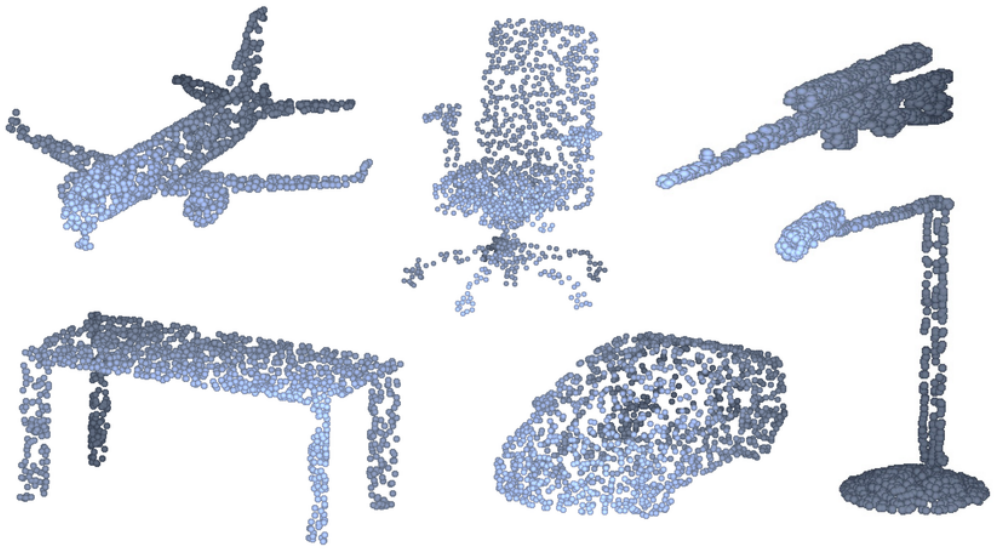
EUPI : École Universitaire de Physique et d'Ingénierie
4, avenue Blaise Pascal - CS 60026
63178 Aubière Cedex

Master PAR - Apprentissage

Apprentissage Profond 3D
Réseau de Neurones pour la Classification de Nuages de Points 3D



Élèves :

Lucas Pichon
Loubna Taleb

Professeur :

Paul Checchin

Rendu le 7 Février 2025

Table des matières

Résumé

Dans ce projet, nous avons exploré l'implémentation et l'évaluation du réseau de neurones **PointNet** pour la **classification de nuages de points 3D**. Introduit par Qi et al. en 2017, PointNet est une architecture novatrice qui traite directement les nuages de points sans nécessiter leur conversion en grilles de **voxels** ou en images, évitant ainsi une augmentation exponentielle des besoins en **mémoire**. L'**objectif principal** était de concevoir des **modèles** performants capables de classifier des objets 3D à l'aide des ensembles de données **ModelNet10** et **ModelNet40**.

Notre travail a consisté à implémenter une version de base de PointNet, suivie de l'intégration d'un module **T-Net** afin d'améliorer l'**alignement des points** dans l'espace tridimensionnel. Nous avons également étudié l'impact de diverses stratégies d'**augmentation des données** sur la **précision** du modèle.

En combinant une utilisation complète des **matrices T-Net** et des techniques d'augmentation des données adaptées, nous avons atteint une précision de **87 %** sur l'ensemble de test, démontrant ainsi l'**efficacité** et la robustesse de cette approche pour des tâches de classification 3D complexes.

Mots-clés : PointNet, classification 3D, nuages de points, T-Net, augmentation des données

1 Introduction

Contexte scientifique et problématique

Avec l'essor des technologies de numérisation 3D et leur adoption dans des domaines tels que la vision par ordinateur et la robotique, le traitement des données géométriques 3D est devenu un enjeu majeur. Avant 2016, les scientifiques utilisaient des voxels pour représenter des objets ou des environnements en grilles régulières, ou projetaient les données 3D en 2D. Les voxels, bien qu'offrant une représentation complète de l'espace, souffrent d'une résolution limitée et d'un coût computationnel élevé dû à la taille des grilles [1]. Les projections 2D, quant à elles, simplifient les données en les rendant traitables comme des images classiques, mais elles perdent des informations géométriques cruciales, telles que la profondeur et la forme 3D, ce qui entraîne une perte de détails et une explosion de la complexité en cas de vues multiples [2].

Toutes ces limitations des méthodes traditionnelles ont poussé les chercheurs à développer une approche permettant de traiter directement les points 3D sans les transformer en voxels ni en projections 2D, tout en les utilisant avec des réseaux de neurones convolutifs (CNN).

En 2017,Qi et al. [3] ont été les pionniers de cette méthode avec PointNet, un réseau de neurones innovant conçu pour traiter des nuages de points 3D. Contrairement aux méthodes classiques, PointNet prend directement en entrée des ensembles de points et utilise des opérations invariantes aux permutations pour capturer efficacement les relations géométriques et structurelles, sans perte d'information. Cette approche a ainsi révolutionné l'analyse des formes 3D, offrant une solution plus rapide et plus précise pour des tâches telles que la classification, la segmentation et la reconnaissance d'objets.

Le défi consiste donc à concevoir et à implémenter un réseau de neurones capable de traiter directement les points 3D en exploitant leurs propriétés géométriques, tout en respectant leur invariance aux permutations, et de réaliser des tâches de classification et de segmentation de manière efficace.

Objectif du projet

En s'appuyant sur les travaux de Qi et al., l'objectif de ce projet est d'explorer l'architecture PointNet, d'en comprendre les mécanismes fondamentaux et d'améliorer ses performances par l'intégration de modules T-Net ainsi que l'utilisation de techniques d'augmentation de données 3D. Nous visons à valider cette approche sur des ensembles de données de référence, tels que ModelNet10 et ModelNet40, et à évaluer son efficacité en comparaison avec les méthodes existantes.

Plan du rapport

Dans la première partie nous avons exploré des réseaux de neurones sur des données géométriques 2D en expérimentant un nuage de points spiral, avec des variations de la fonction d'activation, du taux d'apprentissage et du nombre de couches cachées, afin d'analyser l'impact de chaque paramètre sur les performances du modèle.

Ensuite, nous avons implémenté un perceptron multicouche classique, PointMLP, pour la classification de nuages de points 3D, en utilisant une architecture comprenant trois couches cachées.

Dans une troisième partie, nous avons examiné PointNet, en commençant par une analyse de sa version basique. Nous détaillerons l'architecture du modèle et introduirons le module T-Net, conçu pour améliorer la robustesse face aux transformations géométriques. Nous expliquerons son intégration dans PointNet et évaluerons l'impact de cette amélioration sur les performances du modèle.

Nous consacrerons également une section à l'augmentation de données pour les nuages de points 3D, en présentant des techniques classiques ainsi qu'une méthode d'augmentation innovante que nous proposons.

Enfin, la dernière partie sera dédiée aux tests et résultats, où nous comparerons les performances des différents modèles sur les jeux de données ModelNet40-Ply et ModelNet10, en mettant en évidence les forces et les faiblesses de chaque approche. Une discussion finale permettra de conclure sur les observations majeures et d'envisager des perspectives d'amélioration futures.

2 Analyse de réseaux de neurones sur données 2D et nuages de points 3D

2.1 Exploration du perceptron multicouche (MLP) pour les données géométriques 2D

Le perceptron multicouche, ou *Multi-Layer Perceptron* (MLP), est un modèle clé des réseaux de neurones artificiels. Il se compose de plusieurs couches de neurones connectées : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie. Chaque neurone prend des données en entrée et applique une transformation pour produire une sortie, en utilisant une fonction d'activation non linéaire.

Un neurone reçoit des variables d'entrée réelles $x_1, x_2, \ldots, x_n$, chacune associée à un poids w_i. Le calcul du neurone s'exprime par :

$$\hat{y} = f\left(\sum_{i=1}^{n} w_i x_i + b\right),$$

où b est un biais, et f est une fonction d'activation introduisant la non-linéarité. La valeur $\hat{y}$ est l'activation du neurone, représentant sa sortie.

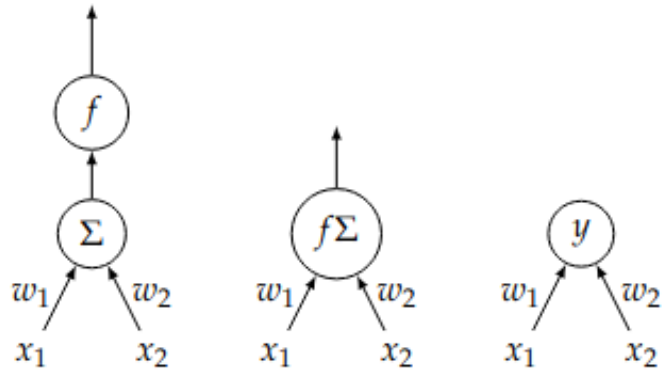


FIGURE 2.1 – Illustration du MLP : à gauche, le calcul détaillé d'un neurone ; au centre, la représentation condensée combinant somme pondérée et fonction d'activation ; à droite, un réseau complet avec propagation des activations.

Lorsqu'il est appliqué à d**es données géométriques en 2D**, comme des coordonnées spatiales (x, y), le MLP traite directement ces points comme des vecteurs d'entrée. L'objectif est d'apprendre des relations complexes, telles que des frontières de décision non linéaires, nécessaires pour classer ou segmenter des données ayant des structures géométriques complexes.

Le **fonctionnement du MLP** repose sur des transformations linéaires suivies de non-linéarités, appliquées dans chaque couche cachée. Ces transformations sont définies par :

$$h_i = f(W_i \cdot h_{i-1} + b_i),$$

où $W_i \in \mathbb{R}^{d_i \times d_{i-1}}$ est la matrice de poids, $b_i \in \mathbb{R}^{d_i}$ est le vecteur de biais, h_{i-1} est la sortie de la couche précédente (ou l'entrée initiale $h_0 = [x, y]$), et f est une fonction d'activation non linéaire (comme ReLU, Sigmoid ou Tanh). Ces couches permettent de projeter les données dans un espace latent de dimension supérieure, modélisant ainsi des relations complexes, comme des frontières non linéaires.

La couche de sortie produit une prédiction adaptée à la tâche. Pour un problème de **classification**, elle génère une probabilité d'appartenance à une classe donnée en utilisant une fonction softmax :

$$y_{\text{pred}}^{(c)} = \frac{\exp(W_L^{(c)} \cdot h_{L-1} + b_L^{(c)})}{\sum_k \exp(W_L^{(k)} \cdot h_{L-1} + b_L^{(k)})},$$

où c est l'indice de la classe, et $W_L^{(c)}$ et $b_L^{(c)}$ sont les poids et biais associés à la classe c. Pour une tâche de **régression**, la sortie est une valeur continue calculée par :

$$y_{\text{pred}} = W_L \cdot h_{L-1} + b_L,$$

où $W_L \in \mathbb{R}^{1 \times d_{L-1}}$ et $b_L \in \mathbb{R}$.

Enfin, le réseau ajuste les poids W_i et les biais b_i à l'aide d'**une méthode d'optimisation**, telle que la descente de gradient. Ce processus vise à minimiser une fonction de perte $L(y_{\text{pred}}, y_{\text{true}})$, qui mesure l'écart entre les prédictions et les cibles. La fonction de perte moyenne est donnée par :

$$L = \frac{1}{N} \sum_{j=1}^{N} \ell(y_{\text{pred}}^{(j)}, y_{\text{true}}^{(j)}),$$

où N est le nombre d'exemples, et ℓ est une perte individuelle (par exemple, l'entropie croisée pour la classification ou l'erreur quadratique moyenne pour la régression).

Les poids et les biais sont mis à jour à chaque étape en utilisant les gradients calculés par :

$$W_i \leftarrow W_i - \eta \frac{\partial L}{\partial W_i}, \quad b_i \leftarrow b_i - \eta \frac{\partial L}{\partial b_i},$$

où η est le taux d'apprentissage [4].

2.2 Analyse du modèle PointMLP pour la classification de nuages de points 3D

Nous nous intéressons maintenant à de la classification de nuages de points 3D. Dans un premier temps, nous allons nous concentrer sur un réseau de neurones multicouches classique (MLP).

2.2.1 Architecture du PointMLP

L'architecture de ce réseau repose sur un MLP à trois couches, dont la première couche prend un vecteur d'entrée de taille 3072 (pour un nuage de 1024 points 3D, chaque point ayant 3 coordonnées), suivi de deux couches cachées avec des tailles respectives de 512 et 256, et une dernière couche de sortie de taille N, où N correspond au nombre de classes du dataset (dans notre cas ce sera 40). Voici comment est organisée l'architecture du PointMLP :

1. **Entrée du réseau :** Le nuage de points est "aplati" (flattened) afin d'obtenir un vecteur de 3072 éléments (1024 points $\times$ 3 dimensions), ce qui constitue l'entrée du réseau.
2. **Première couche :** La première couche est un MLP avec 3072 entrées et 512 neurones.
3. **Deuxième couche :** La deuxième couche est également un MLP, mais avec 512 entrées et 256 neurones. Un taux de "dropout" de 0.3 est appliqué sur cette couche pour éviter le surapprentissage.
4. **Dernière couche :** La dernière couche de l'architecture est un MLP avec 256 entrées et N neurones, où N correspond au nombre de classes dans le dataset ModelNet.
5. **Activation :** Chaque couche est suivie d'une normalisation par lot (`BatchNorm1d`) et utilise la fonction d'activation (`ReLU`).
6. **Sortie :** La sortie du réseau est obtenue grâce à la fonction `LogSoftmax` qui permet de calculer les scores des différentes classes.

2.3 Exploration du modèle PointNet pour la classification de nuages de points 3D

Le modèle PointNet représente une avancée majeure dans le traitement des nuages de points. Dans cette section, nous allons analyser deux variantes du modèle PointNet : la première, qui repose sur l'architecture de base sans réseau T-Net, et la seconde, qui intègre le réseau T-Net pour améliorer la robustesse et la précision du modèle. Cette comparaison nous permettra d'explorer l'impact de l'ajout du T-Net sur la performance et la flexibilité du modèle.

2.3.1 Analyse de la version basique de PointNet

La version de base du modèle **PointNet** n'inclut pas le réseau **T-Net**. Voici son architecture :

1. **Prise en charge des points 3D :** Le modèle commence par prendre en entrée un nuage de points tridimensionnels. Chaque point est représenté par ses coordonnées spatiales (x,y,z).
2. **MLP partagé à 2 couches :** Le modèle applique un réseau MLP partagé pour chaque point afin de calculer des caractéristiques.
   — Première couche : MLP$(3, 64)$
   — Deuxième couche : MLP$(64, 64)$
3. **MLP partagé à 3 couches :** Après avoir extrait les premières caractéristiques, le modèle applique un deuxième MLP partagé à 3 couches pour augmenter la complexité des représentations locales. La dimension des caractéristiques est augmentée de 64 à 1024.
   — Première couche : MLP$(64, 64)$
   — Deuxième couche : MLP$(64, 128)$
   — Troisième couche : MLP$(128, 1024)$
4. **Max Pooling :** Cette fonction est ensuite utilisée pour calculer une caractéristique globale de dimension 1024.
5. **MLP global à 3 couches :** Ensuite, un MLP à trois couches est appliqué à la caractéristique globale pour effectuer la tâche de classification. Ce MLP prend en entrée le vecteur de 1024 dimensions et produit une sortie de taille égale au nombre de classes, ce qui permet de prédire à quelle classe appartient le nuage de points d'entrée.
   — Première couche : MLP$(1024, 512)$
   — Deuxième couche : MLP$(512, 256)$
   — Troisième couche : MLP$(256,$ nombre de classes$)$
6. **Activation :** À chaque étape du réseau, sauf la dernière, une activation `ReLU` est utilisée. La couche finale de classification utilise la fonction d'activation *LogSoftmax* pour produire des scores de probabilité pour chaque classe, ce qui permet une classification multiclasse.

2.3.2 Analyse de l'amélioration avec T-Net

L'architecture de POINTNET a démontré des performances solides pour la classification de nuages de point 3D. Cependant, une limitation importante de POINTNET est qu'il ne prend pas en compte l'alignement spatial des nuages de points avant de les traiter. Cela peut entraîner des erreurs lorsque les nuages de points sont mal orientés. Pour améliorer la précision, on peut utiliser un module d'alignement des nuages de points avant de les entrer dans le réseau POINTNET. Cette tâche d'alignement est réalisée par un sous-réseau appelé T-NET.

2.3.2.1 Le réseau T-Net

Le réseau T-Net est une version simplifiée de PointNet qui est spécifiquement conçu pour effectuer l'alignement des nuages de points 3D. T-Net fonctionne en régressant une matrice de transformation 3×3, ce qui permet de faire correspondre correctement les points dans l'espace 3D. Cette matrice est ensuite utilisée pour ajuster les coordonnées des points avant qu'ils ne soient envoyés dans le réseau PointNet. T-Net est composé de 3 étapes principales :

1. **MLP partagé à 3 couches :** Ce MLP permet d'extraire des caractéristiques des points 3D en entrée et de prédire la matrice de transformation qui aligne correctement le nuage de points pour qu'il soit mieux compris par le réseau PointNet.
   — Première couche : MLP$(3, 64)$
   — Deuxième couche : MLP$(64, 128)$
   — Troisième couche : MLP$(128, 1024)$
2. **Max pooling à travers les points :** Cette opération permet de résumer l'information globale du nuage de points en une seule représentation de taille fixe, de dimension 1024 dans notre cas.
3. **MLP global à 3 couches :** C'est ce MLP qui permet de produire la matrice de transformation à partir des transformations du MLP partagé et le pooling.
   — Première couche : MLP$(1024, 512)$
   — Deuxième couche : MLP$(512, 256)$
   — Troisième couche : MLP$(256,$ kxk$)$, avec k la dimension de la matrice.

2.3.2.2 Ajout du réseau T-Net dans le modèle PointNet

Maintenant que le réseau T-Net a été introduit, nous pouvons l'utiliser dans le modèle PointNet. Originalement, 2 réseaux T-Net sont utilisés dans le modèle PointNet. Le premier, situé en début de réseau, permet d'aligner les nuages de points 3D. Le deuxième est situé juste après le premier MLP partagé et permet d'aligner les points dans l'espace des caractéristiques de dimensions 64. Voici l'architecture finale de PointNet :

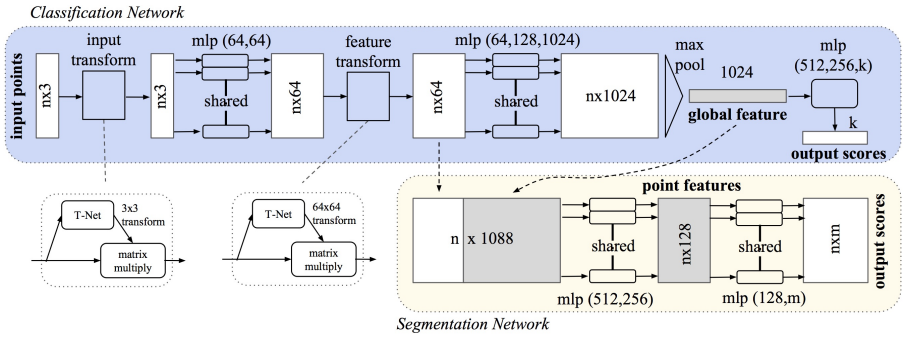


FIGURE 2.2 – Architecture de PointNet (tirée de l'article originale).

Dans notre version complète de PointNet, nous n'utilisons que le premier réseau T-Net. Cela facilite l'implémentation du modèle, car il s'agit simplement d'un réseau T-Net (de taille 3) suivi du modèle PointNet sans T-Net, celui que nous avons évoqué précédemment.

2.4 Augmentation de données pour les nuages de points 3D

L'augmentation de données est une technique qui consiste à générer artificiellement de nouvelles données à partir de données existantes pour entraîner des modèles. Ceci permet d'augmenter le volume du jeu de données en apportant de légères modifications aux données d'origine. Plusieurs méthodes existent et voici celles que nous allons utiliser lors de nos expérimentations :

1. **Rotation aléatoire selon l'axe Z :** Permet d'effectuer des rotations aux objets selon l'axe Z et ainsi permettre au modèle d'apprendre à reconnaître les objets indépendamment de leur orientation dans l'espace.
2. **Mélange des points :** Consiste à mélanger aléatoirement l'ordre des points dans le nuage de point tout en conservant leurs coordonnées. L'intérêt derrière est de forcer le modèle à se concentrer davantage sur la structure globale de l'objet plutôt que sur l'ordre des points.
3. **Bruit gaussien :** Consiste à ajouter des perturbations aléatoires aux coordonnées des points en suivant une distribution gaussienne. L'objectif de cette augmentation de données est de rendre le modèle plus robuste face à des données réelles qui comportent toujours des imprécisions dues à la nature des capteurs par exemple.

Il existe de nombreuses autres techniques d'augmentation de données. Parmi elles, nous avons choisi d'implémenter une quatrième méthode : **la modification de l'échelle**. Cette approche consiste à appliquer un facteur aléatoire au nuage de points, ce qui permet de simuler des variations de taille des objets. En intégrant cette technique, le modèle gagne en robustesse face aux variations dimensionnelles, améliorant ainsi sa capacité à généraliser.

3 Tests et résultats

3.1 Exploration du fonctionnement de MLP sur les données de 2D

Nous avons utilisé le site TensorFlow Playground pour explorer le fonctionnement du perceptron multicouche (MLP) sur les données géométriques en 2D. Cet outil interactif en ligne permet de concevoir et tester des réseaux de neurones sur différents types de données. Il offre une interface simple pour ajuster les paramètres des réseaux de neurones et observer leur impact sur les performances. Nous avons exploré les fonctionnalités suivantes

1. **Choix des données :** Les données d'entrée peuvent être représentées sous plusieurs formes géométriques comme des cercles, des spirales, des distributions XOR ou encore gaussiennes. Ces jeux de données permettent de comprendre comment un réseau peut ou non apprendre des motifs non-linéaires.
2. **Configuration du réseau :** Nous avons testé différentes architectures de réseau en modifiant le nombre de couches, le nombre de neurones par couche, les fonctions d'activation et les paramètres comme le taux d'apprentissage.
3. **Visualisation des résultats :** L'outil affiche en temps réel la frontière de décision apprise par le réseau, permettant d'évaluer sa capacité à généraliser ou à s'adapter aux données d'entraînement.

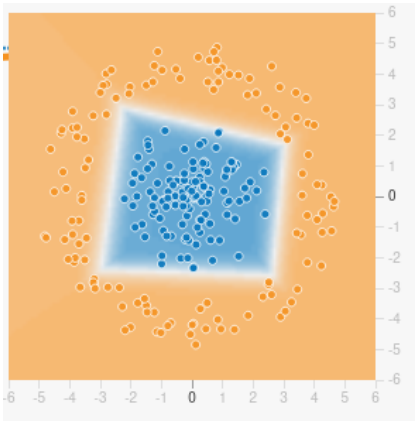


FIGURE 3.1 – Données circulaires

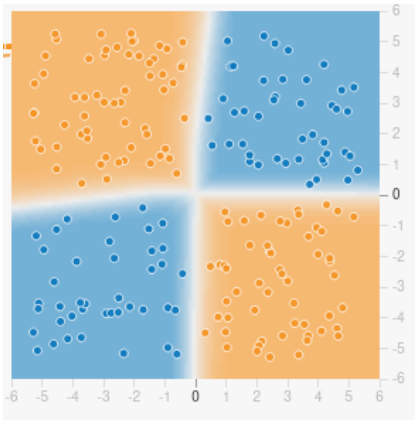


FIGURE 3.2 – Données XOR

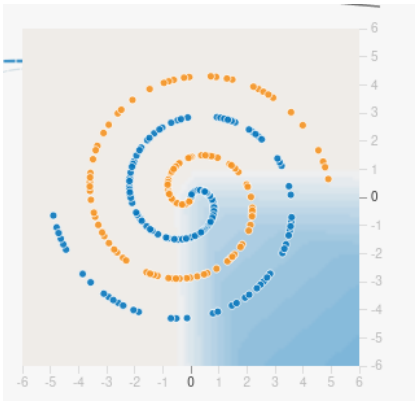


FIGURE 3.3 – Données spirales

FIGURE 3.4 – Exemples de classification de données géométriques avec un perceptron multicouche (MLP) utilisant une fonction d'activation ReLU et un taux d'apprentissage de 0.001

Les résultats montrent que le réseau arrive à réaliser une bonne classification pour des jeux de données prédéfinis tels que le cercle ou l'XOR. Toutefois, pour un jeu de données prédéfini comme la spirale, le réseau a eu des difficultés à apprendre une frontière de décision précise. Cela peut s'expliquer par le fait qu'une spirale est un motif complexe et non-linéaire, ce qui dépasse la capacité d'un réseau avec seulement deux couches et un nombre relativement faible de neurones. Cette exploration met en évidence les limites des architectures simples face à des problèmes complexes.

3.2 Implémentation de PointMLP

3.2.1 Résultats de classification sur le dataset ModelNet40_PLY

Le modèle **PointMLP** a été entraîné sur le dataset *ModelNet40*, un ensemble de données contenant des nuages de points 3D répartis en 40 classes différentes. Les résultats obtenus en termes de perte (*loss*) et de précision (*accuracy*) sont présentés et analysés ci-dessous.

La perte d'entraînement et de validation diminue de manière régulière au fil des époques, atteignant une convergence stable après environ 60 époques. Ce comportement suggère que le modèle apprend efficacement les caractéristiques des données sans signe apparent de surapprentissage. L'intégration de mécanismes tels que la *Batch Normalization* et le *Dropout* dans l'architecture du réseau a contribué à stabiliser l'apprentissage et à prévenir une trop forte adaptation aux données d'entraînement.

En ce qui concerne la précision, elle augmente rapidement lors des 20 premières époques, avant de se stabiliser autour de 22.5% sur l'ensemble de validation après environ 50 époques. Ce résultat montre que, bien que le modèle ait appris certaines relations entre les nuages de points et leurs classes respectives, il reste une marge d'amélioration significative pour atteindre une meilleure classification des 40 classes de *ModelNet40*.

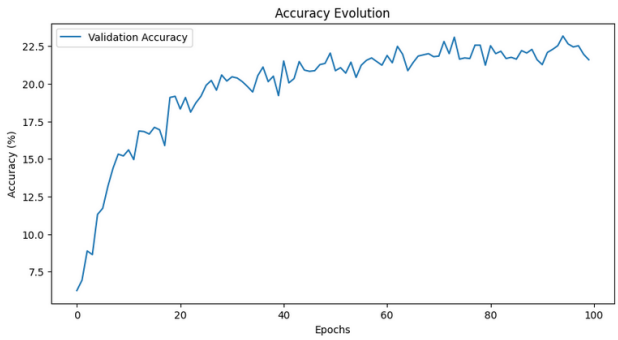


FIGURE 3.5 – Évolution de la précision (*accuracy*) d'entraînement et de validation pour PointMLP.

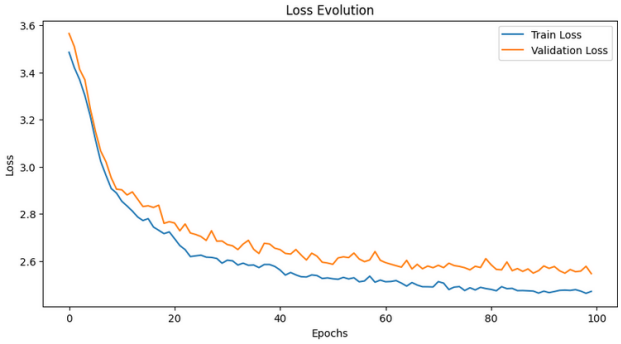


FIGURE 3.6 – Évolution de la perte (*loss*) d'entraînement et de validation pour PointMLP.

FIGURE 3.7 – Performance de PointMLP en termes de perte et de précision sur le dataset ModelNet40_PLY.

3.2.2 Analyse et discussion des résultats obtenus

Le principal avantage du modèle PointMLP réside dans la simplicité de son architecture, qui repose sur un perceptron multicouche classique. Cependant, cette simplicité entraîne des limitations notables dans notre cas d'utilisation. En effet, l'aplatissement des données d'entrée, qui consiste à traiter les points comme un vecteur, entraîne une perte d'information géométrique essentielle liée à l'objet. La géométrie des points, c'est-à-dire leur position relative dans l'espace 3D, est une information cruciale pour une classification correcte des nuages de points. Sans cette information spatiale, le modèle est incapable de capturer les structures géométriques complexes qui définissent l'objet, ce qui nuit à la qualité de la classification, notamment pour des formes non régulières ou des objets ayant une symétrie complexe. Ainsi, PointMLP est limité par son incapacité à prendre en compte les dépendances spatiales entre les points du nuage, ce qui le rend moins efficace pour des tâches de classification de nuages de points 3D.

Pour conclure, PointMLP ne s'avère pas être le modèle le plus adapté ni le plus performant pour notre cas d'utilisation, à savoir la classification de nuages de points 3D. En raison de ses limitations, notamment l'absence de prise en compte des relations spatiales entre les points, il n'est pas idéal pour capturer les structures géométriques complexes des objets 3D. Afin d'améliorer les performances de classification, nous explorerons donc des modèles plus sophistiqués, tels que POINTNET, qui sont mieux adaptés à ce type de données.

3.3 Implémentation de PointNet

3.3.1 Résultats de classification et analyse des performances de la version basique

Dans cette section, nous présentons les performances du modèle PointNet sur les données du dataset *ModelNet40*. Après avoir implémenté l'architecture décrite dans le chapitre précédent, nous avons évalué la capacité du modèle à classifier les nuages de points 3D répartis sur 40 classes. Cette architecture est spécifiquement

conçue pour exploiter les caractéristiques géométriques des nuages de points tout en conservant leur invariance aux permutations des points, ce qui en fait un modèle adapté aux données tridimensionnelles.

Analyse des pertes (Loss)

La perte d'entraînement (*Train Loss*) et la perte de validation (*Validation Loss*) diminuent régulièrement au fil des époques, indiquant que le modèle apprend efficacement à minimiser l'erreur sur les données d'entraînement. Une convergence stable est atteinte après environ 60 époques, avec une perte d'entraînement inférieure à 0.2 et une perte de validation autour de 0.6. Cette réduction régulière des pertes montre que le modèle est bien optimisé et ne présente pas de signes de surapprentissage. Cela est également renforcé par l'utilisation de mécanismes comme la normalisation par lots (*BatchNorm*) et le *Dropout*, qui aident à améliorer la généralisation.

Analyse de la précision (Accuracy)

En ce qui concerne la précision, on observe que celle-ci augmente rapidement au début de l'entraînement, en particulier au cours des 20 premières époques. La précision sur l'ensemble de validation atteint un plateau après environ 60 époques, se stabilisant autour de 85%. Ce résultat indique que PointNet parvient à extraire des caractéristiques pertinentes pour la classification des nuages de points, bien que des améliorations soient encore possibles pour mieux différencier les 40 classes.

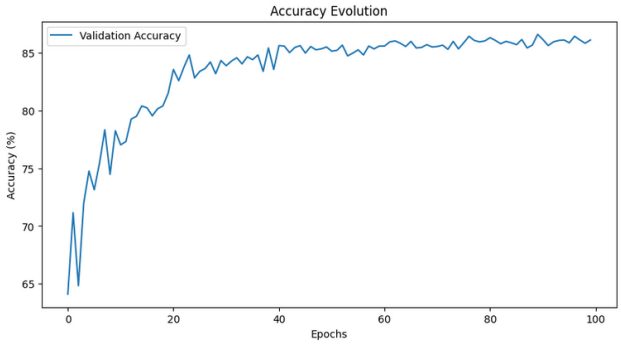


FIGURE 3.8 – Évolution de la précision (*accuracy*) sur l'ensemble de validation pour PointNet.

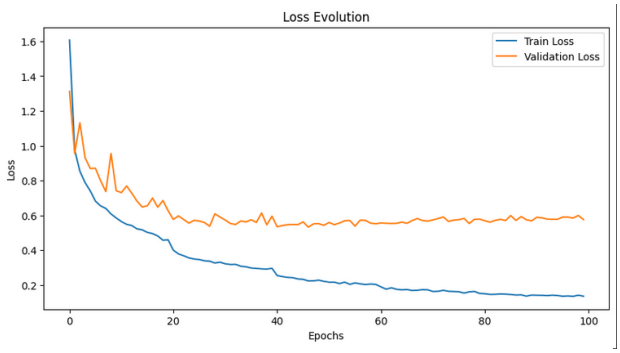


FIGURE 3.9 – Évolution de la perte (*loss*) d'entraînement et de validation pour PointNet.

FIGURE 3.10 – Performance de PointNet en termes de perte et de précision sur le dataset *ModelNet40_PLY*.

3.3.2 Résultats de classification avec T-Net

Comme mentionné précédemment, l'architecture PointNet nécessite l'intégration d'un module T-Net pour aligner les points. Cet alignement vise à rendre le modèle invariant aux transformations géométriques telles que les rotations et les translations, ce qui est essentiel pour traiter efficacement les nuages de points 3D.

Nous avons observé que la perte d'entraînement (*Train Loss*) diminue régulièrement, atteignant une valeur inférieure à **0.12** après environ 50 époques, ce qui indique une convergence rapide. La perte de validation (*Validation Loss*), quant à elle, reste stable autour de **0.5** après 50 époques, ce qui montre que le modèle généralise correctement sans surapprentissage.

La précision sur l'ensemble de validation (*Validation Accuracy*) augmente rapidement au cours des 420 premières époques, avant d'atteindre un plateau après environ 60 époques. Elle se stabilise autour de **87%**, ce qui représente une amélioration notable par rapport à la version basique de PointNet, qui atteignait une précision de **85%**. Cette amélioration est attribuée à l'ajout de T-Net, qui améliore la robustesse du modèle en gérant efficacement les transformations géométriques des nuages de points.

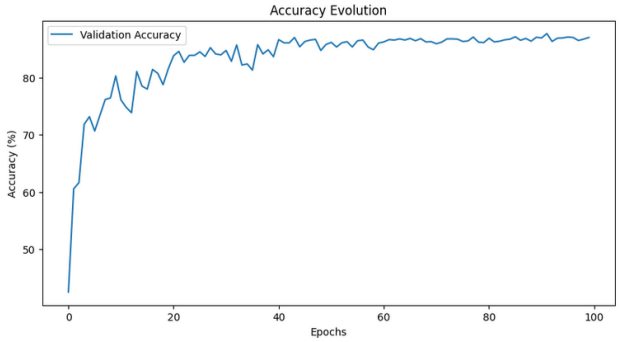


FIGURE 3.11 – Évolution de la précision (*accuracy*) sur l'ensemble de validation pour PointNet avec T-Net.

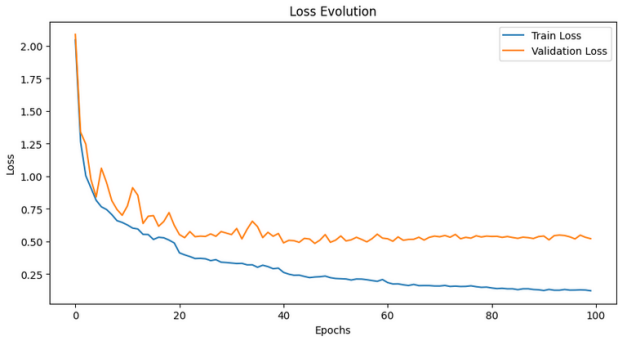


FIGURE 3.12 – Évolution de la perte (*loss*) d'entraînement et de validation pour PointNet avec T-Net.

FIGURE 3.13 – Performance de PointNet avec T-Net en termes de perte et de précision sur le dataset *ModelNet40_PLY*.

3.3.3 Comparaison des performances entre la version basique et la version avec T-Net

Avec la version basique de PointNet, le modèle atteint une précision de validation de **85%** et une perte stable autour de **0.6**, montrant une bonne capacité de classification. Cependant, cette version est limitée par l'absence de mécanisme d'invariance aux transformations géométriques, telles que les rotations et les translations. En revanche, l'ajout de T-Net permet d'aligner les points des nuages 3D, rendant le modèle invariant à ces transformations. Cette amélioration se traduit par une précision accrue de **87%** et une perte de validation réduite à **0.5**, tout en accélérant légèrement la convergence (stabilisation après 50 époques, contre 60 pour la version basique). Bien que T-Net introduise un surcoût computationnel et une complexité accrue, les gains en robustesse et en généralisation justifient pleinement son intégration dans l'architecture. Ainsi, PointNet avec T-Net se révèle plus performant et mieux adapté aux données 3D complexes.

Dans l'article original de PointNet [3] , les auteurs rapportent une précision globale de **89.2%** sur le dataset *ModelNet40* en utilisant leur architecture complète. Nos résultats légèrement inférieurs (**87%**) peuvent s'expliquer par l'absence de l'ajout d'un second réseau T-Net pour aligner les caractéristiques globales, comme cela est proposé dans l'architecture originale. L'intégration de ce second T-Net pourrait potentiellement améliorer nos performances pour se rapprocher des résultats de référence.

3.4 Augmentation de données pour les nuages de points 3D

Après avoir introduit une stratégie d'augmentation des données en appliquant des transformations aléatoires, comme une mise à l'échelle aléatoire (`RandomScaling`) au nuage de points, nous avons évalué les performances du modèle sur le dataset *ModelNet40*. L'objectif de cette augmentation était d'améliorer la robustesse du modèle face aux variations des données et d'éventuellement augmenter la précision. Cependant, les résultats obtenus montrent que la précision reste stable autour de 87%, sans amélioration significative.

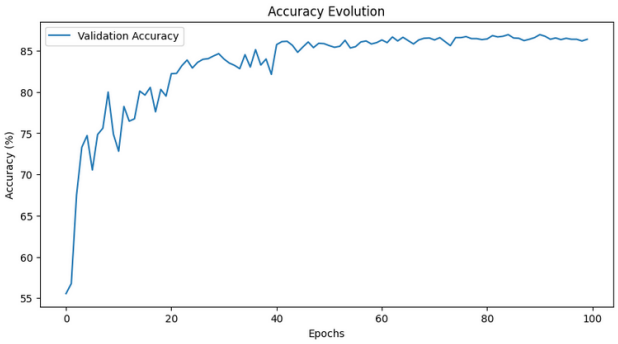


FIGURE 3.14 – Évolution de la précision (*accuracy*) sur l'ensemble de validation après augmentation des données.

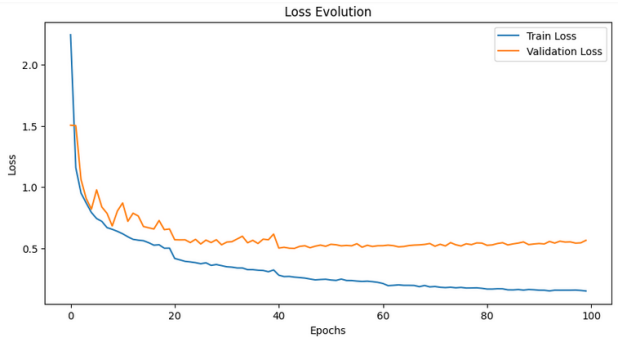


FIGURE 3.15 – Évolution de la perte (*loss*) d'entraînement et de validation après augmentation des données.

FIGURE 3.16 – Performances de PointNet avec augmentation des données sur le dataset *ModelNet40_PLY*.

4 Conclusion

Dans ce projet, nous avons exploré les capacités des réseaux de neurones pour la classification de nuages de points 3D, en nous concentrant principalement sur l'architecture PointNet et ses extensions. Les travaux ont permis d'analyser en profondeur les performances de différentes configurations, allant de la version basique de PointNet à son amélioration avec l'ajout de T-Net, ainsi que l'impact de l'augmentation des données sur la robustesse du modèle.

Nous avons commencé par analyser l'influence des hyperparamètres dans un MLP à couches cachées simples sur des données 2D telles que des motifs circulaires ou XOR. Nous avons constaté que, bien que ces données soient relativement simples à séparer, les données en spirale nécessitent un MLP plus sophistiqué pour identifier la frontière de décision et effectuer une classification correcte. Cela illustre l'importance d'une architecture adaptée à la complexité des données.

En ce qui concerne les nuages de points 3D, la version basique de PointNet a montré de bonnes performances, atteignant une précision de **85%** sur le dataset *ModelNet40*. Ces résultats confirment la capacité de ce modèle à extraire des caractéristiques pertinentes des nuages de points. Cependant, l'absence d'un mécanisme explicite pour gérer les transformations géométriques, telles que les rotations et les translations, limitait ses performances. L'intégration de T-Net a permis d'améliorer ces résultats, en rendant le modèle invariant à ces transformations, atteignant ainsi une précision de **87%**. Bien que ces performances soient légèrement inférieures aux **89.2%** rapportés dans l'article original de PointNet, elles valident l'efficacité de notre implémentation et mettent en lumière l'importance d'ajouter un second réseau T-Net pour l'alignement des caractéristiques globales, comme proposé dans l'architecture originale.

L'étude de l'augmentation des données a montré que les transformations simples appliquées (redimensionnement aléatoire, rotation, bruit gaussien) n'ont pas conduit à une amélioration significative de la précision. Ces résultats suggèrent que des stratégies d'augmentation plus avancées ou adaptées au domaine des nuages de points pourraient être nécessaires pour exploiter pleinement leur potentiel.

Pour améliorer les résultats obtenus, plusieurs pistes peuvent être envisagées. Tout d'abord, l'intégration d'un second réseau T-Net pour l'alignement des caractéristiques globales, comme proposé dans l'architecture complète de PointNet, pourrait renforcer les performances du modèle. Ensuite, des méthodes d'augmentation des données plus sophistiquées, telles que les déformations locales ou l'utilisation de techniques basées sur des réseaux génératifs adverses (GANs), pourraient contribuer à une meilleure robustesse face aux variations des nuages de points. Enfin, une recherche systématique des hyperparamètres pourrait permettre de maximiser les performances du modèle en ajustant finement les paramètres clés de l'entraînement.

Bibliographie

[1] R. Chellappa et al. "Volumetric Scene Representation for Robot Navigation". In : *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. This paper discusses the high memory and processing costs of voxels for complex 3D environments. 2000.

[2] H. Su et al. "Multi-View Convolutional Neural Networks for 3D Shape Recognition". In : *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. This paper discusses the loss of critical depth information when 3D data is projected into 2D and processed using CNNs. 2015.

[3] Charles R. Qi et al. "PointNet : Deep Learning on Point Sets for 3D Classification and Segmentation". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, p. 652-660.

[4] Romain Tavenard. *Livre sur les réseaux de neurones profonds*. 2025. url : `https://rtavenar.github.io/deep_book/fr/content/fr/mlp.html` (visité le 19/01/2025).