Réseau de neurones

SIHAMDI Mostefa, Derras Kahlil

UE ML 2020-2021, Encadrant : Nicolas Baskiotis

M1 DAC

1. Mon premier est . . . linéaire ! :

Avec un module linéaire et une fonction mse de loss,on utilise une fonction linaire bruité (bruit gaussian) ax+b et l'objective est trouver la droit associer à la fonction linéaire. Avec a=15 et b=0,on prend un nombre d'itération qui égale à 600 on trouve la droit suivant :

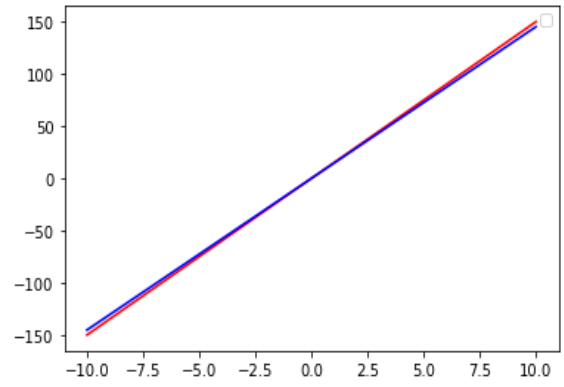


Figure 2 Régression linéaire d'une droite

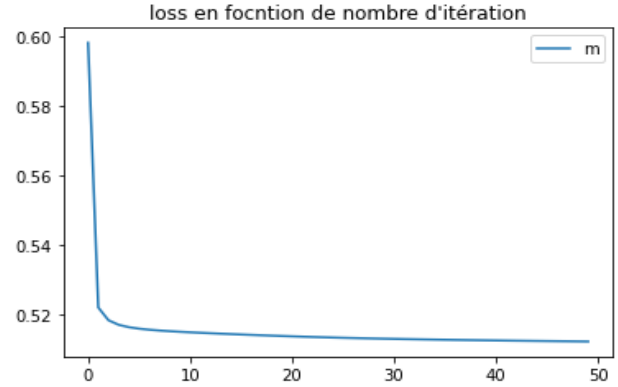


Figure 1 la moyenne des loss en fonction de nombre d'itération

La droit rouge représente la droit a prédira droit bleu représente la droit trouver qui est de paramètre a=14.52 et b=0,on remarque que la droit trouver est très approximative à la première fonction linéaire. Aussi on remarque une convergence rapide de la loss

2. Mon second est . . . non-linéaire !

Dans les tests de cette partie on a utilisé le Générateur de données de la fonction gen_arti défini pour les TPs précédent et qui contient trois type de représentation de données .et on utilise 2 type de réseau de neurone, le but de cette partie est de faire la classification, le premier réseau utiliser est

$$\text{Linear} \rightarrow \text{TanH} \rightarrow \text{Linear} \rightarrow \text{Tanh} \rightarrow \text{Linear} \rightarrow \text{Sigmoid} \ldots\ldots\ldots 1$$

Et la 2éme réseau de neurone est représenter par :

$$\text{Linear} \rightarrow \text{TanH} \rightarrow \text{Linear} \rightarrow \text{Tanh} \rightarrow \text{Linear} \rightarrow \text{Tanh} \rightarrow \text{Linear} \rightarrow \text{Sigmoid} \ldots\ldots\ldots 2$$

Et on utilise MSE comme fonction de cout.

a. Mélanges de deux gaussian

avec data_type=0 dans la fonction gen_arti on test les deux réseau de neurone précèdent.

2.2.1 Avec le premier Réseau

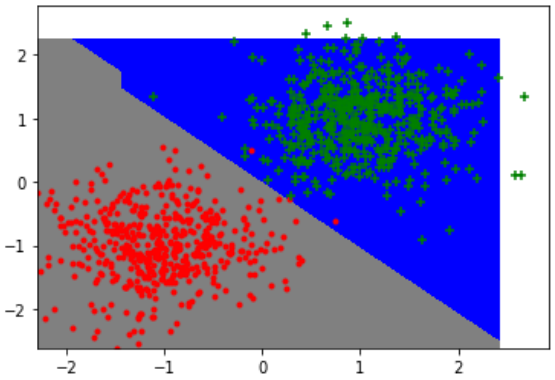


Figure 3 Frontière de décision des données linaires

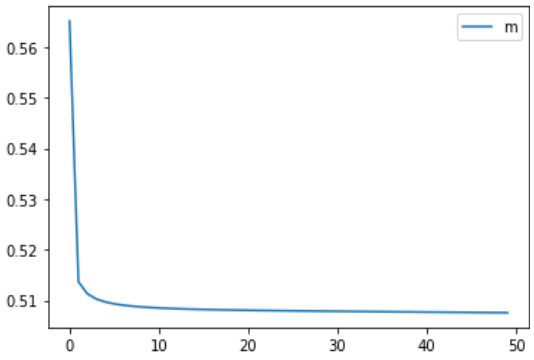


Figure 4 Moyenne de loss en fonction de nombre d'itération

2.2.2 Avec la deuxième Réseau : On trouve les résultats suivant :

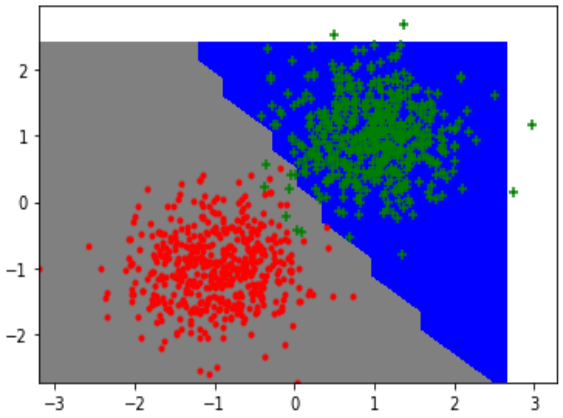
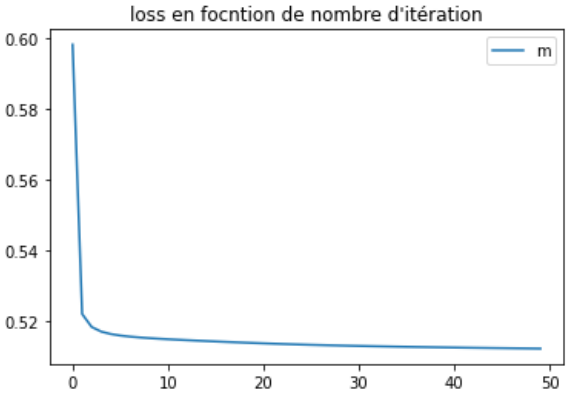


Figure 5 Frontière de décision des données linaires



On remarque que pour les deux réseaux, le system à bien réussir à définir les frontières des données de test, pour le premier réseau on trouve un taux de bonne classification égale a 0.98,pour la deuxième réseau on trouve un score qui égale à 0.91,donc en augmentant le réseau avec une coche linaire ça permet de démunie le réseau, aussi on remarque une convergence rapide de la fonction de loss (ont utilisé le MSE loss)

b. mélange de quatre gaussiennes (XOR) :

2.2.1 Avec le premier réseau

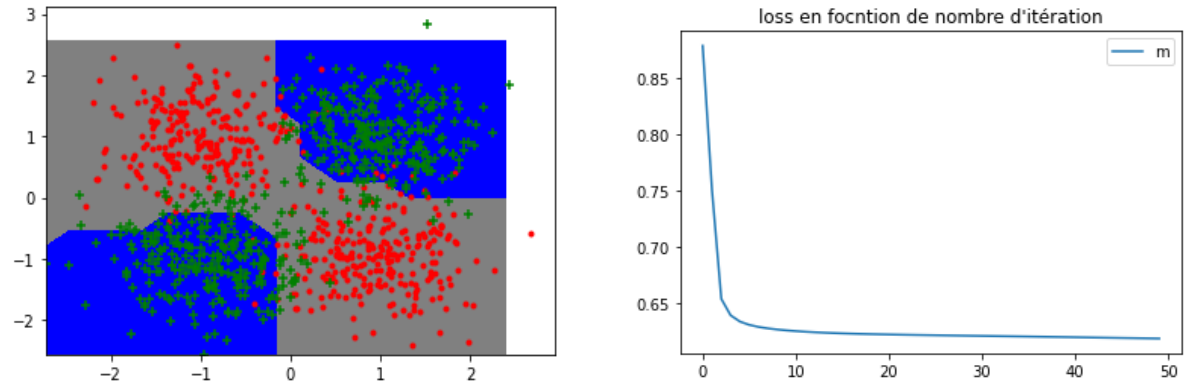


Figure 6 Frontière de décision sur des données de type XOR

2.2.2 Avec la deuxième réseau

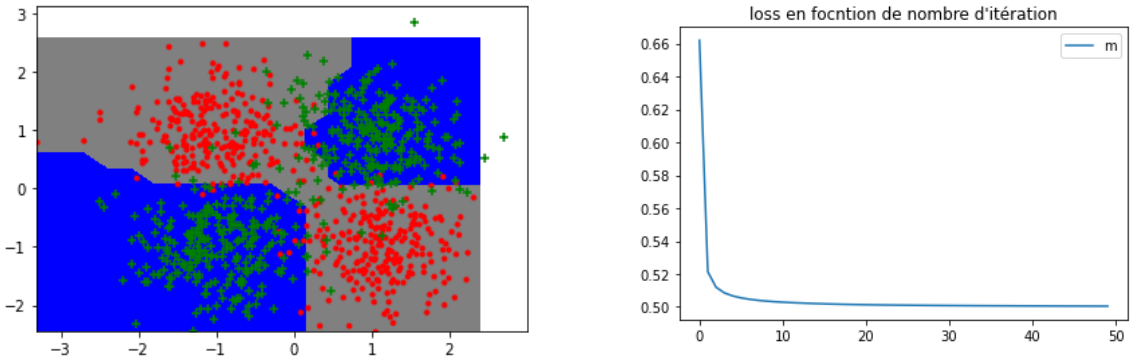


Figure 7 Frontière de décision sur des données de type XOR

On remarque qu'en apprentissage les frontières sont bien précis que celle du test et c'est logique car c'est les données de traininng,le taux de bonne classification trouver dans le premier réseau est égale à 92.6%,avec le 2éme réseau on trouve un score qui égale a 94.6%,on remarque que en augmentant le nombre de couche (même si une couche linaire simple) ça permet d'augmenter le score. Aussi on remarque une convergence rapide de la fonction de loss (on a utilisé le MSE)

2.3 . Echiquier :

2.3.1 Avec le premier réseau .

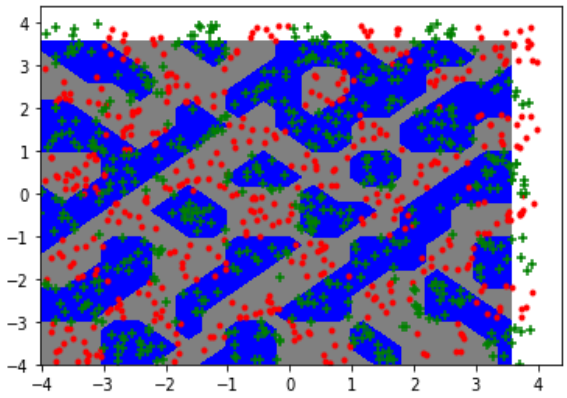


Figure 9 Les frontières de décision avec les données d'apprentissage

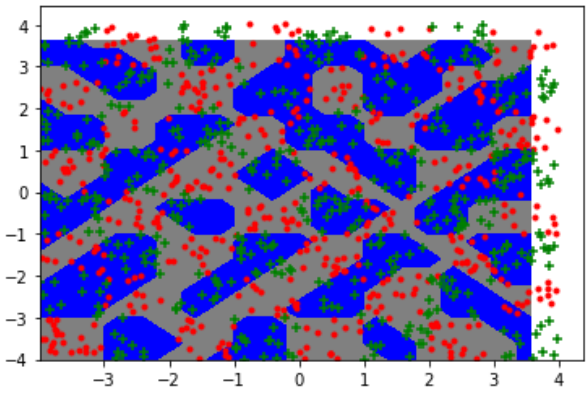


Figure 10 Les frontières de décision avec les données de test

2.3.2 Avec la deuxième réseau .

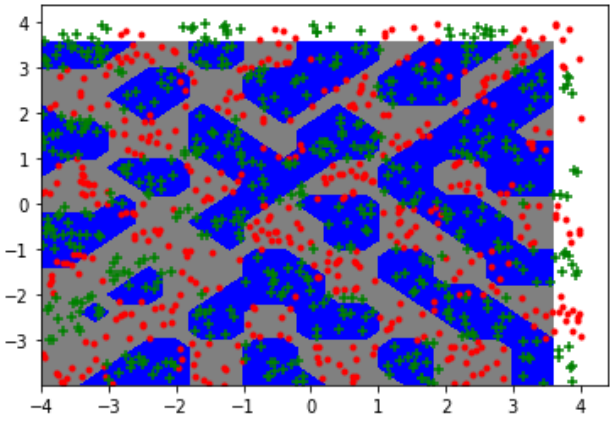


Figure 12 Les frontières de décision avec les données d'apprentissage

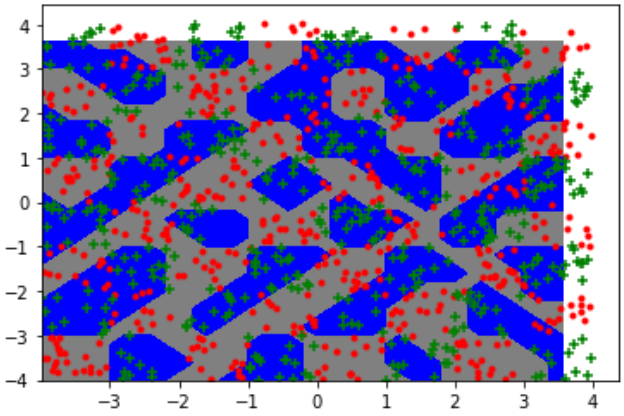


Figure 11 Les frontières de décision avec les données de test

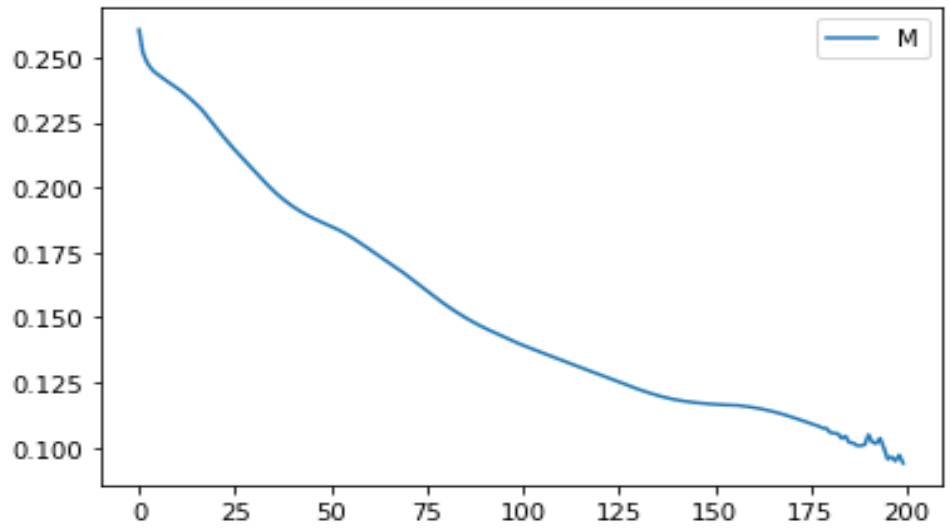


Figure 13 moyenne des loss en fonction de nombre d'itération.

On remarque que pour les données en apprentissage et en test, le programme a bien détecter les frontières ,le taux de bonne classification trouver dans le premier réseau égale à 86,7%,alors que pour les données en test le taux de bonne classification est 80.5%,pour la deuxième réseau le taux de bonne classification trouver est égale a 92.23% et pour les données de test on trouve un taux de bonne classification de 82%,donc on peut observer que en augmentant le nombre permet d'améliorer le score. Aussi on remarque que la convergence de la moyennes des loss est lents.

3. Mon quatrième est multi-classe.

Dans cette partie on va utiliser la base USPS et l'objectifs c'est de classer les nombre de cette base de données. On va tester la classification de multi-classe sur 2 type de réseau,

1. On utilise le réseau.

$$\text{Linear} \rightarrow TanH \rightarrow \text{Linear} \rightarrow TanH \rightarrow \text{Linear} \rightarrow Softmax$$

Avec la cross entropie comme fonction de couts on trouve les résultats suivant.

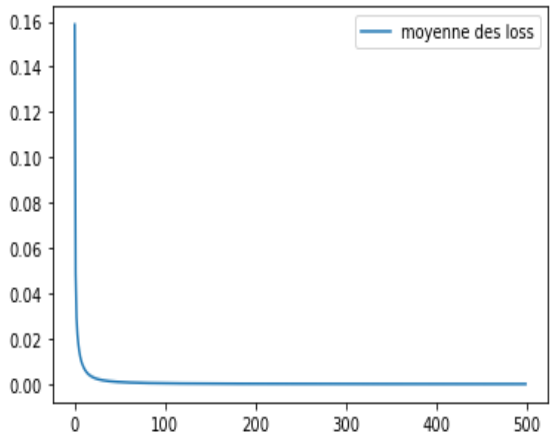


Figure 15 moyenne des loss en fonction de nombre d'itération

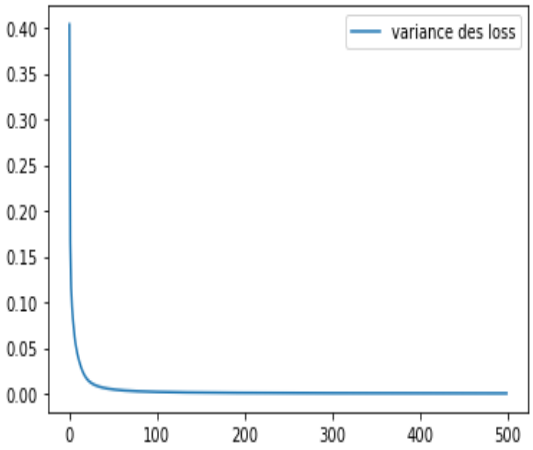


Figure 14 variance des loss en fonction de nombre d'itération

Le graphe représente le nombre d'itération en
fonction de moyenne des loss,on remarque que la moyenne avec la variance des loss converge rapidement tel que plus le nombre d'itération augmente plus les deux métriques converge, le taux de bonne classification trouver est égale a : 97%

2. On utilise le réseau.

$$\text{Linear}, \text{TanH}(), \text{Linear}, \text{TanH}(), \text{Linear}$$

Avec logsoftmax comme fonctions de couts.

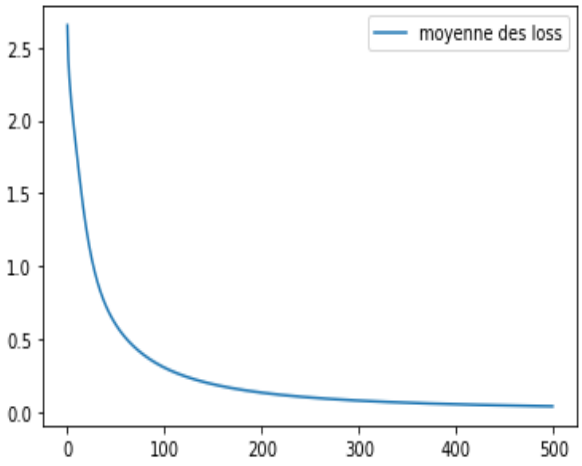


Figure 17 Moyenne des loss en fonction de nombre d'itération

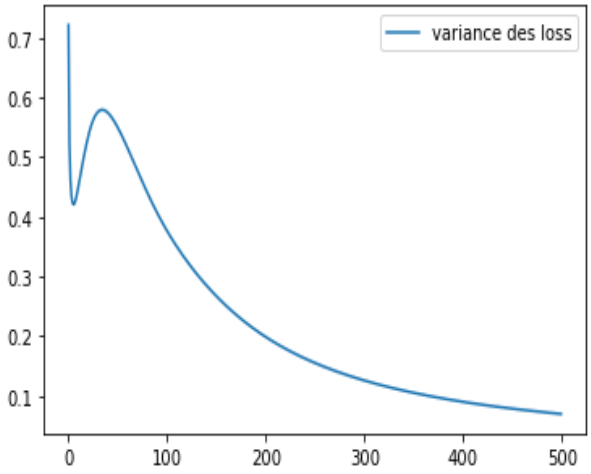


Figure 16 variance des loss en fonction de nombre d'itération

Avec cette réseau on trouve un taux de bonne classification qui égale a 96%,On remarque que la moyenne des loss converge rapidement, par contre la variance ne converge pas rapidement en comparant avec le premier réseau.

4. Mon cinquième se compresse.

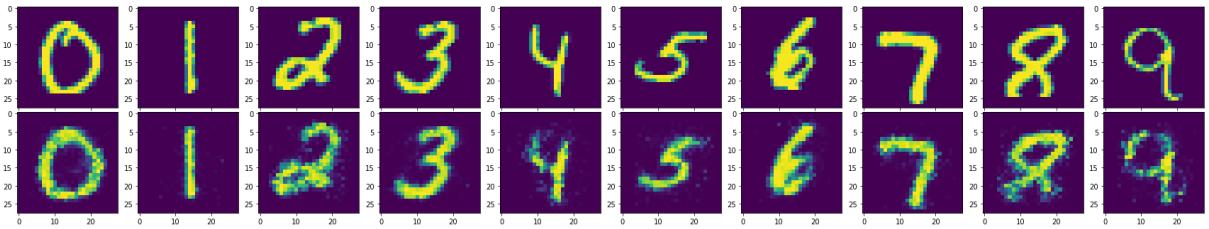
4.1 Visualisation des images après une compression.

Dans la compagnie des tests on va utiliser la base de données usps.

4.1.1 On commence par le réseau de neurone suivant :

$$\text{Linear}(784,512) \rightarrow \text{TanH} \rightarrow \text{Linear}(512,784) \rightarrow \text{sigmoid}$$

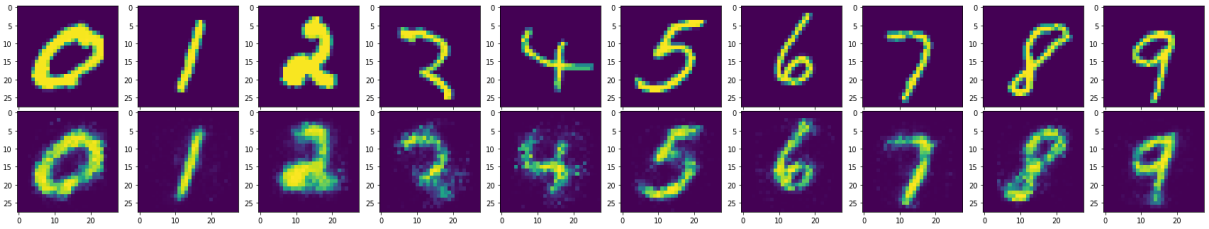
Et on utilise une cross entropie binaire comme fonction de loss, on trouve les résultats suivants :



4.1.2 en changeant les dimensions du module linaire et en diminuant la dimension de sortie on trouve les résultats suivants :

$$\text{Linear}(784,512) \rightarrow \text{TanH} \rightarrow \text{Linear}(512,256) \rightarrow \text{sigmoid}$$

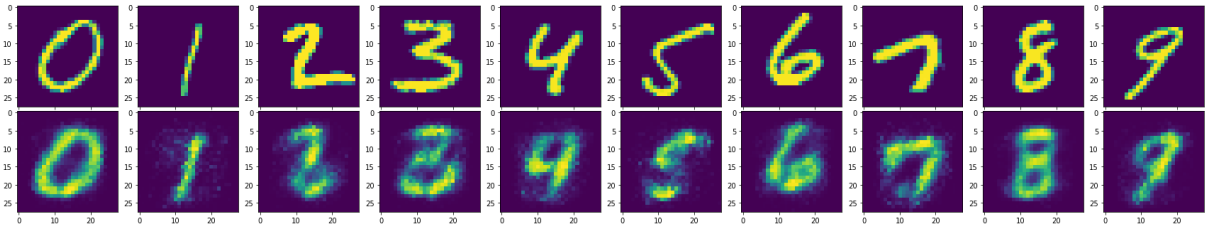
Avec une cross entropie comme fonction de loss,on trouve les résultats suivants :



4.1.3 en diminuant une autre fois la dimension de sorti de 512 :

$$\text{Linear}(784,3) \rightarrow \text{TanH} \rightarrow \text{Linear}(3,256) \rightarrow \text{TanH}$$

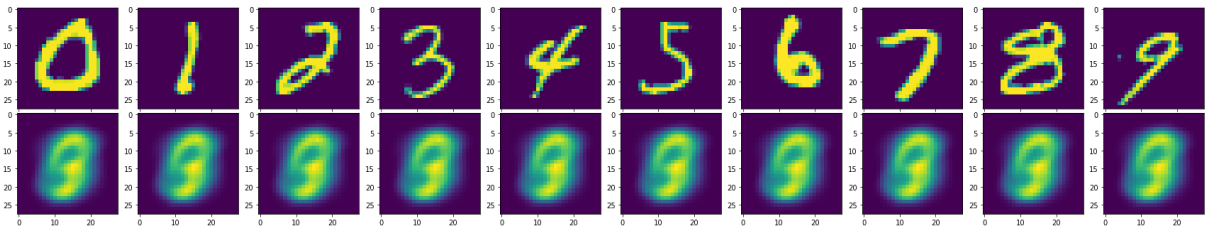
Avec une cross entropie comme fonction de loss,on trouve les résultats suivants :



4.1.4 on essaye de changer la dimension de sortie a 8 :

$$\text{Linear}(784,128) \rightarrow \text{TanH} \rightarrow \text{Linear}(128,8) \rightarrow \text{TANh}$$

Avec une cross entropie comme fonction de loss,on trouve les résultats suivants :

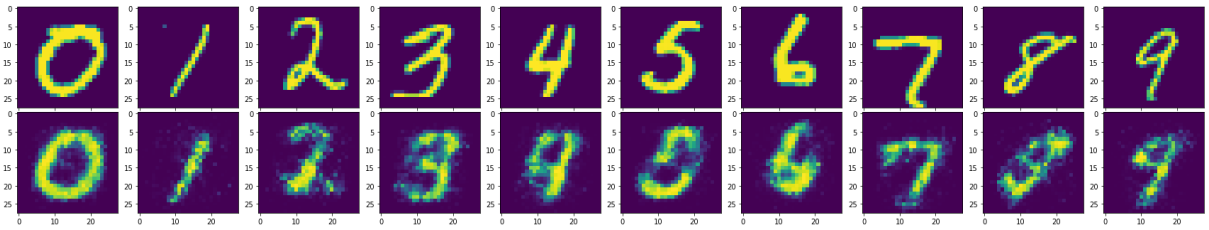


On remarque qu'à chaque fois les dimensions diminuent l'image devient flou et elle perdre ces caractéristique du coup le taux de bonne classification démunie.

5 Avec le réseau:

$$\text{Linear} \rightarrow \text{TanH} \rightarrow \text{Linear} \rightarrow \text{TanH} \rightarrow \text{Linear} \rightarrow \text{sigmoid}$$

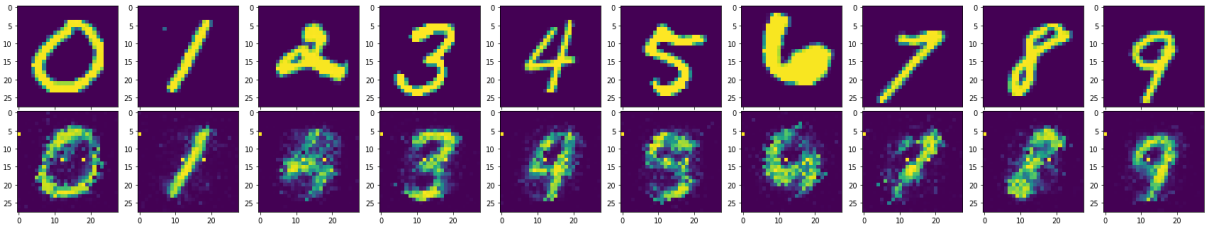
Et avec une cross entropie binaire comme fonction de cout on trouve les résultats suivants :



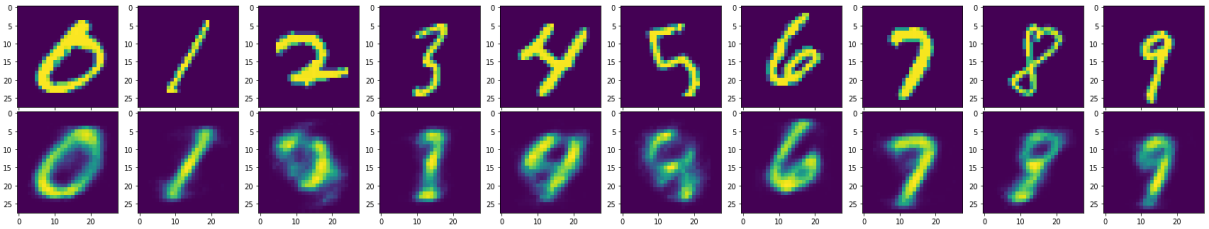
6 Avec le réseau de neurone suivant :

$$\text{Linear} \rightarrow \text{TanH} \rightarrow \text{Linear} \rightarrow \text{Sigmoid}$$

Et avec mse comme fonction de cout.



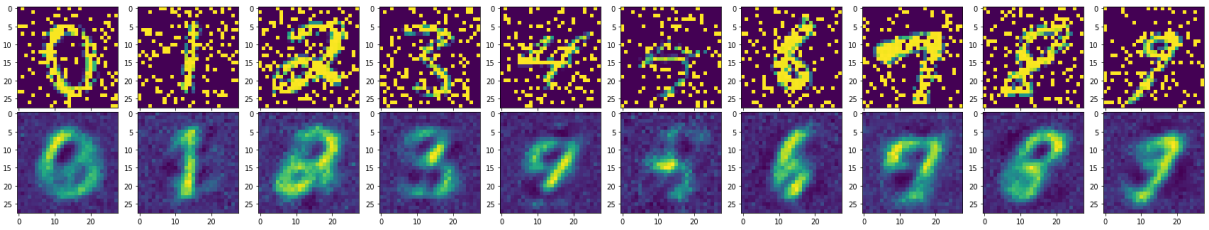
On essaye de faire une reconstruction après une fort compresion,L'image suivant représente les numéros après une compression puis une reconstruction.



On remarque que le numéros 5 et 2 ne sont pas reconstruit correctement. Alors que le 0,9 et 7 sont bien construit, En gros on peut dire que notre programme a bien fait la reconstruction ,le taux de bon classification trouver est égale à 85.45%

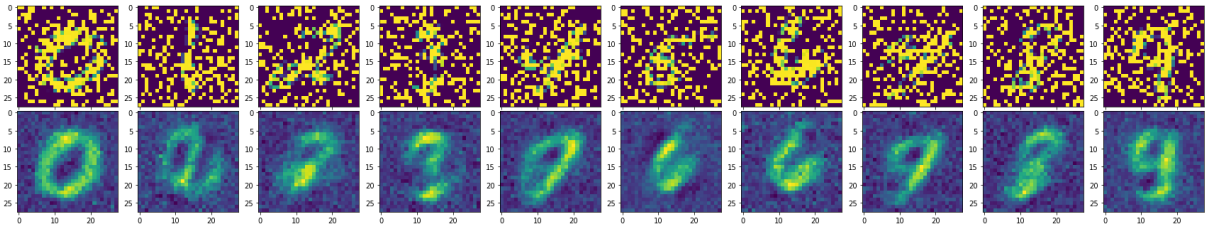
On essaye de bruiter les données et observer l'erreur entre la donnée débruitée par le réseau et la donnée originale.

Avec un 30% de bruit on trouve les résultats suivants :



On trouve une erreur qui égale à 0.39.

Avec un bruit de 50% sur les données



On trouve une erreur qui égale à 0.45

On remarque que plus le bruit est grand, plus l'erreur est grande, aussi on observe que même si les données sont bruitées, l'auto-encodeur a réussi a reconstruit certain lettre comme le 9 et 0.

a. visualisation des représentations obtenues dans un espace 2D ou 3D avec T-sne.

Le t-SNE (t-Distributed Stochastic Neighbor Embedding) est une technique de réduction de la dimensionnalité qui convient particulièrement bien à la visualisation d'ensembles de données à haute dimension.

On utilise la base de données usps, on essaye de visualiser la représentation latent et original avec T-sne.

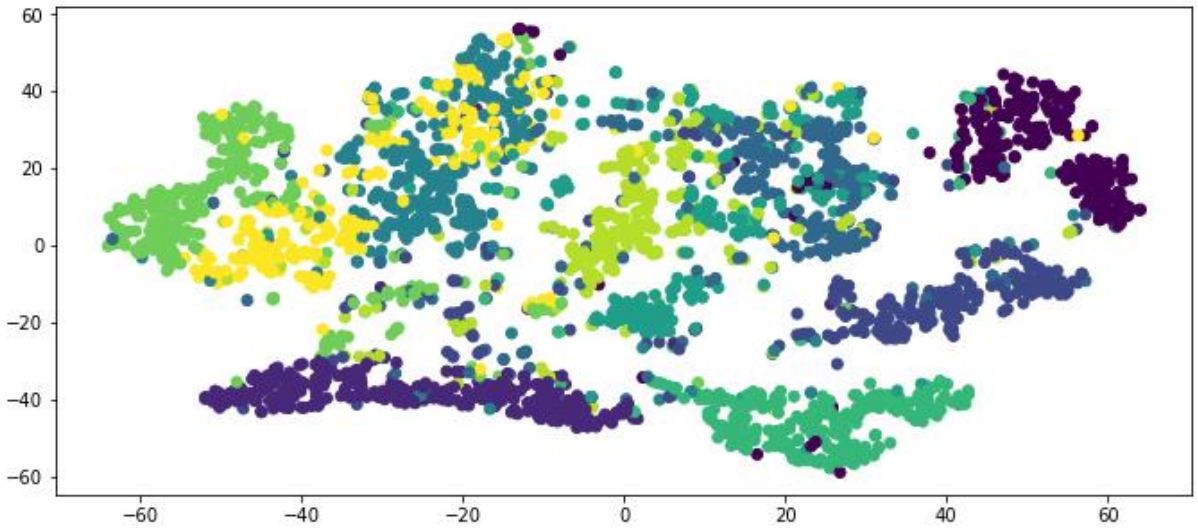


Figure 18:Espace Originale

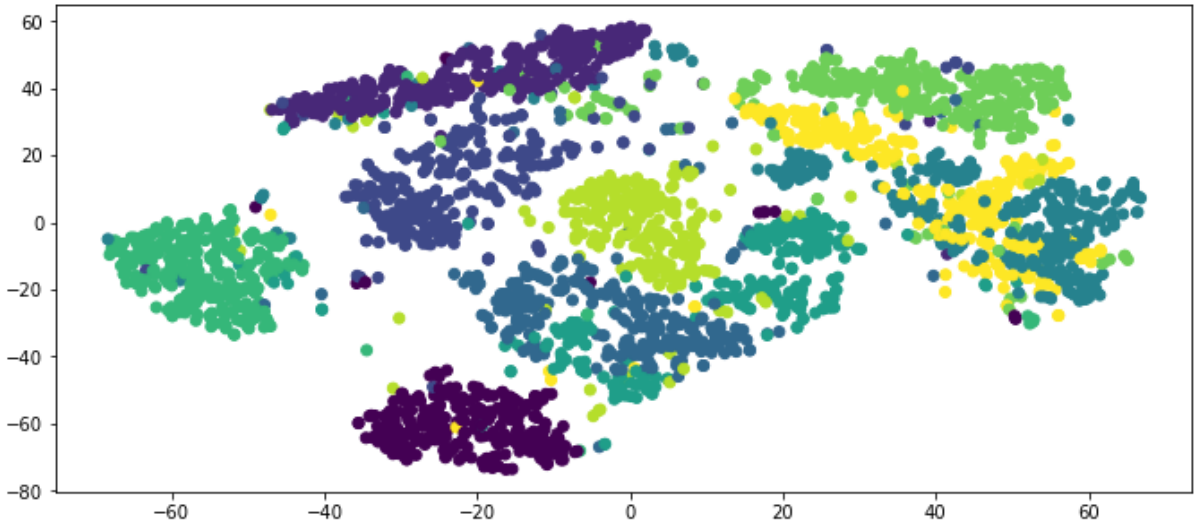


Figure 19:Espace Latent

On remarque que le tsne à bien représenter les données pour l'espace originale, même cas pour l'espace latent ou on trouve que chaque classe est regrouper dans une région ,et c'est à cause de la compression.

3.2 Clustering :

En utilisant les données minst de sklearn on essaye d'appliquer un kmeans sur l'espace latent et l'espace originale.

Pour évaluer nos cluster on va calcule la Purity de chaque cluster pour chaque espaces.

Espace Orignal.

Cluster	0	1	2	3	4	5	6	7	8	0
purity	0.43	0.92	0.7	0.76	0.62	0.58	0.76	0.68	0.71	0.69

Espace Latent.

Cluster	0	1	2	3	4	5	6	7	8	0
purity	0.68	0.96	0.45	0.49	0.42	0.65	0.48	0.76	0.57	0.57

3.3 Classification :

On refaire l'opération de la classification,mais cette fois ci avec les données générer par l'auto-encodeur en utilisant un réseau de neurone a 3 couche caché, les résultats trouvé sont dans le tableau suivant :

Opération	Taux de bonne classification
Données test sans aucune traitement	0.86
Données test générer par l'autoencodeur	0.83
Données test bruité	0.82
Données test bruité par l'auto-encodeur	0.9

On remarque que si on utilise les données générer par l'auto-encodeur et on applique une classification le score dimunie,même avec des données bruité le score démunie aussi, mais si on applique le bruit sur les données et on fait l'opération de l'auto-encodeur après une classification ça permet d'augmenter le score.

Mon sixième se convole :

Définition :

Les couches de convolution convolutions 2D, extrayant des batch 2D à partir de tenseurs d'image et appliquant une transformation identique à chaque batch. De la même manière, on utilise des convolutions 1D, extraire des batches 1D locaux (sous-séquences) à partir de séquences.

Les CNN 1D sont structurés de la même manière que leurs équivalents 2D, ils se composent d'une pile de couches Conv1D et MaxPooling1D, se terminant par une couche de Flatten, qui transforment les sorties 3D en 2D sorties.

Expérimentation :

En appliquant sur le réseau de neurones proposé dans l'énoncé

Conv1D (3,1,32) → MaxPool1D (2,2) → Flatten () → Linear (4064,100) → ReLU() → Linear(100,10)

Observation :

On a obtenu un taux de bonne classification égale à 0.91 avec un nombre d'itération qui égale a 50,avec un nombre d'itération qui égale a 200 on trouve un score qui égale a 0.92,un le réseau linaire on a trouver de score qui égale a 0.89 et 0.9,on voit bien que le réseau convolutif permet d'améliorer le score.