Sorbonne Université

M1 Informatique - DAC
Compte rendu ML

Réseau de neurones : DIY

Etudiants :
Ghiles Ouhenia
Amayas Sadi

Encadrant :
Nicolas Baskiotis

5 mai 2023

Table des matières

1 Introduction

Ce rapport présente l'implémentation d'un réseau de neurones modulaire inspiré des anciennes versions de PyTorch. Chaque couche du réseau est considérée comme un module, offrant une grande flexibilité et modularité. Les fonctions d'activation sont également traitées comme des modules. L'objectif principal de ce projet est de détailler les étapes d'implémentation, de la conception de l'architecture à l'implémentation des fonctions d'activation et des méthodes d'apprentissage. Des choix techniques et des résultats expérimentaux seront également discutés. Cette expérience offre une opportunité de renforcer notre compréhension des réseaux de neurones et d'acquérir des compétences en programmation et en manipulation de modèles d'apprentissage automatique.

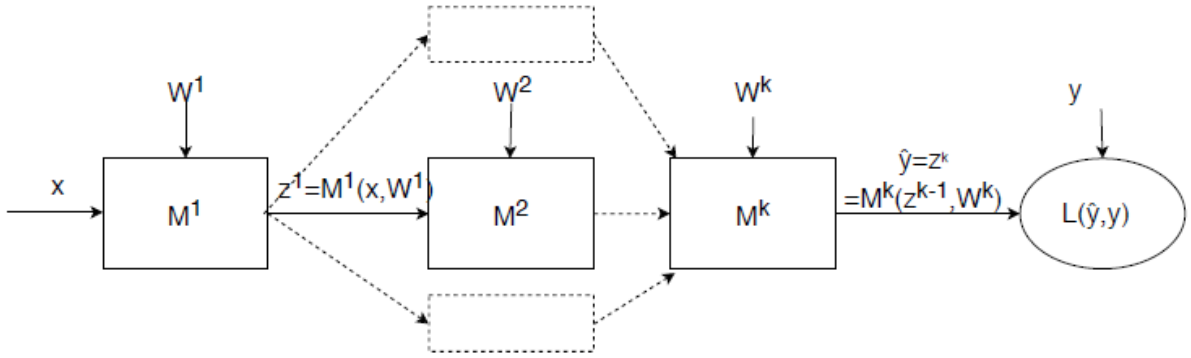


FIGURE 1 – Architecture module d'un réseau

2 Linéaire

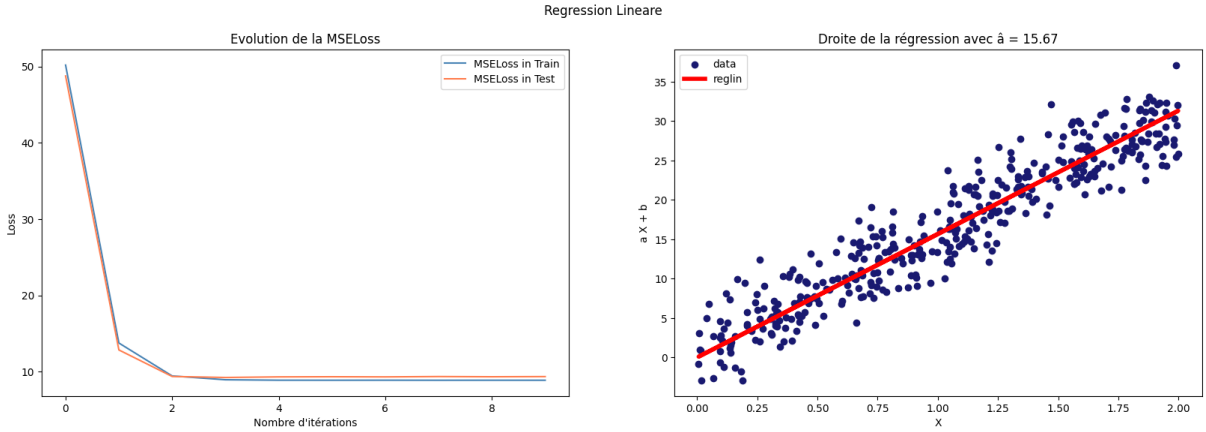
Dans la première phase de notre projet, nous nous sommes concentrés sur la mise en place des bases de nos réseaux de neurones. Pour cela, nous avons créé deux classes clés : MSELoss et Linear.

La classe MSELoss est utilisée pour mesurer l'écart quadratique moyen entre les prédictions et les valeurs supervisées, tandis que la classe Linear représente une couche linéaire dans notre réseau.

2.1 Régression linéaire

Pour commencer, nous mettons en place un réseau de régression linéaire qui se compose d'une seule couche linéaire. Ce réseau a pour objectif de minimiser la fonction de coût des moindres carrés.

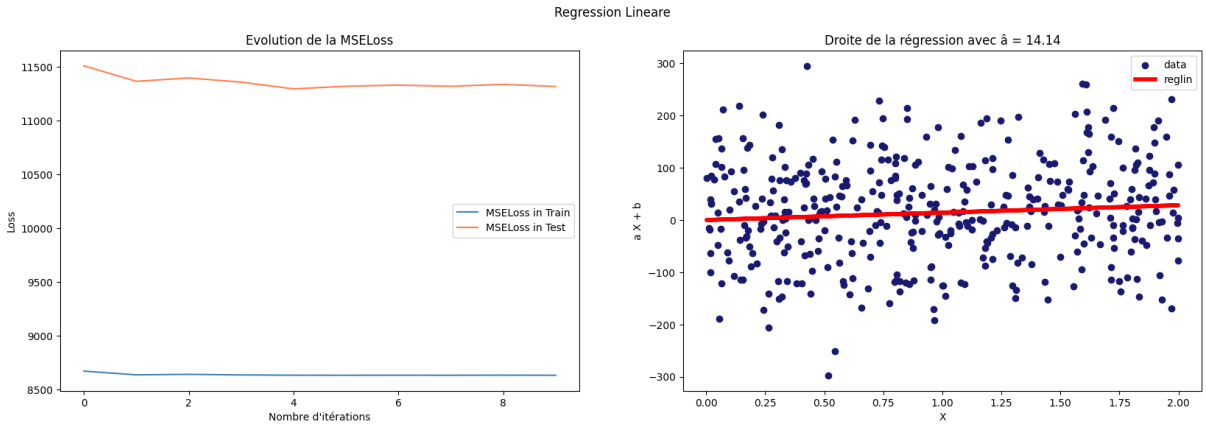
Nous générons un ensemble de points (x, y) tel que $y = a \cdot x + b + \epsilon \cdot \sigma$.



a = 15, b = 1, sigma = 3

Dans notre première illustration, les points représentent les données d'entrée du réseau neuronal, tandis que la droite tracée correspond à la fonction $f(x) = W \cdot x + b$. Nous constatons que la droite de régression linéaire obtenue s'ajuste bien aux données.

Nous allons maintenant augmenter le bruit (σ) à 100 afin de voir son comportement :



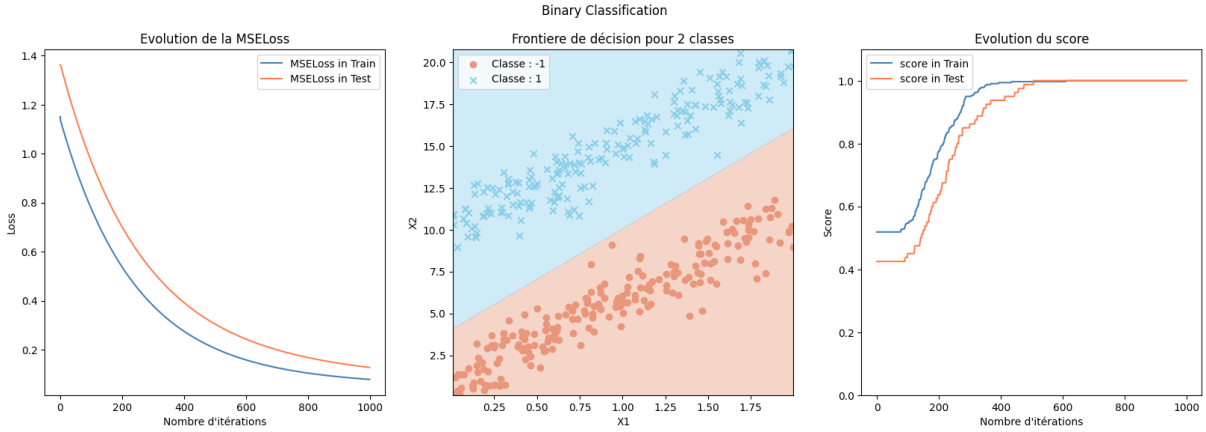
a = 15, b = 1, sigma = 100

Une observation importante est que l'ajout de bruit entraîne une augmentation significative de notre MSE et éloigne notre valeur estimée â de la vraie valeur a.

De plus, une différence considérable est observée dans l'évolution de la MSE entre les données de test et d'entraînement, contrairement à ce qui était observé dans la première figure.

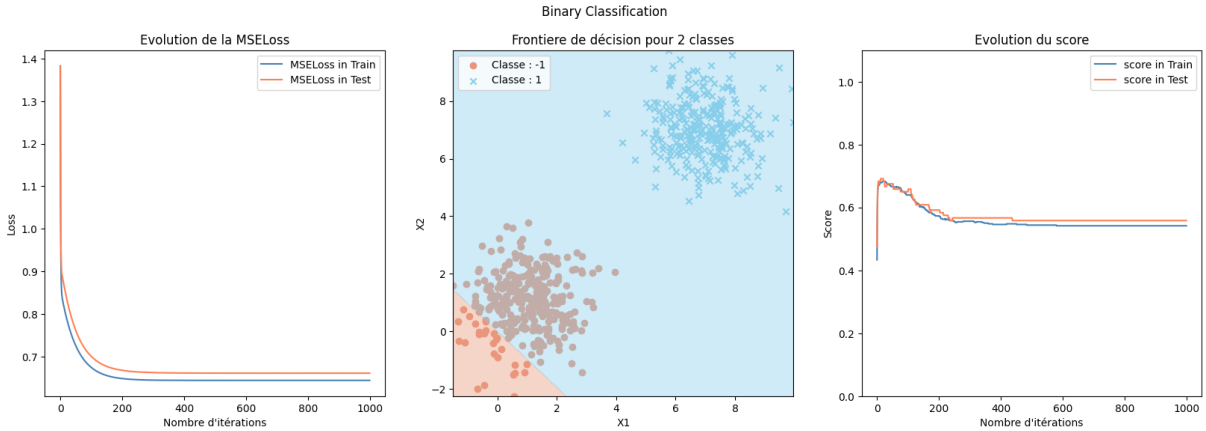
2.2 Classification

Nous allons maintenant aborder un problème de classification où notre objectif est de séparer des classes binaires linéairement séparables. Cela signifie que nous cherchons à trouver une frontière de décision linéaire qui peut diviser les données en deux classes distinctes.

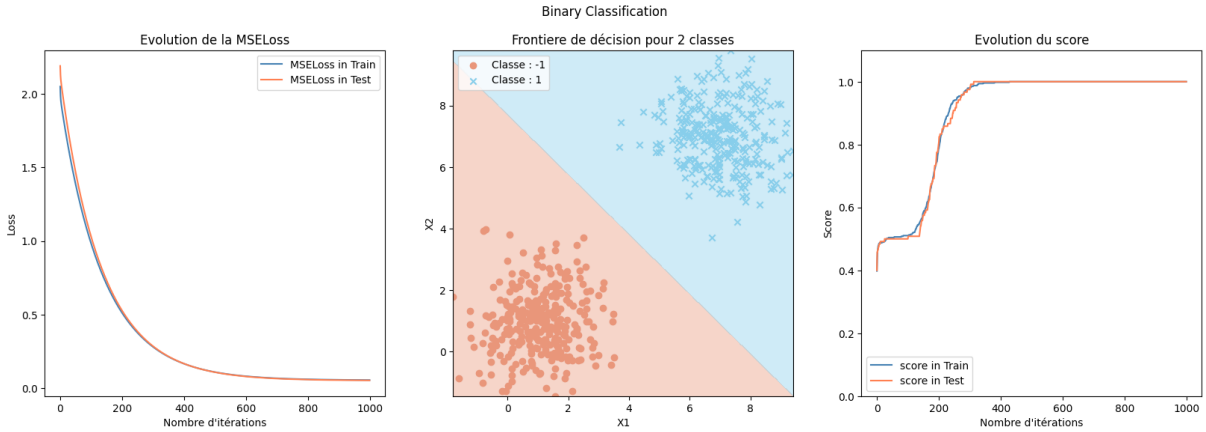


Pour le premier problème, le modèle linéaire a réussi à bien séparer les deux classes, ce qui se reflète par une faible valeur de MSE et un score de classification de 100%.

Maintenant, nous allons nous concentrer sur la classification de données non centrées :



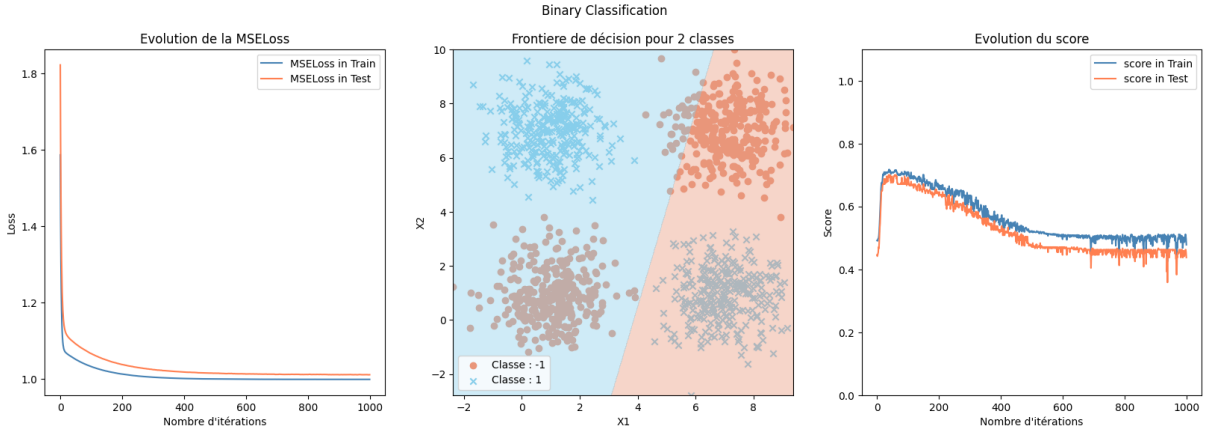
Nous constatons qu'en l'absence de biais, notre classifieur éprouve des difficultés à se déplacer loin de l'origine. Afin de remédier à cette situation, il est essentiel d'introduire un terme de biais pour résoudre ce problème et permettre au classifieur de s'adapter de manière appropriée.



3 Non Linéaire

Lorsqu'il est nécessaire de résoudre des problèmes de classification complexes, l'utilisation de modèles non linéaires est souvent incontournable. Contrairement aux modèles linéaires, qui cherchent à délimiter une frontière de décision linéaire, les modèles non linéaires ont la capacité de capturer des relations plus complexes et flexibles entre les caractéristiques d'entrée et les étiquettes de sortie.

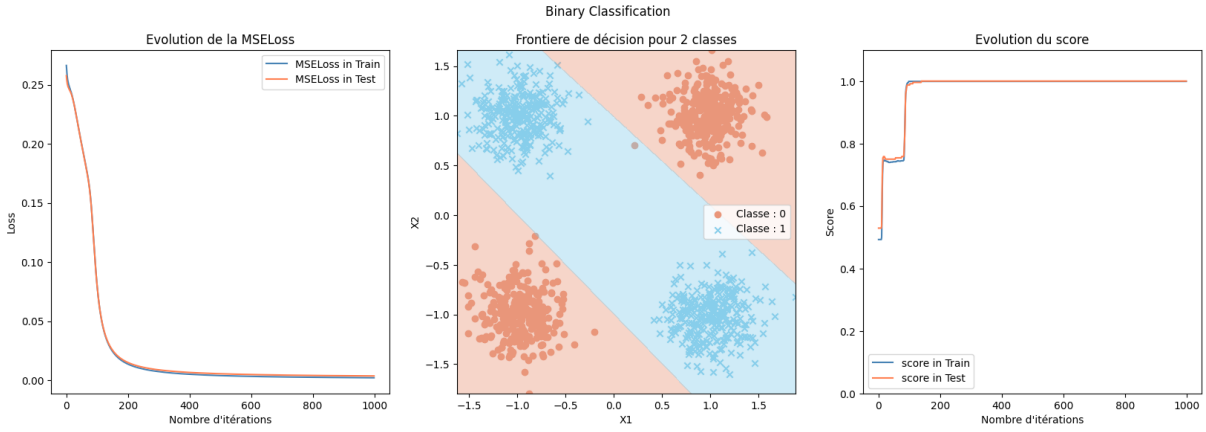
Un exemple classique de la limitation des modèles linéaires est le problème du XOR. En utilisant un modèle linéaire, il est impossible de tracer une seule ligne droite pour séparer les exemples positifs et négatifs du XOR.



La figure ci-dessus montre que le XOR ne peut pas être efficacement résolu par des modèles linéaires, en raison de leur limitation à représenter des frontières de décision linéaires.

Cependant, l'utilisation de modèles non linéaires tels que les réseaux de neurones avec des couches cachées et des fonctions d'activation non linéaires (Sigmoïde, Tanh et ReLU) permet de résoudre plus précisément le problème du XOR et d'autres problèmes de classification non linéaires.

3.1 Réseau à deux neurones

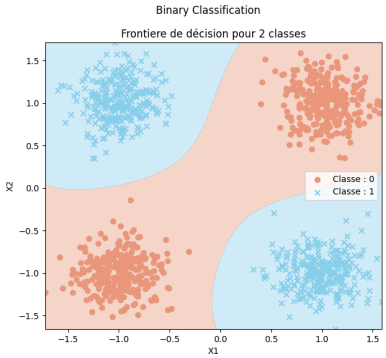


Une observation importante est que, même avec seulement deux neurones et deux couches, nous parvenons à bien séparer nos données en utilisant deux droites distinctes. Cela permet de résoudre efficacement le problème de classification du XOR.

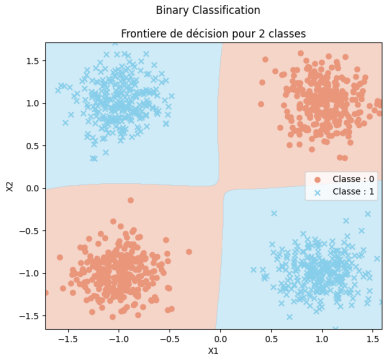
3.2 Réseau multi-neurones

En augmentant le nombre de neurones, nous nous attendons à une amélioration de nos résultats.

3 neurones



5 neurones



10 neurones

En effet, nous observons que l'ajout de couches supplémentaires permet d'améliorer les performances de classification. Toutefois, cette approche présente un inconvénient significatif : elle augmente considérablement le temps de calcul requis pour entraîner le modèle.

3.3 D'autres problèmes plus complexe

Nous allons maintenant évaluer les performances de notre réseau neuronal sur deux bases de données plus complexes : un échiquier et une sphère. Ces bases de données présentent des structures plus complexes que les exemples précédents et nous permettront de tester la capacité de notre réseau à apprendre des motifs plus sophistiqués.

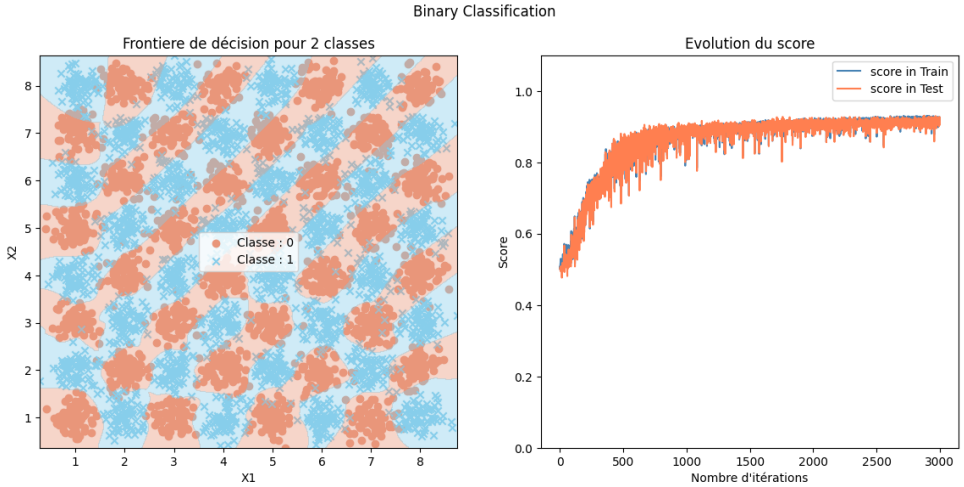


FIGURE 7 – 3 neurones

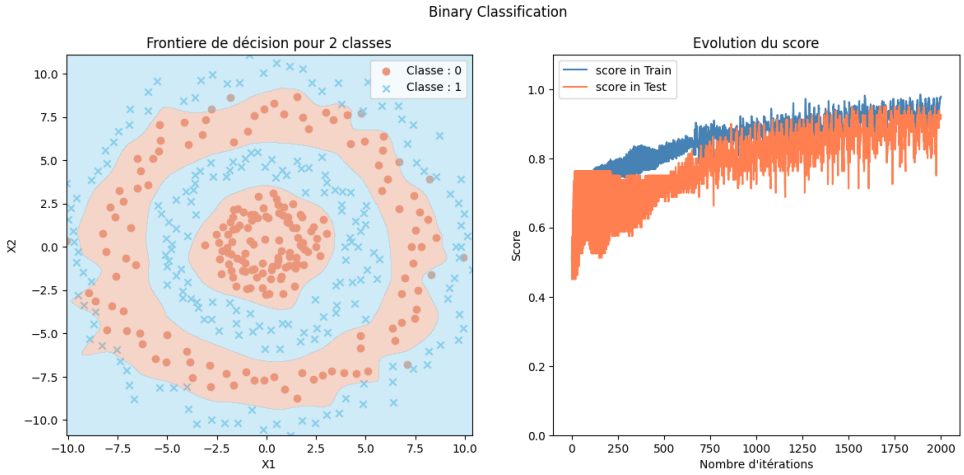


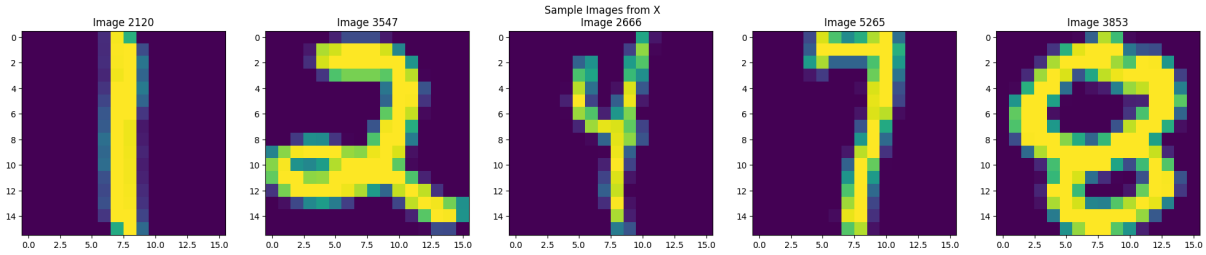
FIGURE 8 – 5 neurones

Nous observons que plus les données deviennent complexes, plus il est difficile d'obtenir une convergence rapide du réseau neuronal. Cela nécessite d'augmenter le nombre d'itérations, ce qui entraîne un temps de calcul plus long. Il est important de trouver un équilibre entre la précision des résultats et le temps de calcul lors de l'entraînement sur des données complexes.

4 Multiclasse

4.1 Données USPS

Dans le cadre de cette problématique, nous avons choisi d'utiliser les données fournies par USPS, qui consistent en des images de chiffres. Nous sommes donc confrontés à un problème de classification à 10 classes, où notre objectif est d'attribuer correctement chaque image à sa classe respective.



Nous allons procéder au lancement d'un réseau de neurones ayant l'architecture suivante afin d'observer les résultats obtenus.

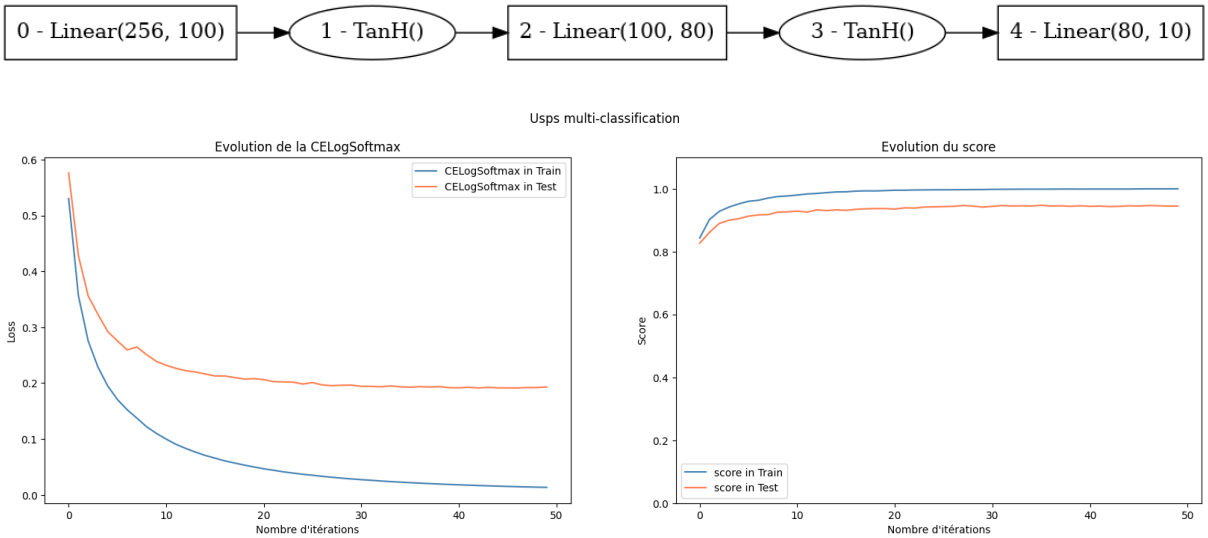


FIGURE 9 – 3 couches, 100 neurones

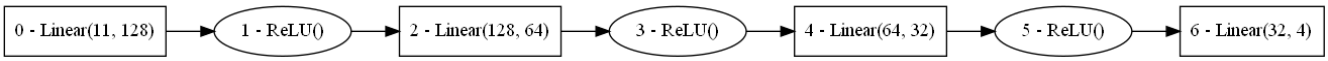
Après avoir lancé le réseau de neurones, nous observons un taux de bonne classification d'environ 93%, ce qui est assez satisfaisant. Cela signifie que le modèle parvient à classifier correctement la plupart des échantillons. De plus, il est intéressant de noter que le score aléatoire serait d'environ 10%, ce qui souligne l'efficacité du modèle dans la tâche de classification.

4.2 Données body Performance

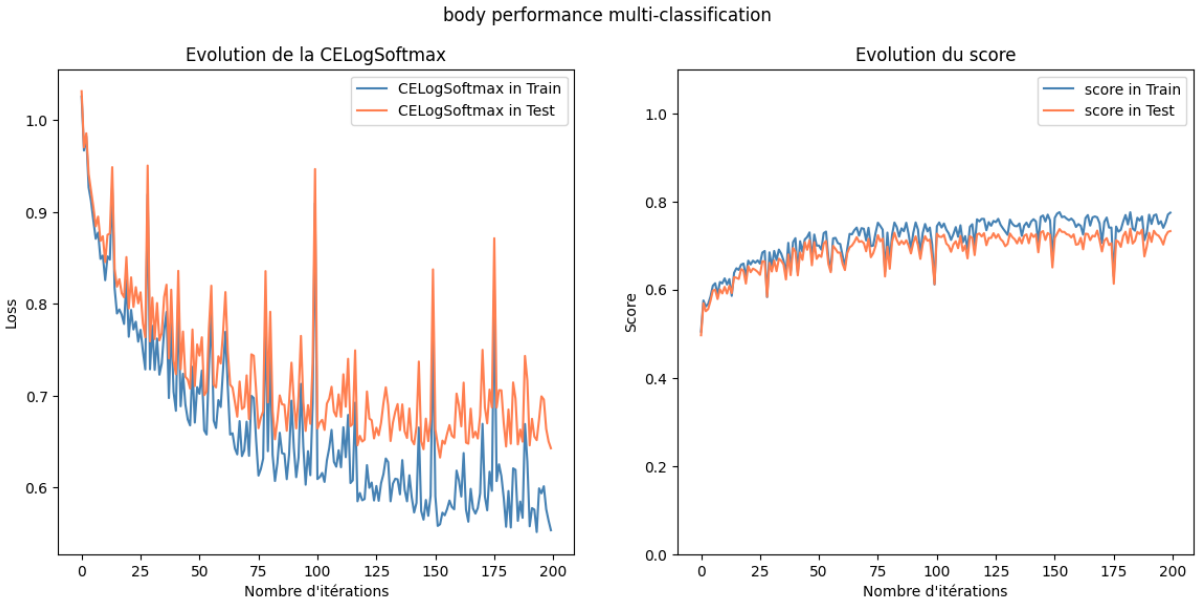
Afin de comparer les performances de notre réseau, nous avons décidé d'utiliser une base de données concrète issue de kaggle. Elle est constituée de 11 colonnes décrivant les données d'un athlète et possède une colonne de classe avec 4 valeurs possibles (A, B, C, D) qui correspondent aux niveaux de performance.

age	gender	height	weight	fat	diastolic	systolic	grip	flexion	sit-ups	broad
27.0	1	172.3	75.2	21.3	80.0	130.0	54.9	18.4	60.0	217.0
25.0	1	165.0	55.8	15.7	77.0	126.0	36.4	16.3	53.0	229.0
31.0	1	179.6	78.0	20.1	92.0	152.0	44.8	12.0	49.0	181.0

Nous avons pris la décision de configurer notre réseau de neurones avec les couches suivantes :



Après avoir entraîné notre réseau, nous avons obtenu les résultats suivants :



Nous avons obtenu un taux de réussite de 75%. Cela démontre que notre réseau a été capable de bien classifier les données de test. Parallèlement, nous avons également utilisé un modèle SVM pour effectuer la même tâche et avons obtenu un score de 64%, il convient de noter que les résultats du SVM pourraient être améliorés en optimisant ses paramètres.

Néanmoins, ces résultats encourageants indiquent que notre réseau de neurones est performant dans cette tâche et démontre son potentiel pour la classification.

5 Auto encodeur

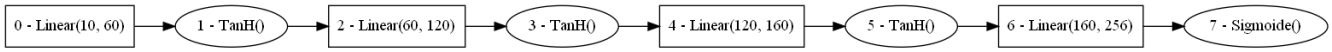
5.1 Encodage d'un chiffre

Nous reprenons la base de données fournie par USPS que nous avons utilisée précédemment.

5.1.1 Variation des hyperparamètres

Dans le cadre de notre démarche, nous procédons à l'entraînement de notre auto-encodeur dont l'architecture est :

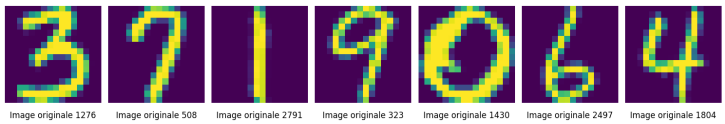
Encodeur



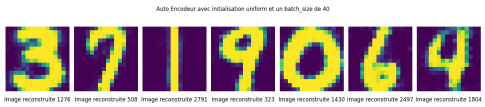
Décodeur

Pour ce faire, nous explorons différentes initialisations pour la matrice W et variations du batch afin de déterminer les meilleurs hyperparamètres.

Images originales

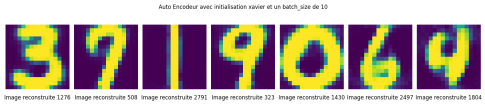


Initialisation **uniform**, batch de **40**

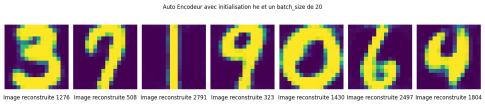


Initialisation **uniform**, batch de **10**

Initialisation **Xavier**, batch de **10**



Initialisation **He**, batch de **20**

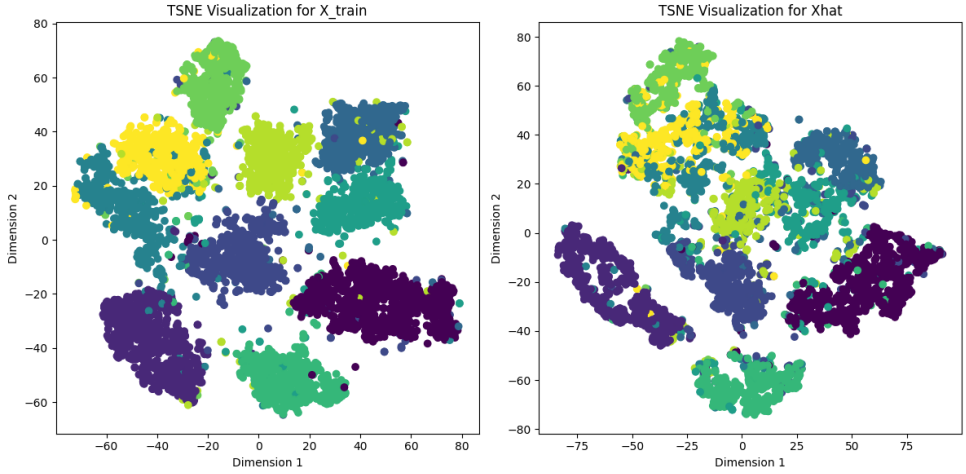


Après observation, nous avons décidé d'opter pour l'initialisation uniform avec un batch de 10 pour notre modèle, tel que :

$$W \sim \mathcal{U}(-1, 1) * 0.4$$

5.1.2 Visualisations des résultats

Pour mieux comprendre la distribution des données et détecter des structures cachées, nous avons appliqué la PCA et le t-SNE pour visualiser les données originales et reconstruites en dimension réduite.



Les résultats obtenus à travers les visualisations montrent que notre autoencodeur est capable de capturer les caractéristiques importantes des données d'origine et de les reconstruire avec précision. Les deux méthodes de visualisation montrent que les données d'origine et les données reconstruites ont une forme et une distribution similaires, ce qui est un indicateur de la qualité de la reconstruction effectuée par l'autoencodeur.

5.1.3 Étude de classification multiclasse des données reconstruites

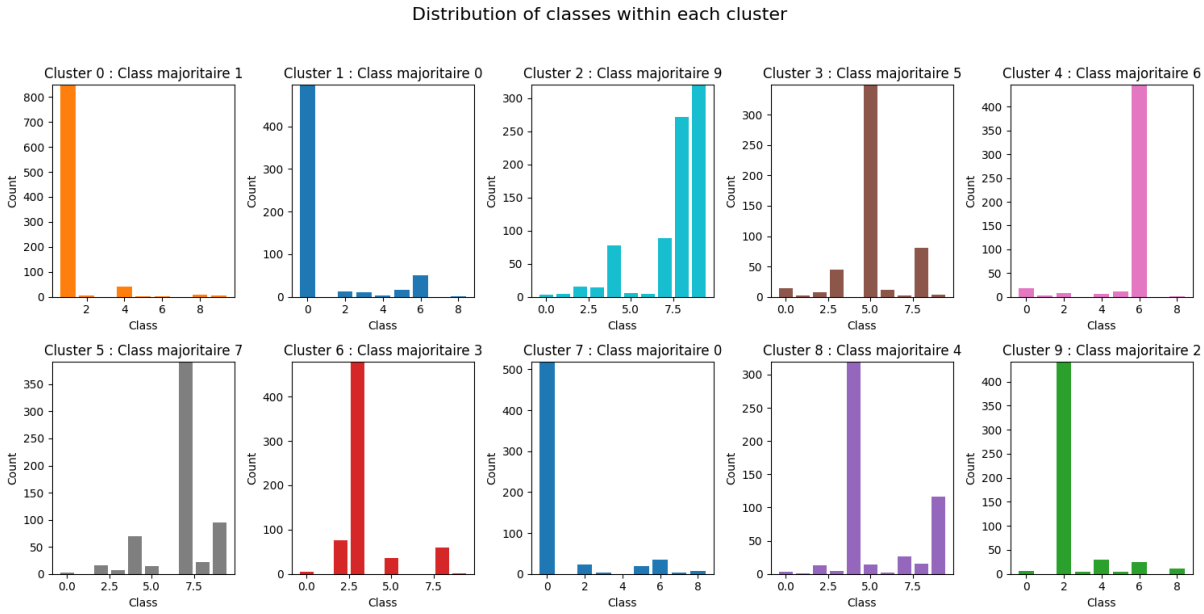
Nous avons évalué l'efficacité de notre encodage en entraînant un modèle multiclasse sur les données originales avec leurs étiquettes, utilisant le même réseau de neurones que celui utilisé pour la classification multiclasse sur les données USPS. Le modèle initial avait obtenu un taux de réussite de 93%. Nous avons ensuite testé ce modèle sur les données reconstruites à partir de notre encodage, en utilisant les mêmes étiquettes pour évaluer l'accuracy. Nous avons obtenu un score de 83%, montrant que notre encodage est toujours capable de fournir des résultats satisfaisants malgré une légère baisse de performance par rapport au modèle initial.

Cependant, il est crucial d'examiner en détail les raisons de cette légère variation de performance. Pour y parvenir, nous avons décidé d'étudier le clustering dans l'espace latent obtenu.

5.1.4 Clustering dans l'espace latent

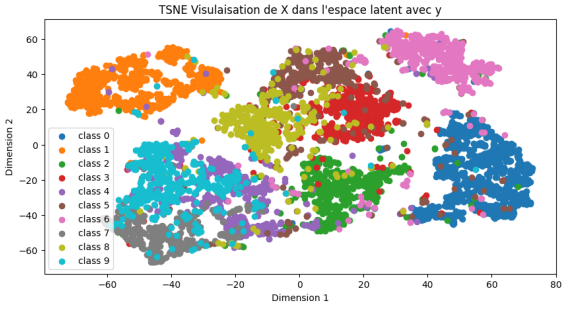
Nous allons utiliser les représentations significatives et réduites des données dans l'espace latent pour explorer les motifs et les relations entre les instances. Nous allons appliquer une technique de clustering dans l'espace latent pour mettre en évidence les différents groupes d'images formés et ainsi analyser l'efficacité de l'espace latent à mieux représenter les données.

En appliquant un algorithme de k-means avec 10 clusters, nous avons obtenu un graphique illustrant la répartition des classes et l'homogénéité intra-cluster.

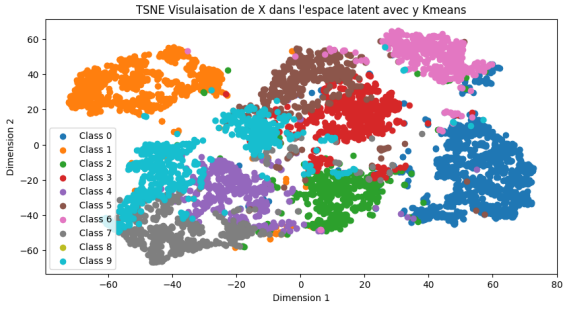


Il est possible de remarquer une nette séparation entre la plupart des classes dans l'espace latent, à l'exception des classes 8 et 9 qui ne sont pas clairement distinctes. Cette disparité est surtout visible dans le 3ème cluster.

Pour mieux mettre en évidence cette disparité, une visualisation t-SNE des images originales a été effectuée et les classes ont été colorées en utilisant à la fois les étiquettes réelles (y) et les étiquettes obtenues par k-means tel que pour chaque exemple, la classe majoritaire de son cluster a été utilisée comme étiquette.



Coloration des classes en utilisant y

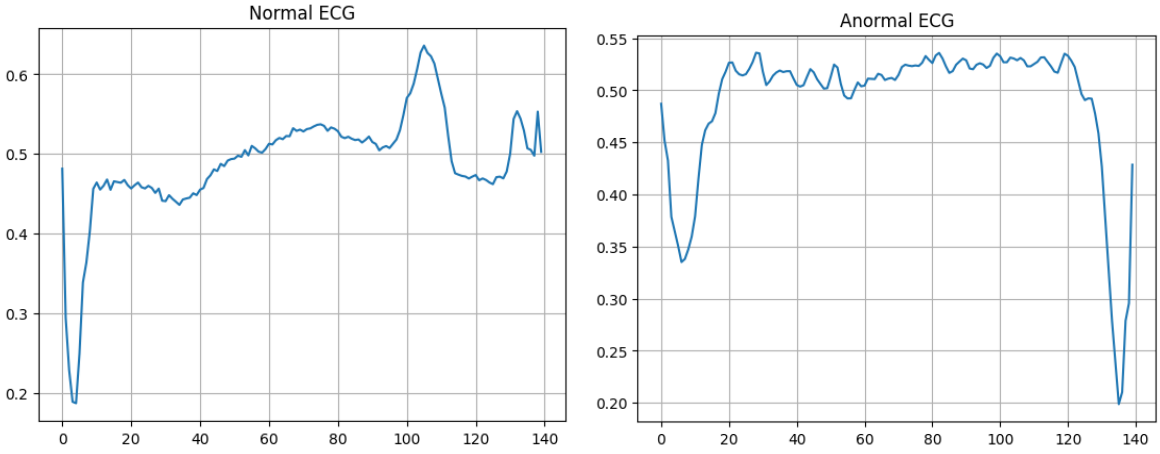


Coloration des classes en utilisant y kmeans

5.2 Détections d'anomalies

5.2.1 Descriptions des données

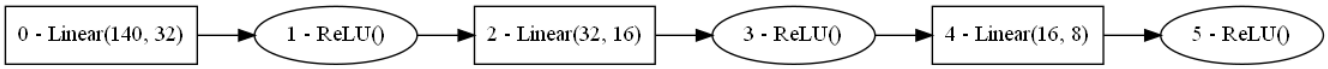
Le jeu de données utilisé dans cette étude est constitué de deux collections de signaux cardiaques provenant de la base de données PTB Diagnostic ECG. Ces signaux représentent des enregistrements d'électrocardiogrammes (ECG) qui décrivent différents types de battements de cœur, incluant des cas normaux ainsi que des cas présentant des arythmies et des infarctus du myocarde. Les données ont été prétraitées et segmentées pour isoler chaque battement cardiaque.



5.2.2 Entraînement sur les données de catégorie Normal

Dans le but de détecter les anomalies, nous avons pris la décision de diviser notre ensemble de données en deux catégories : Normal et Anormal.

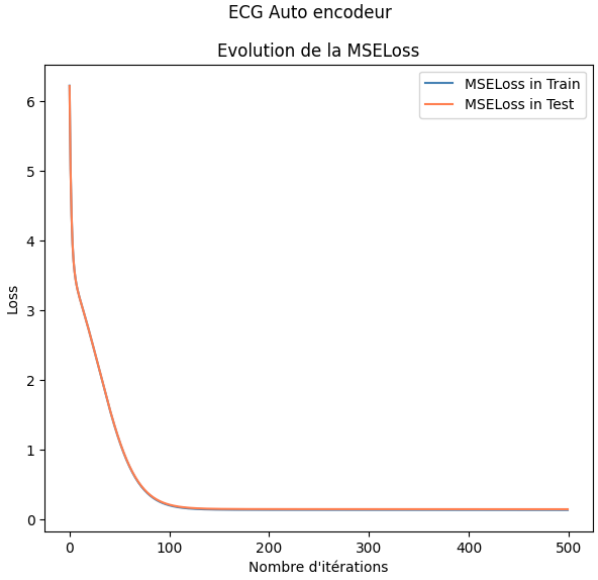
Par la suite, nous avons entraîné le réseau de neurones suivant en utilisant **exclusivement** les données de la catégorie **Normal**.



Encodeur

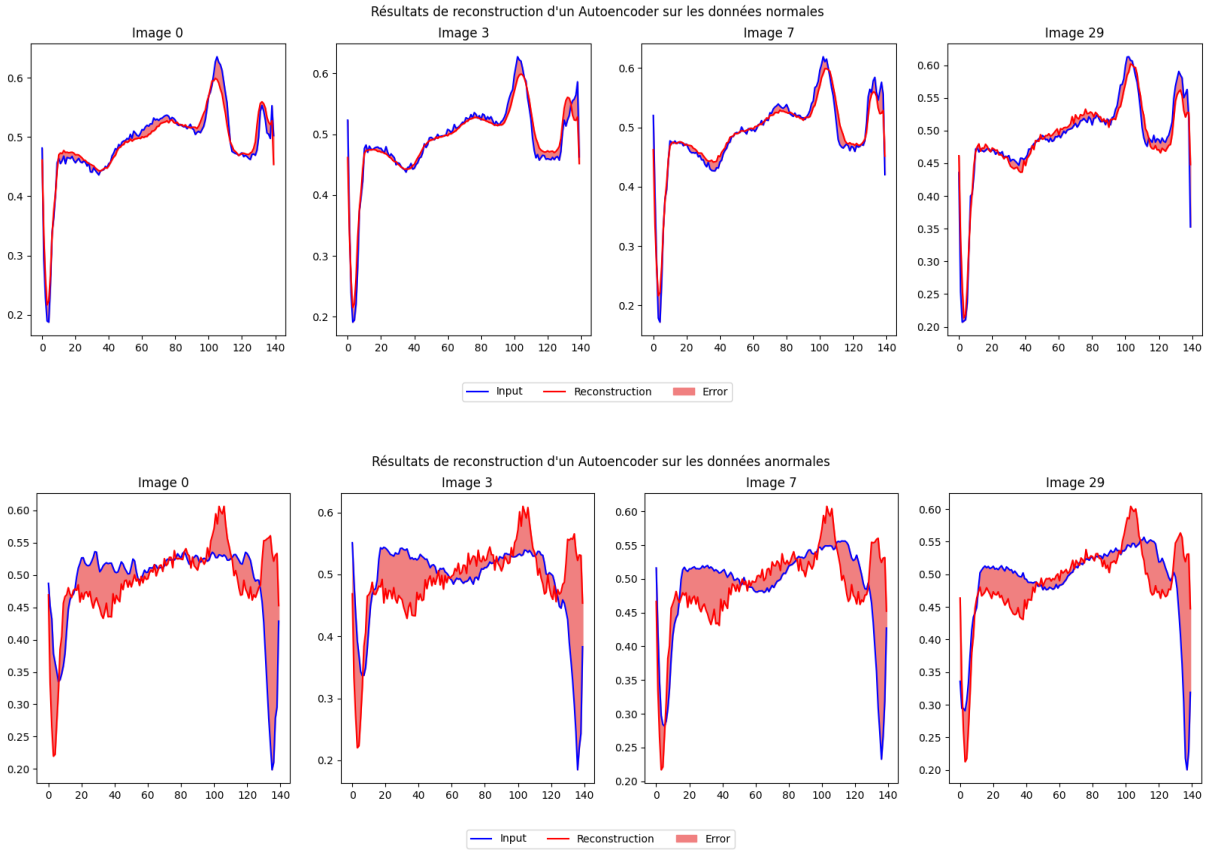
Décodeur

L'objectif de ce modèle est d'apprendre à reconnaître les motifs et les caractéristiques des données normales afin de pouvoir distinguer les anomalies lorsqu'elles se produisent. En excluant les données anormales de l'ensemble d'entraînement, le modèle se concentre uniquement sur les schémas normaux et peut mieux les capturer.



5.2.3 Prédiction des Anomalie

Une fois le modèle entraîné sur les données normales, nous serons en mesure de l'utiliser pour prédire si de nouvelles données sont normales ou anormales en se basant sur les motifs appris. Dans ce processus, nous utiliserons la valeur de la MSE pour détecter les anomalies.



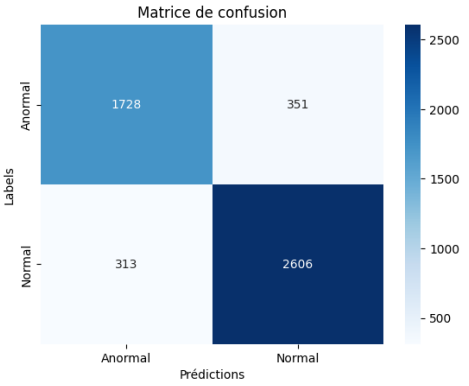
Nous observons que l'erreur de reconstitution augmente pour les données anormales. Afin de détecter les anomalies, nous décidons de seuiller la MSE sur les données d'origine

normal et les données reconstruites. Nous fixons le seuil en fonction de cette observation :

$$t = \mu_{mse(XNormal,\ \hat{XNormal})} + 2 \times \sigma_{mse(XNormal,\ \hat{XNormal})}$$

Tout signal dont la valeur de la MSE est inférieure au seuil t sera considéré comme normal, sinon il sera considéré comme anormal. Par conséquent, nous effectuerons un seuillage sur les données normales et anormales, puis construirons un vecteur de prédictions. Nous évaluerons ensuite ces prédictions en les comparant aux vraies étiquettes pour obtenir les résultats suivants :

class	precision	recall	f1-score	support
Anormal	0.89	0.82	0.87	2079
Normal	0.99	0.96	0.98	2919
accuracy	0.94			4998



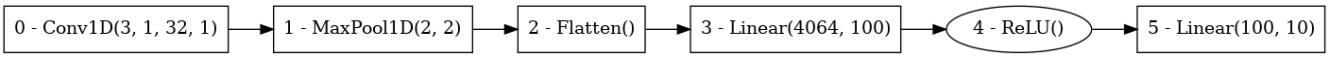
Matrice de confusion

6 Convolution

6.1 Prédiction d'un chiffre

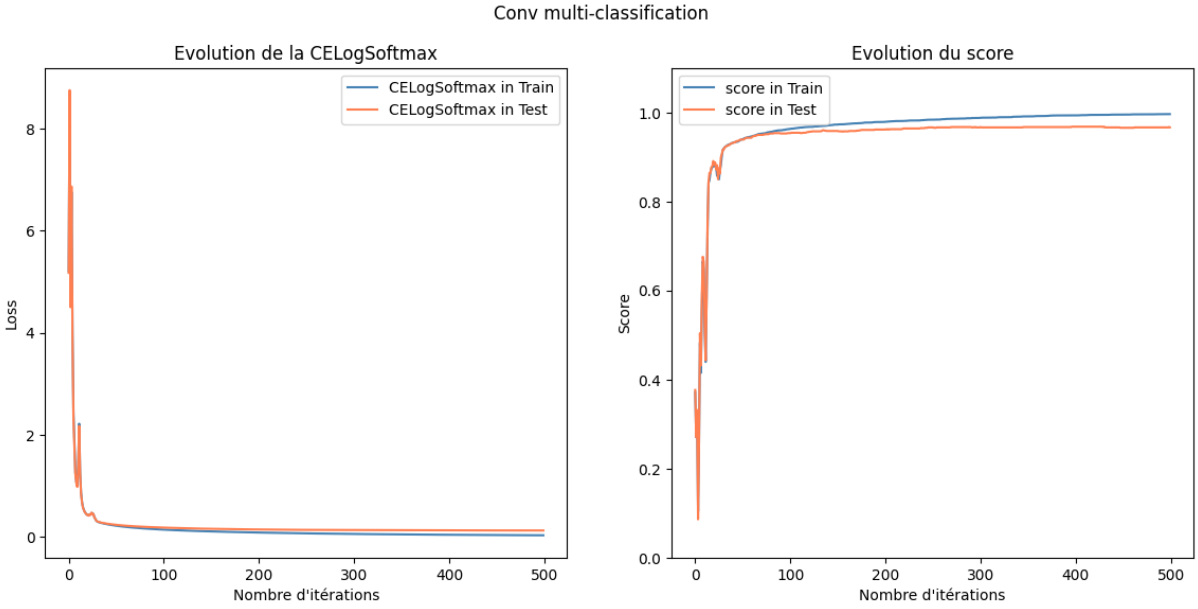
Nous utilisons une architecture spécifique de réseau de neurones pour entraîner notre modèle en utilisant les bases de données de chiffres fournies par USPS. Cette architecture a été choisie pour sa capacité à reconnaître efficacement les motifs visuels et à améliorer les performances de notre modèle.

L'architecture choisie pour notre réseau de neurones est définie comme suit :



De plus, nous avons choisi d'utiliser l'initialisation de **He**, définie ainsi :

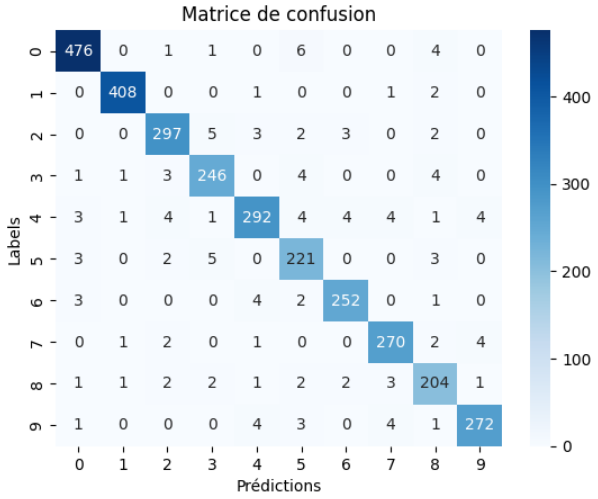
$$W \sim \mathcal{N}(0, \sqrt{\frac{2}{k\_size}})$$



Notre modèle affiche un score de 94%, ce qui est excellent. Cependant, nous avons constaté un inconvénient majeur : le temps de calcul. En effet, nous avons observé une augmentation significative du temps de calcul pour cette architecture. De plus, nous avons remarqué que notre algorithme commence à converger à partir de 150 itérations.

Voici les différents scores obtenus ainsi que la matrice de confusion associée aux résultats.

class	precision	recall	f1-score	support
0	0.97	0.96	0.97	492
1	0.99	0.98	0.99	414
2	0.95	0.92	0.93	318
3	0.92	0.93	0.92	256
4	0.93	0.91	0.92	312
5	0.89	0.94	0.91	231
6	0.93	0.95	0.94	257
7	0.96	0.95	0.95	284
8	0.89	0.91	0.90	219
9	0.95	0.94	0.95	286
accuracy	0.94			3069



Matrice de confusion

7 Conclusion

En conclusion, notre étude a démontré l'efficacité des différentes architectures de réseaux de neurones pour résoudre des problèmes de régression, de classification, de multiclasse, d'auto-encodeur et de convolution. Chaque architecture présente ses propres avantages et peut être adaptée en fonction des besoins spécifiques. Les réseaux de neurones offrent des possibilités prometteuses pour la résolution de problèmes complexes et l'amélioration des performances de prédiction. Cependant, il est important de prendre en compte le temps de calcul nécessaire, afin de trouver un équilibre entre la performance du modèle et l'efficacité du processus.