Architectures des réseaux de neurones avancés

Modèles de diffusion

Références

- Ho, J., Jain, A., & Abbeel, P. Denoising diffusion probabilistic models. NeurIPS 2020
- Nichol, A. Q., & Dhariwal, P. Improved Denoising Diffusion Probabilistic Models. ICML 2021
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. High-resolution image synthesis with latent diffusion models. CVPR 2022.

Modèles de diffusion

I. Principe général

II. Modèle de diffusion probabiliste

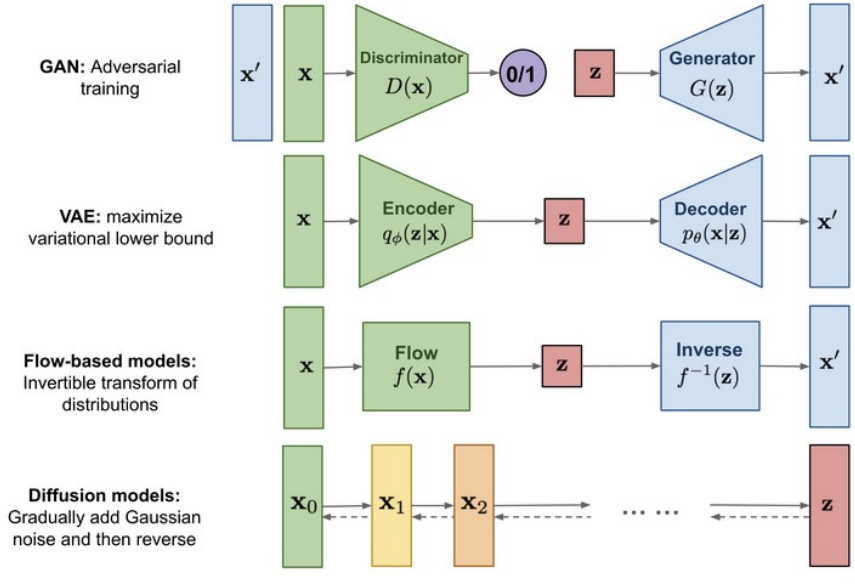
III. Entraînement

IV. Echantillonnage

V. Modèle de diffusion dans un espace latent

VI. Extensions: Stable Diffusion

Différentes approches de modèles génératifs



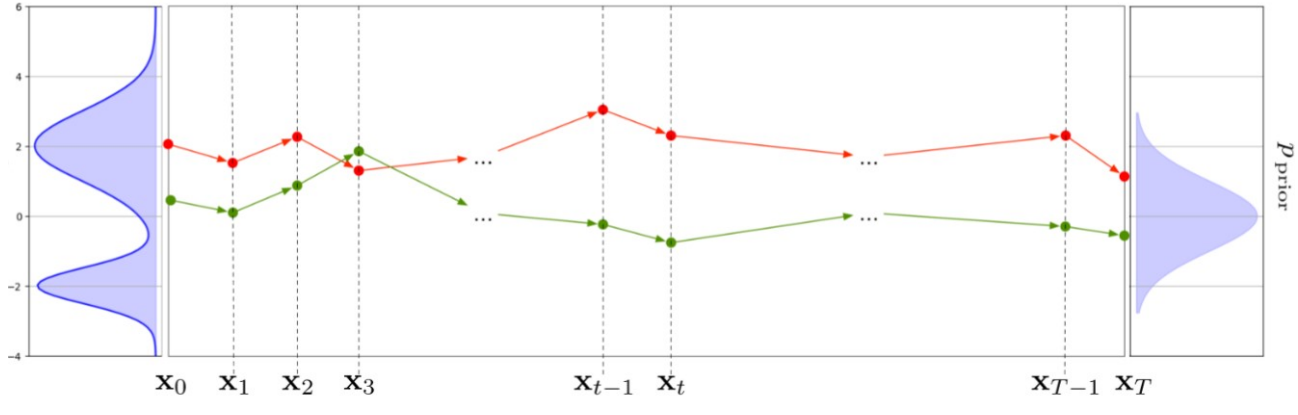
Comparaison des méthodes génératives (src: Lilian Wang's blog)

Principe de la diffusion

- Phénomène inspiré de la physique des particules (mécanique probabiliste)
    - Diffusion de la chaleur, mouvement Brownien des particules, etc.
    - Equation lien la variation temporelle (ordre 1) de la quantité observée (température, position) à ses variations spatiales (ordre 2)
- Pour les modèles génératifs, seul le modèle mathématique général a été conservé

Principe de la diffusion (2)

- On construit un processus de diffusion pour transformer une distribution complexe (celle des données d'entrée $D_{\mathcal{X}}$) en une distribution simple (Gausienne)
- Pratiquement, ceci impose que la taille de l'espace latent est la même que celle des données



Principe de la diffusion (3)

- **But des modèles de diffusion**: soit le processus de diffusion donnée, construire un réseau de neurones produisant le processus inverse
- Mathématiquement, on modélise la distribution réelle des données par une fonction auto-regressive et paramétrique:

$$q(x_0) \approx p_\theta(x_0) = \int p_\theta(x_0, x_1, \ldots, x_T)\, \mathrm{d}x_1 \ldots \mathrm{d}x_T \tag{1}$$

- $p_\theta(x_0, x_1, \ldots, x_T)$ = **processus inverse** ou **backward**
- Sans contrainte supplémentaire, le problème est insoluble...

Diffusion Gaussienne

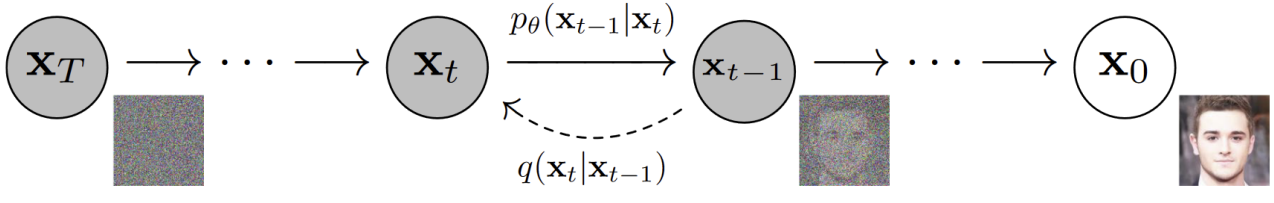
- 2 hypothèses sont ajoutées au modèle de diffusion:
    - Il s'agit d'un **processus Markovien** (la prédiction du future à partir du présent n'est pas rendu plus précise par des éléments d'information issus du passé)
    - La probabilité de transition de l'état à l'instant t à l'état à l'instant $t+1$ est **gaussienne**
- On parle généralement de **difussion gaussienne**
- Propriétés:
    - On peut approximer n'importe qu'elle distribution, aussi difficile soit-elle, si le nombre d'itération est suffisant
    - Il exite une formulation inverse de la même forme (càd, un autre processus de diffusion gaussien)

Diffusion Gausienne (2)

- **Processus Markovien**:

$$p_\theta(x_0, x_1, \ldots, x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

- **Diffusion gaussienne**
  - $q(x_t|x_{t-1})$ est gaussienne et il existe une fomulation inverse $p_\theta(x_{t-1}|x_t)$ aussi gaussienne



Théorie: forward

- On fixe le processus de diffusion (forward) tel que:

$$q(x_t|x_{t-1}) \sim \mathcal{N}\left(\sqrt{1-\beta_t}x_{t-1}; \beta_t\mathbf{I}\right)$$

- Formulation simple mais extrêmement utile pour l'entraînement: on a une forme close pour n'importe quel $q(x_t|x_0)$:

$$q(x_t|x_0) \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0; (1-\bar{\alpha}_t)\mathbf{I}\right) \tag{2}$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) \tag{3}$$

- Il est facile d'echantillonner des x_t à partir des x_0 grâce au **reparametrization trick**

Rappel: Reparametrization trick

- **Contexte**: x suit une distribution gaussienne de moyenne μ et de variance σ^2
- **Objectif**: on veut optimiser μ et σ (ex: via descente de gradient)
- **Problème**: la fonction d'échantillonnage des x n'est pas différentiable...
- **Solution**: On utilise le **reparametrization trick**
    1. On échantillonne ϵ selon une loi gaussienne centrée réduite
    2. On calcule x tel que $x = \mu + \sigma \cdot \epsilon$
- Quelques remarques pour les dimensions > 1:
    - Si les dimensions sont **décorrelées** $\rightarrow$ la matrice de covariance est diagonale, et cela revient simplement à refaire les étapes 1 et 2 autant de fois qu'il y a de dimensions
    - Il existe une forme close (bien plus compliquée) si les dimensions sont correlées

Théorie: backward

- Diffusion gaussienne $\rightarrow p_\theta(x_{t-1}|x_t)$ est gaussien:

$$p_\theta(x_{t-1}|x_t) \sim \mathcal{N}\left(\mu_\theta\left(x_t, t\right); \Sigma_\theta\left(x_t, t\right)\right)$$

- Contrainte additionnelle: Variance fixée pour chaque itération t du processus de diffusion

$$\Sigma_\theta\left(x_t, t\right) = \sigma_t^2 \mathbf{I}$$

- Il y a plusieurs choix de σ_t possible, mais tous sont relativement équivalents, choisir $\sigma_t^2 = \beta_t$ fonctionne très bien
- En simplifiant les équations et après différentes expérimentations, la forme la plus stable pour obtenir $\mu_\theta\left(x_t, t\right)$ est la suivante:

$$\mu_\theta\left(x_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

- ϵ_θ est le réseau de neurones qui va approximer le bruit à retirer à l'itération t, à partir de l'échantillon courant à débruiter, ainsi que du numéro d'itération en cours.

- En utilisant le **reparametrization trick**, on obtient x_{t-1}

Théorie: Entraînement

- Le setup est similaire aux VAEs $\rightarrow$ on optimise la borne inférieure variationnelle (aka **variational lower bound** ou **evidence lower bound** - ELBO)
- Distributions gaussiennes $\rightarrow$ forme close pour le calcul direct
- Evite d'employer des méthodes empiriques imprécises (ex: Monte Carlo)

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Théorie: Echantillonnage

- On connaît notre distribution $p_\theta(x_{t-1}|x_t) \sim \mathcal{N}\left(\mu_\theta\left(x_t, t\right); \Sigma_\theta\left(x_t, t\right)\right)$
- Calcul de $\mu_\theta\left(x_t, t\right)$ nécessite $x_t \rightarrow$ obligé de le faire de manière itérative

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Quel réseau de neurones pour ϵ_θ ?

- Espace latent de même dimension que l'espace des images $\rightarrow$ Architecture type auto-encodeur (ex: U-Net)
- Reconstruction conditionnée par le numéro de l'itération $\rightarrow$ Encodeur positionnel comme dans les Transformers

Quelques valeurs pratiques...

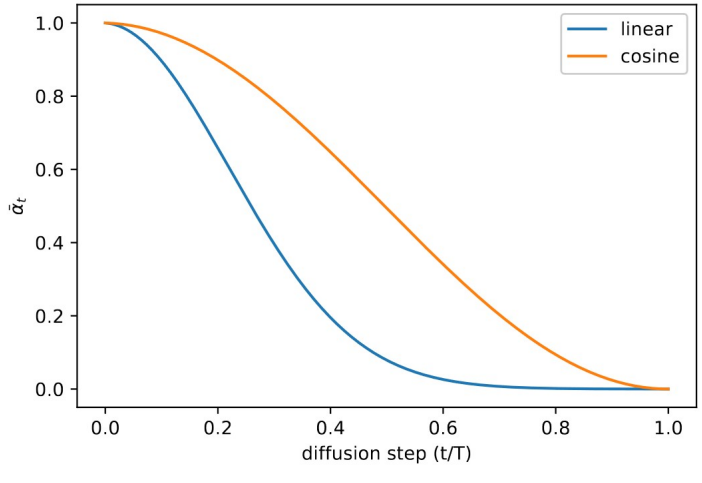
- Stabilité: Les valeurs des pixels {0, ..., 255} sont ré-échelonnées entre [-1...1]
- Pour des soucis de comparaison avec d'autres méthodes, T = 1000
- Les variances β_t sont obtenues par interpolation linéaire entre $\beta_1 = 10^{-4}$ et $\beta_T = 0,02$
- Le réseau de neurones est un peu plus complexe $\rightarrow$ version modifiée de PixelCNN++ (ICLR 2017)

Conclusion

- Modèle de diffusion probabiliste inspiré de la physique
- Diffusion gaussienne: Processus Markovien avec transitions suivant une distribution gaussienne
- Entraînement facilité: similaire à un problème de reconstruction
- Génération: Le reparametrization trick permet de générer des échantillons très facilement

Améliorations : Paramétrisation des β_t

- Article: "Improved Denoising Diffusion Probabilistic Models", ICML 2021
- Paramétrisation des β_t: peu performante en pratique...
- Il est intéressant de regarder les valeurs des $\bar{\alpha}_t$ au lieu des β_t



- β_t fixé $\rightarrow$ amélioration en apprenant les valeurs

Améliorations: Echantillonnage avec nombre de pas réduits

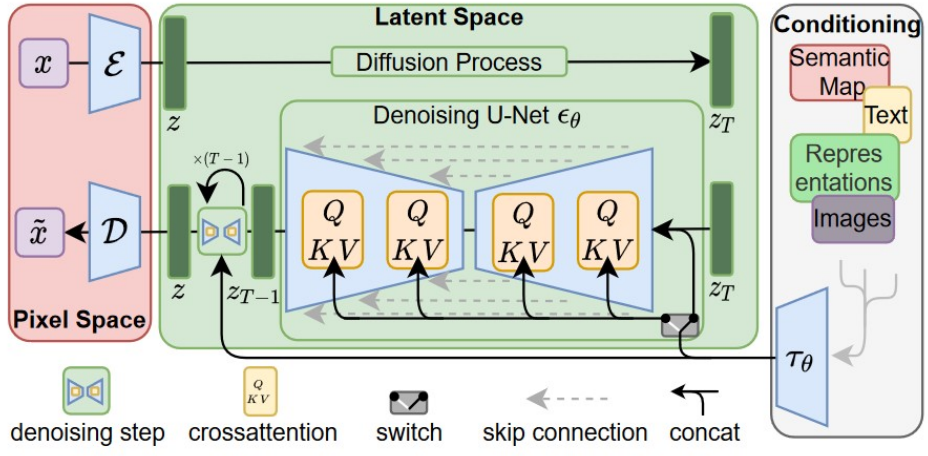
- Même article: "Improved Denoising Diffusion Probabilistic Models", ICML 2021
- Au lieu d'appliquer le processus de diffusion avec un pas de 1, on l'appliquer avec un pas de K
- Modèle d'origine: facteur 3-4 de gain
- Modèle avec la nouvelle paramétrisation de β_t + leur apprentissage: facteur 30-40 de gain

Améliorations: Diffusion dans un espace latent

- Article: Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. High-resolution image synthesis with latent diffusion models. CVPR 2022
- Architecture à la base des approches multi-modales Stable Diffusion
- Idée principale:
    - On encode l'image HD de taille (H x W x 3) dans un espace latent structuré plus petit (h x w x c)
    - On applique le modèle de diffusion dans cet espace latent
    - On décode le résultat pour obtenir une image de taille (H x W x 3)
- Solution proposée: U-Net comme encodeur / décodeur

Améliorations: Modèles conditionnels

- Idée: conditionner le processus de diffusion par un signal externe (classe, texte, ...)
- Les distributions $q(x_t|x_{t-1})$ deviennent conditionnées au nouveau signal: $q(x_t|x_{t-1}, y)$ (même chose pour p_θ)
- Solution proposée: Cross-attention inspirée des Transformers
- Très facile dans un espace latent ! Toute source de conditionnement externe peut être transférée dans ce même espace


Latent Space
Diffusion Process
Denoising U-Net ϵ_θ
Conditioning
Semantic Map
Text
Repres entations
Images
Pixel Space
x
$\mathcal{E}$
$\tilde{x}$
$\mathcal{D}$
z
z_T
z_{T-1}
$\times(T-1)$
Q KV
τ_θ
denoising step
crossattention
switch
skip connection
concat