DDIM: DENOISING DIFFUSION IMPLICIT MODELS

Mayeul Ricard
Télécom SudParis
Paris, France
mayeul.ricard@gmail.com

1 Introduction

Les DDIM [1] (Denoising Diffusion Implicit Models) sont une amélioration des DDPM (Denoising Diffusion Probabilistic Models), dont l'inférence peut être coûteuse et parfois inefficace, car ils simulent une chaîne de Markov sur T étapes.

Le fonctionnement des DDIM n'est pas très différent de celui des DDPM, notamment au niveau de la phase d'entraînement, qui reste la même. La particularité des DDIM est qu'ils reposent sur un processus non markovien, rendant l'inférence déterministe. Cette approche permet d'accélérer l'inférence d'un facteur allant de 10 à 50.

L'intérêt des DDIM est donc de concurrencer les modèles de type GAN, lesquels avaient largement surperformé les DDPM en termes de vitesse d'inférence: un GAN ne nécessite généralement qu'un seul passage dans le réseau, là où les DDPM en nécessitent potentiellement des milliers.

Dans ce papier, nous allons comprendre le principe des DDIM, les construire et enfin les évaluer.

2 Data

Pour tester notre modèle, nous allons utiliser deux bases de données différentes. La première est MNIST, qui constitue un bon point de départ. Elle répertorie des images de chiffres (0 à 9) au format 28×28. Voici un aperçu de cette base :

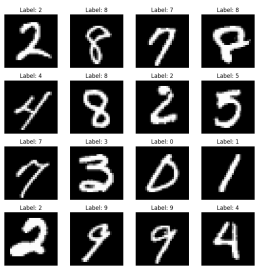


Figure 1: MNIST

La seconde servira à challenger un peu plus notre modèle, car nous allons tenter de générer des avions en couleur à partir de la sous-base dédiée dans CIFAR10. Cette base répertorie des images au format 32×32. Voici un aperçu de la base :

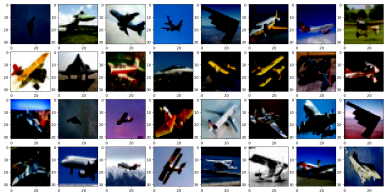


Figure 2: CIFAR10

3 Introduction Mathématique

Afin de reproduire un tel modèle, nous devons poser les bases mathématiques du processus de "forward" et du processus de "backward".

Commençons par définir α_t : il s'agit d'une séquence de scalaires décroissants entre 1 et 0, qui contrôle la quantité de bruit ajoutée au cours du processus de diffusion. Plus T est grand, plus α_t tend vers 0, ce qui signifie que l'image x_t devient essentiellement du bruit gaussien indépendant de l'image initiale x_0.

3.1 Forward Diffusion

Le processus forward consiste à corrompre progressivement les images de notre base de données en leur injectant un bruit gaussien, et ce sur T itérations. Ce processus est identique à celui des DDPM. Nous définissons pour cela q_{sample} comme étant le processus de diffusion forward :

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \tag{1}$$

avec :

$$q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}.x_{t-1}, (1-\alpha_t)\mathbf{I}) \tag{2}$$

On peut prouver par récurrence que :

$$q(x_t \mid x_0) = \mathcal{N}(x_t \mid \sqrt{\overline{\alpha_t}}.x_0, (1-\overline{\alpha_t})\mathbf{I}) \tag{3}$$

On peut donc échantillonner x_t de manière directe :

$$x_t = \sqrt{\overline{\alpha_t}}.x_0 + \sqrt{1-\overline{\alpha_t}}\,\epsilon, \quad \text{où } \epsilon \sim \mathcal{N}(0, \mathbf{I}).$$

Prenons deux images, une par base de données (MNIST et CIFAR10), et appliquons ce procédé forward pour visualiser comment nous "bruitons" itérativement nos données :

Figure 3

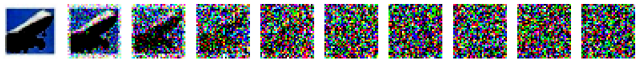


Figure 4

4 Backward Diffusion

Grâce à la formule de Bayes, on peut écrire :

$$q(x_{1:T} \mid x_0) = q(x_T \mid x_0) \prod_{t=2}^{T} q(x_{t-1} \mid x_t, x_0), \tag{4}$$

avec pour tout t > 1 :

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(\sqrt{\overline{\alpha_{t-1}}}x_0 + \sqrt{1-\overline{\alpha_{t-1}}-\sigma_t^2} \cdot \frac{x_t - \sqrt{\overline{\alpha_t}}x_0}{\sqrt{1-\overline{\alpha_t}}}, \sigma_t^2 \mathbf{I}\right)$$

Concernant le processus backward, la formule suivante, dérivée du processus forward, permet de reconstruire x_0 à partir des données bruitées x_t et du bruit $\epsilon(x_t, t)$.

$$f_\theta^{(t)}(x_t) := \frac{x_t - \sqrt{1-\overline{\alpha_t}} \cdot \epsilon_\theta^{(t)}(x_t)}{\sqrt{\overline{\alpha_t}}}.$$

Ainsi, en exploitant la densité $q(x_{t-1} \mid x_t, x_0)$ nous pouvons échantillonner x_{t-1} à partir de $p(x_{t-1} \mid x_t)$

$$p_\theta^{(t)}(x_{t-1} \mid x_t) = \begin{cases} \mathcal{N}\left(f_\theta^{(1)}(x_1), \sigma_1^2 \mathbf{I}\right), & \text{si } t = 1, \\ q(x_{t-1} \mid x_t, f_\theta^{(t)}(x_t)), & \text{sinon.} \end{cases}$$

Nous pouvons alors définir le processus génératif avec un prior fixé $p_\theta(x_T) = \mathcal{N}(0, \mathbf{I})$ et :

$$x_{\tau-1} = \sqrt{\overline{\alpha_{\tau-1}}} \underbrace{\left(\frac{x_\tau - \sqrt{1-\overline{\alpha_\tau}} \cdot \epsilon_\theta^{(\tau)}(x_\tau)}{\sqrt{\overline{\alpha_\tau}}}\right)}_{\text{“predicted } x_0\text{”}} + \underbrace{\sqrt{1-\overline{\alpha_{\tau-1}}-\sigma_\tau^2} \cdot \epsilon_\theta^{(\tau)}(x_\tau)}_{\text{“direction pointing to } x_t\text{”}} + \underbrace{\sigma_\tau \epsilon_\tau}_{\text{random noise}}.$$

Avec σ_{τ_i} :

$$\sigma_{\tau_i}(\eta) = \eta \sqrt{\frac{1-\overline{\alpha}_{\tau_{i-1}}}{1-\overline{\alpha}_{\tau_i}}} \sqrt{1-\alpha_{\tau_i}} \tag{5}$$

Si l'on veut un DDPM, on fixe $\eta = 1$, pour avoir un processus Markovien stochastique.
Si l'on veut un DDIM, on fixe $\eta = 0$, pour avoir une chaine non-markovienne deterministe.

L'avantage des DDIM est que ce processus étant non markovien, on peut diminuer le nombre d'étapes de génération et ainsi échantillonner un x_0 plus rapidement que dans les DDPM qui itèrent sur T. Les DDIM peuvent itérer sur moins d'étapes (par exemple $dim(\tau) < dim(T)$) et inférer jusqu'à 10 ou 100 fois plus vite.

5 Réseau de Neuronnes U-Net

Dans cette partie, nous allons rapidement décrire le fonctionnement du réseau U-Net et son intérêt dans un modèle de diffusion :

Encodage sinusoïdal du pas de temps (Sinusoidal Timestamp Embedding) :

Il s'agit d'un bloc d'encodage temporel très répandu (inspiré des Transformers), qui permet d'ajouter de l'information de position (ou de “temps”) dans le modèle. Dans le cas d'un modèle de diffusion, il est essentiel de signaler l'étape t au réseau, car la quantité de bruit évolue au fil des itérations.

U-Net : Pour le processus de diffusion backward, nous utilisons une architecture de type U-Net. Le U-Net est un réseau convolutif conçu pour les tâches de segmentation d'images, mais il est particulièrement adapté aux modèles de diffusion. Le principe du U-Net repose sur deux étapes principales :

1. Une phase de réduction d'échelle (*downsampling*), où l'image est progressivement compressée à travers des couches convolutives. Cela permet d'extraire des caractéristiques à différents niveaux de détail.

2. Une phase d'augmentation d'échelle (*upsampling*), où l'image est reconstruite en appliquant des convolutions transposées. Cette phase est combinée avec des connexions entre les couches de même niveau dans les phases de réduction et d'augmentation d'échelle. Ces connexions permettent de préserver les détails fins tout en intégrant des informations de haut niveau.

Dans notre cas, le U-Net est utilisé pour apprendre à prédire le bruit ajouté à une image bruitée x_t au temps t. Plus précisément, le réseau estime $\epsilon_\theta(x_t, t)$.

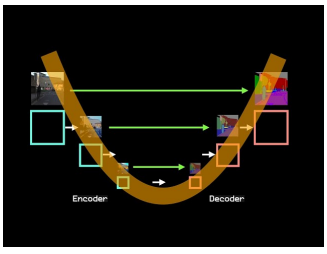


Figure 5: Réseau U-Net

6 Loss Function

Avec la divergence de Kullback-Leibler (KL), on montre que l'on cherche à minimiser la quantité suivante :

$$\frac{\gamma_t}{2\sigma^2}\|\epsilon_t - \epsilon_\theta^{(t)}(x_t)\|^2$$

Mais comme notre réseau de neurones prend t en arguments, chaque réseau est différent à chaque étape t donc nous pouvons supprimer la pondération et minimiser simplement la quantité :

$$\|\epsilon_t - \epsilon_\theta^{(t)}(x_t)\|^2$$

7 Entrainement

Pour l'entraînement sur la base MNIST, J'ai choisi T=1000 étapes et 100 époques (epochs) sur un GPU NVIDIA H100 NVL. L'ensemble de l'entraînement a duré environ 50 minutes. Voici la courbe d'apprentissage sur 100 époques :

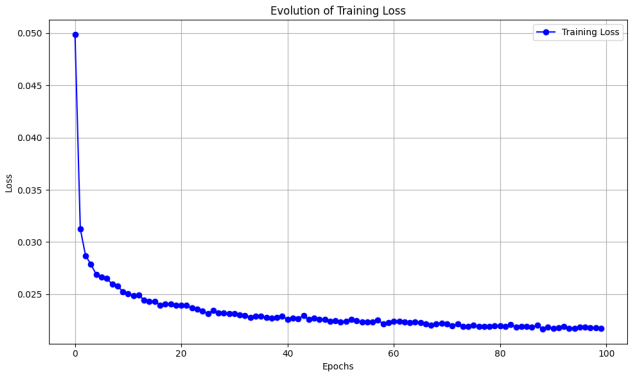


Figure 6

Pour l'entraînement sur la base CIFAR10, j'ai également choisi T=1000, mais cette fois avec 1000 époques. La durée de l'entraînement est restée de l'ordre de 50 minutes, car nous travaillons sur un sous-ensemble plus restreint (les avions). Voici la courbe d'apprentissage sur 1000 époques :

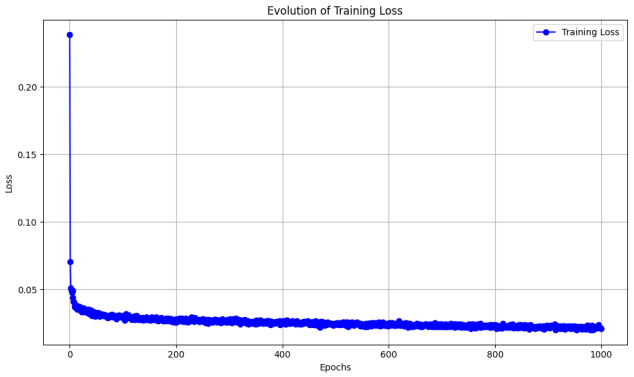
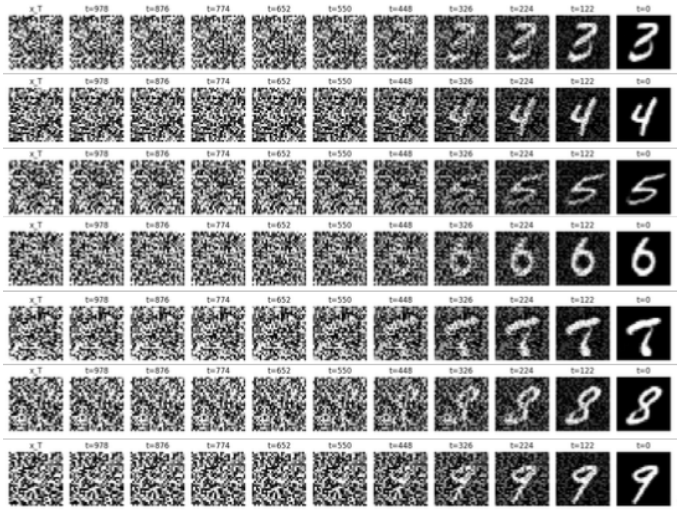


Figure 7

8 Sample Images

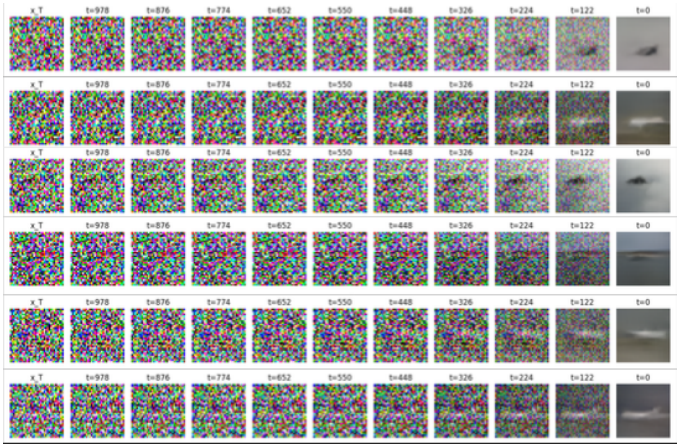
Maintenant que notre U-Net est entraîné à prédire $\epsilon(x_t, t)$, nous pouvons générer des images pour vérifier la qualité de l'apprentissage.

Pour MNIST :



Pour CIFAR :

Les résultats ne sont pas très visible, bien que l'on puisse distinguer des avions. Il aurait probablement fallu entraîner le modèle sur un plus grand nombre d'époques ou disposer d'un plus grand nombre d'images d'avions dans la base de données.



Pour information, tout le code que j'ai produit pour ce projet est disponible sur mon GitHub. [2]

References

[1] Chenlin Meng Stefano Ermon Jiaming Song. 2022. DDIM : DENOISING DIFFUSION IMPLICIT MODELS. *arXiv* (2022). arXiv:2010.02502v4 https://arxiv.org/abs/2010.02502

[2] Mayeul Ricard. 2024. DDIM from scratch in Pytorch. *GitHub* (2024). https://github.com/mylrcd/DDIM