CY Cergy Paris Université

Composante Sciences et Techniques

Master Informatique - 1ère année

Systèmes Intelligents et Communiquants

Mention Informatique et Ingénierie des Systèmes Complexes

Rapport TP3

Intelligence Artificielle +

Auteur :

LABADY Sterley Gilbert

16 mars 2023

1 Introduction

Dans ce rapport, nous présentons notre travail sur la mise en œuvre d'un auto-encodeur pour l'apprentissage de représentations sur la base de données MNIST. L'objectif principal de ce projet est de développer un modèle capable de compresser et de reconstruire des images de chiffres manuscrits, en utilisant un perceptron pour l'encodeur et un perceptron pour le décodeur.

2 Données

Le jeu de données MNIST (Modified National Institute of Standards and Technology) est largement reconnu et utilisé comme un ensemble de données de référence pour l'évaluation des algorithmes d'apprentissage automatique dans le domaine de la reconnaissance d'images. Il contient 70 000 images de chiffres manuscrits (0-9) en niveaux de gris, qui ont été normalisées et centrées. L'ensemble de données est divisé en 60 000 images pour l'entraînement et 10 000 images pour le test.

3 Implémentation

Le modèle est composé de deux parties : un encodeur et un décodeur, chacun étant un perceptron multicouche. Nous analyserons les différentes composantes de cette implémentation.

Architecture de l'auto-encodeur : L'architecture de l'auto-encodeur est définie dans la classe Autoencoder, qui hérite de torch.nn.Module. L'encodeur et le décodeur sont tous deux des perceptrons multicouches avec des fonctions d'activation ReLU. L'encodeur est composé de sept couches linéaires avec des tailles de 784, 512, 256, 128 64, 32, et d, tandis que le décodeur est composé de cinq couches linéaires avec des tailles de d, 32, 64, 128, 256 512, et 784. La sortie du décodeur est passée par une fonction d'activation sigmoïde pour obtenir des valeurs entre 0 et 1.

Chargement des données : Les données MNIST sont chargées à l'aide des fonctions datasets.MNIST et torch.utils.data.DataLoader. Les données sont transformées en tenseurs et normalisées entre 0 et 1. Le jeu de données d'entraînement est utilisé pour entraîner le modèle.

Entraînement de l'auto-encodeur : L'auto-encodeur est entraîné sur 10 époques avec une taille de lot de 128. La fonction de perte utilisée est la perte quadratique moyenne (MSE) et l'optimiseur est l'optimiseur Adam avec un taux d'apprentissage de 0.001. Les images sont aplaties avant d'être passées à travers l'encodeur et le décodeur. Les gradients sont réinitialisés, la sortie est calculée et la perte est calculée par rapport à l'entrée. La rétropropagation est effectuée et les poids sont mis à jour.

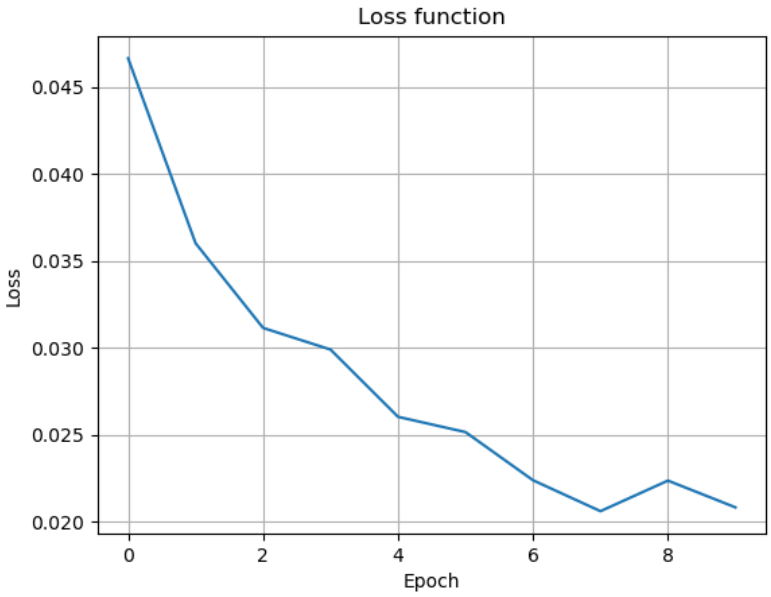
Test de l'auto-encodeur : Le modèle est testé sur le jeu de données de test MNIST. Les images sont aplaties et, si spécifié, un bruit blanc est ajouté aux pixels. Les images bruitées sont passées à travers l'auto-encodeur, puis reconstruites à partir des représentations apprises. Les images originales et reconstruites sont visualisées côte à côte.

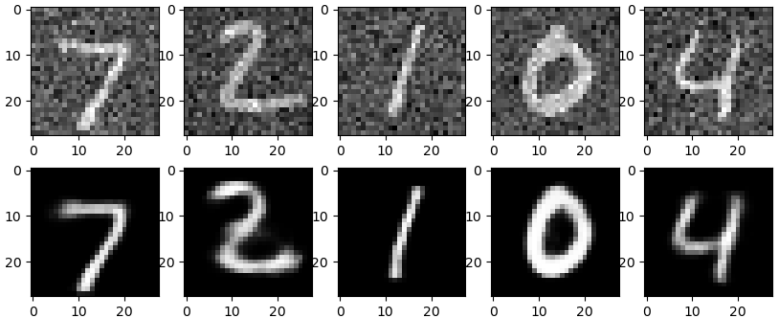
Ajout de bruit blanc : Une fonction permet d'ajouter un bruit blanc à toutes les valeurs des pixels d'un lot d'images. Le bruit est généré en utilisant une distribution normale avec une moyenne de 0,0 et un écart-type de 0,2. Le bruit est ajouté aux images d'entrée avant qu'elles ne soient passées à travers l'auto-encodeur.

4 Analyses

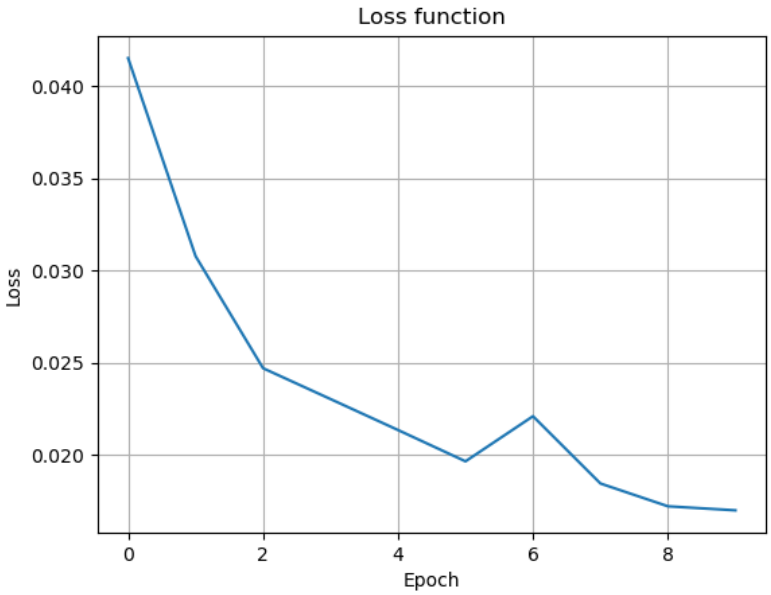
Entraînez le modèle en variant la dimension de la représentation : d = 16, d = 32, d = 64 et comparez vos résultats. Quelle dimension permet la meilleure reconstruction ?

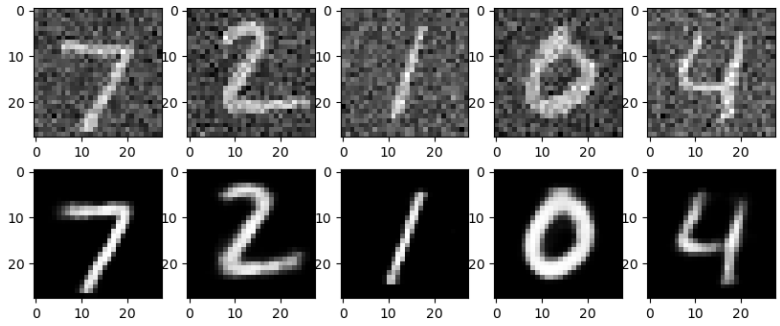
d = 16



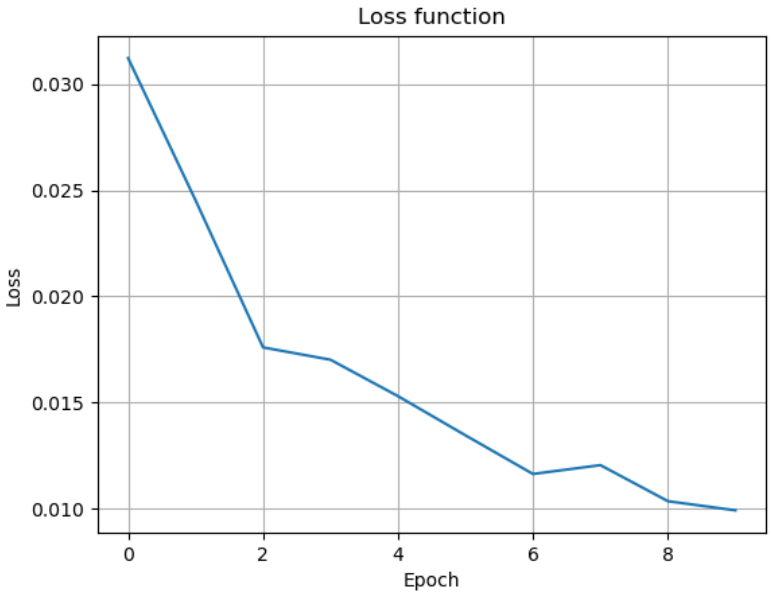


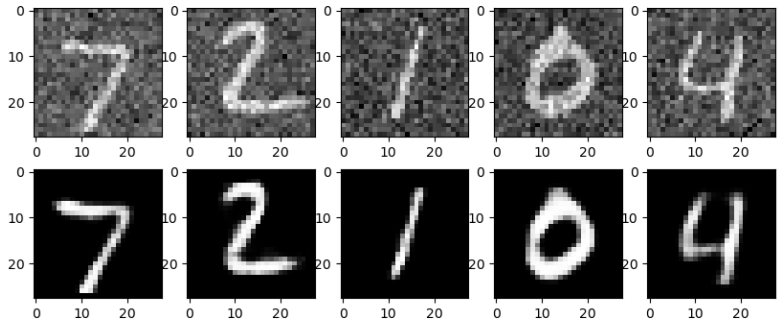
d = 32





d = 64





En observant les résultats de l'entraînement pour les différentes dimensions de représentation (d = 16, d = 32 et d = 64), nous pouvons analyser la qualité de la reconstruction.

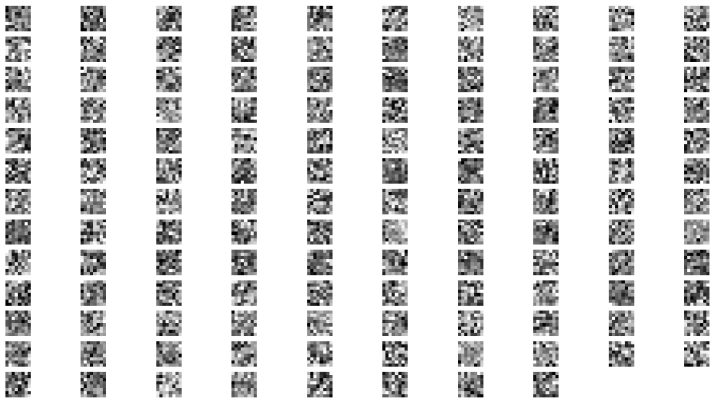
Avec d = 16 : La perte finale (MSE) à la fin de l'entraînement est de 0.0208. Les valeurs de perte diminuent progressivement au cours des époques, suggérant que le modèle apprend à reconstruire les images à partir

de la représentation réduite. Cependant, la faible dimension de la représentation peut limiter la qualité de la reconstruction.

Avec d = 32 : La perte finale (MSE) à la fin de l'entraînement est de 0.0170. Par rapport au cas où d = 16, la qualité de la reconstruction semble s'améliorer. La perte diminue également au fil des époques, ce qui suggère que le modèle continue d'apprendre. Une représentation de dimension plus élevée pourrait permettre de capturer plus d'informations pertinentes pour la reconstruction.

Avec d = 64 : La perte finale (MSE) à la fin de l'entraînement est de 0.0099. C'est la valeur la plus faible parmi les trois configurations testées, ce qui indique que le modèle avec d = 64 a la meilleure qualité de reconstruction. Cela suggère que la représentation de dimension 64 capture suffisamment d'informations pour permettre une meilleure reconstruction des images.

En conclusion, sur la base des résultats de l'entraînement, la meilleure reconstruction est obtenue avec une dimension de représentation d = 64, qui présente la plus faible perte (MSE) à la fin de l'entraînement.



Il se peut que vous ne voyiez pas de motifs clairs correspondant aux chiffres, car l'auto-encodeur apprend à extraire et à reconstruire les caractéristiques générales de l'ensemble des données.

Débruitage

```
cp = 0
sum = 0
with torch.no_grad():
    test_dataset = datasets.MNIST(root='data', train=False, download=True,
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=5,
    for data in test_loader:
        img, _ = data
        imgC = img.view(img.size(0), -1)
        imgN = add_noise(imgC)
        output = autoencoder(imgN)
        debruiting_error = torch.mean((imgC - output) ** 2)
        sum += debruiting_error.item()
        cp += 1
print("MSE debruiting error :", sum/cp)
```

L'erreur de débruitage moyenne est de : 0.053700394934043286

Le but de cette étude est de déterminer si l'auto-encodeur est capable de reconstruire efficacement les images d'origine à partir d'images bruitées et d'analyser l'impact de l'amplitude du bruit sur la qualité du débruitage.

Nous avons calculé l'erreur de débruitage en utilisant l'erreur quadratique moyenne (MSE) pour comparer les images d'origine et les images reconstruites par l'auto-encodeur.

Résultats : Notre expérience a donné une MSE de débruitage de 0.0537. Cette valeur indique que l'auto-encodeur est capable de débruiter les images dans une certaine mesure, mais il existe encore des différences entre les images d'origine et les images reconstruites. Cela suggère que l'auto-encodeur a appris certaines caractéristiques des données MNIST et a réussi à les utiliser pour débruiter les images, bien que la performance ne soit pas parfaite.

5 Questions / Réponses

Quels sont les intérêts d'apprendre une représentation h de nos entrées x ?
R. L'apprentissage d'une représentation h des entrées x offre des avantages en termes de réduction de dimensionnalité, d'extraction de caractéristiques, d'apprentissage non supervisé, de compression des données, de robustesse et de transfert d'apprentissage, qui peuvent être exploités dans diverses applications de l'apprentissage automatique et de l'intelligence artificielle.

Quel est le nombre de paramètres dans le réseau en fonction de la dimension d de la représentation ?
R. Le nombre de paramètres d'un auto-encodeur dépend des dimensions d'entrée (n) et de représentation (d). En ajustant d, on observe la variation du nombre de paramètres.

Supposons que nous souhaitions utiliser notre auto-encodeur pour compresser et envoyer des images. En supposant que le destinataire de l'image ait à sa disposition le décodeur, on peut se contenter de lui envoyer la représentation h correspondant à chaque image. Quelle sera la taille relative de notre fichier par rapport à l'image complète de taille 28x28 ?
R. Pour déterminer la taille relative de la représentation h par rapport à l'image complète de taille 28x28, on doit considérer la taille en octets de chaque élément et la dimension d de la représentation. La taille d'une image 28x28 est de 28 * 28 = 784 pixels. En supposant que chaque pixel est représenté par un seul octet, la taille de l'image complète serait de 784 octets. Si la dimension de la représentation h est d, la taille de la représentation h serait d * 1 (supposant que chaque élément de h est représenté par un seul octet). La taille relative de notre fichier par rapport à l'image complète serait donc (d * 1) / (784 * 1) = d / 784. Cette valeur varie en fonction de la dimension d choisie pour la représentation. Plus d est faible, plus la compression est importante.