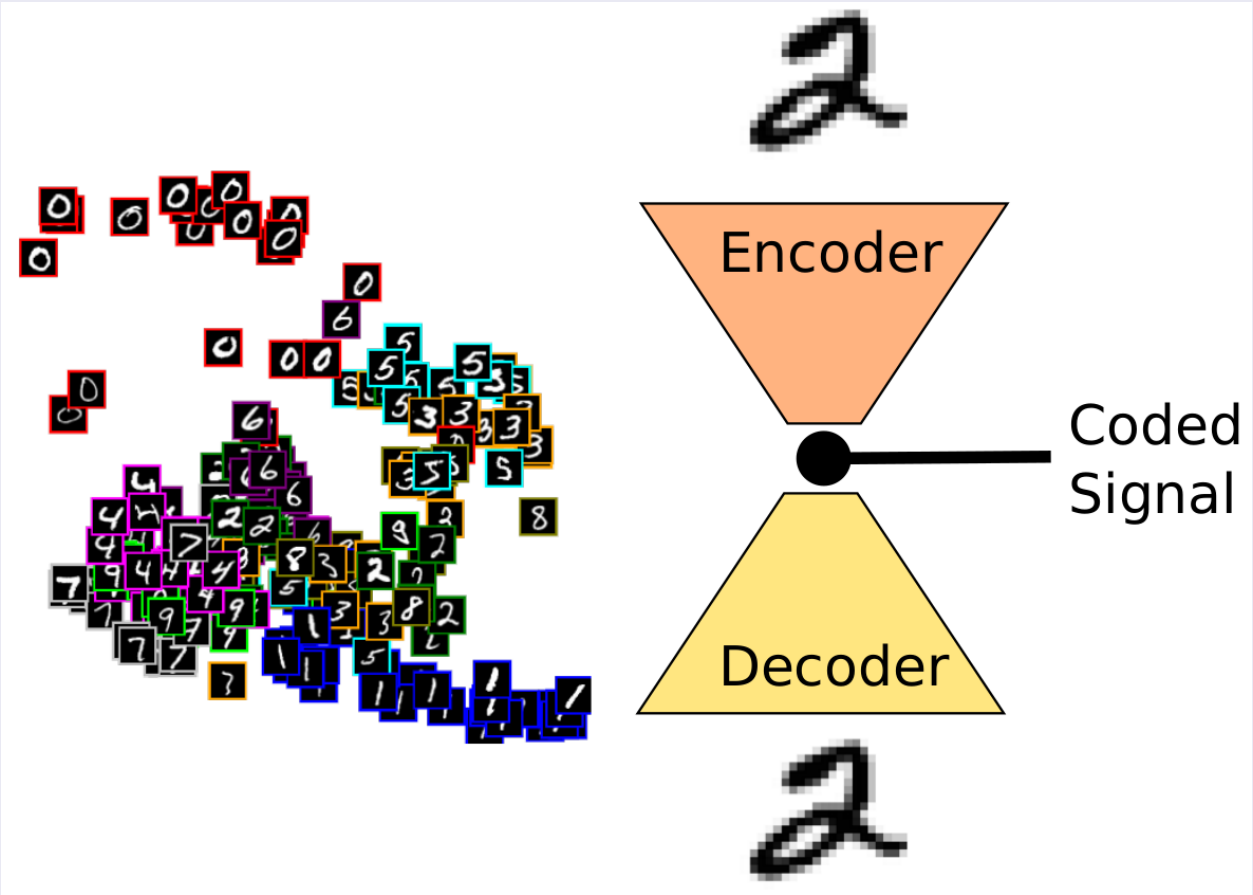




mais on peut le faire intelligement...

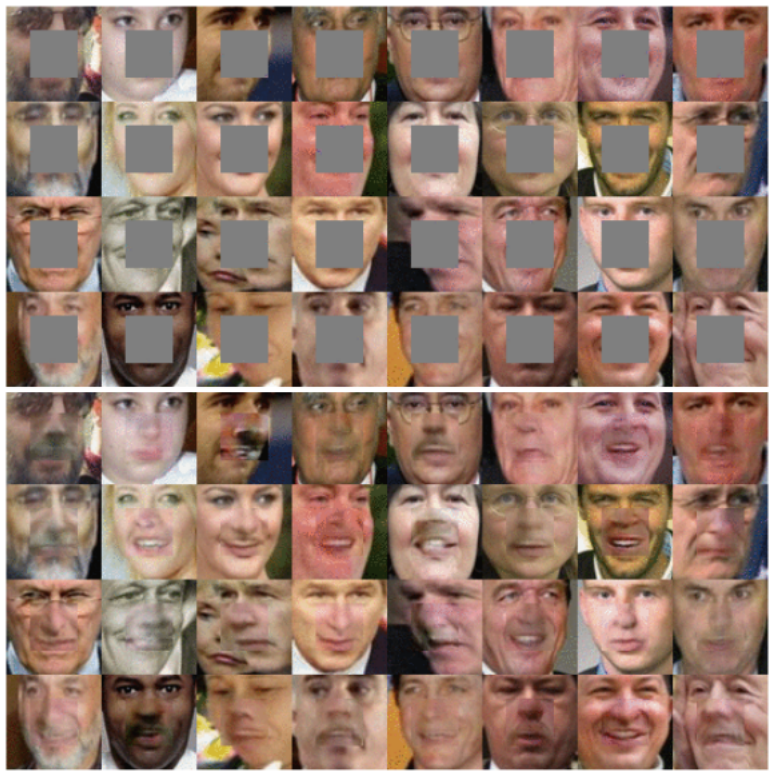
Classification $\longrightarrow$ autoencoder

On cherche à prédire l'entrée en forçant une perte d'information



Classification $\rightarrow$ génération

1 réseau qui génére des images



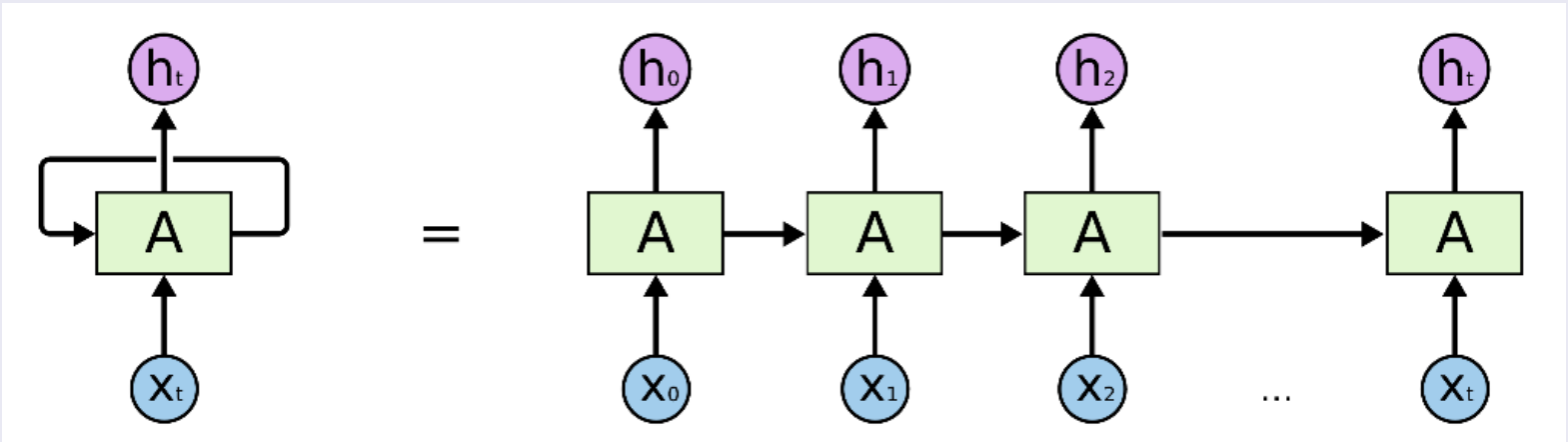
1 réseau qui classe les images en vrai/fausse

les 2 appris pour des objectifs opposées mais en forçant le discriminateur à aider le générateur

Plan

- classification d'images
  - convolution
  - pooling
  - optimisation
  - batch norm, régularisation
- de la classification à un peu tout
  - segmentation
  - detection
  - auto encoding
  - génération
- mais aussi
  - réseau récurrent
  - deep reinforcement learning

Réseau récurrent



conserve une *mémoire* du passé

peut traiter des signaux de longueur variable

2 liens pour ceux que ça intéresse :
www.quora.com/In-LSTM-how-do-you-figure-out-what-size-the-weights-are-supposed-to-be

towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

deep reinforcement learning

juste pour votre culture - ce sera pas à l'exam

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

Apprentissage

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

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 permettre à l'agent de récolter un maximum de score.
Une fois calculer, cette politique est utilisée pour interagir avec l'environnement (les informations fournies durant l'apprentissage peuvent ne plus être présentes).

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 fondammentale de Q^*

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

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

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

Équation fondammentale 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

Équation fondammentale 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 ?

? ?

Deep reinforcement learning ?

Équation fondammentale 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

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

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

Deep reinforcement learning

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

$$relu(r(s, a, s') + \gamma Q(s', a', \theta) - Q(s, a, \theta))$$

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

Deep reinforcement learning

C'est tout ?

Oui mais ATTENTION

C'est encore plus expérimental que le deep learning...