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

Intro

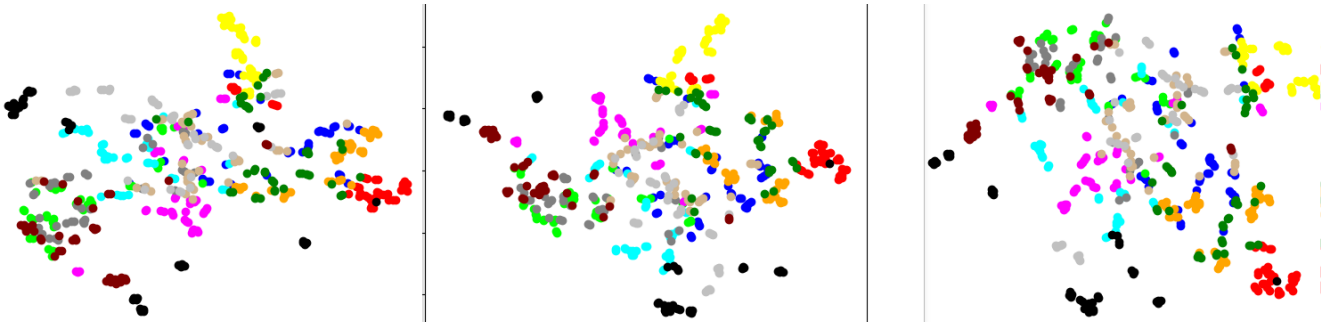
1) Article

bases : PyTorch, CNN, VGG, ImageNet, autres trucs mentionnés par Groueix
exécution du classifieur

2) tSNE

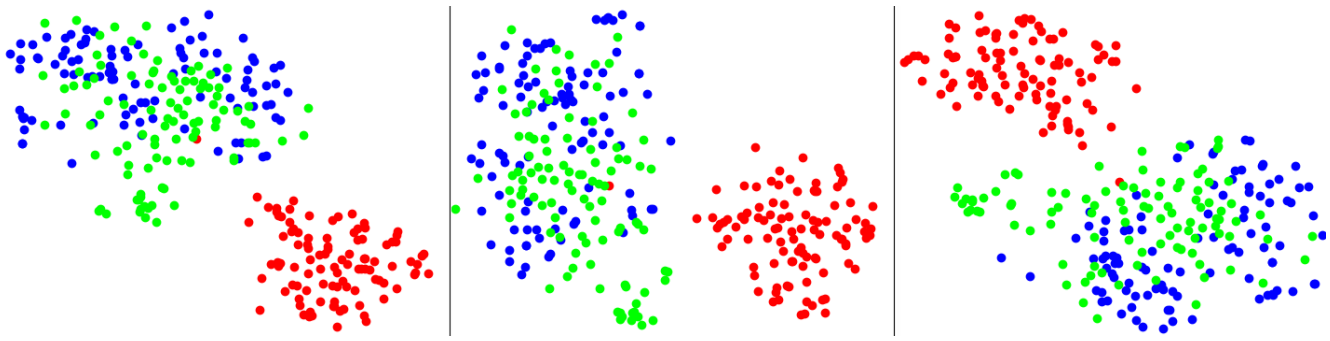
CNN => autoencoder ?
VGG est un réseau de neurones composé de 36 couches consécutives. Les 30 premières couches résument l'entrée en un plus petit vecteur, c'est la partie « feature extraction ». Les couches suivantes réalisent l'étape de classification à proprement parler.
La partie classification de 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.

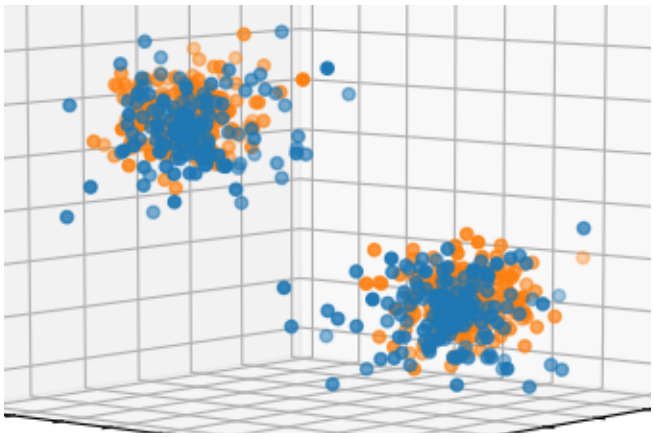
La taille de l'entrée du réseau étant fixe, tous nos nuages contiennent exactement le même nombre de point. Le nombre de points est fixé arbitrairement à la main (en général entre 50 et 360) de telle sorte que les formes soient bien reconnaissables à vue d'œil.

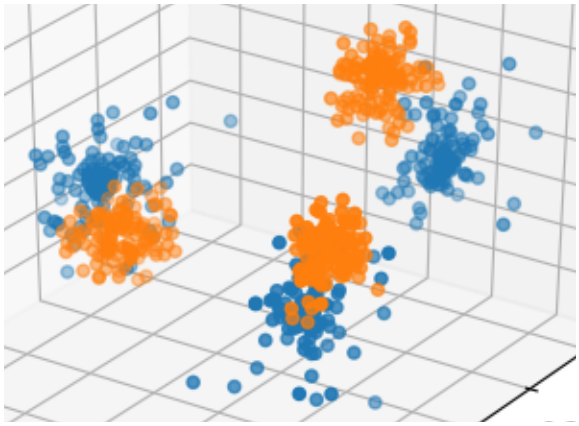
Pour chaque point du nuage initial, l'auto encodeur génère un unique point correspondant. On peut donc définir une fonction de coût définie comme la somme des écarts entre chaque couple de point. En élevant les écarts au carré, on obtient le coût des moindres carrés.

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 devient rapidement négligeable comparé aux moindres carrés.

Pour entraîner le réseau on a utilisé :

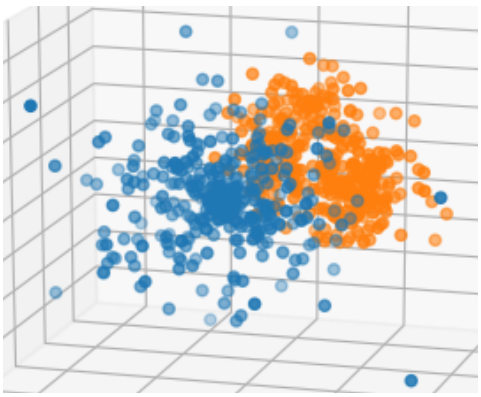
- 5 types de nuages différents. Un nuage de point est un ensemble de points échantillonnés aléatoirement sur :
    1. une gaussienne
    2. deux gaussiennes (avec exactement autant de point sur chacune des deux gaussiennes, et au total autant que dans le type précédent)
    3. trois gaussiennes (de même)
    4. une sphère
    5. deux sphères (de même)
- 10 nuages de chaque type. Chaque nuage n'est généré qu'une fois : une fois générés, ses points sont constants pour toutes les itérations de l'apprentissage.
- 60 points par nuages.
- Un vecteur latent de taille 10.

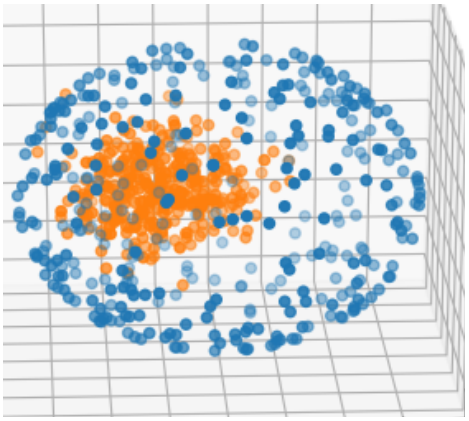


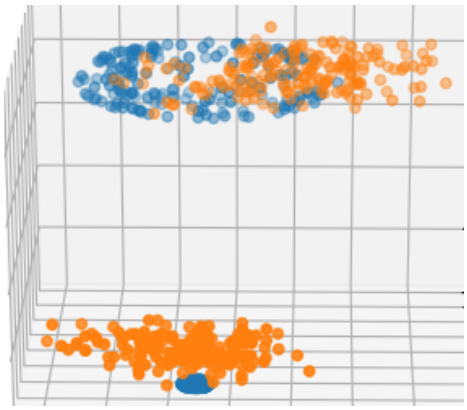


Nuages de point initiaux en bleu et nuages reconstruits en orange (2 gaussiennes à gauche et 3 à droite)

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.

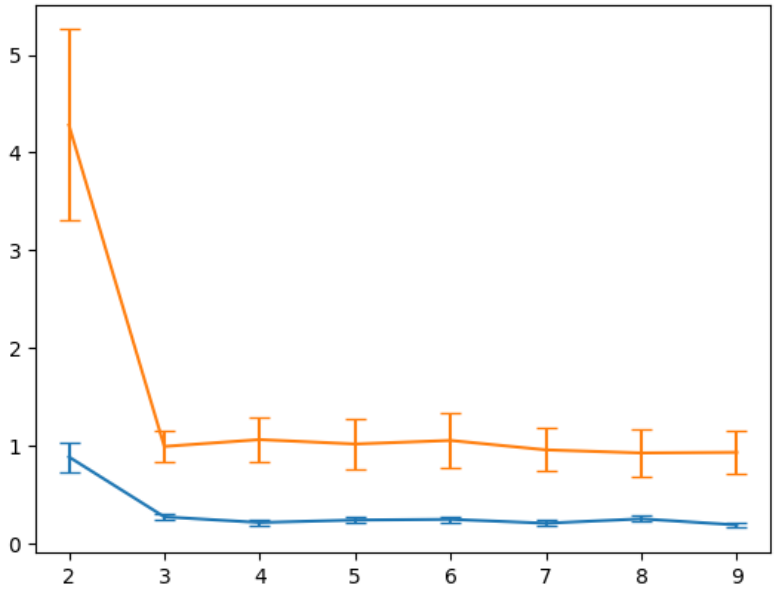


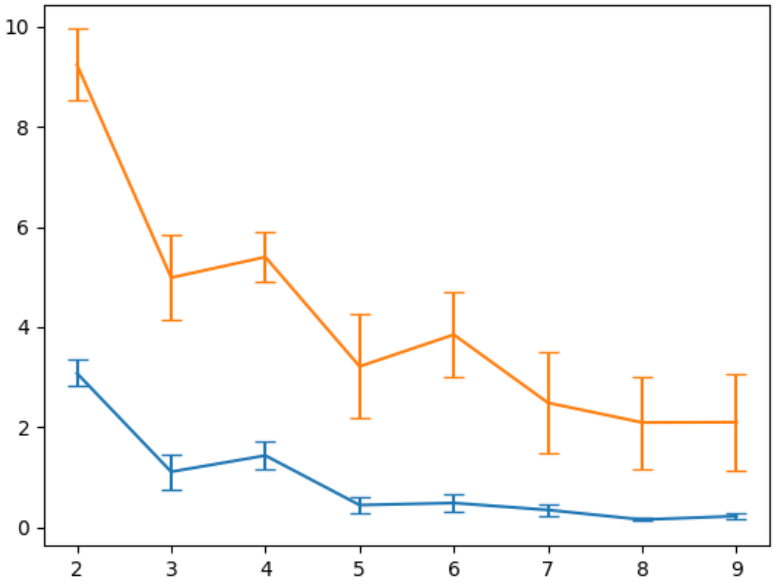




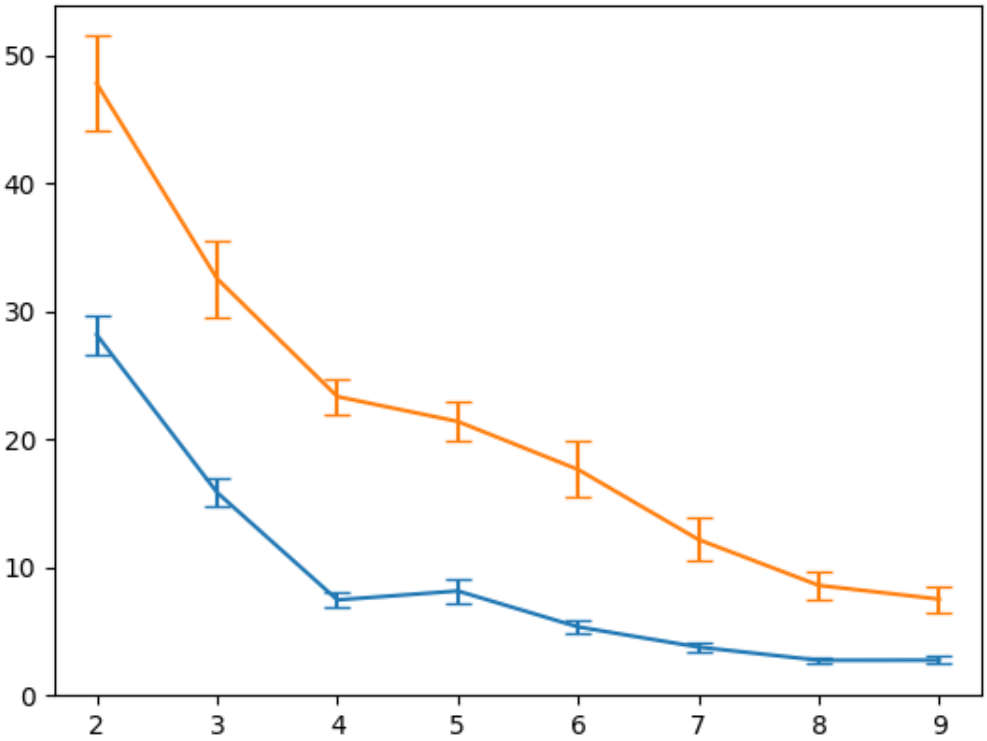
à gauche une gaussienne, au centre une sphère et à droite deux sphères

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



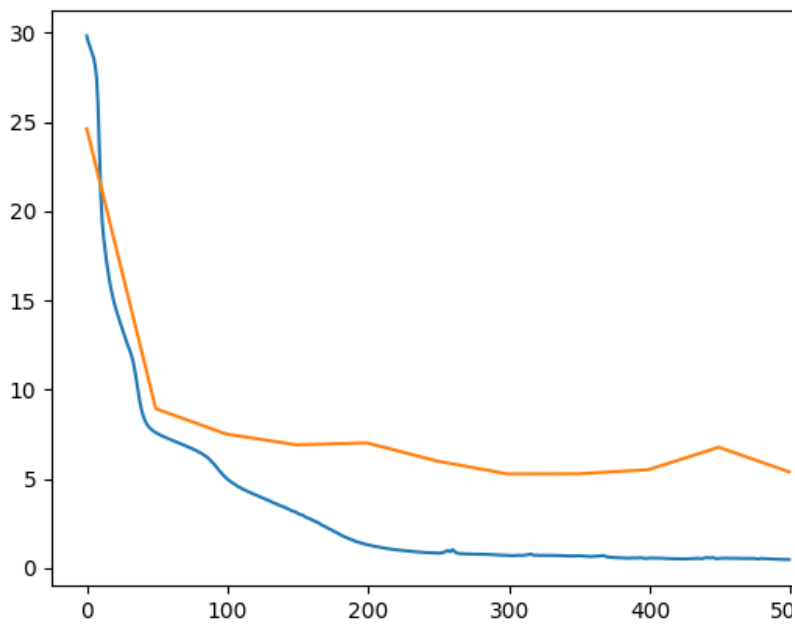


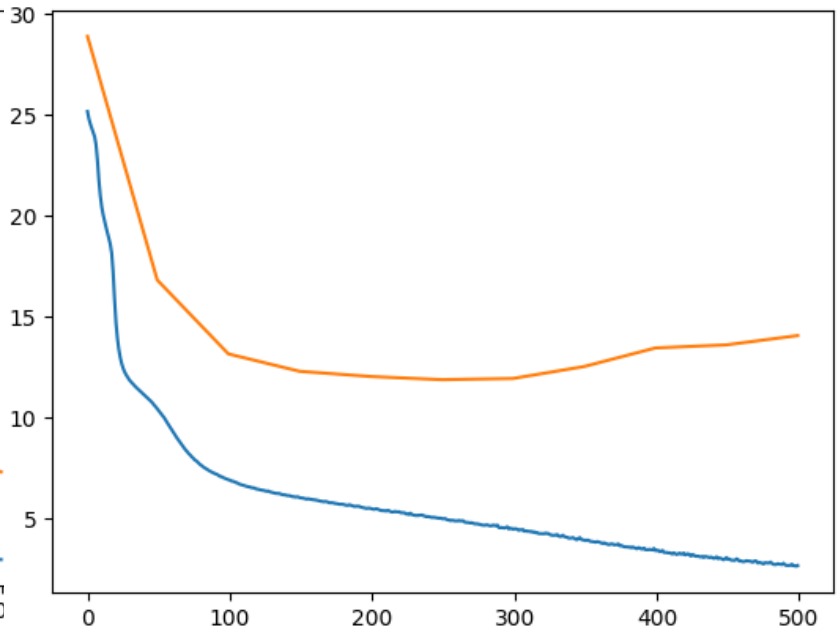
Erreur en fonction de la **taille du vecteur latent** (train en bleu, test en orange)
(barres d'erreur à deux écarts-types sur une cross validation à 3 fold)
(à gauche sur une gaussienne, à droite sur une ou deux gaussiennes)
(environ 30 nuages, 50 points par nuage)



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

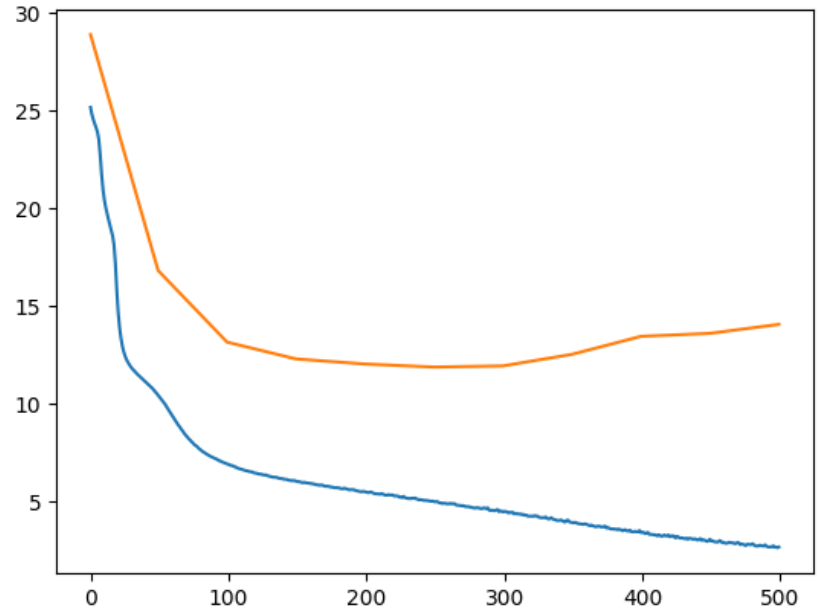
Plus la taille du vecteur latent est grande, meilleurs sont les performances. Le réseau de neurones a toujours de meilleurs performances sur l'ensemble d'apprentissage. Avec une même taille de vecteur latent, il est normal que l'auto encodeur travaillant avec de plus gros nuages ait de moins bonnes performances.

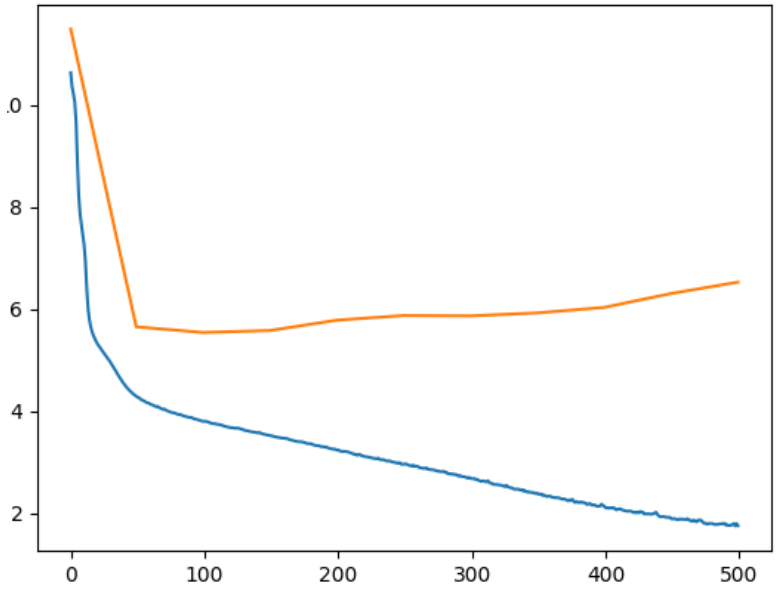


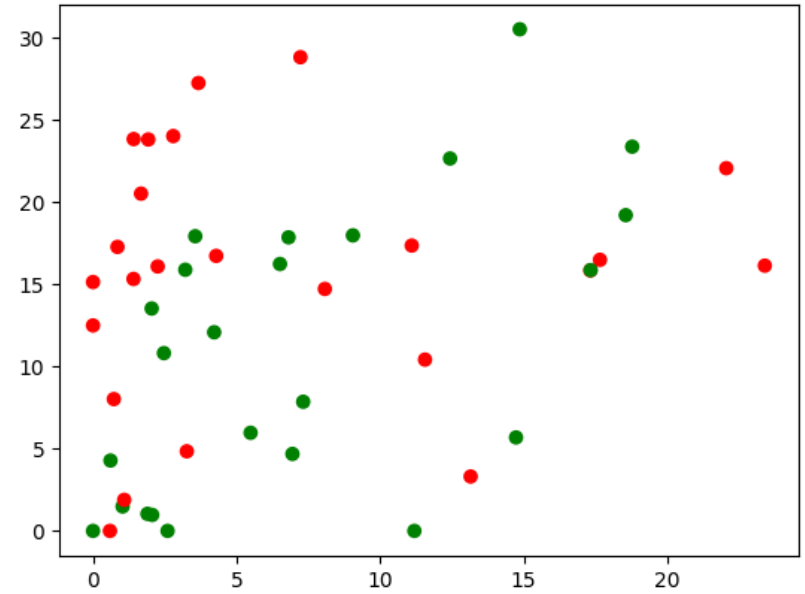


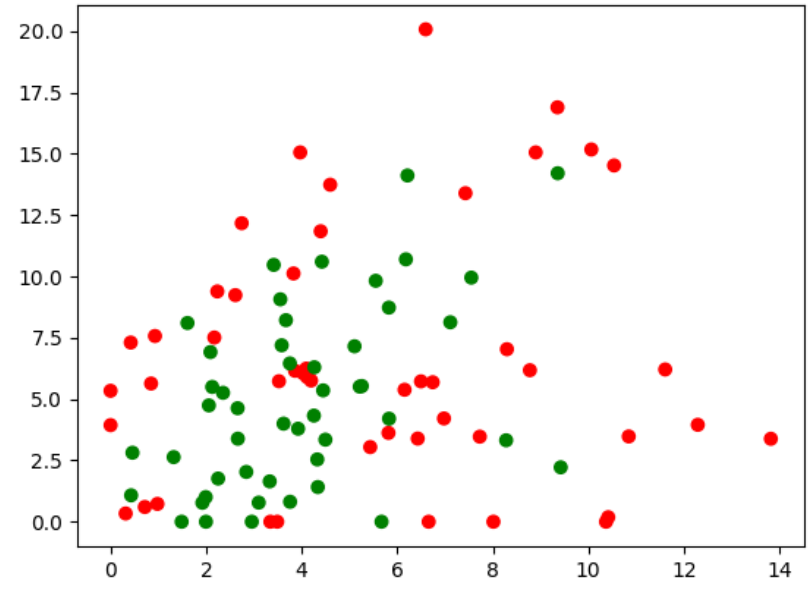
Loss train et test en fonction du nombre d’epochs

(60nuages, 50 pts, vecteur latent de taille 20 à gauche et 2 à droite)



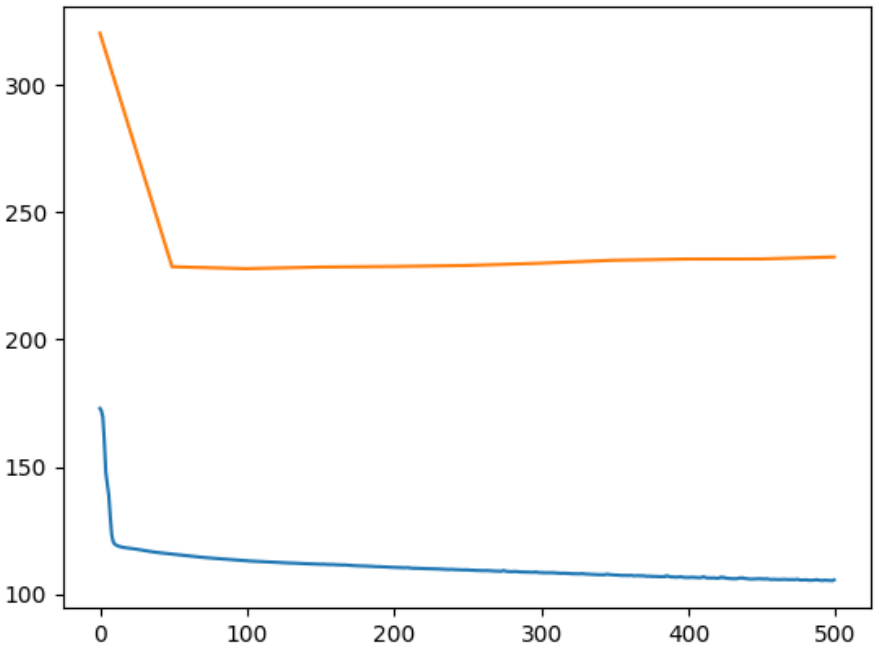


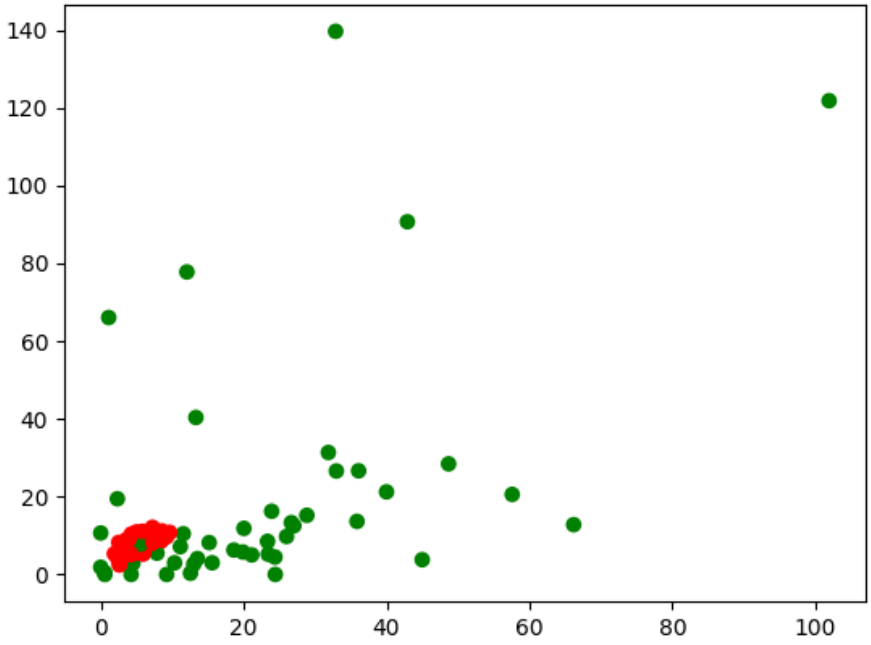


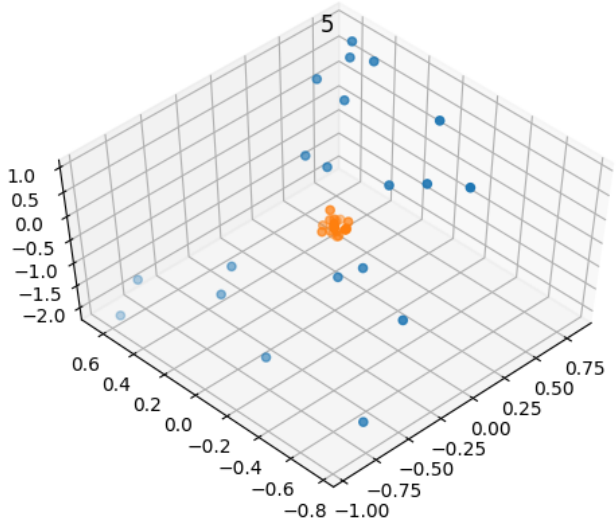


clustering sur train vecteur latent taille 2

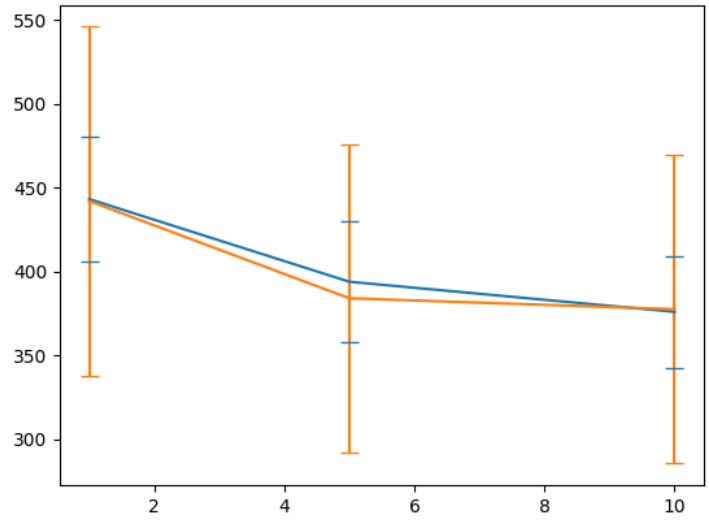
(60 nuages 50 pts à gauche, 120 nuages 20pts à droite)

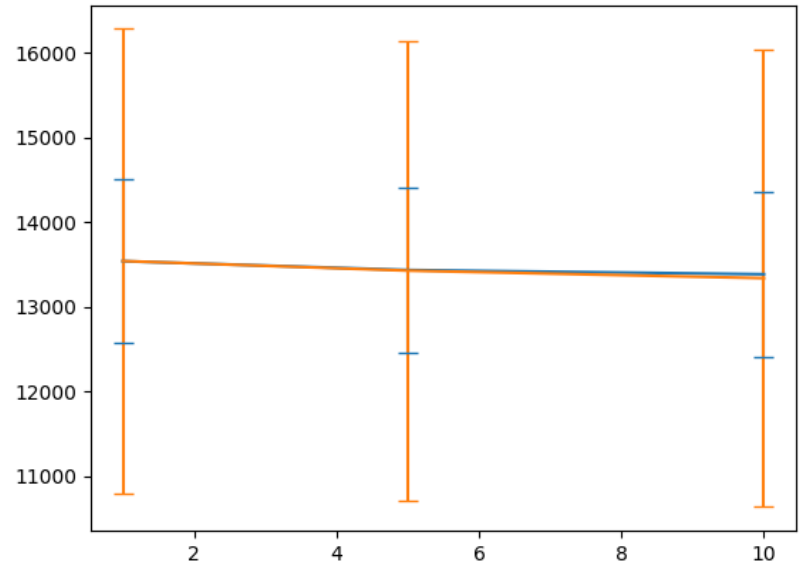

300
250
200
150
100
0
100
200
300
400
500


140
120
100
80
60
40
20
0
0
20
40
60
80
100

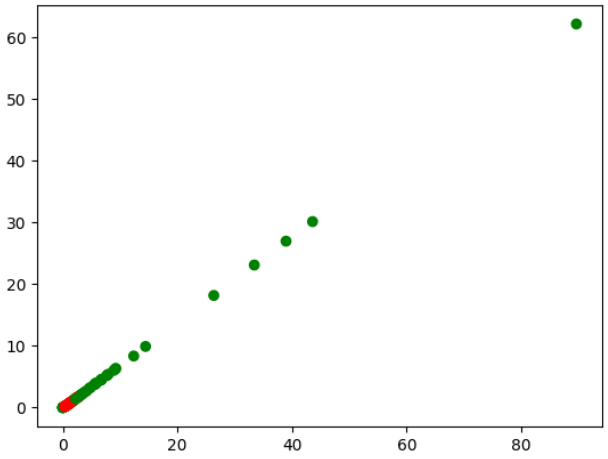

5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75

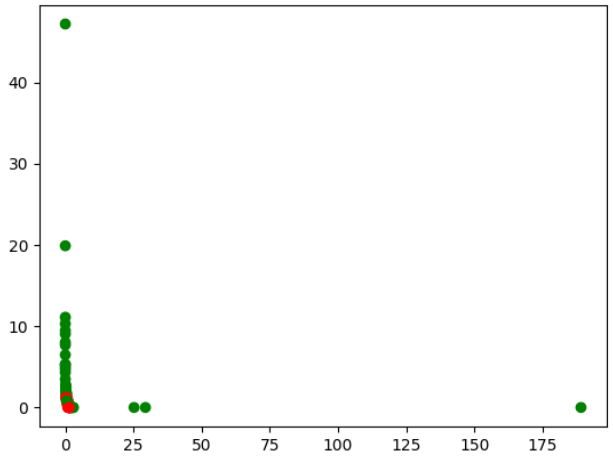
On essaye avec des plans mais ça ne marche pas mieux





loss en fonction du nombre d’epochs. Vecteur latent taille 2, 120 nuages (sphere/plan) 20 points. En relançant avec les mêmes paramètres, les résultats varient considérablement (dépend des nuages générés)





Visualisation du clustering (les sphères et les gaussiennes se regroupent toujours, les plans se distinguent un peu)

5) Distance de Chamfer

Chamfer minimise :

- pour chaque point de S*, la distance au P MLP le plus proche :

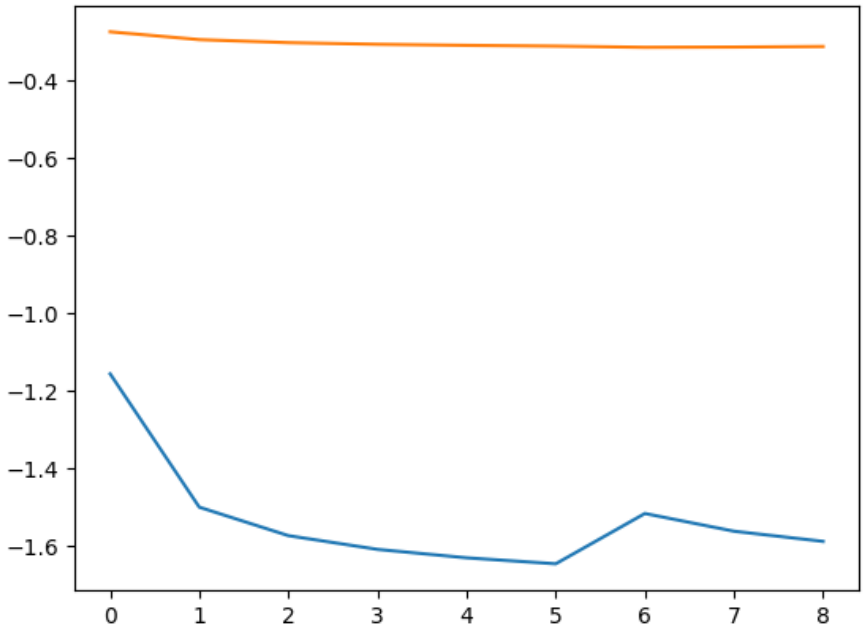
Les points de S* doivent bien être atteints par un point d'un MLP.

- pour chaque P et MLP, la distance au point S* le plus proche :

Tous les points des MLP doivent bien atteindre un point de S*.

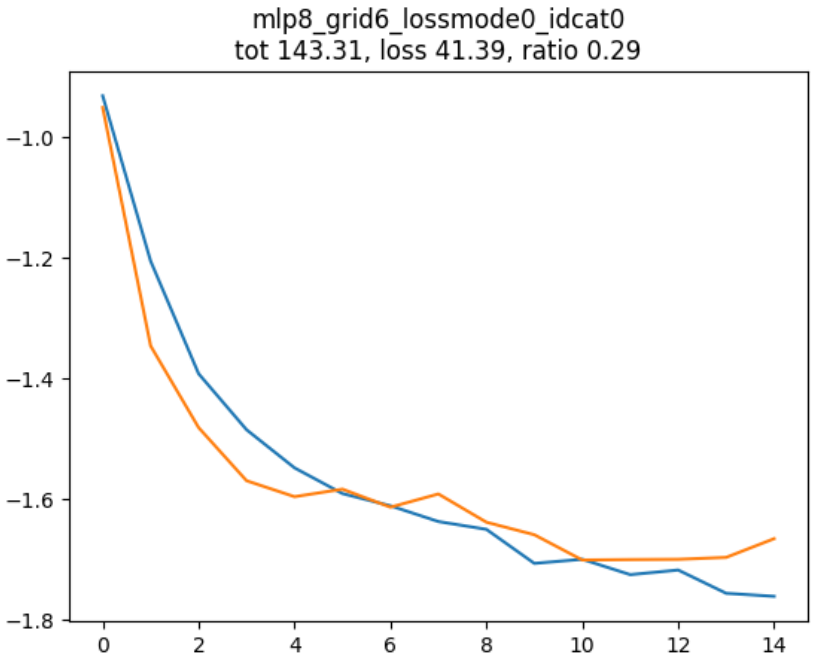
Une interprétation visuelle de l'effet du deuxième terme est qu'il fait se **déplacer les MLP vers la zone de S* la plus proche**. Avec une résolution de sampling moyenne (6x6), le deuxième terme **empêche les MLP d'atteindre des zones utiles** de la figure. Il **bloque la convergence dans des optimums locaux**.

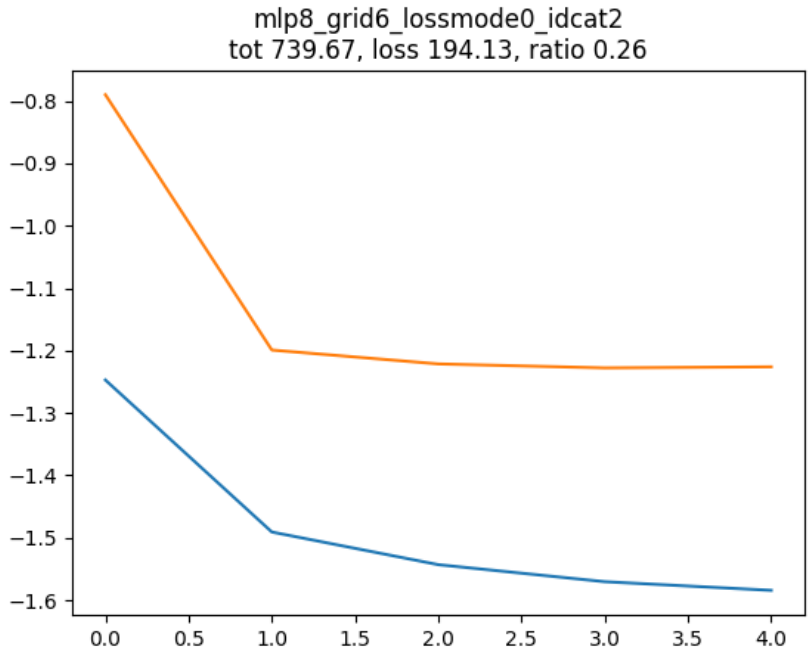
Sans le deuxième terme, si aucun des points d'un MLP n'est le plus proche d'un des points de S*, le MLP n'intervient dans aucune dérivée, il ne va donc pas bouger et le **MLP** va être « **oublié** » par le réseau de neurone. De plus, générer des **points loin de l'objectif** est visuellement une erreur mais, sans le deuxième terme, cela ne serait **pas pénalisé**.



Le train converge mais pas le test à l'epoch 6 on change de loss en train en augmentant la valeur du deuxième terme de la loss. On ne change pas la loss en test, cela n'a aucun effet sur le test et n'est en fait que du sur apprentissage.
(loss à partir de l'epoch 1)

On utilise environ le même nombre de point générés (8 * 6*6 = 288) et ground-truth (300)
Afin de pallier au sur-apprentissage on ré-échantillonne les nuages de ground truth à chaque itération





loss sur 4 boîtes à gauche converge car les boîtes ne sont tous à peu près que de simple rectange
loss sur 50 lampes à droite diminue car les points se répartissent dans l'espace mais on ne peut pas reconnaître les objets reconstruits en test.

6) 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.

7)

8) Affichage

9) **Autres méthodes de Reconstruction 3D**

3D reconstruction from multiple images / Stéréovision :

Il s'agit d'une technique qui permet d'obtenir une représentation 3D d'un objet ou d'une scène à partir d'un ensemble d'images 2D de l'objet voir la scène prises sous différents points de vue.
Basée sur le principe de triangulation : Les points visibles sur une image sont les projections des points réels qu'on peut situer sur des droites et la position dans l'espace réel 3D peut alors être obtenue par intersection de ces droites.
Cette technique pose différents problèmes notamment le problème de calibration (comment se projettent les points de l'objet/la scène sur l'image).
Le problème du matching, correspondant à la capacité de reconnaître et d'associer les points qui apparaissent sur plusieurs images.

Autres contraintes :

L'opacité : les objets observés ne doivent pas posséder des surfaces semi-transparentes pour que à chaque (x,y) ne correspondra qu'un seul point (x,y,z)
La cohérence photométrique : Idéalement , les objets doivent renvoyer la même intensité lumineuse dans toutes les directions

Shape from focus (defocus) :

Technique basée sur l'utilisation de l'effet de flou de dé-focalisation pour estimer la distance ou la forme des objets.
Contraintes : Dans ce cas là, les propriétés du système optique d'acquisition des images doivent être connues (ouverture de l'objectif, profondeur de champ),
** Si la profondeur du champ est finie, le flou dépend de la distance entre l'objet et la caméra **

Shape from shading :

Utilise la variation d'intensité dans l'image pour estimer la distance ou la forme des objets.
Contraintes : Les conditions d'éclairage doivent être connues.
L'intensité de la lumière réfléchie dépend de son incidence.

Reconstruction par space carving :

On reconstruit la forme 3D en éliminant des voxels qui ne se projettent pas sur la silhouette de l'objet.
Pour améliorer le rendement, il est possible d'associer aux voxels des propriétés comme la couleur .
L'inconvénient principal est que la qualité de la forme 3D reconstituée est liée directement au pas de discrétisation.