11/01/2024

Conditional VAE

Advanced Neural Net

Ramiqi Andi

Table des matières

1. Introduction

Les Variational Autoencoders conditionnels (cVAE) sont une avancée dans le domaine des modèles génératifs, permettant une génération de données ciblée et contrôlée. Au cœur de l'intelligence artificielle, les modèles génératifs visent à apprendre la distribution de données pour générer de nouveaux échantillons indiscernables des réels. Parmi eux, le cVAE se distingue par sa capacité à conditionner la génération sur des informations supplémentaires, ouvrant la voie à des applications personnalisées et diversifiées allant de la synthèse d'images à la création de textes adaptés. Cette introduction plonge dans les mécanismes et l'impact des cVAE, soulignant leur rôle dans l'avancement des frontières de l'apprentissage automatique et de la création générative.

Le but de ce rapport est de servir de guide approfondi sur les Variational Autoencoders conditionnels (cVAE), en s'appuyant sur l'exemple concret du jeu de données MNIST. Nous explorerons comment un réseau de neurones peut être entraîné pour non seulement reconstruire des données, mais aussi générer de nouvelles instances conditionnées de manière significative. Pour saisir pleinement le potentiel et le fonctionnement des cVAE, il est essentiel de comprendre d'abord les principes fondamentaux des autoencoders et des VAE. Ces derniers constituent la base sur laquelle les cVAE étendent leurs capacités, en introduisant une structure permettant de manipuler et de contrôler les données générées. Ce rapport détaillera ainsi les composants clés, les mécanismes sous-jacents et les applications variées des cVAE, illustrant comment ces modèles avancés façonnent l'avenir des systèmes génératifs.

2. Autoencoders et VAE

Autoencoders (AE)

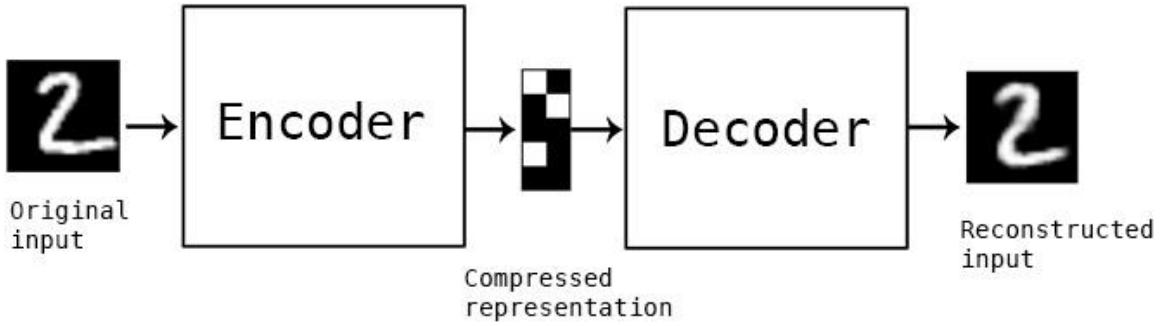
Définition et Structure :

Les autoencoders sont une catégorie de réseaux de neurones utilisés pour l'apprentissage non supervisé. Ils visent à apprendre une représentation (encodage) compressée et efficace des données d'entrée, généralement dans le but de réduire la dimensionnalité ou de retirer le bruit des données. Un autoencoder typique se compose de deux parties principales :

Encodeur (Encoder) : L'encodeur prend une entrée « x » et la transforme en une représentation codée « z », souvent de dimension réduite. Ce processus implique généralement une ou plusieurs couches d'un réseau de neurones qui compriment progressivement les données. L'objectif est de capturer les aspects essentiels de l'entrée dans un espace plus compact. Au lieu de penser à l'espace latent « z » comme à une représentation codée fixe de « x », on peut le considérer comme une variable aléatoire ayant sa propre distribution « p(z) ». L'encodeur définit une distribution conditionnelle « p(z|x) », qui capture la manière dont les données d'entrée sont mappées à l'espace latent.

Espace codé (Coded space) : L'espace codé « z » est une représentation intermédiaire des données. Dans un AE classique, « z » est une représentation déterministe directe de l'entrée « x ».

Décodeur (Decoder) : Le décodeur prend l'espace codé « z » et tente de reconstruire l'entrée originale, notée « x′ ». Comme l'encodeur, le décodeur est souvent constitué de couches de réseau de neurones, mais dans l'ordre inverse, décompressant progressivement les données pour atteindre la dimensionnalité originale. Le décodeur, d'un point de vue probabiliste, définit une distribution conditionnelle « p(x|z) ». Cette distribution modélise comment les données sont reconstruites à partir de l'espace latent. L'objectif est de rendre « p(x|z) » aussi proche que possible de la distribution réelle des données p(x).



Loss function:

$$L = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Objectif et Applications :

Le but est de minimiser la différence entre les données d'entrée et les données reconstruites qu'on appellera « **Reconstruction Loss** », souvent mesurée par une fonction de perte comme l'erreur quadratique moyenne. Les autoencoders sont utilisés dans la réduction de dimensionnalité, la détection d'anomalies, et comme prélude aux architectures plus complexes comme les VAE et cVAE.

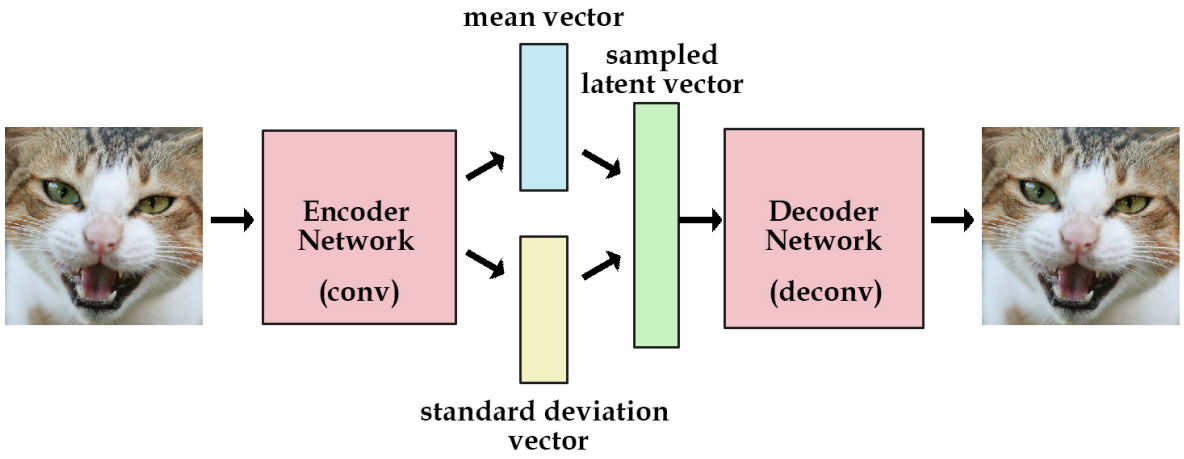
Fonction de Perte comme Probabilité Logarithmique

Maximisation de la Log-vraisemblance : La fonction de perte utilisée dans les autoencoders peut être interprétée comme la négation de la log-vraisemblance des données reconstruites sous le modèle. Minimiser cette perte revient à maximiser la probabilité que le modèle attribue aux données d'entrée, rendant les reconstructions « x′ » plus probables sous la distribution estimée « p(x|z) ». Dans les autoencoders standards, l'espace latent n'est pas structuré de manière probabiliste et ne modélise pas explicitement la variance des données. Cela signifie que bien qu'il puisse réduire la dimensionnalité, il ne capture pas nécessairement la distribution sous-jacente complexe des données.

Pour remédier à certaines de ces limites, les recherches se sont orientées vers des variantes plus sophistiquées comme les Variational Autoencoders (VAE) et les Conditional Variational Autoencoders (cVAE), qui introduisent une structure probabiliste dans l'espace latent et permettent un contrôle plus fin sur la génération de données. Ces modèles tentent de surmonter la rigidité des autoencoders classiques en offrant une meilleure généralisation et en capturant plus fidèlement la distribution des données.

Variational Autoencoders (VAE)

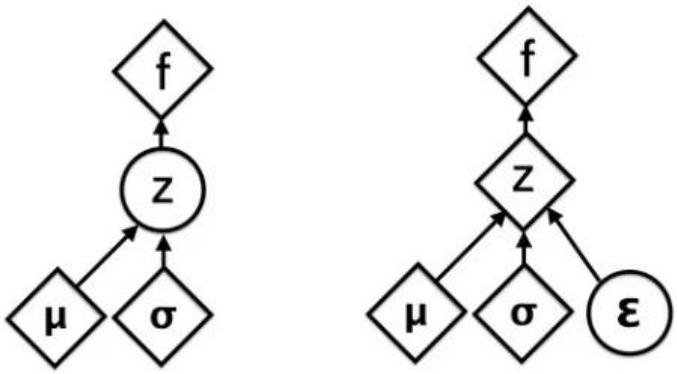
Les Variational Autoencoders sont une extension des autoencoders classiques qui introduit une couche probabiliste. Au lieu de coder une entrée en un point fixe, le VAE encode les entrées en distributions de probabilité, typiquement des distributions gaussiennes, définissant ainsi un espace latent probabiliste.



Structure et Fonctionnement :

Encodeur probabiliste : L'encodeur dans un VAE ne produit pas une valeur unique « z » mais deux vecteurs représentant les moyennes « μ » et les écarts-types « σ » d'une distribution gaussienne. Cela convertit l'espace latent en un espace de distributions probabilistes.

Reparamétrisation : Pour permettre le backpropagation à travers un processus aléatoire, les VAE utilisent une astuce de reparamétrisation. Au lieu d'échantillonner « z » directement de la distribution apprise, on échantillonne d'abord « ϵ » d'une distribution normale standard et on calcule « $z=\mu+\sigma\cdot\epsilon$ ». Cela permet de calculer des gradients de manière efficace tout en intégrant l'aléa nécessaire.

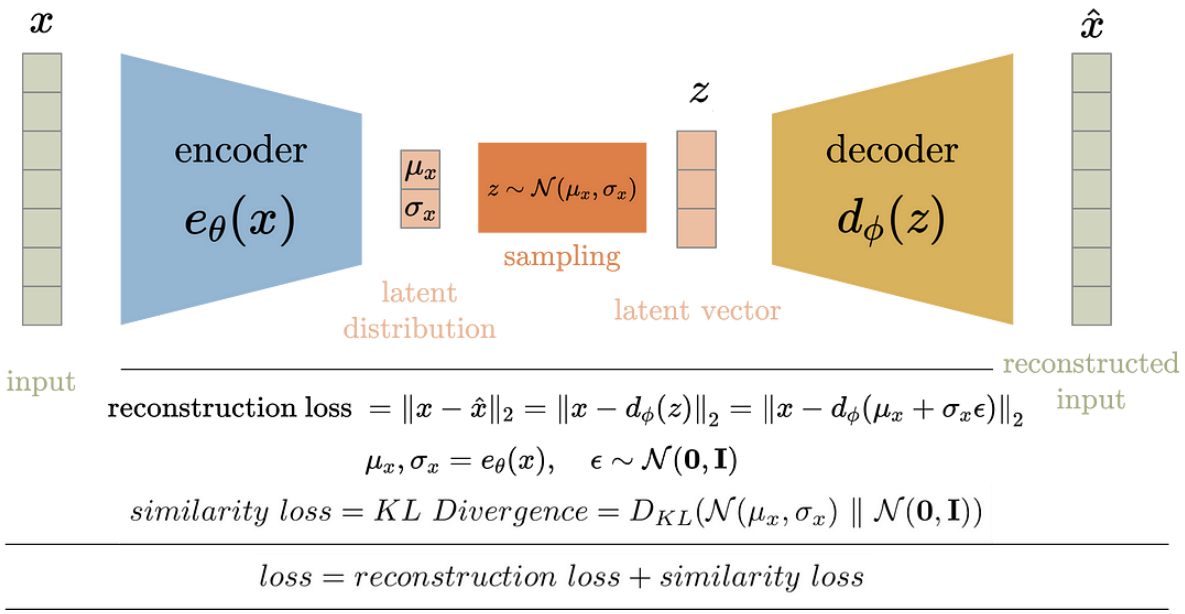


Original Reparametrized

$$\text{Reparametrization Trick : } z = \mu + \sigma * \epsilon; \quad \epsilon \sim \mathcal{N}(0,1)$$

Décodeur probabiliste : Semblable à l'autoencoder standard, le décodeur dans un VAE reconstruit l'entrée à partir de « z ». Cependant, puisque « z » est échantillonné d'une distribution, le décodeur apprend à gérer et à décoder efficacement des variations, ce qui conduit à une meilleure généralisation et capacité de génération.

Fonction de perte : La fonction de perte dans un VAE est la somme de deux termes : un terme de reconstruction, qui pousse la reconstruction à ressembler à l'entrée originale, et une divergence KL, qui pousse la distribution latente apprise à se rapprocher d'une distribution prior (souvent gaussienne standard). La divergence KL agit comme une régularisation et mesure à quel point la distribution apprise « p(z|x) » s'écarte d'une distribution prior « p(z) », souvent choisie pour être une gaussienne standard. Minimiser cette divergence pousse l'espace latent à suivre une structure organisée et continuellement variée, ce qui est bénéfique pour la génération et l'interpolation.



Idéalement, « x » et « x' » devraient être identiques.

Objectif :

Elbo : 'ELBO, ou "Evidence Lower BOund", est un concept central dans l'entraînement des Variational Autoencoders (VAE). Il est utilisé comme fonction objective lors de l'optimisation des VAE. L'ELBO est essentiellement une limite inférieure sur la log-vraisemblance des données observées et vise à rapprocher la distribution approximative de l'espace latent de la vraie distribution sous-jacente. L'objectif ultime d'un modèle génératif comme le VAE est de maximiser la log-vraisemblance des données observées « log(p(x)) ». Cependant, calculer cela directement est souvent intractable en raison de l'intégration sur l'espace latent complexe.

L'ELBO L(x) peut être décomposé en deux termes :

$$\mathbf{L}(\boldsymbol{x}) = \mathbf{E}\boldsymbol{q}(\boldsymbol{z} \mid \boldsymbol{x})[\log(\boldsymbol{p}(\boldsymbol{x} \mid \boldsymbol{z}))] - \boldsymbol{DKL}(\boldsymbol{q}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel \boldsymbol{p}(\boldsymbol{z}))$$

Où :

- « L(x) » est l'elbo
- « Eq(z|x)[logp(x|z)] » est l'espérance de la log-vraisemblance de reconstruction. Ce terme mesure à quel point le décodeur reconstruit bien les données d'entrée à partir de l'échantillon de l'espace latent.
- « DKL(q(z|x)||p(z)) » est la divergence KL entre la distribution approximative postérieure « q(z|x) » et la distribution prior « p(z) » de l'espace latent. Ce terme agit comme une régularisation, encourageant la distribution approximative postérieure à ressembler à la distribution prior.

Durant l'entraînement d'un VAE, on ajuste les paramètres du modèle (les poids de l'encodeur et du décodeur par exemple) pour maximiser l'ELBO. Cela équivaut à maximiser la log-vraisemblance de reconstruction tout en minimisant la divergence KL.

Le terme de reconstruction encourage des reconstructions précises, tandis que le terme de divergence KL pousse l'espace latent à avoir de bonnes propriétés probabilistes et éviter le surajustement. L'ELBO est un élément clé dans l'entraînement des VAE, fournissant un moyen pratique et efficace de maximiser la vraisemblance des données en optimisant une limite inférieure. En maximisant l'ELBO, le VAE apprend à coder les données d'entrée dans un espace latent structuré tout en étant capable de reconstruire des données similaires, soutenant ainsi à la fois l'apprentissage efficace et la génération de données.

3. Conditional Variational Autoencoders (cVAE)

Les cVAE sont une variante des VAE qui permettent de conditionner la génération des données sur certaines informations supplémentaires. Ce conditionnement se fait en intégrant des labels ou d'autres données d'entrée à la fois dans l'encodeur et le décodeur. Si nous voulions générer des données spécifiques avec un VAE, il y aurait un problème, car il n'y a aucun contrôle sur le processus de génération.

Pour cela, nous introduisons ainsi une condition, cela signifie que l'encodeur produit « μ » et « σ » en fonction non seulement de « x » mais aussi de la condition « c », et le décodeur génère des sorties basées sur « z » et « c ». Le décodeur prend en compte à la fois l'échantillon latent « z » et la condition « c » pour reconstruire l'entrée. Cela permet au modèle de générer des données spécifiques à une condition donnée, améliorant la flexibilité et l'applicabilité du modèle.

Pour ce travail, nous assumons que la **condition « c » est indépendante** de z, la structure du cVAE reste similaire, mais l'interaction entre la condition et l'espace latent et modifiée. Dans l'encodage, l'entrée « x » et la condition « c » sont toujours fournies au cVAE, mais si « c » est indépendant de « z », cela signifie que pendant l'encodage, « c » ne contribue pas directement à la formation de la distribution de l'espace latent « z ». L'encodeur produit ainsi toujours des paramètres « μ » et « σ » pour définir une distribution gaussienne. L'espace latent reste une représentation compressée et probabiliste de x et La reparamétrisation se déroule comme d'habitude pour permettre la backpropagation.

C'est dans la phase du décodeur que l'indépendance entre c et z devient plus apparente, Le décodeur reçoit l'échantillon latent « z » et la condition « c ». Même si « c » n'a pas influencé la formation de « z », elle peut être utilisée pour modifier la sortie « x′ »

Notre fonction objective devient ainsi :

$$L(x, y) = Eq\,(z|x, c) \log p(x|c, c) - DKL(q(z|x, c)||p(z))$$

La « Reconstruction Loss » reste une comparaison entre « x » et « x′ », la sortie du décodeur. Ce terme peut prendre en compte « c », favorisant des reconstructions qui sont fidèles non seulement à « x » mais aussi appropriées à la condition. La divergence KL mesure la différence entre la distribution postérieure approximative

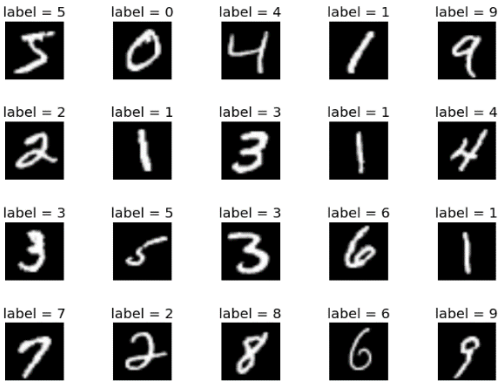
« q(z|x,c) » et la prior « p(z) ». Dans un cadre où « c » est indépendant de « z », ce terme encourage toujours une structure régulière dans l'espace latent « z », mais il n'intègre pas directement « c » dans la comparaison, puisque « p(z) » est indépendant de « c ».

Si « c » est indépendant de « z », cela implique souvent une séparation plus nette des rôles : « z » capture la structure essentielle des données, tandis que « c » guide la spécificité ou la variation dans les données reconstruites ou générées. Cette séparation peut rendre le cVAE plus adaptable et modulaire, car les conditions peuvent être modifiées ou étendues sans perturber l'architecture centrale du réseau ou l'espace latent.

4. Implémentation du code :

Utilisation du Dataset MNIST

Le dataset MNIST est une collection d'images de chiffres écrits à la main, allant de 0 à 9. Chaque image est en noir et blanc, de dimension 28x28 pixels. C'est un standard pour évaluer les modèles d'apprentissage automatique sur des tâches de reconnaissance de chiffres. Les données MNIST sont divisées en un ensemble d'entraînement et un ensemble de test, typiquement utilisés pour former et évaluer les modèles.



Mise en Place d'un MLP Classique

Pour notre projet, nous avons implémenté un modèle de Perceptron Multicouche (MLP) en utilisant PyTorch, une bibliothèque réputée pour l'apprentissage automatique. Ce modèle est un réseau de neurones avec plusieurs couches qui se connectent les unes aux autres. Il est flexible, car on peut choisir le nombre et la taille de ces couches pour s'adapter à différentes tâches. Le MLP a été choisi pour sa capacité à identifier des patterns complexes dans les données, en utilisant des couches qui combinent des calculs linéaires et non-linéaires. La structure du MLP et la manière dont les données y circulent sont définies par la méthode forward. Cette construction permet au modèle d'apprendre à partir du jeu de données MNIST, avec

l'objectif de bien classer les chiffres manuscrits. Ainsi, le MLP sert de fondation solide pour nos expérimentations et analyses ultérieures.

cVAE et Loss Fonction :

Pour implémenter une classe Conditional Variational Autoencoder (cVAE) et sa fonction de perte adaptée, nous avons intégré la notion de conditionnalité dans l'architecture VAE traditionnelle. Le cVAE prend en compte des informations supplémentaires pour diriger la génération de données. Ce modèle est construit sur la structure du VAE standard, avec l'encodeur transformant les entrées en une distribution de l'espace latent, et le décodeur reconstruisant les données à partir de cet espace. Cependant, tant l'encodeur que le décodeur dans le cVAE prennent également en compte les conditions d'entrée, permettant ainsi une génération de données plus ciblée et contrôlée.[1]

La fonction de perte du cVAE incorpore un terme supplémentaire, le coefficient « beta » dans le calcul de la divergence KL. Ce terme permet d'ajuster l'importance relative de la divergence KL par rapport à l'erreur de reconstruction dans la fonction de perte. En augmentant ou diminuant « beta » on peut contrôler l'accent mis sur la régularisation de l'espace latent versus la fidélité de la reconstruction. Un « beta » élevé encourage un espace latent plus régulier et moins sensible aux spécificités des données d'entraînement, facilitant la génération de nouvelles instances. Inversement, un « beta » plus faible favorise des reconstructions plus précises aux dépens d'un espace latent potentiellement moins structuré.

Dans la pratique, la méthode `forward` de la classe cVAE guide les données et les conditions à travers le réseau, assurant que les influences conditionnelles sont bien intégrées dans le processus de génération et de reconstruction. La mise en place de cette classe cVAE et de sa fonction de perte ajustable avec le paramètre « beta » est cruciale pour explorer les capacités de génération contrôlée des modèles et pour équilibrer entre la qualité des reconstructions et la diversité des générations

Entrainement du model

Pour mettre en place l'entraînement du Conditional Variational Autoencoder (cVAE) avec une attention particulière au paramètre « beta » dans la fonction de perte, une série d'étapes structurées et systématiques a été adoptée. L'entraînement du modèle vise à ajuster ses paramètres de manière à minimiser la fonction de perte tout en prenant en compte la qualité des reconstructions et la structure de l'espace latent.

Tout d'abord, les données du dataset MNIST sont préparées et chargées avec les conditions appropriées. Le modèle est ensuite initialisé avec des spécifications pour l'encodeur, le décodeur, et les dimensions de l'espace latent. Avant de commencer l'entraînement, un optimiseur est sélectionné, dans notre cas il s'agira d'Adam qui a été préféré pour sa performance et sa facilité d'utilisation, avec un taux d'apprentissage déterminé.

[1] Pour être plus précis, étant donné que la condition est indépendante de z, c'est surtout le decodeur qui prendra en compte de la condition

L'entraînement se déroule sur plusieurs époques. À chaque époque, le jeu de données est parcouru batch par batch. Pour chaque batch, les étapes suivantes sont effectuées :

1. Propagation Avant : Les données et conditions sont passées à travers le cVAE, produisant une reconstruction et des paramètres de l'espace latent.
2. Calcul de la Perte : La fonction de perte, incluant l'erreur de reconstruction et le terme ajusté de divergence KL avec le coefficient « beta », est calculée. Cette perte reflète la qualité de la reconstruction ainsi que la régularité de l'espace latent.
3. Rétropropagation : Le gradient de la perte est calculé par rapport aux paramètres du modèle, et une étape d'optimisation est effectuée pour ajuster ces paramètres.

Le suivi des performances du modèle se fait en enregistrant la perte moyenne sur l'ensemble d'entraînement et, optionnellement, sur un ensemble de validation pour surveiller la généralisation du modèle.

Après l'entraînement, le modèle est évalué sur l'ensemble de test pour vérifier sa capacité à généraliser à de nouvelles données et à produire des reconstructions de qualité ainsi que des échantillons variés et cohérents. Cette phase d'entraînement est cruciale, car elle affine le modèle pour qu'il capture efficacement les complexités des données tout en répondant aux contraintes imposées par le conditionnement et la structure de l'espace latent.

Evaluation

Dans la phase d'évaluation, notre objectif est de mesurer l'efficacité et la précision de notre modèle cVAE, formé sur le jeu de données MNIST. L'évaluation se concentre principalement sur deux aspects : la qualité de la reconstruction des données et la capacité du modèle à générer de nouvelles instances significatives en fonction des conditions spécifiées.

Pour évaluer la qualité de la reconstruction, nous avons utilisé l'ensemble de test MNIST pour comparer les images originales avec leurs versions reconstruites par le modèle. Cela implique de calculer la « Reconstruction Loss » avec la cross-entropie binaire sur cet ensemble de test. Une faible perte de reconstruction indique que le modèle peut fidèlement reconstruire les chiffres manuscrits, capturant ainsi les nuances et les détails des données originales.

Parallèlement à l'évaluation de la reconstruction, nous avons exploré la capacité du modèle à générer de nouvelles données. Cela a été réalisé en échantillonnant des vecteurs de l'espace latent et en les passant par le décodeur, tout en conditionnant ces générations sur des étiquettes de chiffres spécifiques pour voir si le modèle pouvait générer des chiffres correspondant à ces étiquettes. La variété et la fidélité des chiffres générés offrent une mesure de la capacité du modèle à comprendre et à manipuler l'espace des données MNIST.

La diversité des données générées a également été un point d'intérêt : un modèle efficace devrait être capable de générer une variété de représentations pour chaque chiffre, reflétant la diversité trouvée dans l'ensemble de données réel. Pour évaluer cela, nous avons généré plusieurs instances de chaque chiffre sous des conditions identiques et examiné la variation entre les résultats.

En conclusion, l'évaluation a joué un rôle crucial dans la validation de notre modèle cVAE. Elle a non seulement confirmé la qualité de la reconstruction et de la génération des données, mais a également fourni des « insights » sur la manière dont le modèle comprend l'espace des données. Les résultats de cette phase ont directement informé les itérations ultérieures du modèle, guidant les ajustements et les améliorations pour améliorer les performances.

5. Résultats

```
Epoch [1/20], Loss: 153.1057
Epoch [2/20], Loss: 116.2847
Epoch [3/20], Loss: 108.8701
Epoch [4/20], Loss: 105.1486
Epoch [5/20], Loss: 102.9889
Epoch [6/20], Loss: 101.4426
Epoch [7/20], Loss: 100.2512
Epoch [8/20], Loss: 99.2703
Epoch [9/20], Loss: 98.5103
Epoch [10/20], Loss: 97.9089
Epoch [11/20], Loss: 97.4340
Epoch [12/20], Loss: 97.0101
Epoch [13/20], Loss: 96.6968
Epoch [14/20], Loss: 96.3112
Epoch [15/20], Loss: 96.0433
Epoch [16/20], Loss: 95.7688
Epoch [17/20], Loss: 95.5569
Epoch [18/20], Loss: 95.3249
Epoch [19/20], Loss: 95.0664
Epoch [20/20], Loss: 94.8791
```

Le Conditional Variational Autoencoder (cVAE) a démontré une amélioration constante au cours de l'entraînement avec une perte décroissante de 153.16 à 94.88 sur 20 époques. Cette réduction continue indique un bon apprentissage. Au début, la perte a baissé rapidement, montrant que le modèle a vite capté les grandes caractéristiques des données. Avec le temps, la baisse de la perte est devenue plus lente, ce qui est normal car le modèle ajuste finement ses paramètres. Il n'y a pas de stagnation dans la réduction de la perte, ce qui laisse penser que le modèle pourrait s'améliorer encore avec plus d'entraînement.

```
Evaluation completed :
Average Loss: 90.3619
Hyperparameters:
learning_rate: 0.001
batch_size: 64
epochs: 20
beta: 1
```

L'évaluation du Conditional Variational Autoencoder (cVAE) affiche une performance notable avec une perte moyenne de 90.3619 après 20 époques d'entraînement. Les hyperparamètres choisis sont les suivants : un taux d'apprentissage de 0.001, une taille de lot de 64 et un coefficient beta de 1. Ces paramètres ont permis au modèle de se développer efficacement.

En examinant les images fournies, on observe que les données réelles sont bien représentées, avec des chiffres nettement reconnaissables. Les images générées par le modèle montrent une ressemblance raisonnable avec les données réelles, bien qu'elles apparaissent un peu plus floues et moins précises. Cela indique que, tandis

que le modèle a appris à capturer la structure générale des chiffres, il pourrait encore y avoir une marge d'amélioration dans la netteté et le détail des reconstructions.

Lorsque le « beta » est augmenté, il ne semble pas y avoir de différence significative, si ce n'est que les chiffres « 8 » sont cette fois tous générés correctement. Lorsque le « beta » est plus élevé, cela confirme l'amélioration de génération de nouvelles instances, au détriment de l'espace latent, mais la diversité est ainsi perdue et les chiffres se ressemblent en très grande partie. La perte quant à elle est légèrement supérieure.

6. Difficultés rencontrées

Le principal problème a été de réaliser correctement le réseau du model cVAE. En effet, les tailles des inputs en entrées et sorties différaient, le model utilise un MLP pour générer la partie « encode » ainsi que « decode », et il m'a fallu un certain moment avant de voir que les paramètres des « hidden_layers » n'étaient pas correctement attribué pour le decodeur, provoquant ainsi une erreur dans la sortie d'un résultat du model.

7. Sources

ANWAR, Aqeel, 2021. Difference between AutoEncoder (AE) and Variational AutoEncoder (VAE). *Medium* [en ligne]. 4 novembre 2021. Disponible à l'adresse : https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2 [Consulté le 3 janvier 2024].

How can you compress data or even generate data from random values? That is what Autoencoder and Variational Autoencoder are.

Autoencoders, [sans date]. [en ligne]. Disponible à l'adresse : https://fr.mathworks.com/discovery/autoencoder.html [Consulté le 3 janvier 2024].

Découvrez comment utiliser les autoencoders dans diverses applications, telles que la détection d'anomalies, la génération de texte et le débruitage d'images. Les ressources comprennent des vidéos, des exemples et de la documentation.

Building Autoencoders in Keras, [sans date]. [en ligne]. Disponible à l'adresse : https://blog.keras.io/building-autoencoders-in-keras.html [Consulté le 3 janvier 2024].

CODEEMPORIUM (réal.), 2019. *Variational Autoencoders - EXPLAINED!* [en ligne]. 17 juin 2019. Disponible à l'adresse : https://www.youtube.com/watch?v=fcvYpzHmhvA [Consulté le 3 janvier 2024].

Generative Modeling with Variational Autoencoders (VAEs)

Conditional Variational Autoencoder: Intuition and Implementation - Agustinus Kristiadi, [sans date]. [en ligne]. Disponible à l'adresse : https://agustinus.kristia.de/techblog/2016/12/17/conditional-vae/ [Consulté le 3 janvier 2024].

An extension to Variational Autoencoder (VAE), Conditional Variational Autoencoder (CVAE) enables us to learn a conditional distribution of our data, which makes VAE more expressive and applicable to many interesting things.

Getting Started with Variational Autoencoders using PyTorch, 2020. *DebuggerCafe* [en ligne]. Disponible à l'adresse : https://debuggercafe.com/getting-started-with-variational-autoencoders-using-pytorch/ [Consulté le 3 janvier 2024].

Get started with the concept of variational autoencoders in deep learning in PyTorch to construct MNIST images.

PHD, Matthew Stewart, 2023. Comprehensive Introduction to Autoencoders. *Medium* [en ligne]. 10 février 2023. Disponible à l'adresse : https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368 [Consulté le 3 janvier 2024].