

TP6 - Nuages de Points et Modélisation 3D

Chloé Habasque - Colin Coërchon

26 Février 2025

1 Understand MLP for geometric 2D data

En choisissant la fonction **ReLU** comme fonction d'activation dans notre réseau de neurones, nous observons que l'entraînement prend beaucoup de temps avant de converger vers une solution satisfaisante.

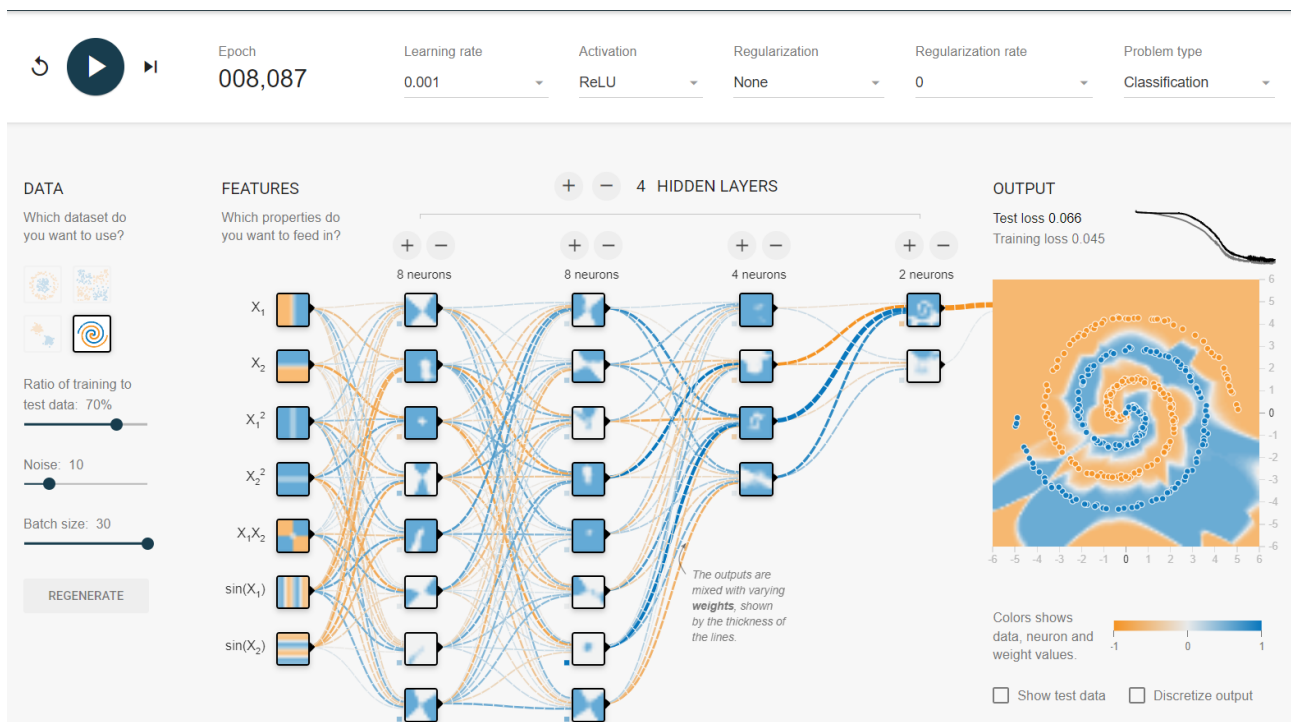


FIGURE 1 – Résultat du MLP de `playground.tensorflow.org` pour le "2D spiral input"

Les données sont compliquées à classifier en raison des 7 features que nous avons en entrée pour entraîner notre modèle. Comme le montre la figure 1, il aura fallu ≈ 5500 epochs avant d'arriver à un résultat convenable.

Note : Durant toute la suite du TP, nous avons utilisé un ordinateur avec son propre petit GPU.

2 MLP on point clouds in PyTorch

Question 1

Pour ModelNet10_PLY :

Nombre d'epochs	Temps d'exécution	Accuracy finale
250	193 secondes	17.1%

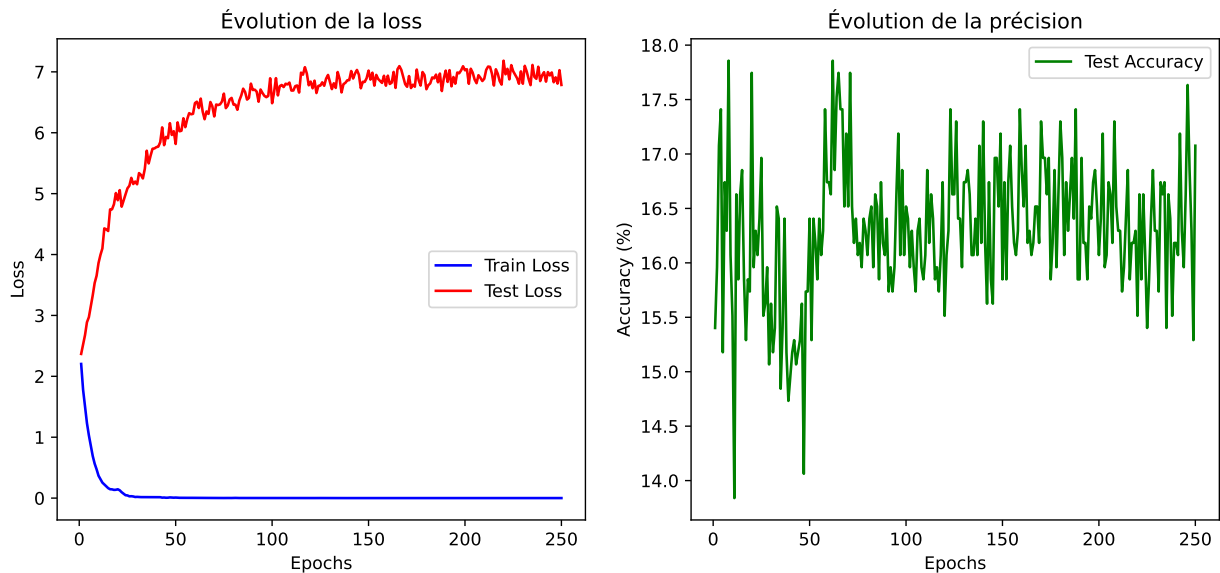


FIGURE 2 – Évolution de la loss et de l'accuracy pour ModelNet10 sur 250 epochs

Pour ModelNet40_PLY :

Nombre d'epochs	Temps d'exécution	Accuracy finale
250	487 secondes	12.3%

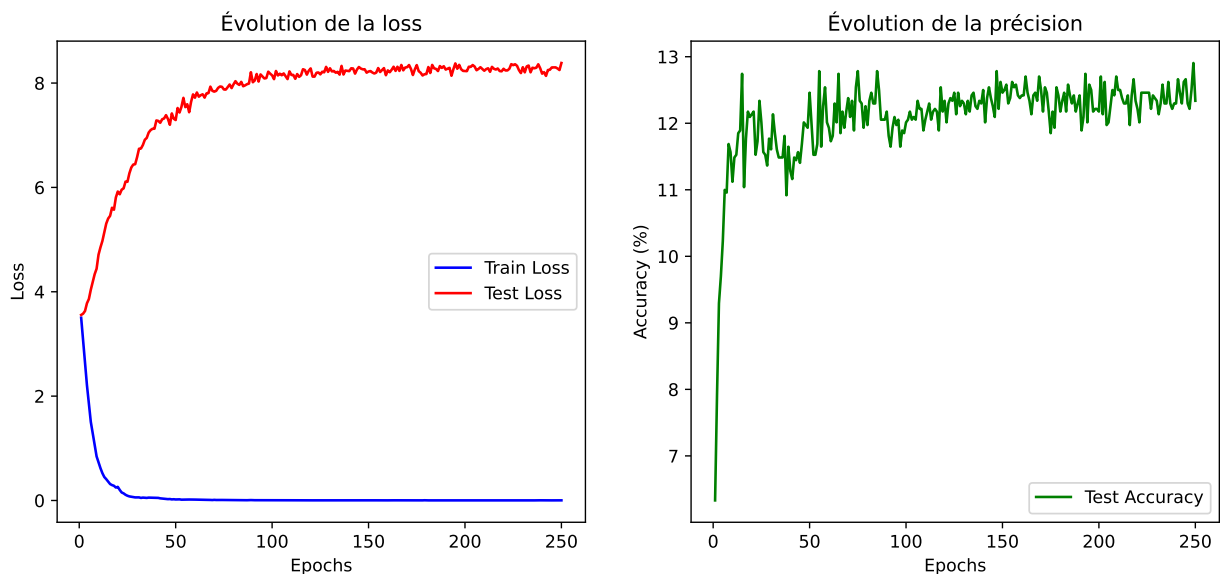


FIGURE 3 – Évolution de la loss et de l'accuracy pour ModelNet40 sur 250 epochs

Conclusion :

Nous avons tout d'abord testé un **MLP classique** (1 708 810 paramètres), qui consiste à aplatir les nuages de points 3D avant de les traiter comme des vecteurs d'entrée standards. Cependant, les résultats obtenus sont très décevants :

- Pour **ModelNet10_PLY**, l'accuracy finale est de **17.1%**.
- Pour **ModelNet40_PLY**, elle chute à **12.3%**.

De plus, la *test-loss* augmente progressivement, tandis que la *train-loss* diminue, ce qui suggère **un fort sur-apprentissage**. Cela indique que le MLP ne parvient pas à généraliser efficacement.

Ce résultat souligne les limites d'une approche qui ne respecte pas la nature 3D et l'invariance à la permutation des points.

3 PointNet in PyTorch

3.1 Basic version of PointNet

Question 2

Pour **ModelNet10_PLY** :

Nombre d'epochs	Temps d'exécution	Accuracy finale
50	298 secondes	89.8%
250	1 481 secondes	91.0%

TABLE 1 – Résumé des résultats avec PointNet sur ModelNet10

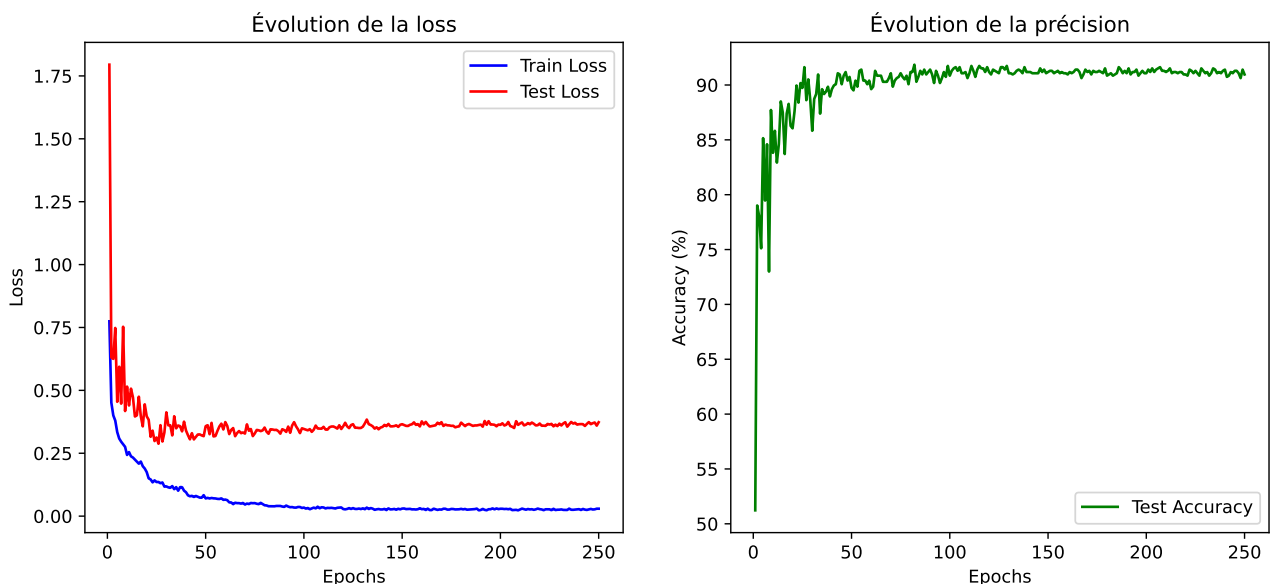


FIGURE 4 – Évolution de la loss et de l'accuracy pour ModelNet10 sur 250 epochs

Pour ModelNet40_PLY :

Nombre d'epochs	Temps d'exécution	Accuracy finale
50	740 secondes	84.3%
250	3 752 secondes	85.3%

TABLE 2 – Résumé des résultats avec PointNet sur ModelNet40

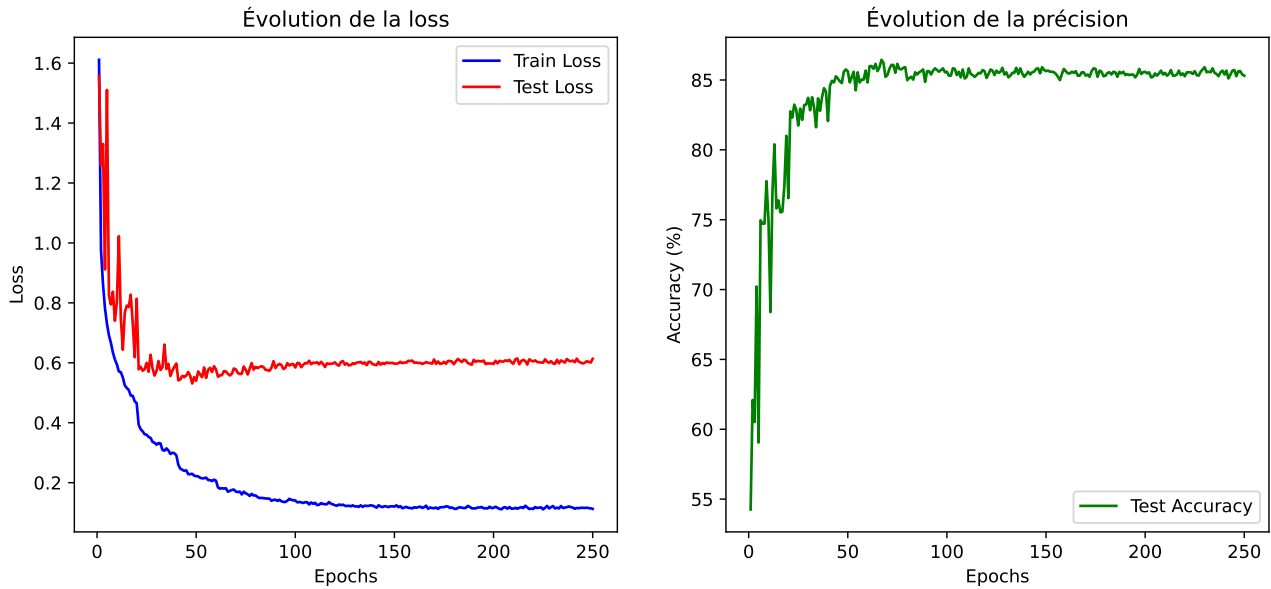


FIGURE 5 – Évolution de la loss et de l'accuracy pour ModelNet40 sur 250 epochs

Conclusion :

Nous avons ensuite évalué un **PointNet basique** (819 624 paramètres). Contrairement au MLP, PointNet atteint une précision bien plus élevée, avoisinant 91% en test sur ModelNet10_PLY.

Les figures 4 et 5 montrent l'évolution de la loss et de l'accuracy pour 250 epochs. On observe qu'une centaine d'epochs suffit souvent à atteindre le plateau. Au-delà, les performances stagnent.

L'amélioration de PointNet par rapport au simple MLP est significative, car **PointNet exploite la structure du nuage** et respecte l'invariance à la permutation des points, tandis que le MLP traite uniquement un vecteur aplati, perdant ainsi l'information spatiale essentielle.

3.2 PointNet with T-Net network

Question 3

Pour ModelNet10_PLY :

Nombre d'epochs	Temps d'exécution	Accuracy finale
50	565 secondes	90.7%
250	2 853 secondes	91.2%

TABLE 3 – Résumé des résultats avec PointNet + T-Net sur ModelNet10

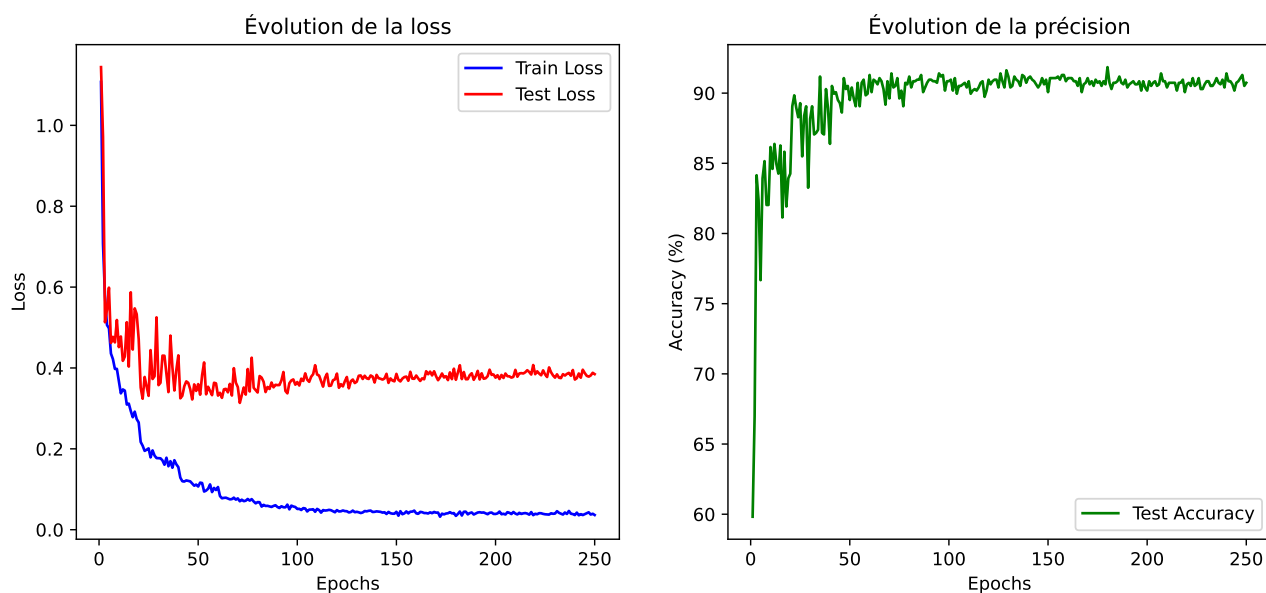


FIGURE 6 – Évolution de la loss et de l’accuracy pour ModelNet10 avec T-Net sur 250 epochs

Pour ModelNet40_PLY :

Nombre d’epochs	Temps d’exécution	Accuracy finale
50	1 422 secondes	86.4%
250	6 997 secondes	87.6%

TABLE 4 – Résumé des résultats avec PointNet + T-Net sur ModelNet40

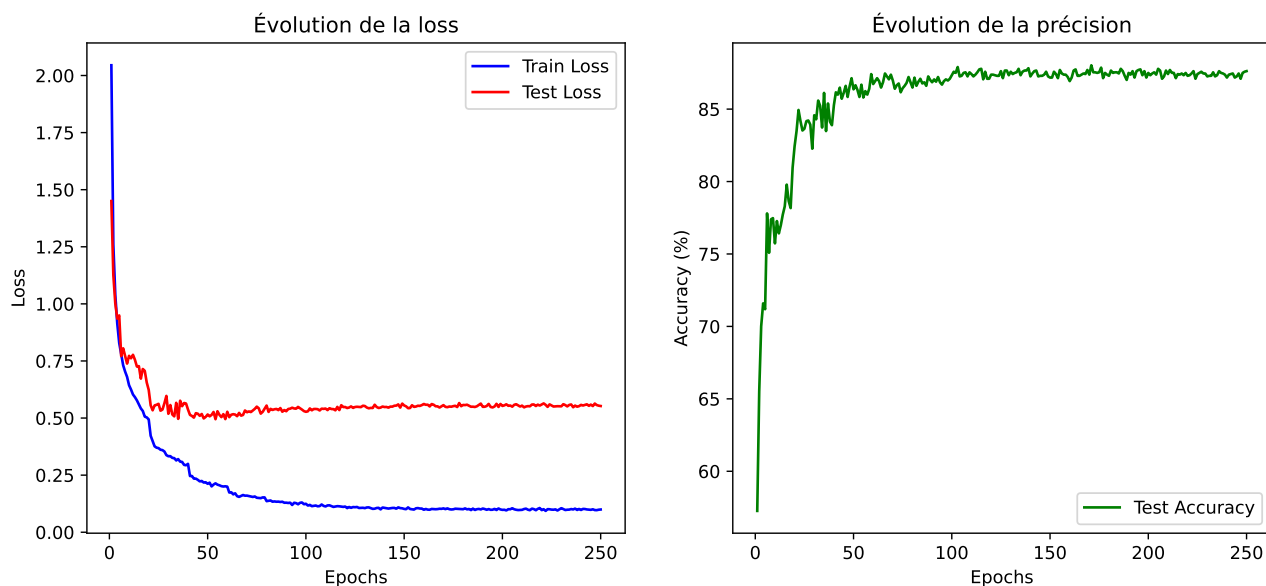


FIGURE 7 – Évolution de la loss et de l’accuracy pour ModelNet40 avec T-Net sur 250 epochs

Conclusion :

Nous avons ensuite évalué un **PointNet avec T-Net** (1 614 995 paramètres). Pour affiner l'alignement des nuages, nous avons intégré un T-Net (c'est un réseau PointNet réduit qui prédit une matrice de transformation 3×3) avant PointNet. Cependant, dans nos tests, **le gain en précision reste faible** (de l'ordre de +1 %, voire parfois équivalent à PointNet seul).

Les résultats restent quand même excellents, et 100 epochs suffisent en général à converger (voir figures 6 et 7). Toutefois, le modèle est presque deux fois plus lent que PointNet seul, ce qui pose la question du compromis entre précision et temps de calcul.

3.3 Data augmentation for 3D data

Question 4

Enfin, nous avons testé une *data augmentation* en plus des rotations selon z et l'ajout de bruit gaussien.

Nous avons implémenté une fonction `RandomFlipAndScale` qui, pour chaque nuage de points, applique un **flip aléatoire** (inversion du signe) sur l'axe x ou y et réalise ensuite un **re-scaling** aléatoire entre 0.9 et 1.1. L'idée est d'essayer de rendre le réseau plus robuste aux changements de symétrie et à de légères variations d'échelle.

Nous avons testé cette approche sur **80 epochs**, en utilisant PointNet avec T-Net.

Dataset	Accuracy finale
ModelNet10	91.9 %
ModelNet40	86.5%

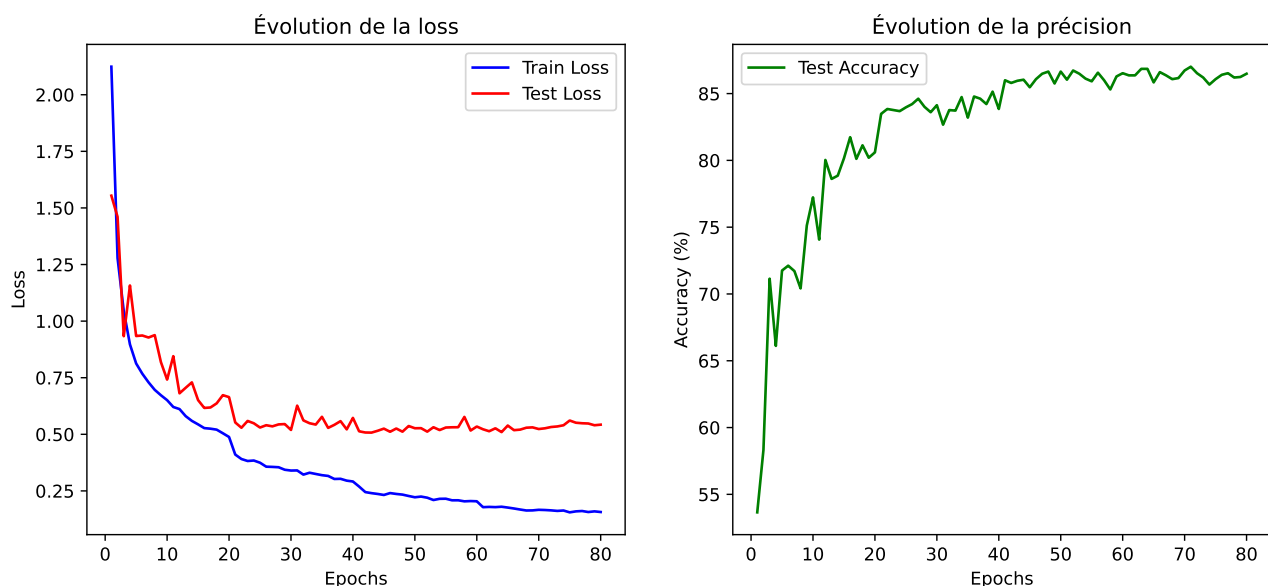


FIGURE 8 – Évolution de la loss et de l'accuracy pour ModelNet40 avec T-Net sur 80 epochs avec une *data augmentation*

Conclusion :

L'ajout de cette augmentation des données apporte un **très léger gain en précision** : moins de 1 %.

On constate donc une légère amélioration, mais cela suggère que PointNet est déjà assez robuste face aux variations mineures de symétrie et d'échelle. D'autres formes d'augmentations plus complexes pourraient être explorées pour essayer d'augmenter cette accuracy.

4 Going further (BONUS)

Nous n'avons pas traité la question bonus pour ce TP.