Probabilistic Mdoels
RBM & Binary Alpha Digit

Auteurs

Chen Marc
Sebti Adam

Décembre 2024
3A Option IA, Palaiseau

Table des matières

1 Introduction

Dans ce projet, nous explorons les Restricted Boltzmann Machines (RBM), un modèle probabiliste utilisé pour l'apprentissage non supervisé.

L'objectif est de concevoir une RBM capable d'apprendre les caractères de la base de données Binary AlphaDigits et de générer des échantillons similaires. Le rapport détaille la construction, l'entraînement et l'évaluation du RBM, avec une analyse des hyperparamètres et une comparaison avec d'autres architectures.

2 Données Binary Alpha Digit

La base de données AlphaDigits contient des images binaires de caractères manuscrits alphabétiques (A-Z) et numériques (0-9), représentées sous forme de matrices avec des valeurs de pixels égales à 1 (noir) ou 0 (blanc).

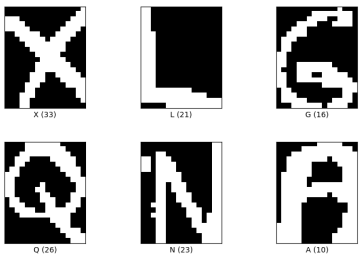

FIGURE 1 – Échantillon du dataset BinaryAlphaDigit

Le dataset a été préalablement traité afin de vectoriser les images pour faciliter la phase d'entraînement.

3 Machine de Boltzmann restreinte

Une Restricted Boltzmann Machine (RBM) est un modèle probabiliste d'apprentissage non supervisé, constitué de deux couches de neurones : une couche visible et une couche cachée.

Une RBM apprend en ajustant les poids entre une couche visible et une couche cachée pour modéliser les données. Les neurones s'activent de manière probabiliste et l'entraînement optimise les connexions pour capturer les relations entre les variables, qui dans notre cas sont les différents pixels d'une image.

4 Analyse du modèle en fonction des hyper-paramètres

Nous allons étudier l'influence des hyper-paramètres sur la performance du modèle. Voici les paramètres que comprend la RBM :

— **Nombre d'unités cachées** : taille de la couche cachée.

— **Taux d'apprentissage** : contrôle la vitesse de mis-à-jours des poids.

— **Taille de batch** : nombre échantillons utilisés à chaque itération.

Pour l'étude nous avons entraîné le modèle sur 2000 epochs et sur un seul caractère.

4.1 Nombre d'unité cachées

Nous avons étudié l'influence du nombre d'unités cachées sur le modèle.

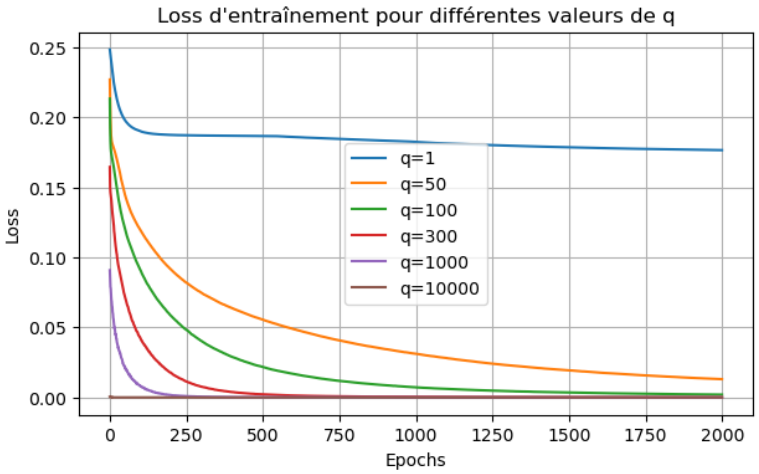

FIGURE 2 – Erreur de reconstruction

L'erreur de reconstruction mesure la capacité du modèle à reconstruire les données d'entrée après les avoir codées dans l'espace latent et décodées à nouveau. On constate que l'erreur de reconstruction diminue avec l'augmentation du nombre d'unités cachées. Toutefois pour un nombre d'unité trop faible la convergence n'est pas optimale ou bien plus lente.

L'énergie libre évalue la capacité du modèle à capturer les relations dans les données : une faible énergie libre moyenne indique un apprentissage efficace avec des probabilités élevées pour les configurations visibles.

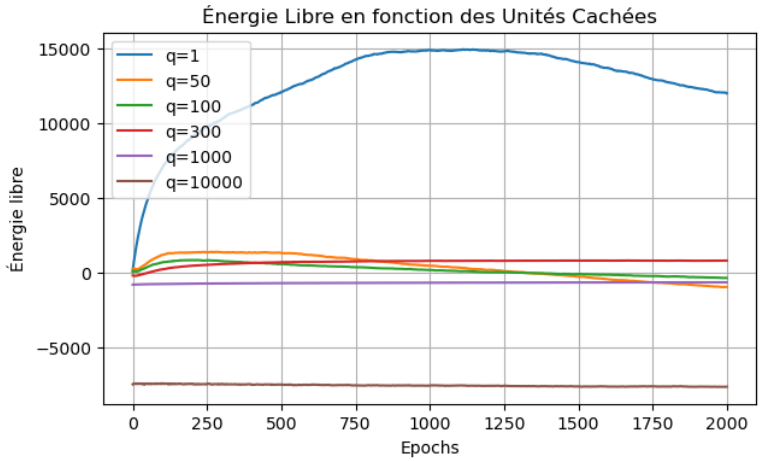

FIGURE 3 – Énergie libre en fonction des unités cachées

Il est intéressant de constater que l'énergie pour q=50 et q=100 diminue au cours de l'entraînement comme prévu et que paradoxalement elle augmente légèrment pour q=300. C'est avec q=10000 que le modèle représente bien les données visibles.

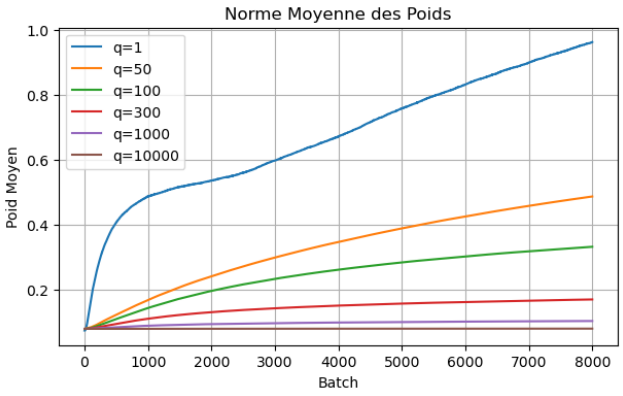

FIGURE 4 – Norme moyenne des poids

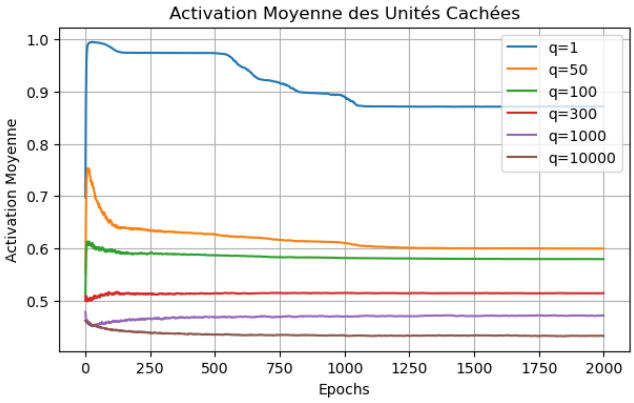

FIGURE 5 – Activations moyennes

On pourrait croire qu'un nombre plus grand d'unités cachées ne peut qu'améliorer le modèle, mais d'après ces résultats la moyenne de la norme des poids diminue fortement avec l'augmentation des unités cachées ce qui s'explique par le fait qu'une grande partie de ces derniers soient nuls. De plus on remarque une sous-utilisation des unités cachées lorsqu'elles sont trop nombreuses, le modèle n'exploite pas pleinement sa capacité à extraire les caractéristiques des données.

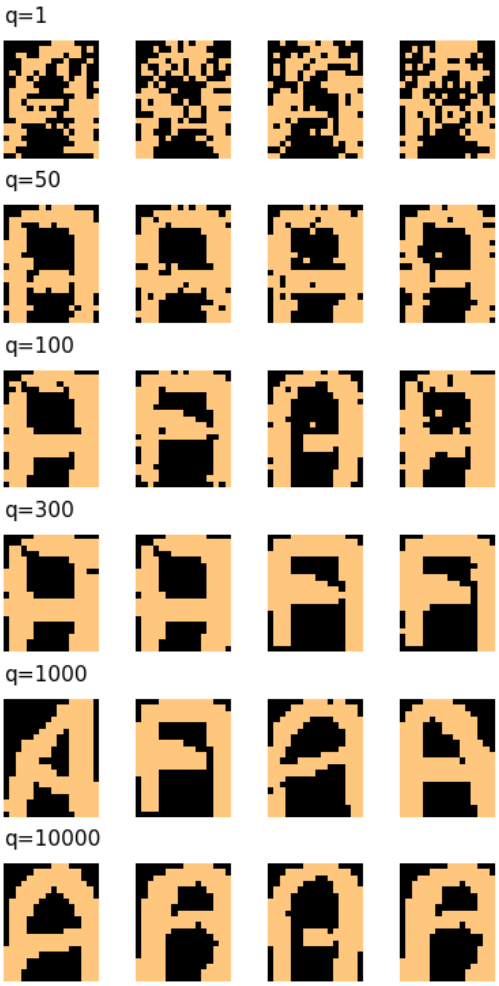

FIGURE 6 – Images générées avec la RBM

Visuellement avec un nombre d'unités cachées plus grand les résultats sont bien meilleurs, ça confirme nos observations sur l'erreur de reconstruction et sur l'énergie libre. Toutefois la complexité spatiale est bien plus élevée et surtout le modèles n'exploite pas toutes ses capacités puisque un grand nombre d'unités sont peu activées et de nombreux poids sont nuls, des techniques de régularisation peuvent être mises en place pour pallier au problème. Il faut donc trouver le bon équilibre entre complexité spatiale, complexité temporelle et qualité des résultats pour bien choisir le nombre d'unités cachées.

4.2 Taux d'apprentissage

Nous avons étudié l'influence du taux d'apprentissage sur le modèle. Pour l'expérience nous avons fixé epochs=2000 et le nombre d'unités à 500.

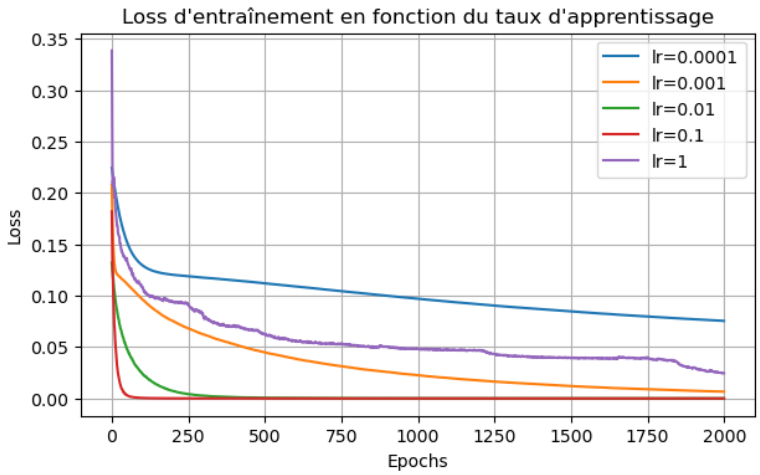

FIGURE 7 – Loss en fonction du taux d'apprentissage

Si le taux d'apprentissage est trop faible le modèle n'arrive pas à apprendre et donc à converger. Si le taux est trop élevé la convergence est lente mais aussi très instable.

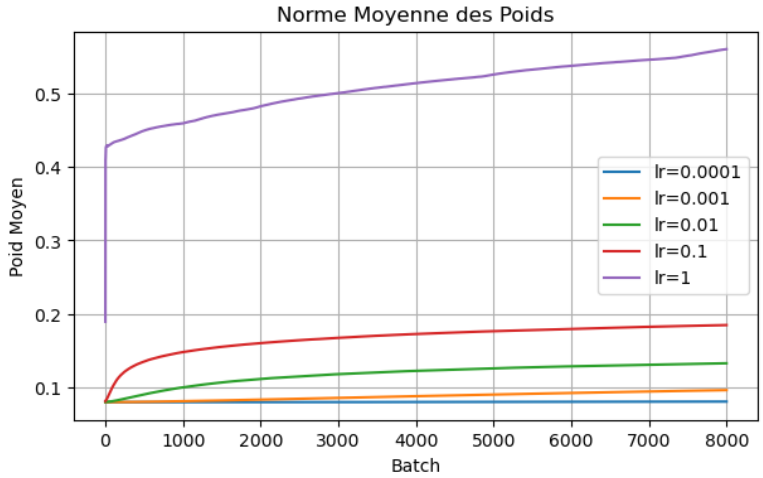

FIGURE 8 – Norme moyenne des poids en fonction du taux d'apprentissage

Avec un taux relativement faible (inférieur à 0,1) les poids sont du même ordre de grandeur. Avec un taux plus élevé on observe une explosion des poids.

Pour essayer de comprendre au mieux nous avons aussi analysé la norme des gradients.

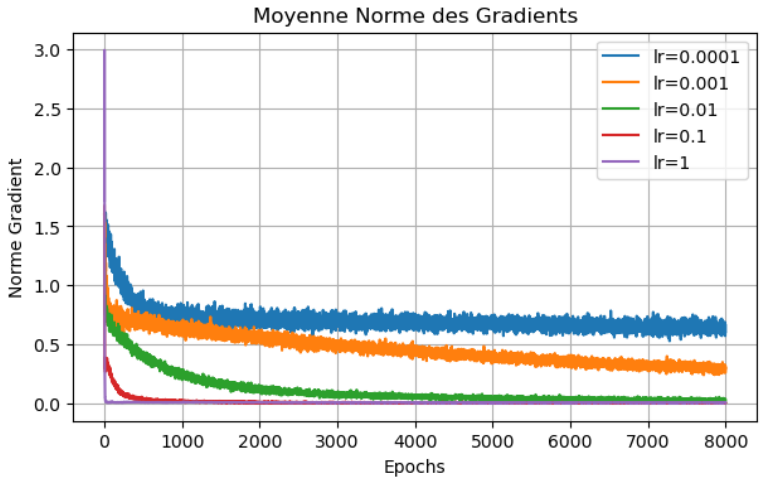

FIGURE 9 – Norme moyenne des gradients selon le taux d'apprentissage

Avec les taux d'apprentissage les plus faibles les gradients restent plutôt élevés puisque le modèle est encore en cours d'"apprendre" et prend plus de temps pour converger. Pour lr=1 et lr=0.1 la convergence est similaire, elle est quasi instantanée. On pourrait croire que les deux modèles ont convergé vers la solution optimale mais ce n'est le cas que pour lr=0.1. En effet pour lr=1 le pic initial important et la chute brutale qui s'en suit suggèrent qu'une partie considérable des poids devient nulle dès le début de l'entraînement, on s'attend donc à de mauvaises performances pour ce modèle.

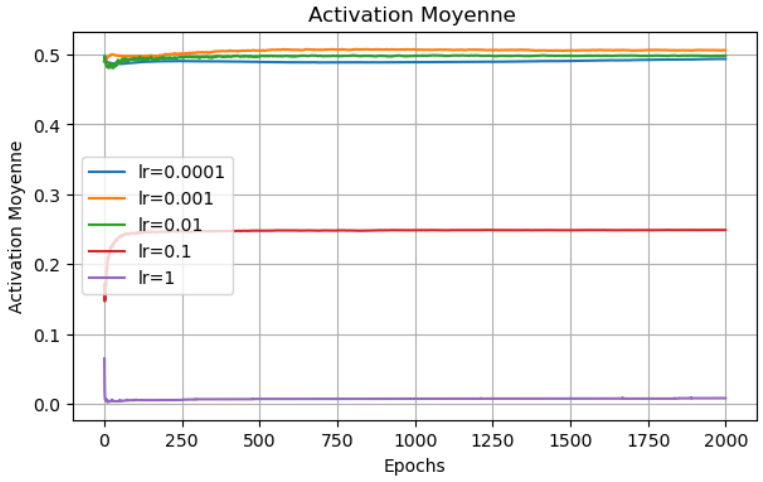

FIGURE 10 – Activation moyenne des unités cachées selon le taux d'apprentissage

On observe que l'activation des unités cachées lors de l'inférence est bien trop faible pour des taux d'apprentissage élevés, ceci s'explique par un apprentissage inefficace et sûrement aussi une saturation des fonctions d'activation qui entraîne une faible diversité dans les activations des unités cachées.

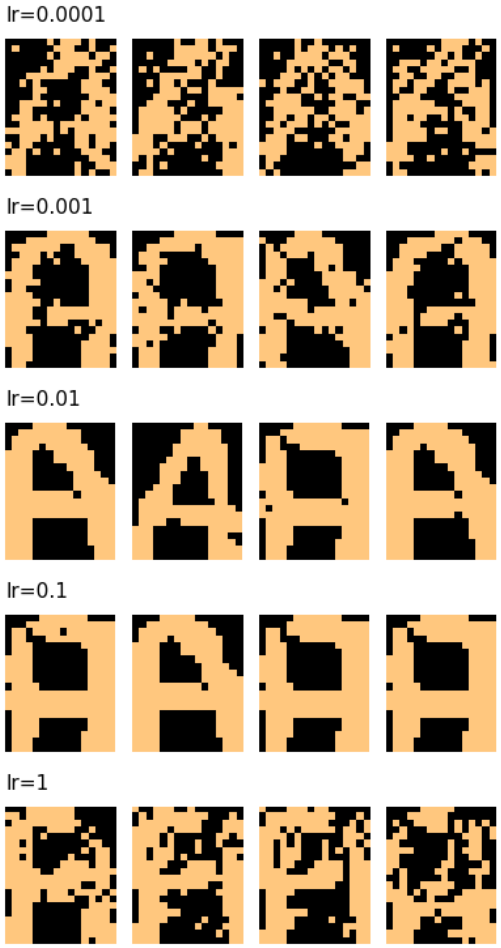

FIGURE 11 – Images générées avec la RBM

On retrouve des images qui sont cohérentes avec les résultats précédents. Pour des taux trop grands ou trop faibles les images générées sont peu fidèles.

Le taux d'apprentissage optimal est compris entre 0,01 et 0,1.

4.3 Taille du batch

Nous avons par la suite étudié l'influence de la taille du batch sur le modèle. Le nombre d'epochs, le taux d'apprentissage et le nombre d'unités cachées sont fixés.

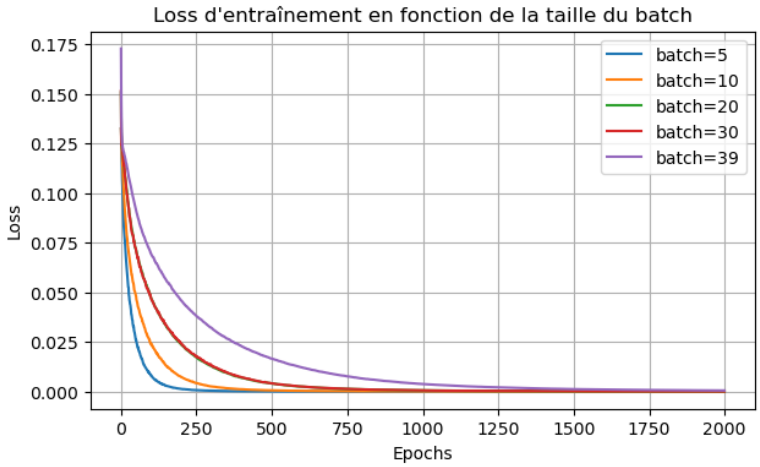

FIGURE 12 – Loss en fonction de la taille du batch

La convergence de la loss devient plus rapide si le batch devient plus petit. Ce résultat est logique puisque avec un batch plus petit les poids seront mis à jours plus souvent au cours d'un seul epoch.

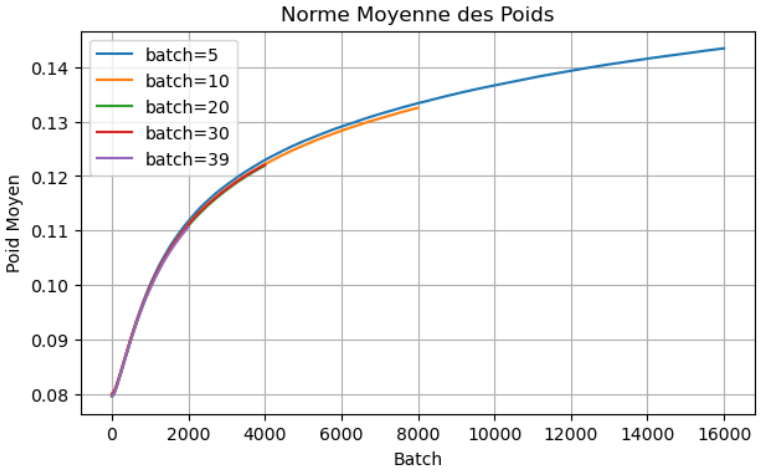

FIGURE 13 – Norme moyenne des poids en fonction de la taille du batch

La taille du batch n'affecte pas directement sur la valeur des poids.

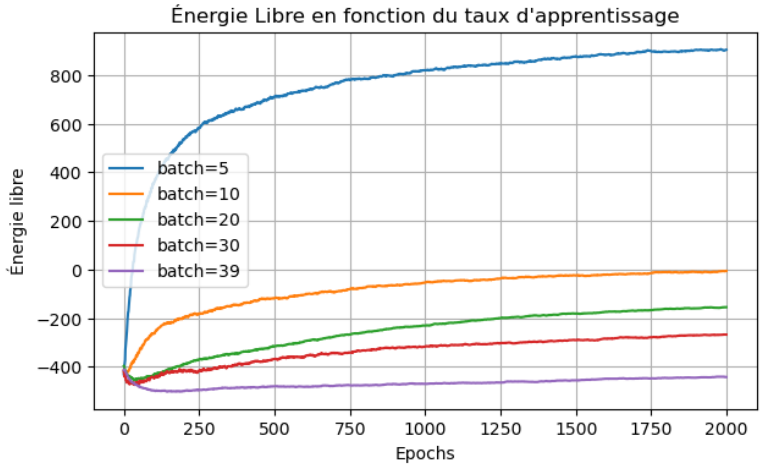

FIGURE 14 – Norme moyenne des poids en fonction de la taille du batch

On constate que un batch plus grand réduit l'énergie libre. Réduire l'énergie libre signifie diminuer l'incertitude ou l'erreur du modèle dans la représentation des données.

Une plus grande taille de batch signifie que l'estimation du gradient est effectuée sur un plus grand nombre de données, ce qui tend à être une estimation plus précise du gradient réel par rapport à des batchs plus petits.

Une énergie libre plus faible s'explique aussi par un réduction du bruits et des perturbations qu'on obtient en utilisant un batch plus grand : un batch plus grand permet de lisser le bruit car le modèle voit plus d'exemples à chaque étape et effectue donc des mises à jour moins instables.

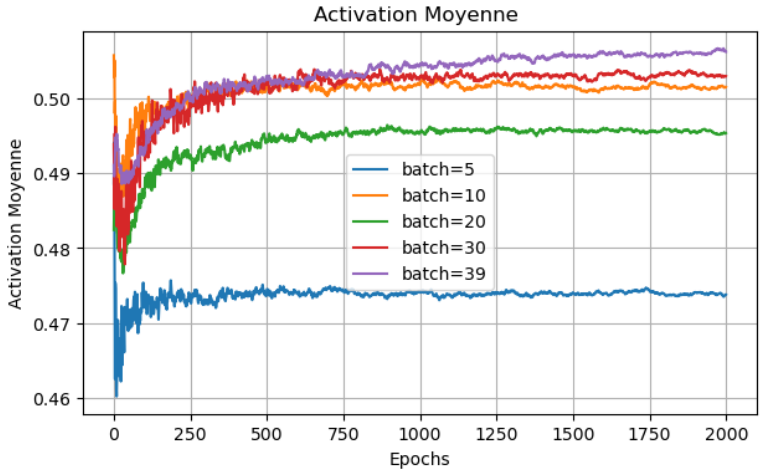

FIGURE 15 – Activation moyenne en fonction de la taille du batch

Avec un batch plus grand l'activation moyenne est plus élevée. En effet un batch plus grand permet de mieux représenter la distribution des données et de mieux explorer la diversité des données, ainsi les unités cachées sont capables de capturer des caractéristiques et des relations plus complexes nécessitant donc une activation plus fréquente.

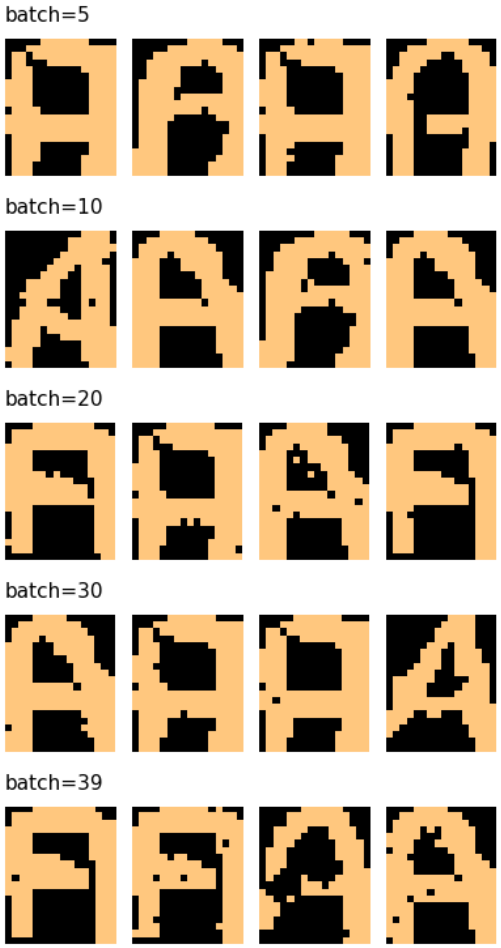

FIGURE 16 – Images générées avec RBM

Malgré les avantages d'une taille de batch plus grande en terme de stabilité et lissage du bruit les résultats obtenus sont moins bons qu'avec une taille plus faible. Ceci est peut être du à une exploration insuffisante de l'espace des solutions : une taille de batch trop grande limite la capacité du modèle à explorer différentes solutions dans l'espace des paramètres, le modèle converge rapidement vers une solution, souvent locale, mais ne trouve pas forcément la meilleure solution.

5 Pouvoir modélisant en fonction du nombre de caractères à apprendre

La capacité du modèle à générer des résultats de qualités et fidèles aux données d'origine dépend fortement du nombre de caractères à apprendre. Nous analyserons ici les différents modèles.

Les données d'entraînement sont différentes pour chaque modèle :

— **modèle 1** : ['A']
— **modèle 2** : ['A','B']
— **modèle 3** : ['A','B','C','D']
— **modèle 4** : ['1','2','3','4','5','6','7','8','9','0']
— **modèle 5** : ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
— **modèle 6** :['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9']

5.1 Erreur de reconstruction

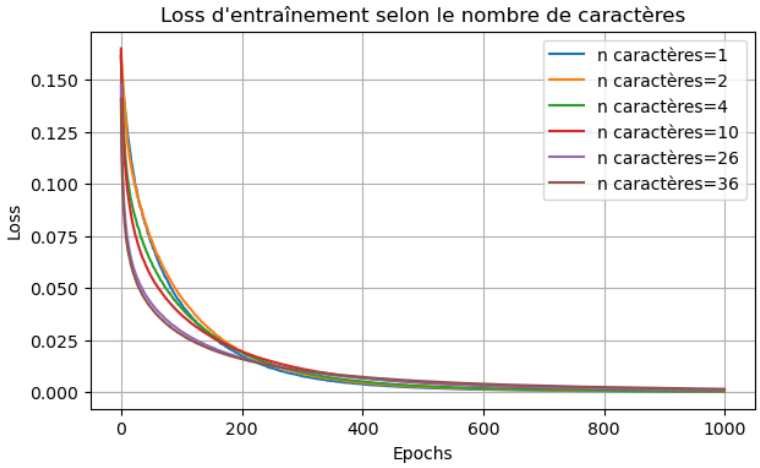

FIGURE 17 – Erreur de reconstruction en fonction du nombre de caractères

L'erreur de reconstruction du modèle converge indépendamment du nombre de caractères.

TABLE 1 – Erreur de reconstruction minimale

Modèle	1	2	3	4	5	6
Nombre Caractères	1	2	4	10	26	**36**
Erreur	0.0003824	0.0004713	0.0005336	0.0011187	0.0011528	**0.0016191**

Toutefois si on observe de plus près on remarque que plus le nombre de caractères est élevé plus l'erreur de reconstruction est grande : l'erreur de reconstruction avec un seul caractère est 4 fois meilleure qu'avec 36.

On voit que la généralisation est moins efficace et fiable avec l'augmentation des caractères.

5.2 Énergie Libre

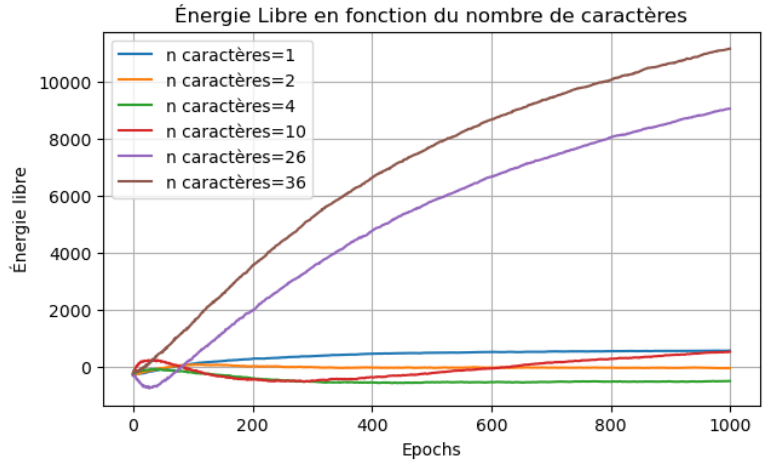

FIGURE 18 – Erreur de reconstruction en fonction du nombre de caractères

On remarque pour tous les modèles contenant tout l'alphabet, les chiffres et toute les données l'énergie libre augmente au cours du temps. Cela peut s'expliquer par un sous-ajustement du modèle :il ne parvient pas à capturer la distribution des données. Une solution serait de revoir les hyper-paramètres notamment le taux d'apprentissage et le nombre d'unités cachées pour mieux s'adapter à la distribution réelle.

5.3 Divergence de Jensen-Shannon

La divergence de Jensen-Shannon (JS divergence) mesure la similarité entre deux distributions de probabilité en calculant la moyenne symétrique des différences d'information entre chaque distribution et la distribution moyenne des deux. Plus la divergence est faible plus les distributions des données étudiées sont proches.

TABLE 2 – Divergence de Jensen-Shannon

Modèle	1	2	3	4	5	6
Nombre Caractères	1	2	4	10	26	**36**
Divergence JS	0.07324	0.09629	0.08856	**0.34035**	0.02099	0.10483

Les résultats montrent qu'en général le modèle a plus de mal à générer des données qui sont proches de la distribution réelle lorsque le nombre de caractères augmente.

Nous sommes surpris de voir que les données générées les plus éloignées de la distribution d'origine sont celle du modèle entraîné exclusivement sur les chiffres malgré un nombre de caractères inférieur à l'alphabet. Cela peut s'expliquer notamment par une très grande similitude entre certains caractères (par exemple 1 et 7) ou peut être une représentation trop grossière et approximative de certains caractères.

Suite à une analyse plus approfondie nous avons découvert que le caractère "1" est à l'origine de cet écart. Lorsque le modèle est entraîné avec tous les chiffre en excluant le "1" on retrouve une divergence de l'ordre de grandeur de celle de l'alphabet.

TABLE 3 – Divergence de Jensen-Shannon : influence du caractère 1

Modèle	Chiffres avec 1	Chiffres sans 1
Nombre Caractères	**10**	9
Divergence JS	**0.35057**	0.038748

Cette différence s'explique par une représentation trop aléatoire et approximative du caractère "1". Le modèle a du mal à généraliser sa représentation.

FIGURE 19 – Échantillon de "1"

5.4 Images générées

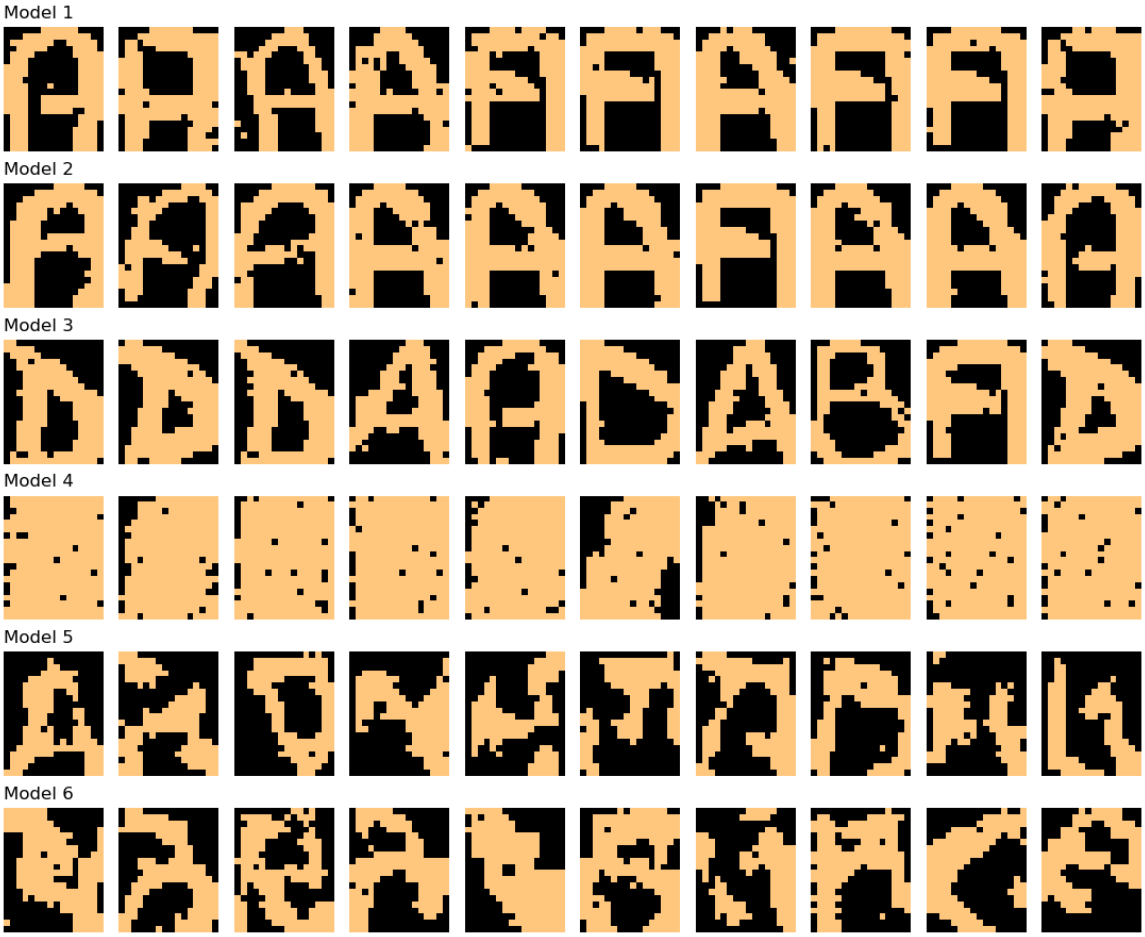

FIGURE 20 – Images générées avec la RBM

On observe des images qui sont cohérentes avec les résultats précédents. Plus le nombre de caractères est grand plus la qualité est moindre, ceci est dû au fait que le même modèle doit capturer une distribution plus grande et plus complexe.

Il est aussi intéressant de voir comme le modèle 4 qui est entraîné sur tous les chiffres performe très mal juste à cause de la présence du chiffre 1.

Pour que le modèle soit plus performant avec un grand nombre de caractères il faut revoir ses paramètres notamment en augmentant le nombre des unités cachées afin d'apprendre cette nouvelle distribution beaucoup plus complexe. On pourrait aussi augmenter la taille de batch pour réduire le bruit et augmenter le nombre d'epochs pour avoir plus d'entraînement.

6 Bases de données alternative et comparaison avec GAN

6.1 Base de données alternative

Nous avons testé la RBM sur la bas de données de chiffres manuscrits MNIST.

FIGURE 21 – Échantillon base de données MNIST

Les images MNIST ont une plus grande résolution nous avons donc testé avec un nombre d'unités cachés plus élevé : q = [100, 500, 1000].

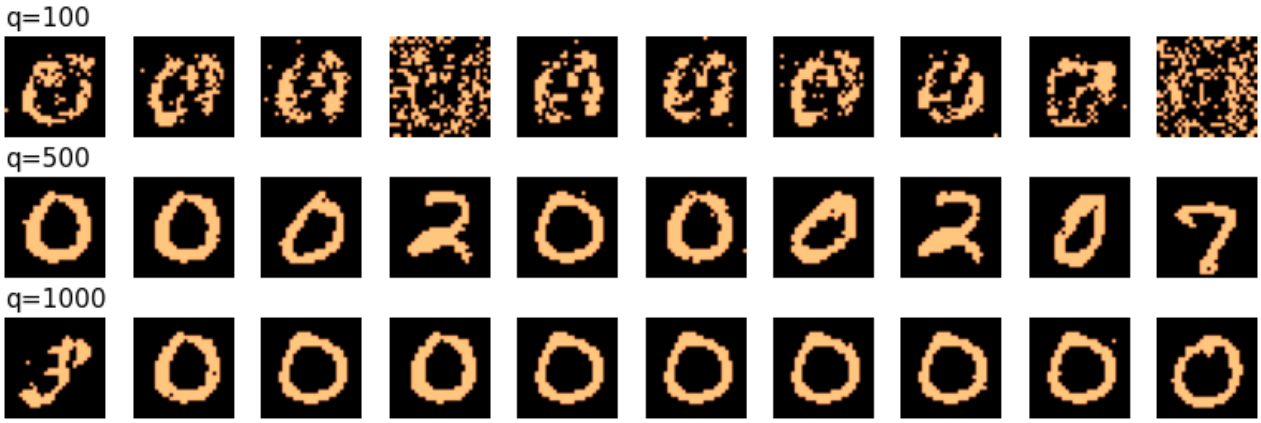

FIGURE 22 – Images générées avec RBM sur MNIST

En effectuant des test ultérieurs et en observant ces images on constate que le modèle souffre d'over fitting et du problème de mode collapse. En effet si on observe pour un même chiffre le modèle reproduit à chaque fois la même représentation (notamment pour le 2), de plus on constate que certaine catégories sont surreprésentées ce qui est le cas notamment du chiffre 0.

Il serait sans doute possible d'améliorer les résultats en ajustant plus finement les hyperparamètres du RBM, en introduisant notamment des régularisations pour éviter le mode collapse.

6.2 Comparaison avec un GAN

Nous avons comparé les résultats obtenus avec un GAN.

FIGURE 23 – Images générées avec un GAN [1]

On remarque que les images générées avec une RBM sont de moins bonne qualité, en plus d'être de meilleur qualité les images du GAN sont plus cohérentes avec la distribution réelle et nettement plus diversifiées.

Les RBMs sont adaptés à des tâches plus simples et à la représentation probabiliste, tandis que les GANs excellent dans la génération réaliste de données complexes tels que la base de données MNIST qui contient des images d'une plus grande résolution.

7 Conclusion

Ce projet a mis en évidence l'efficacité des RBM pour modéliser des distributions et générer des données similaires, tout en soulignant leurs limites surtout face à des modèles plus complets. Bien qu'adaptées à des tâches spécifiques, leur dépendance aux hyperparamètres et leur difficulté à traiter des données complexes appellent à des améliorations pour obtenir des résultats de qualité.

Références

[1] https ://github.com/lyeoni/pytorch-mnist-GAN.