Rapport de Projet E4
Axe recherche
2023-2024

Développement d’un modèle de diffusion pour la génération de données synthétiques de type image

Réalisé par

Tiéba BAMBA
Meryle BAZEBI
Léo CALTEAU
El Hadji Bara CAMARA
Nicolas HAMEAU
Laëtitia LEROI

Supervisé par

Ségolène COMBETTES
Adrien UGON

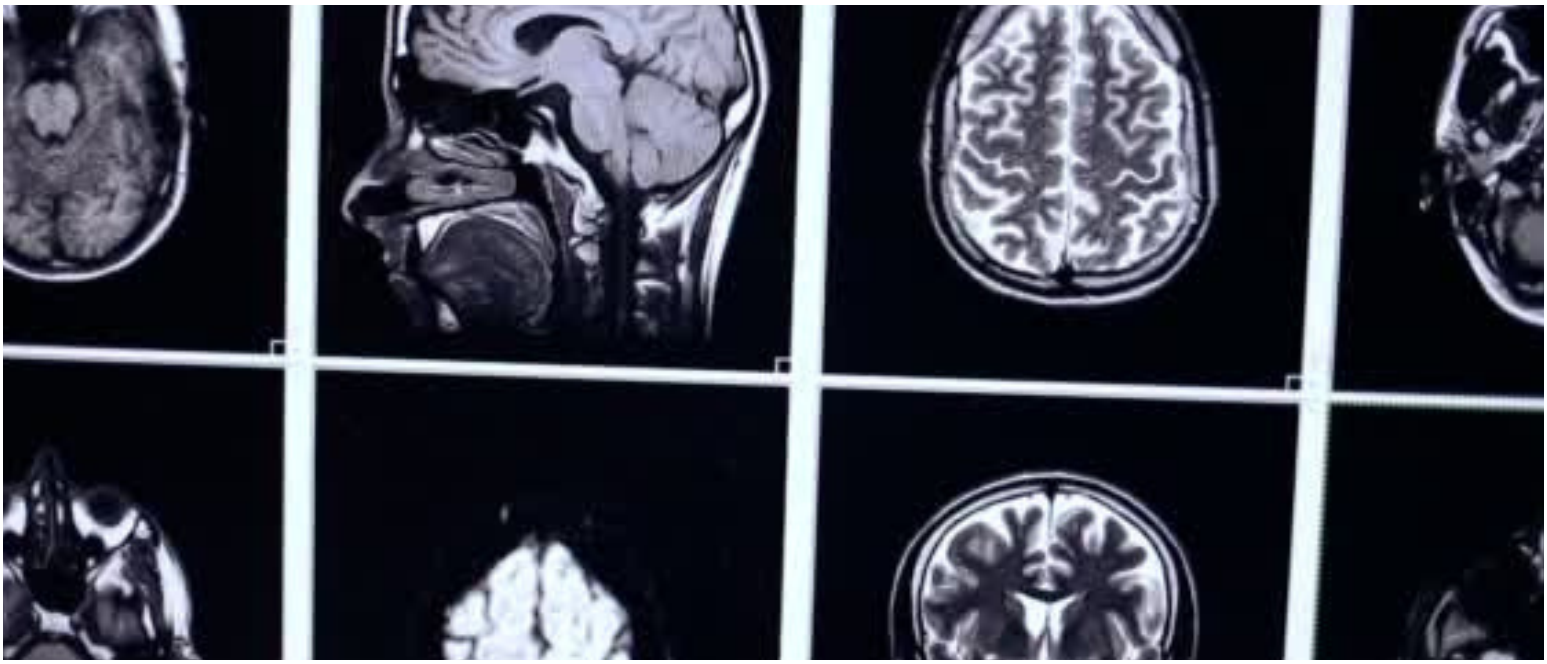
REMERCIEMENTS

Avant de plonger dans les détails de ce projet, nous tenons à exprimer notre gratitude envers ceux qui nous ont soutenus et ont rendu cette expérience enrichissante.

Tout d'abord, nous remercions chaleureusement notre interlocuteur de projet, M. Adrien UGON, pour nous avoir accordé l'opportunité de travailler sur ce sujet. Nos remerciements vont également à notre superviseuse de projet, Mme Ségolène COMBETTES, pour ses précieux conseils et son soutien tout au long de ces trois mois. Sa disponibilité et son écoute ont grandement contribué à notre progression, et nous lui en sommes reconnaissants.

Nous adressons nos remerciements à l'entreprise Alia Santé pour nous avoir permis de participer activement à un projet innovant et pour nous avoir offert la chance de découvrir l'écosystème des start-ups et de l'innovation dans le domaine de la santé.

Enfin, nous voulons remercier Julien PAGAZANI et Eric LLORENS pour nous avoir mis à disposition un serveur GPU et pour leur réactivité par rapport aux problèmes rencontrés.



INTRODUCTION

La quantité de données disponibles ne cesse d'augmenter, mais malgré cela, près de 85% des projets liés aux données échouent, principalement en raison de problèmes de qualité des données. Dans le domaine de l'intelligence artificielle, le manque de données de qualité constitue un obstacle majeur, car ces données sont souvent coûteuses et difficiles à obtenir en raison de problèmes de confidentialité et de leur nature sensible. Cette situation est particulièrement critique dans le domaine de la santé, où les données sont limitées, notamment pour les maladies rares, et hautement confidentielles, ce qui entrave leur partage.

Imaginez un monde où il serait possible de produire des quantités illimitées de données de haute qualité, à faible coût, et entièrement anonymes et sécurisées. Cette vision est désormais possible grâce aux données synthétiques. Les données synthétiques sont des données artificielles créées par des algorithmes d'intelligence artificielle entraînés sur des données réelles. Elles reproduisent fidèlement les caractéristiques et les relations présentes dans les données réelles, sans limitation de quantité, et garantissent l'anonymat des individus concernés.

Plusieurs méthodes sont utilisées pour générer des données synthétiques, notamment les GANs et les VAEs qui sont les plus populaires. Les modèles de diffusion représentent également une option prometteuse pour la génération de données synthétiques, bien qu'ils soient moins courants dans ce domaine. Par exemple un des utilisateurs très connus de modèle de diffusion est DALL-E développé par OpenAI. Ces modèles, basés sur des réseaux de neurones, altèrent les données d'entrée en y ajoutant du bruit Gaussien jusqu'à ce qu'elles deviennent du bruit pur, puis les débruitent pour obtenir de nouvelles données.

Le choix du modèle dépendra principalement des caractéristiques du jeu de données réelles (type, quantité, complexité, etc.). Il est donc essentiel de développer et d'optimiser chaque type de modèle pour pouvoir répondre à diverses problématiques. Dans le cadre spécifique de la santé, le défi principal consiste à développer un modèle de diffusion performant pour la génération de données synthétiques d'images médicales.

SOMMAIRE

1. PRESENTATION DU PROJET

1.1 Entreprise et choix du projet

Dans le cadre de notre projet E4, nous avons convenu de travailler sur le sujet initial intitulé *"Développement d'un modèle de diffusion pour la génération de données synthétiques de type image"*. Notre équipe de projet est composée de trois étudiants en Biotechnologie et e-santé, de deux étudiants en Data science et Intelligence Artificielle, ainsi que d'un étudiant en informatique. C'est donc naturellement que nous avons décidé d'approfondir notre compréhension de la technologie de génération de données synthétiques et de nous perfectionner dans le domaine de l'intelligence artificielle appliquée à la santé. Ce projet que nous avons choisi a été proposé par l'entreprise Alia santé. Pour la bonne réalisation de cet axe de recherche, nous avons donc travaillé en étroite collaboration avec cette dernière.

Alia Santé est une entreprise située à Toulouse qui se spécialise dans le développement d'algorithmes d'intelligence artificielle pour le secteur de la santé. Leur offre inclut plusieurs solutions technologiques : des systèmes de gestion de bases de données et des outils d'annotation de données. Ils proposent également la création de modèles d'intelligence artificielle et la production de données synthétiques, le tout destiné à favoriser l'innovation dans les domaines de la biotechnologie et de la technologie médicale, tout en conformité avec les normes de confidentialité telles que le RGPD. Ils aspirent à démocratiser l'utilisation de l'IA dans la santé, renforçant ainsi l'expertise des soignants grâce à des solutions éthiques et scientifiquement fondées.

Figure 1 : Logo Alia Santé

1.2 Le projet

1.2.1 Les données synthétiques

Les données synthétiques sont produites artificiellement par un algorithme d'intelligence artificielle entraîné sur des données réelles. Le but est de reproduire les caractéristiques et les modèles statistiques d'un ensemble de données existant en modélisant leur distribution probabiliste et en effectuant un échantillonnage.

Les données synthétiques offrent plusieurs avantages significatifs, en particulier dans le secteur de la santé. Premièrement, elles éliminent les risques liés à l'utilisation de données personnelles identifiables, garantissant ainsi la conformité avec les lois strictes sur la confidentialité des données. Deuxièmement, les données synthétiques peuvent être générées en quantités illimitées et adaptées spécifiquement aux besoins des projets de recherche, offrant une grande flexibilité pour tester différentes hypothèses et scénarios. Enfin, l'utilisation de données synthétiques réduit les coûts associés à la collecte, au stockage et à la gestion des données réelles.

Cependant, l'utilisation de données synthétiques dans le secteur de la santé n'est pas sans défis. Bien qu'elles soient conçues pour être représentatives, elles peuvent ne pas toujours capturer la complexité et la variabilité des conditions médicales réelles. Ceci peut affecter l'exactitude des résultats des recherches. De plus, la génération de données synthétiques doit être soigneusement gérée pour s'assurer qu'elles restent conformes aux normes éthiques et légales en vigueur. C’est pourquoi il est nécessaire d’avoir différentes mesures pour vérifier la validité de celles-ci ainsi que l’avis d’un professionnel de santé.
Cependant, notre rôle dans ce projet n’est pas de vérifier la qualité de nos données, ce rôle est attribué à un autre groupe qui doit s’occuper de mettre en place des métriques d’évaluation pour la qualité des données.

1.2.2 Problématiques observées

Les principales problématiques que nous avons rencontrées dans notre projet sont :

La complexité des modèles : les modèles de diffusion, bien que prometteurs, sont très complexes à paramétrer et à optimiser, ce qui nécessite une expertise approfondie en apprentissage profond et en traitement d'image. C'est ce que nous allons voir dans la suite de notre rapport.

Adaptabilité et scalabilité : la solution que nous développons doit être non seulement efficace mais aussi suffisamment flexible pour s'adapter à divers types de données et à des volumes croissants, tout en garantissant la rapidité et la précision des résultats.

Dans le cadre de ce projet, nous allons nous concentrer en particulier sur ces deux problématiques. Cependant, il en existe d'autres. En effet, les données de santé sont régies par des règles extrêmement strictes , afin de protéger la confidentialité. Les données de santé se font donc rares et ne sont pas diffusables.

Après avoir mené une phase de recherche approfondie et échangé avec notre superviseuse, Ségolène Combettes, nous avons identifié les besoins essentiels de notre projet ainsi que toute l'organisation que nous allons devoir mettre en œuvre pour notre projet.

2. ORGANISATION DU PROJET

2.1 Organisation du travail

Pour mener à bien notre projet, nous avons d'abord entrepris une phase de recherche approfondie sur notre sujet en consultant diverses ressources pour nous informer au mieux. Ces ressources nous ont permis de faire une première prise en main de notre projet et ainsi de mieux prévoir les tâches qui allaient suivre. *Tous les documents utilisés sont dans la bibliographie*.

Ces documents offrent un aperçu de diverses approches et techniques dans le domaine du traitement d'images et des modèles de diffusion, allant des modèles probabilistes de débruitage aux méthodes d'apprentissage non supervisé profond.

Voir Gant en annexe A. Nous entamons notre projet par une phase initiale entièrement consacrée à la recherche et à la prise en main du sujet. Comme mentionné précédemment, cette étape inclut une recherche approfondie à travers des documents, des vidéos, des articles et des codes relatifs au thème abordé.

Cette première phase s'est suivie d'une phase de planification durant laquelle, nous avons établi un cahier des charges fonctionnel et une liste chronologique des tâches à effectuer pour une meilleure organisation du projet. De plus, nous avons identifié et pris en compte les différents risques potentiels auxquels nous pourrions être confrontés tout au long du processus. Nous avons reparti les différentes tâches en fonction des compétences de chacun, afin d'optimiser notre temps de travail. Cette répartition est illustrée dans notre tableau que nous avons fait sur l'application Notion présentée en *annexe*.

Dans la seconde phase de notre projet, nous maximisons nos efforts sur différents tests de code avec des structures différentes pour notre modèle de diffusion. Dans cette partie, nous avons uniquement expérimenté des données avec de petites dimensions comme MNIST (28x28 pixels avec 1 canal car images en niveaux de gris, CIFAR-10 (32x32 pixels avec 3 canaux car elles sont en couleur RGB)... Après tous ces différents tests, nous nous sommes concentrés sur un seul modèle qui correspondait à nos attentes.

Après la sélection d’un seul modèle sur lequel nous avons joint nos efforts, nous sommes passés sur l’expérimentation et la modification du code, pour que celui-ci fonctionne avec nos données finales. Nous avons récolté ces données sur le site Kaggle, qui propose des jeux de données avec différentes classes et des images à tailles variables.Dans notre cas nous utilisons des images IRM de coupes de cerveau pour la détection de la maladie d’Alzheimer. Les classes que nous considérons sont les suivantes : malade, avec des dissonances cognitives et sain.

Et pour finir, dans la dernière partie, nous allons nous occuper de l’optimisation de nos hyperparamètres et de l’ajustement de notre programme pour avoir les meilleurs résultats possibles. Ces ajustements pourront varier d’une simple modification d’un paramètre unique à des rajouts de fonctions/code. L'objectif est de garantir que notre programme soit compatible avec tous les types d'images, quelles que soient leur taille, couleur ou forme.

2.2 Environnement/logiciels utilisés

Notre projet a débuté le mardi 6 février et se terminera le mardi 30 avril, nous offrant ainsi une période de 12 semaines pour sa réalisation.
Afin de faciliter la communication et d'assurer une collaboration harmonieuse, notre principal canal de communication avec Alia Santé est **Slack.**

Pour garantir une organisation optimale et une transparence totale, nous utilisons **Notion** pour suivre et visualiser les tâches restantes. Cet outil est constamment mis à jour pour refléter notre avancement. *Voir annexe B*.

Dans notre projet, nous exploitons et faisons des tests sur des programmes en DeepLearning utilisant la bibliothèque Pytorch, plus particulièrement des CNN (Convolution Neural Network), un réseau spécialisé dans le traitement et la reconnaissance d'image. Ceci demande des ressources très importantes pour le traitement que seuls des GPU performants peuvent offrir. Cela signifie que nous étions dans l'incapacité de faire ces traitements sur nos machines personnelles.

C'est pourquoi, nous avons obtenu l'autorisation d'utiliser un GPU de l'école. Pour accéder et exploiter ce GPU, nous établissons d'abord une connexion SSH avec une machine rebond de l'école. Une fois connectés à cette machine intermédiaire, nous pouvons nous connecter en SSH au GPU de développement. Ce GPU possède l'accélération Cuda ce qui nous permet d'accélérer le traitement des données et d'utiliser la capacité de calcul pour nos applications de deep learning.
Voici un schéma explicatif du système :

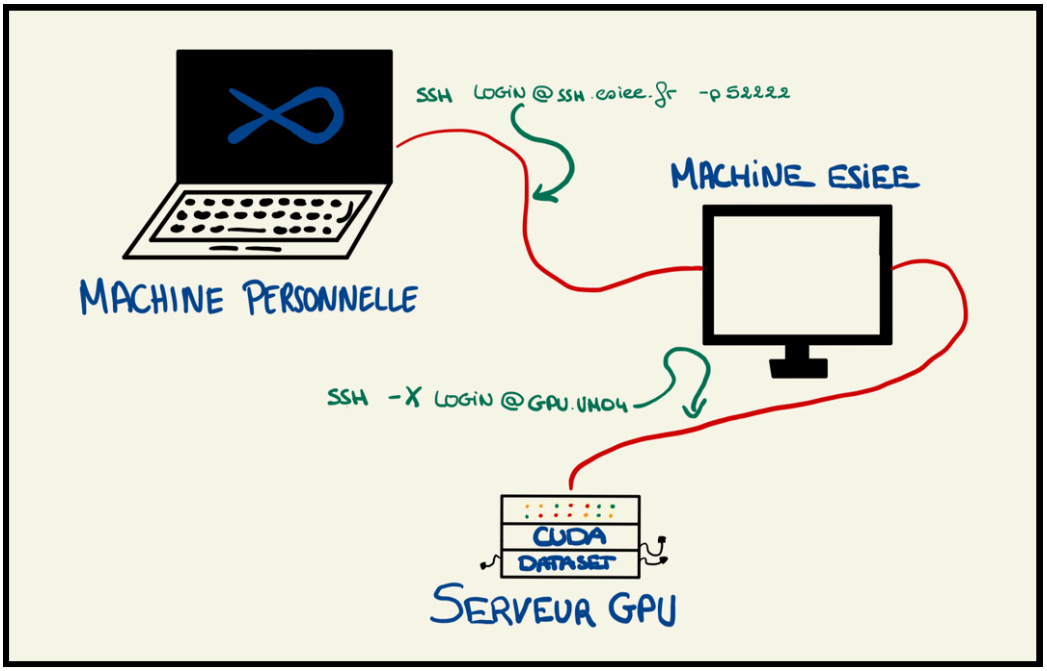


Figure 2 : Schéma Explicatif de notre environnement de travail

Sur notre ordinateur personnel, nous utilisons **Visual Studio Code** équipé de l'extension Remote SSH, qui nous permet de nous connecter directement à la machine rebond de l'école pour coder. Nos codes sont stockés sur cette machine intermédiaire. L'exécution des programmes ainsi que le stockage des jeux de données se font directement sur le serveur GPU.

3. PHASE DE TEST

3.1 Principe du modèle de diffusion

Les modèles de diffusion sont des nouvelles techniques innovantes de génération qui transforment progressivement un signal bruité en une image ou tout autre type de données. Ils sont basés sur un processus itératif comportant deux étapes principales : le bruitage progressif et le débruitage progressif.

Processus de bruitage (Forward Noising) :

Ce processus commence par une image claire X0 et y ajoute progressivement du bruit pour la transformer en un signal aléatoire XT. Chaque étape de bruitage est décrite par une chaîne de Markov, où l'état suivant ne dépend que de l'état actuel, sans mémoire des états précédents. À chaque étape t, on ajoute du bruit gaussien pour obtenir Xt de Xt−1, avec une variance contrôlée par une séquence de variance βt. Les coefficients αt sont calculés à partir de βt, permettant de réguler le mélange du bruit et de l'image.
La formule pour obtenir l'image bruitée à l'étape t est :

$$\alpha_t := 1 - \beta_t$$
$$\bar{\alpha}_t := \prod_{s=1}^{T} \alpha_s$$
$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$$

La probabilité d'une chaîne complète de bruitage peut être exprimée comme le produit de chaque étape de bruitage individuelle :

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t I)$$

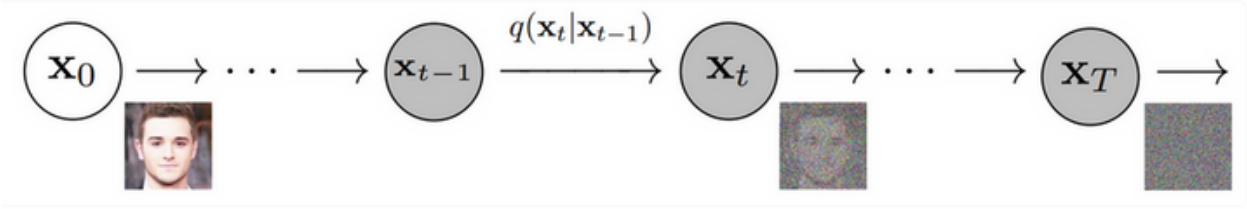


Figure 3 : Chaine Markov Forward , source "Denoising Diffusion Probabilistic Models" par Jonathan Ho

Processus de débruitage (Backward Denoising) :

Après le processus de bruitage, le modèle de diffusion entreprend de reconstruire l'image originale à partir de l'image complètement bruitée. Cela est fait en inversant le processus de bruitage : à chaque étape, le modèle apprend à prédire le bruit qui a été ajouté à l'étape précédente et à le soustraire de l'image bruitée.
Ce processus utilise un réseau de neurones (généralement un U-Net) pour prédire ce bruit. Le réseau est entraîné pour minimiser la différence entre le bruit prédit et le bruit réel ajouté pendant la phase de bruitage.

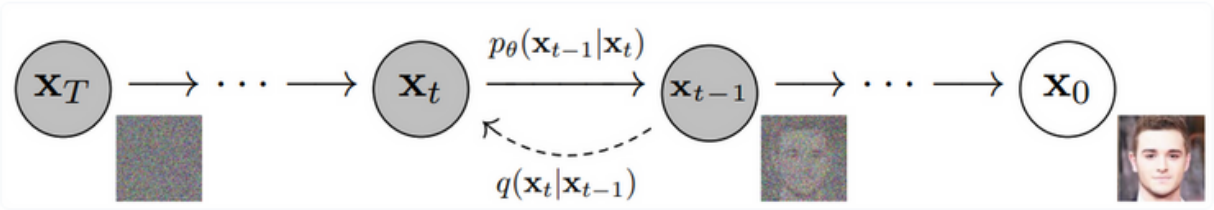


Figure 4 : Chaine Markov Backward, source "Denoising Diffusion Probabilistic Models" par Jonathan Ho

Implémentation et optimisation:

En pratique, ces modèles sont complexes à entraîner car chaque pixel de l'image est traité comme une dimension dans l'espace de données. De plus, le processus d'apprentissage doit être stable sur un grand nombre de pas de temps et maintenir la capacité du modèle à prédire correctement le bruit à différentes étapes du processus.
Les modèles de diffusion sont particulièrement puissants pour la génération d'images, surpassant souvent les GANs en termes de qualité et de fidélité des images générées. Ils sont également adaptés pour d'autres tâches de génération de contenu telles que la création audio et de texte.

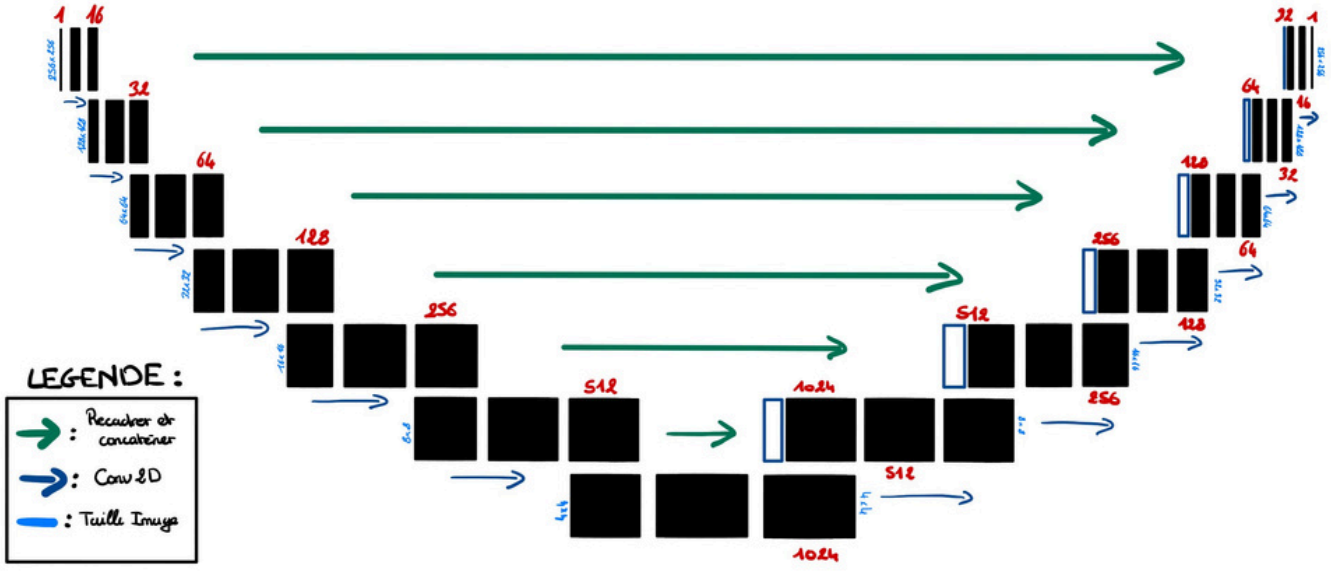


Figure 5 : Architecture U-Net

3.2 Les tests réalisés

Pour le modèle préconçu :

Une équipe dédiée se concentre sur l'adaptation de modèles préexistants en suivant des tutoriels spécifiques, particulièrement avec des données MNIST, tout en les ajustant à nos ensembles de données uniques pour optimiser les résultats. Lors de cette phase de test, nous travaillions sur l'adaptation à nos données, en nous attaquant notamment aux défis liés à l'ajustement des couches du modèle et à la gestion de la mémoire. Notre objectif était de rendre le code fonctionnel, de l'optimiser et d'incorporer des fonctionnalités spécifiques pour répondre à nos besoins.

Quant au développement à partir de zéro :

Un autre groupe au sein de l'équipe s'est consacré au développement d'un algorithme de bout en bout dans la même optique. Nous avons réussi à coder une fonction de bruitage et le modèle de débruitage en se basant sur un modèle de convolutional neural network (CNN) de type auto encoder. Nous l'avons par la suite testé sur les données du data set MNIST, puis l'avons adapté aux images de type IRM.
Cependant, plusieurs problèmes ont été rencontrés. Notre attention s'est portée sur la résolution des problèmes d'affichage des images bruitées et débruitées, ainsi que sur les prochains tests du modèle de débruitage.

Figure 6 : Exemple de données MNIST

3.2.1 Test modèle préexistant avec MNIST

Préparation des données :
Le jeu de données MNIST, composé d'images de chiffres écrits à la main, constitue la base de notre entraînement. Chaque image est transformée en tenseur et subit une normalisation pour que ses valeurs soient comprises entre -1 et 1. Cette étape est cruciale car le modèle est conçu pour prédire des valeurs de bruit, qui doivent être comparables en termes de gamme avec les données d'entrée.

Architecture du modèle :
Le modèle utilisé est une adaptation du réseau U-Net, configuré pour gérer les processus de bruitage et de débruitage des images. Notre U-Net prend un maximum de 64 channels, traitant des images de 28x28 pixels. Ce modèle stocke des valeurs de bêtas, qui quantifient les niveaux de bruit ajoutés à chaque étape du processus de diffusion, permettant ainsi de simuler la dégradation des images jusqu'à une transformation complète en bruit.

Processus de diffusion :
Le processus de diffusion est biphasique : le processus avant, qui ajoute progressivement du bruit à l'image propre jusqu'à obtenir uniquement du bruit :

Processus de bruitage (Forward) :

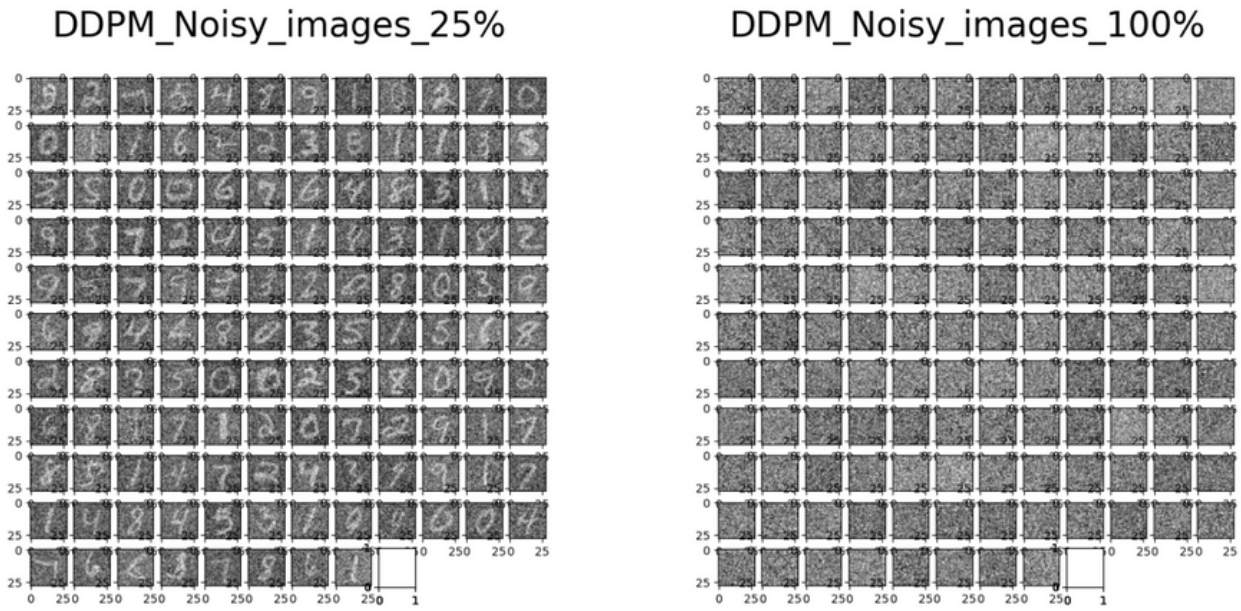


Figure 7 : Représentation du jeu MNIST à 25% et 100% de bruit

Et le processus arrière, qui essaie de reconstruire les images à partir du bruit :

Processus de débruitage (Backward) :

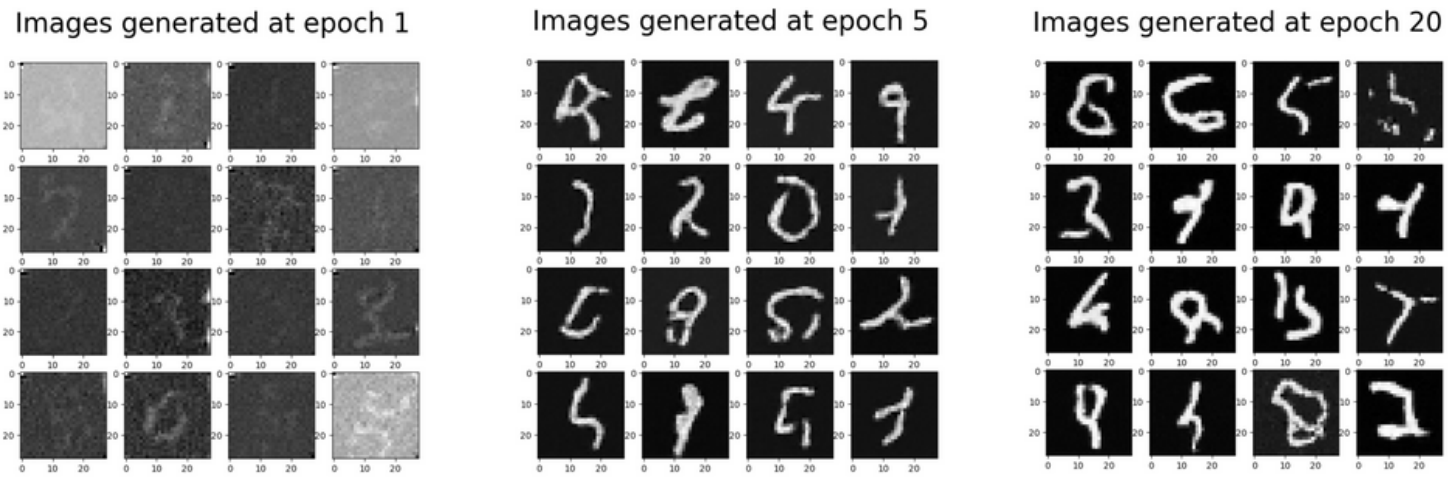


Figure 8 : Représentation du jeu MNIST à la 1ére epoch, 5ème et 20ème epoch

Le processus avant est essentiel pour que le modèle apprenne correctement le processus inverse, tandis que dans le processus arrière, le modèle utilise la moyenne prédite du bruit pour tenter de restaurer l'image originale.

Résultat final :

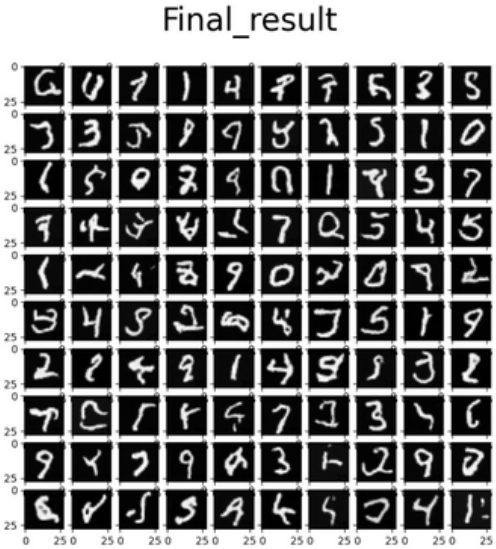


Figure 9 : Résultat finale obtenue sur le jeu de données MNIST

Entraînement du modèle:

L'entraînement du modèle est réalisé par l'optimisation d'une fonction de perte de l'erreur quadratique moyenne (MSE) entre le bruit réellement ajouté et celui prédit par le modèle à partir des images bruyantes. L'entraînement se déroule sur plusieurs époques, permettant au modèle de voir les images à différents niveaux de bruit et d'apprendre à inverser efficacement ce bruit.

Génération d'images et évaluation :
Après l'entraînement, le modèle est capable de générer de nouvelles images en démarrant de bruit aléatoire, et en appliquant de manière itérative le processus appris pour nettoyer ces images. Diverses méthodes de visualisation sont utilisées pour évaluer la qualité des images générées et pour vérifier l'efficacité du modèle dans le processus de débruitage.

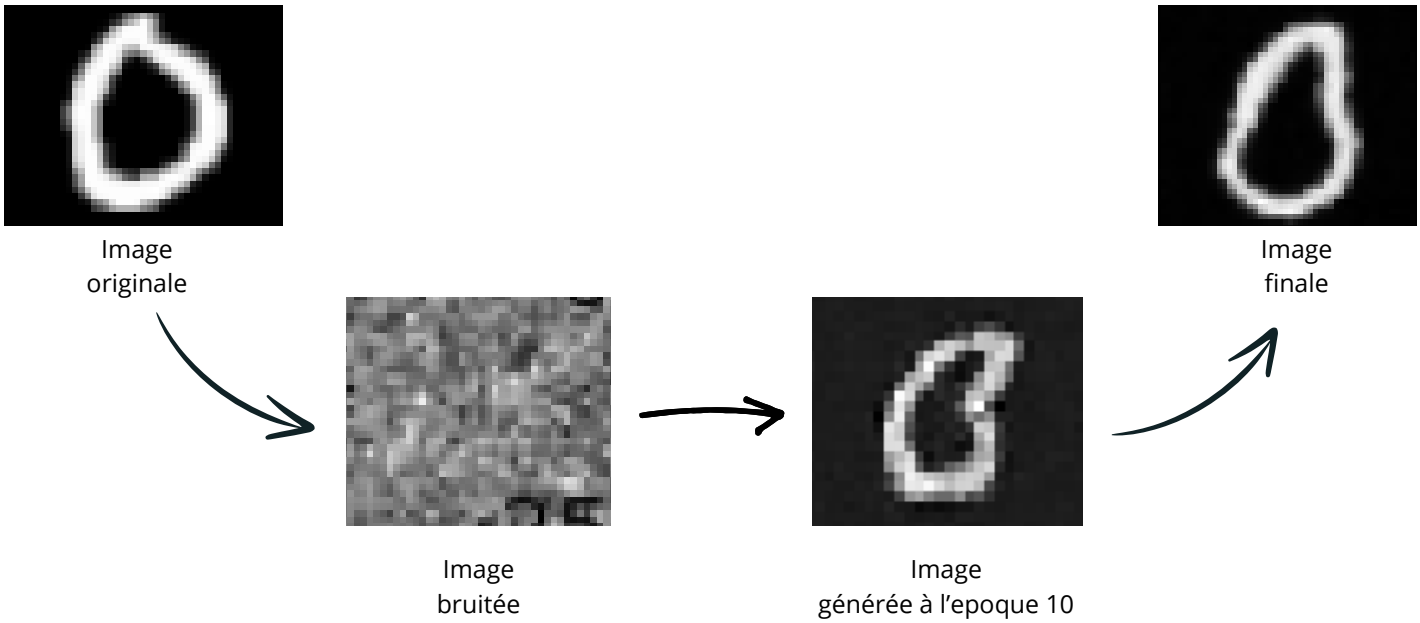


Figure 10 : processus simplifié de bruitage - débruitage

3.2.2 Modèle from scratch

Dans cette partie, nous avons entrepris la construction d'un modèle de bruitage et débruitage de nos données de santé (images de coupes de cerveau) à partir de zéro, en suivant une approche méthodique en plusieurs étapes, en utilisant la bibliothèque TensorFlow Keras pour la mise en œuvre:

- **Construction de l'encodeur** :
  - L'encodeur est construit comme un modèle séquentiel Keras.
  - Il comporte plusieurs couches de convolution (Conv2D) suivies de couches de max-pooling (MaxPooling2D), qui réduisent progressivement les dimensions de l'image d'entrée.
  - La dernière couche de l'encodeur est une couche Dense avec une dimension de sortie correspondant à la dimension latente spécifiée.

- **Construction du décodeur** :
  - Le décodeur est également construit comme un modèle séquentiel Keras.
  - Il comporte une couche Dense initiale qui prend la dimension latente en entrée et la redimensionne en un volume 3D.
  - Ensuite, plusieurs couches de déconvolution (Conv2DTranspose) sont utilisées pour reconstruire l'image à partir de la représentation latente.

- **Construction de l'autoencodeur** :
  - L'autoencodeur est construit en reliant l'encodeur et le décodeur à l'aide de la classe **Model** de Keras.
  - L'entrée de l'autoencodeur est spécifiée comme une image de taille (256, 256, 1) en niveaux de gris.

- **Entraînement de l'autoencodeur** :

  - L'autoencodeur a été entraîné sur des images bruitées (**X_train_noisy**) et leurs versions non bruitées (**train_images**) pendant 50 époques sur les données du data set MNIST.

En raison du temps nécessaire pour perfectionner ce modèle sur les données de santé et de ses exigences croissantes, nous avons choisi de ne pas poursuivre son utilisation. À la place, nous avons décidé d'opter pour des modèles préétablis, tout en adaptant les différentes couches pour correspondre au format de nos données spécifiques. Cette approche nous permettra de gagner du temps tout en maintenant la pertinence de nos solutions pour les données de santé.

3.4 Analyse des risques

Surapprentissage (Overfitting) :
Comme pour tout modèle de machine learning, les DDPMs peuvent être sujets au surapprentissage, surtout s'ils sont entraînés sur un jeu de données limité ou peu diversifié. Dans un tel cas, le modèle apprend à reproduire parfaitement les données d'entraînement mais perd en capacité de généralisation, ce qui réduit son efficacité sur de nouvelles données ou dans des situations non couvertes par l'ensemble d'entraînement.

Coût computationnel élevé :
Les DDPMs nécessitent généralement un grand nombre de pas de diffusion pour générer des images de haute qualité, ce qui entraîne des coûts computationnels élevés et une consommation importante de ressources, comme la puissance de calcul GPU. Cela peut limiter leur utilisation à des environnements où les ressources matérielles sont suffisantes. Nous avons lors de nos tests rencontré plusieurs fois un problème lié à la mémoire disponible sur le GPU à cause de la taille du modèle et de certains hyperparamètres.

Qualité et diversité des images générées :
La qualité des images générées par les DDPMs peut parfois être incohérente, surtout si les paramètres du modèle ne sont pas bien ajustés. De plus, la diversité des images générées peut être limitée si le modèle n'est pas suffisamment robuste ou si le jeu de données d'entraînement n'est pas varié.

Choix des hyperparamètres :
Le succès de la formation d'un DDPM dépend fortement du choix des hyperparamètres, tels que le nombre de pas de diffusion et les coefficients de bruit. Un mauvais choix de ces paramètres peut conduire à un apprentissage inefficace ou à une qualité médiocre des images générées.

Risques liés au modèle from scratch

- Temps de développement : Le développement de solutions à partir de zéro peut prendre plus de temps que l'adaptation de modèles préconçus, ce qui peut retarder la mise en production.
- Risque d'échec de conception : Il existe un risque que les solutions développées ne répondent pas adéquatement aux besoins du projet ou qu'elles soient moins performantes que les modèles préconçus.

4. PROGRAMME FINAL

4.1 Architecture du programme

Structure :

Le programme est structuré de manière à faciliter l'organisation et la compréhension du code. Voici une description de chaque élément de l'architecture:

- **config.ini** : Fichier de configuration contenant les paramètres par défaut pour l'entraînement et la sortie des images. Ces paramètres sont essentiels pour personnaliser le comportement du modèle de diffusion selon les besoins spécifiques du projet.
- **doc/** : Dossier contenant la documentation technique, sous forme de fichiers HTML et PDF (train.html, train.pdf, unet.html, unet.pdf). Ces documents fournissent des détails sur l'entraînement et la structure du réseau U-Net utilisé.
- **output.py** : Script responsable de la gestion des sorties du modèle, notamment la génération et la sauvegarde des images ainsi que la visualisation des statistiques de performance.
- **requirements.txt** : Fichier listant toutes les dépendances nécessaires à l'installation pour que le projet puisse fonctionner correctement.
- **train.py** : Script principal pour l'entraînement du modèle. Il utilise les configurations définies dans config.ini et intègre toutes les étapes nécessaires, du chargement des données à l'entraînement du modèle.
- **unet.py** : Fichier contenant la définition du modèle U-Net, qui est une architecture de réseau de neurones convolutif profond utilisée pour la segmentation d'images et adaptée ici pour le processus de diffusion des images.

Configuration du fichier config.ini :

Le fichier config.ini joue un rôle crucial en définissant les paramètres utilisables dans les scripts. Voici les détails des configurations :

[DEFAULT] :
dataset : Chemin d'accès aux données d'entraînement.
class : Option permettant de filtrer les données par classe.
model : Chemin d'accès au modèle pré-entraîné.
img_size : Taille des images après redimensionnement pour l'entraînement.
n_channel : Nombre de canaux de l'image (1 pour les images en niveaux de gris, 3 pour les images en couleur, type RGB).
batch_size : Nombre d'images par lot lors de l'entraînement.
n_epochs : Nombre total de cycles d'entraînement.
lr : Taux d'apprentissage pour l'optimiseur.
n_steps : Nombre d'étapes pour le processus de diffusion.
min_beta, max_beta : Paramètres contrôlant l'intensité du bruit ajouté durant le processus de diffusion.
debug : Activation du mode débogage pour visualiser des images et des graphiques durant l'entraînement.
[Output] :
n_images : Nombre d'images générées pour visualiser les résultats.
output_path : Chemin où les images générées seront sauvegardées.

Cette configuration permet de contrôler finement les paramètres du modèle et de l'entraînement, facilitant ainsi les ajustements nécessaires pour optimiser les performances.

4.2 Le modèle

Dans le cadre du projet de génération de données d'image utilisant un modèle de diffusion, le code de la partie unet.py et train.py est essentiel. Ces fichiers définissent l'architecture du modèle de réseau de neurones et les routines d'entraînement. Cette description technique approfondira le modèle U-Net utilisé, ses composants personnalisés, et les méthodes de diffusion différentiable (DDPM) implémentées.

Description de Sinusoidal Embedding:

Le sinusoidal embedding est une technique utilisée pour incorporer des informations de position ou de temps dans les modèles de traitement de séquences ou d'images, en particulier dans des architectures de réseaux de neurones profonds. Dans le contexte de votre modèle de diffusion d'images, le sinusoïdal embedding est utilisé pour encoder le temps, ce qui est crucial dans le contrôle du processus de bruitage et de débruitage au cours des étapes de diffusion.

Fonctionnement de sinusoidal_embedding:

Cette fonction génère une matrice d'embedding où chaque ligne correspond à un moment dans le temps, et chaque colonne représente une fonction sinusoïdale de fréquences variées. Les valeurs sinus et cosinus alternées permettent de capturer des relations périodiques et de fournir des gradients stables, ce qui est bénéfique pour le processus d'apprentissage du modèle.

Ce type d'embedding est particulièrement adapté aux modèles qui doivent gérer de longues séquences ou des processus temporels, car il permet de distinguer de manière unique chaque position ou temps sans nécessiter une formation supplémentaire, évitant ainsi le surapprentissage sur des positions spécifiques.

Rôle Critique des Time Embeddings :

Les embeddings temporels sont essentiels pour intégrer la notion de progression au sein du processus de diffusion. Ils permettent au modèle de savoir "quand" dans le processus de diffusion il se trouve, ce qui est crucial pour ajuster l'intensité du traitement du signal d'entrée (image bruitée). En ajustant ces embeddings, le modèle peut finement ajuster les opérations de débruitage à des étapes spécifiques, améliorant ainsi la fidélité de la reconstruction d'image.

Description de MyBlock:

MyBlock est un bloc de construction modulaire conçu pour les réseaux de neurones convolutifs, intégrant des opérations de convolution, de normalisation et d'activation. Cette classe est utilisée pour construire les différentes couches de l'architecture U-Net dans le modèle de diffusion.

Structure et fonctionnalités de MyBlock:

- **Convolution**: Le bloc utilise des couches de convolution (nn.Conv2d) pour transformer les entrées en fonction des apprentissages des filtres. Il comporte deux couches convolutives pour une extraction plus riche des caractéristiques.
- **Normalisation**: Une normalisation par couches (nn.LayerNorm) est appliquée pour stabiliser l'apprentissage en normalisant les activations à travers les caractéristiques spatiales, ce qui est particulièrement utile pour les images.
- **Activation**: Le bloc utilise l'activation GELU (nn.GELU), qui est une fonction d'activation non linéaire permettant de modéliser des relations complexes entre les caractéristiques.
- **Flexibilité**: Le bloc peut être configuré avec ou sans normalisation et avec différentes fonctions d'activation, offrant ainsi une adaptabilité en fonction des besoins spécifiques de différentes parties du réseau.

Ce design modulaire et la réutilisation de MyBlock à travers l'architecture U-Net permettent une construction cohérente et efficace du réseau, en maximisant la capacité d'extraction des caractéristiques tout en conservant la stabilité et l'efficacité du calcul.

Flexibilité et Performance:

La flexibilité de MyBlock avec des options pour activer ou désactiver la normalisation et pour utiliser différentes fonctions d'activation rend le modèle adaptable à divers types de données et exigences de traitement. Cette adaptabilité est essentielle pour optimiser la performance du réseau en fonction des caractéristiques spécifiques des données traitées.

Architecture de MyUNet:

MyUNet est une classe basée sur l'architecture U-Net, bien connue pour ses applications en segmentation d'images mais adaptée ici pour les tâches de génération d'images par diffusion. Le modèle est construit pour traiter des images de différentes tailles (128, 256, 512 pixels), avec une architecture en forme d'entonnoir profondément symétrique pour la descente et la montée graduelle de résolution.

Éléments clés de MyUNet:

- **Sinusoidal Embedding**: Utilisé pour l'incorporation temporelle, ce qui permet au modèle de manipuler le bruit ajouté à chaque étape spécifique durant la diffusion.
- **Blocks (MyBlock)**: Des blocs de convolutions personnalisés qui incluent la normalisation par couches, des convolutions et l'activation GELU. Ces blocs sont utilisés pour construire à la fois les parties descendantes (encodeur) et montantes (décodeur) du réseau.
- **Transitional Layers**: Des couches de convolution et de convolution transposée sont utilisées pour réduire et ensuite augmenter la résolution des cartes de caractéristiques à travers le réseau.
- **Time Encoding**: L'ajout de l'embedding temporel à différentes étapes du réseau pour intégrer la notion de temps, qui est critique pour la régulation du processus de diffusion.

Cette structure permet au MyUNet de manipuler efficacement les données d'image pour le bruitage et le débruitage à travers les étapes de diffusion, essentiel pour le processus de génération de données.

Utilisation de MyBlock dans MyUNet:
Chaque instance de MyBlock dans MyUNet est configurée pour effectuer des tâches spécifiques qui varient selon leur position dans l'architecture. Par exemple, les premiers blocs peuvent se concentrer sur l'extraction de caractéristiques de bas niveau, tandis que les blocs plus profonds traitent des caractéristiques de plus haut niveau. Cette hiérarchisation est essentielle pour la reconstruction précise dans les étapes de débruitage du modèle de diffusion.

Sinusoidal Embedding et Time Embedding:
Le sinusoïdal embedding joue un rôle crucial non seulement dans l'encodage du temps mais aussi dans l'amélioration de la capacité du modèle à gérer des variations temporelles longues sans perte d'information. L'embedding temporel est utilisé pour moduler l'activité des blocs convolutifs en fonction de l'étape de diffusion, permettant ainsi une adaptation dynamique du processus de débruitage.

Modèle de Diffusion Différentiable (DDPM):

Le modèle de diffusion différentiable (MyDDPM) est conçu pour simuler et inverser un processus de Markov où des images sont progressivement converties en bruit, puis reconstruites en images. Ce processus est contrôlé par un calendrier de "température" beta, qui ajuste l'intensité du bruit ajouté à chaque étape.

Composants de MyDDPM:

- **Forward Pass**: Génère des images bruitées en utilisant un calendrier beta pour les pas de temps donnés.
- **Backward Pass**: Estime et retire le bruit des images, en utilisant le réseau sous-jacent (U-Net) pour prédire le bruit à chaque étape.

Boucle d'Entraînement:

La boucle d'entraînement implémente l'entraînement du modèle sur des lots de données, ajuste les poids du réseau basé sur l'erreur entre le bruit réel et le bruit estimé, et surveille la progression de l'entraînement à travers les époques.
Points clés de la routine d'entraînement:

- **Optimisation**: Utilisation de l'Adam Optimizer pour minimiser l'erreur quadratique moyenne (MSE) entre le bruit ajouté et celui estimé par le réseau.
- **Debugging et sauvegarde**: En mode debug, visualisation des images intermédiaires et enregistrement des meilleurs modèles.

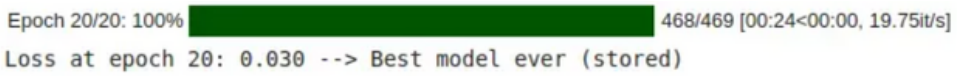


Figure 11: visualisation de l'entrainement dans le terminal

Le modèle MyUNet et MyDDPM ensemble forment une architecture puissante pour la tâche de génération d'images par diffusion, avec des mécanismes avancés pour le contrôle temporel et l'optimisation du bruit. L'implémentation détaillée assure que le réseau peut apprendre efficacement à partir des données, atteindre une haute performance dans la génération d'images, et être ajusté précisément grâce aux configurations disponibles dans config.ini.

4.3 Optimisation des Hyperparamètres

Dans cette section, nous examinons la sélection et l'ajustement des hyperparamètres cruciaux pour le modèle de diffusion utilisé dans le projet de génération d'images de cerveaux. L'optimisation des hyperparamètres est essentielle pour maximiser les performances du modèle, en particulier en ce qui concerne la qualité et la fidélité des images générées.

Choix de la Fonction d'Activation : SiLU vs GELU

SiLU (Sigmoid Linear Unit): Initialement envisagée, cette fonction d'activation est reconnue pour son efficacité dans de nombreux modèles de réseaux de neurones grâce à sa nature non linéaire qui permet de réintroduire des valeurs négatives, utiles pour certaines caractéristiques des données.
GELU (Gaussian Error Linear Unit): Suite à une révision du code récente, le GELU a été identifié comme plus performant pour le projet en cours. GELU, qui simule une activation basée sur des probabilités gaussiennes, permet de mieux capturer les détails fins et de fermer de manière plus précise les régions d'images cérébrales.

Avantages pour notre projet: L'adoption du GELU a permis une amélioration significative de la précision des détails internes des images, un aspect crucial pour les applications médicales où la distinction des petites structures cérébrales est essentielle pour un diagnostic précis.

Conv2D vs MaxPool2D pour le Downsampling

Conv2D vs MaxPool2D: Dans le cadre du downsampling, deux techniques principales étaient en compétition : l'utilisation de Conv2D et celle de MaxPool2D. MaxPool2D, souvent employé pour sa capacité à réduire rapidement les dimensions spatiales tout en préservant les caractéristiques dominantes, était une option envisageable. Cependant, après des tests comparatifs, Conv2D s'est avéré plus efficace pour notre application spécifique.
Avantages de choisir Conv2D pour notre projet: L'adoption de Conv2D pour le downsampling a offert plusieurs avantages clés. Contrairement à MaxPool2D qui effectue une réduction agressive pouvant entraîner une perte d'informations importantes, Conv2D permet une transition plus douce et contrôlée des caractéristiques. Cette méthode conserve mieux les informations essentielles et détaillées des images, ce qui est crucial pour les applications médicales où chaque détail peut être significatif. En conséquence, nous avons observé une amélioration notable de la qualité visuelle et de la continuité des caractéristiques dans les images reconstruites.

Choix de la Normalisation: GroupNorm vs LayerNorm

GroupNorm: Bien que GroupNorm ait été considéré pour améliorer la normalisation en traitant des groupes de canaux simultanément, il n'a pas montré d'amélioration significative dans nos tests.

LayerNorm: La normalisation par couches a été préférée car elle s'est révélée plus efficace pour nos données spécifiques, normalisant les activations à travers les caractéristiques spatiales des images.

Avantages pour notre projet: LayerNorm a facilité un entraînement plus stable et efficace en normalisant les entrées de chaque couche, contribuant ainsi à une meilleure convergence du modèle.

Utilisation de Multi Head Attention

Multi Head Attention: Bien que populaire dans de nombreux modèles de traitement du langage naturel et d'image, l'intégration de mécanismes d'attention multi-tête n'a pas apporté d'amélioration significative dans notre contexte et a augmenté la complexité et la consommation de mémoire du modèle.

Conséquences pour notre projet: Nous avons décidé de ne pas inclure Multi Head Attention dans l'architecture finale pour conserver une efficacité de calcul optimale tout en maintenant une performance satisfaisante sur les tâches de génération d'image.

Ces décisions d'optimisation des hyperparamètres ont été guidées par des tests exhaustifs et des révisions de code, permettant de configurer le modèle pour obtenir les meilleures performances possibles sur la tâche spécifique de génération d'images médicales.

Indécision sur les Hyperparamètres : Évaluation et Choix

La sélection et l'ajustement des hyperparamètres dans les modèles de génération d'images de type diffusion restent un domaine rempli d'incertitudes, notamment en raison de l'absence de métriques d'évaluation standardisées qui permettraient une comparaison objective et précise entre différentes configurations. Pour notre projet, la décision sur les hyperparamètres a principalement reposé sur des expérimentations itératives, la comparaison des pertes durant l'entraînement, et l'évaluation visuelle des images générées. Cette méthode, bien que nécessaire, peut souvent donner l'impression d'avancer à l'aveugle, car les critères d'évaluation ne sont pas toujours clairement quantifiables ou directement liés à des améliorations qualitatives mesurables.

Un autre groupe de projet travaille sur la mise en place de métriques d'évaluation spécifiques pour la qualité des données synthétiques de type image. Ce projet aurait été extrêmement bénéfique pour notre équipe, car il aurait offert des outils pour évaluer plus rigoureusement et objectivement la performance de nos modèles. Cela aurait potentiellement permis de réduire l'incertitude et d'optimiser plus efficacement nos choix d'hyperparamètres en se basant sur des résultats quantifiables plutôt que sur des observations subjectives.

Toutefois, afin de respecter les objectifs et l'autonomie de chaque projet, nous avons choisi de ne pas empiéter sur leur domaine de recherche. Cela nous a laissé sans autre choix que de continuer à nous appuyer sur notre approche initiale, malgré ses limitations. L'intégration future de telles métriques pourrait non seulement valider rétroactivement nos choix mais aussi guider l'amélioration continue de notre modèle en fournissant des feedbacks précis sur l'efficacité de différentes configurations d'hyperparamètres

4.4 Résultat

Qualité Visuelle des Images Synthétiques

La phase de résultats de notre projet démontre une réussite notable dans la génération de données d'image synthétiques médicales à l'aide de modèles de diffusion. Les images obtenues au terme des 200 époques révèlent une qualité visuelle impressionnante, avec une clarté et une fidélité aux données originales qui dépassent les attentes initiales. En comparant les images synthétiques avec les images originales, on constate une conservation remarquable des structures et des contours cérébraux essentiels pour les applications médicales. Cette fidélité visuelle est primordiale pour garantir l'utilité clinique des données synthétiques. Cependant comme vue plus tôt nous n'avons pas les ressources pour certifié que ces images sont valides.

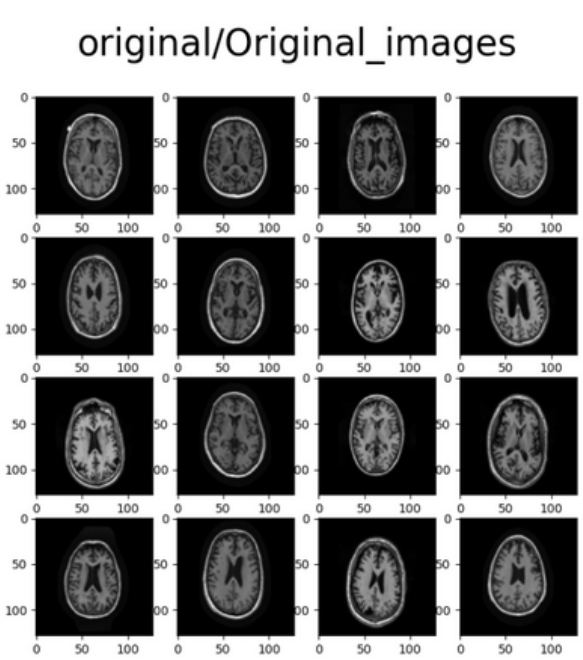


Figure 12 : Affichage des images originales

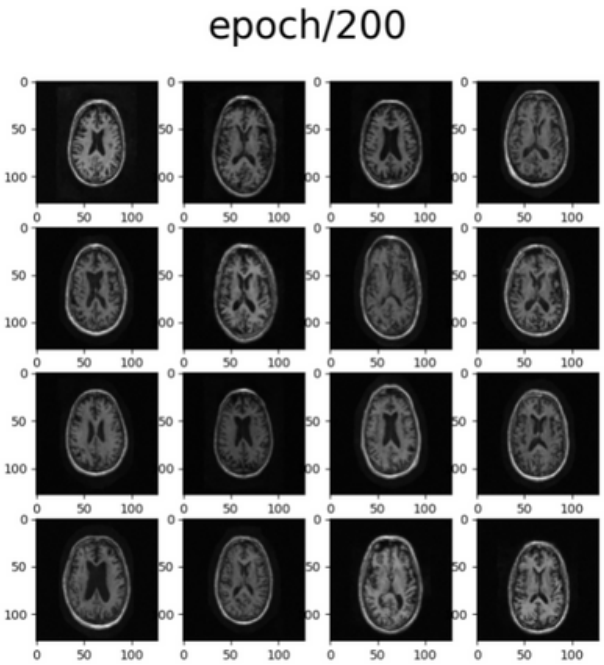


Figure 13 : Affichage de l'epoch 200

Réduction du Loss durant l'Entraînement

L'évolution du loss durant les entraînements indique une convergence stable et robuste du modèle. La courbe de perte, en décroissance rapide initialement, s'aplatit progressivement, signifiant une optimisation efficace des paramètres du modèle. Cette évolution souligne la capacité du modèle à apprendre avec précision les caractéristiques pertinentes des images médicales au fil des époques, et sa capacité à généraliser à partir des données originales pour produire des images de haute qualité.Cependant dans le cas de génération d'image nous avons pu remarquer qu'un fonction de coût faible voir très faible ne signifiait pas forcement que notre modèle aller nous donner des bon résultats en sortie.

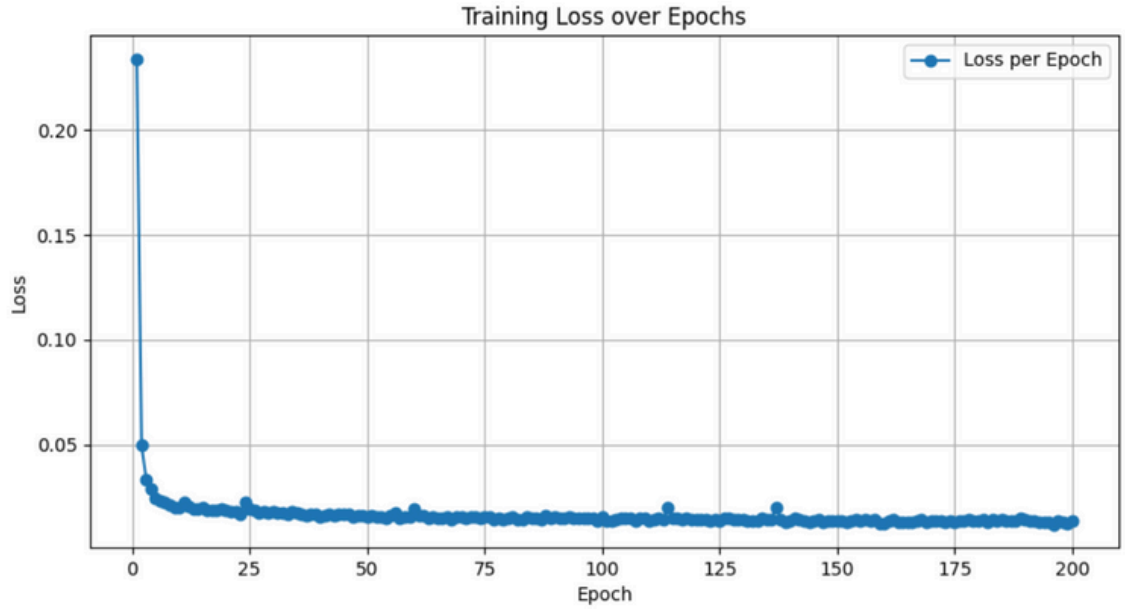


Figure 14 : Affichage de la fonction de coût sous forme de graphique

Comparaison entre une image réelle et une synthétique

L'objectif ultime est de parvenir à un niveau de qualité où même un expert ne pourrait distinguer une image synthétique d'une image réelle. Cependant, lors de cette comparaison directe, certaines différences notables ont été observées, mettant en lumière des axes d'amélioration pour notre modèle. Parmi ces différences, la netteté des motifs internes au cerveau apparaît comme un domaine crucial où le modèle pourrait être amélioré. De plus, l'épaisseur des parois du crâne dans les images synthétiques ne correspond pas toujours exactement à celle observée dans les images réelles, ce qui affecte la vraisemblance générale des images générées.

Ces observations démontrent que, bien que des progrès significatifs aient été réalisés, il reste des étapes importantes à franchir pour atteindre la perfection dans la simulation d'images IRM de cerveau. Ces résultats fournissent une direction claire pour les futures itérations de développement du modèle, avec un focus sur l'amélioration de la précision des détails internes et de la conformité structurelle externe.

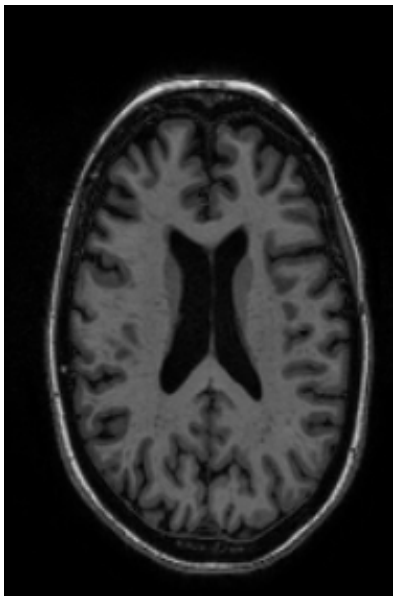


Figure 15 : Image originale

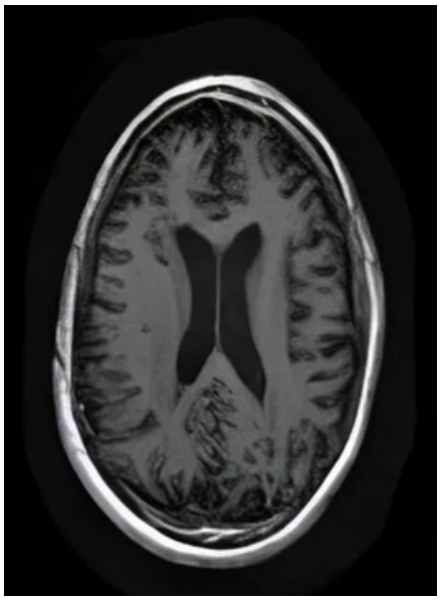


Figure 16 : Image généré

5. RETOUR D’EXPERIENCE

Durant ces trois mois de projet, nous avons pu expérimenter un aperçu du fonctionnement du travail en équipe qui nous sera très utile pour les années à venir. Ayant un objectif commun, nous avons su nous organiser de façon efficace en fonction des aptitudes de chacun.

En effet, les membres de l’équipe ont su échanger et se responsabiliser les uns les autres, afin de rendre un travail de qualité.

Cette expérience nous a donc permis de nombreux apprentissages, tels que le développement de soft skills, à savoir : le partage des compétences interpersonnelles, la répartition de la charge de travail, le soutien mutuel, l’amélioration de la qualité du travail et de la gestion des risques.
Cependant, lors de la phase d’exécution, nous avons aussi rencontré plusieurs difficultés techniques lors de la phases de tests avec notre code.

En effet, la gestion du temps a été un défi majeur en raison de la complexité du sujet. Néanmoins, nous avons pu tirer profit de cette expérience, notamment dans le développement des hard skills. En effet, nous avons pu donner vie à une idée abstraite.
Nous avons à présent une meilleure compréhension des modèles de diffusion, mais également une meilleure maitrise en termes de génération de données synthétiques de type images.

Finalement lors de ce projet, nous avons su nous entraider les uns les autres, afin de résoudre les quelques problèmes rencontrés, ce qui a permis de contribuer à la réussite et à l’efficacité de notre projet.

CONCLUSION

En conclusion, notre projet a exploré les vastes possibilités offertes par les données synthétiques, en particulier dans le secteur de la santé où les défis liés à la disponibilité et à la confidentialité des données sont prééminents. Grâce à l'utilisation innovante des modèles de diffusion, nous avons démontré que la production de données synthétiques peut non seulement compléter les ensembles de données existants mais également accroître l'accessibilité et la sécurité des informations médicales sans compromettre la confidentialité des patients.

Au cours de ce projet, nous avons franchi des étapes significatives dans la compréhension et l'application des techniques de génération de données synthétiques. Les modèles de diffusion, en particulier, ont montré un potentiel exceptionnel pour créer des images médicales. Toutefois, malgré les avancées technologiques, nous avons rencontré des défis inhérents à l'absence de métriques d'évaluation robustes pour quantifier la qualité des données générées, ce qui a rendu l'optimisation des hyperparamètres particulièrement complexe.

Le projet a également mis en lumière la nécessité d'une collaboration multidisciplinaire pour surmonter les barrières techniques et éthiques. La collaboration avec d'autres équipes travaillant sur des métriques d'évaluation pourrait permettre d'obtenir des validations plus précises et de raffiner nos modèles de manière plus objective. Cela souligne l'importance d'une approche intégrée et collaborative dans le développement de technologies avancées en IA.

À l'avenir, la poursuite de la recherche et le développement dans ce domaine seront cruciaux pour surmonter les obstacles restants et pour maximiser le potentiel des données synthétiques. Cela comprend l'amélioration des algorithmes de génération pour qu'ils produisent des données encore plus réalistes et applicables à un éventail plus large de conditions médicales, ainsi que l'élaboration de cadres réglementaires et éthiques pour guider l'utilisation de ces technologies.

BIBLIOGRAPHIE

- A. Helwan, "Diffusion Models from Scratch," Medium, [En ligne]. Disponible : https://abdulkaderhelwan.medium.com/diffusion-models-from-scratch-d00b8d8e1f7a.

- "Diffusion Models from Scratch: MNIST Data Tutorial in 100 Lines of PyTorch Code," Medium, [En ligne]. Disponible : https://papers-100-lines.medium.com/diffusion-models-from-scratch-mnist-data-tutorial-in-100-lines-of-pytorch-code-a609e1558cee.

- V. Jumle, "Image Generation with Diffusion Models Using Keras and TensorFlow," Medium, [En ligne]. Disponible : https://medium.com/@vedantjumle/image-generation-with-diffusion-models-using-keras-and-tensorflow-9f60aae72ac.

- V. Jumle, "Class-conditioned Diffusion Models Using Keras and TensorFlow," Medium, [En ligne]. Disponible : https://medium.com/@vedantjumle/class-conditioned-diffusion-models-using-keras-and-tensorflow-9997fa6d958c.

- Tech for Future, "Tech for Future," [En ligne]. Disponible : https://techforfuture.fr/#1674645575834-76bbce69-e2d6.

- "Make Diffusion Model from Scratch: Easy Way to Implement Quick Diffusion Model," Tree Rocks, [En ligne]. Disponible : https://tree.rocks/make-diffusion-model-from-scratch-easy-way-to-implement-quick-diffusion-model-e60d18fd0f2e.

- "How to Build Diffusion Model From," AI Content Lab, Oct. 2023, [En ligne]. Disponible : https://www.ai-contentlab.com/2023/10/how-to-build-diffusion-model-from.html.

- E. D. Tech, "What Is the Size of a Convolutional Layer?" Baeldung, [En ligne]. Disponible : https://www.baeldung.com/cs/convolutional-layer-size.

- Accenture, "Qu'est-ce que les données synthétiques et l'IA ?" [En ligne]. Disponible : https://accenture.com.

- L'Usine Digitale, "Que sont les données synthétiques et comment aident-elles à créer de meilleurs modèles d'IA ?" [En ligne]. Disponible : https://usine-digitale.fr.

- J. Ho, A. Jain, et P. Abbeel, "Denoising Diffusion Probabilistic Models," UC Berkeley, rapport.

- A. Nichol et P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," rapport.

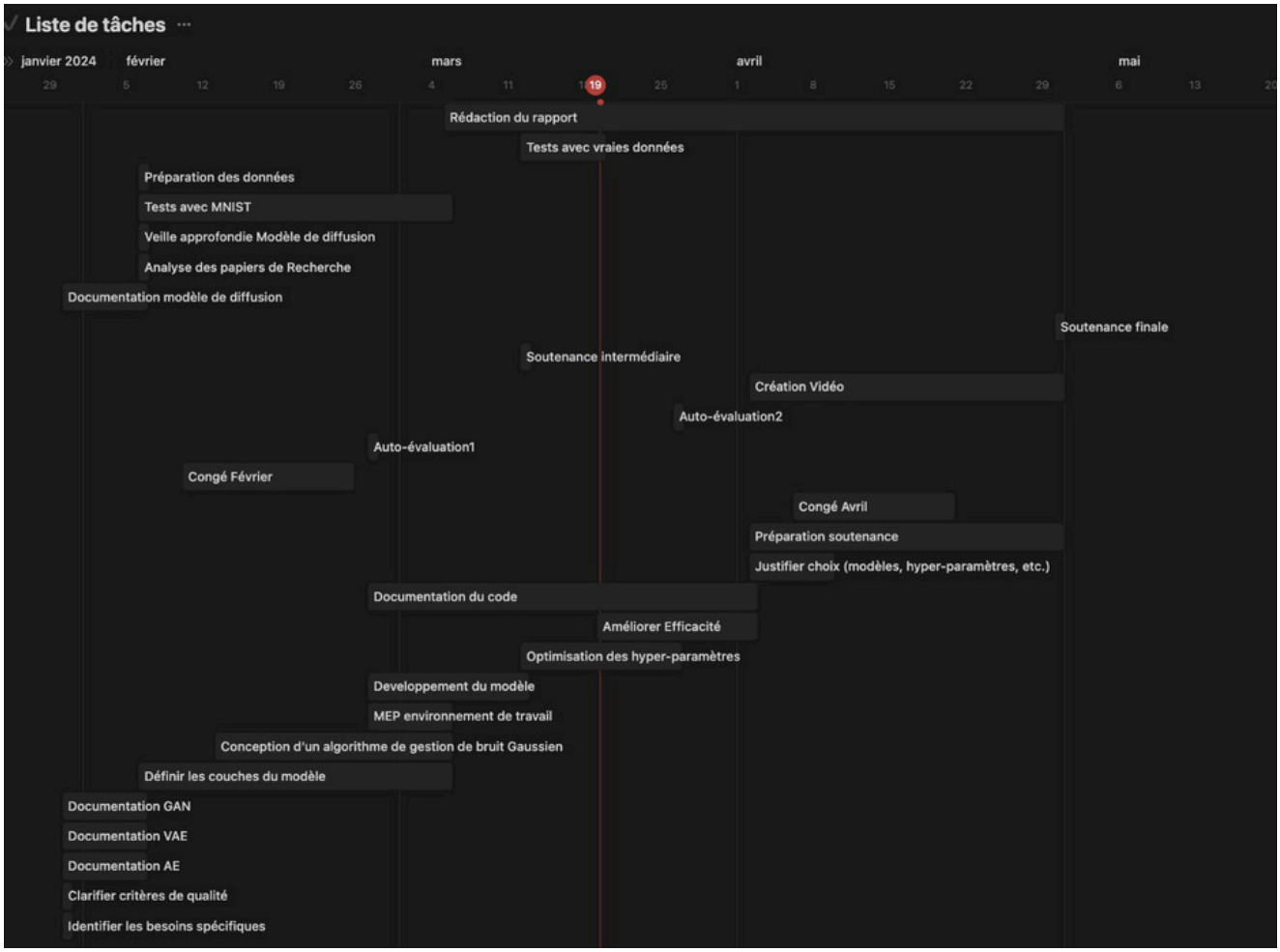
- J. Sohl-Dickstein, E. A. Weis, N. Maheswaranathan, et S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," UC Berkeley, Stanford University, rapport.

- TensorFlow Documentation, "Modèle from Scratch," [En ligne]. Disponible : https://tensorflow.org.

TABLE DES FIGURES

ANNEXES

Annexe A : GANT sur Notion



Annexe B : Listes des tâches sur Notion