Projet de Machine Learning
Réseau de neurones : DIY

KHALFAT Célina PIRIOU Victor

24 mai 2021

Table des matières

1 Introduction

L'objectif de ce projet est d'implémenter un réseau de neurones. L'implémentation est inspirée des anciennes versions de pytorch (en Lua, avant l'autograd que vous verrez l'année prochaine) et des implémentations analogues qui permettent d'avoir des réseaux génériques très modulaires. Les fonctions de coût implémentées dans ce projet sont le coût aux moindres carrés, la cross-entropie, la cross-entropie appliquée au log d'un softmax et la cross-entropie binaire. Concernant les différents modules, qui représentent les couches du réseau, nous avons implémenter le module linéaire ainsi que les modules de transformation tangente hyperbolique, sigmoide et softmax. Nous avons implémenté la partie obligatoire du projet et nous allons vous présenter les différentes expérimentations faites et résultats obtenus.

2 Mon premier est ... linéaire !

2.1 Régression Linéaire

Pour tester notre premier réseau de neurones à une couche linéaire, nous allons nous intéresser au problème de la régression linéaire. Nos données seront des points générés aléatoirement à partir d'un paramètre a, qui correspond à la pente de la droite, avec un bruit σ. La fonction de coût utilisée sera la MSE.

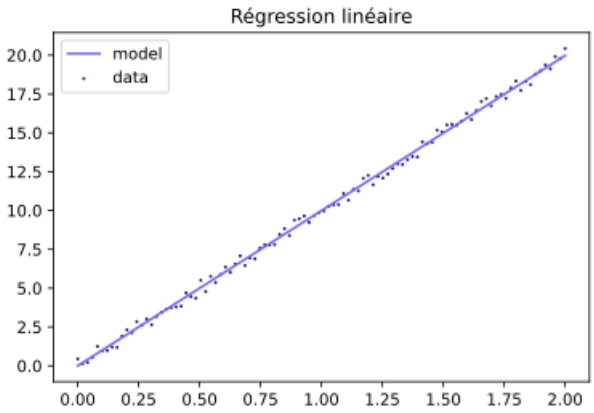


FIGURE (1) Régression linéaire pour $a = 10$ et $\sigma = 0.5$

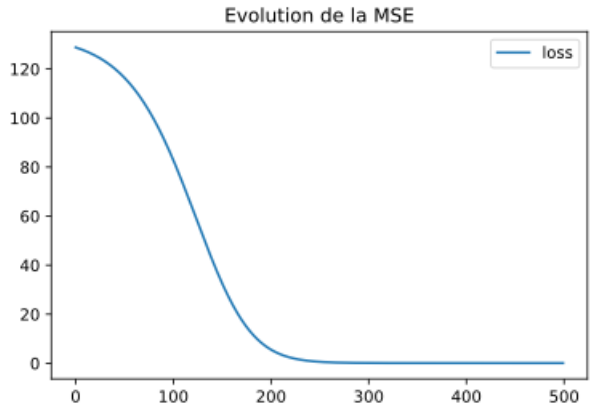


FIGURE (2) Évolution de la MSE associée à la figure 1

Au bout de 500 itérations la MSE vaut 0.08. Le coefficient de corrélation linéaire prédit est 9.988369678717126. Nous observons par ailleurs que la MSE converge aux alentours de 200 itérations.
Nous allons maintenant essayer d'ajouter plus de bruit : on prend sigma = 6.
Nous obtenons au bout de 500 itérations une MSE égale à 10.31 et le coefficient de corrélation linéaire prédit vaut 10.09669473976822. Bien que la MSE soit plus élevée pour sigma égal à 6, le coefficient de corrélation linéaire a bien été prédit. Cela s'explique simplement par le fait que si les données sont plus éloignés de la droite, la moyenne de la somme des distances de chaque point à la droite le sera aussi.

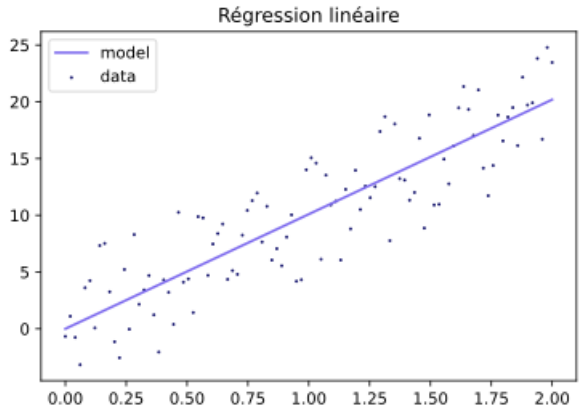


FIGURE (3) Régression linéaire pour $a = 10$ et $\sigma = 6$

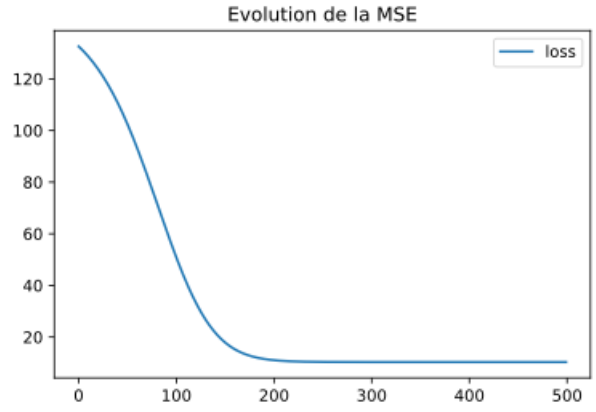


FIGURE (4) Évolution de la MSE associée à la figure 3

2.2 Classification

Nous allons maintenant tester notre réseau de neurones linéaire sur des données générées par deux gaussiennes avec très peu de bruit.

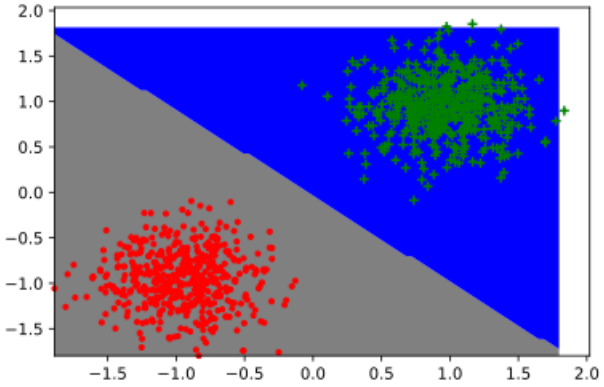


FIGURE (5) Frontière générée pour des données suivants 2 gaussiennes et un bruit égal à 0.1

Les données sont linéairement séparable, ainsi le taux de bonne classification obtenu est 100%.
Le bruit étant plus présent dans les données, les données ne sont plus linéairement séparables. Par ailleurs, la droite trouvée permet tout de même de maximiser le taux de bonne classification, qui dans cet exemple, est égale à 0.915.

3 Mon second est ... non-linéaire !

Comme pour la classification linéaire, nous allons utiliser les données de gen_arti mais sur des données générées par quatre gaussiennes.

3.1 Variation du nombre de neurones

Nous allons nous intéresser ici au nombre de neurones nécessaires pour bien classer nos données et au nombre d'itérations. Les résultats suivants ont été trouvés avec une descente de gradient par batch et un pas de 0.001.

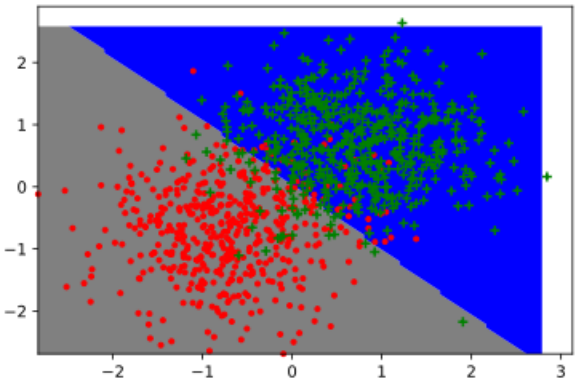


FIGURE (6) Frontière générée pour des données suivant 2 gaussiennes et un bruit égal à 1

Pour 1 neurone

Nombre d'itérations	Taux de bonne classification
500	0.741
700	0.737
900	0.738
1100	0.741
1300	0.741
1500	0.741

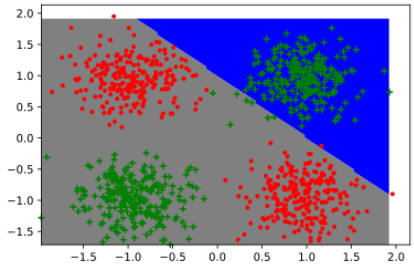


FIGURE (7) Itérations = 500

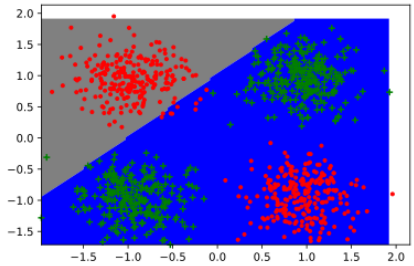


FIGURE (8) Itérations = 900

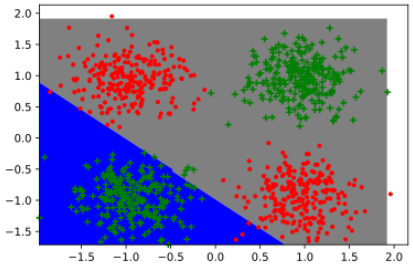


FIGURE (9) Itérations = 1500

Avec un neurone, les données sont séparées à l'aide d'une droite. Or nos données ne sont pas séparables linéairement. C'est pourquoi les résultats ne sont pas très bons. Nous allons essayer d'augmenter le nombre de neurones.

Pour 2 neurones

Nombre d'itérations	Taux de bonne classification
500	0.66
700	0.684
900	0.644
1100	0.947
1500	0.952
1700	0.701

Avec deux neurones on peut commencer à observer des frontières non linéaire. Néanmoins, les résultats sont instables.

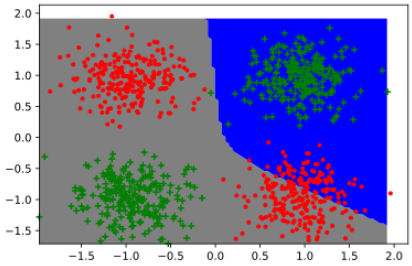


FIGURE (10) Itérations = 500

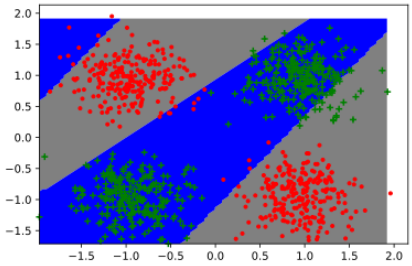


FIGURE (11) Itérations = 1100

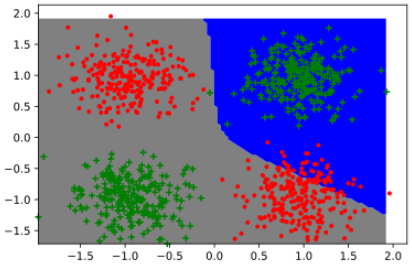


FIGURE (12) Itérations = 1700

Pour 3 neurones

Nombre d'itérations	Taux de bonne classification
500	0.997
700	0.995
900	0.997
1100	0.998
1500	0.995
1900	0.998

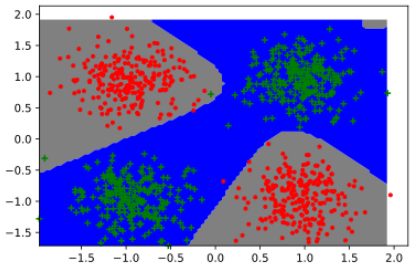


FIGURE (13) Itérations = 500

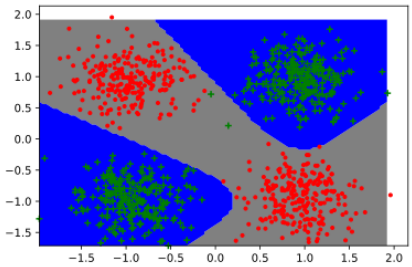


FIGURE (14) Itérations = 1100

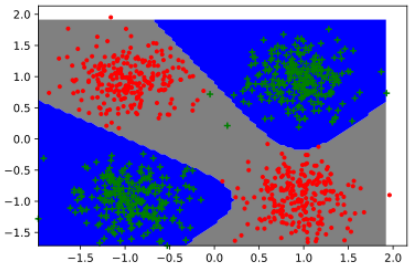


FIGURE (15) Itérations = 1900

Tout d'abord, avec trois neurones nous pouvons voir que les résultats sont nettement meilleurs. Globalement, plus le nombre de neurones est grand plus la frontière est complexe puisqu'elle résulte d'une combinaisons de fonctions non-linéaires, grâce aux différentes fonctions d'activations.

3.2 Variation du pas du gradient et de la taille du batch

Nous allons maintenant essayer de faire varier le pas du gradient ainsi que la taille du batch et voir comment les résultats évoluent. Nous allons fixer le nombre de neurones à 3.

Pour un pas de 0.0005

Nombre d'exemples par batch	Nombre d'itérations	Taux de bonne classification
1	10	0.492
1	40	0.536
1	120	0.5
200	10	0.451
200	40	0.739
200	120	0.051
500	10	0.616
500	40	0.744
500	120	0.994
800	10	0.658
800	40	0.723
800	120	0.995
1000	10	0.343
1000	40	0.74
1000	120	0.996

Pour un pas de 0.001

Nombre d'exemples par batch	Nombre d'itérations	Taux de bonne classification
1	10	0.703
1	40	0.627
1	120	0.604
200	10	0.727
200	40	0.626
200	120	0.975
500	10	0.509
500	40	0.948
500	120	0.699
800	10	0.723
800	40	0.737
800	120	0.985
1000	10	0.559
1000	40	0.982
1000	120	0.994

Pour un pas de 0.01

Nombre d'exemples par batch	Nombre d'itérations	Taux de bonne classification
1	10	0.503
1	40	0.39
1	120	0.53
200	10	0.72
200	40	0.994
200	120	0.993
500	10	0.608
500	40	0.992
500	120	0.997
800	10	0.992
800	40	0.994
800	120	0.998
1000	10	0.964
1000	40	0.738
1000	120	0.993

Pour un pas de 0.1

Nombre d'exemples par batch	Nombre d'itérations	Taux de bonne classification
1	10	0.13
1	40	0.711
1	120	0.742
200	10	0.942
200	40	0.993
200	120	0.994
500	10	0.499
500	40	0.733
500	120	0.742
800	10	0.743
800	40	0.742
800	120	0.739
1000	10	0.745
1000	40	0.739
1000	120	0.742

Observations

Toutes données nous permettent dans un premier temps de voir que le nombre d'itérations doit être plus important lorsqu'on fait une descente de gradient stochastique, que par batch. Le pas 0.0005 est trop petit : la descente de gradient est trop lente. Contrairement au pas 0.1 qui lui est trop grand : les résultats ne sont pas très bon, on peut penser que l'on est souvent amené à diverger. Les pas supérieurs à 0.1 seront ainsi également mauvais. Bien que les résultats pour un pas de 0.001 étaient bons en mini-batch et batch dès une centaine d'itérations, l'algorithme a l'air de converger plus rapidement avec un pas 0.01. On pourrait continuer d'explorer dans ces zones pour trouver le meilleur pas. On ne voit pas tellement de différence entre le mini-batch et le batch, cela est peut être dû au fait que les données ne sont pas bruitées.

4 Mon troisième est un encapsulage

Le travail demandé pour cette partie a été réalisé avec succès. Cette partie a permis d'avoir un code plus factorisé.

5 Mon quatrième est multi-classe

Nous allons maintenant nous intéresser aux réseaux de neurones multi-classe : le réseau a autant de sortie que de classe et chaque sortie contient la probabilité d'appartenir à cette classe. Pour cette partie, le réseau a été entraîné avec un pas de gradient de 0.01 et 100 itérations. La classification multi-classe a été testée sur le jeu de données USPS. L'architecture utilisée pour le calcul de ces taux est :

Linear(intput + 1,nombre de neurones) $\rightarrow$ **TanH()** $\rightarrow$ **Linear(nombre de neurones,output)** $\rightarrow$ **Sigmoide()**

La figure 24 montre que, sur ce jeu de données, le taux de bonne classification est souvent très légèrement meilleur lorsqu'on utilise comme coût l'erreur quadratique moyenne plutôt que la cross entropie combinée au Softmax.

Dans la pratique on ne s'intéresse pas uniquement au taux de bonne classification. Il existe d'autres mesures d'évaluation comme le rappel et la précision.

Les graphiques ci-dessous montrent qu'il n'y a pas de différence significative entre la MSE et la cross entropie. Cela est peut-être dû au fait que les données utilisées sont trop simples.

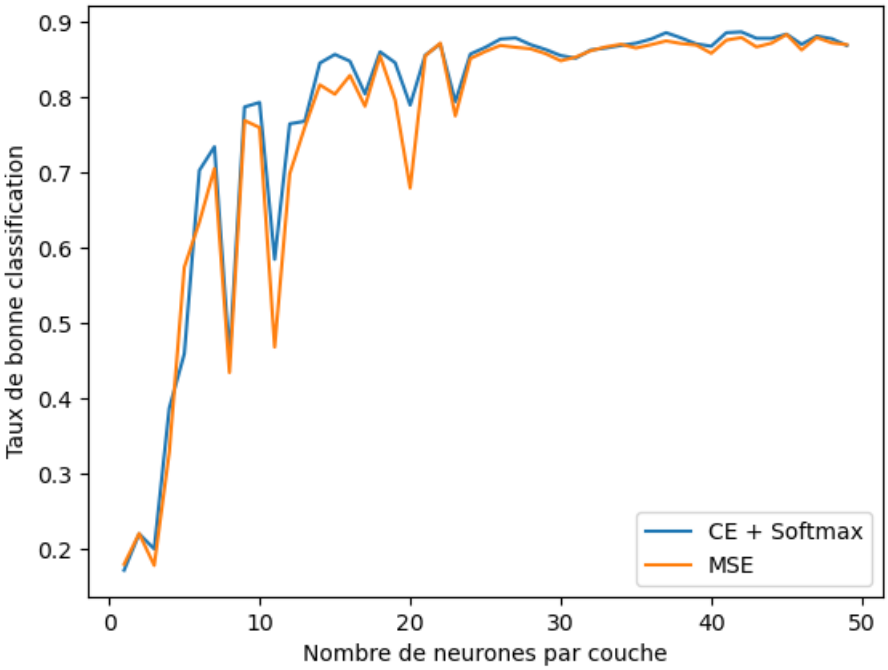
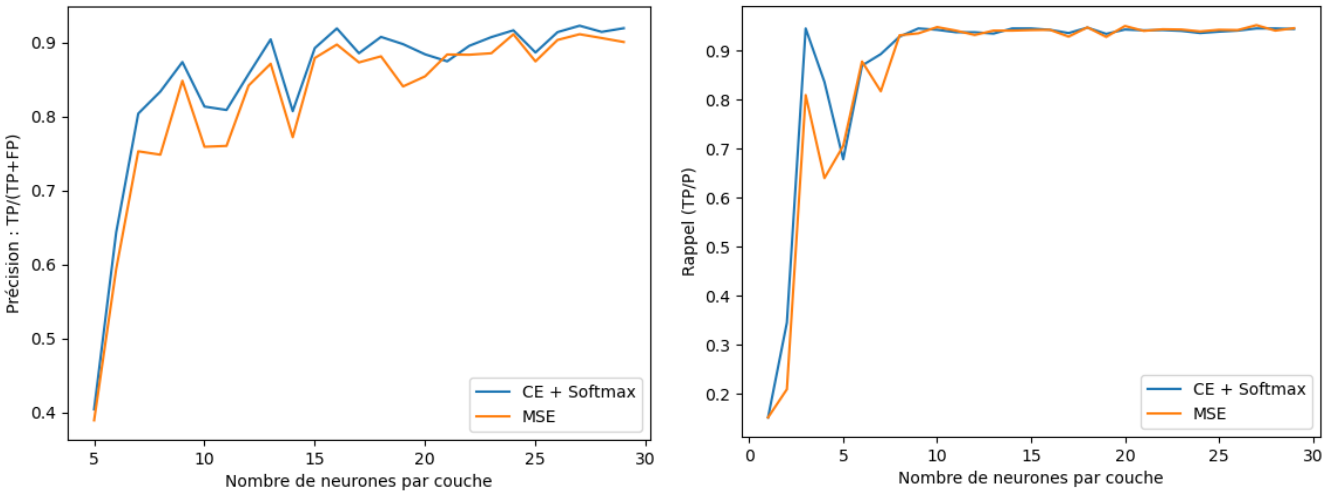
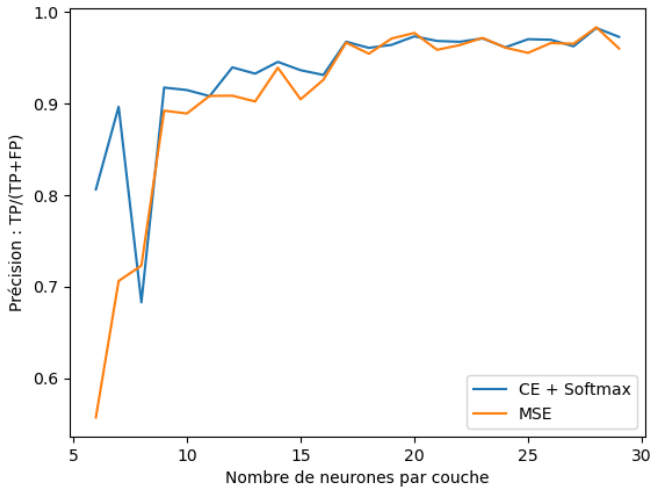


FIGURE (16) Comparaison des taux de bonne classification obtenus avec la MSE et la cross entropie, en fonction du nombre de neurones

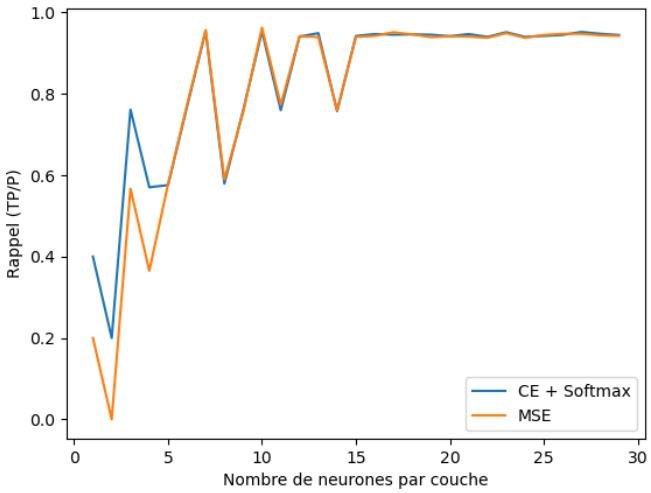


Précision Rappel

FIGURE (17) Précision et rappel pour la classe des zéros

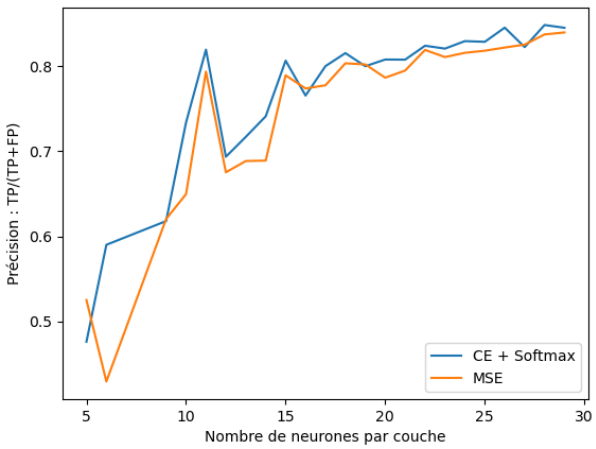


Précision

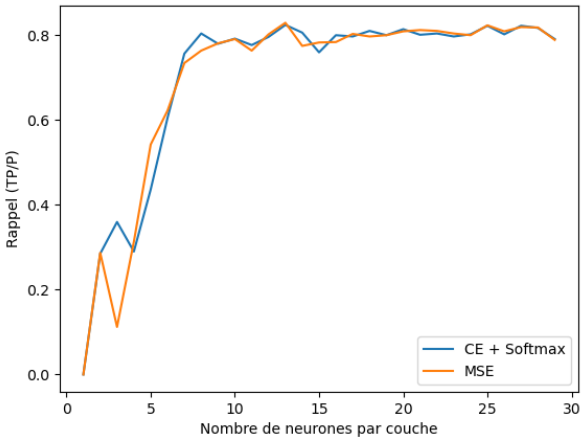


Rappel

FIGURE (18) Précision et rappel pour la classe des uns



Précision



Rappel

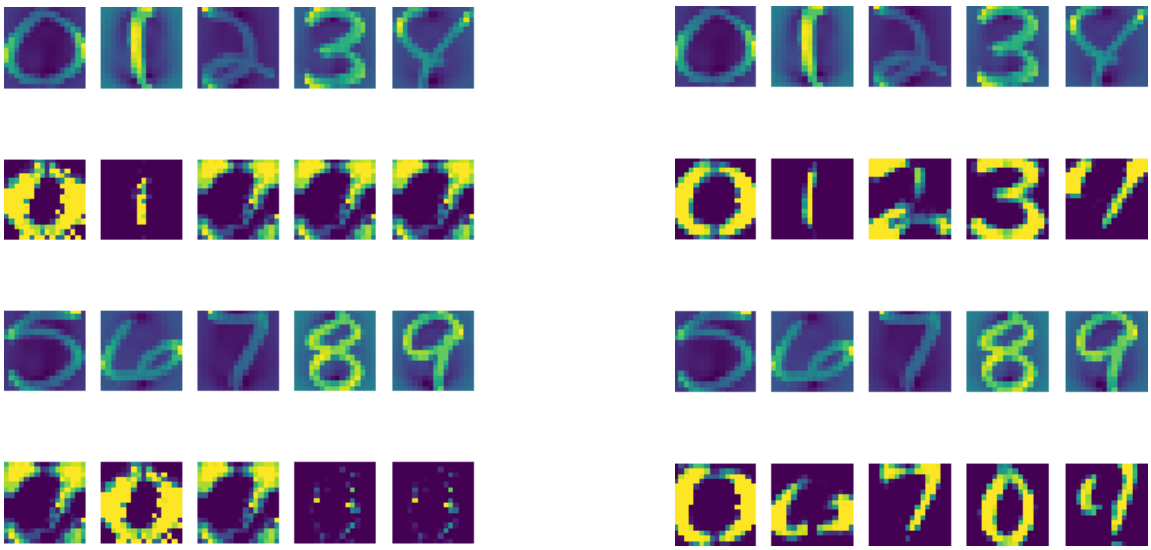
FIGURE (19) Précision et rappel pour la classe des deux

6 Mon cinquième se compresse

On s'est donné comme objectif d'étudier l'influence de la taille de l'espace latent sur le clustering qu'il induit et sur la qualité du codage.

6.1 Comparaison entre la moyenne des images originales et la moyenne des images reconstruites

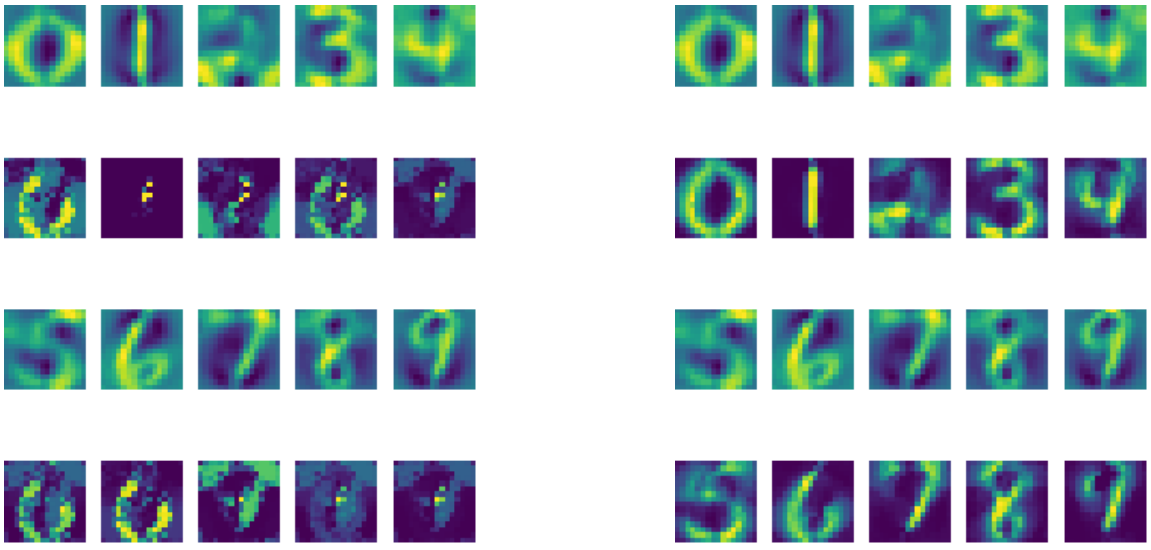
Pour ce faire, on a commencé par comparer, sur quelques images, les reconstructions obtenues avec des espaces latents de tailles différentes.



Taille de l'espace latent : 3 Taille de l'espace latent : 10

FIGURE (20) Les images originales sont sur les lignes 1 et 3. Les images reconstruites sont sur les lignes 2 et 4.

Puis, pour chaque classe, on a comparé l'image moyenne des images originales à l'image moyenne des images reconstruites.



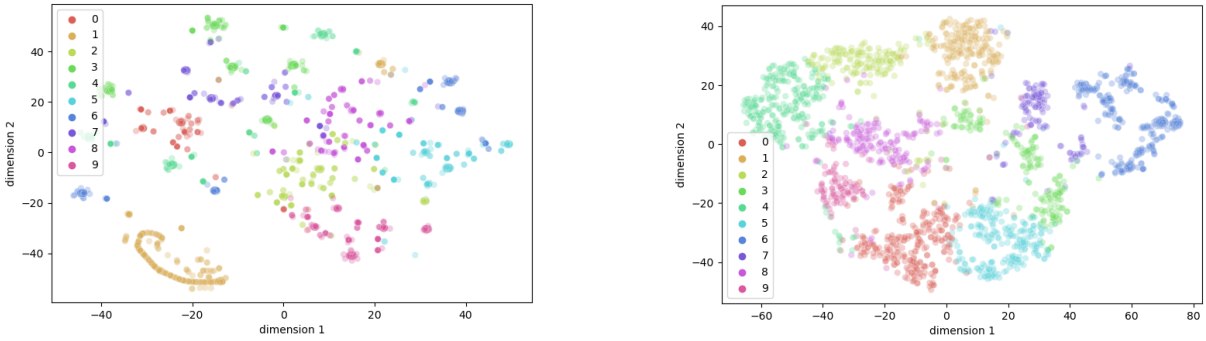
Taille de l'espace latent : 3 Taille de l'espace latent : 10

FIGURE (21) Les moyennes des images originales sont sur les lignes 1 et 3. Les moyennes des images reconstruites sont sur les lignes 2 et 4.

La figure 21 montre qu'avec une taille d'espace latent égale à 3 on ne parvient presque jamais à reconstuire les

images de façon satisfaisante. Une taille fixée à 10 permet de reconstruire une bonne partie des images de test. Par ailleurs, on peut noter que l'auto-encodeur a tendance à simplifier le fond des images mais cela ne change pas leur sémantique.

6.2 Visualisation du clustering induit dans l'espace latent avec T-SNE et K-means



Taille de l'espace latent : 10 Taille de l'espace latent : 90

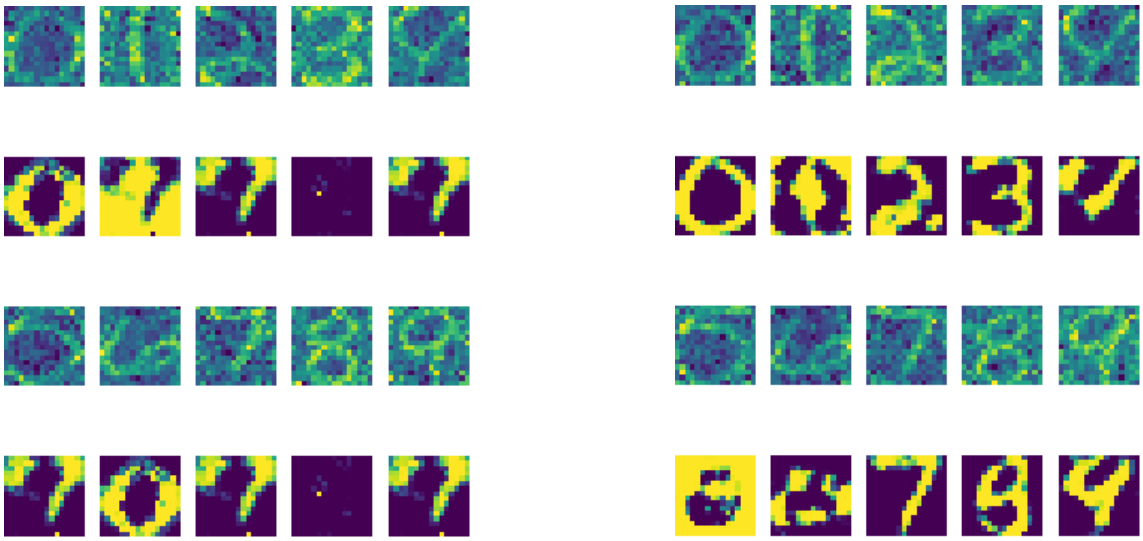
FIGURE (22) Projections 2D (avec t-sne) des représentations compressées avec différentes tailles d'espace latent.

On a effectué un K-means (avec K = 10) sur l'ensemble des vecteurs obtenus à la sortie de l'encodeur et on les a projeté en 2D grâce à t-sne.

Avec une taille d'espace latent fixée à 90, t-sne offre une représentation satisfaisante puisque les vecteurs de chaque classe apparaissent regroupés entre eux. En réduisant cette taille à 10, on constate que les vecteurs d'une même classe sont dispersés en petits clusters situés à des zones différentes. Cette diminution des distances entre les vecteurs des petits clusters est le résultat de la compression : les informations les moins discriminantes ne sont pas conservées donc il y a moins de variabilité entre les données compressées.

6.3 Etude des performances en débruitage

Cette réduction de dimension permet de multiples applications. Par exemple, elle peut être utilisée pour débruiter des images.



Taille de l'espace latent : 3 Taille de l'espace latent : 180

FIGURE (23) Les images bruitées sont celles de la figure 20. Elles sont sur les lignes 1 et 3. Les images que l'auto-encodeur a tenté de débruité sont sur les lignes 2 et 4.

On peut observer l'influence de la taille de l'espace latent sur les performance de débruitage. Pour ce faire, on bruite l'ensemble des images de test et on calcule l'entropie croisée binaire (BCE) moyenne entre les images originales et les images reconstruites à partir des images bruités.

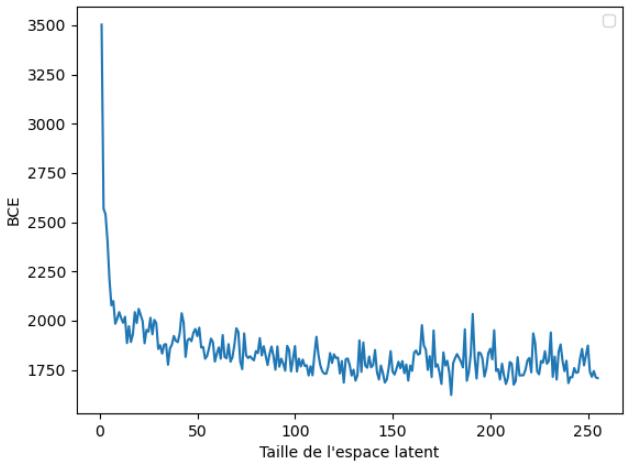


FIGURE (24) Evolution de la BCE en fonction de la taille de l'espace latent. Sur chaque pixel p_i, un bruit $\epsilon_i \sim \mathcal{N}(0, 0.6)$ a été ajouté.

La taille optimale (180) de l'espace latent est celle qui minimise ce risque empirique. Idéalement il faudrait estimer chaque point en faisant plusieurs descente de gradient car la BCE obtenue dépend en partie du point de départ de la descente.

6.4 Pureté des clusters obtenus avec un K-means sur l'ensemble des reconstructions

Pour évaluer de manière rigoureuse les performances de l'auto-encodeur, on a calculé la pureté moyenne des clusters obtenus avec un K-means (K = 10) sur l'ensembles des images reconstuites. Pour calculer cette pureté moyenne, chaque cluster est assigné à la classe la plus fréquente du cluster, puis la précision de cette affectation est mesurée en comptant le nombre de reconstructions correctement assignées, et finalement on divise par le nombre d'exemples.

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

où $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ est l'ensemble des clusters et $C = \{c_1, c_2, \ldots, c_J\}$ l'ensemble des classes, ω_K l'ensemble des images assignées au cluster ω_K et c_j l'ensemble des images reconstruites dont l'image originale est de classe c_j.

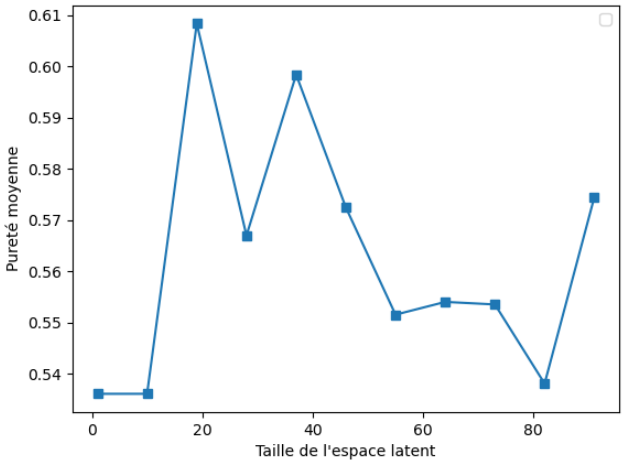


FIGURE (25) Pureté moyenne en fonction de la taille de l'espace latent

La figure 25 montre qu'une augmentation de la taille de l'espace latent ne se traduit pas forcément pas une augmentation de la pureté des clusters.