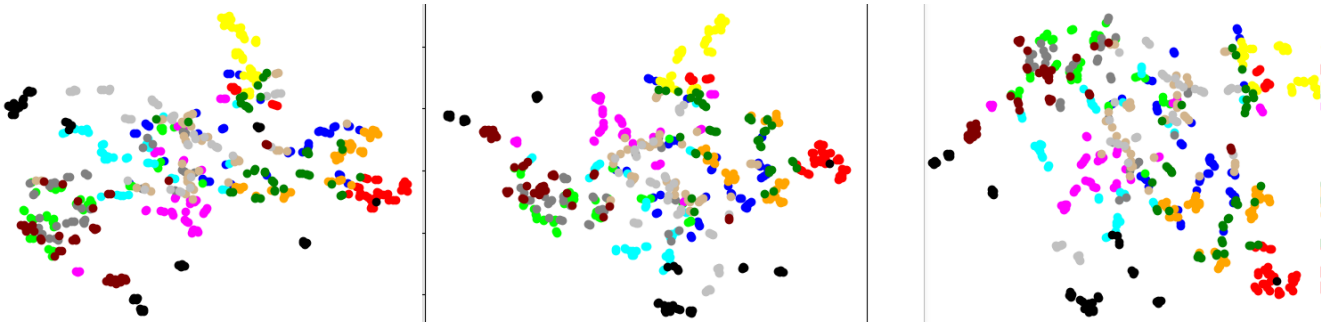
Compte rendu PLDAC géométrie reconstruction 3D
BEROUKHIM CHAOUCHE
encadrants Baskiotis Guigue
date

Intro
1) Article
bases : PyTorch, CNN, VGG, ImageNet, autres trucs mentionnés par Groueix
exécution du classifieur

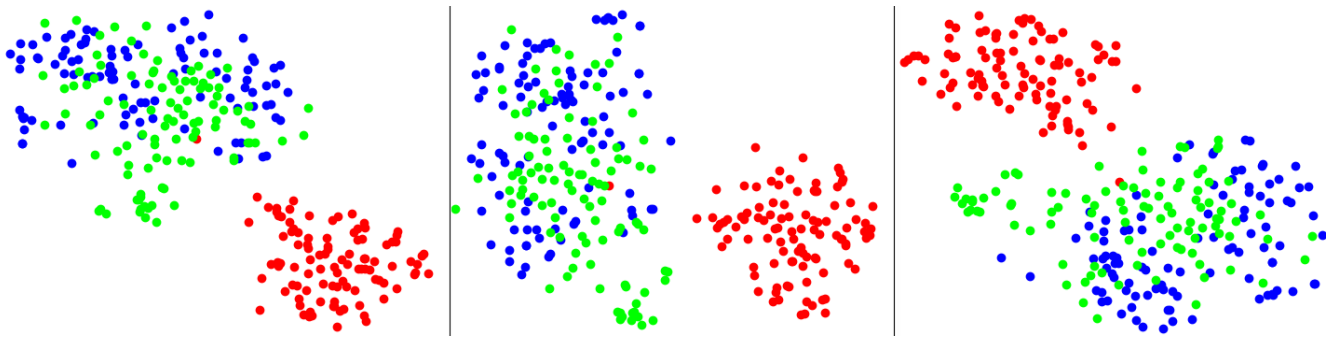
2) tSNE
CNN => autoencoder ?
VGG est un réseau de neurones composé de 36 couches consécutives. Les 30 premières couches résument l'entrée en un plus petit vecteur, c'est la partie « feature extraction ». Les couches suivantes réalisent l'étape de classification à proprement parler.
La partie classification de VGG ne nous intéressant pas, nous nous servons du réseau comme extracteur de caractéristique en stoppant l'exécution du réseau de neurone à une couche intermédiaire (on récupère un vecteur latent).
VGG prend en entrée des images RGB 224*224, la dimension de l'entrée est donc 224*224*3 = 150 528. Le vecteur latent récupéré est de dimension 25 088.

L'algorithme t-SNE (T-distributed Stochastic Neighbor Embedding) est un algorithme servant à visualiser des données de grande dimension. Il prend en entrée des points dans un espace de grande dimension et les ramène dans un espace à 2 ou 3 dimensions tout en tentant de préserver les distances entre les points.
La dimension des vecteurs latents étant trop élevée, nous commençons par réduire le nombre de dimensions par PCA (Principal Component Analysis) afin de réduire le bruit (comme demandé dans la documentation de sklearn.tSNE).



Résultat avec les 13 classes (pour 3 différents learning rate)

Les différentes classes d'objet se regroupent à peu près en cluster, cela démontre la capacité du réseau de neurone à résumer les informations dans un espace de plus petite taille que celle de l'entrée.



Exhibition de deux classes confondues et une distincte (pour 3 différents learning rate)
(rouge: avion – bleu: meuble – vert: enceintes)

3) Classifieur
Nous avons entraîné un réseau simple prenant en entrée les vecteurs latents et classifiant en deux classes. Les performances obtenues sont :
meuble – avion : train 100 % test 100 % (écart type nul)
meuble – enceintes : train 78 % test 66 % (écart type fort)

4) Auto encodeur de nuages de points 3D
L'auto encodeur implémenté est un réseau de neurones défini par :

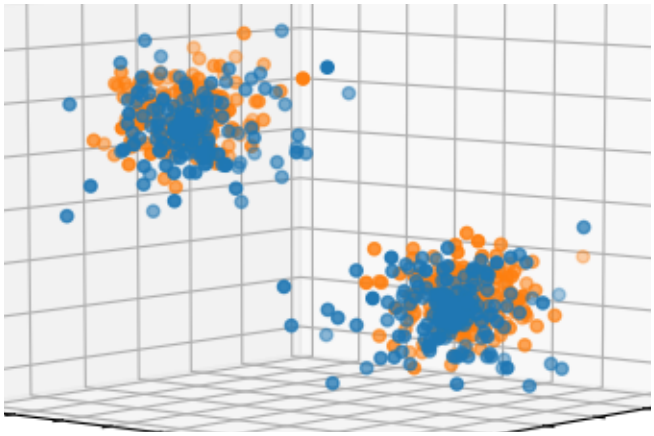
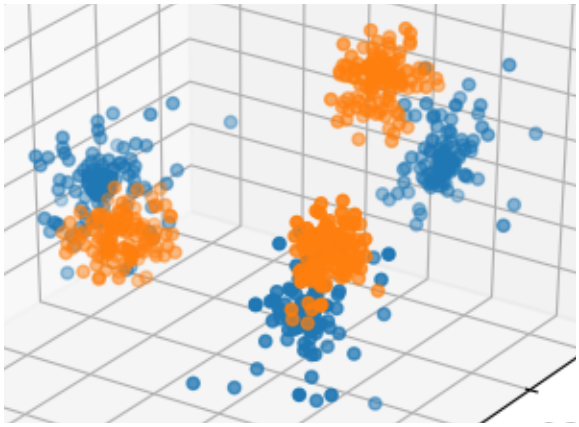
- une entrée "3N" : un ensemble de n points dans $[-1; 1]^3$
- un encodeur : 3N -> FC 512 -> ReLU -> FC 128 -> ReLU -> FC 2
- (on récupère un vecteur latent de taille 2)
- un décodeur : 2 -> FC 128 -> ReLU -> FC 512 -> ReLU -> FC 3N -> tanh
- une sortie de même dimension que l'entrée

Pour entraîner le réseau, on a généré nos propres nuages de points. Certains nuages sont constitués de points tirés selon une distribution gaussienne autour du centre du nuage, d'autres sont des sphères. Afin de complexifier les nuages, on a aussi créé des nuages contenant deux ou trois gaussiennes.

La taille de l'entrée du réseau étant fixe, tous nos nuages contiennent exactement le même nombre de point. Le nombre de points est fixé arbitrairement à la main (en général entre 50 et 360) de telle sorte que les formes soient bien reconnaissables à vue d'œil.

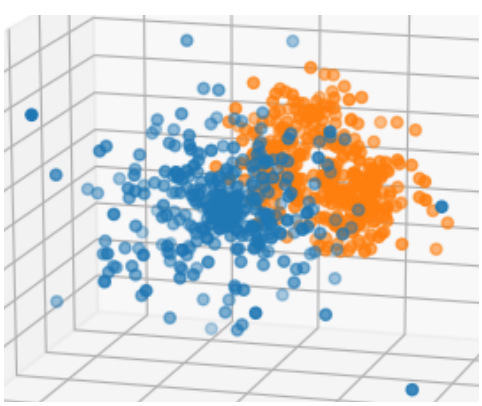
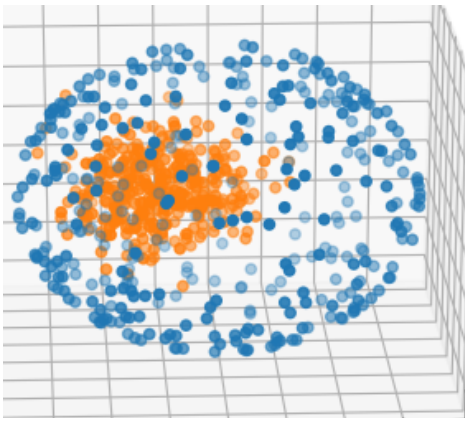
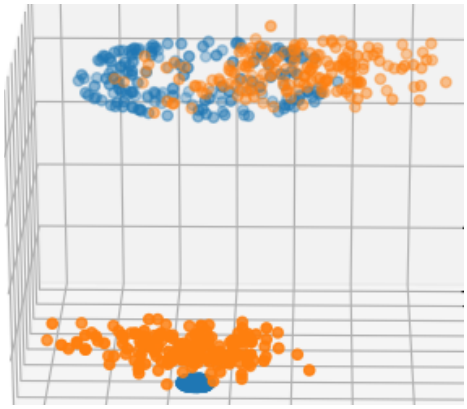
Pour chaque point du nuage initial, l'auto encodeur génère un unique point correspondant. On peut donc définir une fonction de coût définie comme la somme des écarts entre chaque couple de point. En élevant les écarts au carré, on obtient le coût des moindres carrés.

La fonction de coût utilisée pour entraîner le réseau est inspirée de la distance de Chamfer (cf partie suivante). Cette version simplifiée correspond au coût des moindres carrés additionné à la plus petite distance entre un point du nuage original et un point du nuage reconstruit. Par la suite, nous avons remarqué que la distance minimale était négligeable comparée aux moindres carrés, le ratio étant de l'ordre de 4 ordres de grandeurs quel que soit la situation.

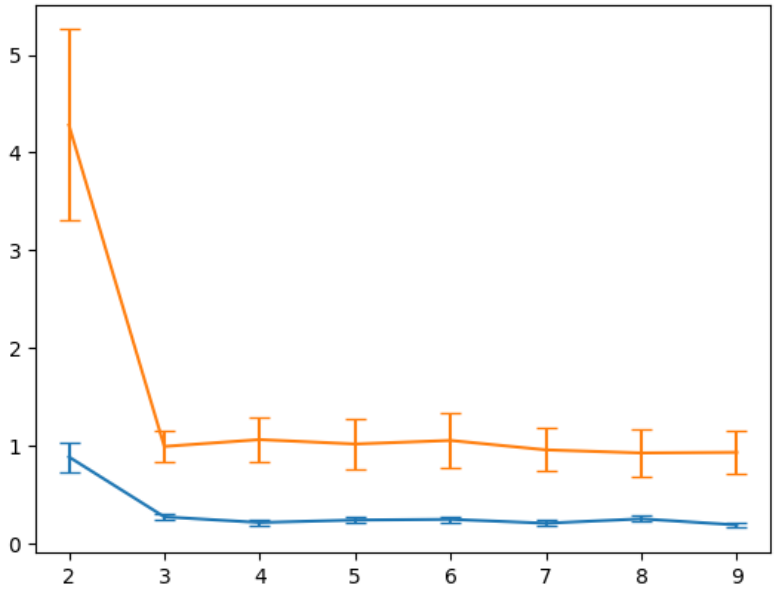
*Nuages de point initiaux en bleu et nuages reconstruits en orange (2 gaussiennes à gauche et 3 à droite)*

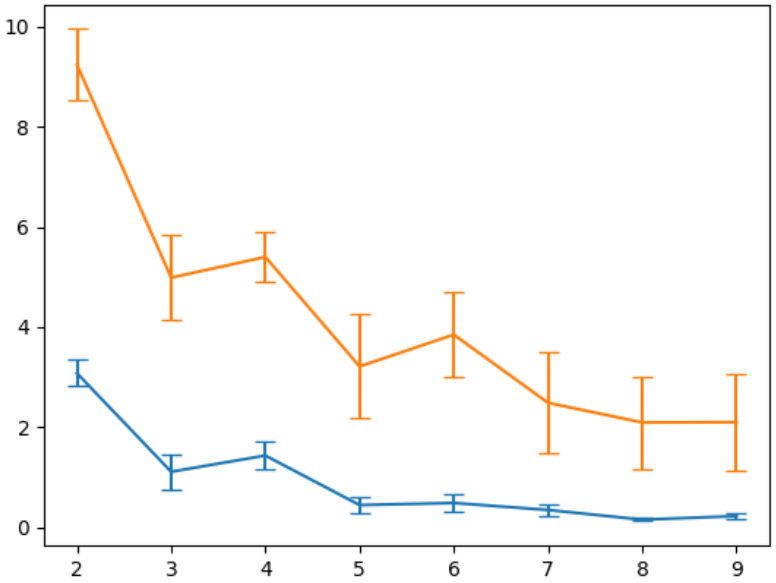
La valeur du coût est bien représentative de ce que l'on voit. A gauche, les nuages sont confondus et on a un coût de 3,5 alors qu'à droite les nuages sont légèrement séparés et le coût est de 16,2.

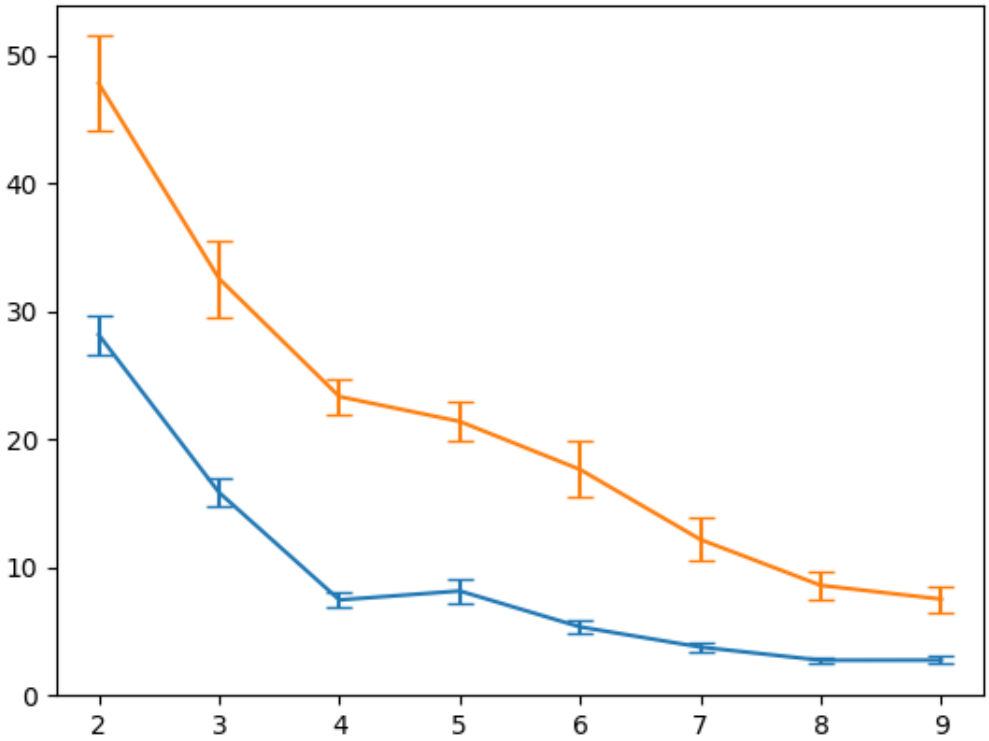
*à gauche une gaussienne, au centre une sphère et à droite deux sphères*

En utilisant un vecteur latent de taille 10, le réseau de neurones a parfois du mal à reconstruire les nuages.




Erreur en fonction de la taille du vecteur latent (train en bleu, test en orange)
(barres d'erreur à deux écarts-types sur une cross validation à 3 fold)
(à gauche sur une gaussienne, à droite sur une ou deux gaussiennes)
(environ 30 nuages, 50 points par nuage)




de même avec des nuages contenant une ou deux gaussiennes,
60 nuages, 288 points par nuage

Plus la taille du vecteur latent est grande, meilleurs sont les performances. Le réseau de neurones a toujours de meilleurs performances sur l'ensemble d'apprentissage. Avec une même taille de vecteur latent, il est normal que l'auto encodeur travaillant avec de plus gros nuages ait de moins bonnes performances.

4) Distances
Chamfer minimise
pour chaque P et MLP la distance au Q le plus proche.
Les MLP doivent bien atteindre un Q

pour chaque Q la distance au P MLP le plus proche
Les Q doivent bien être atteint par un MLP

5) Explication article
A partir de nuages de points, on génère des images 2D. A partir de ces images 2D, on peut entraîner un réseau pour deux tâches distinctes :
- reconstruire le nuage de point (c'est ce que fait PointNet)
- classifier les images (c'est ce que font ResNet et VGG)

Bien que notre but soit de reconstruire un nuage de point, on pourra essayer de reconstruire les nuages de points à partir des vecteurs latents issus des réseaux de neurones classifieurs.