1 Introduction

Les modèles génératifs constituent une classe de modèles puissants en intelligence artificielle, permettant de créer de nouvelles données réalistes à partir de distributions de probabilité. Dans ce rapport, nous explorons l'efficacité de plusieurs de ces modèles, notamment les Restricted Boltzmann Machines (RBM) et les Deep Belief Networks (DBN). Nous ferons aussi des tests sur des Deep Neural Network (DNN) qui sont des DBN auxquelles on ajoute une couche de RBM de classification. On finira à la fin par une implémentation du VAE et comparaison avec les RBM et les DBN.

2 Dataset

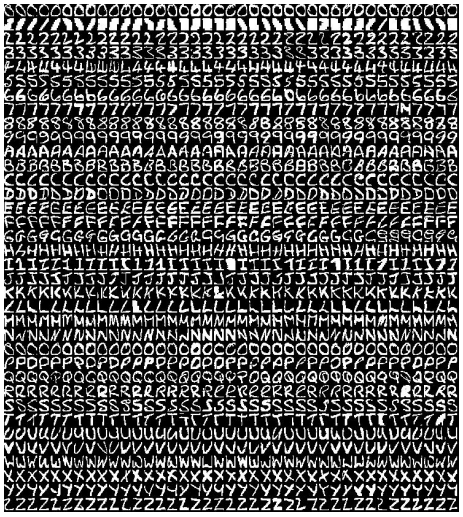
Dans ce projet, deux datasets bien connues ont été utilisées:

- **Binary AlphaDigits:** Servira pour faire des tests rapides.
- **MNIST:** Une base couramment utilisée en machine learning et servira de benchmark.

Binary AlphaDigits

Binary AlphaDigits est une dataset qui contient des images de chiffres de '0' à '9' et des lettres majuscules de 'A' à 'Z' sous forme d'images binaires. Chaque image est de taille fixe: 20×16. La dataset contient 39 exemples de chaque classe.

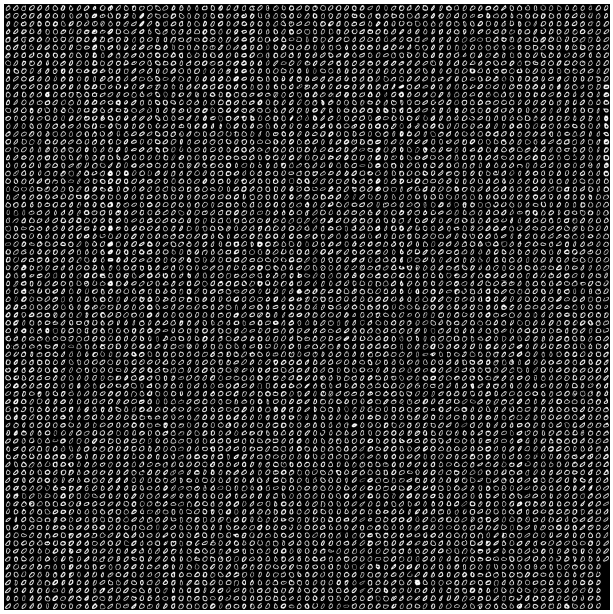
Voici une visualisation de la dataset complète.On peut très bien voir les différentes classes (qui sont au nombres de 36)



MNIST

La dataset MNIST (Modified National Institute of Standards and Technology) est une collection de données très populaire en apprentissage automatique. Elle contient 10 classes, correspondant aux chiffres de '0' à '9'. Les images sont en niveaux de gris avec une taille 28×28 pixels. MNIST contient 60000 images pour l'entraînement et 10000 images pour les tests. Ces images sont des scans de chiffres écrits à la main par des étudiants et des employés du Bureau de recensement des États-Unis.

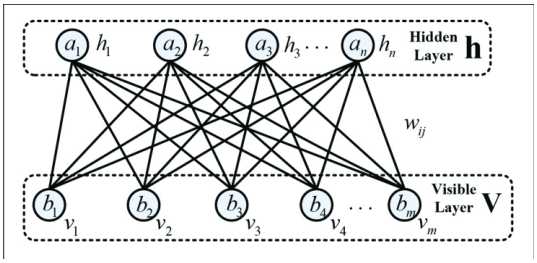
Ci joint, une visualisaion de la classe '0' utilisé pour le train en utilisant MNIST. On peut remarquer qu'il y a beaucoup de variétés dans la façon d'écrire le 0 dans la dataset.



3 Etude sur Binary AlphaDigit

3.1 RBM

Les Restricted Boltzmann Machines (RBM) sont des modèles de réseaux de neurones non supervisés, composés de deux couches de neurones : une couche visible et une couche cachée. Les RBM apprennent à représenter les motifs dans les données en minimisant l'énergie libre. Elles sont efficaces dans la réduction de dimensionnalité, la génération de nouvelles données et l'initialisation de réseaux de neurones profonds. Les RBM utilisent l'apprentissage par contrastes pour ajuster les poids entre les couches, ce qui les rend robustes pour la représentation des données complexes.



J'ai entrainé le modèle RBM pour 100 epochs, puis pour 500 epochs pour finalement l'entrainer pour 1000 epochs. Le but étant de voir si on augmente le nombre d'epochs, on améliore la performance. Ci joint, les résultats obtenus pour les classes '0' et 'A'. (Les RBM ont une configutation [320,200], 320 étant imposé par la dataset Binary AlphaDigits)

Pour la class '0':
Pour 100 epochs, l'erreur de reconstruction est égale à 0.069
Pour 500 epochs, l'erreur de reconstruction est égale à 0.017
Pour 1000 epochs, l'erreur de reconstruction est égale à 0.004

Figure 1: 100 epochs

Figure 2: 500 epochs

Figure 3: 1000 epochs

Pour la class 'A':
Pour 100 epochs, l'erreur de reconstruction est égale à 0.082
Pour 500 epochs, l'erreur de reconstruction est égale à 0.015
Pour 1000 epochs, l'erreur de reconstruction est égale à 0.003

Figure 4: 100 epochs

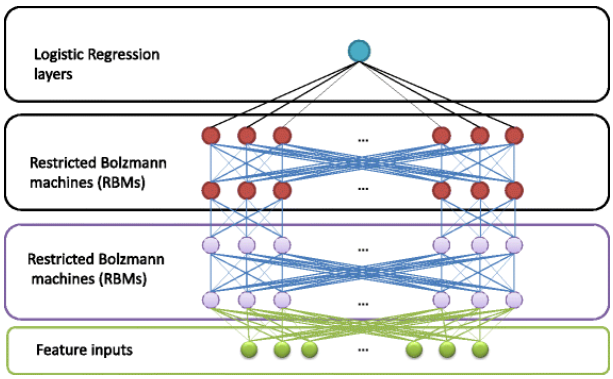
Figure 5: 500 epochs

Figure 6: 1000 epochs

Dans les deux classes testées, on obtient des résultats similaires. On en déduit que les RBM ne sont pas sensibles à la classe (d'ou leur robustesse). De plus, on remarque que plus on augmente le nombre d'epochs, plus l'erreur de reconstruction diminue et donc on obtient des résultats meilleurs.

3.2 DBN

Les Deep Belief Networks (DBN) sont des réseaux de neurones profonds constitués de couches de RBM. Ces réseaux sont des modèles génératifs qui peuvent être utilisés pour l'apprentissage non supervisé et supervisé. Les DBN sont caractérisés par une architecture en couches où les unités dans chaque couche sont connectées à toutes les unités de la couche suivante. L'apprentissage dans un DBN se fait par une phase d'apprentissage non supervisée, où chaque couche est pré-entrainée de manière non supervisée en utilisant l'algorithme de rétropropagation du contraste, suivi d'une phase de fine-tuning supervisée pour ajuster les poids du réseau en utilisant des données étiquetées. Les DBN sont particulièrement efficaces pour extraire des représentations de haut niveau à partir de données brutes, ce qui les rend utiles dans divers domaines tels que la vision par ordinateur, la reconnaissance de la parole et le traitement du langage naturel.

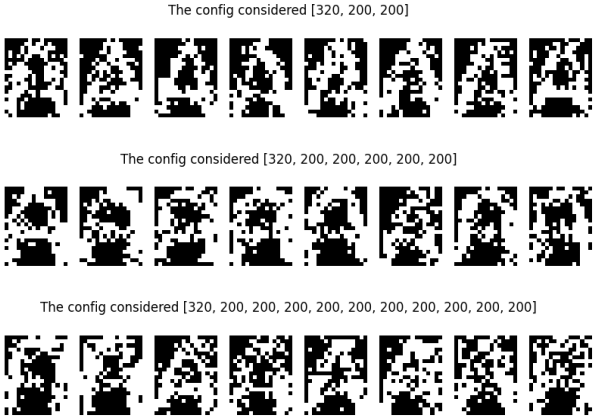


J'ai choisis de travailler ici avec 100 epochs pour pouvoir effectuer des comparaisons significatis dans la suite. (utiliser 1000 epochs va donner un résultat parfait pour tous les essais, chose qui ne peut aider à émettre des conclusions)

Variation du nombre de couches de RBM

Pour la classe 'A', on affichera les erreurs des couches RBM pour étudier l'effet de l'augmentation du nombre de couches (les couches de neurones possèdent 200 nodes par couches):

Pour 2 layers (configuration = [320,200,200]), l'erreur dans les couches est égale à **(0.081, 0.102)**
Pour 5 layers (configuration = [320,200,200,200,200,200]), l'erreur dans les couches est égale à **(0.81, 0.103, 0.111, 0.117, 0.112)**
Pour 10 layers (configuration = [320,200,200,200,200,200,200,200,200,200,200]), l'erreur dans les couches est égale à **(0.081, 0.101, 0.111, 0.111, 0.114, 0.114, 0.114, 0.113, 0.115, 0.113)**

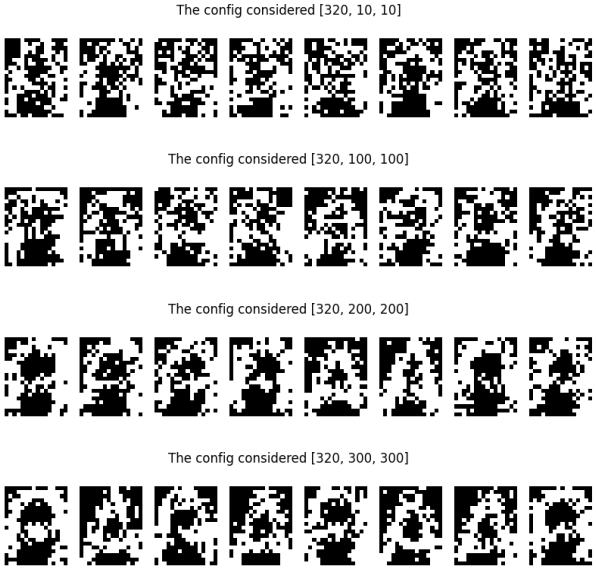


On remarque que pour la configuration de 10 couches, l'entrainement possède une certaine stabilité significative par rapport aux autres configurations. L'erreur ne varie pas beaucoup en passant d'une couche à une autre, chose qui pourrait etre souhaité pour rendre son modèle plus robuste.

Variation du nombre de neurones

Pour la classe 'A', on fera une variation dans le nombre de nodes dans les couches de RBM (on travaillera avec deux couches):

Pour 10 neurones, l'erreur de reconstruction dans les deux couches est égale à **(0.180, 0.074)**
Pour 100 neurones, l'erreur de reconstruction dans les deux couches est égale à **(0.109, 0.115)**
Pour 200 neurones, l'erreur de reconstruction dans les deux couches est égale à **(0.081, 0.100)**
Pour 300 neurones, l'erreur de reconstruction dans les deux couches est égale à **(0.063, 0.077)**



On remarque qu'en augmentant le nombre de neurones, la performance s'améliore (l'erreur dans la 1ère couche diminue, suivi par la 2ème couche). La génération s'améliore, chose qui peut bien etre observé dans les figures ci dessus.

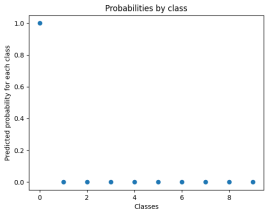
4 Etude sur MNIST

4.1 DNN pré-entrainé

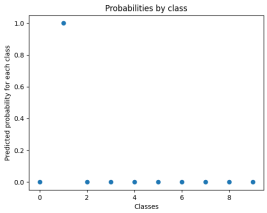
Le DNN est une architecture puissante qui peut être utilisée pour résoudre une grande variété de tâches d'apprentissage automatique, notamment la classification d'images, la prédiction de séries chronologiques et la modélisation de langage.

Dans ce projet, le DNN utilise une architecture de type Deep Belief Network (DBN) pour effectuer un pré-entraînement des couches cachées, suivi d'un fine-tuning pour ajuster les poids du réseau sur les données cibles. L'architecture du DNN est définie par une configuration utilisateur spécifiant le nombre de neurones dans chaque couche cachée, ainsi que la dimension de la couche de sortie. Une fois le DNN entraîné, il peut être utilisé pour prédire les étiquettes de données de test. (Dans ce projet, j'ai définis le DNN comme étant un DBN avec une couche de RBM de classification en plus)

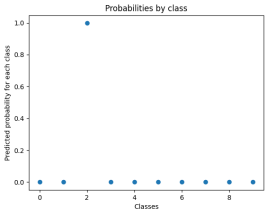
J'ai alors utilisé des DBN préentrainés sur la dataset MNIST, puis en faisant une backward propagation, je fine-tunais les DNN. J'ai utilisé 100 epochs pour cette partie pour entrainer les DBN puis 100 epochs pour entrainer les DNN. Finalement, j'ai visualisé les probabilités prédites pour chaque classe pour voir la robustesse du modèle en ayant utilisé des modèles pré-entrainés.



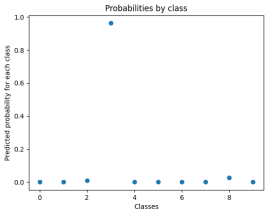
(a) class 0 with error rate equal to 6.06%



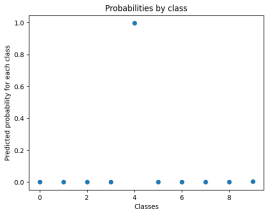
(b) class 1 with error rate equal to 5.9%



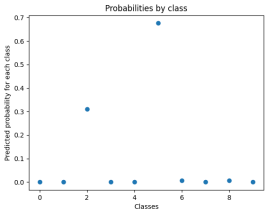
(c) class 2 with error rate equal to 5.8%



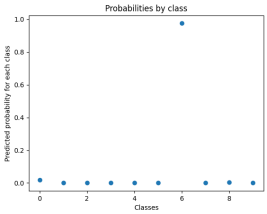
(d) class 3 with error rate equal to 5.79%



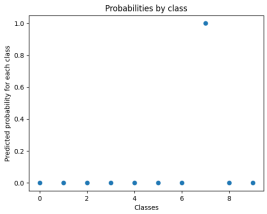
(e) class 4 with error rate equal to 6.18%



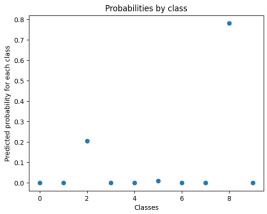
(f) class 5 with error rate equal to 5.83%



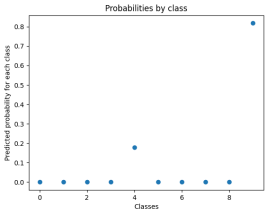
(g) class 6 with error rate equal to 5.86%



(h) class 7 with error rate equal to 6.07%



(i) class 8 with error rate equal to 5.78%



(j) class 9 with error rate equal to 5.51%

On peut remarquer que la performance des DNN est très bonne en terme de discrimination des classes vu que le réseau arrive à différencier les classes en donnant une grande probabilité de classification à la bonne classe. Cependant, on peut noter des résultats "moyens" avec les classes '5', '8', '9'. En effet, ce résultat est bien logique vu que ces chiffres sont corrélés avec d'autres chiffres (8 ressemble à 3 par exemple)

4.2 Comparaison: DNN pré-entrainé vs DNN non pré-entrainé

Dans cette partie, j'ai initialisé deux réseaux DNN avec une configuration [,200,200] (La dernière couche est une couche de classification RBM). J'ai appliqué un préentrainement au premier et puis entrainé les deux avec la backward propagation. Le but étant de voir l'erreur de classification pour les deux cas: avec préentrainement ou sans préentrainement.

Voici le résultat obtenu:

Model DNN with pretrained=True for Train dataset:
Error rate ———————— : 4.58%
Model DNN with pretrained=True for Test dataset:
Error rate ———————— : 5.26%
Model DNN with pretrained=False for Train dataset:
Error rate ———————— : 4.74%
Model DNN with pretrained=False for Test dataset:
Error rate ———————— : 5.58%

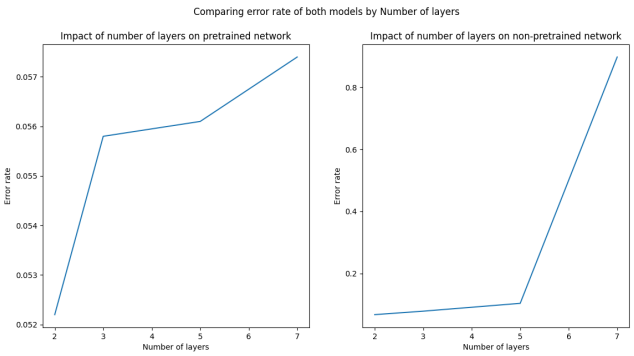
On peut remarquer que l'erreur de classification sur le Train et sur le Test est supérieure pour le cas sans préentrainement, chose qui est bien logique vu que le préentrainement aide le modèle à mieux s'entrainer et mieux se stabiliser.

4.3 Variation du nombre de couches

Dans cette partie, le but est de voir l'impact du nombre de couches de neurones sur l'erreur de classification. J'ai alors entrainé deux DNNs (pré-entrainé et non pré-entrainé) avec différent nombre de couches (la variation du nombre se fait avec la classe DBN).

Voici le résultat obtenu:

Model DNN with pretrained=True for nb_layer= 2:
Error rate ———————— : 5.22%
Model DNN with pretrained=False for nb_layer= 2:
Error rate ———————— : 6.73%
Model DNN with pretrained=True for nb_layer= 3:
Error rate ———————— : 5.58%
Model DNN with pretrained=False for nb_layer= 3:
Error rate ———————— : 7.85%
Model DNN with pretrained=True for nb_layer= 5:
Error rate ———————— : 5.61%
Model DNN with pretrained=False for nb_layer= 5:
Error rate ———————— : 10.37%
Model DNN with pretrained=True for nb_layer= 7:
Error rate ———————— : 5.74%
Model DNN with pretrained=False for nb_layer= 7:
Error rate ———————— : 89.73%



On peut remarquer qu'en augmentant le nombre de couches, le modèle diminue en performance de classification. Cependant, pour le cas du pré-entrainement, on peut remarquer une certaine stabilité qui ne fait pas que l'erreur de classification augmente soudainement. Pour le cas du non pré-entrainement, il est facile

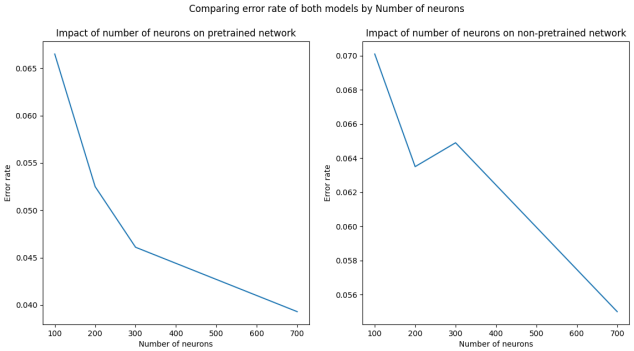
de remarquer que l'erreur de classification augmente exponentiellement à partir du 7ème couche, chose qui montre la non stabilité du modèle.

4.4 Variation du nombre de neurones

Dans cette partie, le but est de voir l'impact du nombre de neurones sur l'erreur de classification. J'ai alors entrainé deux DNNs (pré-entrainé et non pré-entrainé) avec différent nombre de neurones par couche (la variation du nombre de neurones se fait avec la classe DBN). On a utilisé une configuration de [, nb_neurons, nb_neurons] pour les deux DNN.

Voici le résultat obtenu:

Model DNN with pretrained=True for nb_neurons= 100:
Error rate ———————— : 6.65%
Model DNN with pretrained=False for nb_neurons= 100:
Error rate ———————— : 7.01%
Model DNN with pretrained=True for nb_neurons= 200:
Error rate ———————— : 5.25%
Model DNN with pretrained=False for nb_neurons= 200:
Error rate ———————— : 6.35%
Model DNN with pretrained=True for nb_neurons= 300:
Error rate ———————— : 4.61%
Model DNN with pretrained=False for nb_neurons= 300:
Error rate ———————— : 6.49%
Model DNN with pretrained=True for nb_neurons= 700:
Error rate ———————— : 3.93%
Model DNN with pretrained=False for nb_neurons= 700:
Error rate ———————— : 5.50%



On peut remarquer qu'en augmentant le nombre de neurones dans les deux couches du DNN, l'erreur de classification diminue (chose qui est bien souhaité). Cependant, on peut remarquer que l'augmentation du nombre de neurones a un meilleur résultat sur le cas pré-entrainé que sur le cas qui n'est pas pré-entrainé. En effet, pour la configuration [,700,700] le modèle pré-entrainé atteint une erreur de classification de 0.0393 alors que le modèle non pré-entrainé atteint une erreur de classification de 0.055

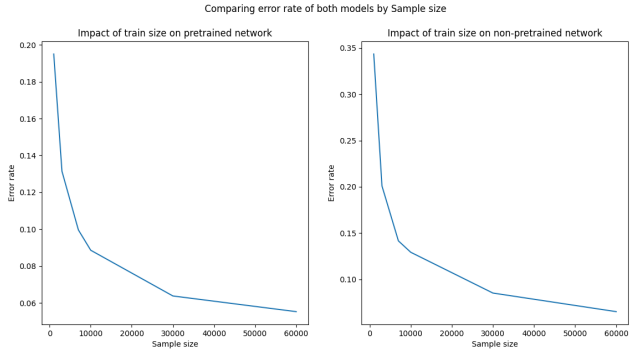
4.5 Variation de la taille de la dataset du train

Dan cette partie, le but est de voir l'impact de la taille du train sur l'erreur de classification. J'ai alors entrainé deux DNNs (pré-entrainé et non pré-entrainé) avec différent taille de train. Le choix de la data du train est fait en utilisant la fonction shuffle de sklearn et en prenant la taille qu'on veut tester.

Voici le résultat obtenu:

Model DNN with pretrained=True for train_size= 1000:
Error rate ———————— : 19.50%
Model DNN with pretrained=False for train_size= 1000:
Error rate ———————— : 34.35%
Model DNN with pretrained=True for train_size= 3000:
Error rate ———————— : 13.14%
Model DNN with pretrained=False for train_size= 3000:

Error rate ———————— : 20.09%
Model DNN with pretrained=True for train_size= 7000:
Error rate ———————— : 9.96%
Model DNN with pretrained=False for train_size= 7000:
Error rate ———————— : 14.15%
Model DNN with pretrained=True for train_size= 10000:
Error rate ———————— : 8.86%
Model DNN with pretrained=False for train_size= 10000:
Error rate ———————— : 12.92%
Model DNN with pretrained=True for train_size= 30000:
Error rate ———————— : 6.38%
Model DNN with pretrained=False for train_size= 30000:
Error rate ———————— : 8.51%
Model DNN with pretrained=True for train_size= 60000:
Error rate ———————— : 5.53%
Model DNN with pretrained=False for train_size= 60000:
Error rate ———————— : 6.50%



On peut remarquer que plus le train size augmente et plus l'erreur de classification diminue. (Donc on a bien intéret de choisir de grandes tailles pour le train). Le modèle pré-entrainé a l'air d'avoir la meme variation que le modèle non pré-entrainé en augmentant le size du train, cependant, on peut remarquer que pour de petites size, l'erreur de classification pour le cas non-pré-entrainé peut atteindre de grandes valeurs ce qui montre une non stabilité face à la taille du train. (le pré-entrainement stabilise le modèle dans le cas où on travaille avec de faible taille de train)

5 Autres Modèles Génératifs

Dans cette partie, on va comparer entre le RBM, le DBN, et le VAE. On va les entrainer sur la dataset Binary AlphaDigits (Le VAE a été entrainé pour 200 epochs) pour ensuite générer les classes de '0' à '9'. Ci joint, les résultats obtenus:


Figure 8: RBM Generation


Figure 9: DBN Generation

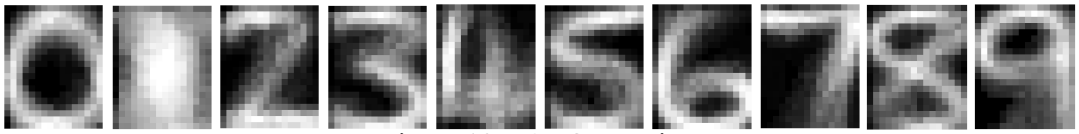


Figure 10: VAE Generation

On remarque que le numéro 1 n'est pas bien généré, ceci est du à la petite taille de la classe '1' dans la Dataset AlphaDigits.

Chose qui est facile à remarquer est cet aspect lisse des VAE dans la génération. Cette caractéristique fait de sorte qu'on obtient des résultats visuellement satisfaisant.

6 Conclusion

En conclusion, notre étude comparative des modèles génératifs et discriminatifs, tels que les Restricted Boltzmann Machines (RBM), les Deep Belief Networks (DBN) et les Variational Autoencoders (VAE), sur les datasets Binary AlphaDigits et MNIST, a mis en lumière plusieurs résultats significatifs.

Tout d'abord, nous avons constaté que les RBM et les DBN ont montré une robustesse et une stabilité dans leur performance, en particulier avec l'augmentation du nombre d'epochs d'entraînement. L'augmentation du nombre de couches pour les DBN améliore sa stabilité lors du l'entrainement, et l'augmentation du nombre de neurones améliore sa performance.

En ce qui concerne les réseaux de neurones profonds (DNN), nous avons observé que le pré-entraînement des couches cachées, suivi d'un fine-tuning, a conduit à une amélioration significative des performances de classification, réduisant ainsi l'erreur de prédiction. De plus, l'augmentation du nombre de neurones et de couches dans les DNN a généralement conduit à une diminution de l'erreur de classification, bien que la stabilité du modèle soit mieux préservée avec le pré-entraînement.

En résumé, notre étude a démontré l'efficacité de différents modèles dans des tâches telles que la classification et la génération de données, tout en mettant en évidence l'importance du pré-entraînement, du dimensionnement du réseau et de la taille de la dataset d'entrainement.