Deep Learning

Rapport de projet

Score-based generative modeling through stochastic differential equations

Yann DE COSTER et Rémi VIDAL

3 mai 2022

Table des matières

Introduction

L'article *Score-Based Generative Modeling through Stochastic Differential Equations* [1], écrit par Y. Song et al. en 2020, considère la génération d'échantillons de données à partir de bruit grâce aux équations différentielles stochastiques. Se voulant être une alternative aux GANs, ces modèles basés sur l'approximation de la fonction de score ont de multiples applications. En établissant une nouvelle architecture et en proposant différentes méthodes de sampling, les auteurs montrent que leur modèle atteint des performances inédites en *generative modeling*.

Après avoir résumé les concepts détaillés dans l'article, on appliquera leur algorithmes à la génération d'images de voitures. Les résultats seront quantifiés grâce à l'Inception Score.

1 Résumé de l'article

1.1 Bruitage des données avec les équations différentielles stochastiques

Bruiter des données est une tâche relativement facile : il suffit d'ajouter continuellement du bruit sur chacune des composantes de la donnée. Après un certain temps, c'est toute la structure de la donnée qui va s'effacer, transformant cette dernière en bruit. Inverser un tel processus permettrait donc de générer une grande quantité de données en inversant des vecteurs composés de bruit uniquement.

On a donc besoin d'un processus de bruitage qui soit facilement inversible. On désignera par $p_0(x)$ la distribution de nos données, $p_T(x)$ la distribution du prior (qui correspond au bruit de sortie du processus) et $p_t(x)$ la distribution intermédiaire au temps $t \in [0, T]$. Il est proposé de bruiter graduellement la distribution de nos données vers la distribution du prior en utilisant un processus stochastique. Un tel processus est solution d'une équation différentielle stochastique de la forme :

$$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$$

Les équations différentielles stochastiques (SDE) généralisent les équations différentielles ordinaires (ODE) : $f(x,t)$ représente la partie déterministe du processus stochastique. Le terme $g(t)\mathrm{d}w$ contrôle quant à lui la diffusion stochastique du processus, avec w le mouvement brownien standard et $\mathrm{d}w$ le bruit blanc infinitésimal.

Le choix de la SDE n'est pas unique : prendre $f(x,t) = 0$ et $g(t) = e^t$ perturbe par exemple la donnée avec un bruit gaussien de moyenne 0 et de variance qui croît de manière exponentielle. Les auteurs utilisent quant à eux trois SDE qui fonctionnent particulièrement bien pour les images : *Variance exploding SDE* (VE SDE), *Variance Preserving SDE* (VP SDE) et sub-VP SDE, dont nous ne rappelons pas les formules ici.

L'intérêt des équations différentielles stochastiques est que, pour tout processus donné par une équation différentielle stochastique, il existe une équation différentielle stochastique qui inverse ce dernier [2] :

$$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}w$$

$\mathrm{d}t$ représente ici un intervalle de temps négatif infinitésimal, et w est un mouvement brownien qui fonctionne à rebours dans le temps. Pour résoudre l'équation différentielle

stochastique inverse, il est nécessaire de connaître $\nabla_x \log p_t(x)$: cette quantité est appelée la fonction de score de la distribution $p_t(x)$.

Avec l'utilisation d'équations différentielles stochastiques, on obtient un processus inverse qui est décrit par une équation différentielle stochastique de forme connue. Il reste à savoir comment générer des données à partir de cette équation inverse.

1.2 Génération d'échantillons via l'équation différentielle stochastique inverse

Afin de générer de nouveaux échantillons à partir du bruit, il est nécessaire de résoudre l'équation différentielle stochastique inverse (SDE inverse). Pour ce faire, il faut d'abord estimer le score $\nabla_x \log p_t(x)$ à chaque temps $t \in [0, T]$.

Cette fonction de score peut être estimée grâce à un *Time-Dependant Score-Based model*, un réseau de neurones paramétrisé par θ : on obtient alors $s_\theta(x, t)$, proche de la vraie fonction de score $\nabla_x \log p_t(x)$. La fonction objectif de ce modèle est une pondération continue des divergences de Fisher :

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(x)}\left[\lambda(t) \parallel \nabla_x \log p_t(x) - s_\theta(x,t) \parallel_2^2\right]$$

où la variable t est tirée selon une loi uniforme sur $[0, T]$.

Cette fonction objectif ne peut cependant pas être évaluée directement à cause de la présence de la vraie fonction de score $\nabla_x \log p_t(x)$, qui est inconnue. Pour contourner cette difficulté, les auteurs ont recours aux méthodes de *score matching* [3], qui transforment la fonction objectif en une forme équivalente ne dépendant pas de la vraie fonction de score. On peut citer par exemple *Denoising Score matching* [4] ou *Sliced score matching* [5]. Une fois le *score-based model* entraîné, on peut simplement injecter $s_\theta(x, t)$ dans la SDE inverse afin d'en obtenir une approximation :

$$\mathrm{d}x = \left[f(x,t) - g^2(t)s_\theta(x,t)\right]\mathrm{d}t + g(t)\mathrm{d}w$$

On peut enfin résoudre cette équation via des méthodes classiques de résolution de SDE, comme la méthode d'Euler-Maruyama. x est initialisée grâce au prior π, qui par construction est proche de la distribution marginale $p_T(x)$. On a donc $x(T) \sim p_T(x)$. Puis, l'équation différentielle stochastique inverse est discrétisée :

$$\text{Répéter jusqu'à } t = 0 \begin{cases} \Delta x \longleftarrow \left[f(x,t) - g^2(t)s_\theta(x,t)\right]\Delta t + g(t)z \quad \text{où} \quad z \sim \mathcal{N}(0, |\Delta t| I) \\ x \longleftarrow x + \Delta x \\ t \longleftarrow t + \Delta t \end{cases}$$

À l'arrivée, on obtient x_0, un échantillon approximatif de la distribution des données p_0.

De plus, certaines propriétés de la SDE inverse permettent de *fine-tuner* les trajectoires obtenues avec les méthodes de résolution de SDE. En particulier, les auteurs proposent une méthode de prédiction-correction :

— Pour la prédiction, une méthode de résolution de SDE comme vue précédemment détermine un $\Delta t < 0$ et prédit un échantillon $x(t + \Delta t) \sim p_{t+\Delta t}(x)$ à partir de l'échantillon $x(t)$.

— L'échantillon $x(t + \Delta t)$ est ensuite amélioré grâce à une procédure de Monte-Carlo par chaînes de Markov (MCMC) telle qu'une dynamique de Langevin ou un algorithme de Monte-Carlo hamiltonien (HMC). En s'appuyant sur la fonction de score $s_\theta(x, t + \Delta t)$ estimée par le *score-based model*, $x(t + \Delta t)$ devient un échantillon plus fidèle de $p_{t+\Delta t}(x)$.

Avec ce *fine-tuning*, les auteurs dépassent les algorithmes de l'état de l'art sur la base de données CIFAR-10 [1]. Ils sont ainsi capables de générer des échantillons d'images de grande qualité (notamment des visages de dimensions 1024x1024).

1.3 Calcul de la log-vraisemblance exacte par l'équation différentielle ordinaire

Jusqu'à présent, on a vu comment générer de nouvelles données à partir de la SDE inverse qui utilise un *Time-Dependant Score-Based model* entrainé pour estimer le score $\nabla_x \log p_t(x)$ à chaque temps $t \in [0, T]$.

La SDE peut être transformée en ODE sans changer les distributions marginales $p_t(x)$. De manière analogue que pour les SDE, on peut générer de nouvelles données en inversant l'ODE. À noter que cette ODE ne peut être déterminée que lorsque l'on connaît la fonction de score de p_t, qui peut être estimée par le *Time-Dependant Score-Based model*. Cette ODE est nommée *Probability flow ODE* et s'écrit comme :

$$\mathrm{d}x = \left[f(x, t) - \frac{1}{2} g^2(t) \nabla_x \log p_t(x) \right] \mathrm{d}t$$

En injectant les résultats du *Time-Dependant Score-Based model* dans l'équation ci-dessus, on obtient une ODE neuronale qui s'écrit de la forme suivante :

$$\mathrm{d}x = \left[f(x, t) - \frac{1}{2} g^2(t) s_\theta(x, t) \right] \mathrm{d}t$$

Ce passage d'une SDE vers une ODE a deux principaux avantages. D'une part, il permet de réduire le temps de calcul nécessaire pour la résolution de l'équation différentielle. D'autre part, il permet le calcul exact de la vraisemblance pour n'importe quel jeu de données à partir d'une formule de changement de variable issu des ODE neuronales. Cette formule relie la densité des données avec la densité du prior via une intégrale unidimensionnelle qui peut être résolue de manière précise avec des méthodes numériques.

La formulation *probability flow ODE* permet de calculer les vraisemblances des *score-based models* sur la base CIFAR-10. Les scores FID sont légèrement moins bons que ceux obtenus de la résolution de l'équation différentielle stochastique ; néanmoins ils sont plus rapides à calculer. Le modèle dépasse alors toutes les méthodes de l'état de l'art en terme de log-vraisemblance sur CIFAR-10.

1.4 Génération d'échantillons de manière contrôlable

Un des gros avantages de l'utilisation d'équations différentielles stochastiques est la possibilité de faire de la génération conditionnelle en entraînant un seul *score-based model*, pas forcément optimisé pour une tâche spécifique. En effet, par la règle de Bayes :

1. Les scores utilisés sont la FID (*Fréchet Inception Distance*) et l'IS (*Inception Score*).

$$p_t(x|y) = \frac{p_t(x \cap y)}{p_t(y)} = \frac{p_t(x)p_t(y|x)}{\int p_t(x)p_t(y|x)\mathrm{d}x}$$

En appliquant $\nabla_x \log$, on obtient :

$$\nabla_x \log p_t(x|y) = \underbrace{\nabla_x \log p_t(x)}_{\substack{\text{Score non conditionnel} \\ \text{approché par } s_\theta(x) \\ (\textit{score matching})}} + \underbrace{\nabla_x \log p_t(y|x)}_{\text{Connu par un entraînement spécifique}}$$

Il est alors possible de calculer la fonction de score conditionnelle $\nabla_x \log p(x|y)$ et d'échantillonner à partir de cette dernière.

Un exemple très parlant est celui de la génération conditionnelle à une classe donnée : y représente le label de l'image, et $p_t(y|x)$ est un modèle de classification (dépendant du temps). Il est ainsi possible de générer des images d'une classe donnée. Les auteurs donnent d'autres exemples, comme la complétion d'images masquées ou la colorisation. Dans tous les cas, les images générées sont très variées, ce qui confirme que la notion d'incertitude est bien présente au sein de l'échantillonnage.

2 Application à la génération d'images

Le code est disponible sur ce répertoire Github et sur ce notebook Google Colab.

2.1 Génération d'images de voitures

Nous avons repris le code des auteurs, dont les notebooks sont disponibles ici. Nous l'avons appliqué à la base de données Cars, qui recense plus de 16 000 images de voitures. La première étape consiste à construire le réseau de neurones nous permettant d'estimer la fonction de score $\nabla_x \log p_t(x)$. Puis, on spécifie la SDE utilisée pour bruiter la distribution p_0 en un prior p_t. Nous reprenons celle proposée dans le notebook des auteurs :

$$\mathrm{d}x = \sigma^t \mathrm{d}w, \quad t \in [0, 1]$$

Étant donné cette SDE, une première fonction renvoie l'écart type de la distribution marginale, soit $\frac{1}{2\log\sigma}(\sigma^{2t} - 1)$, tandis qu'une deuxième calcule le coefficient de diffusion σ^t. Une fois le modèle entraîné, on dispose d'une approximation de la fonction de score. On peut alors l'injecter dans l'équation différentielle stochastique inverse. Dans notre cas, la SDE inverse s'écrit :

$$\mathrm{d}x = -\sigma^{2t}\nabla_x \log p_t(x)\mathrm{d}t + \sigma^t \mathrm{d}w$$

Cette équation peut enfin être résolue via différentes méthodes, comme la méthode classique d'Euler Maruyama, un fine-tuning des trajectoires par une méthode de prédiction-correction, ou encore en utilisant la probability flow ODE dont la résolution backward permet d'échantillonner la même distibution que la SDE inverse. Dans notre cas, l'ODE s'écrit :

$$\mathrm{d}x = -\frac{1}{2}\sigma^{2t} s_\theta(x, t)\mathrm{d}t$$

En entraînant notre modèle sur 600 epochs avec un learning rate de 0,0001, un batch size de 32, et en utilisant le générateur ODE, on obtient des échantillons représentés sur le figure 1. On aperçoit des formes de véhicules, ce qui est plutôt convaincant. La prochaine sous-section sera réservée à une évaluation quantitative du modèle.

FIGURE 1 – Génération d'images de voiture de taille 60x60 px

2.2 Mesure quantitative du modèle

Afin de mesurer la performance de leur *score-based generative model*, les auteurs utilisent deux métriques : la Frechet Inception Distance (FID) et l'Inception Score (IS). Contrairement à la FID, plus l'IS est élevé et plus les images sont réalistes. Nous proposons dans cette partie d'utiliser ce dernier pour mesurer la qualité des images de véhicules générés.

L'Inception Score, introduit en 2016 [6], est à l'origine une métrique qui quantifie la qualité des sorties des *Generative Adversarial Networks* (GANs) La plupart des papiers traitant des GANs utilisent l'IS pour montrer leur avancement par rapport aux algorithmes existants dans l'état de l'art. Par extension, on peut appliquer ce score aux *generative models*. Le score prend en compte deux aspects :

— Chaque image doit ressembler précisément et distinctement à quelque chose. Dans notre cas, tel véhicule est clairement une berline, tel autre un SUV, etc.

— La variété des images : chaque image doit par exemple être un modèle différent de voiture.

La première propriété est mesurée en passant l'image générée x au travers du classifieur *Inception* de Google. Ce dernier est notamment disponible dans la librairie Torchvision. Il sort de ce classifieur une distribution de probabilité du label y. Dans l'idéal, comme le montre la figure 2, elle doit s'éloigner le plus possible d'une distribution uniforme pour se concentrer autour d'un seul label. Quant à la deuxième caractéristique, on la mesure simplement en faisant passer un grand nombre d'images générées dans le classifieur pour faire émerger une distribution marginale $p(y|x)$. Une distribution uniforme traduit alors une grande variété d'images.

FIGURE 2 – Distribution idéale du label et distribution marginale idéale, dans le cas d'une génération d'images d'animaux

Enfin, il suffit de quantifier la différence entre ces deux distributions pour avoir un score conforme. Pour ce faire, on utilise la **Divergence de Kullback-Leibler** pour des probabilités discrètes :

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Le score pour une image est alors la KL-divergence entre sa distribution du label et la distribution marginale (ou plus exactement son exponentielle pour voir plus facilement les améliorations du score) :

$$\mathrm{IS}(x) = \exp(D_{\mathrm{KL}}(p(y|x)\|p(y)))$$

Ayant un score pour chaque image, on peut alors calculer le score moyen de l'échantillon (l'Inception Score) et l'écart-type. C'est ce qui est fait dans le notebook. Une procédure de *grid-search* a été réalisée afin de calculer les valeurs optimales de certains hyperparamètres (batch size, learning rate et méthode de sampling). Les paramètres optimaux sont identiques à ceux donnés pour générer la figure 1. On trouve un Inception Score de 2,18, ce qui est bien plus faible que le score des auteurs (9,89). Bien que les ressources techniques ne soient pas les mêmes, la conversion en échelle de gris a peut-être détérioré les performances, et nous aurions sans doute dû garder des images couleurs, qui auraient été mieux détectées par le classifieur de l'Inception Score. De la même manière, la taille des images influe peut-être sur le score : on utilise de petites images, que le classifieur a sans doute plus de mal à labéliser que des images de taille 256x256 comme celles utilisées dans l'article.

Conclusion

Au travers de ce rapport, nous avons vu comment il était possible de générer de nouveaux échantillons d'un jeu de données à partir de l'estimation de la fonction de score par un *Time-Dependant Score-based model* et de la résolution de l'équation différentielle stochastique inverse. Le modèle développé dans cet article surpasse les méthodes de l'état de l'art (notamment les GANs) dans de nombreuses tâches, comme l'IS sur des bases de données communément utilisées ou la vraisemblance grâce à l'utilisation des Probability flow ODE. De plus, les nombreuses applications des *score-based models* (génération conditionnellement à une classe donnée, colorisation, ...) en font une classe d'algorithmes incontournable. L'implémentation a permis de générer des images de véhicules de taille 60x60 pixels relativement convaincantes, et de calculer un Inception Score de 2,18 sur les échantillons créés.

Bibliographie

[1] Y. Song et al. : *Score-based generative modeling through stochastic differential equations*, 2020.

[2] B. Anderson : Reverse-time diffusion equation models, 1982.

[3] A. Hyvarinen : Estimation of non-normalized statistical models by score matching, 2005.

[4] P. Vincent : A connection between score matching and denoising autoencoders, 2011.

[5] Y. Song et al. : Sliced score matching: A scalable approach to density and score estimation, 2020.

[6] T. Salimans : Improved Techniques for Training GANs, 2016.