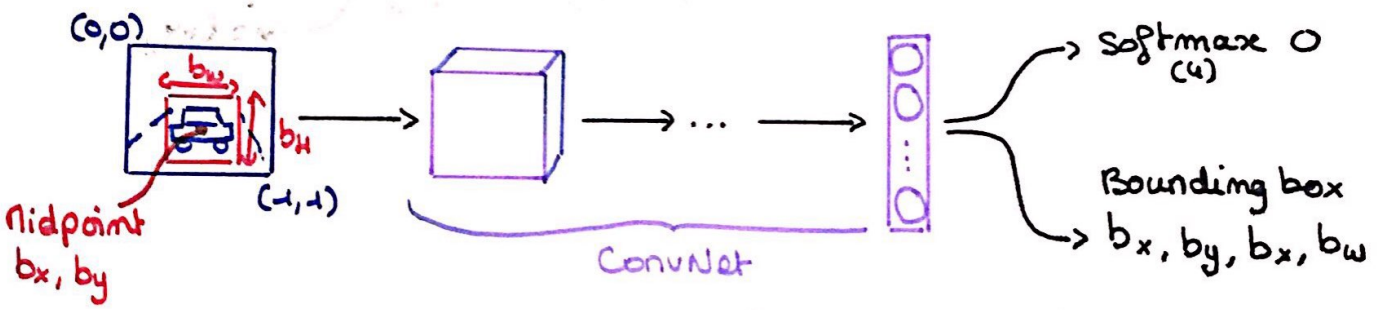
Détection d'objets (1)

=> L'algorithme est responsable de la classification avec localisation de l'image (ie. mettre un cadre autour de l'objet) = 1 objet

≠ Problème de détection = plusieurs objets à détecter dans l'image (de ≠ catégo).

Classification avec localisation



=> New target vector

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_H \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

p_c : probabilité qu'il y ait un objet qui on essaye de détecter (loss = logistic regression)

b_x, b_y, b_H, b_w : bounding box (loss = squared error)

c_1, c_2, c_3 : si $p_c = 1$, probabilité d'être la classe 1, 2 ou 3... (loss = log like)

Si il n'y a pas d'objets, $p_c = 0$ et le reste vaut ? pour "don't care"

Loss

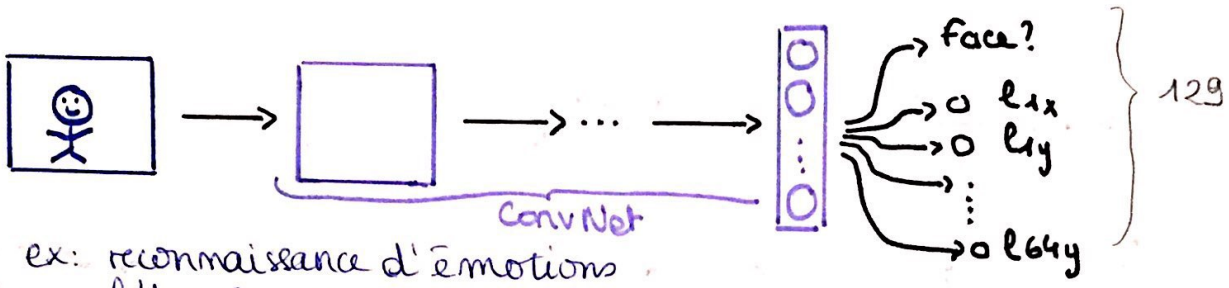
$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_8 - y_8)^2 & \text{si } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{si } y_1 = 0 \end{cases}$$

(on utilise l'erreur au carré pour simplifier).

↳ on s'intéresse qu'à la précision de l'algo pour estimer y_1

landmark detection

=> Algo qui reconnait des points particuliers du visage (par exemple, les contours de l'oeil, de la bouche)
Rend les coordonnées (x, y) des points.



ex: reconnaissance d'émotions
filtres Snapchat.

On doit avoir les labels.

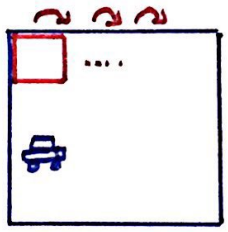
des labels doivant rester les mêmes d'1 img à une autre.

Sliding windows detection

Idée :

(1) Entrainer un ConvNet à reconnaitre une voiture en prédisant juste 0 ou 1 (c'est une voiture ou non)

(2)



Choisir une boite et pour chaque partie de l'image, dire si c'est une voiture ou pas en la faisant glisser.

On peut ensuite prendre une boite plus large.

Paramètre à définir : le stride.

Inconvénients : coût de calcul.

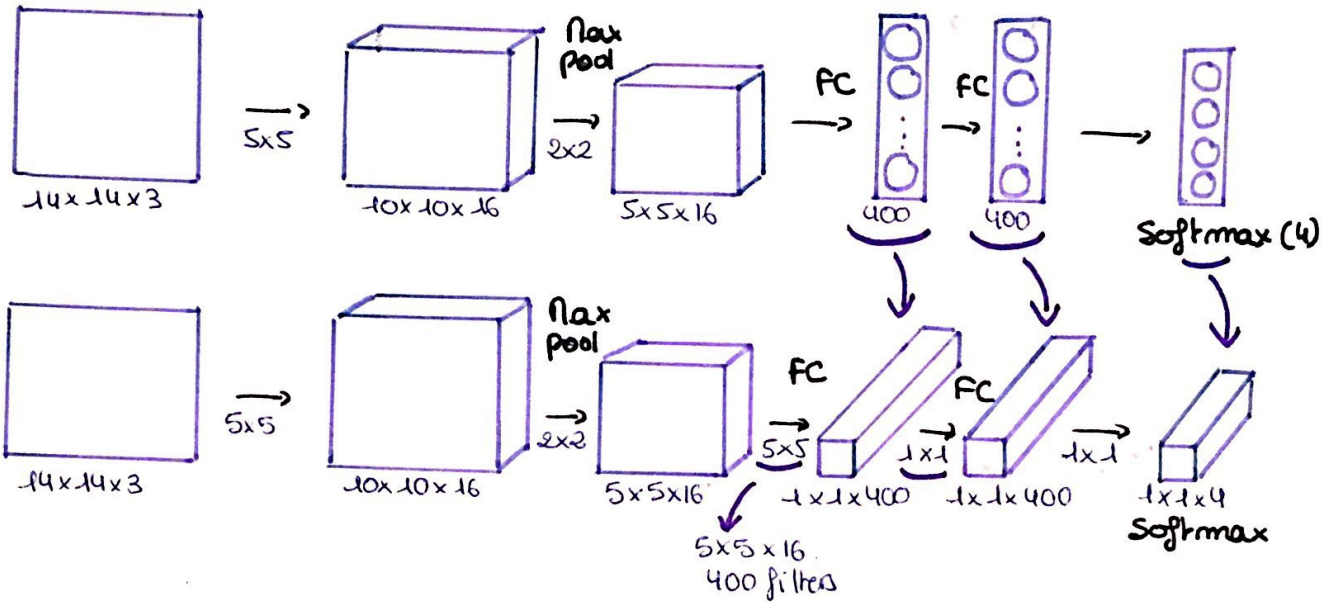
Petit stride => grand nb de calcul => bonne perf.
Grand stride => petit nb de calcul => peut ↘ perf

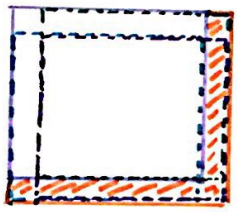
Bonne méthode mais il existe des méthodes plus efficaces.

Détection d'objets (2)

Implementation de la fenêtre glissante avec convolution

=> Conversion de couches FC en couches de convolution.



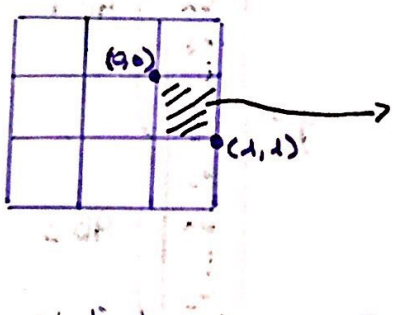


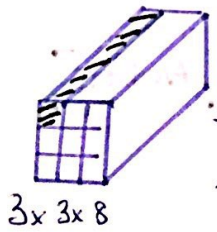
Convolution sur toute l'image

A la place de faire un Convnet sur chaque sous image, il suffit de faire un Convnet sur toute l'image car on obtient les résultats dans l'image finale.

Inconvénients : la position de la boîte ne sera pas précise.

=> Solution : YOLO algorithme. = met une grille sur l'image et analyse chaque cellule de la grille.
(You only look once)



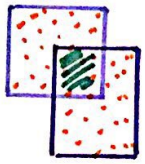


Target sortie.

Ce réseau de neurones => rend des "bounding box" précise

d'objet est assignée qu'à une seule des cellules.
b_H et b_W peuvent être > 1 si la boîte est sur plusieurs cellules.

Intersection over union



$$IoU = \frac{\text{Taille de l'intersection}}{\text{Taille de l'union}}$$

En général, si IoU > 0,5 => "correct" bounding box
Plus l'IoU est élevé, plus la bounding box est précise
On peut choisir un autre seuil

IoU = mesure de recouvrement entre deux bounding box
à quel point les bounding boxe sont similaires

Non-max suppression

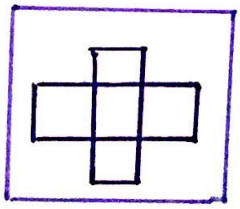
=> permet de s'assurer que l'algo détecte chaque objet une seule fois.

(1) On supprime toutes les boites qui ont une $p_c \leq$ seuil
(2) Tant qu'il reste des boites :
- Sélectionner la boite avec la plus gde p_c
- Retirer toutes les boites qui ont une IoU $\geq$ seuil avec la boite précédente.

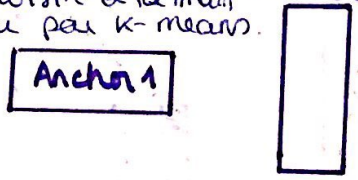
=> Si plusieurs objets de ≠ catégories, il faut faire tourner cet algo plusieurs fois.

Anchor boxes

=> Pour que chaque cellule détecte plusieurs objets.



On peut les choisir à la main ou par K-means.



Chaque objet est assigné à la cellule qui contient son point "milieu" et l'anchor box qui a le plus gd IoU

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ \vdots \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \vdots \\ c_2 \\ c_3 \end{bmatrix} \begin{matrix} \} \text{AB 1} \\ \\ \\ \\ \\ \\ \} \text{AB 2} \\ \\ \\ \\ \\ \end{matrix}$$

Face recognition

Face verification VS **Face recognition**

- Input image, name/ID
- Sortie : l'image correspond à la personne

1:1

- Base de données de K personnes
- Sortie : ID si l'image est parmi les K personnes

1:K

=> Construire un système de face verification et si la précision est suffisamment grande, l'utiliser dans un système de face recognition

One-shot learning

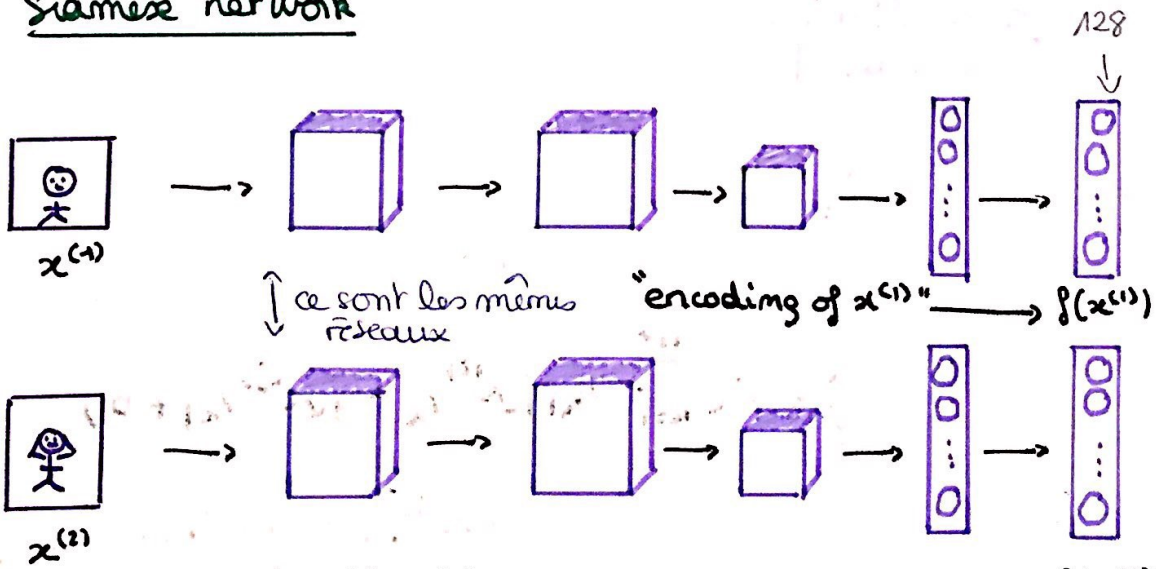
=> Pour construire un syst de reconnaissance faciale, on doit être capable de reconnaitre une personne avec seulement 1 image.

Si on entraine le CONVNET avec les images des ≠ personnes et qu'on utilise une softmax en sortie, ça ne marchera pas très bien car petit jeu de données ou alors à réentrainer à chq fois qu'on a une nouvelle personne.

=> on va plutôt apprendre une **fonction de similarité**

$d(img1, img2)$ = degré de ≠ entre les images

Siamese network



$d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$

la distance est petite si il s'agit de la même personne.

Triplet loss

Etant donnée 3 images A, P, N

A: anchor
P: positive
N: Negative

$$\mathcal{L}(A, P, N) = \max\left(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0 \right)$$

permet de garder la loss positive

hyperparamètre α permet d'éloigner la paire positive de la paire négative = margin

Fonction de perte :

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

=> on a besoin d'un jeu de données avec plusieurs paires d'images de la même personne.

Si on choisit les triplets aléatoirement => trop de triplets seront trop faciles pour l'algo
=> la descente de gradient ne fera pratiquement rien

On choisit des triplets difficiles
=> l'algo va vraiment essayer d'éloigner les deux paires en faisant en sorte qu'il y ait une marge entre les deux.

↗ l'efficacité du calcul

Binary classification

=> Pour prédire si les deux images sont les mêmes (=1) ou pas (=0)

$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \underbrace{|f(x^{(i)})_k - f(x^{(j)})_k|}_{\frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k} = \chi^2} + b \right)$$

taille du vecteur de sortie (en exemple)