Nantes Université
Polytech Nantes

Master informatique, Visual computing

Deep Learning sur des données 3D

Présenté par:
Émilien BARBAUD
Julie QUEIROS

Encadrants:
Matthieu PERREIRA
Mona ABID

Janvier-Mai 2020
Équipe IPI - LS2N

Résumé
In recent years, with the growth in computing power of modern computers and the availability of a considerable amount of data, Deep Learning approaches have become increasingly popular. This has led to breakthroughs in many tasks : from speech recognition to image analysis. Knowing that one of the common points between a 2D image and a sound is that they remain "Euclidean" data. As a result, despite the phenomenal advances obtained in some fields, they remain nevertheless limited to "Euclidean" data. Nevertheless, in a multitude of disciplines, it is not uncommon to come across data that is not Euclidean and until recently Deep Learning operations did not apply to these data structures.

Keywords: 3D data, euclidean data, non-euclidean data, Deep Learning, Geometric Deep Learning

Introduction

Au cours des dernières années, la croissance de la puissance de calcul des ordinateurs modernes et la disponibilité d'une quantité considérable de données ont permis la montée en popularité des approches de Deep Learning. Cela a mené à des percées sur de nombreuses tâches comme la reconnaissance de la parole (*speech to text*) à l'analyse de l'image.
Que se soit une image 2D ou un son, cela reste des données dites euclidiennes. De ce fait, malgré les avancées phénoménales obtenues dans certains domaines, elles restent néanmoins cantonnées aux données euclidiennes. Néanmoins, dans une multitude de disciplines, il n'est pas rare de croiser des données non-euclidiennes, jusqu'à récemment les opérations de Deep Learning ne s'appliquaient pas à ces structures de données, plus difficiles à traiter.
Ce rapport va traiter du *Geometric Deep Learning*, une extension des techniques d'apprentissage profond aux données non-euclidiennes. De plus, dans notre cas, il faut que ces approches traitent des données non-euclidiennes brutes, c'est-à-dire sans avoir à transformer celles-ci en données euclidiennes, auquel cas cela ne serait qu'appliquer des approches déjà connues.

La première partie traitera des techniques des techniques d'apprentissage existantes, la deuxième partie expliquera les différentes façons de représenter des données tridimensionnelles, la troixième partie traitera des approches d'apprentissage profond sur des données 3D. La quatrième et dernière partie sera un benchmark de deux approches d'apprentissage profond sur les domaines 3D.

Table des matières

Table des figures

Première partie

Techniques d'apprentissage

La première partie de ce travail de recherche avait pour but de nous familiariser avec le *Deep Learning* (ou apprentissage profond). L'apprentissage profond fait partie de la famille des méthodes d'apprentissage automatique, capables d'apprendre par elles-mêmes, les distinguant d'un simple algorithme qui n'est que le reflet des conaissance d'un expert. Ces méthodes permettent l'apprentissage de modèles de données.

1 Machine Learning

L'apprentissage automatique est un champ d'étude de l'intelligence artificielle. Les algorithmes d'apprentissage machine construisent un modèle mathématique basé sur des données d'échantillon, connues sous le nom de "données d'entraînement", afin de faire des prédictions ou de prendre des décisions sans être explicitement programmées pour le faire. Il existe plusieurs modèles qui permettent de traiter des données afin de faire des prédictions, dont voici une liste non exhaustive :

1.1 Réseaux de neurones artificiels

Il s'agit d'un système schématiquement inspiré des neurones biologiques. Les réseaux de neurones (ou *Neural Network* en anglais) permettent l'apprentissage par l'expérience. À partir de situations ponctuelles déjà rencontrées, ils sont capables d'inférer une décision sur les nouveaux cas rencontrés.

Leur fonctionnement sera détaillé ci-après.

1.2 Arbres de décision

Un arbre de décision est un outil d'aide à la décision. Un arbre représente un ensemble de choix. Les feuilles de l'arbre représentent les différentes décisions possibles.

Pour prendre un décision, on part de la racine de l'arbre, puis en fonction des choix possibles à chauque nœud on remonte jusqu'au feuilles qui symbolisent la décision finale.

Les arbres de décisions sont utilisés en informatique pour de la fouille de données par exemple, mais peuvent aussi être utilisés dans d'autres domaines comme la médecine ou l'économie.

1.3 Régressions

Il s'agit d'un ensemble de méthodes statistiques pour analyser la relation entre plusieurs variables. On parle de problème de régression pour des variables quantitatives.

Certaines régressions comme la régression logistique peut être utilisée pour des problèmes de catégorisation, il s'agit alors de déterminer la probabilité d'appartenir à une certaine classe.

1.4 Réseaux bayésiens

Un réseau bayésien représente un ensemble de variables sous la forme d'un graphe orienté acyclique. De tels réseaux permettent de représenter de façon compacte les distributions de variables aléatoire.

2 Réseaux de neurones

Un réseau de neurones (*Neural Network*) est un graphe constitué d'objets élémetaires : les neurones artificiels. L'idée principale était de s'inspirer du fonctionnement d'un neurone biologique. Chaque réseau a des spécificités : neurones agencés par couches, présence de boucles de rétroaction, etc.

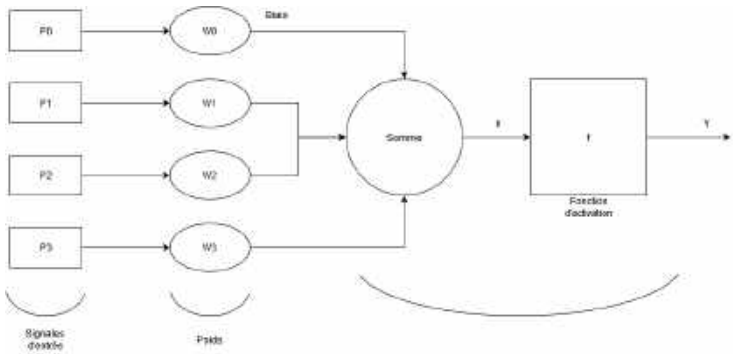


Figure 1. Un réseau de neurones simplifié

Les réseaux de neurones sont utilisés pour résoudre de nombreux problèmes. Ils fonctionnent de cette manière :

1. On organise les neurones en couche successives : chacun a une entrée (une information) issue des neurones des couches précédentes.
2. Ces informations sont pondérées et un biais leur est ajouté.
3. Elles sont additionnées.
4. Elles sont traitées par une fonction qui est là pour adapter la valeur de sortie sur une plage de valeurs particulières (c'est la fonction d'activation).
5. Cette valeur ainsi obtenue est transmise aux couches suivantes.

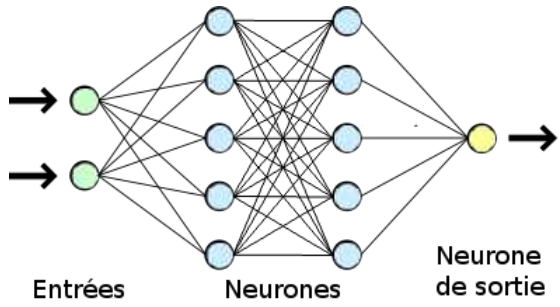


Figure 2. Un réseau de neurones

L'entraînement du réseau de neurones consiste à déterminer les valeurs de poids qui permettent aux couches de classer, d'extraire au mieux les données d'entraînement. On cherche à minimiser l'erreur du modèle, pour optimiser la fonction de perte, on va par exemple utiliser l'algorithme de rétro-propagation du gradient.

3 Tâches réalisées

De nombreuses tâches peuvent être réalisées par un réseau de neurones, voici une liste des tâches les plus courrantes.

3.1 Catégorisation

La catégorisation (ou *classification* en anglais) consiste à ranger dans des catégories prédéfinies les individus à classer, en se basant sur sur un ensemble d'individus dont la catégorie est connue. Il s'agit par exemple d'attribuer un courriel donné à la catégorie "spam" ou "non-spam", ou d'attribuer un diagnostic à un patient donné sur la base des caractéristiques observées du patient (sexe, tension artérielle, présence ou absence de certains symptômes, etc.)

La catégorisation est un apprentissage supervisé : on connait les classes a priori.

3.2 Classification

La classification (ou *clustering* en anglais) consiste à partitionner un ensemble d'individus en sous groupes ayant du sens. Afin d'optimiser le regroupement, on maximise la similarité intra-classe et on minimise la similarité inter-classe.

La classification est un apprentissage non supervisé : les classes sont déteminées en fonction des données d'entrée, on ne les connait pas a priori.

3.3 Régression

La régression consiste à trouver les relations entre plusieurs variables, si il y en a. Les problèmes de prédiction portant sur des variables quantitatives sont des problèmes de régression.

3.4 Segmentation

La segmentation d'image est un procédé permettant de diviser une image en plusieurs zones. Son but est de fournir une représentation plus simple à traiter et à analyser.

La segmentation d'image peut être supervisée ou non supervisée.

4 Apprentissage profond

Un objet peut être représenté de différentes manières par un vecteur de données. Les modèles d'apprentissage profond sont capables avec des données brutes non labellisées de reconnaître des représentations avec un niveau d'abstraction plus élevée, par exemple la reconnaissance d'un animal à partir d'une image.

La spécificité de ces modèles est leur profondeur, leur nombre important de couches leur permet d'extraire d'eux mêmes les caractéristiques (*features*) intéressantes à analyser. De cette manière, il n'y a même plus besoin d'avoir des connaissances spécifiques sur le domaine du problème à traiter, contrairement à une approche classique qui nécessite des données pré-traitées pour être efficace.

Avec ces méthodes, il sera possible de remplacer certains travaux laborieux et répétitifs par des modèles d'apprentissage supervisés ou non supervisés.

Il existe différents types d'architectures d'apprentissage profond, tels que :

4.1 Réseau neuronal convolutif (*CNN*)

Un réseau neuronal convolutif (*Convolutional deep Neural Network*) est un type de réseau profond dans lequel le motif de connexion entre les neurones correspond à des régions qui se chevauchent dans une image : on applique un filtre convolutionnel. Leur fonctionnement est semblable à notre façon de traiter les informations visuelles, grâce à l'empilement de plusieurs perceptrons.

Ces réseaux sont principalement utilisés dans la reconnaissance d'image, les systèmes de recommandations et dans le traitement du langage naturel.

Une section complète est dédié aux *CNN* ci-après.

4.2 Réseaux adverses génératifs (*GAN*)

Les réseaux adverses génératifs (*Generative Adversal Networks*) font partie des réseaux d'apprentissage non supervisés. Deux réseaux sont placés en compétition : le premier est un réseau générateur, et le second est un réseau discriminateur qui essaie de détecter si un échantillon est réel ou généré.

Ce type de réseau est utilisé en bio-informatique afin de créer de nouvelles molécules intéressantes pour soigner des maladies.

4.3 Long Short-Term Memory

Il s'agit d'un réseau de neurones récursif qui contrairement aux réseaux à propagation avant, utilise la rétropropagation. Il peut traiter non seulement des images, mais aussi des vidéos et des dialogues.

Ce type de réseau est utilisé dans la reconaissance du langage (*speech to text*).

4.4 Réseaux sabliers

Les réseaux sabliers (*hourglass networks*) est un type réseau de neurones profond utilisé notamment dans la segmentation d'objets. On les nomme ainsi car la forme du réseau fait penser à un sablier : la taille des couches se rétrécie de plus en plus jusqu'au milieu du réseau, pour ensuite grandir jusqu'à la sortie.

Parmi les réseaux de neurones sabliers on retrouve notament les FCN *(Fully Convolutional Networks)*, utilisés pour le traitement d'images et leur segmentation.

5 Fonctions d'activation

Ces fonctions permettent que chaque couche d'un réseau de neurones fonctionne de manière non-linéaire. Elles sont nombreuses et peuvent aussi être combinées afin d'obtenir une meilleure extraction des caractéristiques.

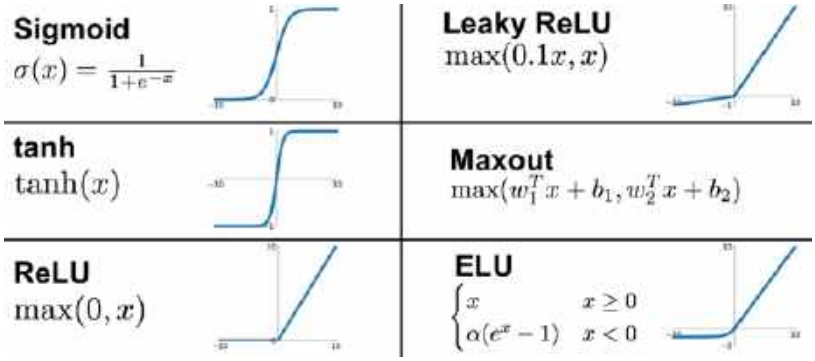


Figure 3. Exemple de fonctions d'activations

6 Réseaux de neurones convolutifs (*CNN*)

Les réseaux de neurones convolutionnels sont les méthodes principales permettant le traitement de données de type "signal", comme les images, les sons, etc. Ils permettent de capturer les invariants (temporels, géométriques).

De part leur domaine d'application, les réseaux de neurones convolutifs (*Convolutional deep Neaural Network*) sont parfaitement adaptés à notre sujet de recherche.

6.1 La convolution

On appelle **masque de convolution** une fenêtre coulissante dont la taille est inférieure à celle de la matrice d'entrée (l'image d'entrée). Cette fenêtre coulissante se déplace de gauche à droite et de haut en bas afin de parcourir l'image entièrement. Un calcul est effectué pour attribuer à chaque position du masque une seule valeur, ce calcul dépend du masque de convolution et de la valeur des pixels sous le masque.

6.2 Les filtres de convolution

Tout l'enjeu pour un *CNN* est de trouver les masques de convolution les plus intéressants afin d'extraire au mieux les caractéristiques des données d'entrée.

Par exemple, dans le cadre d'une image, un masque particulier va mettre en evidence des formes, des couleurs ou des contrastes distinctifs pour permettre de reconnaitre un objet précis.

6.3 Le réseau

Un *CNN* est un réseau de neurones avec certaines spécificités. C'est un réseau à propagation avant (*feedforward neural network*) ce qui signifie que l'information ne se déplace que dans un sens, vers l'avant. Il est aussi dénué de cycles.

Dans un *CNN* les premières couches de neurones fonctionnent différemment des autres : il n'y a pas de combinaison linéaire entre les entrées (par exemple des pixels) mais une opération de convolution. Cela agit comme un filtre pour ne garder que l'information utile.

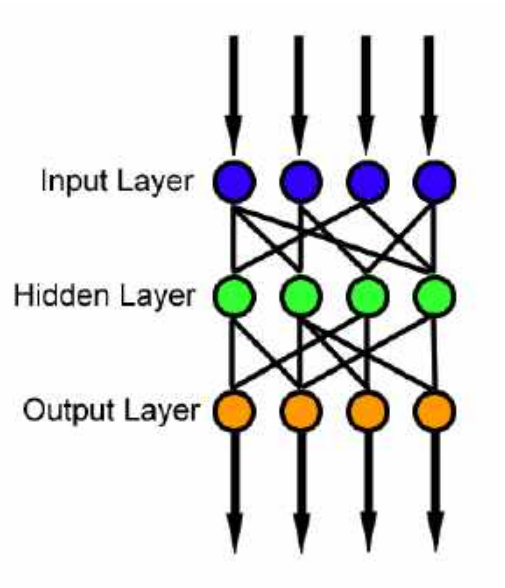


FIGURE 4. Un réseau à propagation avant

6.4 L'avantage des *CNN* dans le traitement de l'image

6.4.1 Les paramètres

Le nombre de paramètres dans un réseau de neurones augmente rapidement avec l'augmentation du nombre de couches. Cela peut rapidement créer un modèle lourd en termes de calculs (et parfois impossible). Régler un si grand nombre de paramètres peut être une tâche énorme. Le temps nécessaire pour régler ces paramètres est significativement réduit grâce aux *CNN*.

6.4.2 La dimentionalité

Les images ont une haute dimensionnalité étant donné que chaque pixel est considéré comme une caractéristique. On se retrouve rapidement avec de très grands réseaux impossibles à entrainer.

Les *CNN* réduisent la dimensionnalité grâce à la convolution : on s'intéresse étape par étape, à des petits morceaux de l'image. Les techniques de *pooling* appliquées entre les couches, permettent aussi de réduire la quantité d'information à traiter. Dans l'exemple du *Max-pooling*, on divise la taille de la matrice en gardant uniquement les valeurs les plus élevées sur chaque zone.

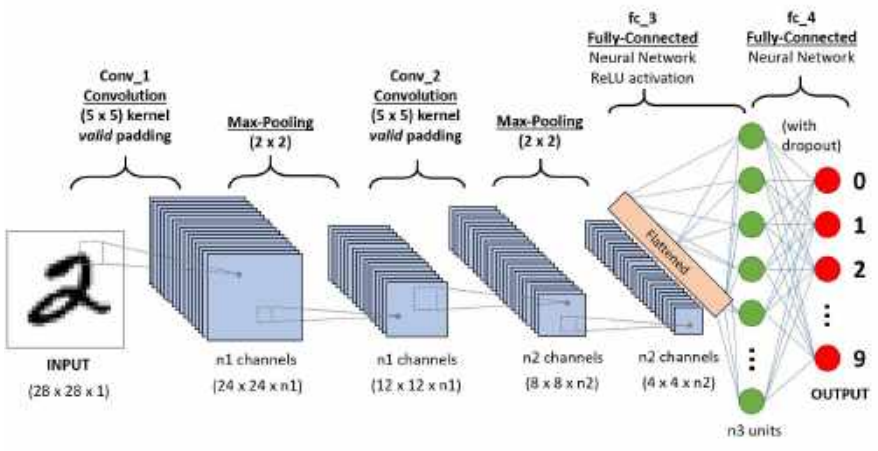


FIGURE 5. Un réseau convolutif

7 Limites

7.1 Quantité de données

L'apprentissage s'effectue sur de grandes quantités de données, il existe aujourd'hui des bases de données conséquentes pour les domaines les plus courants comme les images. Cependant, pour travailler sur des données moins courantes ou plus spécifiques, il devient difficile de trouver des jeu de données suffisamment grands pour entrainer le réseau.

7.2 Capacité de calcul

Il est nécessaire d'avoir une machine assez puissante pour effectuer l'entrainement des réseaux, qui peut durer de plusieurs heures à plusieurs jours. Cela est cependant de moins en moins problématique, avec des réseaux de plus en plus optimisés et des machines de calculs de plus en plus puissantes.

7.3 Boîte noire

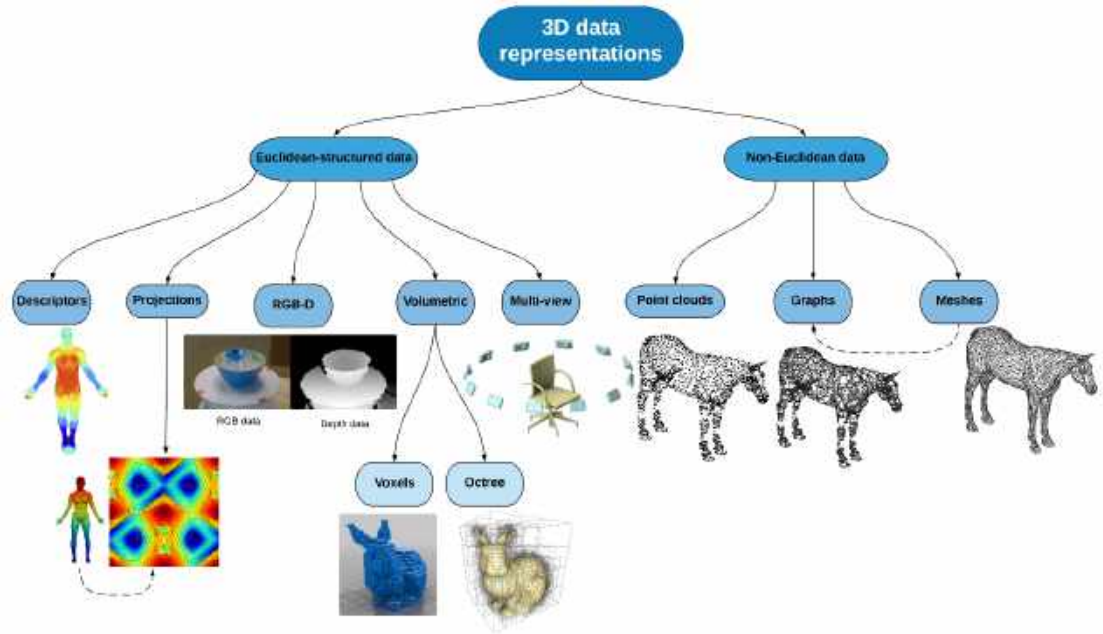
Les résultats d'un réseau de neurone sont dits "boîte noire" car il est souvent difficile de comprendre le fonctionnement des réseaux profonds une fois ceux-ci entrainés. Cela peut poser des problèmes si des tâches importantes de notre société lui sont confiées.

Deuxième partie

Contenus 3D

Ahmed *et al.* [1]

Il existe beaucoup de facons de représenter les données en trois dimentions, chacunes avec ses avantages et ses défauts. On peut séparer ces représentations en deux catégories : les représentations des données euclidiennes et non-euclidiennes.



1. E. Ahmed *et al.*, "Deep Learning Advances on Different 3D Data Representations: A Survey", *ArXiv* abs/1808.01462 (2018).

Figure 6. Représentations des données 3D

1 Représentations euclidiennes

Ces représentations sont dites euclidiennes car elles reposent sur une sctructure de géométrie euclidienne, c'est à dire qui repose sur la théorie des éléments d'Euclide.

1.1 Descripteur

D'une manière générale, les descripteurs de forme sont des représentations simplifiées d'objets pour décrire les caractéristiques géométriques ou topologiques de la forme 3D. Les descripteurs de forme peuvent être obtenus à partir de la géométrie, de la topologie, de la surface, de la texture ou de toute autre caractéristique de l'objet ou d'une combinaison de toutes ces caractéristiques.

1.2 Projection

La projection de données 3D dans un autre espace 2D est une autre représentation de données 3D brutes où la projection convertit l'objet 3D en une grille 2D avec des caractéristiques spécifiques. Le type de caractéristiques préservées dépend du type de projection. Il est possible par exemple de projeter les données 3D sur des domaines sphériques ou bien cylindrique, comme pour les cartes du monde.

Toutefois, ces représentations ne sont pas optimales pour les tâches complexes de vision par ordinateur en 3D, telles que la correspondance dense due à la perte d'informations lors de la projection RGB-D. Une représentation assez populaire notamment due à l'émergence du capteur Kinect de Microsoft.

Figure 7. Une Kinect

Les données RGB-D fournissent une information dite 2,5D sur l'objet 3D capturé en fournissant la carte de profondeur (D pour *Depth*) avec une information couleur 2D (RGB). Ainsi, on ajoute à chaque point 2D une information de profondeur. Les données RGB-D sont des représentations simples, mais efficaces d'objets 3D à utiliser pour différentes tâches telles que la reconnaissance d'identité, la régression de pose, la reconstruction de scènes et la correspondance. Le nombre d'ensembles de données RGB-D disponibles est conséquent comparé à d'autres ensembles de

données 3D tels que les nuages de points ou les maillages 3D. De plus, récupérer des données RGB-D peu couteux en calculs, ce qui en fait un candidat intéressant.

1.3 Données volumétriques

1.3.1 Voxels

Les données 3D peuvent être caractérisées comme une grille régulière dans l'espace tridimensionnel. Les voxels sont utilisés pour modéliser les données 3D en décrivant comment l'objet 3D est distribué à travers les trois dimensions de la scène. Malgré la simplicité de la représentation basée sur les voxels et sa capacité à coder des informations sur la forme 3D, elle souffre de certaines limitations contraignantes. La représentation par voxels n'est pas toujours efficace, car elle représente à la fois les parties occupées et non occupées de la scène, ce qui demande beaucoup de mémoire de stockage. C'est pourquoi la représentation par voxels n'est pas adaptée à la représentation de données à haute résolution.

Figure 8. Une voiture en voxel

1.3.2 Octrees

Une représentation volumétrique 3D plus efficace est celle à base d'octree, qui consiste simplement à faire varier la taille des voxels. La représentation en octree modélise les objets 3D sous la forme d'une structure de données hiérarchique. La représentation de l'octree est basée sur la décomposition récursive des voxels de la racine, similaire à la structure d'un arbre quaternaire. L'arbre se divise la environnement 3D en cubes qui sont soit à l'extérieur soit à l'intérieur de l'objet.

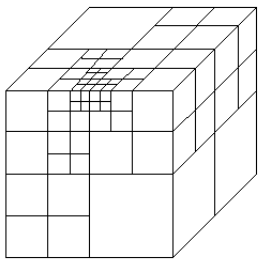


Figure 9. Un octree

Malgré la simplicité de formation des octrees 3D, ils sont efficaces pour représenter des détails précis des objets 3D par rapport aux voxels avec moins de calculs, puisqu'ils sont capables de partager la même valeur pour de grandes portions d'espace. Cependant, les représentations de voxels et d'octrees ne préservent pas la géométrie des objets 3D en termes de formes et d'aspect de la surface.

1.4 Multi vues

Les données 3D peuvent être présentées comme une combinaison de plusieurs images 2D, resultants de projections de l'objet 3D sous différents points de vue. Cela permet d'apprendre

plusieurs ensembles de caractéristiques afin de réduire l'importance du bruit, de la capture partielle et des problèmes d'éclairage sur les données capturées.

Cependant, représenter l'objet 3D avec un nombre de vues insuffisamment réduit risque de ne pas saisir les propriétés de la forme 3D dans son ensemble et de poser un problème de surajustement (*overfitting*). En outre, un nombre trop élevé de vues entraîne une surcharge de calcul inutile.

2 Représentations non euclidiennes

Ces représentations ne se basent pas entièrement sur une géométrie euclidienne. Il est par exemple théoriquement possible d'avoir plusieurs points au même endroit dans ces représentations.

2.1 Nuage de points

Un nuage de points (*pointcloud*) peut être considéré comme un ensemble non structuré de point 3D qui approxime la géométrie des objets 3D. La plupart des techniques d'apprentissage s'efforcent de capturer les caractéristiques globales d'un objet pour effectuer des tâches complexes telles que la reconnaissance, la correspondance, l'appariement ou la récupération.

Le principal avantage des nuages de points est la quantité de données disponibles, en effet grâce aux technlogies actuelles, il est possible de numériser facilement des lieux ou des objets pour en faire des nuages de points.

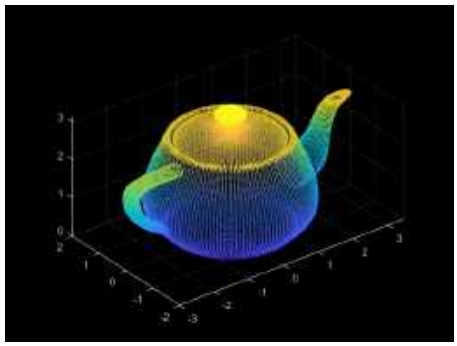


Figure 10. Un nuage de point représentant une théière

Leur traitement reste cependant une tâche difficile en raison de certains problèmes liés à leur manque de structure. Ces problèmes apparaissent généralement en raison de l'absence d'informations de connectivité dans les nuages de points, ce qui entraîne une ambiguïté sur les informations de surface. On peut aussi citer l'odre dans lequel les points sont inscrits dans les fichiers, qui en soit ne change pas l'objet en lui même, mais qui est un obstacle pour le traitement.

2.2 Graphes et Maillages 3D

Une structure de maillage 3D (*3D mesh*) consiste en un ensemble de formes géométriques de base (*primitives*), décrites par ensemble de sommets (*vertices*) indiquant les coordonnées du maillage. Ces sommets sont associés à une liste de connectivités qui décrit comment ils sont reliés entre eux. La géométrie locale des mailles peut être caractérisée comme un sous-ensemble de l'espace euclidien suivant les données structurées en grille. Comme pour les nuages de points,la structure globale des maillages 3D est non-euclidienne.

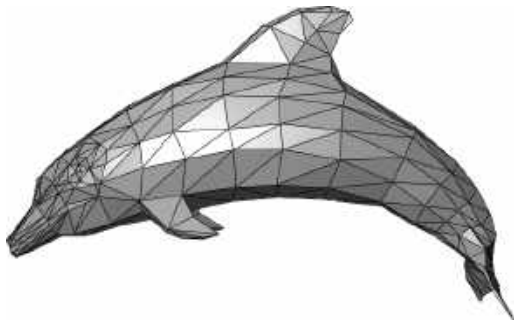


FIGURE 11. Un maillage représentant un nuage

Les maillages 3D peuvent également être présentés sous forme de données structurées par un graphe où les nœuds correspondent aux sommets du maillage et les bords représentent la connectivité entre ces sommets. Ces graphes peuvent être orientés ou non.

Troisième partie

Apprentissage profond sur les objets 3D

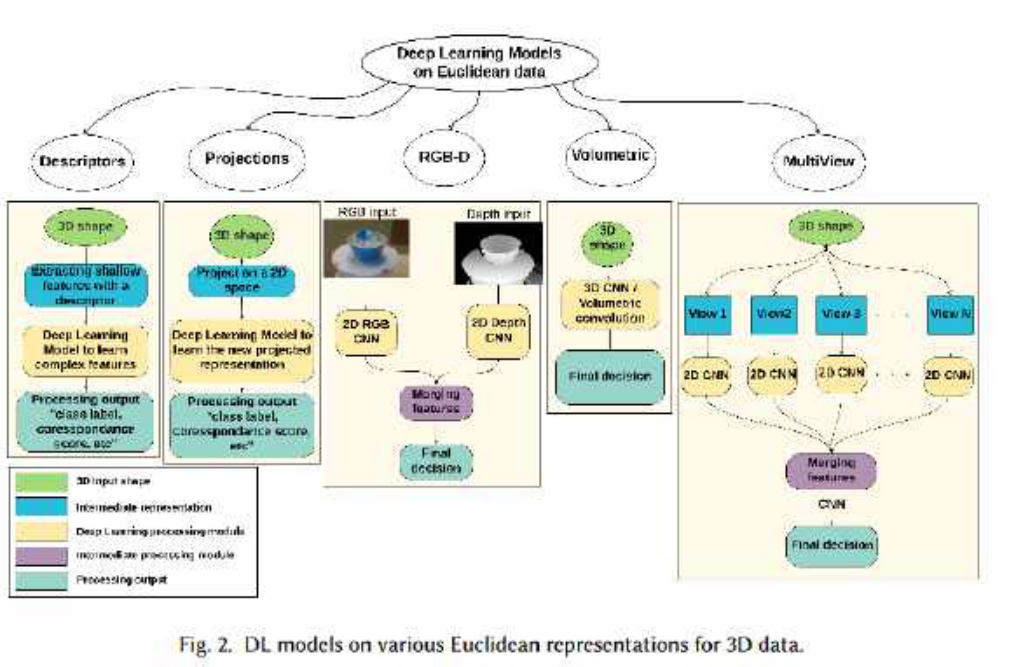


Fig. 2. DL models on various Euclidean representations for 3D data.

FIGURE 12. Modèles de Deep Learning appliqués aux données euclidiennes

1 Différentes approches

Il existe plusieurs approches pour traiter les données 3D avec du *Deep Learning*, elles dépendent du type des données d'entrée (certaines acceptent plusieurs types de données, d'autres sont plus spécifiques) et du type de tâche à effectuer (classification, segmentation, etc.).

1.1 PointNet et PointNet++

La structure point cloud est un type de représentation dit non-euclidien. Du fait de sa structure irrégulière, certaines méthodes vont passer par l'étape de transformation des données pour obtenir

une représentation regulière, comme les données volumétriques (*voxels*) ou les multi-vues. Le problème avec cette approche est la taille des données, qui augmente considérablement. Cela cause des erreurs lors du traitement, telles que l'apparition d'artefacts sur le rendu final.

1.1.1 PointNet

C. R. Qi *et al.*, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", cite arxiv:1612.00593, 2016, http://arxiv.org/abs/1612.00593

Comme dit précédemment, le réseau PointNet prends des nuages de points en entrée. PointNet permet de réaliser trois types de tâches sur les données : de la catégorisation (*classification*), de la segmentation d'objets (*part segmentation*) et de la segmentation sémantique (*semantic segmentation*).

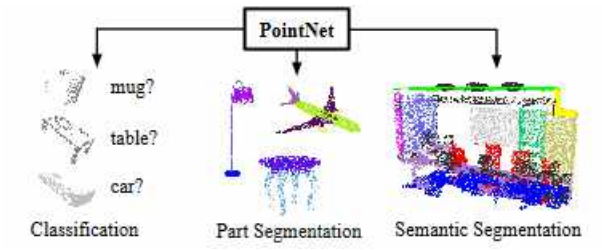


FIGURE 13. Tâches effectuées par PointNet

1.1.2 PointNet++

C. R. Qi *et al.*, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", *arXiv preprint arXiv:1706.02413*, 2017,

Pointnet++ est une approche basée sur Pointnet, qui lui ne parvient pas à comprendre la structure locale et à la généraliser à des scènes complexes. Pointnet++ est un réseau hiérarchique qui applique Pointnet récursivement sur une portion du nuage de points d'entrée.

Comme les *CNN*, Pointnet++ extrait des caractéristiques locales d'une petite zone et les regroupe en unités plus grandes, puis les traite pour produire des caractéristiques de plus haut niveau.

1.2 Réseaux de neurones géodésiques convolutifs

J. Masci *et al.*, "Geodesic Convolutional Neural Networks on Riemannian Manifolds.", 2015, 832-840, http://dblp.uni-trier.de/db/conf/iccvw/iccvw2015.html#MasciBBV15

Des chercheurs ont tenté une approche en créant un GCNN (*Geodesic Convolutional Neural Network*). L'idée est d'utiliser les distances et les angles pour comparer les points entre eux. L'espace est modifié pour créer des coordonnées polaires géodésiques sur plusieurs zones de la surface de l'objet.

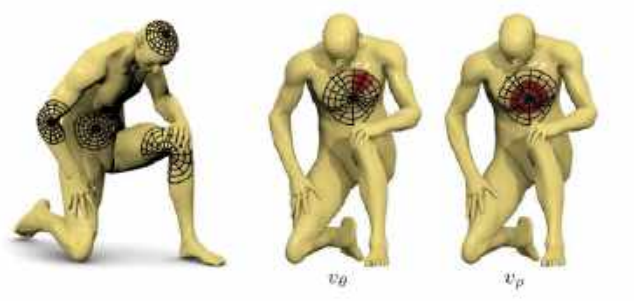


FIGURE 14. Construction de coordonnées polaires géodésiques locales sur un collecteur. A gauche : exemples de parcelles géodésiques locales, au centre et à droite : exemples de coordonnées angulaires et radiales

1.3 Réseau de neurones anisotrope convolutif

D. Boscaini *et al.*, “Learning shape correspondence with anisotropic convolutional neural networks”, sous la dir. de D. D. Lee *et al.*, 2016, 3189-3197, http://papers.nips.cc/paper/6045-learning-shape-correspondence-with-anisotropic-convolutional-neural-networks.pdf

Les réseaux neuronaux convolutionnels ont obtenu des résultats extraordinaires dans de nombreuses applications de vision par ordinateur et de reconnaissance des formes. Cependant, leur adoption est limitée en raison de la structure non euclidienne de leurs données. Cette approche propose un ACNN (*Anisotropic Convolutional Neural Network)*, une généralisation des *CNN* classiques aux domaines non euclidiens.

Les convolutions classiques sont remplacées par des projections sur un ensemble de noyaux de diffusion anisotrope orientés. Cela permet d’apprendre efficacement les correspondances intrinsèques entre les formes ressemblantes. Les résultats sont très encourageants même sur des données difficiles.

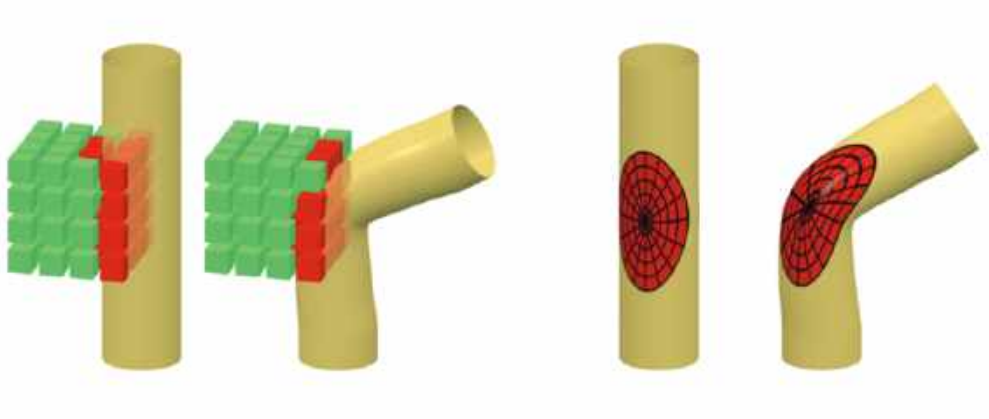


Figure 15. Illustration de la différence entre les méthodes d’apprentissage approfondi extrinsèques (à gauche) et intrinsèques (à droite) sur les données géométriques. La transformation en données euclidiennes a permi de rendre la forme invariante aux distortions.

1.4 SPNet

M. Yavartanoo, E. Kim et K. M. Lee, “SPNet: Deep 3D Object Classification and Retrieval Using Stereographic Projection.”, sous la dir. de C. V. Jawahar *et al.*, Lecture Notes in Computer Science, 11365 (2018) : 691-706, http://dblp.uni-trier.de/db/conf/accv/accv2018-5.html#YavartanooKL18

Cette approche est basée sur un réseau neuronal de projection stéréographique (*Stereographic Projection Neural Net-work*) ou SPNet, efficace pour l’apprentissage des représentations d’objets en 3D.

Dans un premier temps, une projection 2D est réalisée sur l’objet 3D en utilisant la projection stéréographique. Cette image est ensuite passée dans un réseau neuronal convolutionnel 2D peu profond (*CNN*) pour estimer la catégorie d’objet, puis un ensemble de vues, suivi par un ensemble de vues, qui combinent les réponses de plusieurs vues de l’objet pour améliorer encore les prédictions.

Plus précisément, l’approche proposée consiste en quatre étapes :

1. Projection stéréographique d’un objet 3D
2. Apprentissage de la perception de la vue
3. Sélection de la vue
4. Ensemble de vues

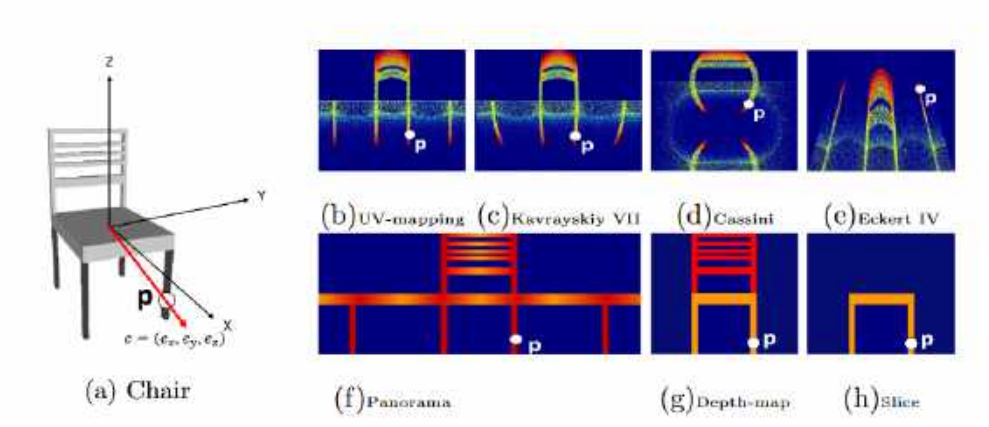


FIGURE 16. Projections stéréographiques d'une chaise

Cette approche est comparable aux méthodes existantes tout en ayant une mémoire *GPU* et des paramètres de réseau nettement inférieurs. Malgré sa légèreté, les expériences sur la classification d'objets 3D et les ré-essais de formes ont montré de bonnes performances.

1.5 LightNet

S. Zhi *et al.*, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning", *Comput. Graph.* 71 (2018) : 199-207

Cette méthode promet une reconnaissance d'objets 3D en temps réel, c'est-à-dire en moins de 6 *ms* selon les auteurs. Pour prédire le type et l'orientation de l'objet, le modèle LightNet est utilisé.

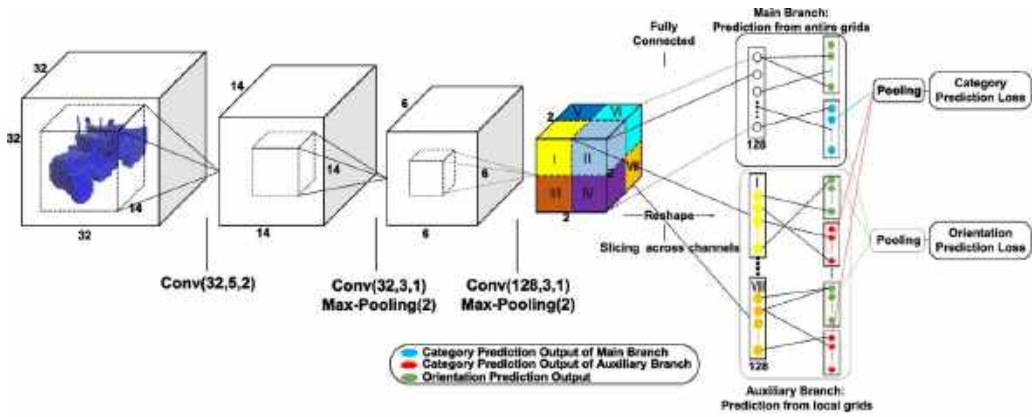


FIGURE 17. Architecture de LightNet

1.5.1 Traitement de données en amont

Pour faire marcher le modèle, en premier lieu les données 3D (qui sont dans ce cas représentées sous forme de nuages de point ou bien de maillage) sont converties en structure données régulières, ici une grille de *voxels*. Chaque *voxel* a une valeur d'occupation binaire, 1 lorsque qu'il se trouve à l'intérieur de l'objet, 0 s'il est à l'extérieur. Chaque objet transformé en grille de *voxels* a une taille de $24 \times 24 \times 24$. Une zone de *voxels* vides est ajoutée tout autour de l'objet afin d'améiorer les convolutions et réduire les artéfacts, ce qui amène l'objet à une taille de $32 \times 32 \times 32$.

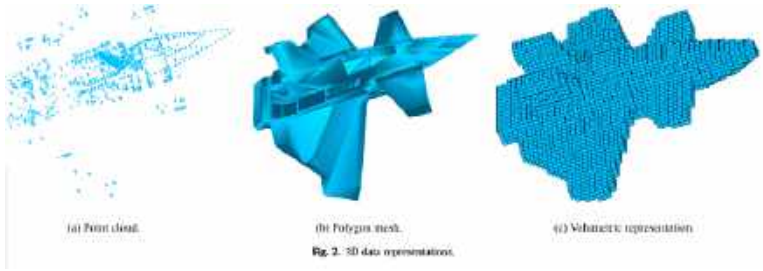


Figure 18. Processus de voxelization des données.

Quatrième partie

Benchmark des méthodes

Dans cette partie, nous allons nous attacher à présenter l'outil utilisé pour mettre en oeuvre le benchmark ainsi que les résultats obtenus. Le benchmark ne s'appliquera qu'à deux méthodes axées sur les nuages de points : PointNet et PointNet++.

1 Outil utilisé

PyTorch Points 3D

Pour réaliser le benchmarck nous avons utilisé le framework Torch Points3d basé sur Pytorch-Geometric. ("Documentation torch points3d", https://torch-points3d.readthedocs.io/en/latest/)

Ce framework permet de faire tourner des modèles de Deep Learning utilisés sur les nuages de points. Il permet notamment de faire de l'analyse sur les tâches (segmentation, classification) et va nous permettre de comparer les deux différents modèles.

2 Mise en oeuvre

Spécification de l'ordinateur sur lequel sont effectués les prochaines manipulations.

Grapics :	Card-1 : Intel Device 3e9b bus-ID : 00 :02.0
	Card-2 : NVIDIA Device 1f91 bus-ID : 01 :00.0
	Display Server : x11 (X.Org 1.19.6) drivers : nvidia FAILED : modesetting
	Resolution : 1920x1080@60.03hz
	OpenGL : renderer : GeForce GTX 1650 with Max-Q Design/PCIe/SSE2
	version : 4.6.0 NVIDIA 440.64 Direct Render : Yes
OS :	Ubuntu 18.04 Bionic Beaver
Python :	Environnement virtuel python 3.6.10
Cuda :	> 10

La tâche à effectuer sera une tâche de segmentation.

2.1 Dataset utilisé pour l'entrainement des deux réseaux :

2.1.1 Shapenet

17,000 nuages de point 3D séparé en 16 catégories :

— "Airplane"

— "Bag"

— "Cap"
— "Car"
— "Chair"
— "Earphone"
— "Guitar"
— "Knife"
— "Lamp"
— "Laptop"
— "Motorbike"
— "Mug"
— "Pistol"
— "Rocket"
— "Skateboard"
— "Table"

2.1.2 Échantillon de Shapenet

Dans cet échantillon, n'est gardé que la catégorie "Cap". De ce fait, le dataset est significativement plus petit et donc plus facile et rapide pour commencer à travailler.
En effet, nous avons entraîné les réseaux avec une petite partie de ShapeNet afin d'être sûr que l'outil pourrait fonctionner et aboutirait, en sachant qu'avec l'intégralité de ShapeNet certains entraînements sont assez long.
Il est notamment possible de modifier la catégorie ou les catégories sur lesquelles on fait s'entraîner le modèle, pour se faire il suffit de modifier le fichier de configuration "shapenet.yaml", comme présenté dans la figure 20 :

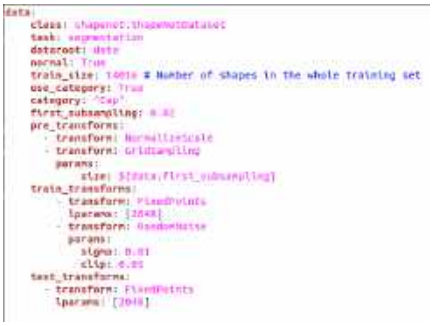


FIGURE 19. Fichier de configuration Shapenet-Cap

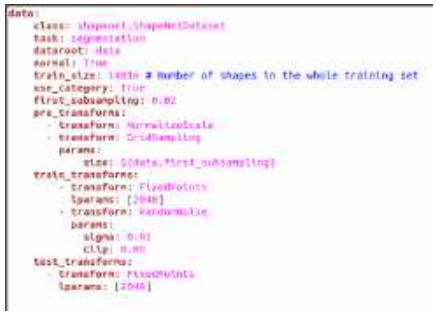


FIGURE 20. Fichier de configuration Shapenet

Il suffit simplement de changer le paramètre catégorie avec le nom de la catégorie qui vous intéresse (si vous vous voulez ne travailler que sur une seule catégorie).

2.2 Résultats

Dans la section qui va suivre, les résultats des expérimentations seront reportés. Il y aura donc les temps des expérimentations (en général une moyenne sur plusieurs expérimentations), les métriques la Mean Class IoU et la Mean Instance IoU. A la fin de la phase d'entraînement, les métriques sont gardées comme ceci d'après la documentation du framework :

```
DEFAULT_METRICS_FUNC = {
"iou": max,
"acc": max,
```

```
    "loss": min,
    "mer": min,
}
```

Ces résultats ont pour but de donner un ordre d'idée du temps qu'il faut pour réentrainer un réseau en utilisant le framework.
Dans ce cas, nous ne l'avons fait que sur ShapeNet mais il est tout à fait possible d'utiliser un autre dataset. Il est aussi possible d'utiliser d'autres modèles, dans notre cas nous avons utilisé PointNet et PointNet++.

2.2.1 Résultats sous forme de tableau :

Tous les temps seront exprimés en seconde.

2.2.1.1 PointNet sur Shapenet-Cap :

Nombre d'époch	10 epoch	50 epoch	100 epoch
Temps median	9.34	29.33	53.36
Temps min	8.97	27.84	52.14
Temps max	13.26	29.72	125.13
CMIOU	63.09	68.23	75.81

2.2.1.2 PointNet sur Shapenet :

Nombre d'époch	10 epoch	50 epoch	100 epoch
Temps median	324.33	1442.86	2664.63
Temps min	313.41	1382.69	2653.83
Temps max	326.49	1495.42	2675.42
CMIOU	63.22	68.68	67.32
IMIOU	69.34	71.96	71.76

2.2.1.3 PointNet++ sur Shapenet-Cap :

Nombre d'epoch	10 epoch	50 epoch	100 epoch
Temps median	25.31	107.568	208.450
Temps min	23.01	98.85	185.26
Temps max	33.93	117.68	228.54
CMIOU	54.51	88.33	88.13

2.2.1.4 PointNet++ sur Shapenet :

Nombre d'époch	10 epoch	50 epoch	100 epoch
Temps	3049.653	16306	32411.41
CMIOU	75.46	81.96	82.66
IMIOU	81.28	84.78	85.02

2.2.2 Résultats pour 100 epoch sous forme de graphique :

La partie suivant présente des répresentations graphiques de l'évolution des métriques au cours de la phase d'entraînement (suivi de la phase de test) pour 100 epoch. Il est surtout intéressant de noter qu'il est possible, grâce au framework, d'avoir accès sous cette forme aux métriques.

2.2.2.1 PointNet sur Shapenet :

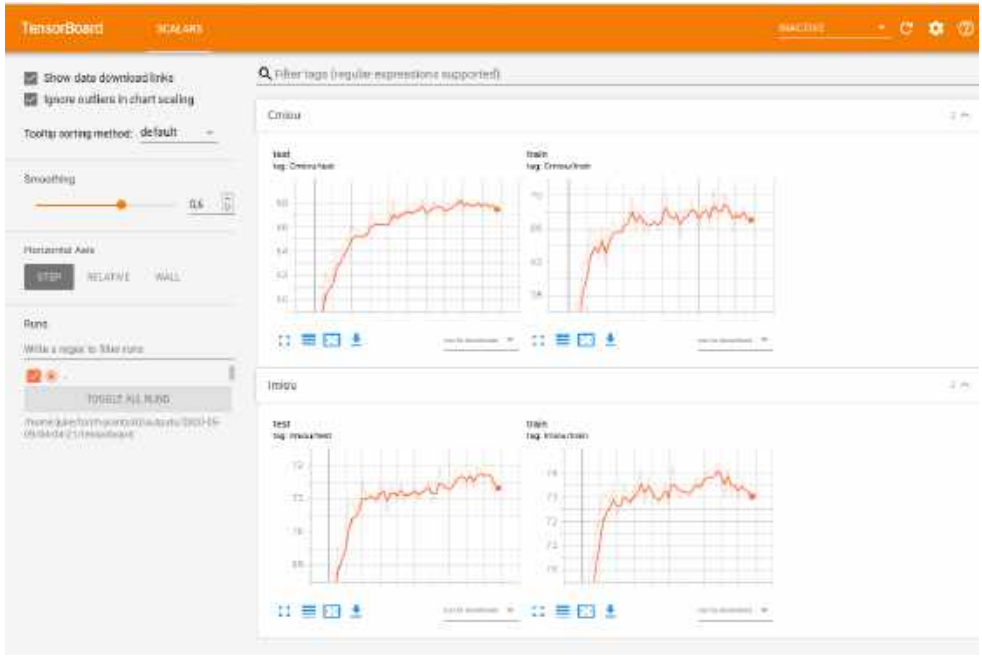


FIGURE 21. Graphiques de l'éxecution de 100 epoch sur ShapeNet utilisant PointNet

2.2.2.2 PointNet++ sur Shapenet :

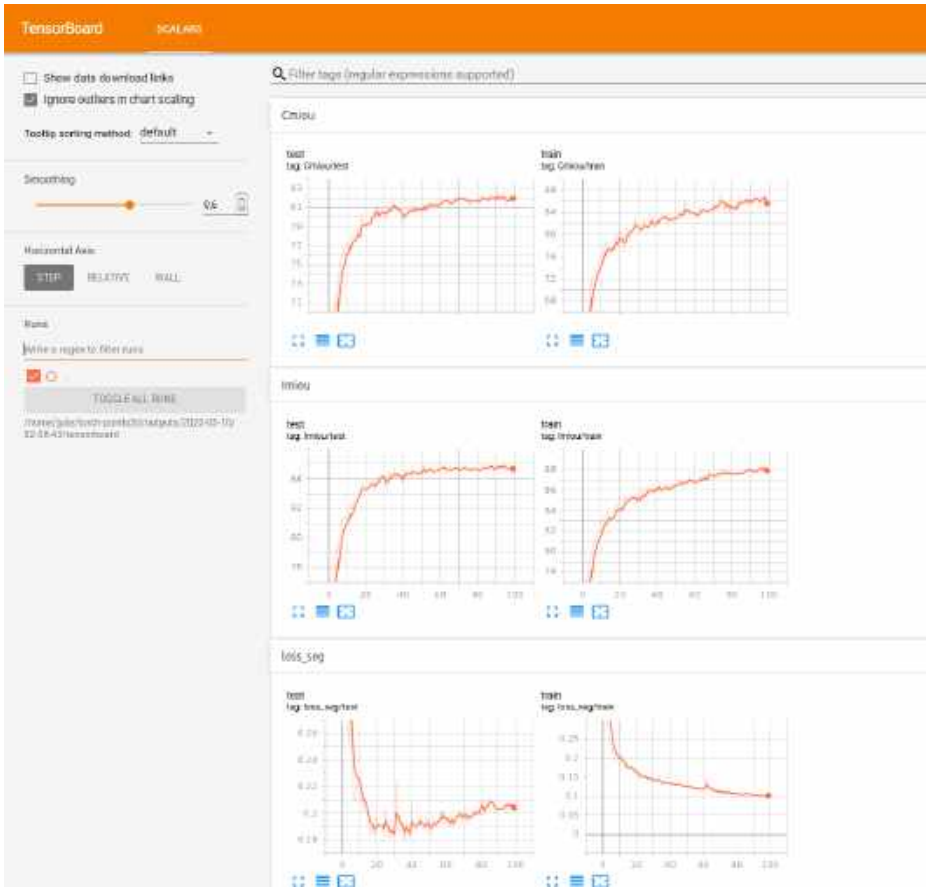


Figure 22. Graphiques de l'éxecution de 100 epoch sur ShapeNet utilisant PointNet++

Il est possible de soulever plusieurs points :

— PointNet++ est bien plus long à s'éxecuter que PointNet (ce qui est normal)
— Par contre, PointNet++ est bien plus efficace

2.3 Reproduire l'expérimentation

2.3.1 Pré-requis (dans notre cas) :

— Ubuntu 18.04
— PyTorch, Pytorch Geometric
— Cuda
— Le dépôt local de torch-point3d

2.3.2 Optionnel :

— Poetry (permet de gérer les dépendances)
— Pyenv, python > 3.6 (permet de gérer les versions python), dans notre cas 3.6.10

2.3.3 Comment lancer un entraînement :

```
poetry run python train.py
task={tâche}
model_type={type du modèle}
model_name={nom du modèle} dataset={dataset choisi}
```

2.3.3.1 Exemple :

```
poetry run python train.py
task=segmentation
model_type=pointnet2
model_name=pointnet2_charlesssg dataset=shapenet
```

2.3.3.2 Résultat :

Figure 23. Aperçu de la sortie de train.py

Les lignes en bleues sont les nouvelles valeurs des métriques qui sont conservés (celles ci doivent être plus élevées que celles d'avant).

Si vous voulez trouver les noms et les types de modèles qui sont disponibles à l'heure actuelle :

`/{ROOT du projet}/conf/models/{task}/`

Vous aurez accès au fichier de configuration de chaque type de modèle, par exemple pointnet2 :

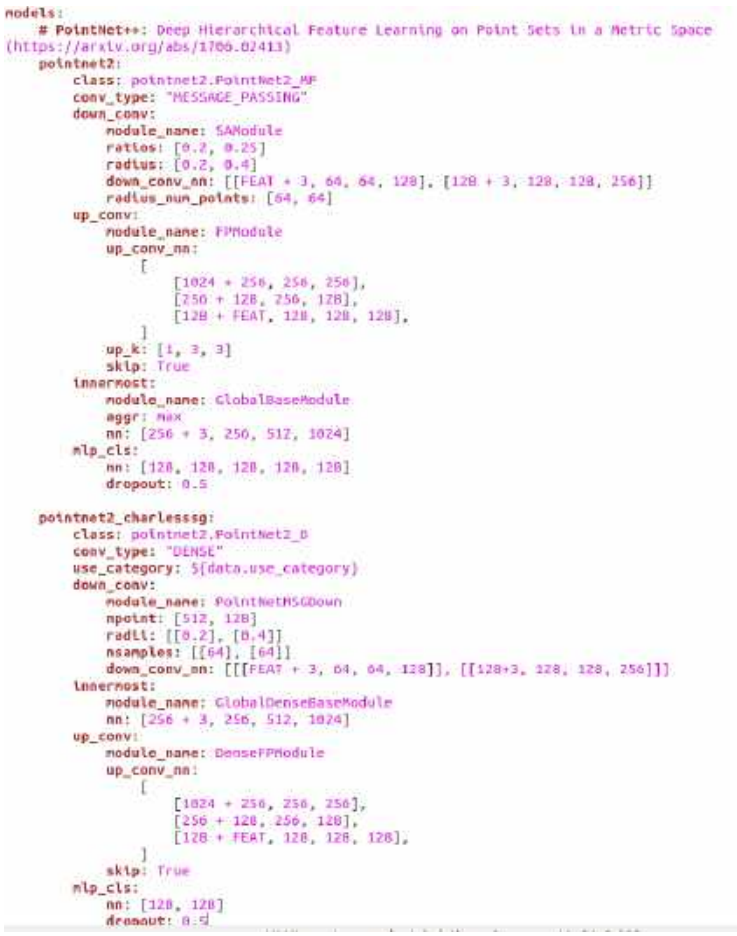


FIGURE 24. Fichier de configuration du modèle Shapenet

Le nom du modèle est l'un des modèles proposé dans le document, quant au titre c'est la première partie du nom du document .yaml.

2.3.4 Comment évaluer un modèle :

```
poetry run python eval.py
model_name={nom du modèle (default: KSConv)}
```

2.3.4.1 Exemple :

```
poetry run python eval.py
model_name=pointnet2_charlesssg
```

2.3.4.2 Résultat :

FIGURE 25. Sortie de eval.py

2.3.4.3 Accèder aux visualisations graphiques :

```
tensorboard --logdir={votre_chemin}/torch-points3d/outputs/
{experimentation au choix}/tensorboard
```

2.3.4.4 Résultat :

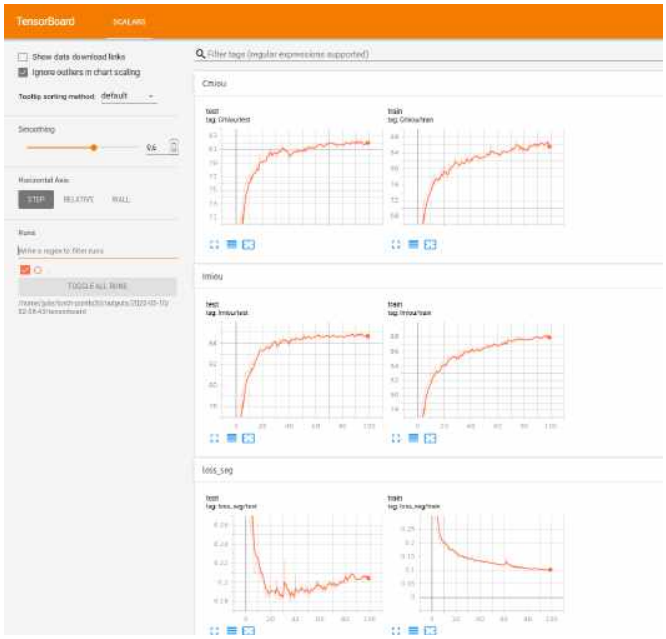


FIGURE 26. Graphiques de l'évolution des métriques sur 100 epoch, Shapenet avec PointNet++

Ces représentations permettent, par exemple, de lire les valeurs des métriques à un moment donné.

2.4 Aller plus loin

2.4.1 Visualisation

Comme vous l'avez vu, cet outil permet de faire marcher quelques modèles de Deep Learning sur des nuages de points. En plus de l'affichage sur terminal ou bien des représentations en graphique présenté plus haut, il est aussi possible d'avoir un affichage de la segmentation des objets. Pour cela, il suffit d'effectuer un entraînement et ensuite vous pourrez utiliser les notebooks mis à disposition dans le répertoire /dashboard. En théorie, nous devrions obtenir ceci avec le dashboard.ipynb :

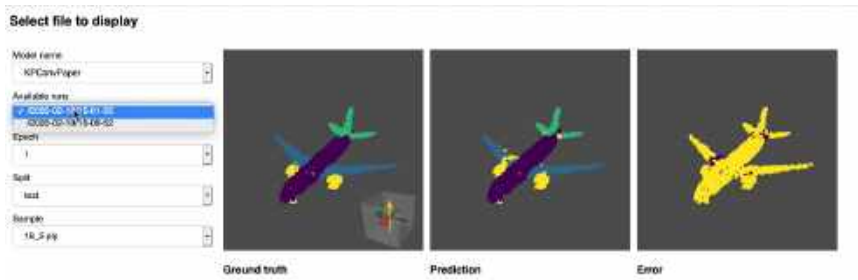


FIGURE 27. Affichage graphique des résultats

Cet outil peut être très pratique pour visualiser vos résultats, cependant dans notre cas nous n'avons pas réussi à faire marcher le notebook à cause d'une erreur dont nous n'avons pas réussi à trouver la cause.

2.4.2 Autres modèles

"Tutoriel pour créer un nouveau modèle", https://torch-points3d.readthedocs.io/en/latest/src/tutorials.html#create-a-new-model

Nous avons testé PointNet et PointNet++, il est cependant possible d'utiliser d'autres modèles à l'image de :

— KPConv
— RSConv

De plus, d'autres modèles seront ajoutés au framework dans le futur.

Bien sur, il est aussi possible de créer son propre modèle. Dans ce cas, il faudra recrée les modules et recrée un fichier de configuration <model>.yaml

2.4.3 Autres dataset

"Tutoriel pour créer un nouveau dataset", https://torch-points3d.readthedocs.io/en/latest/src/tutorials.html#create-a-new-dataset

Nous n'avons utilisé que ShapeNet, il est néanmoins possible comme dit plus haut, d'utiliser d'autres dataset que ShapeNet.

Des datasets tels que :

— S3DIS
— Scannet
— ShapeNet

Pour chacuns des trois datasets, ils existent deux versions ; une version brute et une version encapsulé dans une classe. Cette classe fait les tests, l'entraînement et la validation pour l'utilisateur.

Cependant, il est aussi possible de crée ses propres dataset. Il faut pour cela créer le fichier python associé au dataset (<dataset>.py) ainsi que le fichier de configuration <dataset>.yaml.

Conclusion

Pour conclure ce travail de recherche, le traitement de données 3D par le *Deep Learning* est un domaine en pleine expansion, où il reste encore beaucoup à découvrir. De nouvelles approches sont tentées d'années en années et réussissent à être plus performantes et rapides que les précédentes.

Il existe plusieurs façons de représenter des données 3D, d'un côté les représentations euclidiennes et d'un autre, les représentations non-euclidiennes. Les représentations euclidiennes sont le sujet de beaucoup de travaux de recherches et bénéficient déjà de solutions assez performantes. Les représentations non-euclidiennes sont plus complexes à appréhender et à utiliser, mais les progrès sont rapides.

Chaque type de représentation possède des spécificités à prendre en compte au moment de concevoir un modèle d'apprentissage. Chaque représentation a ses avantages et ses inconvénients. La grille de *voxels* par exemple, bien que très simple, nécessite de grandes quatité de mémoire ainsi que des calculs coûteux pour gagner en précision. D'un autre côté, les nuages de points et les maillages, sont des représentations 3D fidèles de l'objet original mais sont plus complquées à traiter car celles-ci sont désordonnées.

En ce qui concerne les différentes approches, celles qui ont été abordées dans cette recherche se basent sur des nuages de points. Les nuages de points sont des données faciles trouver, grâce à la démocratisation des LIDAR.

Parmi les deux méthodes analysées, PointNet++ est l'approche la plus efficace mais elle demande un plus long temps de traitement comparativement à PointNet, son cadet.

Références

Ahmed, E., A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada et B. E. Ottersten. "Deep Learning Advances on Different 3D Data Representations: A Survey". *ArXiv* abs/1808.01462 (2018).

Boscaini, D., J. Masci, E. Rodolà et M. Bronstein. "Learning shape correspondence with anisotropic convolutional neural networks". Sous la direction de D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon et R. Garnett, 2016, 3189-3197. http://papers.nips.cc/paper/6045-learning-shape-correspondence-with-anisotropic-convolutional-neural-networks.pdf.

"Documentation torch points3d". https://torch-points3d.readthedocs.io/en/latest/.

Masci, J., D. Boscaini, M. M. Bronstein et P. Vandergheynst. "Geodesic Convolutional Neural Networks on Riemannian Manifolds.", 2015, 832-840. http://dblp.uni-trier.de/db/conf/iccvw/iccvw2015.html#MasciBBV15.

Qi, C. R., H. Su, K. Mo et L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". Cite arxiv:1612.00593, 2016. http://arxiv.org/abs/1612.00593.

Qi, C. R., L. Yi, H. Su et L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". *arXiv preprint arXiv:1706.02413*, 2017.

"Tutoriel pour créer un nouveau dataset". https://torch-points3d.readthedocs.io/en/latest/src/tutorials.html#create-a-new-dataset.

"Tutoriel pour créer un nouveau modèle". https://torch-points3d.readthedocs.io/en/latest/src/tutorials.html#create-a-new-model.

Yavartanoo, M., E. Kim et K. M. Lee. "SPNet: Deep 3D Object Classification and Retrieval Using Stereographic Projection." Sous la direction de C. V. Jawahar, H. Li, G. Mori et K. Schindler, Lecture Notes in Computer Science, 11365 (2018) : 691-706. http://dblp.uni-trier.de/db/conf/accv/accv2018-5.html#YavartanooKL18.

Zhi, S., Y. Liu, X. Li et Y. Guo. "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning". *Comput. Graph.* 71 (2018) : 199-207.