

APPRENTISSAGE PROFOND ET IMAGES

# VISION PAR ORDINATEUR

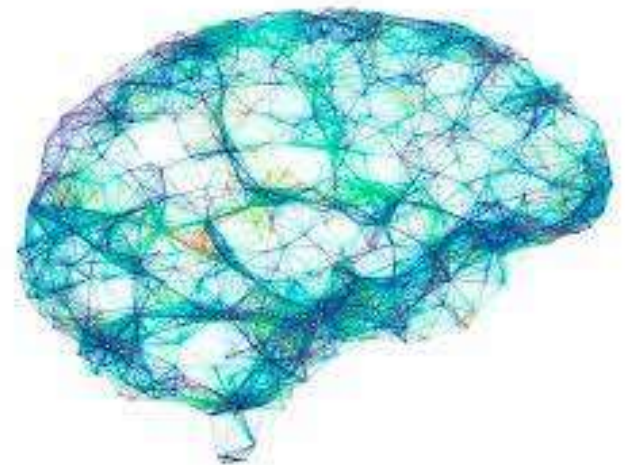
---

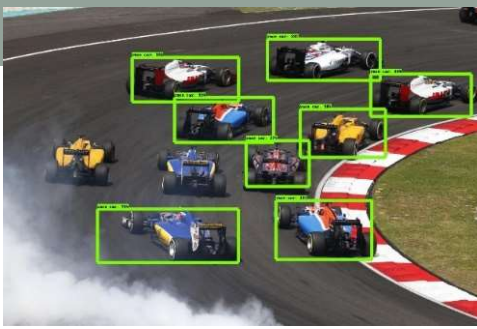
Alexandre Meyer<sup>1</sup>

<sup>1</sup>Equipe SAARA, laboratoire LIRIS



Master ID3D/IA





# DEEP LEARNING A PERMIS DES AVANCÉES FORTES EN VISION PAR ORDINATEUR

- Classification d'images
- Segmentation
- Détection et suivi d'objets
- Génération d'images
- Reconstruction de pose
- Reconnaissance d'action
- ...

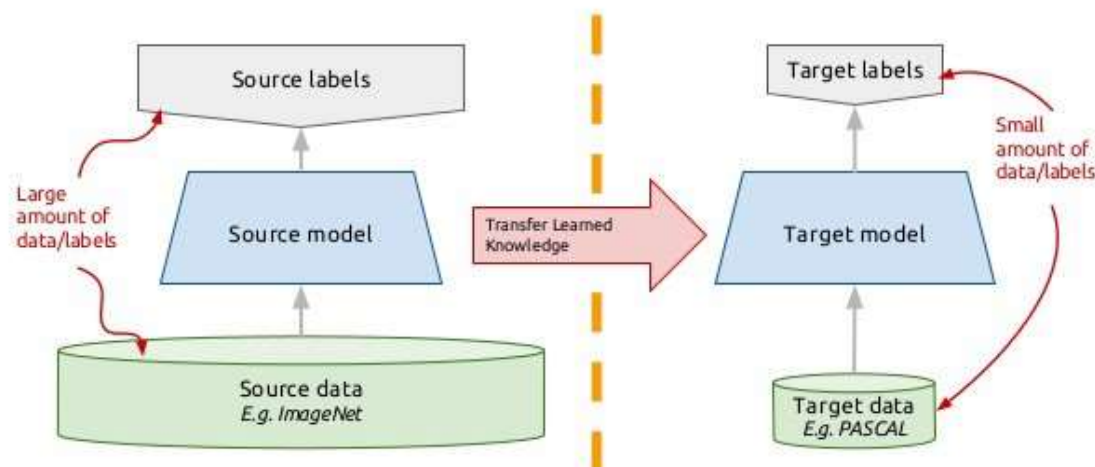


# Transfert d'apprentissage avec CNN

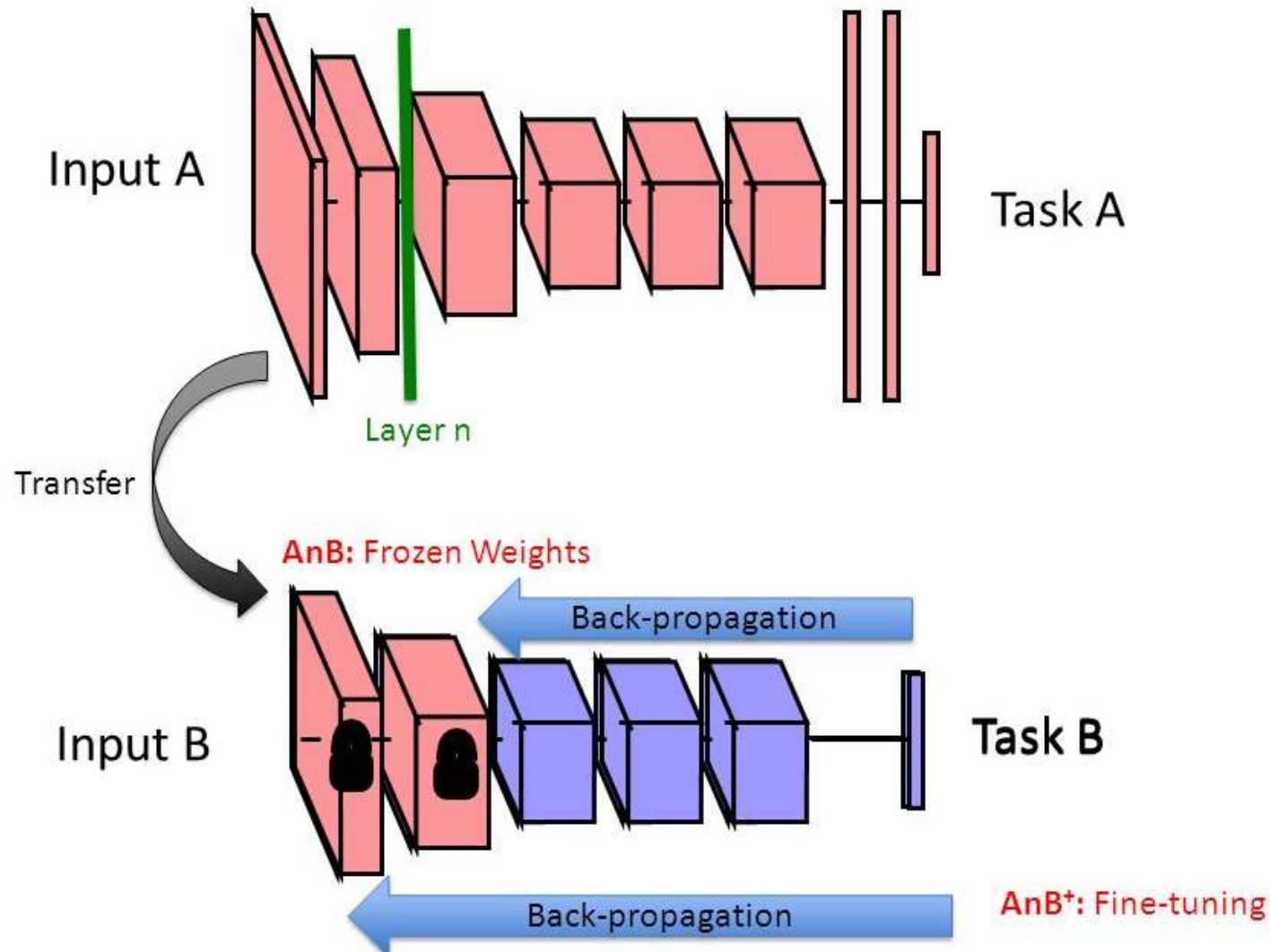
Approche simple pour se spécialiser

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

## Transfer learning: idea



# Transfert d'apprentissage avec CNN



# Transfert d'apprentissage avec CNN

- Un exemple avec PyTorch

```
from torchvision import models  
model = models.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))
```

# Transfert d'apprentissage avec CNN

## Transfert learning

- Un domaine vaste
- Adapter un apprentissage est un des axes forts de la recherche actuelle
  - Dans tous les sous problèmes
  - ...

**How transferable are features in deep neural networks?**

[Jason Yosinski](#), [Jeff Clune](#), [Yoshua Bengio](#), [Hod Lipson](#)

NIPS2014

(papier initial ... → 9500 citations)



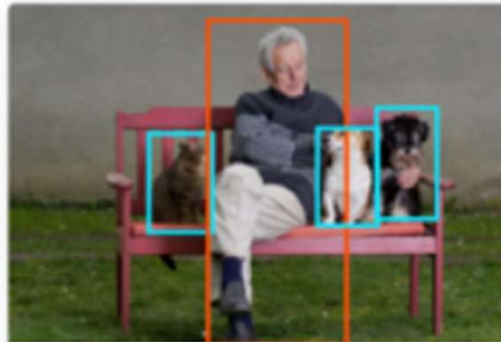
# Segmentation d'images

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



# Segmentation d'images

- Segmentation = un outil de base en vision



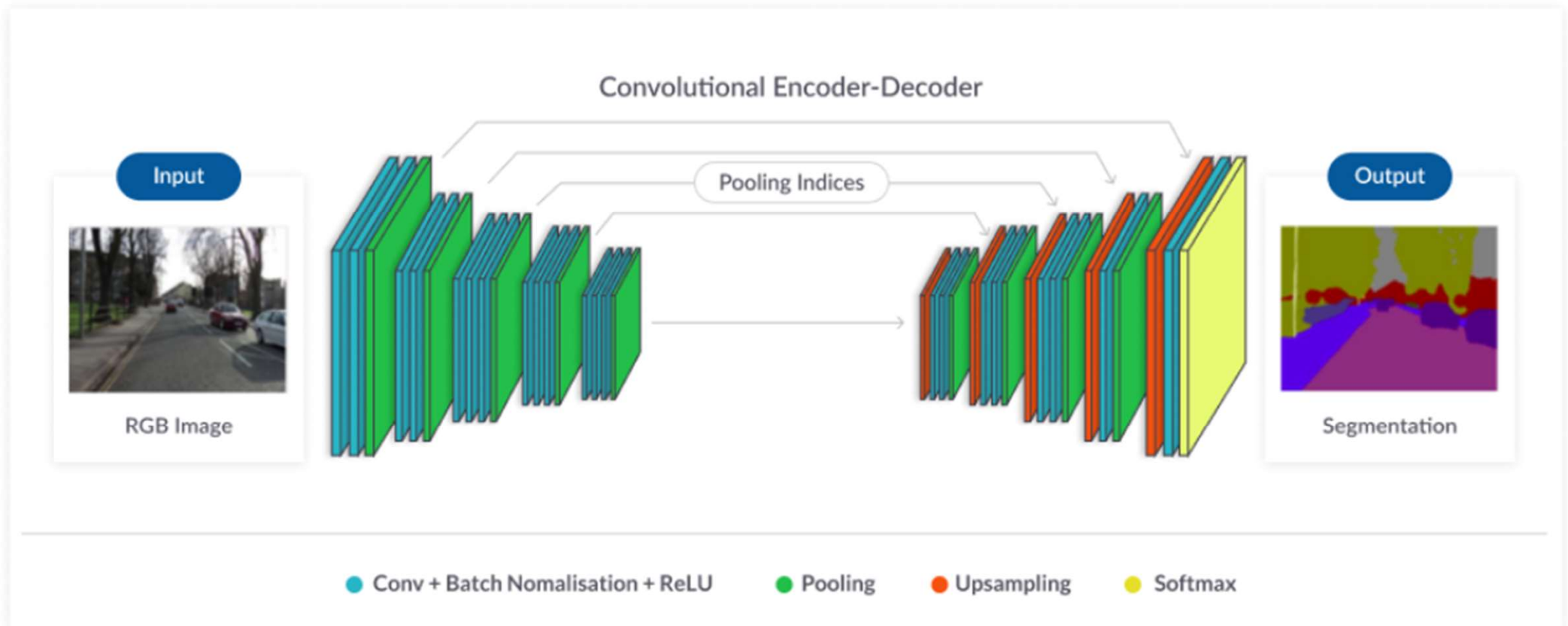


# « Old school segmentation »

- **Thresholding**—divides an image into a foreground and background. A specified threshold value separates pixels into one of two levels to isolate objects. Thresholding converts grayscale images into binary images or distinguishes the lighter and darker pixels of a color image.
- **K-means clustering**—an algorithm identifies groups in the data, with the variable K representing the number of groups. The algorithm assigns each data point (or pixel) to one of the groups based on feature similarity. Rather than analyzing predefined groups, clustering works iteratively to organically form groups.
- **Histogram-based image segmentation**—uses a histogram to group pixels based on “gray levels”. Simple images consist of an object and a background. The background is usually one gray level and is the larger entity. Thus, a large peak represents the background gray level in the histogram. A smaller peak represents the object, which is another gray level.
- **Edge detection**—identifies sharp changes or discontinuities in brightness. Edge detection usually involves arranging points of discontinuity into curved line segments, or edges. For example, the border between a block of red and a block of blue.

# Deep Segmentation

- Encoder => decoder  
Symétrie dans les réseaux

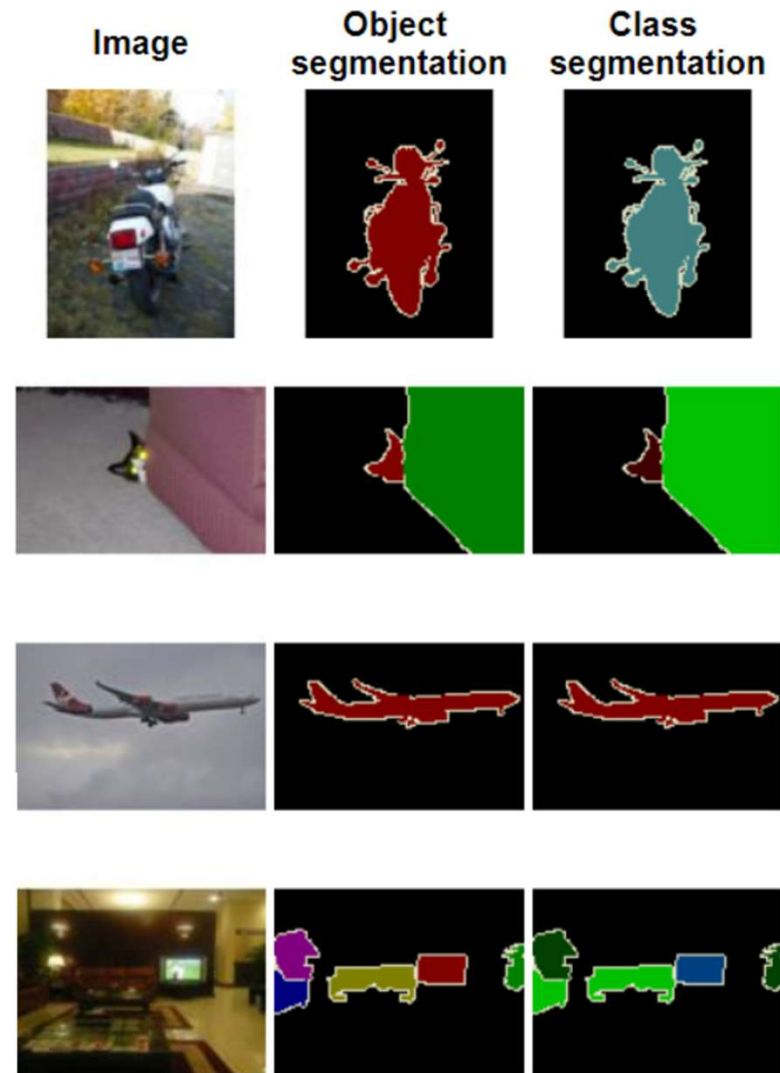


# Deep Segmentation

- 2012 Pascal VOC dataset
  - 10k images for training
  - 10k images for test
  - 10k images for validation
- 2017 COCO
  - 200k images with over 500k c instance segmented
- 2016 Cityscapes
  - 23.5k images, 50 cities



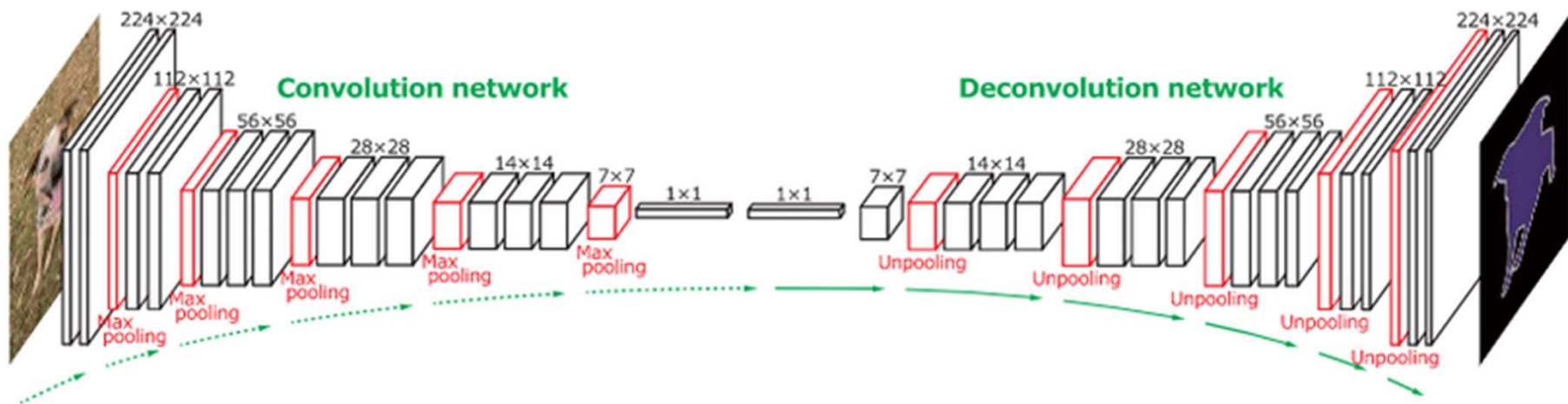
Example of the COCO dataset for object segmentation. Source: <http://cocodataset.org/>



Examples of the 2012 PASCAL VOC dataset for image segmentation. Source: <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html>

# Deep Segmentation

- Based on VGG16



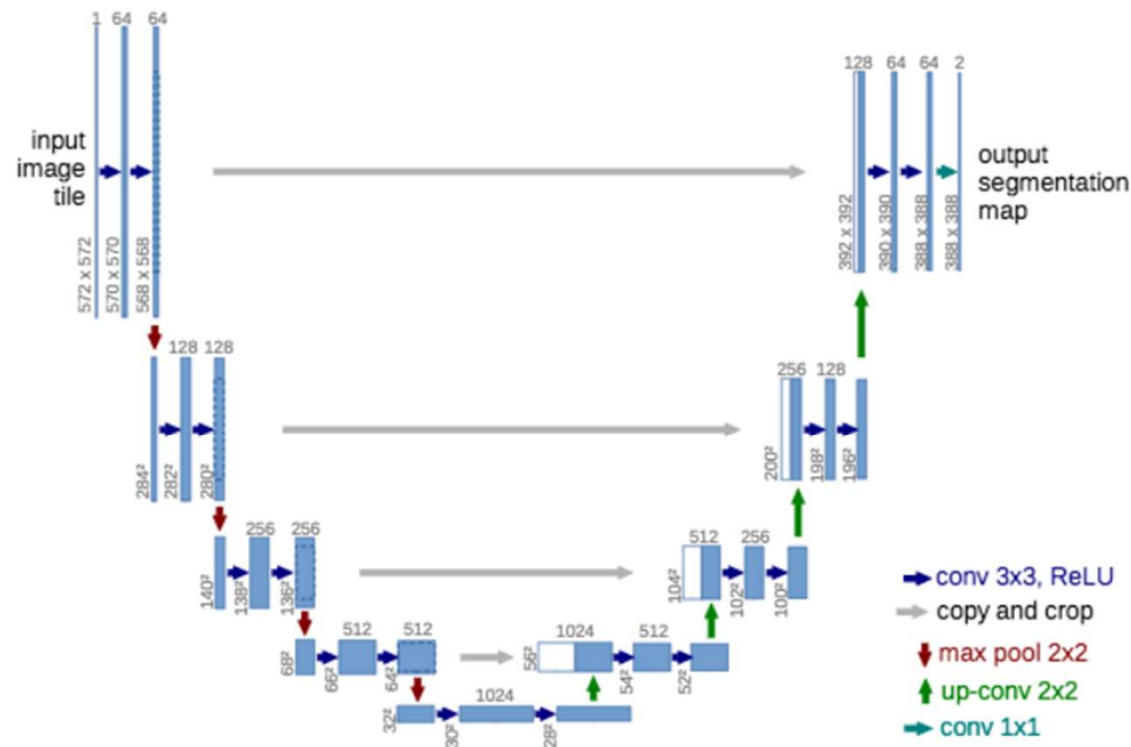
Architecture of the full network. The convolution network is based on the VGG16 architecture. The deconvolution network uses unpooling and deconvolution layers. Source: [H. Noh et al. \(2015\)](#)



# Deep Segmentation

- U-net

[O. Ronneberger et al. \(2015\)](#)



Architecture of the U-net for a given input image. The blue boxes correspond to feature maps blocks with their denoted shapes. The white boxes correspond to the copied and cropped feature maps. Source: [O. Ronneberger et al. \(2015\)](#)



# Sémantique et image

**+ VIDEO**

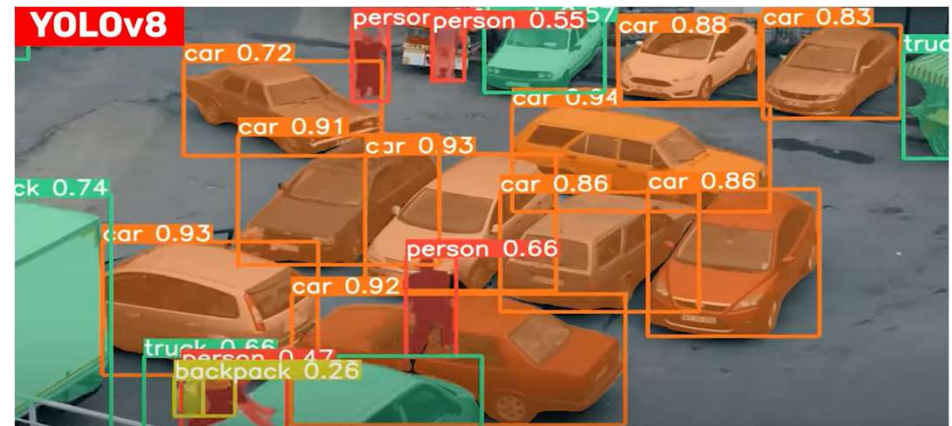
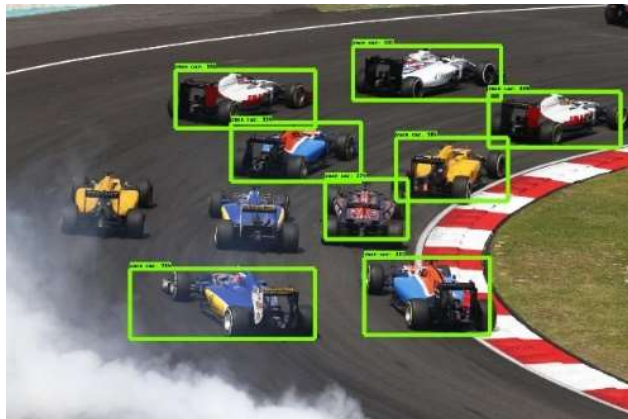
- Entraîne un réseau à regrouper des régions pour segmenter et mettre des labels sur une image



# DEEP LEARNING A PERMIS DES AVANCÉES FORTES EN VISION PAR ORDINATEUR

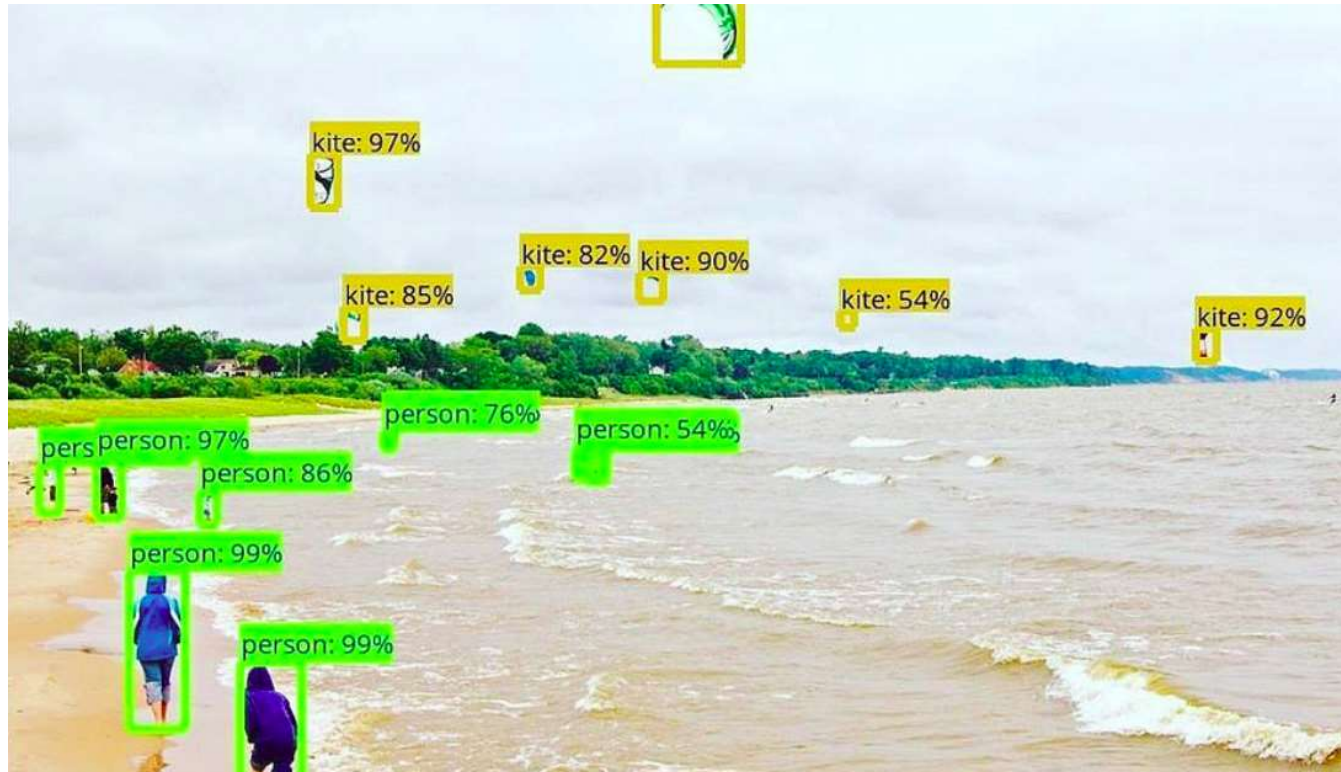
---

- ...
- Détection et suivi d'objets : YOLO
- ...



# 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 peut être grand

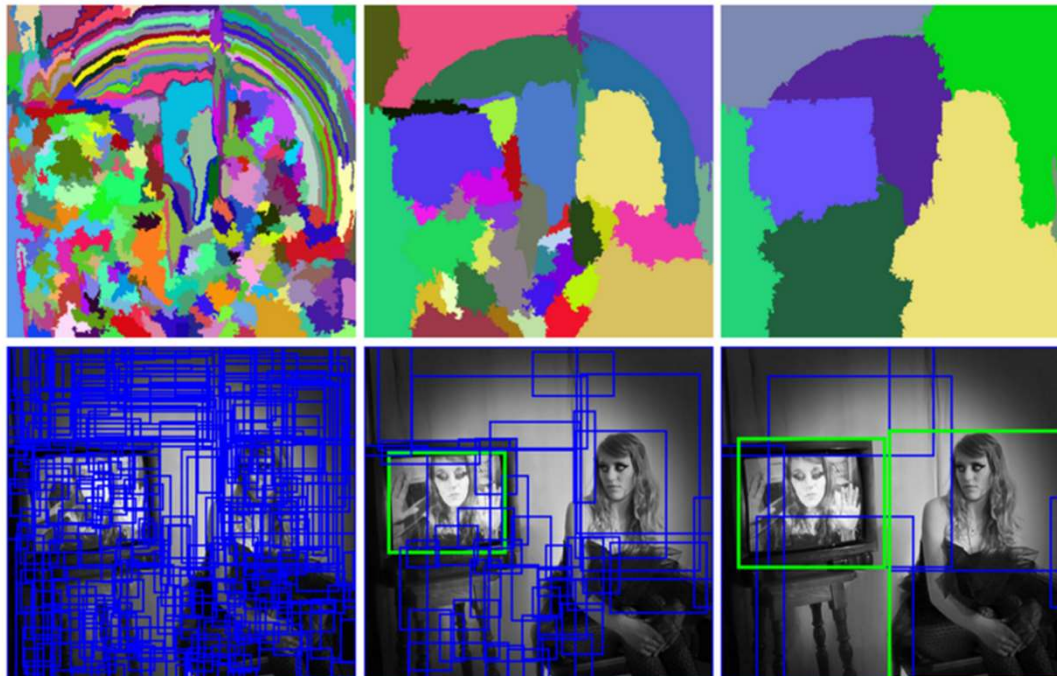




# Détection d'objets

## Region-Based Convolutional Neural Network (R-CNN )

- 1.a Segmentation (algorithme quelconque)
- 1.b Fusion de région : similarité de couleur, texture, forme, ...
- 1.c Chaque région produit une région d'intérêt : ROI



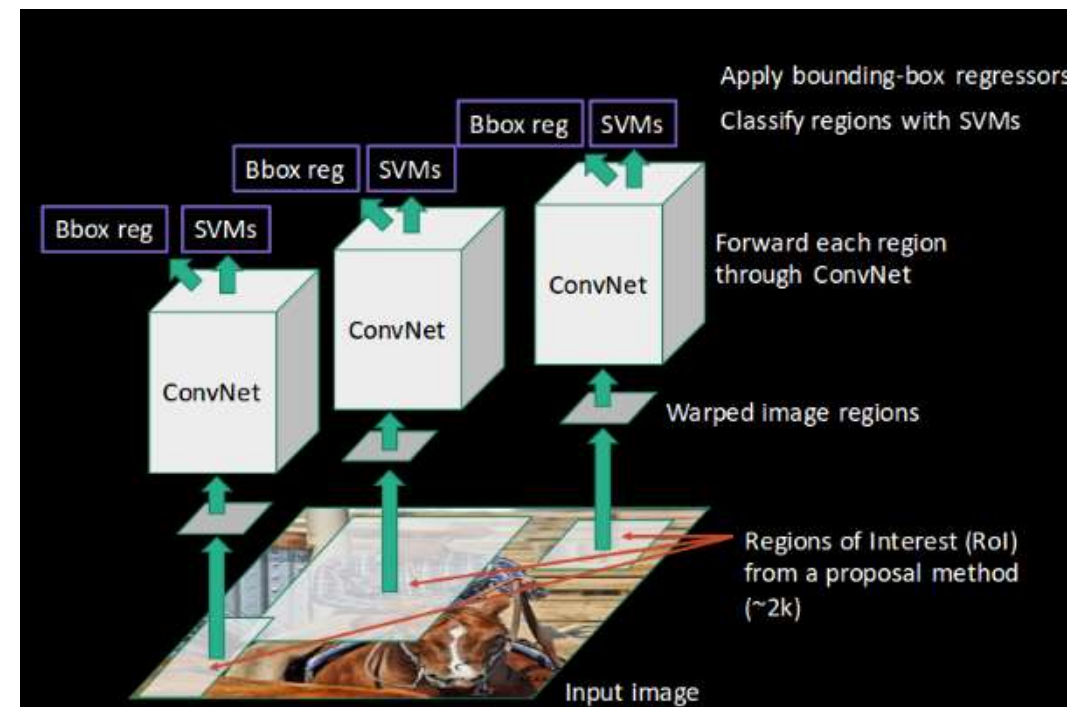
# Détection d'objets

## Region-Based Convolutional Neural Network (R-CNN )

- ROI
  - 2.a CNN → descripteurs
  - 2.b SVM classifie
  - 2.c BoundingBox regression pour ajuster la Bbox

### Problème avec R-CNN

- 2000 régions
  - 2000xNB descripteurs
- 1 minute par image





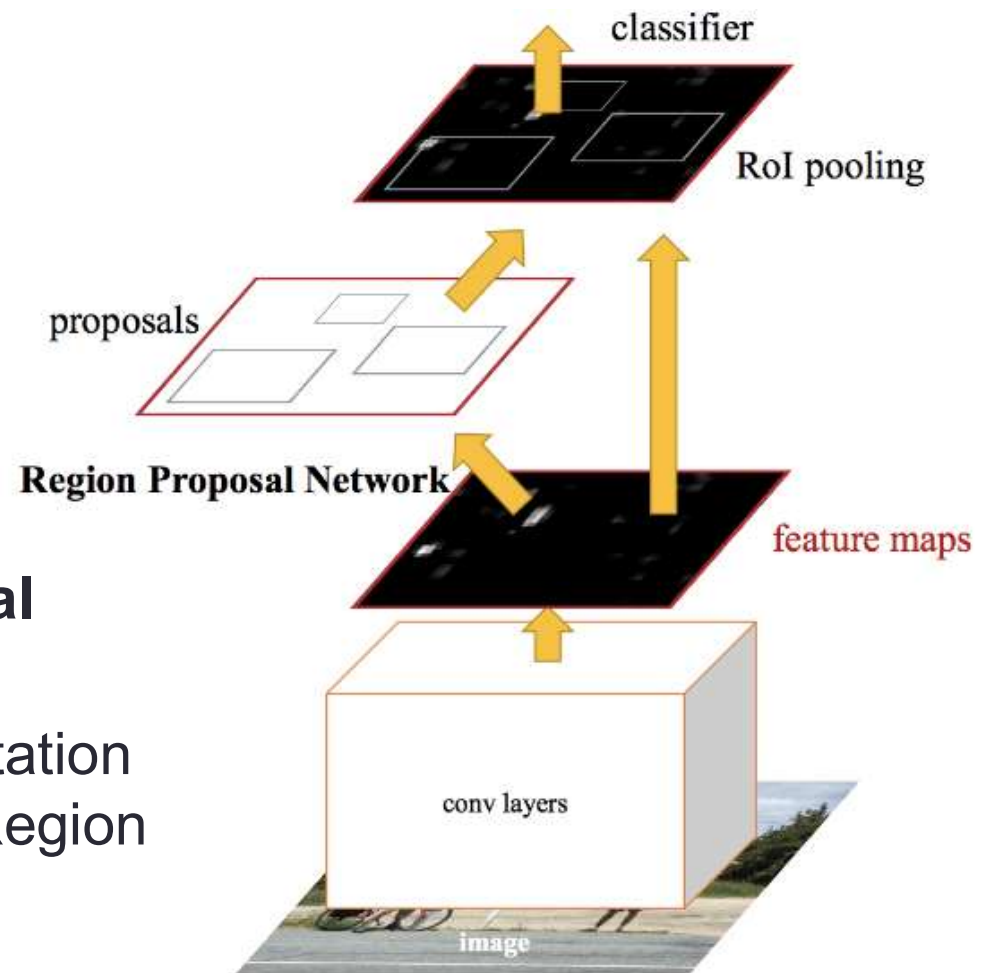
# Détection d'objets

## Fast Region-Based Convolutional Neural Network

- Replace la phase (1) de segmentation/ROI par un CNN
- 2 secondes par images

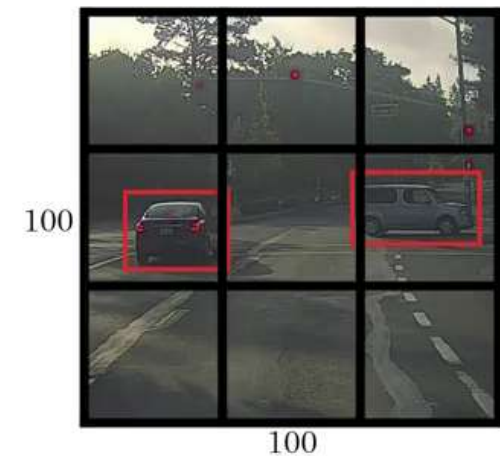
## Faster Region-Based Convolutional Neural Network

- Replace les phases (1) de segmentation et phase (2) sélection de ROI par Region Proposal Network (RPN)
- Temps 0.2 seconde par image

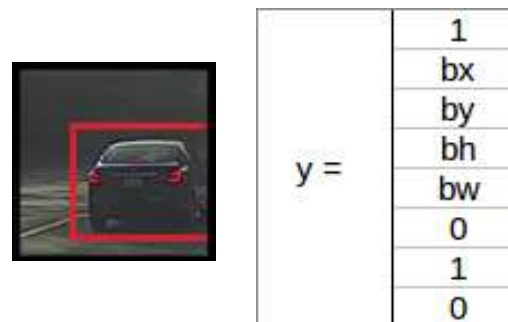
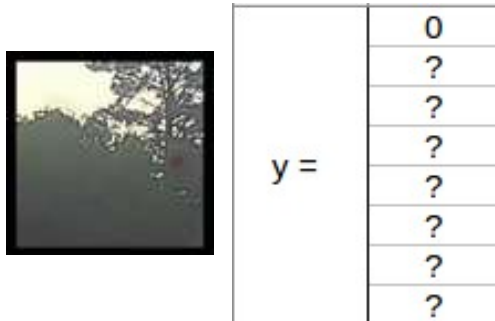


# Détection d'objets

- R-CNN; Fast R-CNN; Faster R-CNN : ok
- 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

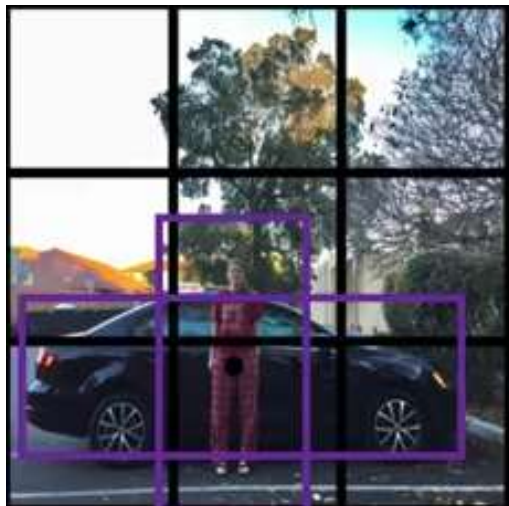
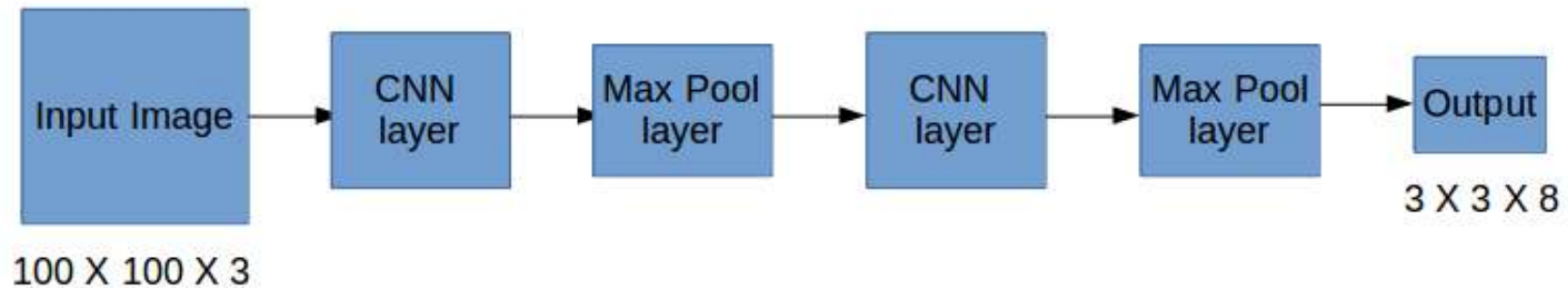
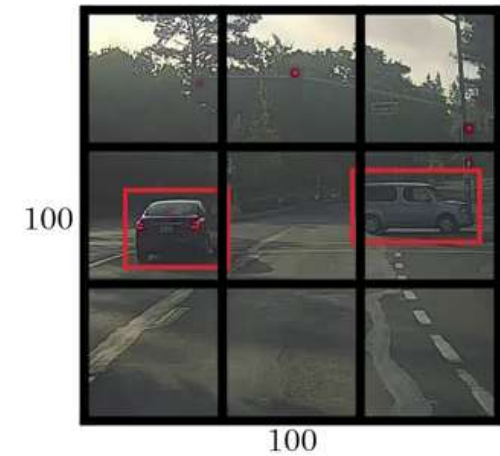


y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

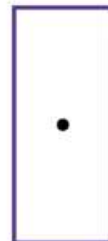


# Détection d'objets

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



Anchor box 1:



Anchor box 2:



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

# YOLO

<https://deci.ai/blog/history-yolo-object-detection-models-from-yolov1-yolov8/>

- v1 (CVPR2016)
- v2 (2017)
  - des boîtes d'ancrage améliorées et une résolution plus élevée
- v3
  - un score d'objectivité supplémentaire pour la prédiction de la boîte englobante et des connexions aux couches du réseau de base. Amélioration des performances sur les objets minuscules grâce à la possibilité d'effectuer des prédictions à trois niveaux de granularité différents
- v4
  - amélioration de l'agrégation des caractéristiques
- v5
  - amélioration de l'architecture
- ...
- v8

# YOLO moderne

- Architecture

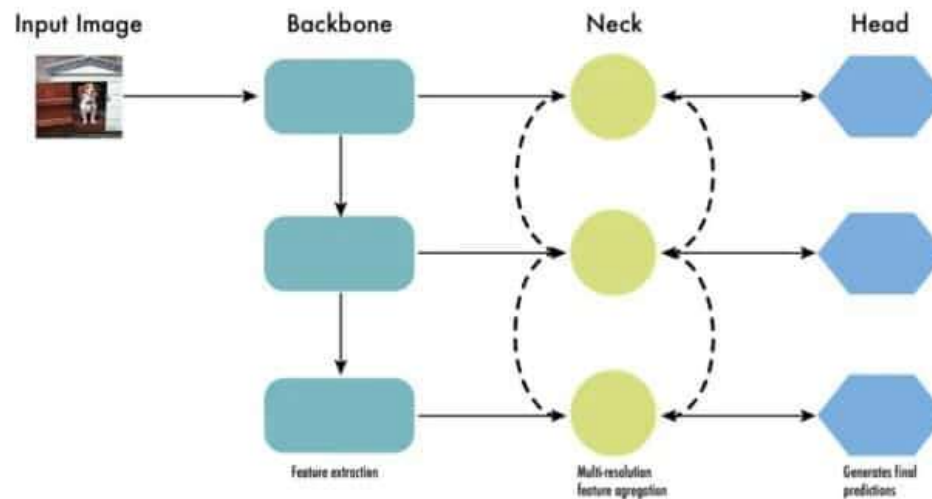
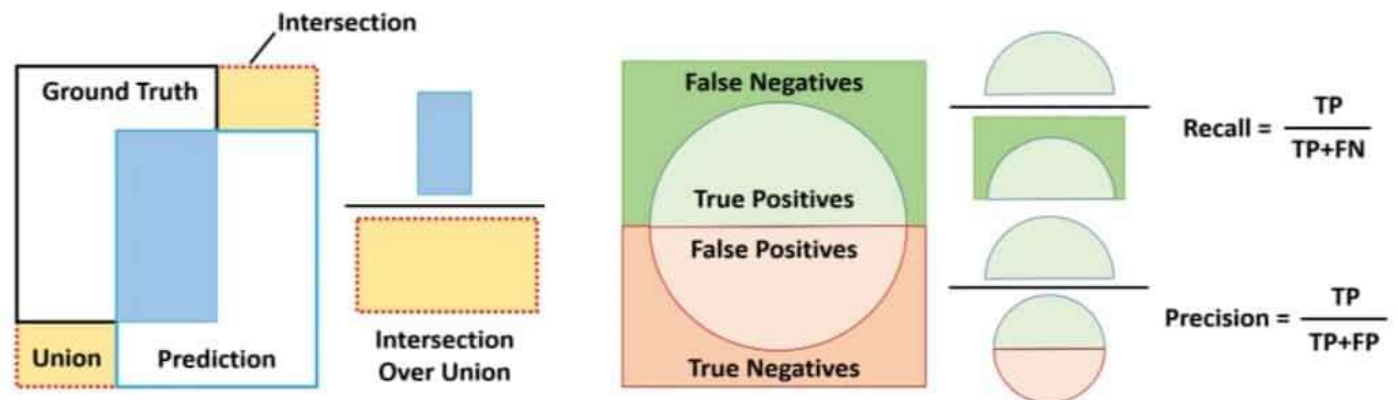


Figure 10: The architecture of modern object detectors can be described as the backbone, the neck, and the head.

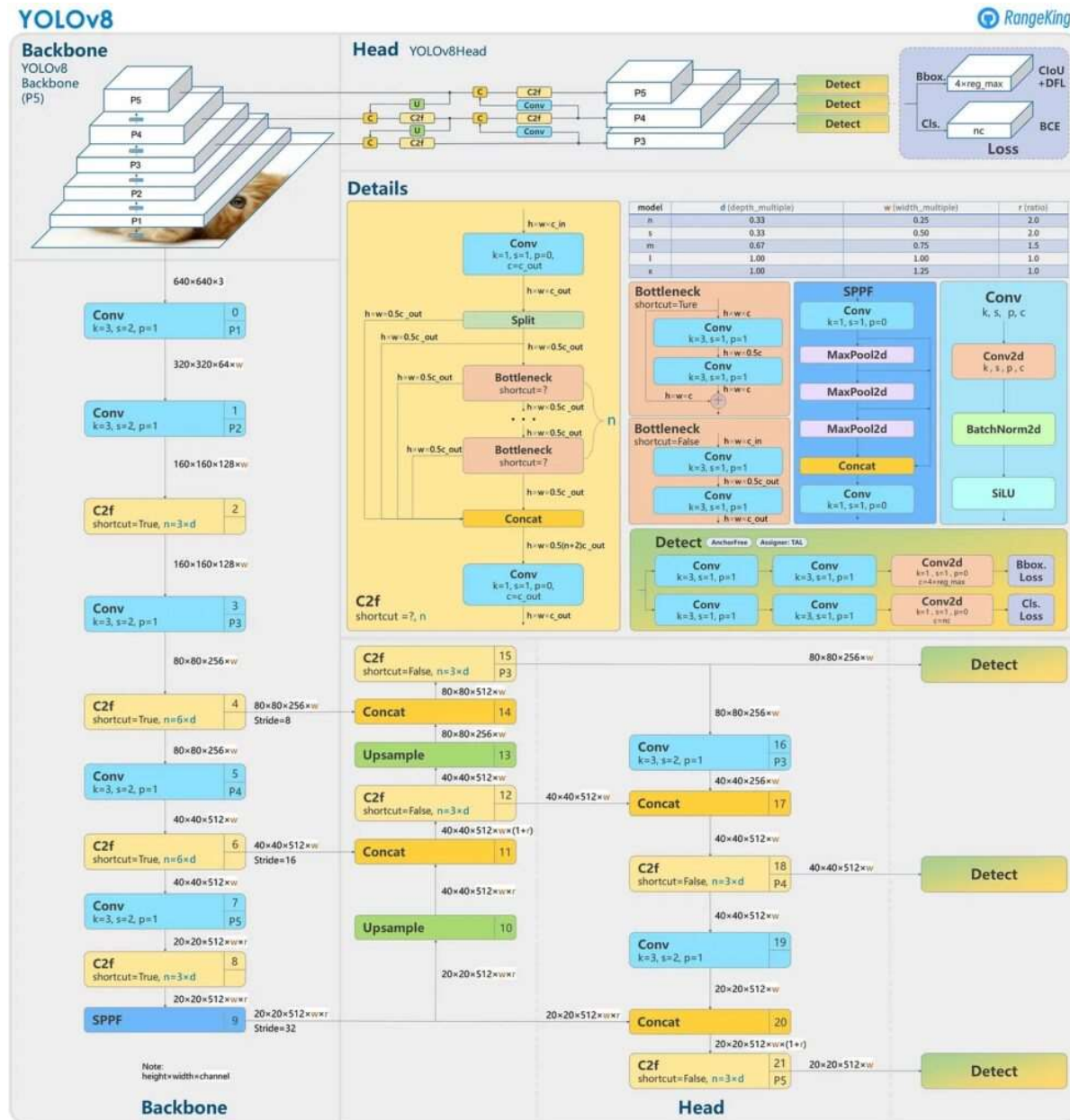
- Mean average précision (mAP)

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{|TP_c|}{|FP_c| + |TP_c|}$$





# Yolo v8



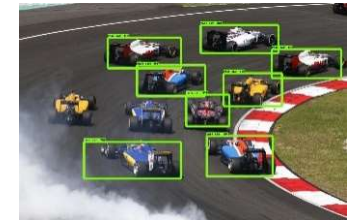
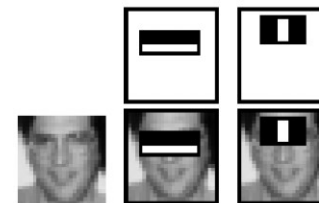
# YOLO par la pratique

- Yolo v1 serait un bon TP
- Utiliser YOLO ferait un bon projet dans une application de plus haut niveau
  - Spécialiser YOLO à vos données
    - Tracking de poissons, d'animaux, de ballons/joueurs, etc.
  - <https://pjreddie.com/darknet/yolo/>
- benchmark Roboflow 100 (RF100)
  - détection d'objets open source et crowdsourcé. Il comprend 100 ensembles de données, 7 domaines d'imagerie, 224 714 images et 829 étiquettes de classe avec plus de 11 170 heures d'étiquetage.

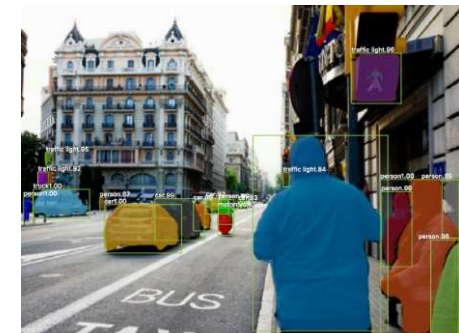
# Deep Learning a permis des avancées fortes en vision par ordinateur

- **Souvent des idées simples mais malignes marchent le mieux**

- Viola–Jones object detection 2004
- ...
- ...
- Yolo 2016
- ...



- Ces outils de visions peuvent être maintenant être
  - utilisés dans des applications de tous les jours (mobile, jeux vidéo, ...)
  - utilisés en recherche pour résoudre des problèmes d'une autre nature : Reconstruction 3D, MoCap, Extraction de textures, illumination, etc.



# POUR ALLER PLUS LOIN AVEC LES RÉSEAUX ET L'IMAGES

---

- Différents types de réseaux
  - CNN, RNN, LSTM, etc.
- Pour de l'apprentissage semi ou non supervisée
  - Autoencoder
  - FaceNet (clustering)
- Pour la génération
  - Autoencoder, GAN
- ...

# 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

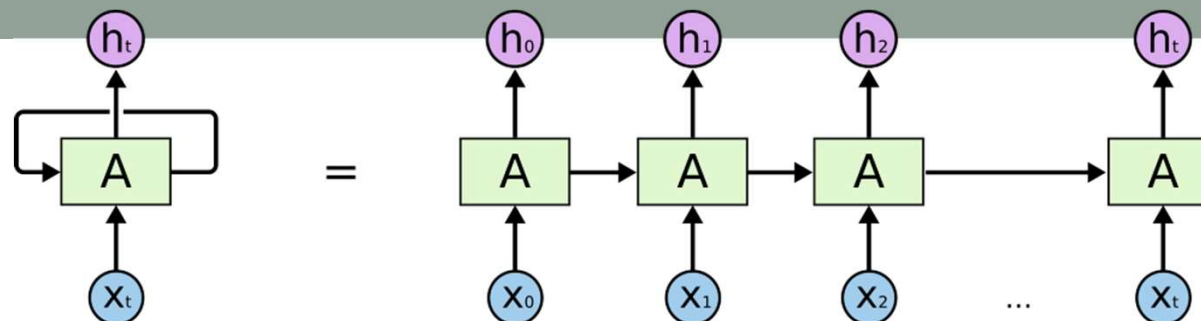
- Auto-encoder
- Construction de cluster : par exemple FaceNET

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.



# RNN

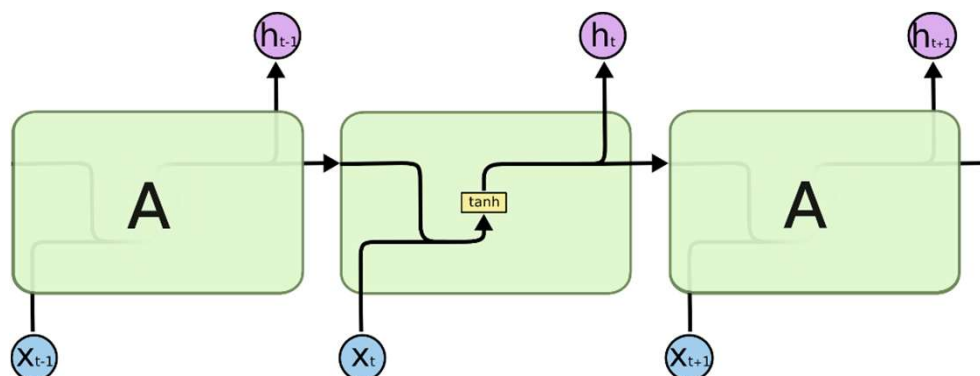


- Pour des données temporelles
  - Prédire le prochain mot
  - Composer de la musique
  - Reconnaître le langage parlé
  - Détection d'erreur dans une série d'évènements
  - Prédiction de la bourses, matchs/sport, etc.

→ Recurrent Neural Network, LSTM, Gated Recurrent Unit,

...

RNN simple



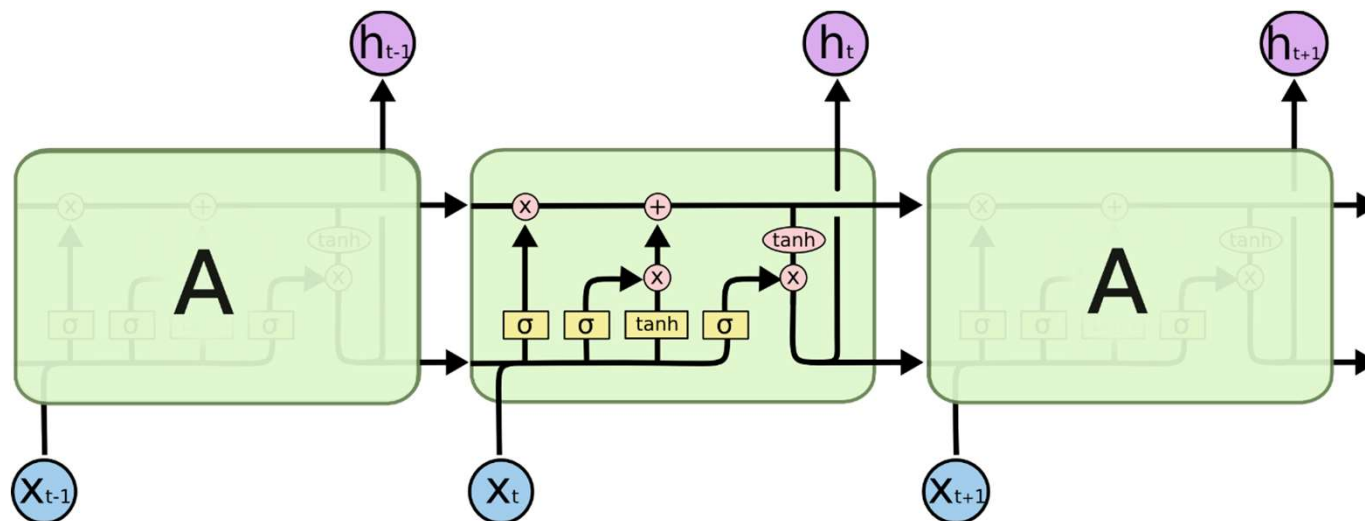
# LSTM

- Pour des données temporelles

→ **Long Short-Term Memory**

Par exemple

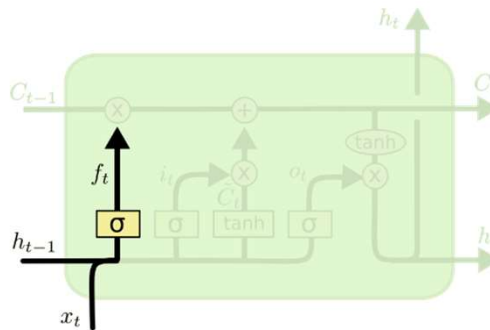
- 6 -> 7 -> 8 -> ?      On voudrait 9
- 2 -> 4 -> 8 -> ?      On voudrait 16
- Se baser sur 8 ne suffit pas
- A une mémoire courte et long terme
- **Apprend quand se souvenir et quand oublier**



# LSTM

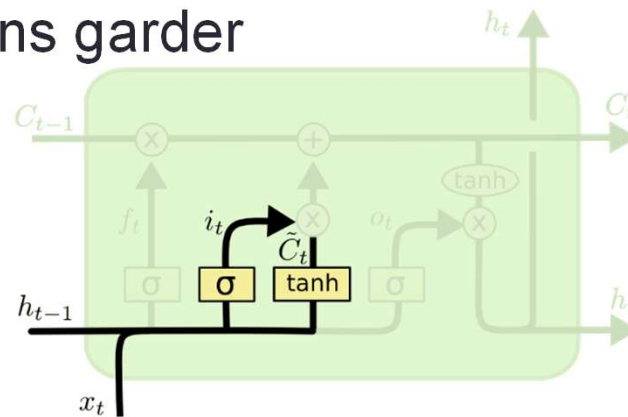
## Les couches

- Oublier ou garder: 0..1



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

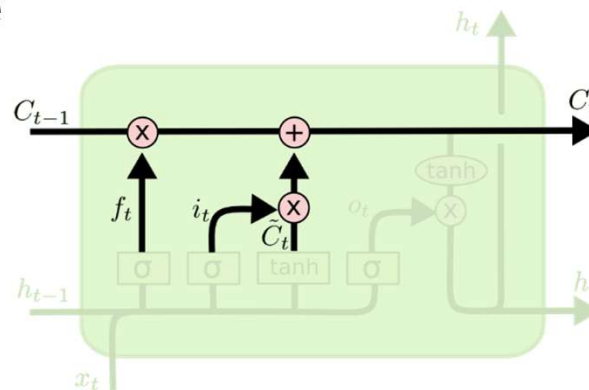
- Quelles informations garder



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

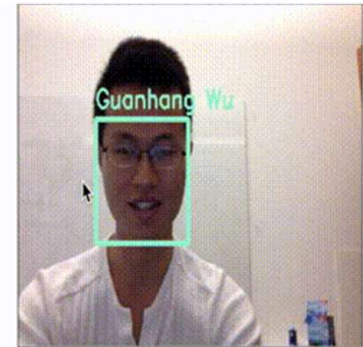
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- Produire la sortie



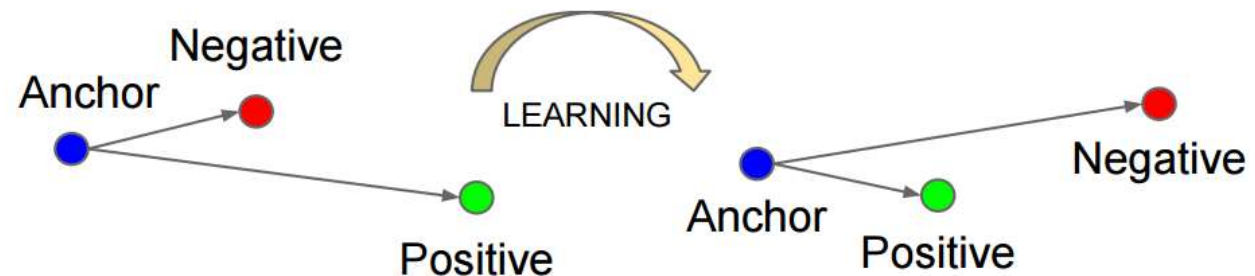
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

# Triplet LOSS (FaceNet)



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

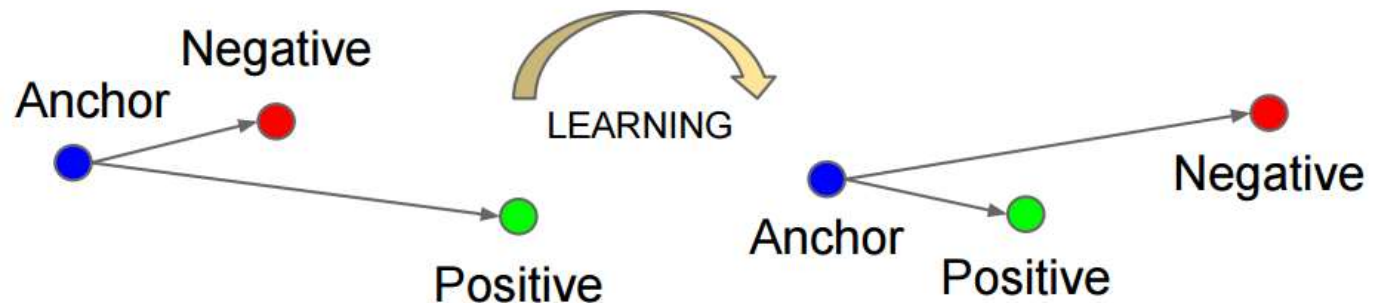
- Plus efficacement que CNN de base
- Apprend une représentation des données pour rapprocher les éléments ayant un rapport entre eux et éloigner les autres → **construire des clusters**



FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR 2015

<http://cmusatyalab.github.io/openface/>

# Triplet LOSS (FaceNet)



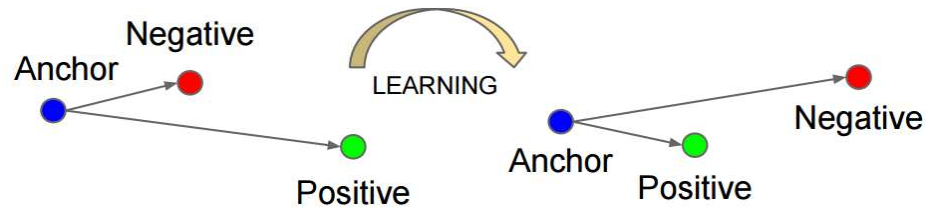
- Utilise un triplet
  - une instance “anchor”  $x$  : le visage d’une personne A
  - une instance positive  $x^+$  : le visage de la même personne A
  - une instance negative  $x^-$  : le visage d’une autre personne B
- $f(x)$  la representation de  $x$ , la fonction de coût

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

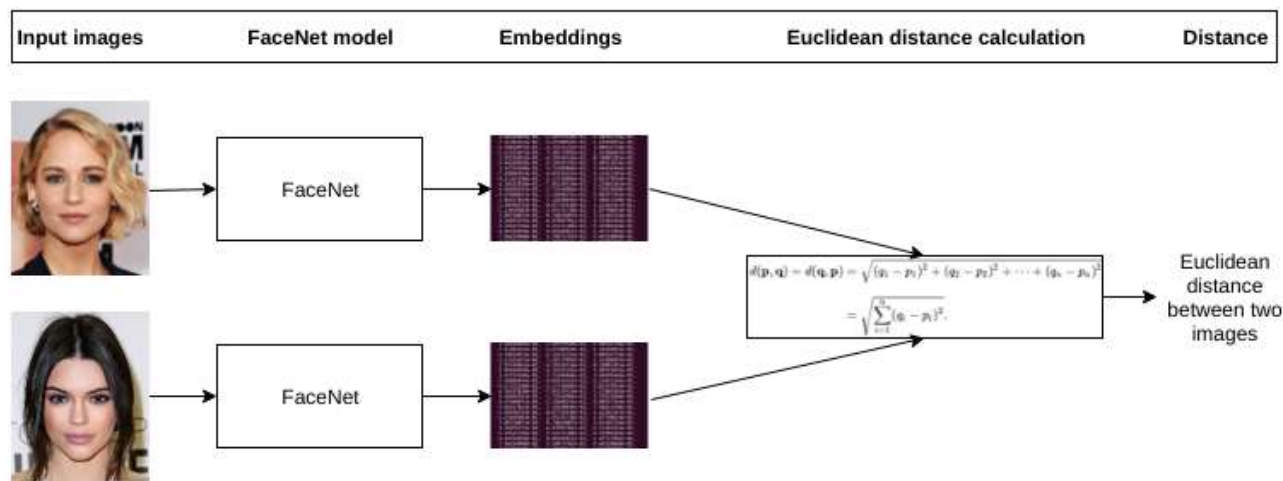
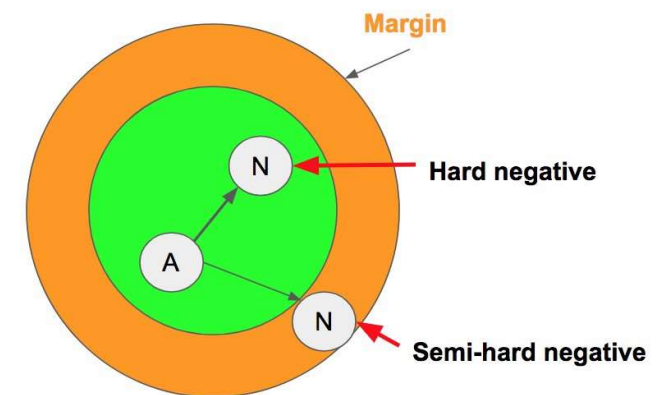
- Ignore le triplet quand  $x^+$  est déjà plus proche que  $x^-$
- Apprendre  $f(x)$  en utilisant la backpropagation



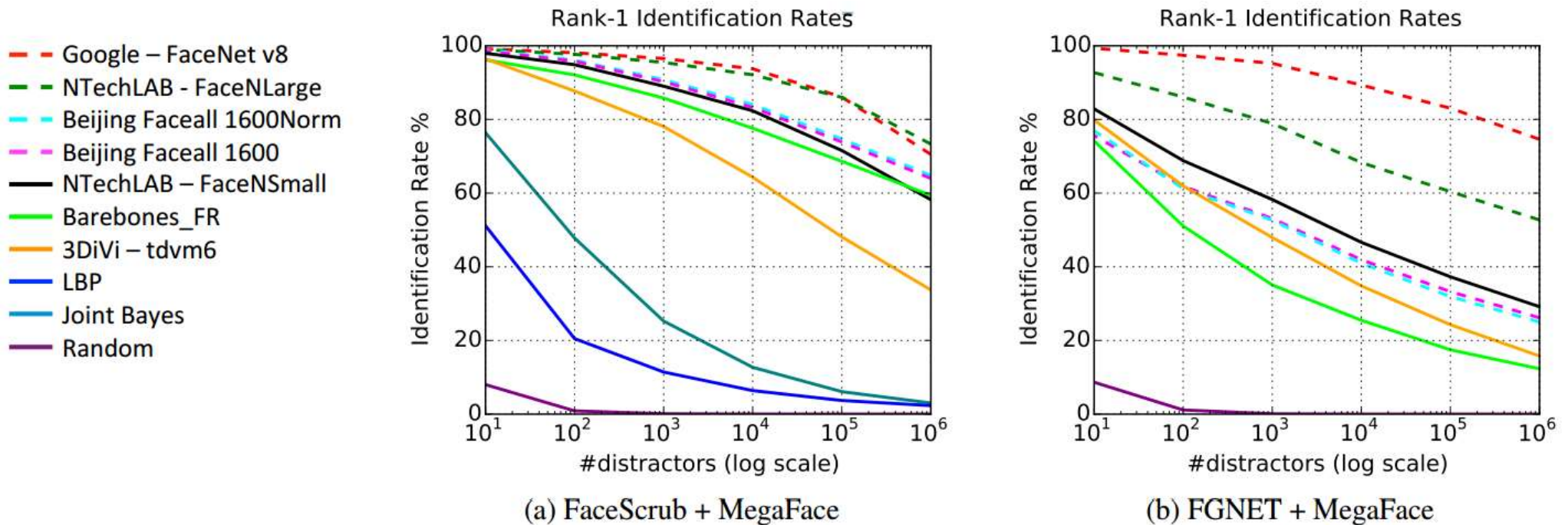
# TripletLoss



- Le nombre de triplet est gigantesque
  - $10^6$  de visage  $\Rightarrow 10^{18}$  triplets
  - $\rightarrow$  progression
  - Choisir des triplets semi difficiles
  - Choisir des triplets difficiles
- Produit une description d'un visage en 128 dimensions



# MegaFace Benchmark



# 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

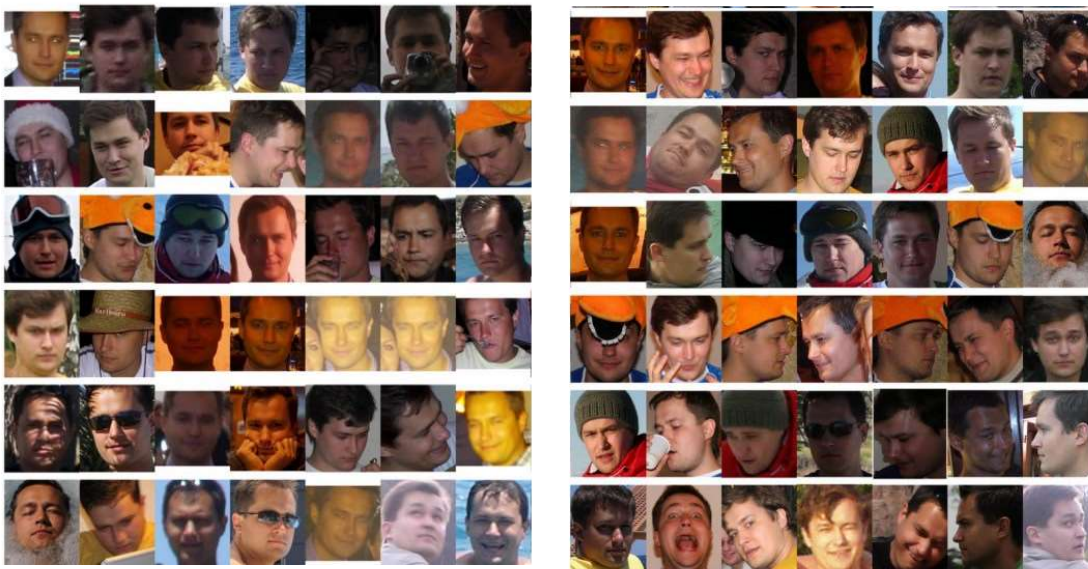


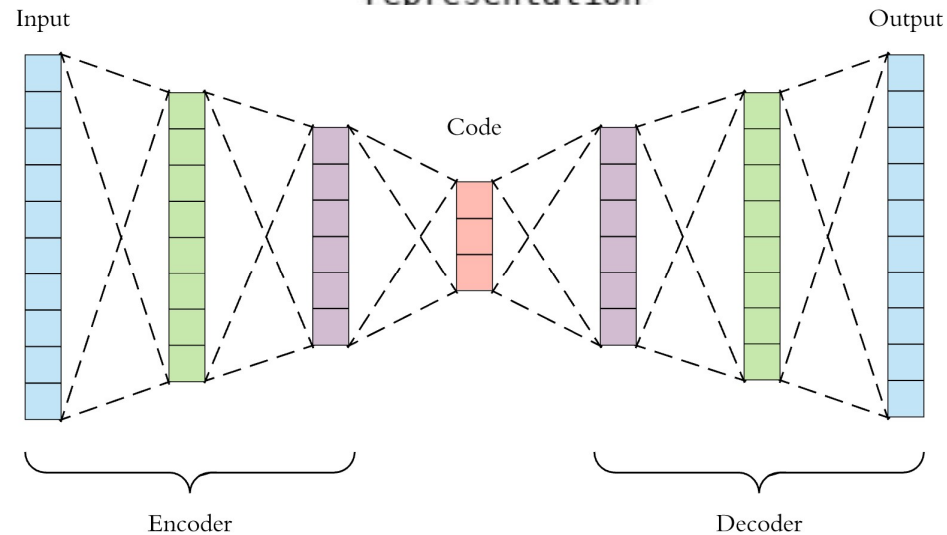
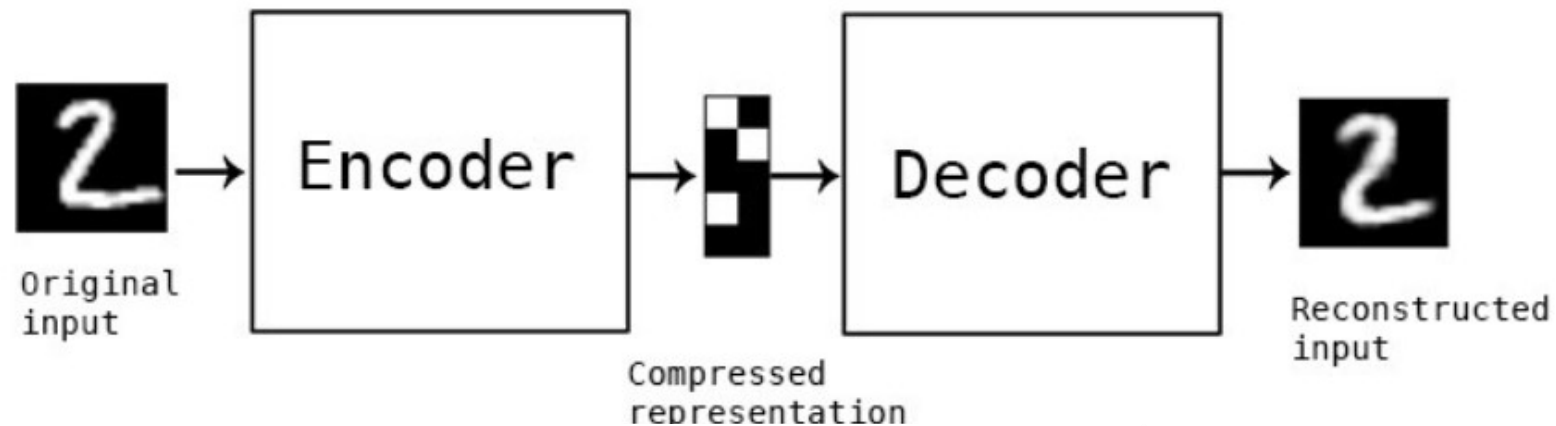
Figure 7. **Face Clustering.** Shown is an exemplar cluster for one user. All these images in the users personal photo collection were clustered together.



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.



# Auto Encoder (AE) : principe

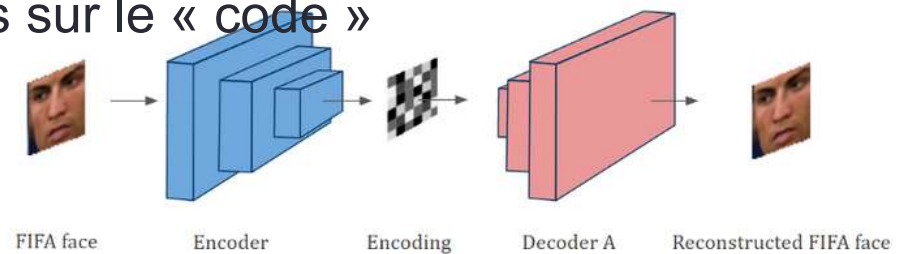
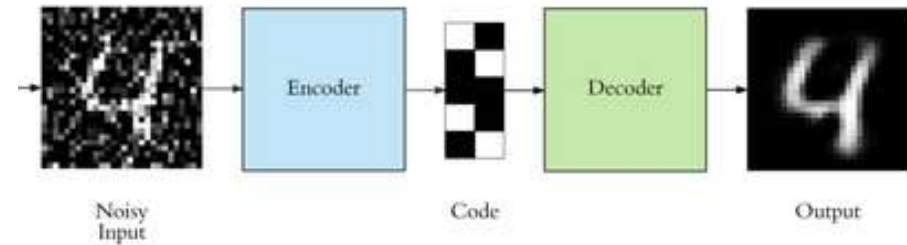


$$Obj = L(x, \hat{x})$$

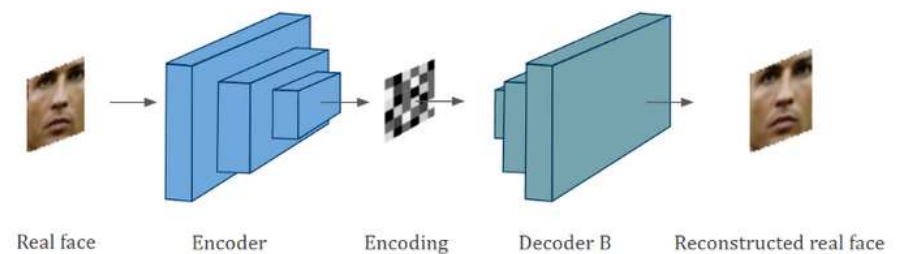
- Il existe de nombreux types d'autoencoder

# Auto Encoder (AE) : principe

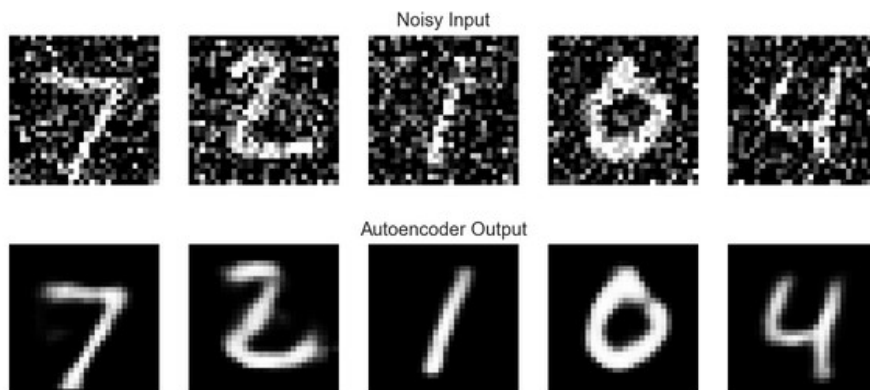
- A quoi ca sert ?
  - Débruitage
  - Compression
  - Super résolution
  - Code compact
    - Possibilité d'appliqué des traitements sur le « code »
    - Un peu le même esprit que la PCA
- Prémisse du Génératif



First autoencoder network learning from FIFA graphics



Second autoencoder network from learning actual pictures

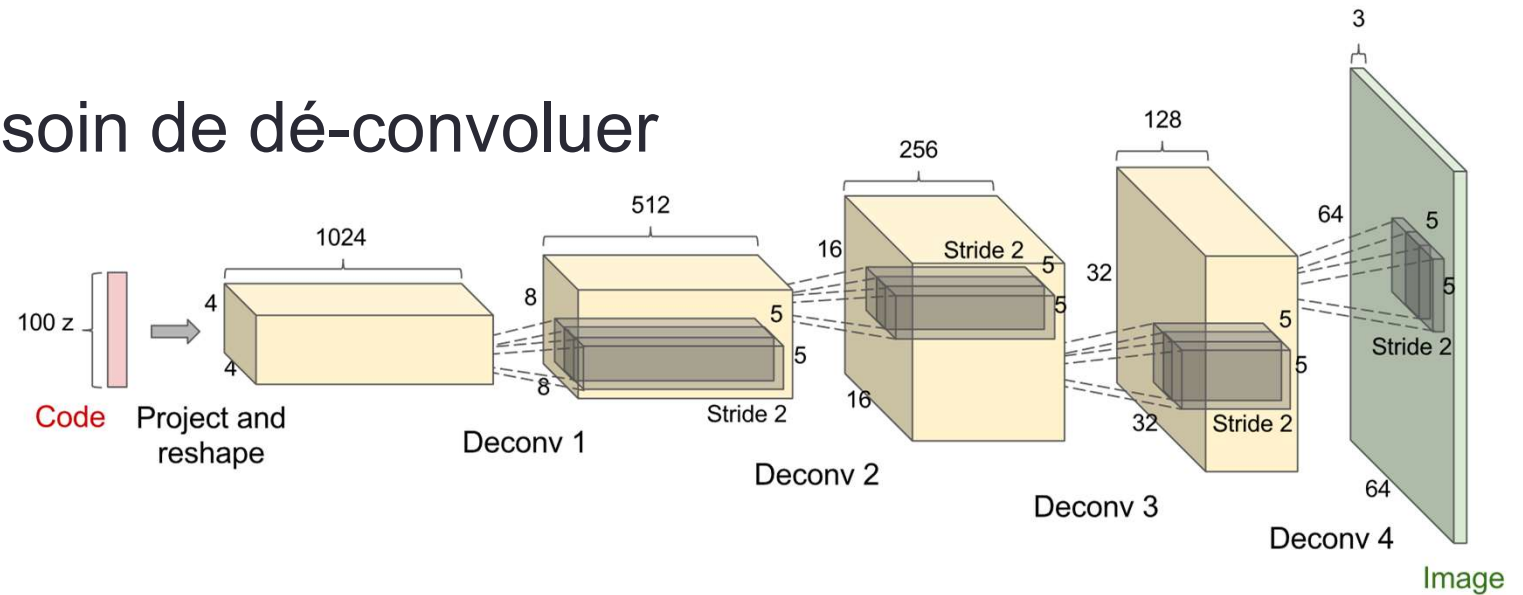


Amélioration visage de Ronaldo dans FIFA

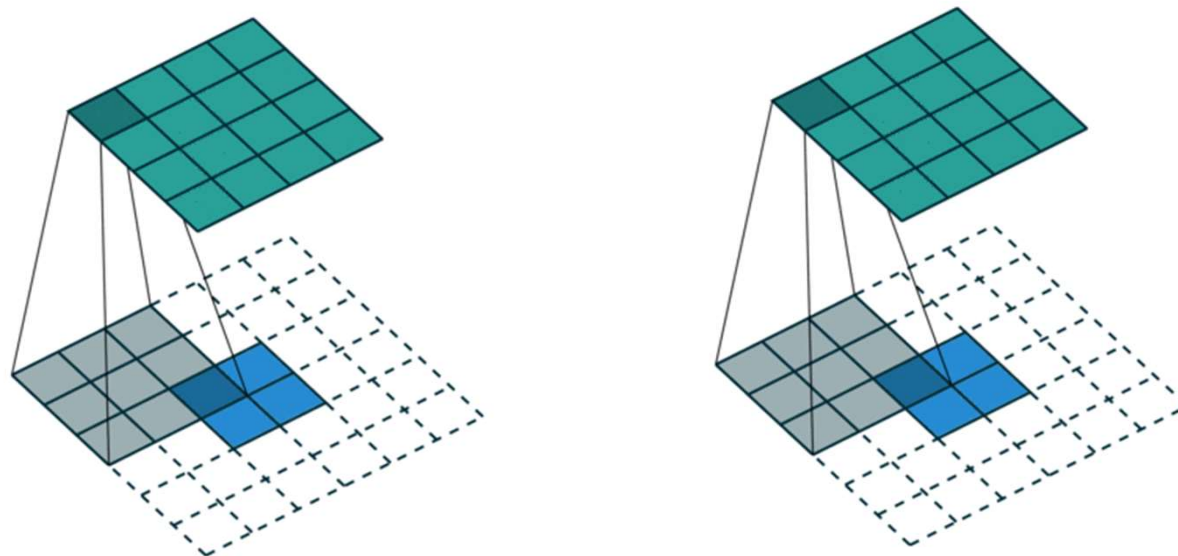


# Decode

Si image : besoin de dé-convoluer

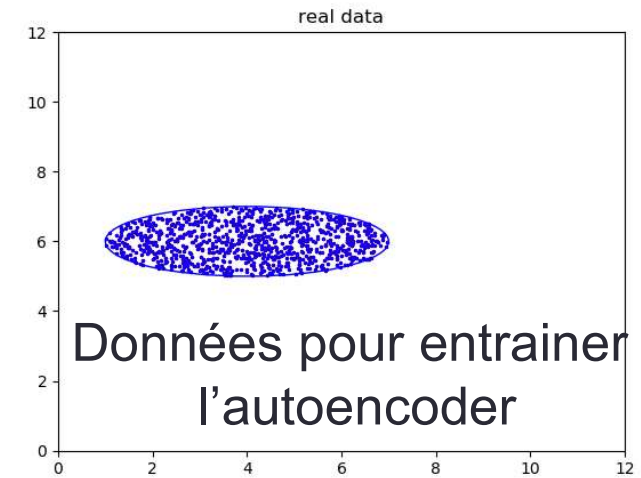


Devrait s'appeler plutôt transposed convolutional layer

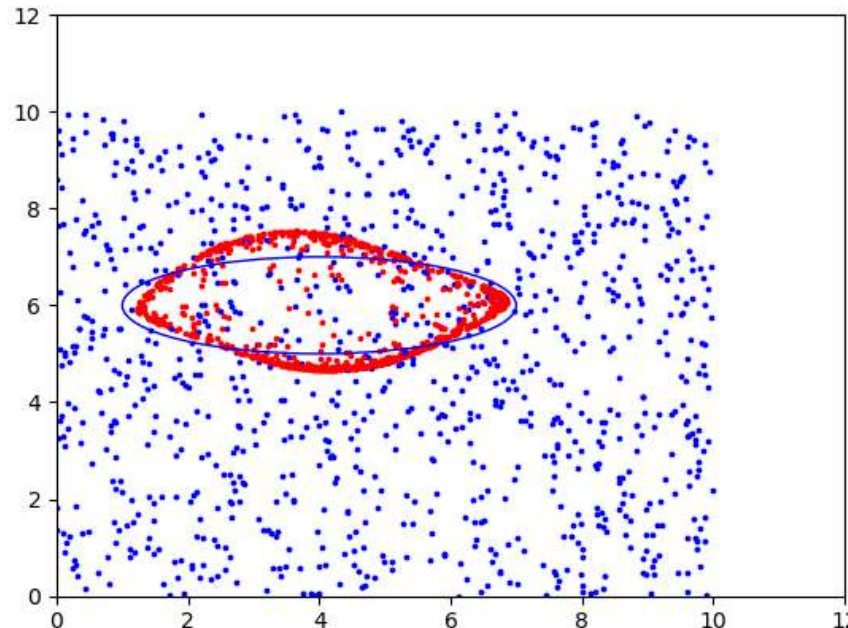


# Auto-encoder

- 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 sa

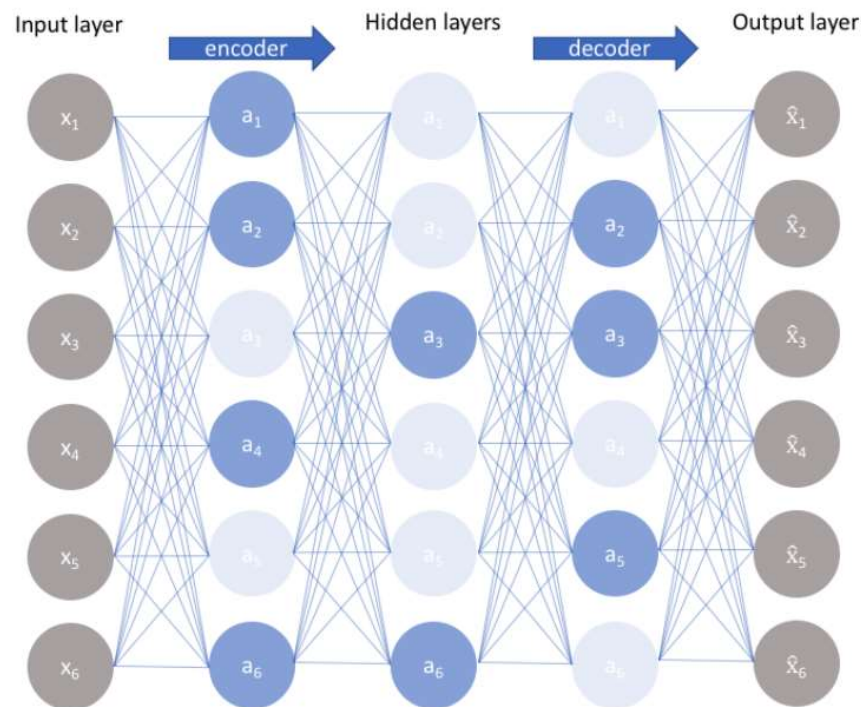


# Sparse Auto Encoder

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 » et « compact »



Sparse Autoencoder

Distance entre input et output

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

Favorise la sparsity

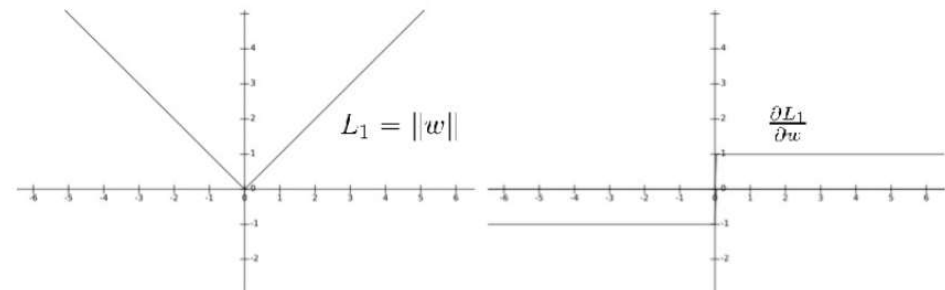
# Sparse Auto Encoder : norme L1 / L2

- Utiliser la norme L1 dans la fonction de coût

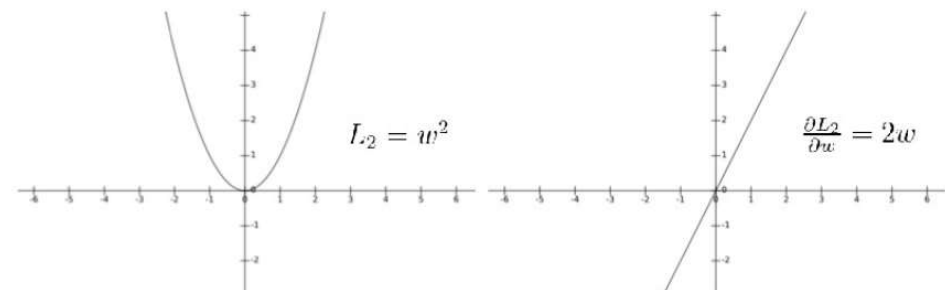
$$Obj = L(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}|$$

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

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



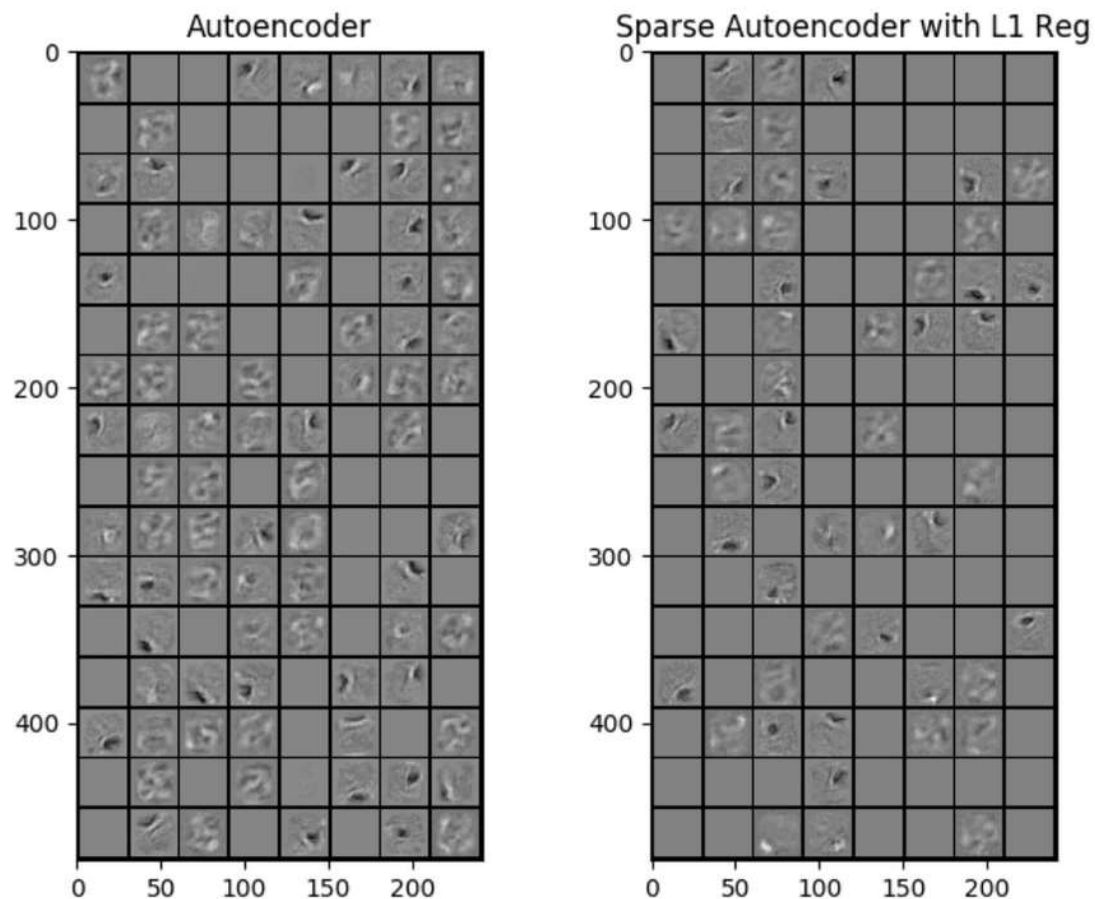
L1 regularization and its derivative



L2 regularization and its derivative

# Sparse Auto Encoder : norme L1

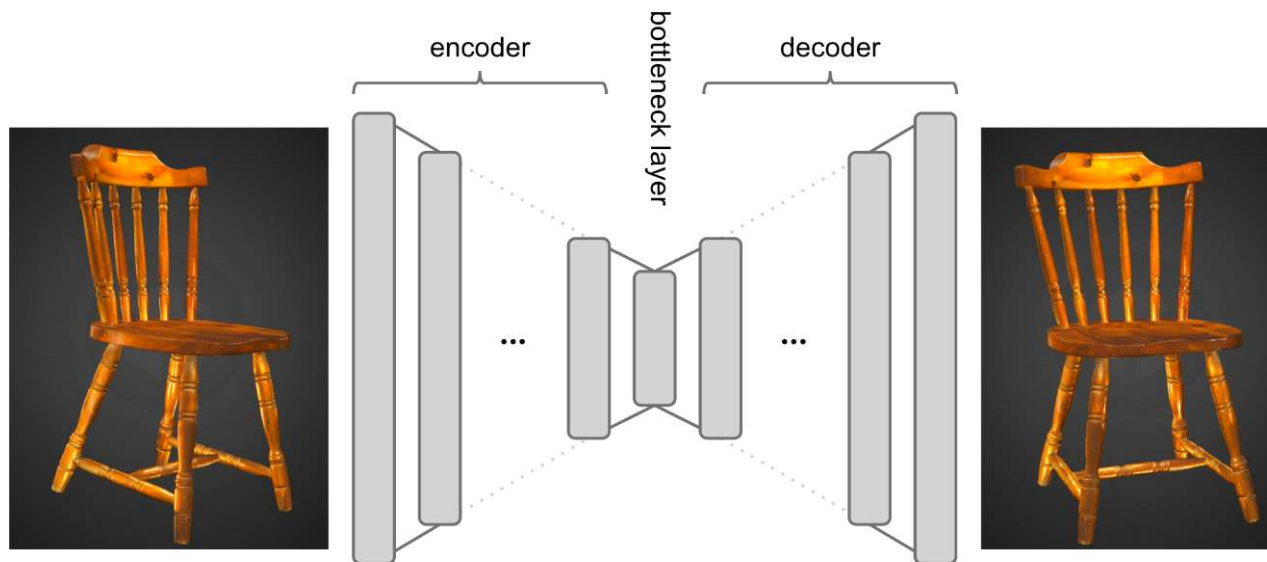
- Moins de neurones activés → les neurones utiles plus efficaces → représentation plus « intelligente »/ « efficace »





# Auto-encoder ...

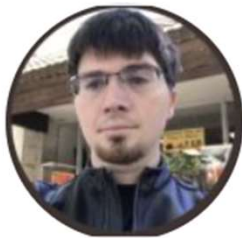
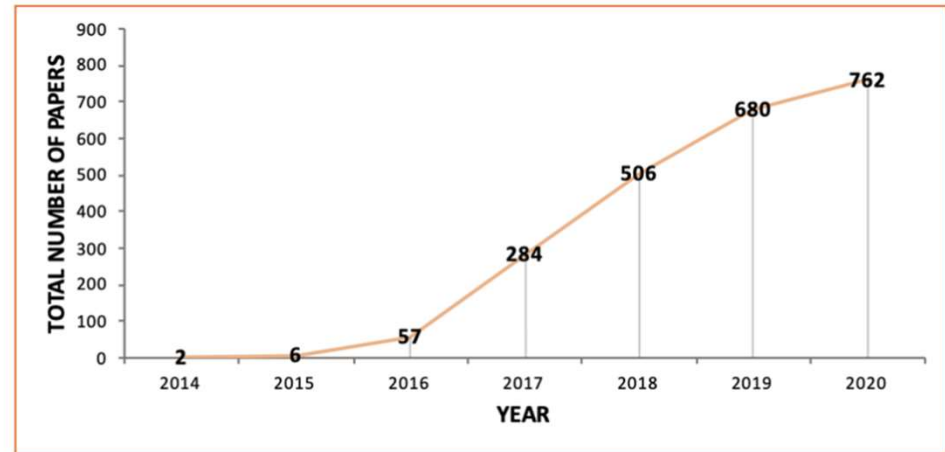
- De nombreux auto encodeur existent
  - Variational AE, ...
  - Inclus dans un GAN
  - Possibilité de les entrainer sur autres choses que des images
    - Maillage 3D
    - Animation ...
- Encore un fois un problème vaste ... mais un outils puissant



# GAN

## GENERATIVE ADVERSERIAL NETWORK

---



Ian Goodfellow

DeepMind

Verified email at deepmind.com - [Homepage](#)

[Deep Learning](#)

TITLE

CITED BY

[Generative adversarial networks](#)

67024

I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, ...  
Advances in neural information processing systems 27

# Réseau de neurones et génération : GAN

- Generative Adversarial Networks (2014),
  - Goodfellow et al.
  - Plusieurs milliers de variantes
- Les applications « cools » des GAN
  - Génération de visages
  - Personne avec poses différentes
  - Transfert de texture
  - Super résolution
  - Texte vers images
  - ...



DCGAN  
11/2015



EBGAN-PT  
9/2016



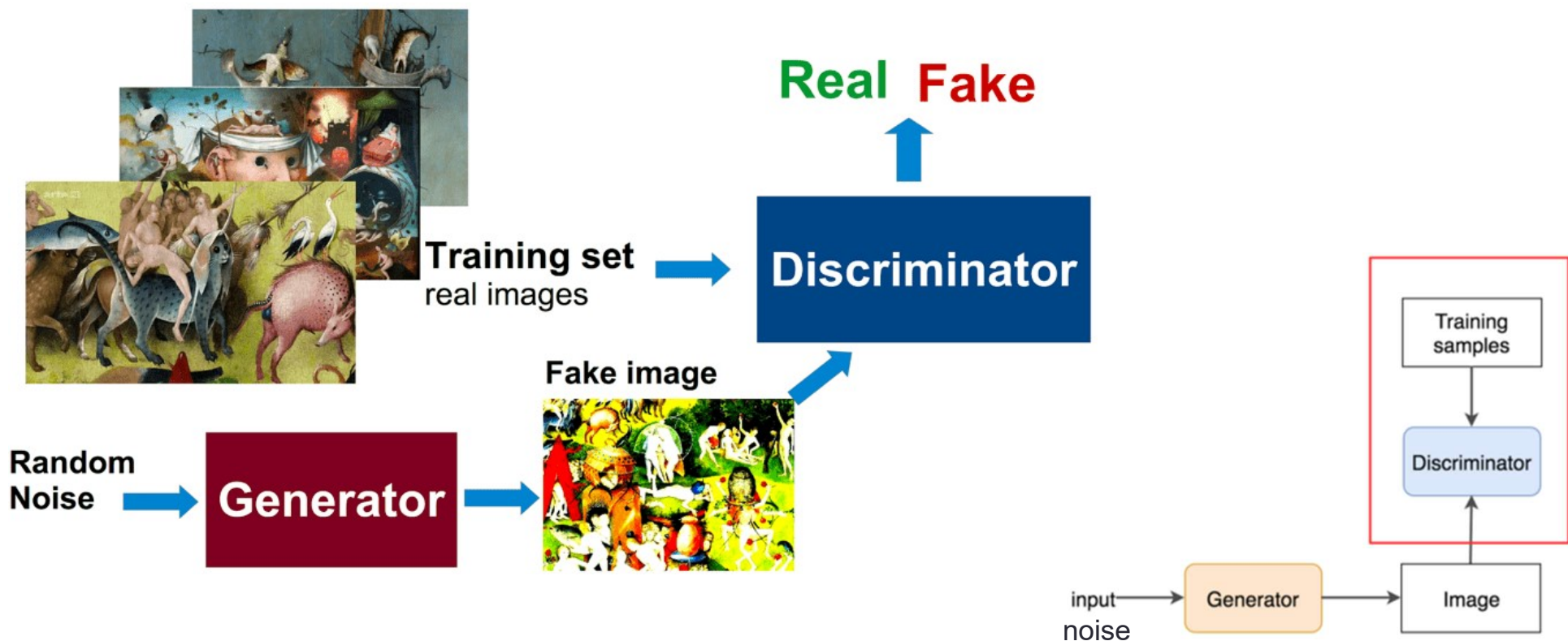
BEGAN  
3/2017  
128 x 128



Progressive GAN  
10/2017  
1024 x 1024

# Réseau de neurones et génération : GAN

- GAN = Generative Adversarial Networks
  - + Fonctionne avec une quantité de données moindre car en génère
  - Instable



# GAN

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

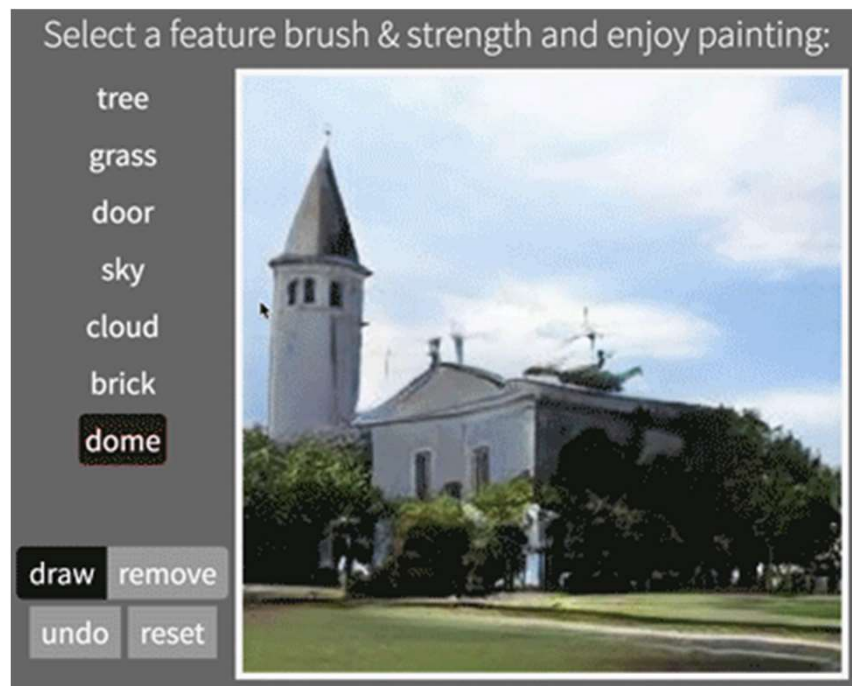


Figure 7: Generated samples



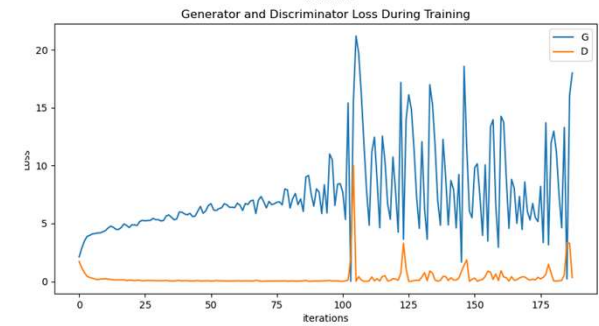
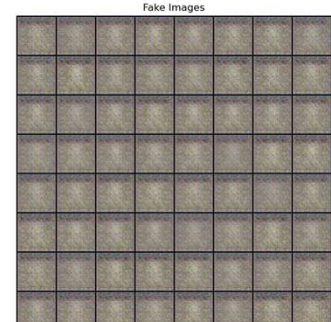
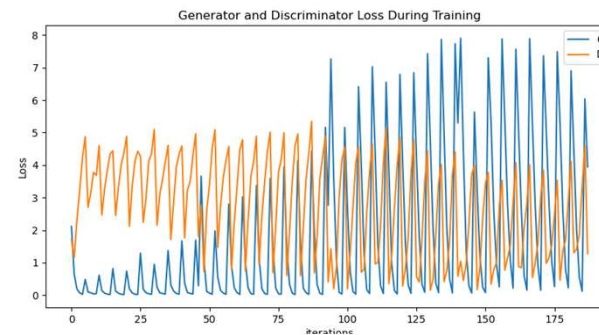
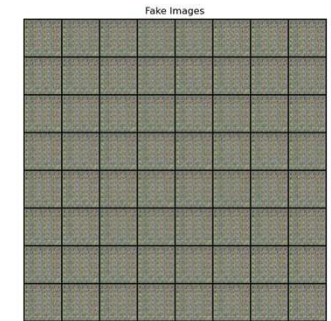
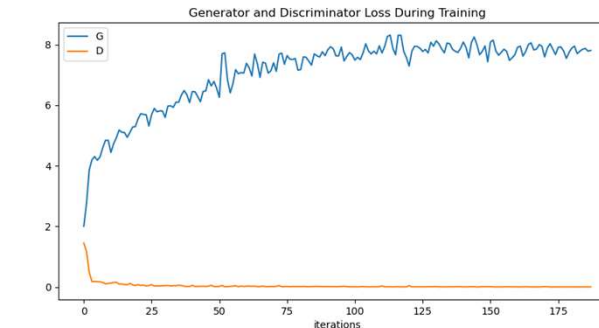
# GAN : les problèmes

Entraîner un GAN n'est pas simple

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

# GAN : des problèmes

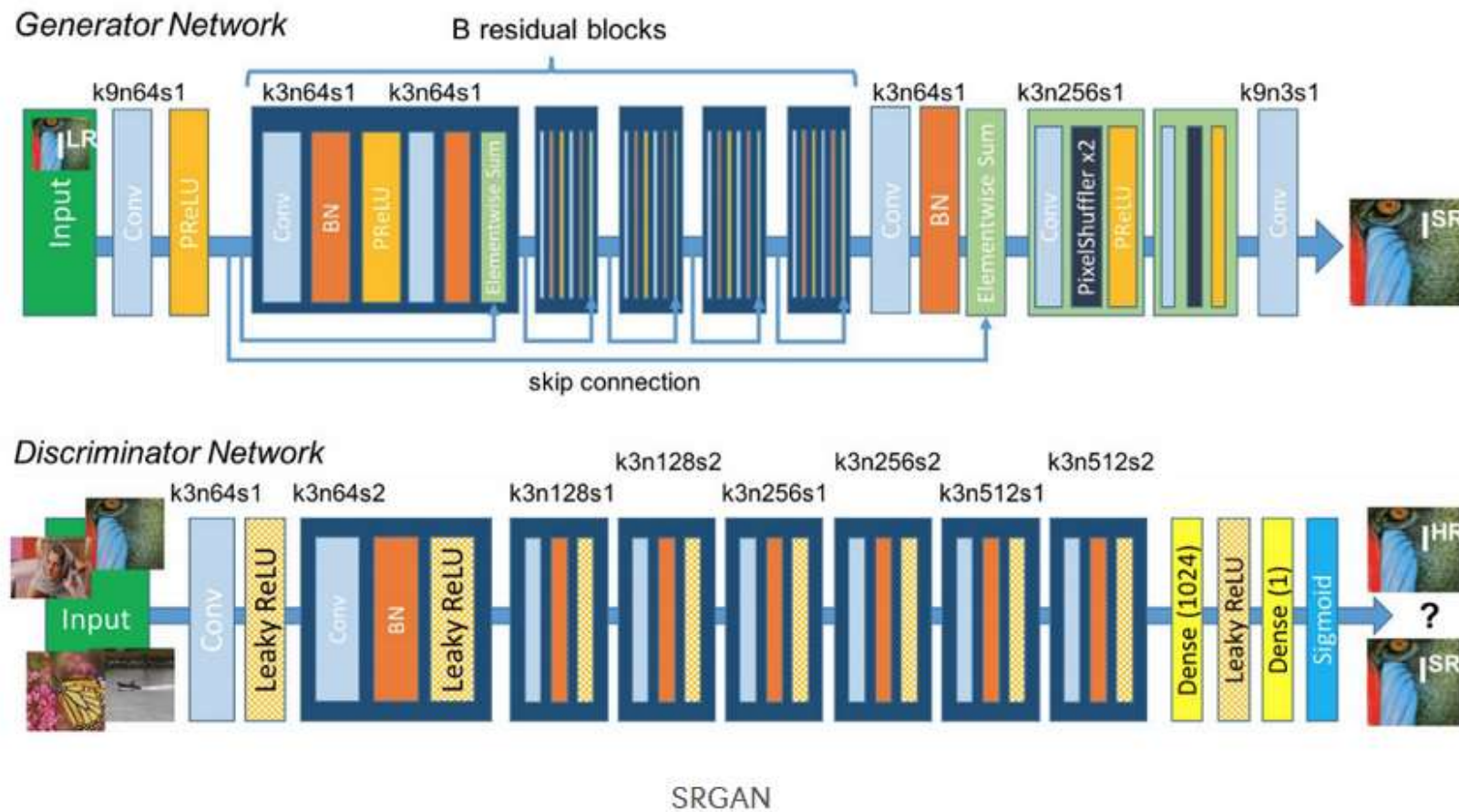
- Effondrement du gradient
- Non convergence
- Solution
  - Favoriser le générateur
  - Réduire la taille de l'espace latent en entrée
    - Ici passer de 100 à 16



après 4 epochs

# Un exemple de GAN : SRGAN

- Super résolution GAN





# Un exemple de GAN : SRGAN

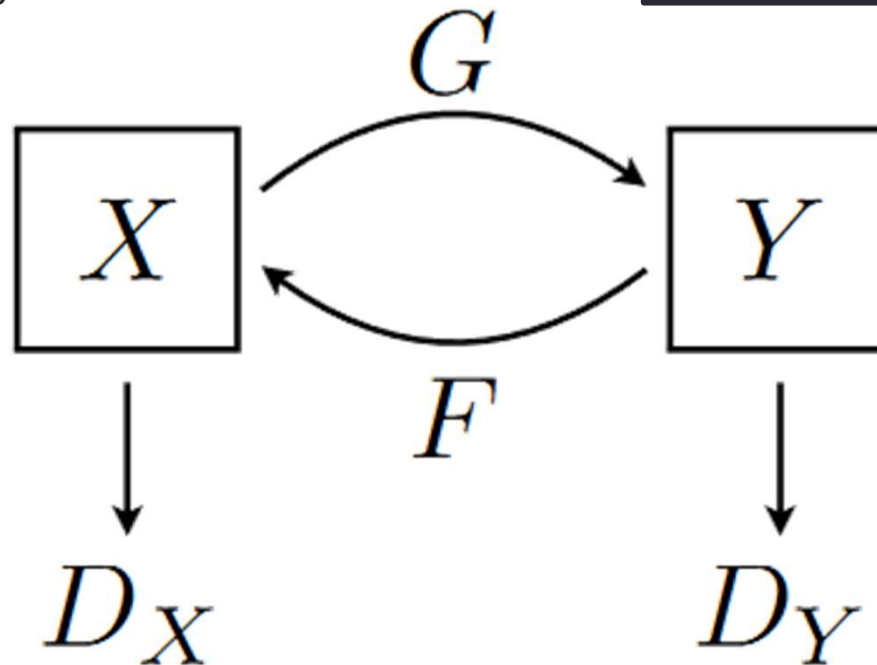


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

SRGAN

# Transfert entre distributions : CycleGAN

Une distribution  $X$   
Par exemple des photos réelles



GAN de  $X$  vers  $Y$  :

- Générateur  $G$
- Discriminateur  $D_Y$



Une distribution  $Y$   
Par exemple des peintures de Monet

GAN de  $Y$  vers  $X$  :

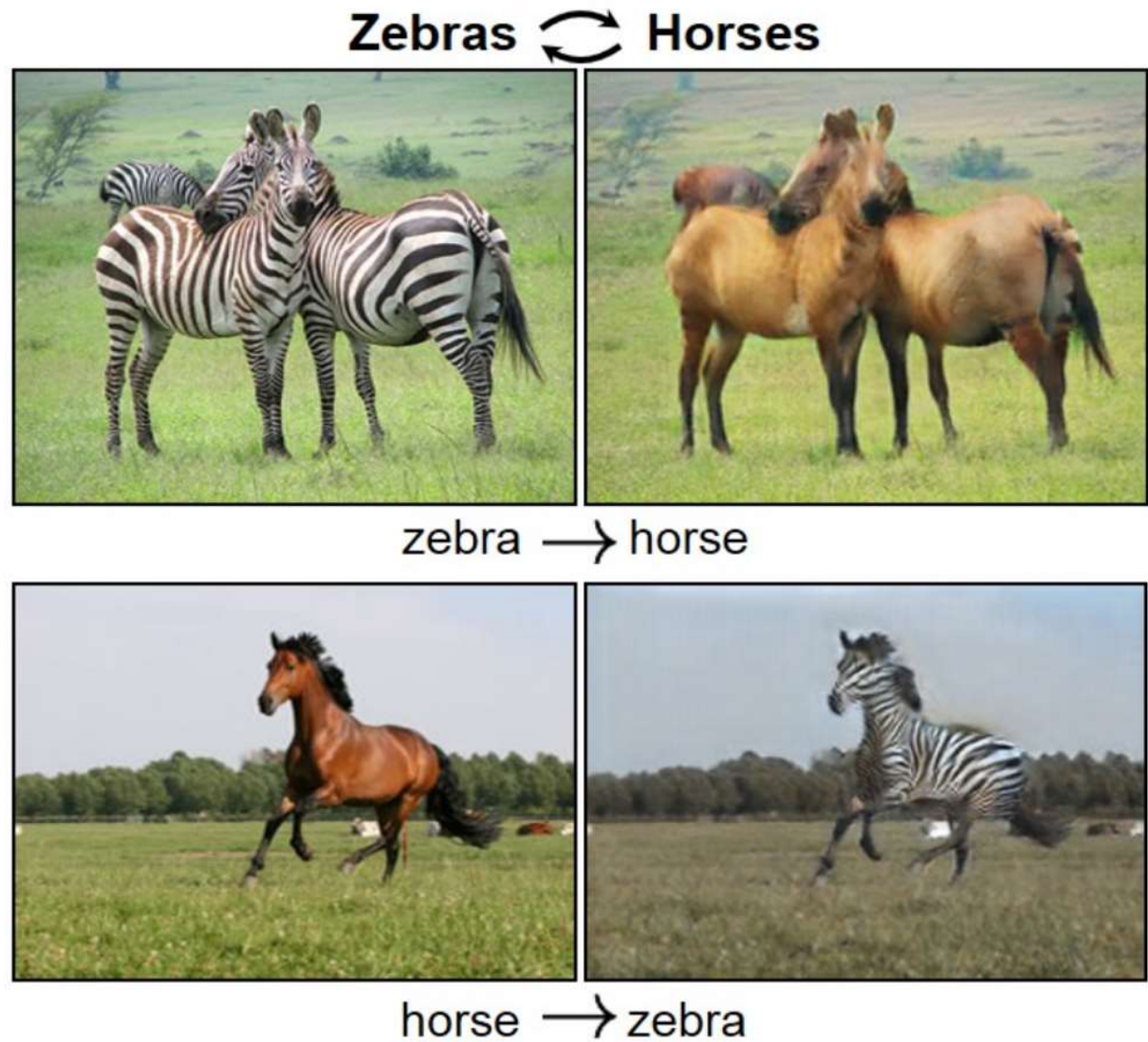
- Générateur  $F$
- Discriminateur  $D_X$



# Cycle GAN

- Distribution

- Cheval
- Zèbre

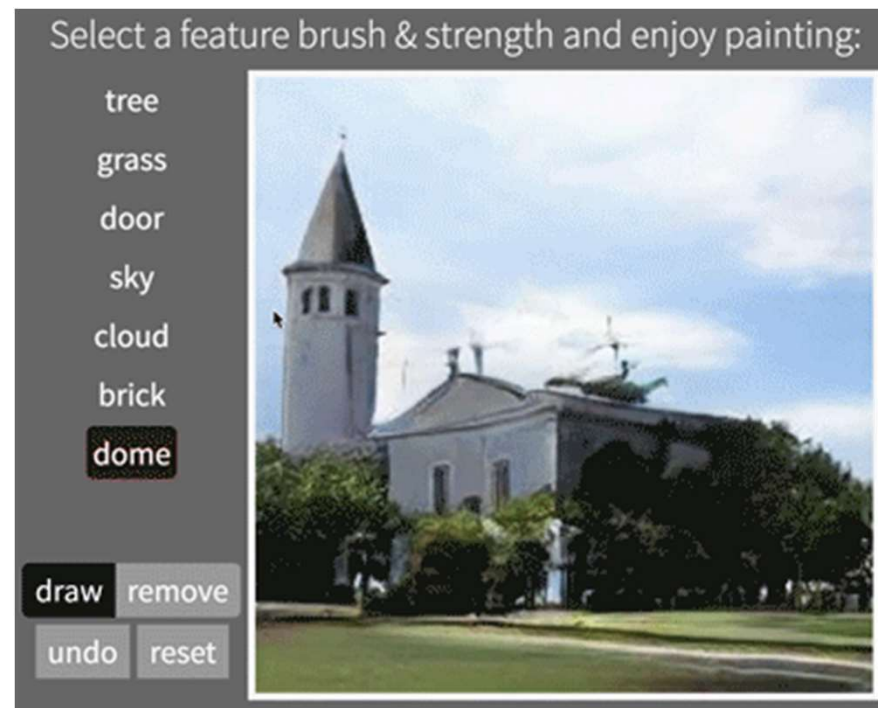


# GAN : un article à lire

GAN Dissection: Visualizing and Understanding Generative Adversarial Networks

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum<sup>1</sup>, William T. Freeman, Antonio Torralba

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



# GAN : édition

- Pour éditer il faut un espace latent avec
  - des informations uniques pour chaque dimension. De cette manière, l'espace latent sera démêlé (**disentangled**) et le générateur pourra effectuer toutes les modifications souhaitées sur l'image.
  - Mauvais exemple : raccourcir les poils ... change la morphologie
  - Construire un espace latent « bon » est souvent difficile
    - un 2<sup>e</sup> réseau « comprend », « se déplace » dans l'espace latent du GAN



raccourcir les poils



Exemple de ce qu'on ne veut pas



# GAN : modification d'un visage

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

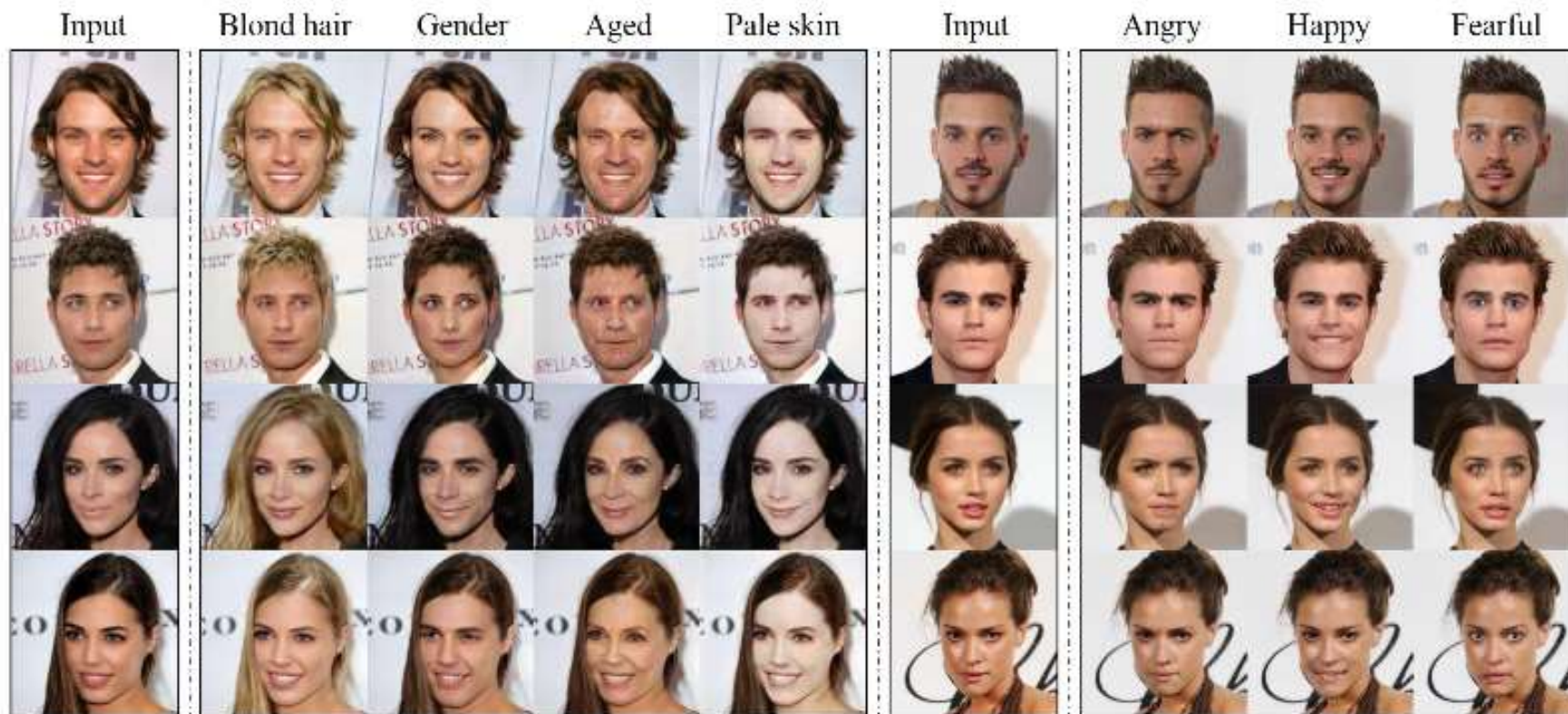


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator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

# GAN : modification d'un visage

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

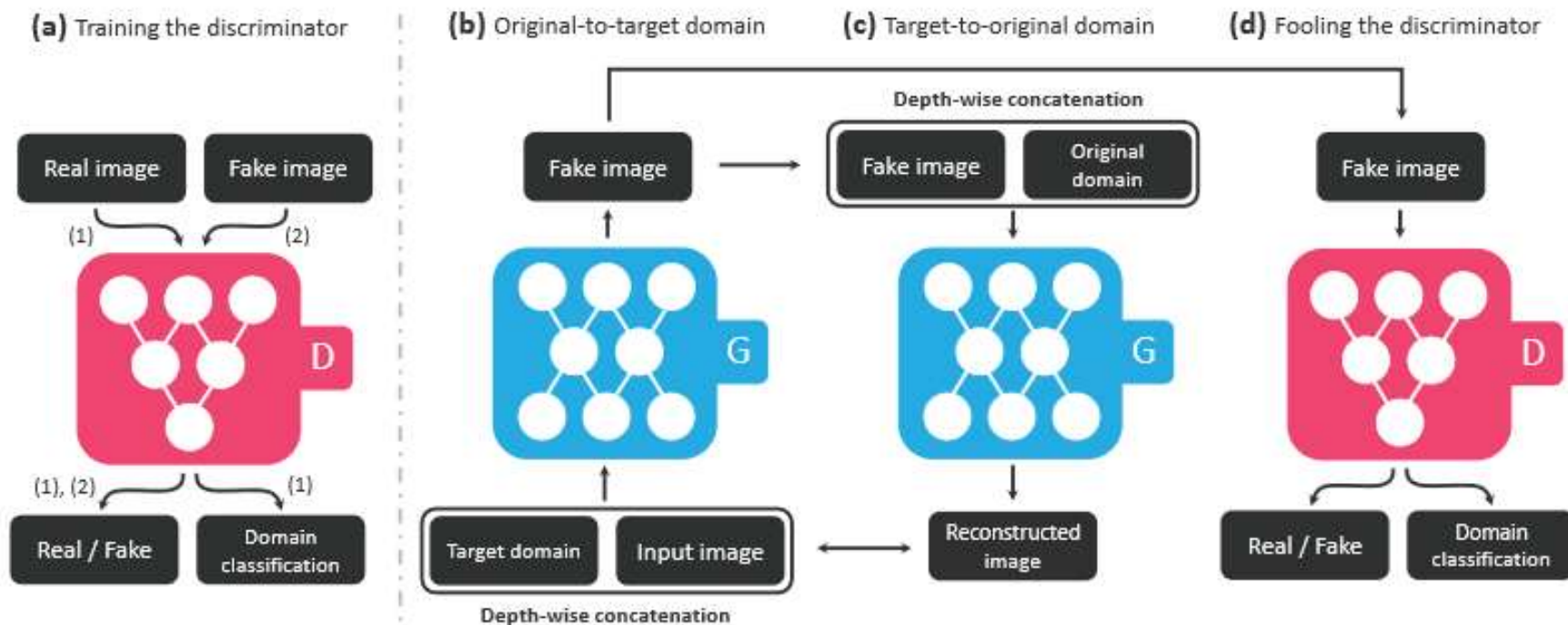
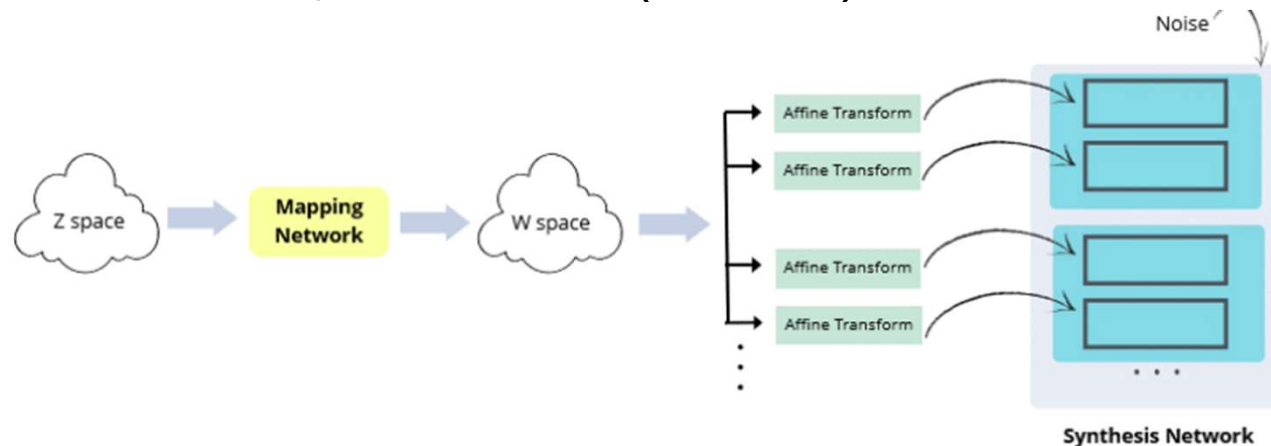


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator  $D$  and a generator  $G$ . (a)  $D$  learns to distinguish between real and fake images and classify the real images to its corresponding domain. (b)  $G$  takes in as input both the image and target domain label and generates 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  $D$ .



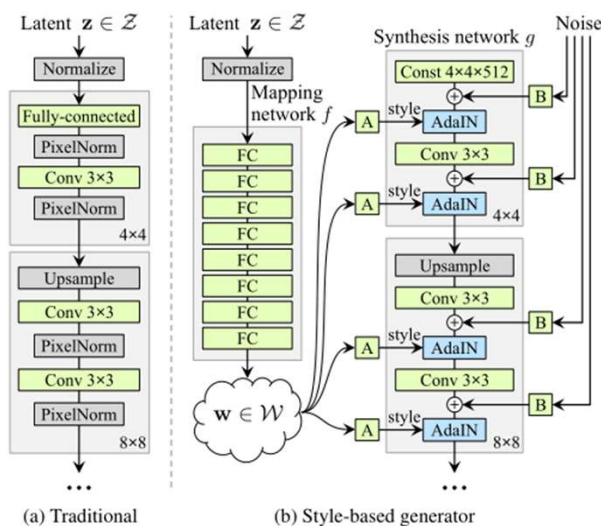
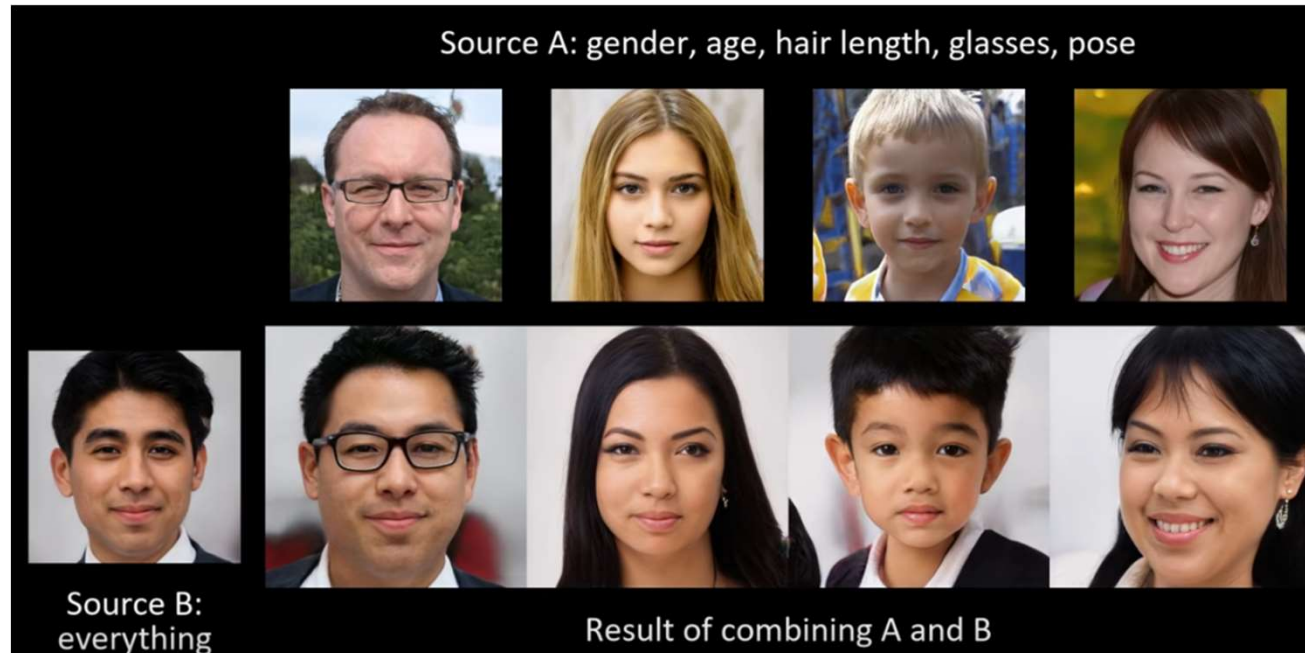
# StyleGAN

- A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019, Karras et al (NVIDIA)



- Comment savoir si l'espace  $W$  présente moins d'intrication que  $Z$  (est plus démêlé) ? ➔ nouvelle métrique
  - **Longueur du chemin perceptif** : mesure le degré de modification de l'image lors de l'interpolation. Des modifications « douces donnent » de meilleurs résultats.
  - **Séparabilité linéaire** : favoriser que deux points de l'espace latent correspondant à deux classes d'images sont séparés via un hyperplan ➔ calcul de l'entropie conditionnelle afin de déterminer la quantité d'informations nécessaires pour classer avec précision ces points latents.

# StyleGAN

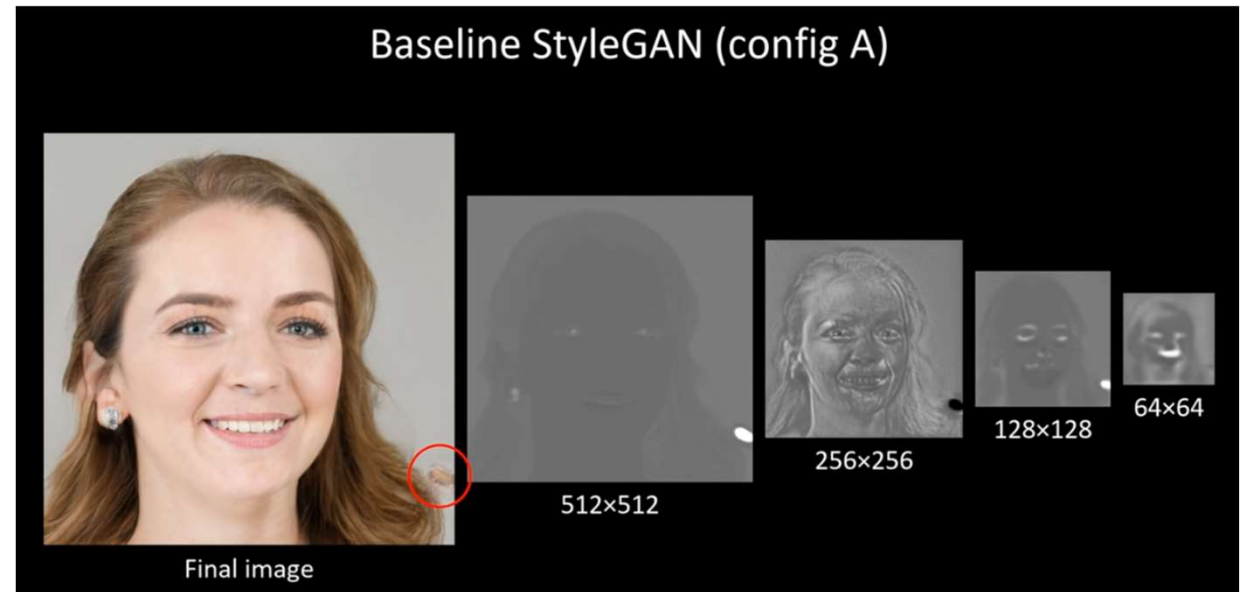


Our generator thinks of an image as a collection of “styles”, where each style controls the effects at a particular scale

- Coarse styles → pose, hair, face shape
- Middle styles → facial features, eyes
- Fine styles → color scheme

# StyleGAN2

- Pipeline revisité
  - Moins de défauts
  - Moins de données (still big)



## StyleGAN2



# GAN

# VS

# AE

= besoin d'images mais pas de label

< Souvent le GAN offre des résultats plus réalistes car le AE sont limités à observer uniquement les données

> Une données initiale passe dans l'espace latent avec l'encodeur  
> Facilité d'entraînement

- Les GAN sont utilisés pour la génération d'images, le transfert de style, la traduction d'image à image, la super-résolution et la génération de d'images réalistes.
- Les autoencodeurs trouvent des applications dans le débruitage d'images, la réduction de la dimensionnalité, la détection d'anomalies et l'apprentissage de caractéristiques.

# POSES ET IMAGES

---

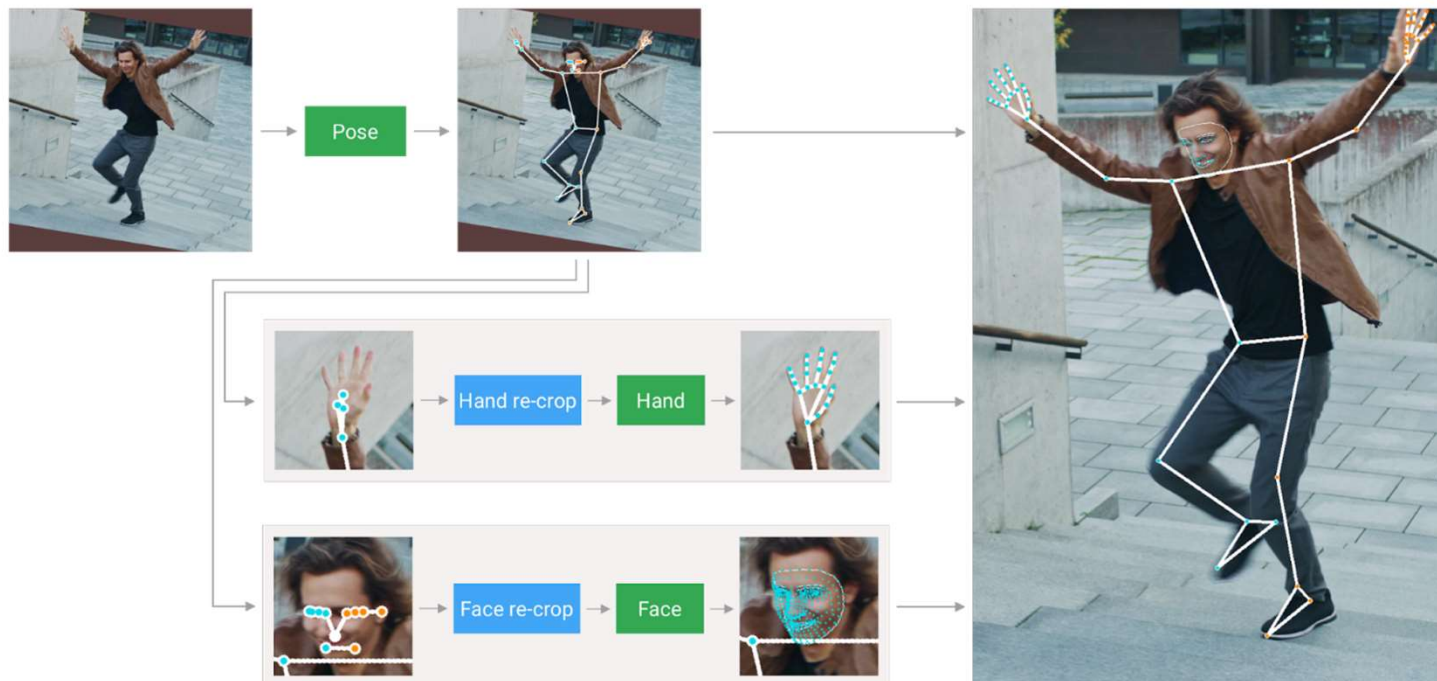
Le TP



# La capture

La capture monoculaire de poses en temps réel

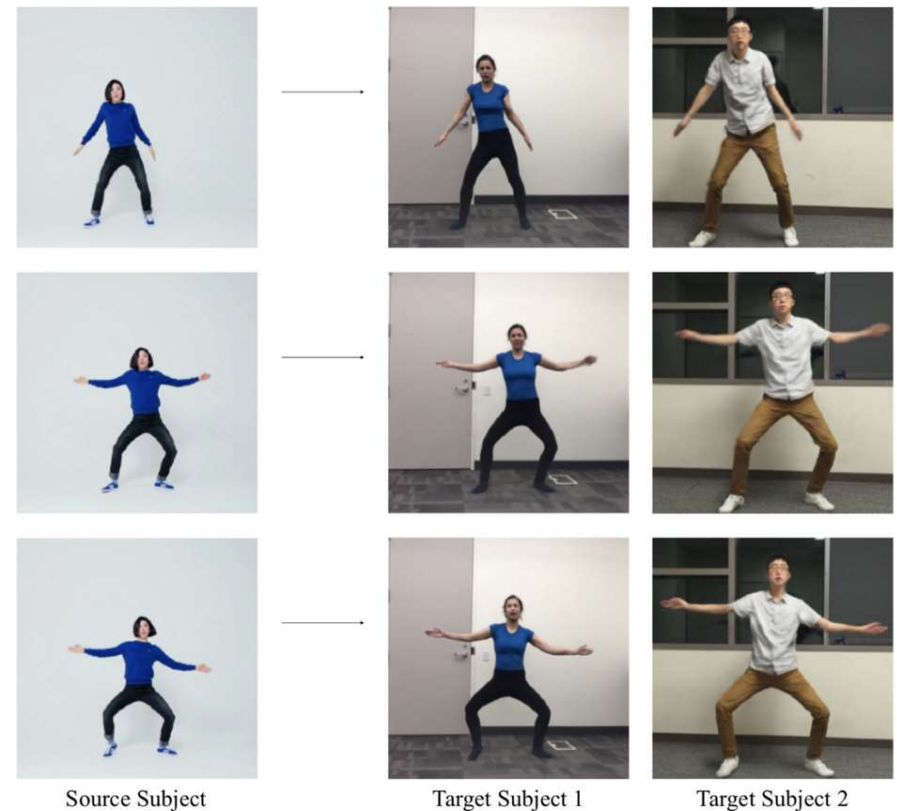
- avec mediapipe, OpenPose, etc.
- ouvre de nombreuses possibilités



# Everybody Dance Now

- But : transfert de mouvements
  - Video A: mouvement source
  - Video B: personne cible
    - Mouvements différents de A

➔ Video C de la personne B  
bougeant comme la personne A



Prise de contrôle du mouvement comme une marionnette

# Everybody Dance Now

- Only on image deformation
  - But use skeleton
- Training on the video B (the target)
  - Learn to produce image from skeleton
  - Pose detection: VNect for instance
  - Global pose normalization
- GAN generate new pose/image
  - Generator: optimized with the discriminator + Image comparison (VGG)
  - Discriminator
- Face GAN specific for faces
- Temporal smoothing

# Everybody Dance Now

- Principe

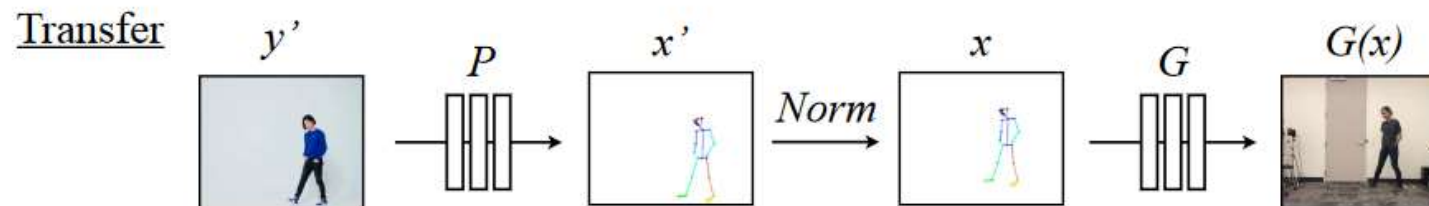
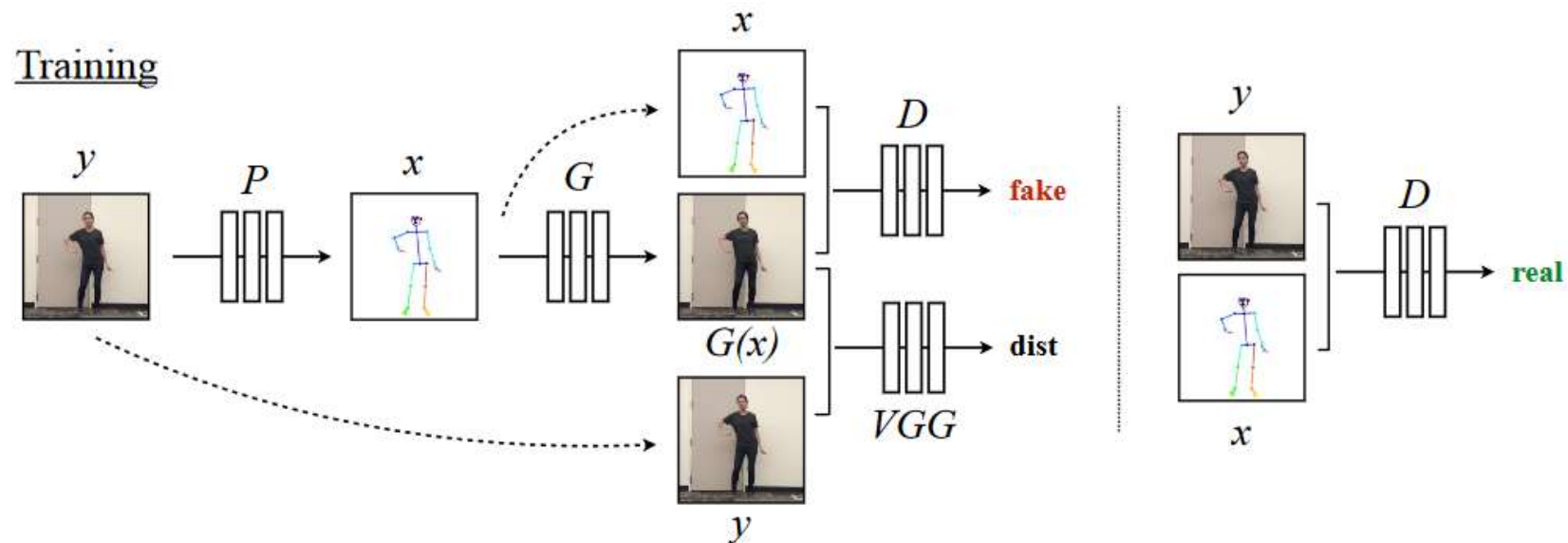
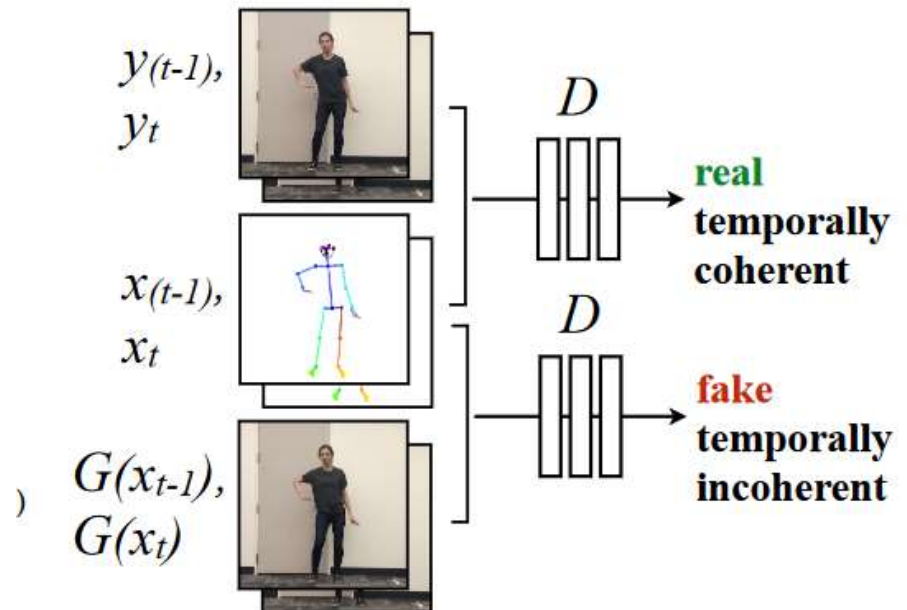
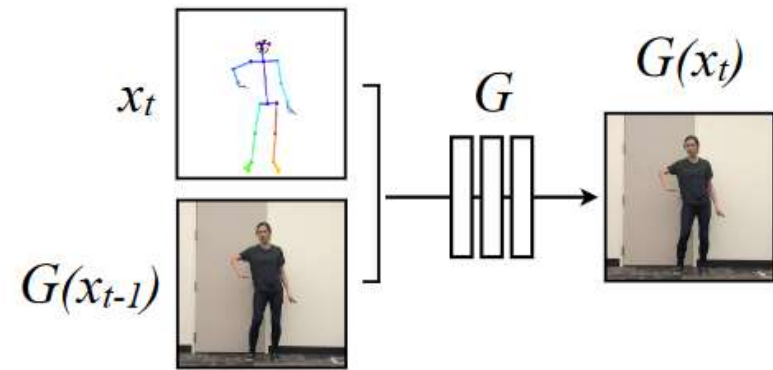


Fig. 3. (Top) **Training**: Our model uses a pose detector  $P$  to create pose stick figures from video frames of the target subject. During training we learn the mapping  $G$  alongside an adversarial discriminator  $D$  which attempts to distinguish between the “real” correspondence pair  $(x, y)$  and the “fake” pair  $(G(x), y)$ . (Bottom) **Transfer**: We use a pose detector  $P : Y' \rightarrow X'$  to obtain pose joints for the source person that are transformed by our normalization process  $Norm$  into joints for the target person for which pose stick figures are created. Then we apply the trained mapping  $G$ .

# Everybody Dance Now

- Temporal smoothing
  - Discriminator D differentiate real temporal coherence and fake sequence





# Everybody Dance Now

- Face generator: GAN

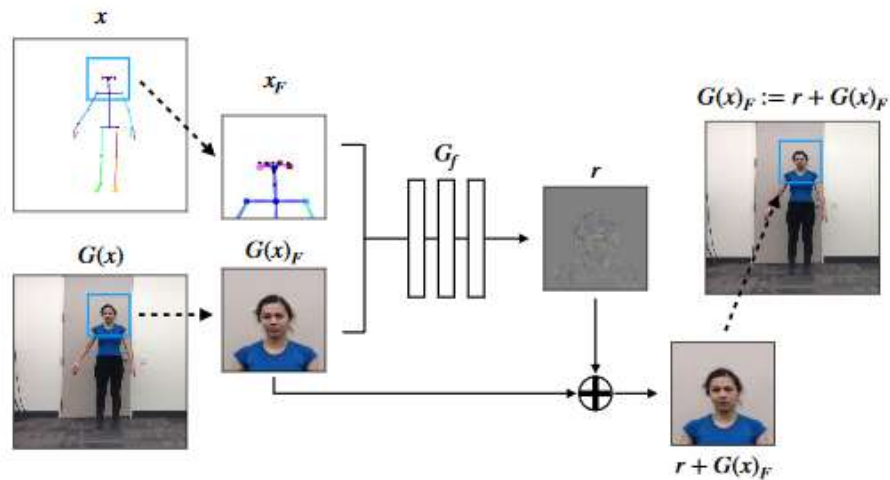


Fig. 5. Face GAN setup. Residual is predicted by generator  $G_F$  and added to the original face prediction from the main generator.

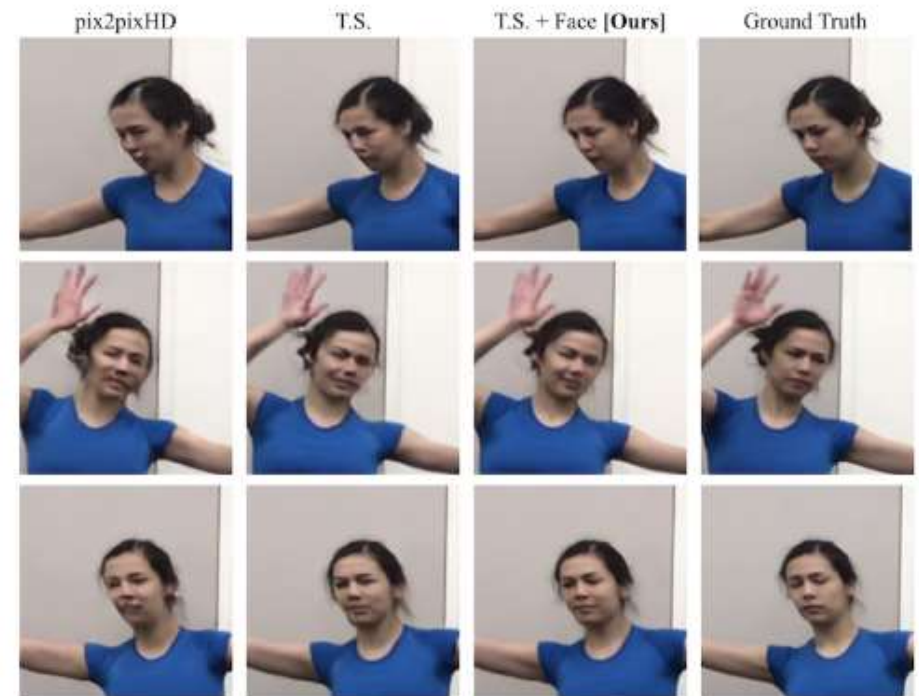
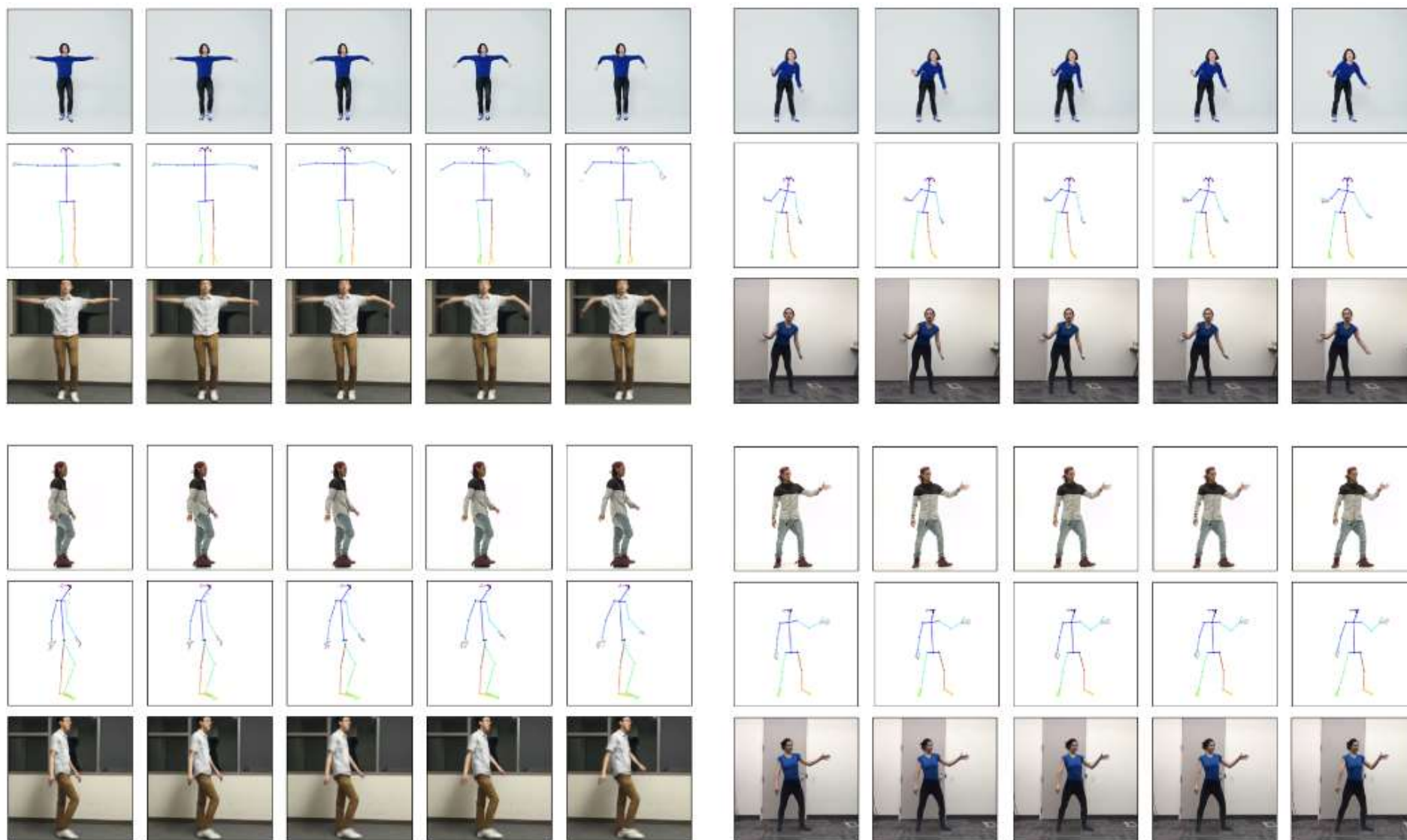


Fig. 8. Face image comparison from different models on the validation set. T.S. denotes a model with our temporal smoothing setup, and T.S. + Face is our full model with both the temporal smoothing setup and Face GAN. Details improve and distortions decrease upon the additions of the temporal smoothing setup and the face GAN.

# Everybody Dance Now



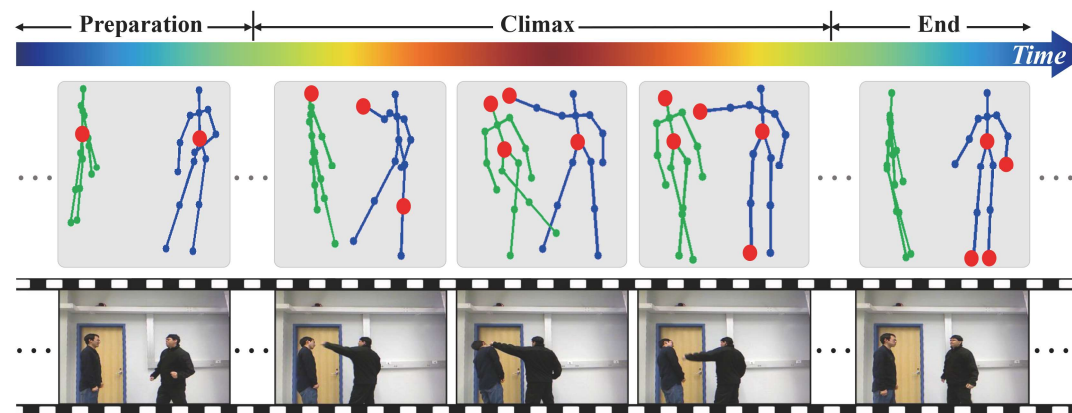
# RECONNAISSANCE ET ANALYSE À PARTIR DES MOUVEMENTS DU CORPS

---

Pour de l'interaction : Jeu vidéo, IHM, RV, etc.

Étude du geste sportif

Etude de la marche en pathologie ambulatoire



# A NEW REPRESENTATION OF SKELETON SEQUENCES FOR 3D ACTION RECOGNITION

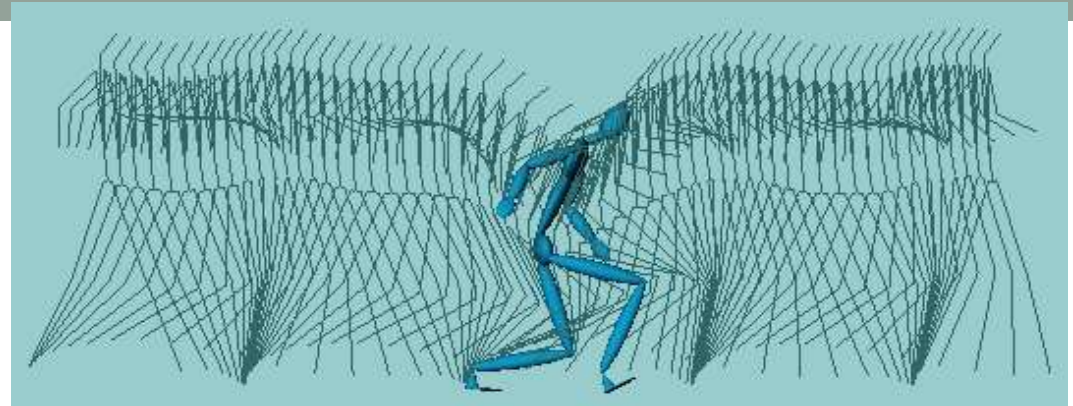
---

QiuHong Ke, Mohammed Bennamoun, Senjian An,  
FerdousSohel, Farid Boussaid

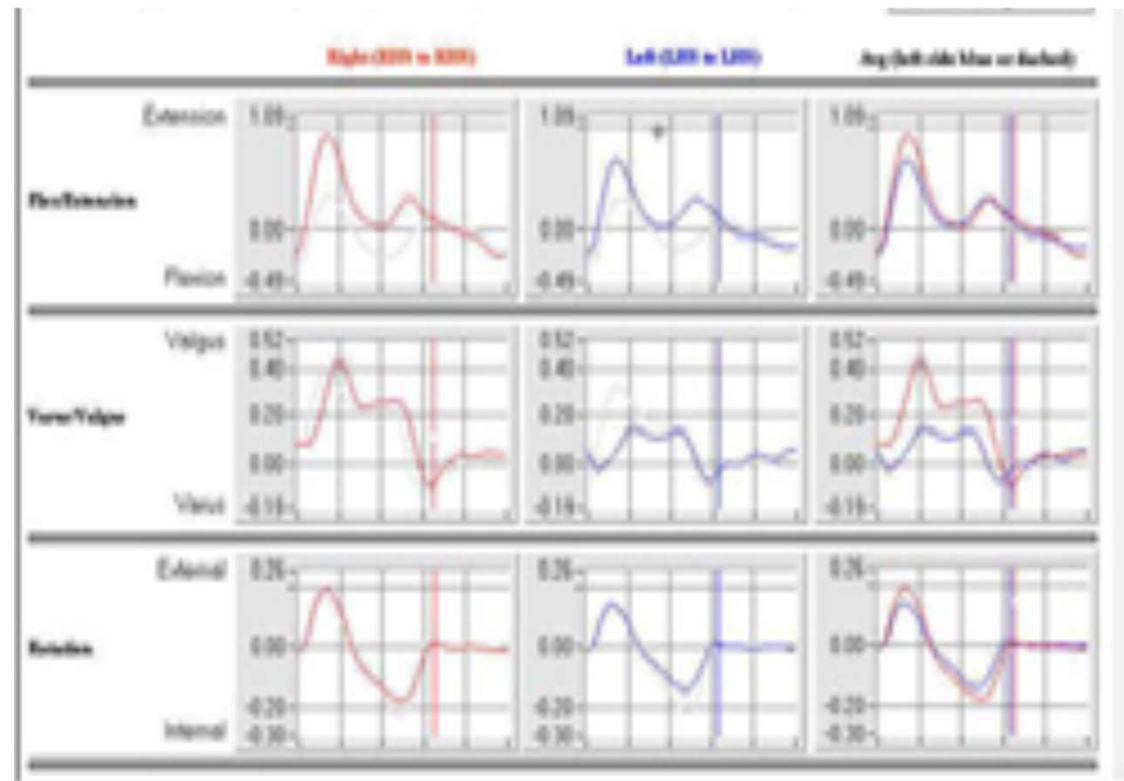
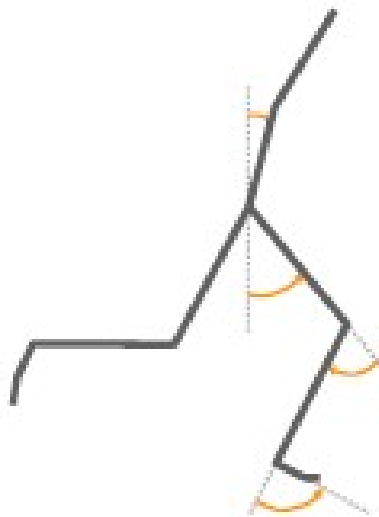
University of Western Australia, Murdoch University

CVPR 2017

# Problem



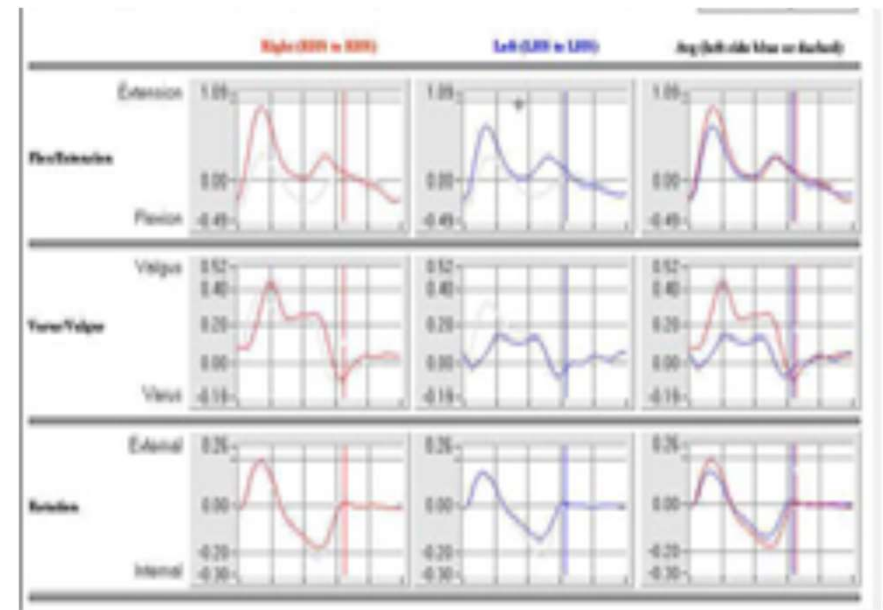
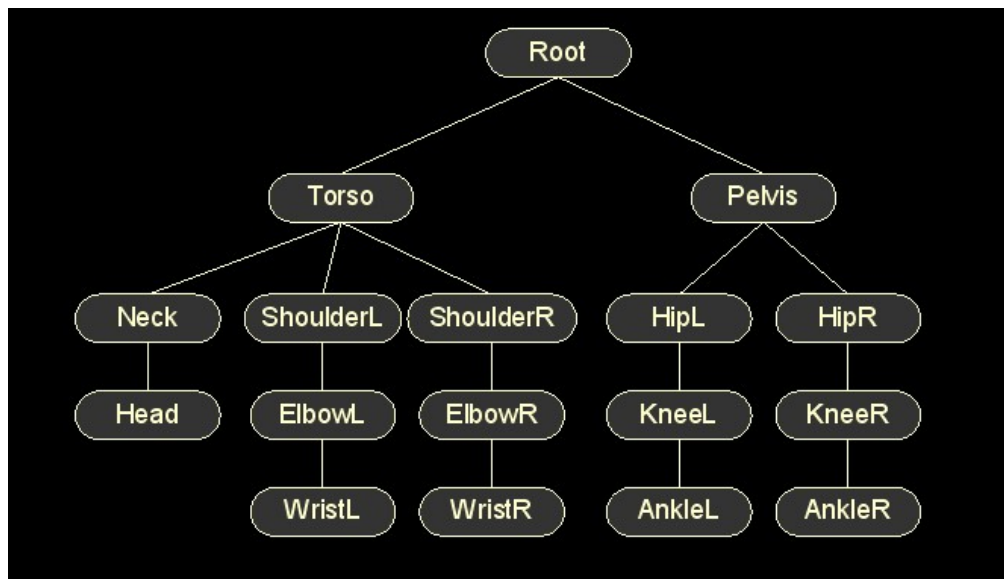
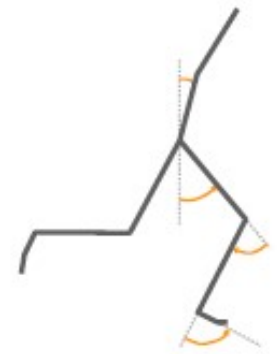
- Skeleton animation → Action ?
  - Several papers every year





# Usual skeleton representation

- Joint/bone: transformation relative to parent
  - Translation (length of the bone)
    - define the rest pose, not important for recognition
  - Rotation = orientation of the bone (angles)
    - Time variation
- Efficient representations ?
  - Matrix, Euler angles, quaternion, 3D joint position, ... ?
  - local VS global ?



# Architecture

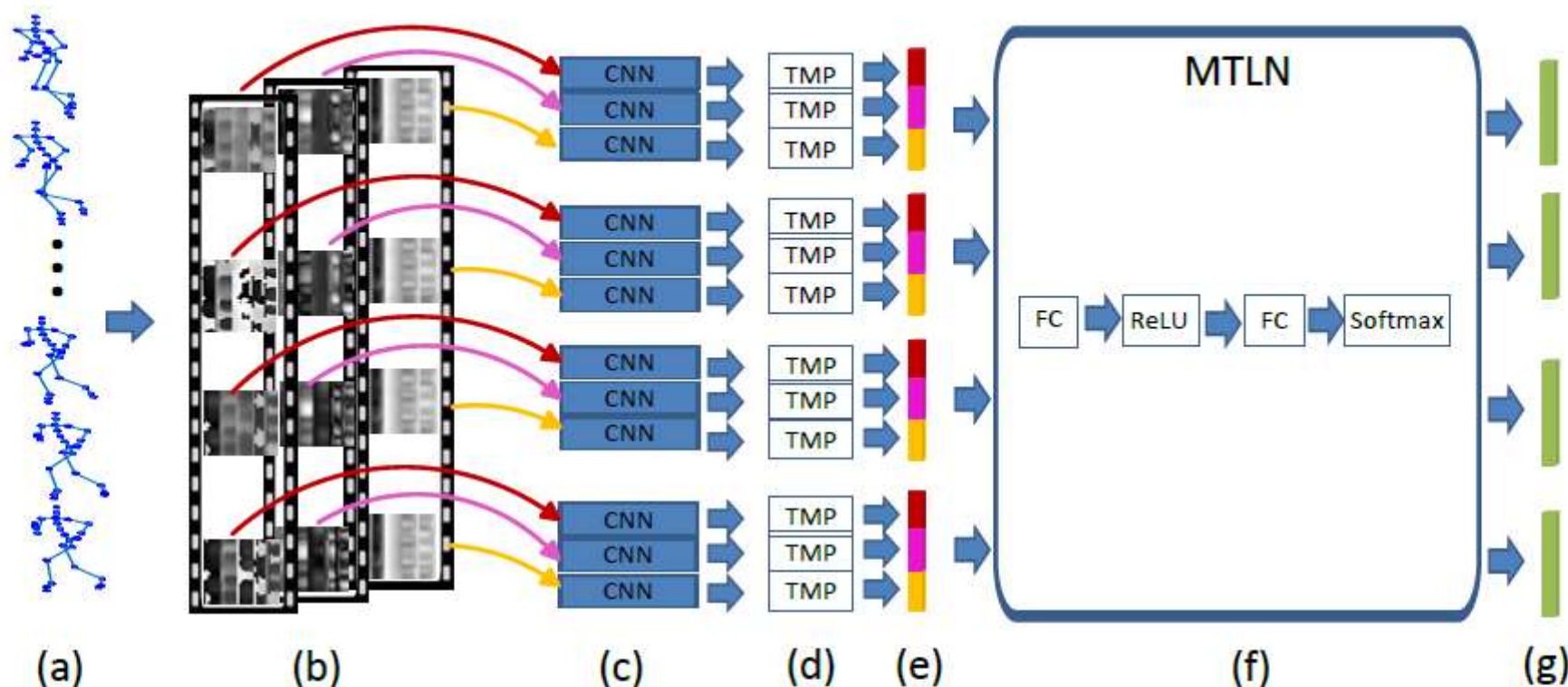


Figure 1. Architecture of the proposed method. Given a skeleton sequence (a), three clips (b) corresponding to the three channels of the cylindrical coordinates are generated. A deep CNN model (c) and a temporal mean pooling (TMP) layer (d) are used to extract a compact representation from each frame of the clips (see Figure 3 for details). The output CNN representations of the three clips at the same time-step are concatenated, resulting four feature vectors (e). Each feature vector represents the temporal information of the skeleton sequence and a particular spatial relationship of the skeleton joints. The proposed MTLN (f) which includes a fully connected (FC) layer, a rectified linear unit (ReLU), another FC layer and a Softmax layer jointly processes the four feature vectors in parallel and outputs four sets of class scores (g), each corresponding to one task of classification using one feature vector. During training, the loss values of the four tasks are summed up to define the loss value of the network used to update the network parameters. For testing, the class scores of the four tasks are averaged to generate the final prediction of the action class.



# Clip Generation of a Skeleton sequence

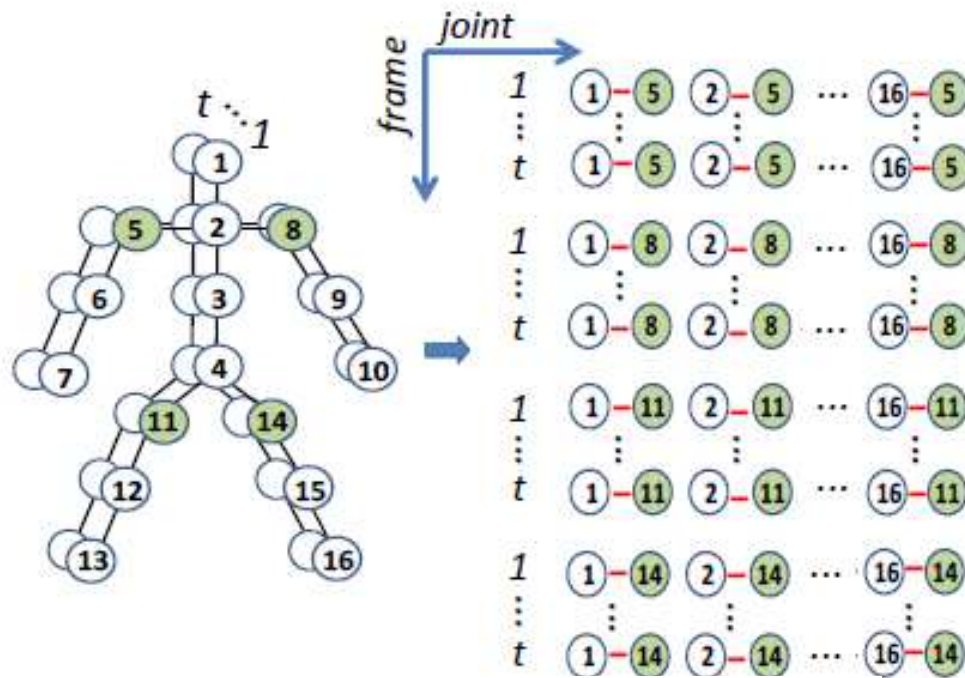
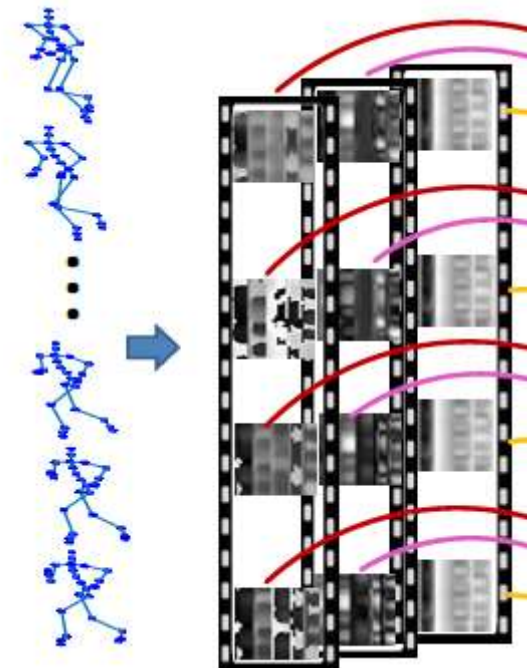
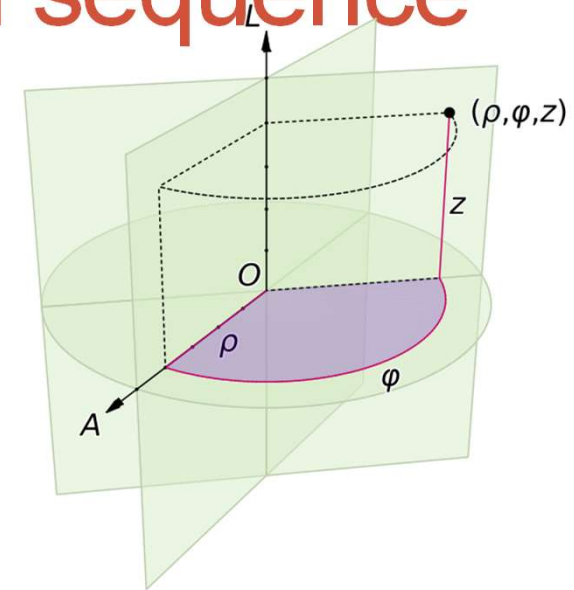
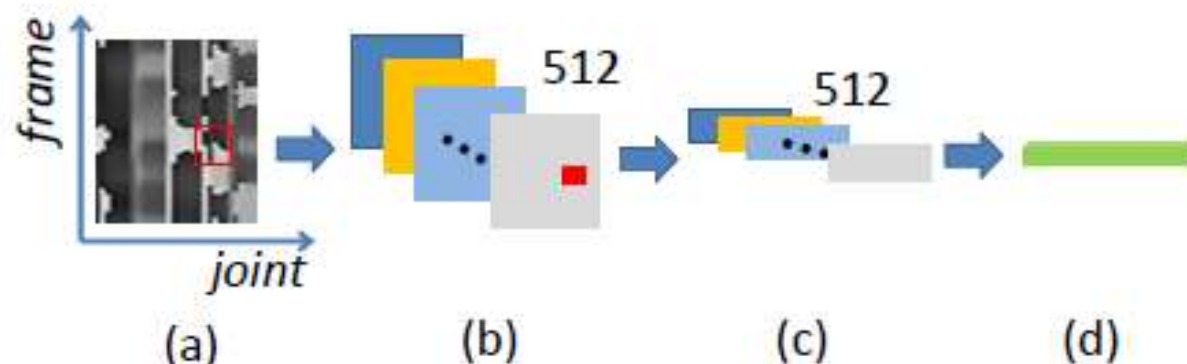


Figure 2. Clip Generation of a skeleton sequence. The skeleton joints of each frame are first arranged as a chain by concatenating the joints of each body part (*i.e.*, 1-2-3-...-16). Four reference joints shown in green (*i.e.*, left shoulder 5, right shoulder 8, left hip 11 and right hip 14) are then respectively used to compute relative positions of the other joints to incorporate different spatial relationships between the joints. Consequently, four 2D arrays are obtained by combining the relative positions of all the frames of the skeleton sequence. The relative position of each joint in the 2D arrays is described with cylindrical coordinates. The four 2D arrays corresponding to the same channel of the coordinates are transformed to four gray images and as a clip. Thus three clips are generated from the three channels of the cylindrical coordinates of the four 2D arrays.



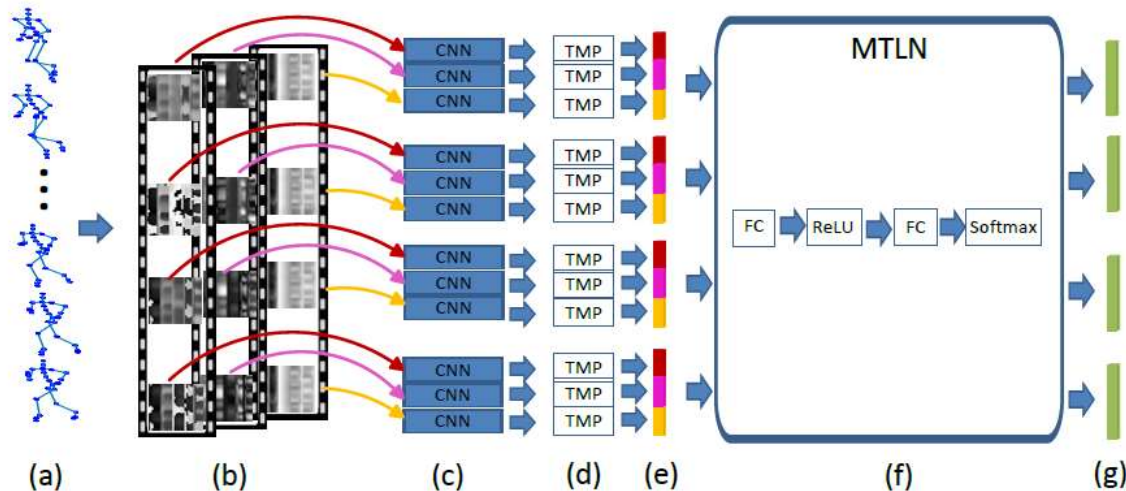
# Temporal Mean Pooling



**CNN=VGG**  
**trained**  
**on real images**  
**+**  
**TMP**

Figure 3. Temporal mean pooling of the CNN feature maps. (a) An input frame of the generated clips, for which the rows correspond to the different frames of the skeleton sequence and the columns correspond to the different vectors generated from the joints. (b) Output feature maps of the conv5\_1 layer. The size is  $14 \times 14 \times 512$ . Each activation (shown in red) of the feature map is a feature correspond to the local region of the original image (shown with a red square). (c) Temporal features of all joints of the skeleton sequence, which are obtained by applying mean pooling to each feature map in the row (temporal) dimension. (d) Output feature, which is achieved by concatenating all the feature maps in (c).

# MTLN: multi task learning network



- 3 x 7168D features of the 3 clips at the same time-step are concatenated  
 ➔ 4 feature vectors with intrinsic relationships
- MTLN jointly process the 4 feature vectors
  - FC ➔ RELU ➔ FC ➔ Softmax
- Loss function

$$\mathcal{L}(Z, y) = \sum_{k=1}^4 \ell_k(z_k, y)$$

$$\begin{aligned} \ell_k(z_k, y) &= \sum_{i=1}^m y_i \left( -\log \left( \frac{\exp z_{ki}}{\sum_{j=1}^m \exp z_{kj}} \right) \right) \\ &= \sum_{i=1}^m y_i \left( \log \left( \sum_{j=1}^m \exp z_{kj} \right) - z_{ki} \right) \end{aligned}$$



# Results

Table 1. Performance on the NTU RGB+D dataset.

Methods	Accuracy	
	Cross Subject	Cross View
Lie Group [42]	50.1%	52.8%
Skeletal Quads [7]	38.6%	41.4%
Dynamic Skeletons [16]	60.2%	65.2%
Hierarchical RNN [6]	59.1%	64.0%
Deep RNN [37]	59.3%	64.1%
Deep LSTM [37]	60.7%	67.3%
Part-aware LSTM [37]	62.9%	70.3%
ST-LSTM [26]	65.2%	76.1%
ST-LSTM + Trust Gate [26]	69.2%	77.7%
Coordinates + FTP	61.06%	74.64%
Frames + CNN	75.73%	79.62%
Clips + CNN + Concatenation	77.05%	81.11%
Clips + CNN + Pooling	76.37%	80.46%
Clips + CNN + MTLN	<b>79.57%</b>	<b>84.83%</b>

Table 2. Performance on the SBU kinect interaction dataset.

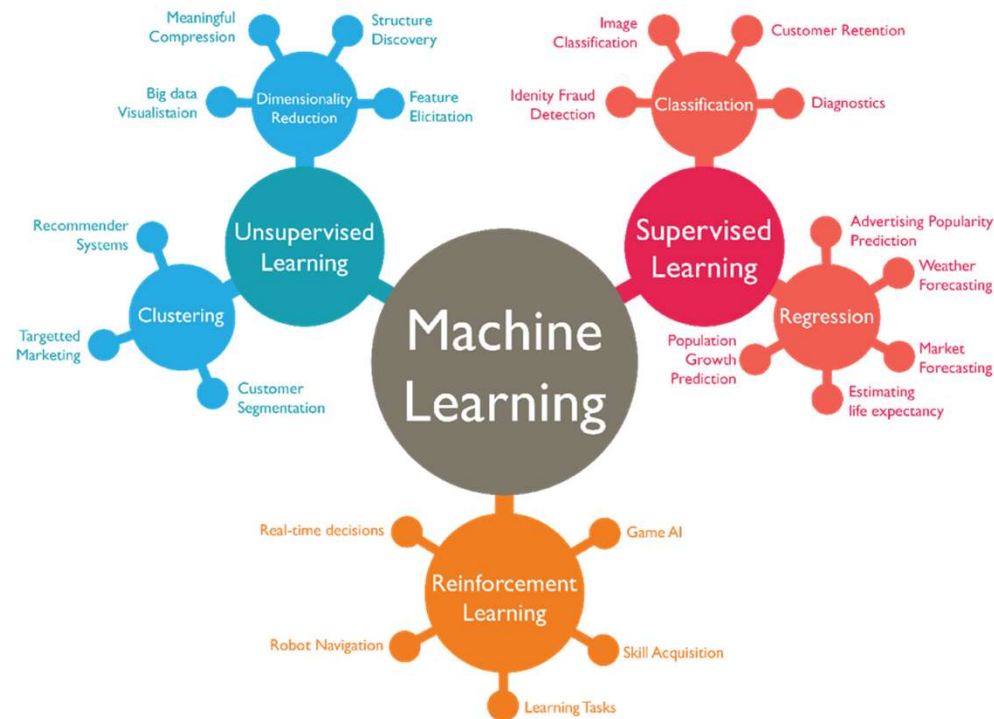
Methods	Accuracy
Raw Skeleton [53]	49.7%
Joint Feature [18]	86.9%
CHARM [25]	83.9%
Hierarchical RNN [6]	80.35%
Deep LSTM [54]	86.03%
Deep LSTM + Co-occurrence [54]	90.41%
ST-LSTM [26]	88.6%
ST-LSTM + Trust Gate [26]	93.3%
Coordinates + FTP	79.75%
Frames + CNN	90.88%
Clips + CNN + Concatenation	92.86%
Clips + CNN + Pooling	92.26%
Clips + CNN + MTLN	<b>93.57%</b>

Table 3. Performance on the CMU dataset.

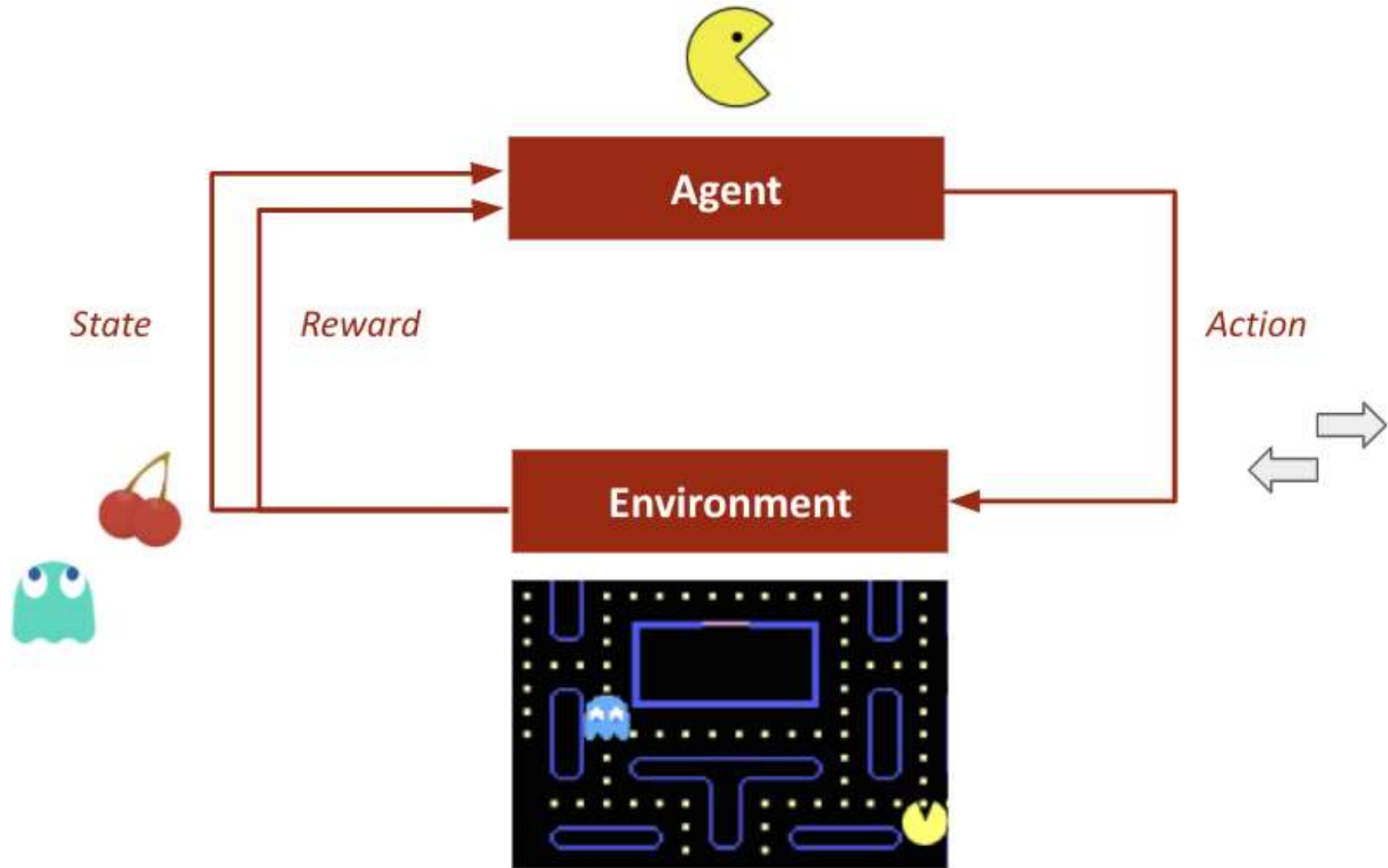
Methods	Accuracy	
	CMU subset	CMU
Hierarchical RNN [6]	83.13%	75.02%
Deep LSTM [54]	86.00%	79.53%
Deep LSTM + Co-occurrence [54]	88.40%	81.04%
Coordinates + FTP	83.44%	73.61%
Frames + CNN	91.53%	85.36%
Clips + CNN + Concatenation	90.97%	85.76%
Clips + CNN + Pooling	90.66%	85.56%
Clips + CNN+ MTLN	<b>93.22%</b>	<b>88.30%</b>

# APRENTISSAGE PAR RENFORCEMENT POUR LA GÉNÉRATION DE GESTES

---

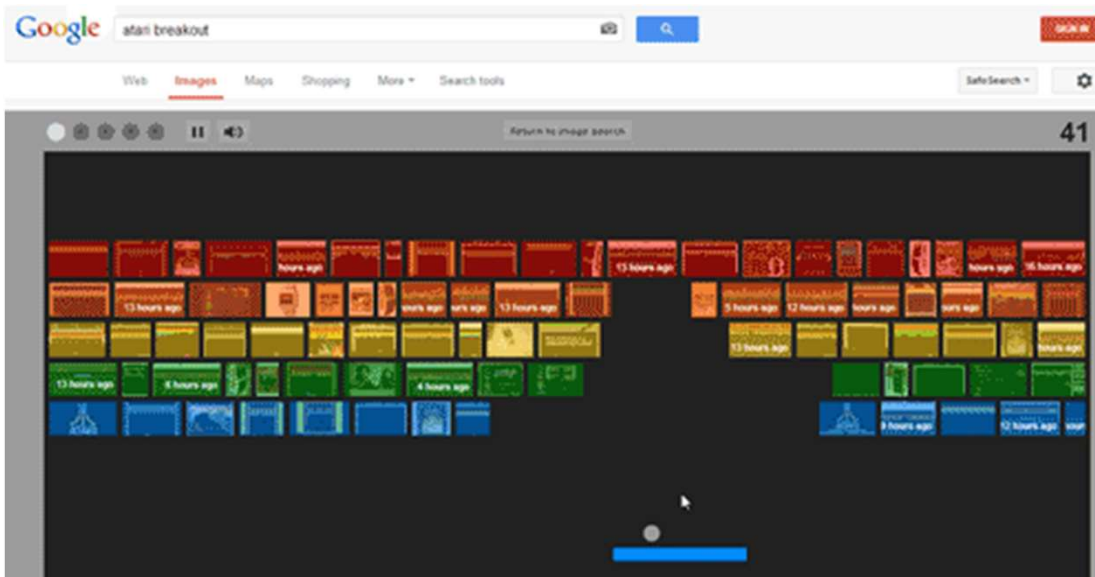
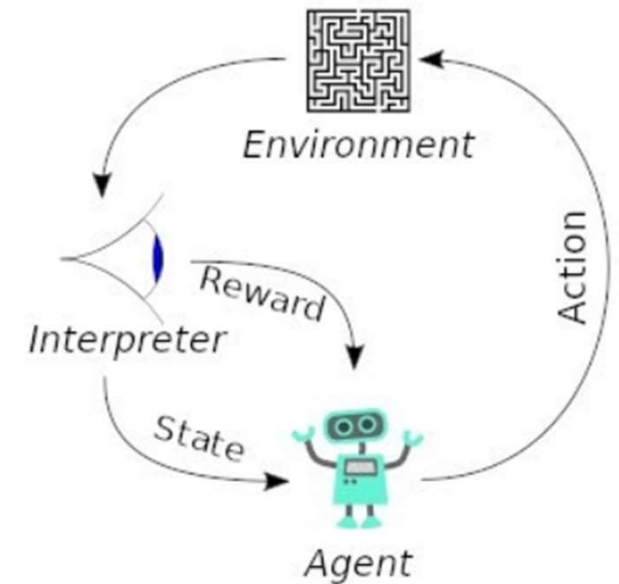


# Apprentissage par renforcement



# Apprentissage par renforcement

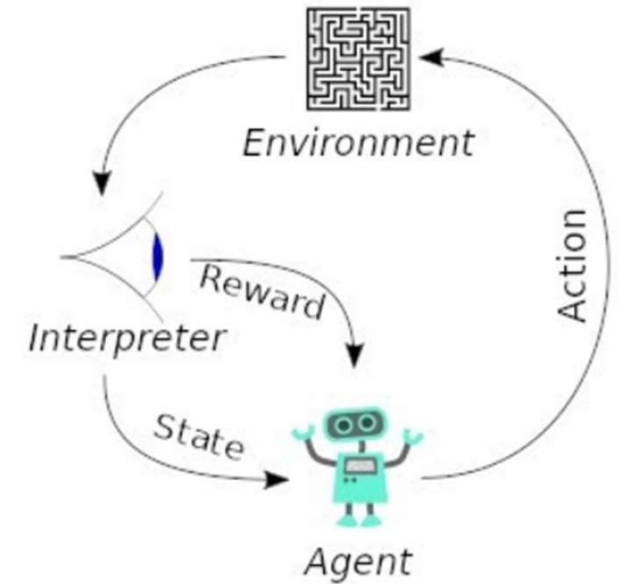
- Applications
  - Apprendre à jouer (jeux vidéo, etc.)
  - Pronostiques (économie, sports, etc.)
  - Recommandations (web commercial)
  -



# Apprentissage par renforcement

- Applications

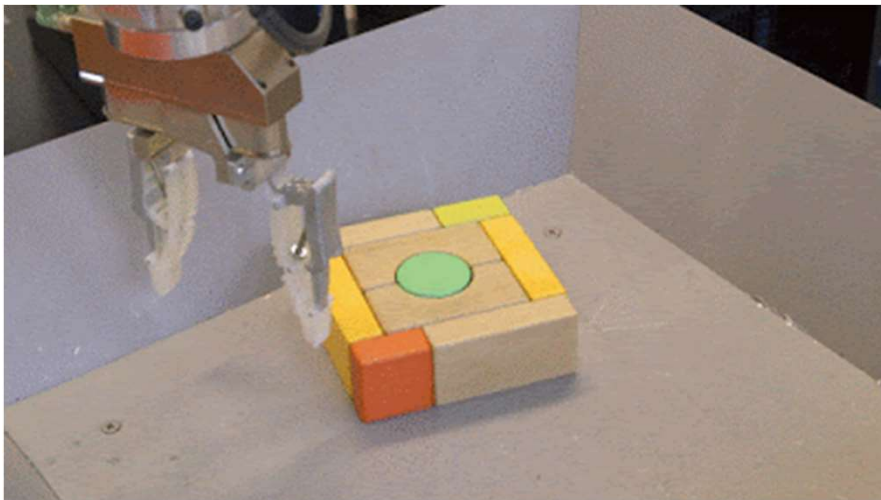
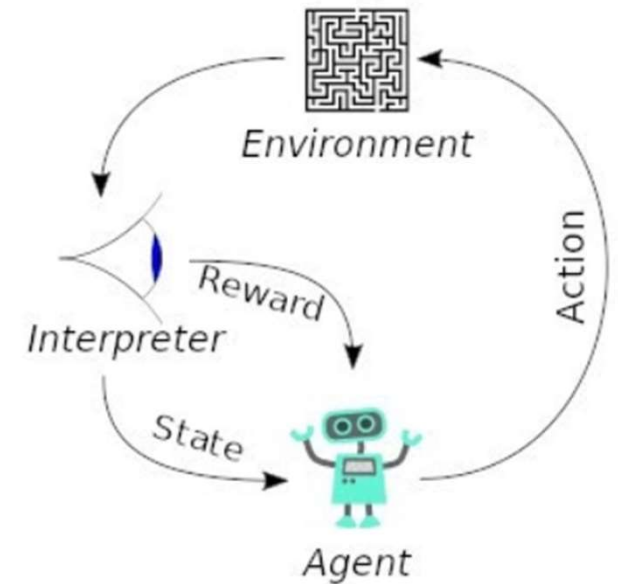
- Apprendre à jouer (jeux vidéo, etc.)
- Pronostiques (économie, sports, etc.)
- Recommandations (web commercial)
- Véhicules autonomes





# Apprentissage par renforcement

- Applications
  - Apprendre à jouer (jeux vidéo, etc.)
  - Pronostiques (économie, sports, etc.)
  - Recommandations (web commercial)
  - Véhicules autonomes
  - Robotique / geste



# 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  $\pi_t(s, a) = \text{Prob que } a_t = a \text{ si } s_t = s$
- Les transitions et récompenses ne dépendent que de l'état et de l'action précédents : processus Markovien

# Apprentissage par renforcement

- *Politique* :

ensemble d'associations *situation*  $\rightarrow$  *action* (une application)

Une simple table ... un algorithme de recherche intensive

Eventuellement stochastique

- *Fonction de renforcement* :

- Définit implicitement le but poursuivi

- Une fonction :  $(\text{état}, \text{action}) \rightarrow \text{récompense} \in \mathbb{R}$

- *Fonction d'évaluation*  $V(s)$  ou  $Q(s,a)$  :

- Récompense accumulée sur le long-terme

- *Modèle de l'environnement* :

- Fonctions  $T$  et  $R$  :  $(\text{état}(t), \text{action}) \rightarrow (\text{état}(t+1), \text{récompense})$

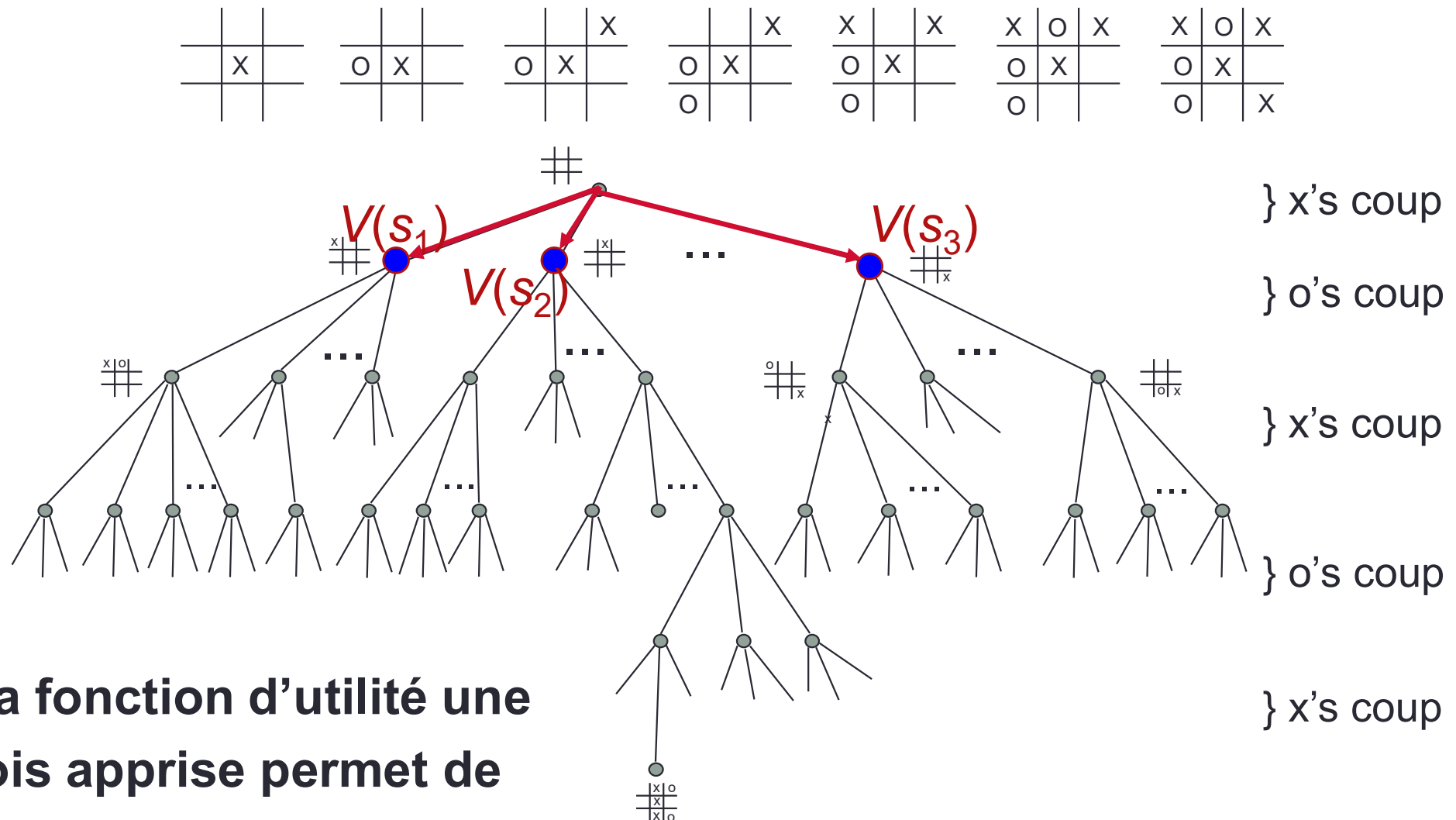
# Apprentissage par renforcement

## Principe

- Choisir une action sans avoir besoin de faire une exploration 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

➔ Les réseaux sont de très bon outils pour ca !

# Apprentissage par renforcement

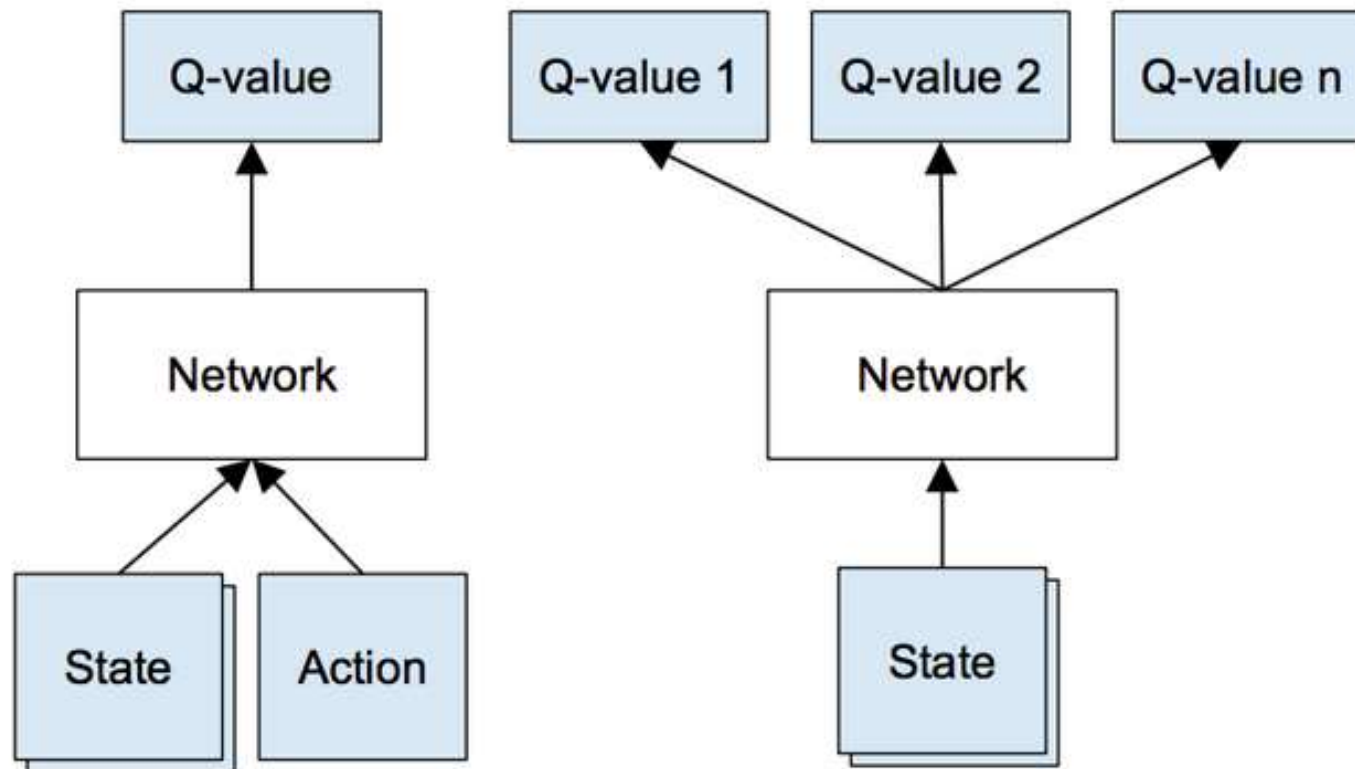


**La fonction d'utilité une fois apprise permet de jouer sans exploration de l'arbre de jeu**



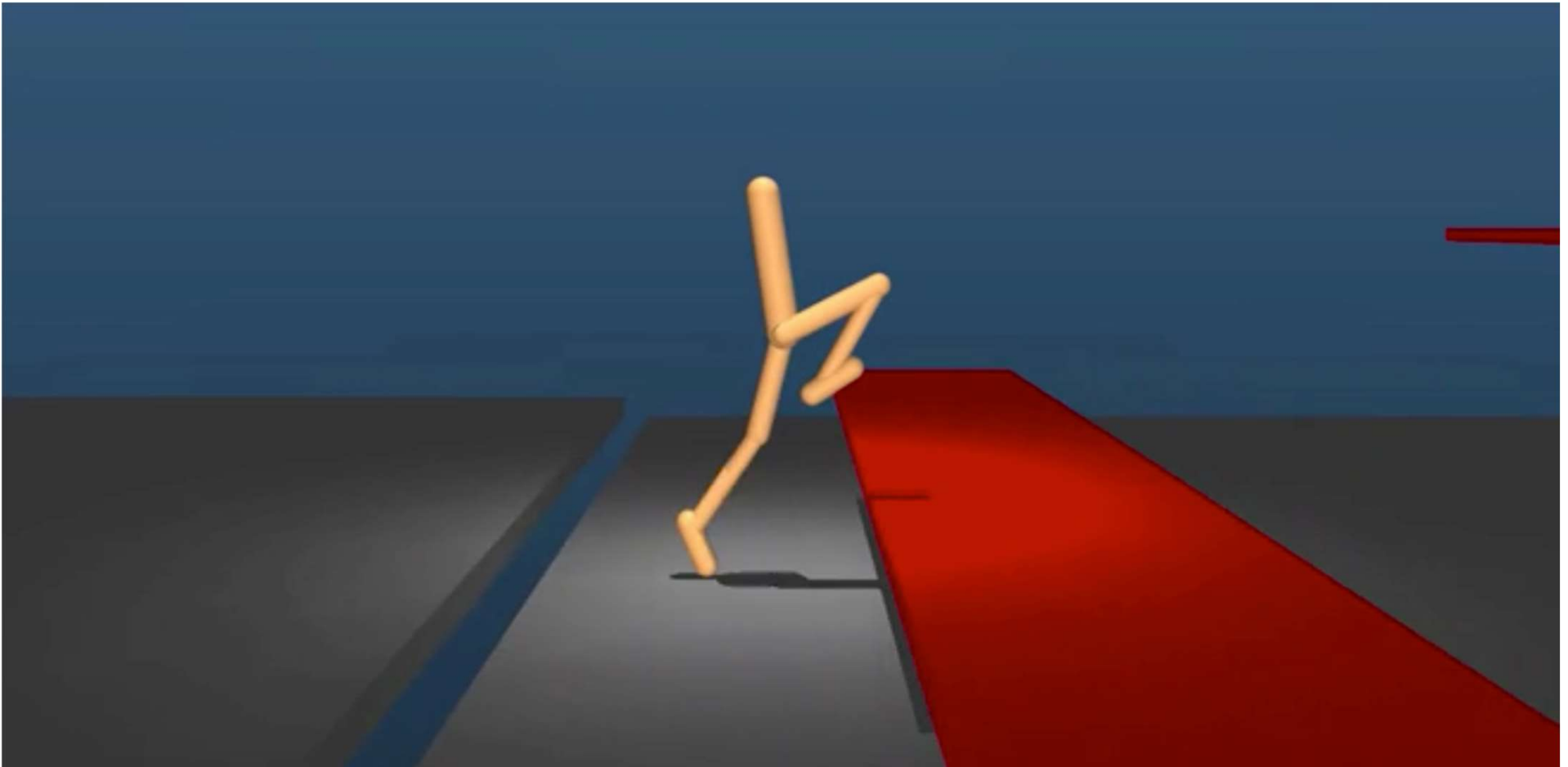
# Deep Q Learning

- Réseau de neurones bien adaptés au problème



# Learning to move

+vidéos



... de nombreuses autres applications ...

à inventer

# Suite de ce cours

- Réseaux de diffusion
- Ouvrir à d'autres données que les images en graphisme
  - Mesh
  - Nuage de points
  - Nerf (en rendu)  
➔ Julie Digne
- Voir aussi un des cours d'animation
  - Animation de personnage virtuel et apprentissage profond dans l'UE ACAMP
- Il faudrait aussi regarder
  - Apprentissage par renforcement plus en détails ...

VRAC

---



# 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>

<https://tml.stanford.edu/publications/2022/transformer-inertial-poser-real-time-human-motion-reconstruction-sparse-imus>

SIGGRAPH Asia 2022

# Reconstruction d'un mouvement à partir de 6 accéléromètres

Evolution d'un problème :

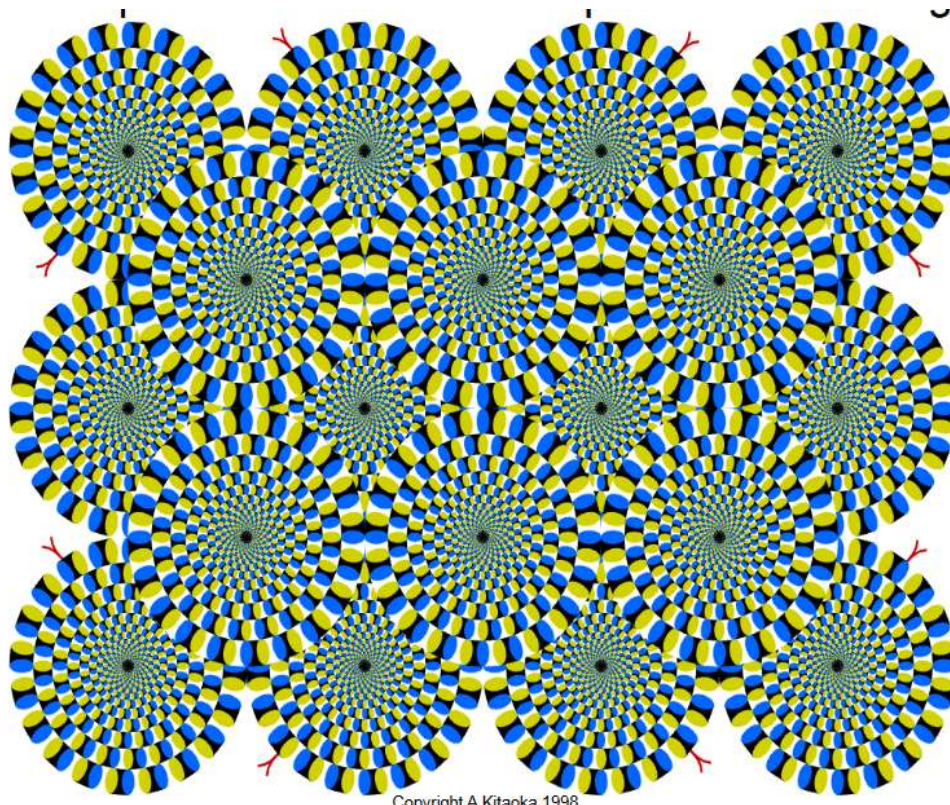
1. D'abord montrer que c'est possible : 1er année
2. Avoir une base de données qui donne une vérité terrain, juste la locomotion : 2e année
3. Améliorer la précision : 3e année
4. Problème plus dur en faisant marcher la personne sur un sol non plat
  - besoin de reconstruire le mouvement + le sol :

<https://tml.stanford.edu/publications/2022/transformer-inertial-poser-real-time-human-motion-reconstruction-sparse-imus>

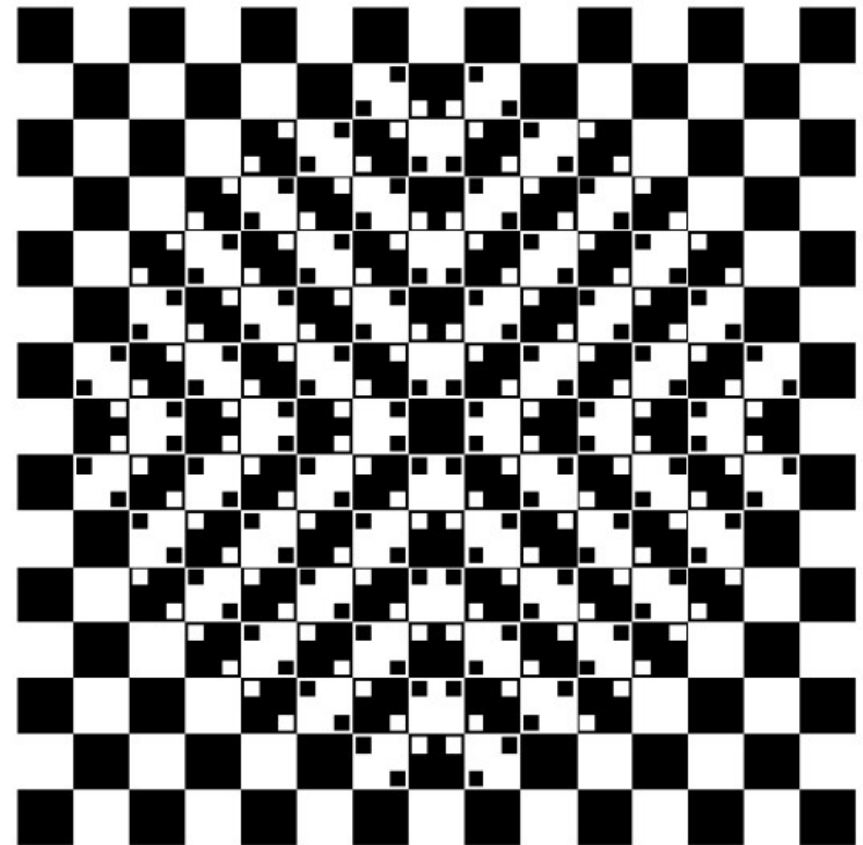
SIGGRAPH Asia 2022

# Vision par ordinateur et vision humaine

- Vision par ordinateur reste limitée
  - Pour l'instant



Copyright A. Kitaoka 1998

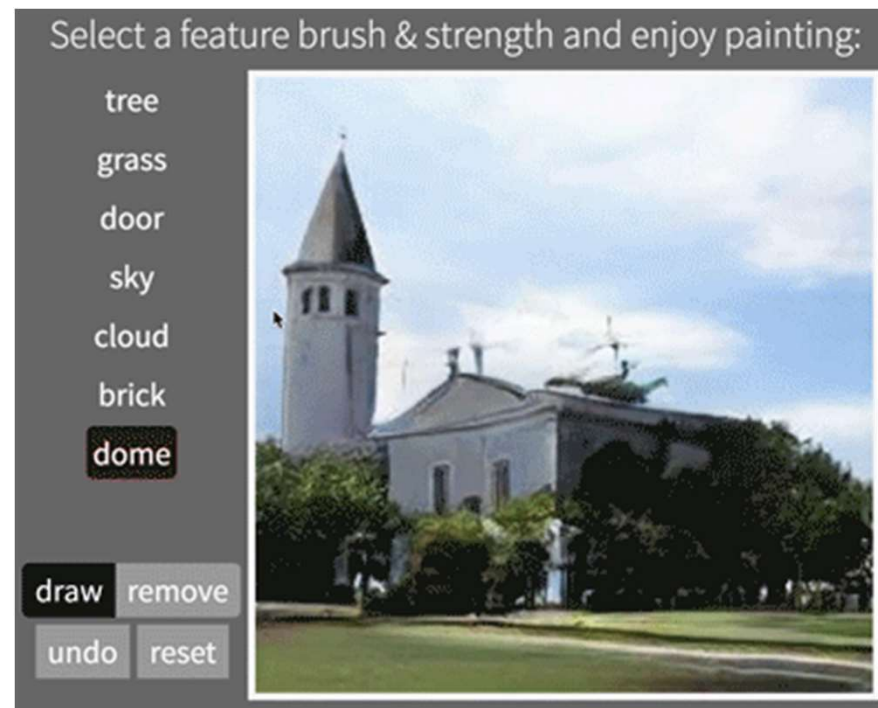


# GAN : un article à lire

GAN Dissection: Visualizing and Understanding Generative Adversarial Networks

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum<sup>1</sup>, William T. Freeman, Antonio Torralba

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





# GAN : modification d'un visage

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

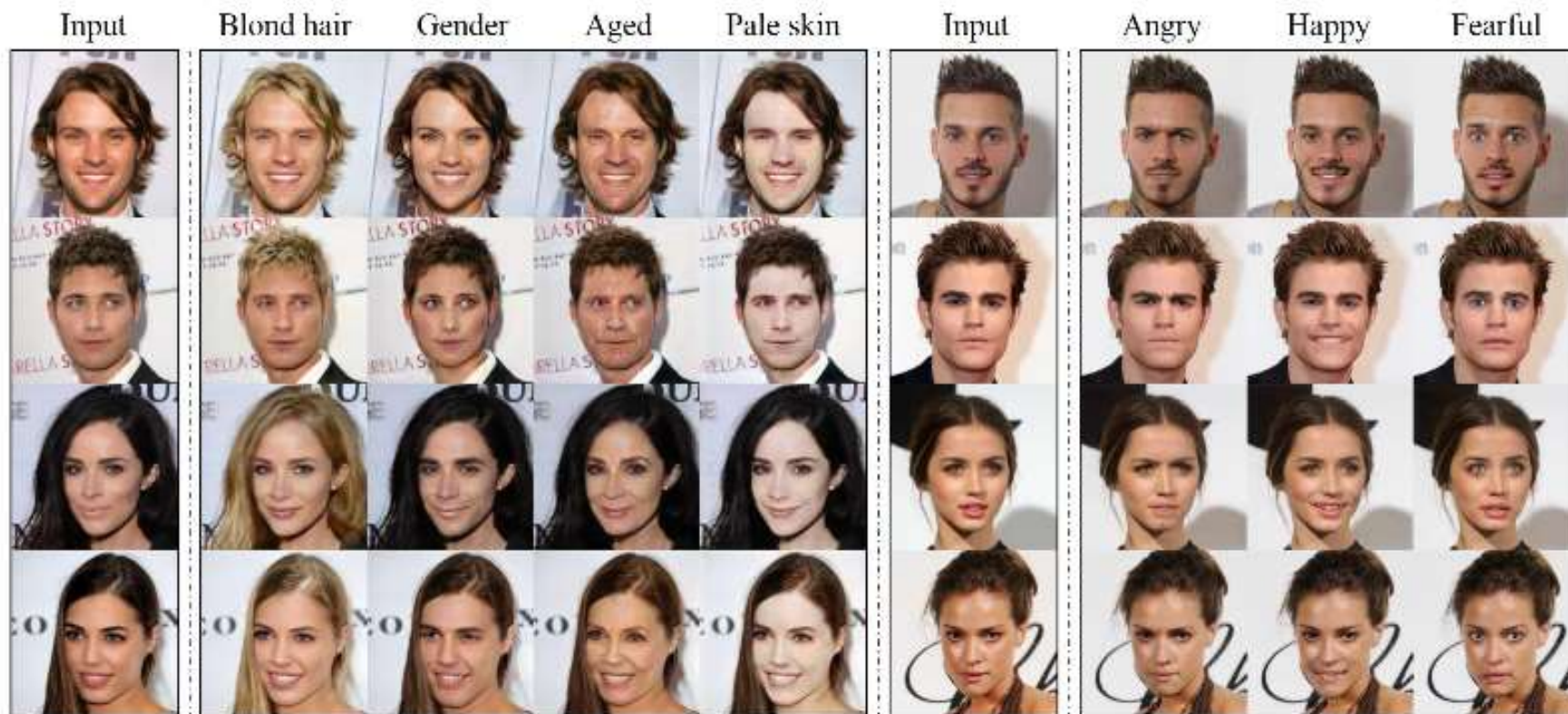


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator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



# GAN : modification d'un visage

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

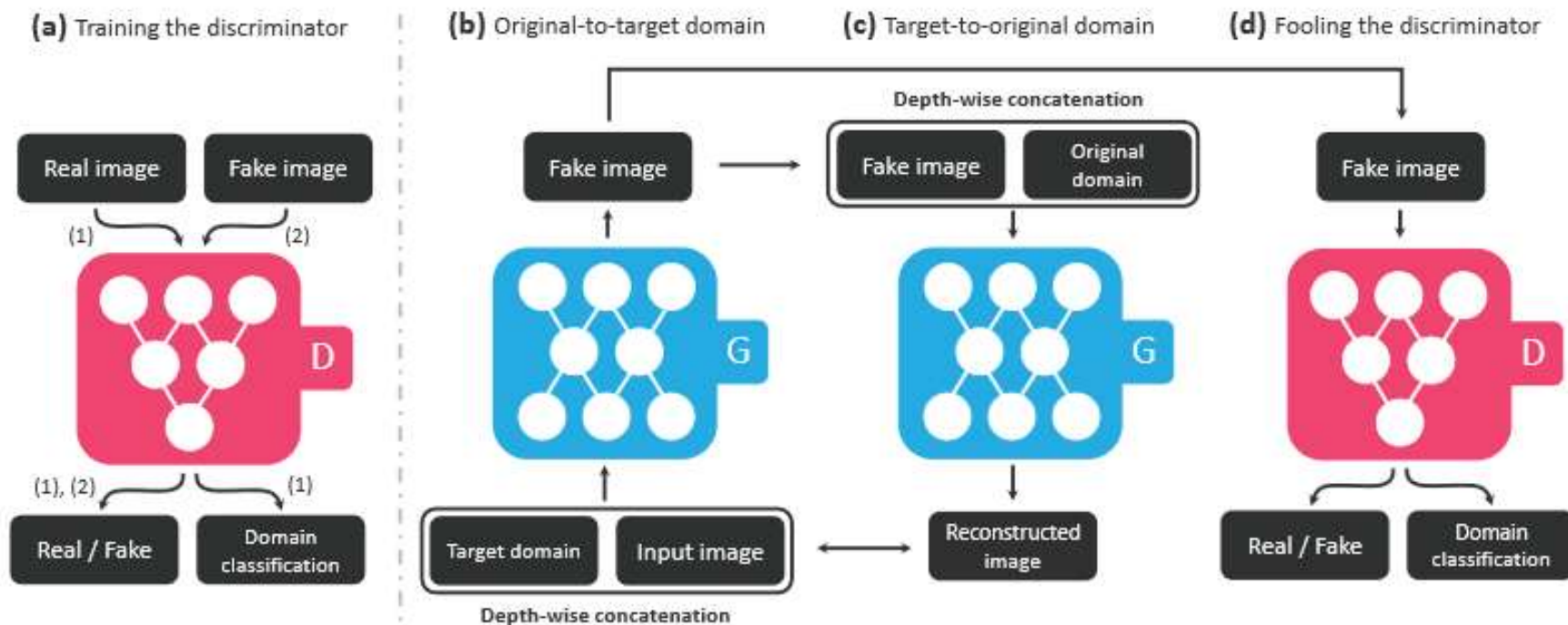


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator  $D$  and a generator  $G$ . (a)  $D$  learns to distinguish between real and fake images and classify the real images to its corresponding domain. (b)  $G$  takes in as input both the image and target domain label and generates 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  $D$ .