Projet de Machine Learning

ML

Amine YOUCEF KHODJA
Koceila KEMICHE
Groupe 01

Sorbonne Univeristé
Paris , France
20/05/2023

Table des matières

Introduction

L'implémentation de notre réseau de neurones est inspirée des anciennes versions de pytorch (en Lua) et des implémentations analogues qui permettent d'avoir des réseaux génériques très modulaires. Chaque couche du réseau est vu comme un module à part ainsi un réseau est constitué d'un ensemble de modules.

1 Problème Linéaire

Dans cette section, nous nous concentrons sur la résolution d'un problème de classification binaire en utilisant un réseau de neurones à une couche cachée linéaire (perceptron). Le problème que nous abordons est linéairement séparable, ce qui signifie que les deux classes peuvent être séparées par une ligne droite.

Pour évaluer les performances de notre modèle, nous avons généré deux ensembles de données simulées en utilisant deux distributions gaussiennes distinctes, représentant respectivement les deux classes (0 et 1). Nous avons divisé ces données en un ensemble d'entraînement et un ensemble de test. De plus, afin de sélectionner les meilleurs paramètres, nous avons subdivisé l'ensemble d'entraînement en un sous-ensemble de train et un sous-ensemble de validation.

Cela nous permettra d'évaluer la capacité de notre modèle à généraliser s de nouvelles données, En utilisant les données de validation, nous serons en mesure de sélectionner les meilleurs paramètres pour notre modèle, tels que le learning rate optimal, afin d'obtenir les meilleures performances possibles lors de la phase de test avec les données inconnues.

1.1 Optimisation des hyperparametres

Lors de l'optimisation de notre modèle, le choix du learning rate est crucial pour atteindre de bonnes performances. Pour déterminer le meilleur learning rate, nous avons testé plusieurs valeurs en nous entraînant sur les données d'entraînement. Ensuite, nous avons évalué les performances du modèle sur les données de validation pour chacune des valeurs de learning rate testées. Finalement, nous avons sélectionné le learning rate qui a donné les meilleures performances sur les données de validation. Cette approche nous permet de choisir le learning rate le plus adapté à notre modèle et qui permet d'obtenir les meilleures performances sur les données de test, ce qui est important pour assurer une bonne généralisation du modèle.

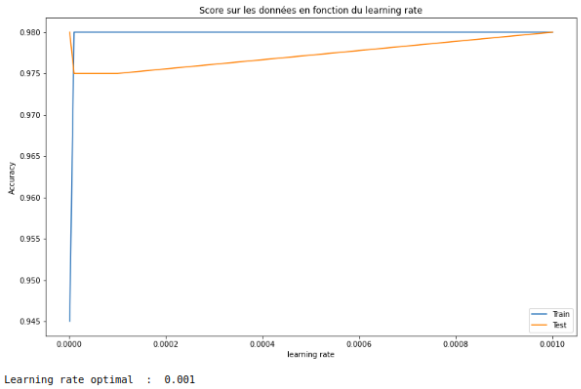


FIGURE 1 – Evolution de l'accuracy en fonction du learning rate pour des données centrés en 0

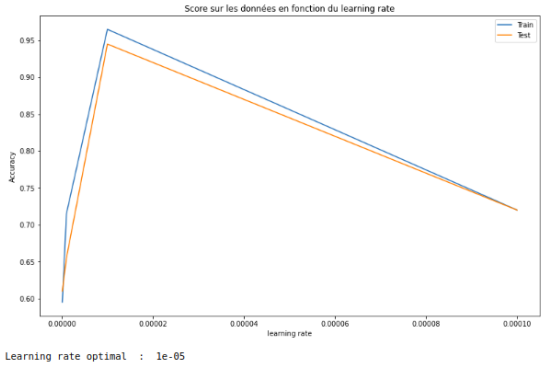


FIGURE 2 – Evolution de l'accuracy en fonction du learning rate pour des données non centrés en 0

1.1.1 Choix des paramètres

Nous constatons que le learning rate le plus optimal pour notre exemple de validation est de 0.001 pour les données centrés en 0 et 1e-5 pour les donnés non centrés en zeros. Nous avons donc décidé d'utiliser ces valeurs dans la suite de notre analyse pour entraîner et tester nos modèles.

1.2 Evaluation du réseau de neuronne optimal

1.2.1 Données centrés en 0

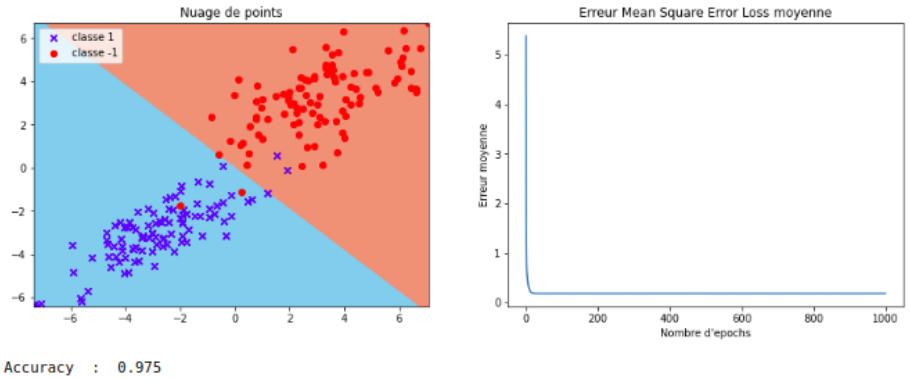


FIGURE 3 – frontières de décision et évolution du coût en Validation

1.2.2 Données non centrés en 0

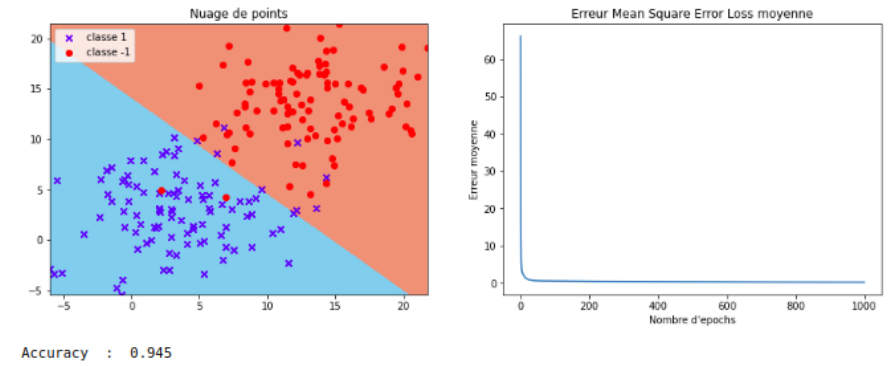


FIGURE 4 – frontières de décision et évolution du coût en Validation

1.2.3 Analyse

En utilisant le learning rate optimal et des données centrées et non centrés en 0, nous avons observé que la fonction de perte (loss) décroit de manière significative et atteint une valeur minimale proche de 0 après un certain nombre d'itérations pour les données d'entraînement et de validation. Cela montre que notre modèle est capable d'apprendre les caractéristiques discriminantes des données et de généraliser sur les données de validation.

1.2.4 Conclusion

Ces résultats démontrent que notre modèle est capable de s'adapter aux variations de la distribution des données et de maintenir de bonnes performances de classification. Même si les données ne sont pas centrées en 0, le modèle est capable de capturer les relations linéaires entre les variables d'entrée et de séparer efficacement les deux classes. Cela confirme la robustesse de notre modèle face à des variations dans la distribution des

données, ce qui est essentiel pour assurer une bonne généralisation lors de l'application du modèle à de nouvelles données.

En outre, le fait que la fonction de perte n'accroît pas pour les données de test suggère que le modèle n'est pas en situation de surapprentissage (overfitting), Ces résultats encourageants montrent que notre modèle est efficace pour résoudre le problème de classification binaire considéré.

2 Problème non Linéaire

Dans notre étude, nous avons exploré l'impact de différents taux d'apprentissage (learning rates) sur la performance de notre modèle de réseau de neurones. Nous avons testé quatre valeurs de taux d'apprentissage : 0.01, 0.001, 0.0001 et 0.00001

Pour chaque taux d'apprentissage, nous avons entraîné notre modèle avec un batch size de 50 et sur 1000 epochs. Notre architecture de réseau de neurones comprenait une couche linéaire d'entrée, suivie d'une fonction d'activation tangente hyperbolique, une deuxième couche linéaire, et enfin une fonction d'activation sigmoïde pour la classification binaire.

Pour étudier les performmance de notre algorithme, nous decidons de le tester sur deux probleme disticts : XOR et Dirac

2.1 Optimisation de parametres

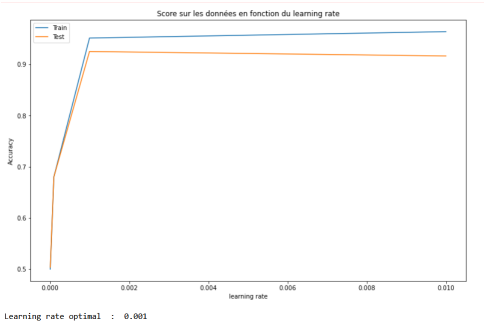


FIGURE 5 – Accuracy en fonction du learning rate pour le jeu de donné XOR

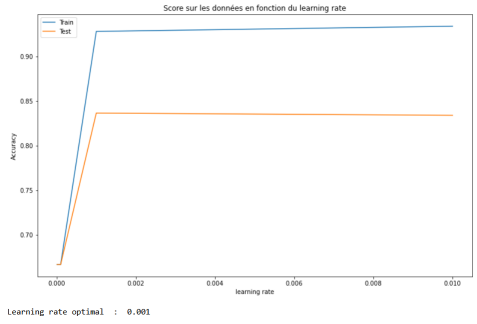


FIGURE 6 – Accuracy en fonction du learning rate pour le jeu de donné Dirac

2.1.1 Choix des parametres

Nous avons constatons que le learning rate le plus optimal pour notre exemple de validationt est de 0.001 pour les deux jeu de données. Nous avons donc décidé d'utiliser ces valeurs dans la suite de notre analyse pour entraîner et tester nos modèles.

2.2 Evaluation du réseau de neuronne optimal

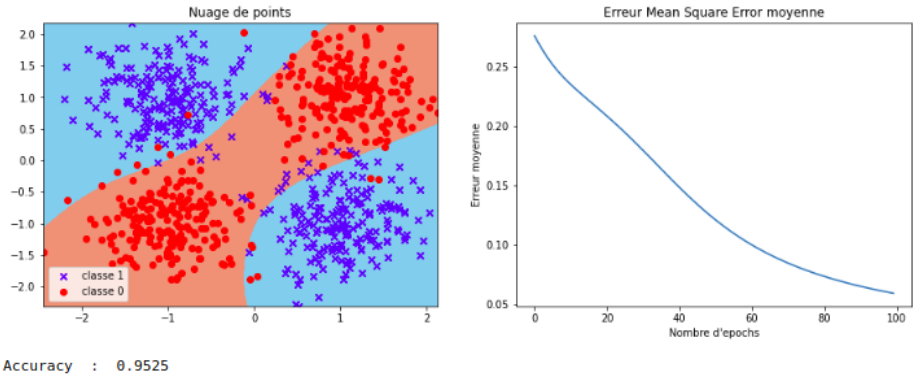


FIGURE 7 – Evaluation du jeu de donnée XOR

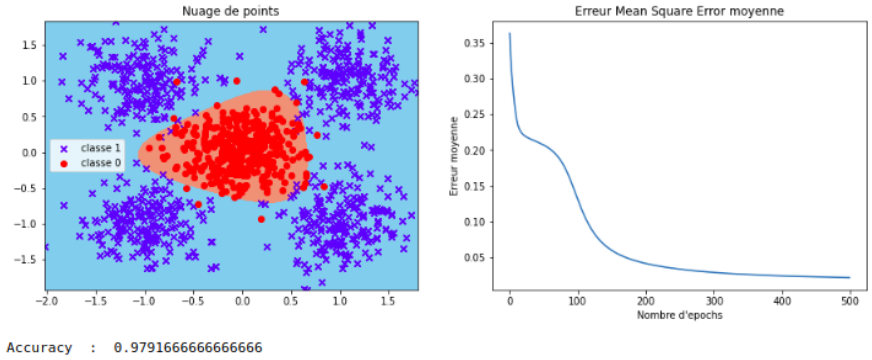


FIGURE 8 – Evaluation du jeu de donnée Dirac

2.3 Analyse

En utilisant le pas d'apprentissage optimal et des données non linéaires, nous avons observé une diminution significative de la fonction de perte au cours de l'entraînement de notre modèle. Après un certain nombre d'itérations, la fonction de perte a atteint une valeur minimale proche de zéro pour les données d'entraînement, ce qui démontre la capacité de notre modèle à apprendre les caractéristiques discriminantes des données non linéaires.

Nous avons ensuite évalué la performance de notre modèle sur un ensemble de données de test. Les résultats ont montré que notre modèle est capable de généraliser avec succès sur des données non linéaires, en obtenant des performances élevées en termes d'accuracy sur les données de test (0.95 pour le probleme XOR et 0.97 pour le Dirac).

2.3.1 Conclusion

Notre modèle a démontré sa capacité à apprendre à partir de données non linéaires et à généraliser avec succès sur de nouvelles données. Ces résultats renforcent la fiabilité et l'efficacité de notre modèle dans la résolution de problèmes de classification non linéaire.

3 Traitement automatique du language NLP

Notre objectif de recherche se concentre sur la classification des sentiments des avis de films dans le domaine du traitement automatique du langage. Pour cela, nous avons utilisé le modèle Word2Vec pour convertir les avis des utilisateurs en vecteurs.Une fois les avis des utilisateurs vectorisés, nous avons alimenté ces vecteurs dans notre réseau de neurones pour la classification des sentiments.

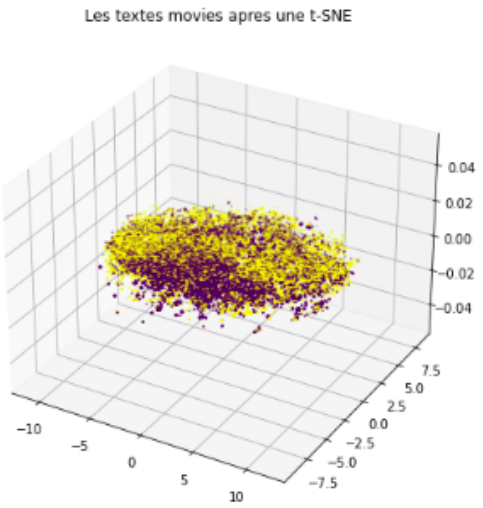


FIGURE 9 – Données de sentiments des films

Nous avons obtenu un accuracy de 82% sur l'ensemble d'entraînement et de 80% sur l'ensemble de test.

Nous avons également exploré la possibilité d'apprendre de nouveaux vecteurs de mots en utilisant un auto-encodeur. Nous avons vérifié que les données décodées étaient similaires aux données d'origine, car leurs différences en valeurs absolues étaient minimes, avec une moyenne de seulement 0,07. Cependant, nous n'avons pas réussi à reconstruire le mot d'origine en utilisant le nouveau vecteur de mot.

4 Problème MultiClasses

4.1 Choix du jeu de données

Nous avons fait le choix d'utiliser le jeu de données USPS pour notre projet. Ce jeu de données contient des images de chiffres de 0 à 9, ce qui en fait un choix idéal pour notre tâche de classification multiclasse.

4.2 Visualisation des données

Pour visualiser les données, nous avons utilisé des techniques de réduction de dimensionnalité telles que l'analyse en composantes principales t-SNE afin de projeter les données dans un espace bidimensionnel.

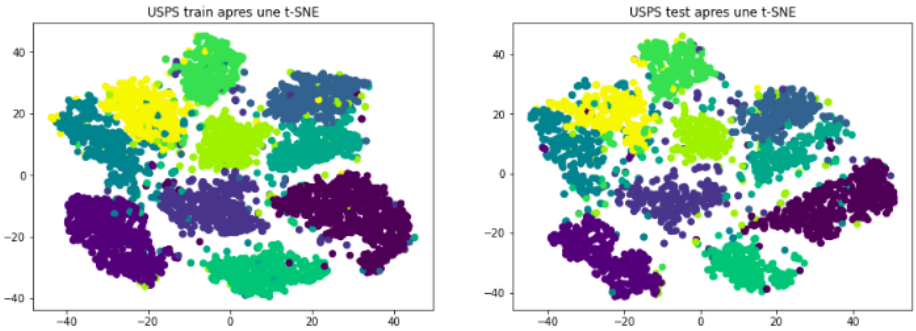


FIGURE 10 – Données USPS train et test

4.3 Optimisation des hyperparametres

4.3.1 Fonctions d'activations

Dans notre architecture de réseau de neurones, nous avons utilisé un total de trois couches linéaires. L'objectif principal de cette section est de trouver la meilleure combinaison de fonctions d'activation pour les deux premières couches linéaires, puis d'utiliser une activation softmax dans la dernière couche pour la classification multiclasse.

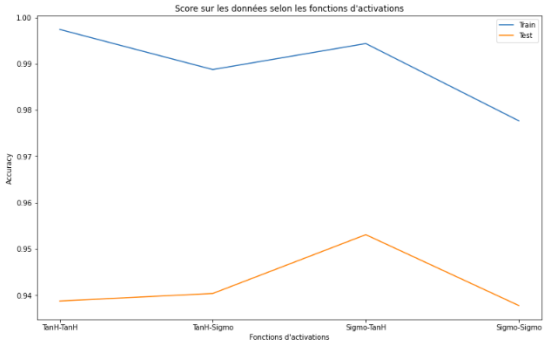


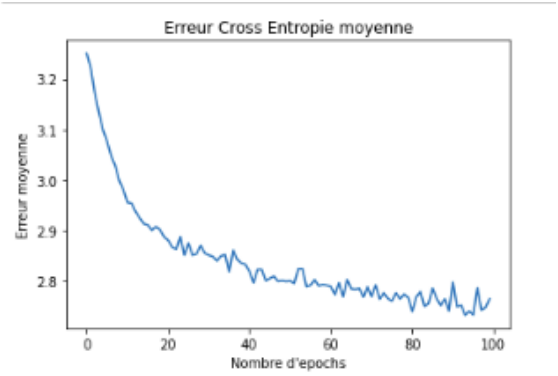
FIGURE 11 – Accuracy en fonction des combinaisons de fonction d'activation

On remarque un comportement intéressant dans les performances de notre réseau de neurones. Pendant la phase d'entraînement, la combinaison de fonctions d'activation tanh-tanh pour les deux premières couches linéaires a donné de meilleurs résultats en termes d'accuracy. Cependant, lors de la phase de validation, la combinaison sigmoïde-tanh a montré une performance supérieure.

Ce résultat peut être expliqué par la nature des deux fonctions d'activation utilisées. La fonction tangente hyperbolique (tanh) est symétrique et comprise dans la plage [-1, 1]. Elle permet au modèle de capturer des relations non linéaires complexes et d'apprendre des représentations plus riches des données, qui est justement le but principal la phase d'entrainement.Cepandant, lors de la validation l'accent est mis sur la capacité du modèle

à généraliser sur des données inconnues. La fonction sigmoïde peut aider à régulariser les sorties des neurones en limitant leur plage de valeurs entre 0 et 1, facilitant ainsi une meilleure séparation des classes et une prise de décision plus précise lors de la classification multiclasse. Nous faisons donc le choix d'utiliser la fonction sigmoïde comme activation pour la première couche linéaire, la fonction tanh comme activation pour la deuxième couche linéaire, et enfin, nous appliquerons une activation softmax dans la dernière couche.

4.4 Evaluation du reseau de neuronne optimal

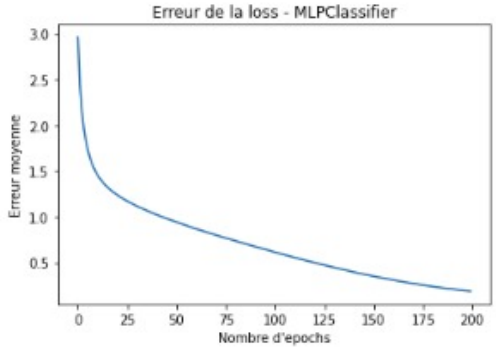



```
Optimisation finie au bout de  12  min
Train accuracy :  0.8226525949298227
Test accuracy :  0.7128712871287128
```

Après avoir évalué notre réseau de neurones sur notre jeu de données de validation, nous avons observé une diminution de la fonction de coût, ce qui indique que notre modèle a appris à ajuster ses poids et ses biais pour mieux correspondre aux données. Cependant, nous avons également remarqué des oscillations dans la courbe de la fonction coût, cela peut être expliqué par le fait que learning rate soit trop élevé (0.01).

Après avoir exploré différentes valeurs de learning rate plus petites, nous avons observé qu'elles nécessitaient considérablement plus de temps et un plus grand nombre d'epochs pour entraîner notre réseau de neurones. De plus, même avec ces valeurs réduites, nous avons constaté une différence minimale en termes de performances par rapport à notre learning rate initial. En tenant compte de ces observations, nous avons pris la décision de conserver la valeur de learning rate initiale pour notre modèle.

4.5 Comparaison des resultats avec MLPClassifier de Sckit-learn




```
Optimisation finie au bout de  13  min
Train Accuracy: 97.28%
Test Accuracy: 72.63%
```

Bien que des différences mineures puissent exister en raison de variations dans les détails de l'implémentation ou des paramètres spécifiques utilisés, les résultats globaux sont comparables, en effet sur les données de test l'accuracy est pratiquement la même, Cette convergence des performances confirme la validité de notre approche et souligne notre réussite à créer un réseau de neurones capable de traiter les tâches de classification multiclasse avec une précision satisfaisante.

5 Auto Encodeur

L'optimisation d'un auto-encodeur est essentielle pour garantir une représentation efficace de l'information d'origine. Cela implique de trouver un espace latent optimal, c'est-à-dire déterminer le nombre de dimensions idéal pour la couche finale de l'encodeur. Cette dimensionnalité joue un rôle crucial dans la capacité de l'auto-encodeur à encoder fidèlement toutes les caractéristiques de l'entrée d'origine.

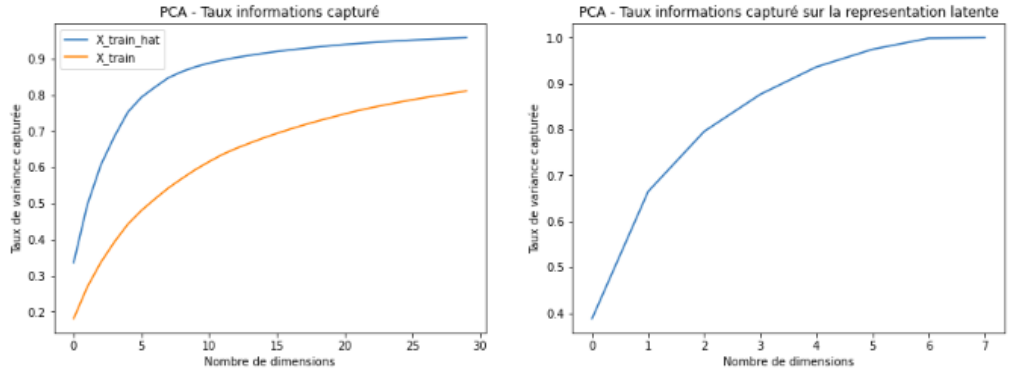
Une fois l'espace latent défini, la deuxième étape consiste à sélectionner soigneusement les couches et le nombre de neurones pour l'encodeur et le décodeur. Ces choix déterminent la complexité et la capacité de l'auto-encodeur à capturer les motifs et les détails pertinents de l'entrée, tout en évitant la surapprentissage ou la sous-représentation.

Outre ces considérations, la taille du batch utilisée lors de l'entraînement de l'auto-encodeur a un impact significatif sur ses performances de reconstruction des données d'origine. Un batch de taille adéquate peut permettre une meilleure généralisation et une convergence plus rapide lors de l'apprentissage, tandis qu'une taille inappropriée peut entraîner des problèmes de surcharge mémoire ou une convergence sous-optimale.

5.1 PCA

Dans le cadre de notre recherche visant à déterminer la dimension optimale de l'espace latent pour notre autoencodeur, nous avons entrepris une étude basée sur l'analyse en composantes principales (PCA). L'objectif était de déterminer le nombre de dimensions nécessaires pour capturer un pourcentage élevé d'informations à partir de l'espace d'origine.

Nous avons réalisé une PCA sur nos données d'origine pour obtenir les composantes principales. En examinant les valeurs propres, nous avons remarqué qu'un nombre restreint de dimensions principales capturaient la majeure partie de l'information. Nous avons fixé un seuil de 80% de variance totale et déterminé le nombre de dimensions nécessaires pour atteindre ce seuil. Cette première compression a été suivie d'une deuxième PCA pour réduire davantage la dimension de l'espace latent tout en maintenant un niveau élevé d'information.




```
La quantité d'information moyenne perdue pendant le decodage :  0.02902419221680825
```

FIGURE 12 – PCA

Cette approche de compression en deux étapes nous a permis d'avoir un apriori sur la dimension optimale de l'espace latent pour notre autoencodeur qui est de 8.

Verification de l'appriori : Après avoir utilisé notre apriori sur la dimension optimale de l'espace latent,nous observons le resultat suivant :

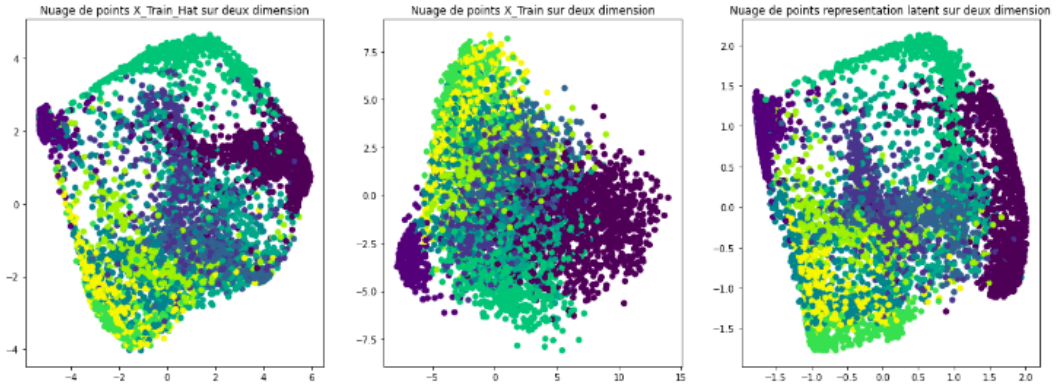
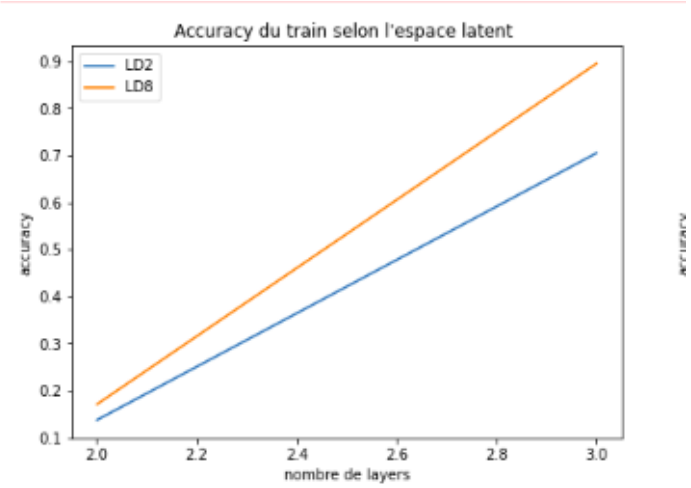


FIGURE 13 – PCA

Ces résultats confirment l'efficacité de notre méthode de compression en deux étapes et valident l'utilisation de la dimension 8 comme à priori pour l'espace latent de notre autoencodeur. Cette dimension optimale permet de maintenir une représentation compacte des données tout en conservant un niveau élevé d'informations pertinentes

5.2 Optimisations

Dans le cadre de notre étude, nous avons entrepris d'optimiser l'auto-encodeur en explorant différentes configurations pour l'espace latent et le nombre de couches. Nous avons testé plusieurs combinaisons en faisant varier la dimension de l'espace latent en utilisant l'apriori (on a pris une plage de 8 à 12) , le nombre de couches ainsi que les fonctions d'activations.




```
Auto encodeur optimal :
Nombre layer optimal :  3
Dimension optimale de l'espace latent :  8
Fonctions d'activation prises :
['TanH', 'Sigmoid', 'TanH']
['TanH', 'TanH', 'Sigmoid']
```

Après une analyse minutieuse des résultats obtenus, nous avons pu déterminer la configuration optimale pour notre réseau. Il s'est avéré que l'auto-encodeur offrait les meilleures performances lorsqu'il était composé de 3 couches linaires et que l'espace latent avait une dimension de 8 avec les fonctions d'activation qu'on voit sur la figure ci-dessus.

Cette configuration spécifique a permis d'atteindre un équilibre parfait entre la capacité de représenter l'information d'origine en sa totalité et la complexité du modèle. Les huit dimensions de l'espace latent ont démontré une capacité optimale à capturer et à encoder les caractéristiques pertinentes de l'entrée initiale. De plus, les trois couches ont permis une représentation hiérarchique des données, permettant ainsi de mieux apprendre les motifs et les structures sous-jacentes.

5.3 Expérimentations

5.3.1 Visualisations des images reconstruites après une forte compression

Après avoir créé l'auto-encodeur optimal, nous avons entrepris d'explorer ses capacités en termes d'encodage et de décodage des images USPS. Ces images ont été soumises à une compression significative à travers l'espace latent de notre auto-encodeur.

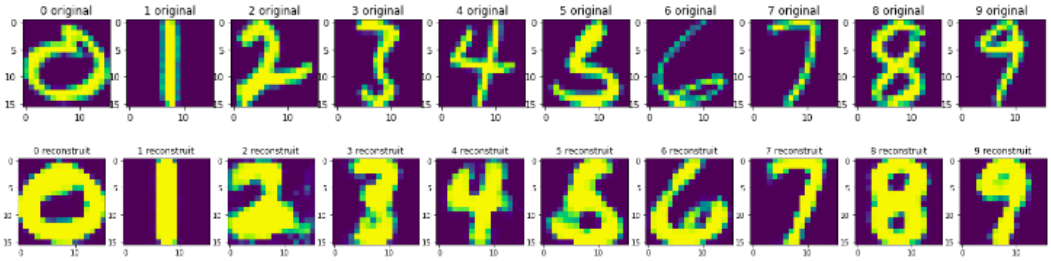


FIGURE 14 – Images USPS apres reconstruction

Les résultats obtenus ont été extrêmement prometteurs. Malgré la forte compression appliquée, l'auto-encodeur a réussi à reconstruire les images USPS avec une précision remarquable. Les détails et les caractéristiques essentielles des images ont été préservés, témoignant ainsi de la capacité de notre modèle à extraire et à encoder l'information de manière efficace.

5.3.2 Visualisations des representations obtenus dans un espace 2D

Dans notre étude, nous avons procédé à une étape de visualisation pour mieux comprendre les performances de notre auto-encodeur. Nous avons exploré les images encodées dans l'espace latent, ainsi que les images décodées à partir de cet espace.

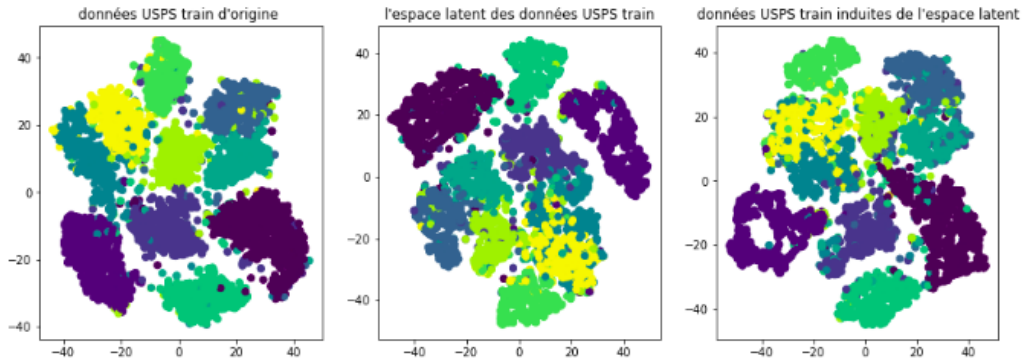


FIGURE 15 – t-SNE sur les données USPS

Cette étape de visualisation nous montre la capacité de l'auto-encodeur à représenter l'information dans l'espace latent et a confirmé l'efficacité de notre modèle dans la reconstruction précise des images à partir de cet espace.

5.3.3 Étude du clustering induit dans l'espace latent

Nous avons entrepris une étape de clustering sur l'espace latent de notre auto-encodeur afin de mieux comprendre les regroupements et les similarités des données. Nous avons ensuite utilisé l'algorithme t-SNE pour afficher un graphique qui met en évidence ces regroupements, à la fois avec les labels d'origine et avec les labels attribués par l'algorithme K-means sur les données de l'espace latent.

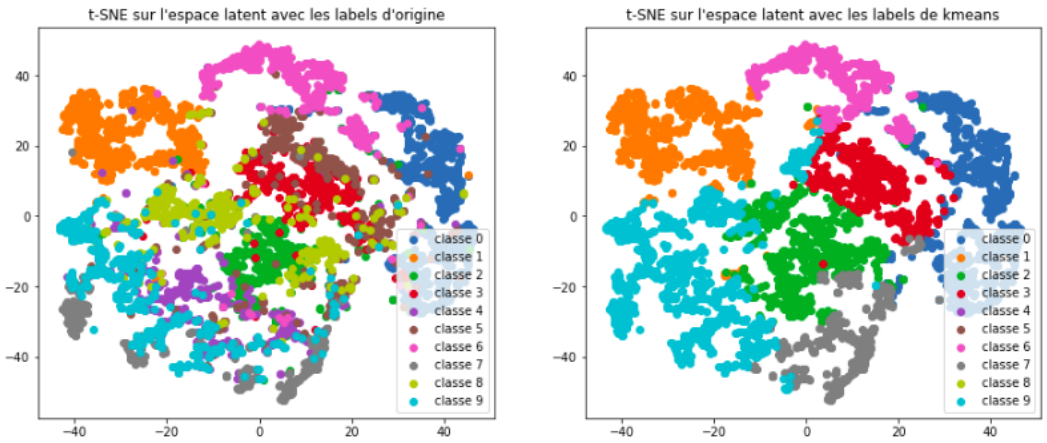


FIGURE 16 – t-SNE sur les données USPS

Lors de l'analyse de la représentation t-SNE, nous avons observé que l'attribution des labels par l'algorithme K-means sur les données de l'espace latent était remarquablement précise. Cela indique que le K-means a réussi à effectuer un regroupement optimal des données dans l'espace latent, ce qui témoigne de la capacité de cet espace à encoder efficacement l'information d'origine. En d'autres termes, cette observation renforce notre confiance dans la qualité de l'encodage réalisé par l'espace latent de notre auto-encodeur. Les regroupements cohérents et précis obtenus par le K-means sur les données de cet espace suggèrent que les caractéristiques et les informations essentielles des données d'origine ont été fidèlement encodées.

5.3.4 Évaluation de l'espace latent sur un réseau de neurones multi-classe

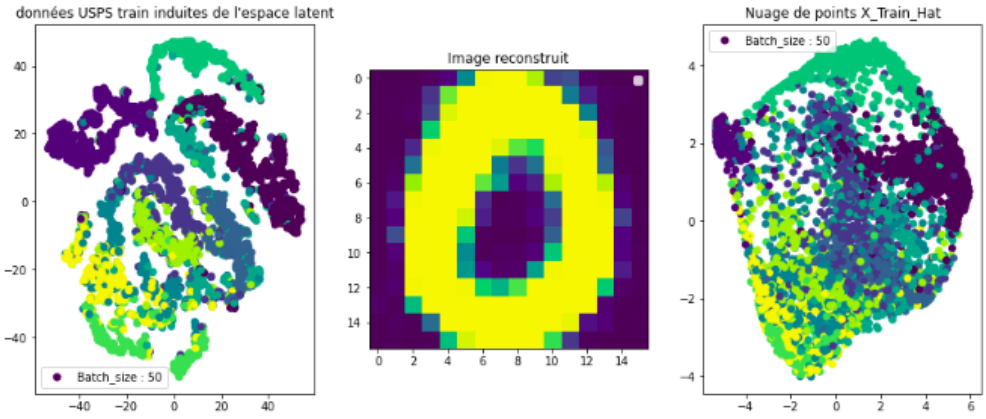
Dans notre étude, nous avons procédé à une évaluation de l'espace latent sur un réseau de neurones multi-classes.Pour pouvoir évaluer les performances , on a opté pour deux métriques à savoir l'accuracy et à la pureté du clustering issue de l'espace concerné.

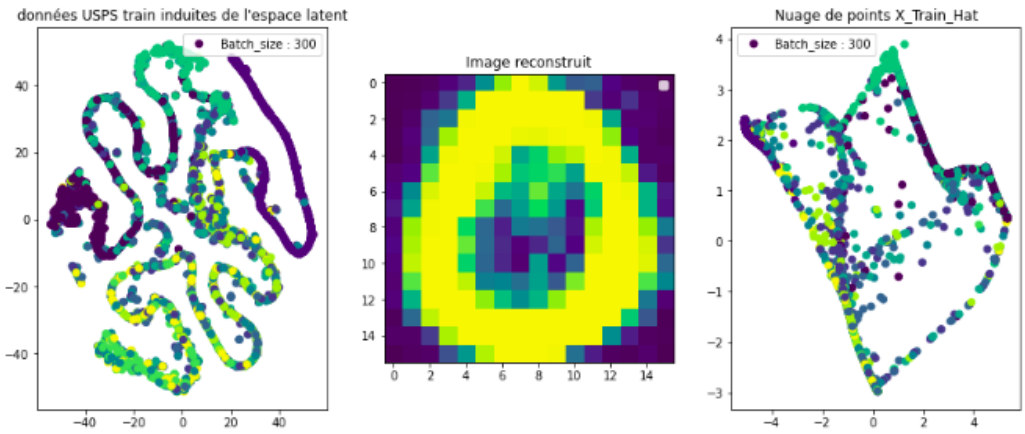
```
Optimisation de :  X
100%|██████████████████████████████████████████| 50/50 [00:19<00:00,  2.51it/s]
Accuracy sur les images issues de l'auto encodeur 0.9945416599775245
Pureté du clustering :  0.30325895007224274
Rand score  :  0.4087318489675728
Adjusted Rand Score :  0.06473570192554079

Optimisation de :  Representation Latente
100%|██████████████████████████████████████████| 50/50 [00:01<00:00, 38.70it/s]
Accuracy sur les images issues de l'auto encodeur 0.691764328142559
Pureté du clustering :  0.4026328463637823
Rand score  :  0.5966058545022128
Adjusted Rand Score :  0.15440074493551928

Optimisation de :  X_hat
100%|██████████████████████████████████████████| 50/50 [00:19<00:00,  2.52it/s]
Accuracy sur les images issues de l'auto encodeur 0.8950072242735592
Pureté du clustering :  0.2949109006261037
Rand score  :  0.4292577975291778
Adjusted Rand Score :  0.06833288127541266
```

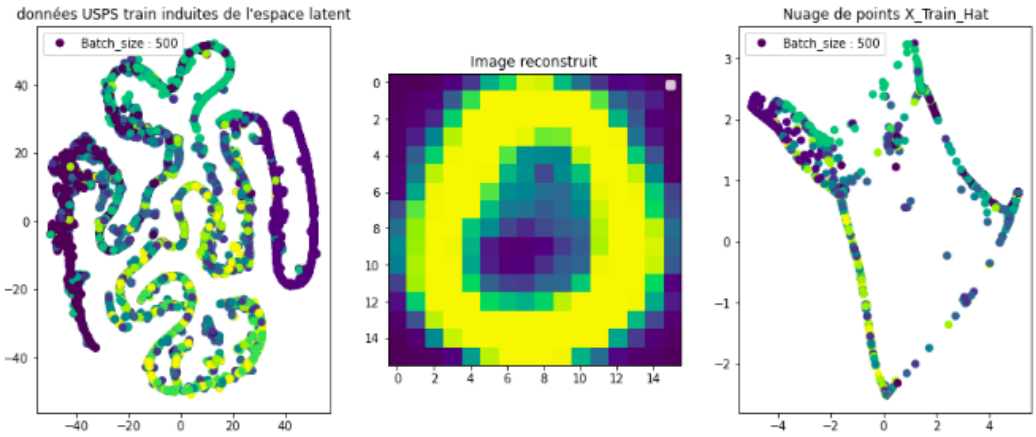
Les résultats obtenus sont significatifs : l'exactitude (accuracy) des prédictions réalisées directement sur l'espace latent atteint 70%. Cependant, lorsque nous utilisons les données décodées à partir de cet espace latent, nous constatons une augmentation notable de l'exactitude, atteignant un taux de 89%. Nous avons observé que la pureté du clustering était maximale dans l'espace latent plutôt que dans l'espace d'origine des données. Cette constatation s'explique par le fait que l'information encodée au niveau de l'espace latent est dépourvue de bruit et de redondance.Cette représentation compressée de l'espace d'origine des données, qui peut contenir des variations et des informations moins discriminantes facilite ainsi le processus de clustering.Cela favorise la formation de clusters plus purs, où les points similaires se regroupent plus étroitement, tandis que les points dissimilaires sont séparés avec une plus grande clarté.

5.3.5 Impact de la taille du batch

Nous menons une étude visant à évaluer l'effet de la taille du batch sur les performances de notre modèle.







En augmentant le nombre de batch, nous réduisons la quantité d'exemples pris en compte lors du calcul de l'erreur. Cela signifie que le modèle effectue des corrections en se basant sur un échantillon limité de données, ce qui perturbe l'apprentissage et conduit à des résultats réduits. En revanche, en réduisant le nombre de batch, nous prenons en compte un plus grand nombre d'exemples et avons ainsi la possibilité de couvrir l'ensemble des classes possibles.

Lorsque le nombre de batch est réduit, l'erreur calculée est une moyenne globale qui prend en compte un échantillon plus représentatif de l'ensemble des données. Par conséquent, lors des corrections, le modèle dispose d'une vision plus complète et plus équilibrée de l'ensemble des classes, ce qui favorise l'obtention de meilleurs résultats.

5.3.6 Étude des performances en débruitage

Après avoir identifié l'auto-encodeur optimal, nous avons entrepris de tester ses performances en matière de débruitage. Pour ce faire, nous avons pris l'ensemble de données d'entraînement initial, auquel nous avons ajouté un bruit gaussien ou poisson. Ensuite, nous avons utilisé l'auto-encodeur pour encoder les données bruitées et les décoder afin de les reconstruire.

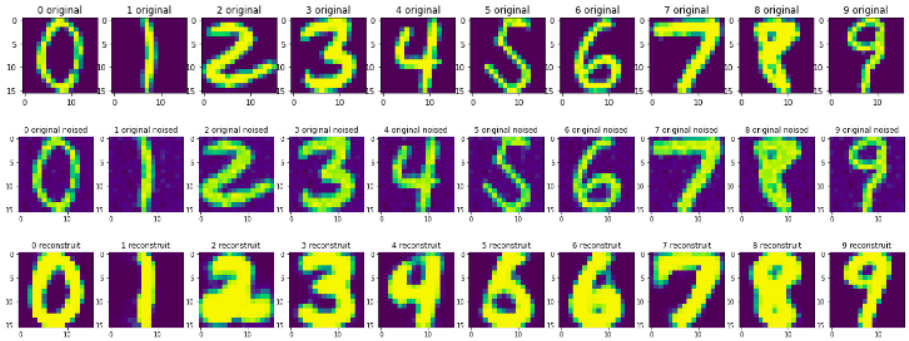


FIGURE 17 – Données originales, bruitées, reconstruites

L'autoencodeur a réussi à réduire considérablement le bruit dans les données d'entrée, ce qui est confirmé par une nette amélioration de la qualité de la reconstruction, ainsi, les données débruitées présentent une ressemblance plus étroite avec les données d'origine,

ce qui indique que l'autoencodeur a réussi à capturer les caractéristiques essentielles des données tout en éliminant le bruit indésirable.

5.3.7 Adaptabilité de l'autoencodeur à différents types de bruit

Nous souhaitons évaluer l'adaptabilité de l'autoencodeur à différents types de bruit en faisant varier leur intensité. Nous avons choisi de tester deux variantes de bruit, à savoir le bruit gaussien et le bruit poisson, afin d'explorer la capacité de l'autoencodeur à traiter des sources de bruit distinctes.

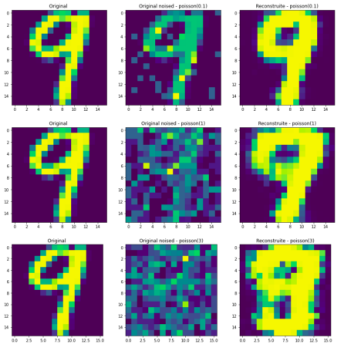


FIGURE 18 – Bruit poisson

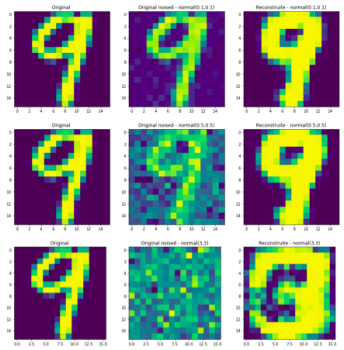
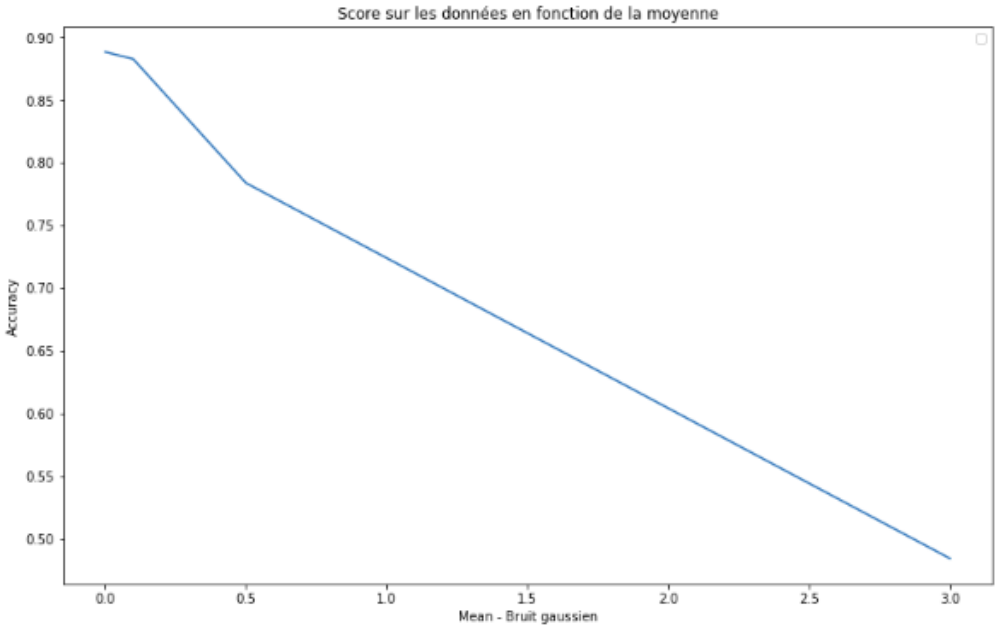


FIGURE 19 – Bruit poisson

Ces résultats mettent en évidence l'adaptabilité de l'autoencodeur à différents types de bruit, qu'il s'agisse de bruit gaussien ou de bruit poisson. L'autoencodeur a démontré sa capacité à réduire efficacement le bruit, quelle que soit la nature de celui-ci, tout en préservant les caractéristiques essentielles des données.

5.3.8 Évolution de l'accuracy en fonction du bruit

Nous affichons une courbe de l'accuracy en fonction de la moyenne du bruit gaussien appliqué sur les données.



Lorsque nous introduisons du bruit dans les données, cela perturbe les informations originales et ajoute des variations indésirables. En conséquence, la capacité du modèle à distinguer les classes et à effectuer des prédictions précises est compromise. Le bruit agit comme une perturbation qui masque ou déforme les caractéristiques discriminantes, rendant ainsi la tâche de classification plus difficile.

5.3.9 Analyse

Lorsque les données bruitées sont introduites dans l'autoencodeur, l'encodeur apprend à ignorer les variations inutiles et à se concentrer sur les informations essentielles. Ainsi, la représentation latente obtenue par l'encodeur est moins sensible au bruit. Lorsque cette représentation latente est ensuite utilisée par le décodeur pour reconstruire les données, le décodeur essaie de reproduire les caractéristiques originales des données sans inclure le bruit indésirable, ce qui explique la bonne capacité du modèle à reproduire les images même lorsqu'elles sont très bruitées. Cependant, il est important de noter que cette capacité de débruitage a ses limites. Lorsque le niveau de bruit devient trop élevé, l'auto-encodeur peut rencontrer des difficultés à supprimer complètement le bruit et à reconstruire fidèlement les caractéristiques d'origine des données.

5.3.10 Conclusion

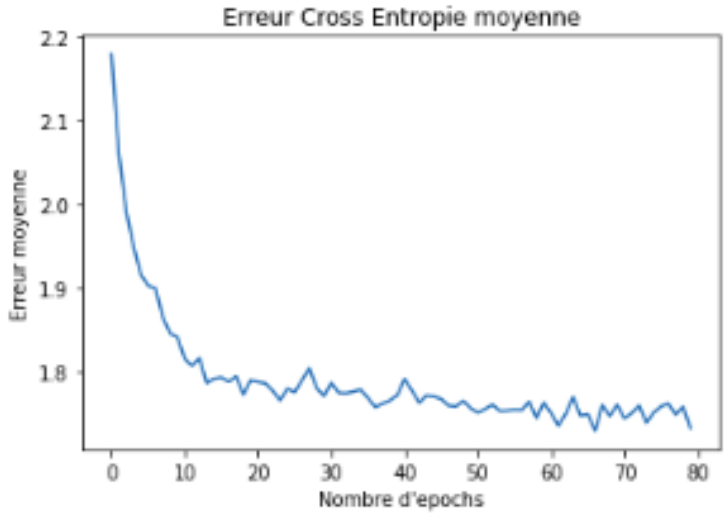
En conclusion, notre étude sur l'autoencodeur a montré des résultats prometteurs en termes de débruitage des données et de reconstruction de haute qualité. Cependant, il y a toujours des améliorations possibles à considérer pour renforcer les performances du modèle.

Une approche courante pour améliorer l'autoencodeur consiste à utiliser les matrices de poids transposées de l'encodeur pour le décodeur, ce qui permet une régularisation dans le réseau. Ce partage de poids entre l'encodeur et le décodeur favorise une représentation plus compacte des données et peut contribuer à améliorer la capacité du modèle à généraliser.

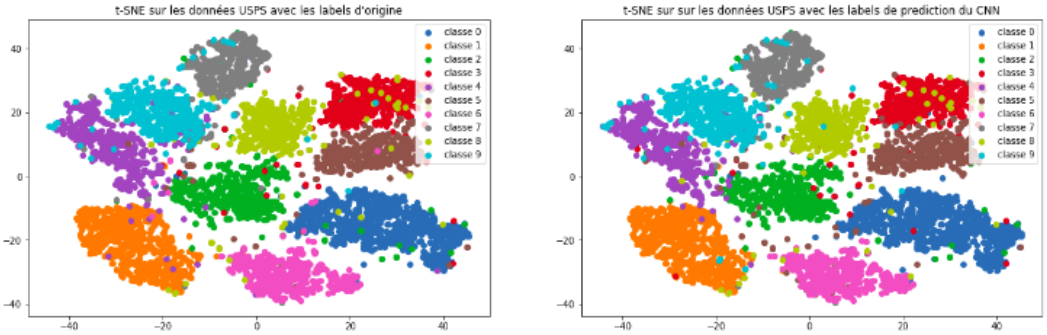
6 Réseau de neurones convolutionnel CNN

6.1 Données USPS

Nous avons utilisé une architecture de réseau de neurones convolutif comprenant des couches Conv1D, MaxPooling1D et Flatten, suivies de couches linéaires avec des activations ReLU. En particulier, la couche Conv1D a utilisé une fonction d'activation ReLU pour introduire de la non-linéarité dans le réseau. Enfin, la dernière couche linéaire était suivie d'une activation Softmax pour obtenir des probabilités de classification pour chaque classe.



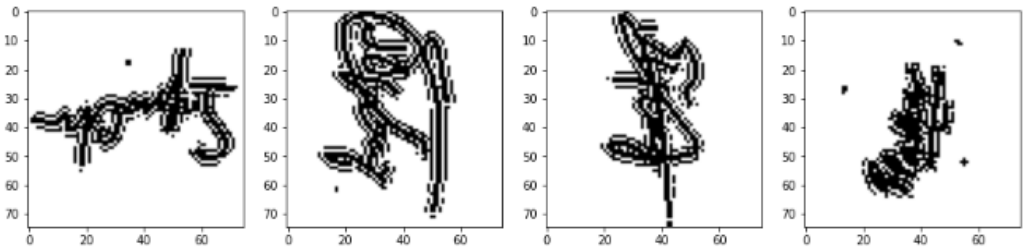

```
Accuracy  :  0.9627548563172259
CNN sur les données USPS
Optimisation finie au bout de  69  min
```

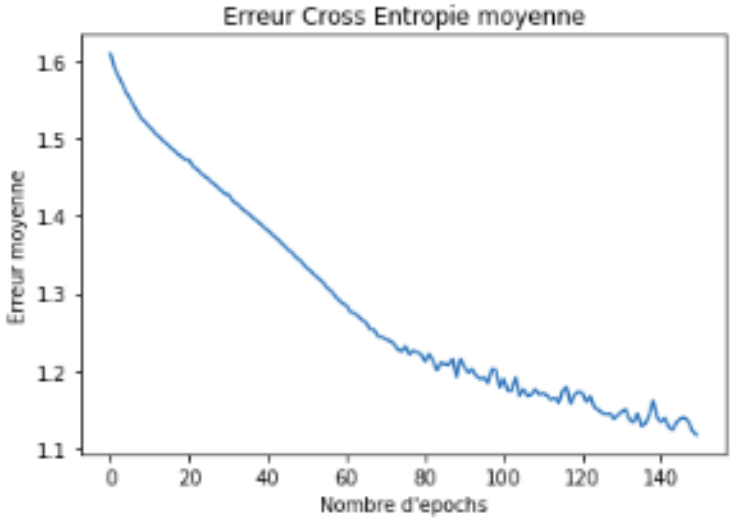


nous avons obtenu une précision de classification de 96% sur les chiffres manuscrits. Cependant, le temps d'exécution total pour entraîner le modèle a été de 69 minutes. Pour de futures améliorations, nous pourrions explorer des stratégies d'optimisation pour réduire le temps d'exécution sans compromettre les performances du modèle. En conclusion, notre réseau de neurones convolutif a démontré son efficacité dans la classification des chiffres USPS, mais des optimisations peuvent être envisagées pour améliorer le temps d'exécution.

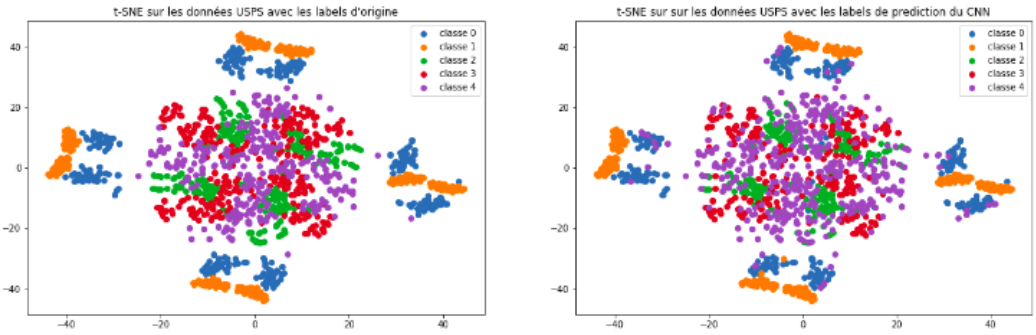
6.2 Données Signatures

Pour renforcer les capacités de notre modèle, nous avons enrichi notre ensemble de données en utilisant une base de données de signatures. Cependant, la taille de cette base de données était relativement petite, ce qui aurait pu limiter les performances de notre réseau de neurones convolutif (CNN). Pour surmonter cette limitation, nous avons appliqué une technique de data augmentation en effectuant des rotations sur les images de signature. Cela nous a permis de générer de nouvelles variations de chaque image, augmentant ainsi la diversité des données disponibles pour l'entraînement. Ensuite, nous avons entraîné notre CNN en utilisant cet ensemble de données augmenté, ce qui a permis d'améliorer la généralisation du modèle et d'obtenir de meilleures performances de classification sur les signatures.






```
Accuracy  :  0.87625
CNN sur les données signature
Optimisation finie au bout de  42  min
```



Après avoir appliqué la data augmentation et entraîné notre CNN sur la base de données de signatures augmentée, nous avons obtenu des résultats prometteurs. Notre modèle a atteint une précision de classification de 87%, ce qui démontre son efficacité dans la tâche de reconnaissance de signatures. Le temps d'entraînement total pour parvenir à ces performances a été de 42 minutes, ce qui est relativement raisonnable compte tenu de la complexité de l'apprentissage profond. Ces résultats montrent que l'utilisation de la data augmentation a contribué à améliorer la capacité de notre modèle à généraliser et à reconnaître les différentes variations des signatures.

7 Conclusion

En conclusion,Nous avons réalisé des expériences sur des ensembles de données variés, notamment l'ensemble USPS pour la classification de chiffres manuscrits et une base de données de signatures.

ce projet nous a permis de développer une compréhension approfondie des techniques de classification linéaires, non linéaires et multiclasse,auto-encodeur, ainsi que des réseaux de neurones convolutionnels. Les résultats obtenus ont montré le potentiel de ces approches pour la classification et la reconnaissance d'images. L'exploration continue de ces techniques et l'introduction d'améliorations supplémentaires pourraient conduire à des avancées encore plus significatives dans ces domaines.