Normalizing Flow appliqué aux problèmes de pré-image de noyau

Clément Glédel, Benoit Gaüzère, Paul Honeine

Clément Glédel, Benoît Gaüzère, and Paul Honeine.
Normalizing flow appliqué aux problèmes de pré-image de noyau.
in 24-ème Conférence d'Apprentissage automatique (CAp) - 24th annual meeting of the francophone Machine Learning community, Vannes, France, 05 - 08 July 2022.

Normalizing Flow appliqué aux problèmes de pré-image de noyau

Clément Glédel, Benoît Gaüzère, et Paul Honeine

LITIS Lab, INSA et Université de Rouen, Rouen, France

Résumé

Dans cet article, nous proposons une approche permettant la résolution du problème de pré-image rencontré en reconnaissance de formes et apprentissage statistique notamment les méthodes à noyaux. Pour cela, nous proposons d'utiliser les récents modèles génératifs appelés Normalizing Flows (NF) qui permettent de construire une distribution simple à partir d'une distribution complexe par une série de fonctions bijectives, bénéficiant ainsi d'une efficace génération de données grâce à son inversibilité. Nous proposons d'aligner l'espace généré par le NF sur l'espace de noyau, ce qui permet de résoudre le problème de pré-image grâce à la nature inversible du NF. Les performances de la méthode proposée sont validées sur le jeu de données MNIST, démontrant la capacité des NF à résoudre efficacement le problème de pré-image.

Mots-clés : Normalizing Flow, Méthodes à noyaux, Pré-image, Reconnaissance des formes.

1 Introduction

En reconnaissance de formes et apprentissage statistique, les méthodes à noyaux opèrent un plongement des données dans un espace de plus grande dimension et plus pertinent pour résoudre le problème en question. Cet espace, implicitement créé grâce à l'utilisation d'une fonction noyau, est appelé Espace de Hilbert à Noyau Reproduisant (Reproducing Kernel Hilbert Space, RKHS).

Le problème de pré-image se pose lorsque l'on souhaite faire l'opération inverse du plongement, c'est-à-dire générer une donnée dans l'espace initial à partir d'un point de l'espace associé au noyau. Ce problème existe dans de nombreuses applications. En effet, lorsque l'on utilise des techniques de débruitage ou de compression dans un RKHS, il est nécessaire de retrouver la pré-image correspondante dans l'espace initial. Cependant, la fonction de plongement associée au noyau étant implicite avec un espace engendré de grande dimension, la transformation inverse est compliquée voir impossible à déterminer avec exactitude. Des solutions d'estimation de pré-image ont été proposées [HR11], notamment en traitement du signal [TDY+20]. Dans cet article, nous proposons une nouvelle stratégie décomposant le problème de pré-image en deux sous-problèmes. Premièrement, la définition d'une méthode générative entre un espace $\mathcal{Z}$ similaire au RKHS et notre espace initial. Deuxièmement, une méthode permettant d'optimiser le point dans $\mathcal{Z}$ à partir duquel la pré-image sera générée.

Les modèles génératifs permettent la génération de nouvelles données à partir d'échantillons tirés depuis l'espace latent associé au modèle. Pour cela, de nombreuses stratégies ont été proposées. Les modèles autorégressifs [HS97, Gra13] sont des modèles de générations simples mais non adaptés aux données de grandes dimensions. D'un autre côté, les auto-encodeurs variationnels (VAE) [KW13, KW19] se basent sur la divergence de Kullback-Leibler afin d'approximer la distribution initiale à l'aide d'une distribution paramétrique. Cependant, de part l'utilisation de bruit lors de leurs entraînements, les générations obtenues sont souvent non réalistes. Les réseaux antagonistes génératifs (GAN) [GPAM+14] proposent l'utilisation de structure conflictuelle sous forme de *min-max game* entre un générateur et un discriminateur permettant l'entraînement d'un modèle génératif sans expliciter une fonction de coût à minimiser. Ils permettent des générations performantes mais ces modèles peuvent être difficiles à entraîner de par leurs instabilités et ne définissent pas un espace latent de représentation. Enfin, les Normalizing Flows (NF) [KPB20] sont des méthodes génératives basées sur la composition d'une ou plusieurs fonctions bijectives inversibles [CRBD18, DKB14, DSDB16, KD18]. Celles-ci permettent une correspondance exacte entre deux distributions, la distribution initiale et la distribution générée, ainsi qu'une transformation d'un espace à l'autre immédiate via l'ensemble des fonctions précitées. Contrairement aux

GAN [GPAM+14] qui modélisent implicitement leur espace latent, les méthodes NF sont entraînées en utilisant une fonction de coût explicite permettant un entraînement plus simple et plus stable.

Dans cet article, nous proposons de tirer parti des récentes avancées sur les NF, notamment de leur capacité d'inversibilité exacte, afin de proposer une nouvelle stratégie pour la résolution du problème de pré-image lié aux méthodes à noyaux. Pour ce faire, notre méthode repose sur la maximisation de l'alignement entre l'espace généré par le NF et l'espace RKHS associé au noyau utilisé. Couplé avec une méthode à noyaux, le NF permet ainsi de générer une solution au problème de pré-image. De plus, notre modèle d'alignement agissant dans l'espace du NF permet l'amélioration de la solution de pré-image. Précisons que la méthode proposée est générique, permettant d'injecter toute méthode de NF dans le schéma proposé. Dans les expérimentations réalisées, nous montrons l'efficacité de notre approche sur le jeu de données MNIST, en intégrant 3 méthodes de NF de références et utilisant différents formalismes : Glow [KD18], FFJORD [GCB+18] et OT-Flow [OWFLR21].

L'article est organisé comme suit. La section 2 introduit les concepts préliminaires relatifs aux NF nécessaires à la compréhension des modèles utilisés dans le papier. Ensuite, nous présentons le problème de pré-image (section 3). Puis, notre méthode est présentée dans la section 4, en détaillant le modèle génératif (section 4.1), le modèle d'alignement (section 4.2) et la procédure d'optimisation (section 4.3). Enfin, nous concluons par l'évaluation de notre approche (section 5).

2 Préliminaires sur les Normalizing Flows

Un NF décrit une transformation inversible $f : \mathcal{X} \to \mathcal{Z}$ entre une distribution complexe $P_{\mathcal{X}}(x)$ que l'on souhaite approximer et une distribution $P_{\mathcal{Z}}(z)$, souvent choisie simple telle qu'une distribution gaussienne. Une telle approche permet une génération rapide de donnée $x \in \mathcal{X}$ en tirant un élément $z \in \mathcal{Z}$ puis en appliquant la fonction inverse du NF, i.e., $x = f^{-1}(z)$.

Les NF s'appuient sur la formule de changement de variable permettant de définir la relation entre deux densités de probabilité :

$$P_{\mathcal{X}}(x) = P_{\mathcal{Z}}(z) \left|\det \left(\frac{\partial z}{\partial x}\right)\right| = P_{\mathcal{Z}}(f(x)) \left|\det \left(\frac{\partial f(x)}{\partial x}\right)\right|, \quad (1)$$

où $\frac{\partial f(x)}{\partial x}$ correspond à la Jacobienne associée à la fonction f. Le déterminant de $\frac{\partial f(x)}{\partial x}$ représente le facteur de déformation entre les deux distributions. Les NF ont l'avantage d'utiliser l'Eq. (1) qui décrit la relation exacte entre les deux distributions. Cet aspect permet, contrairement aux VAE, de ne pas définir le problème d'optimisation à partir d'une borne inférieure [KW13, KW19] mais selon l'exacte vraisemblance. Cependant, afin de pouvoir l'exploiter, l'architecture d'un NF doit respecter certaines contraintes. En effet, le modèle doit être facile à inverser et le déterminant de la Jacobienne doit être calculable facilement. Par souci de dérivation, l'Eq. (1) est transformée pour obtenir une fonction du log-vraisemblance à minimiser lors de l'entraînement du modèle :

$$\log P_{\mathcal{X}}(x) = \log P_{\mathcal{Z}}(f(x)) + \log \left|\det \left(\frac{\partial f(x)}{\partial x}\right)\right|. \quad (2)$$

Les NF définissent la transformation f de $P_{\mathcal{X}}$ vers $P_{\mathcal{Z}}$ par une composition de fonctions f_i inversibles permettant ainsi à la fonction englobante f d'être inversible. La transformation f est définie par :

$$f = f_1 \circ f_2 \circ \cdots \circ f_\ell,$$

avec ℓ le nombre de fonctions bijectives utilisées. De plus, le déterminant de la Jacobienne de f devient le produit de ceux de f_i. Motivé par ces points, plusieurs approches de NF ont été proposées. Parmi toutes les approches proposées, celles basées sur des *affine coupling layers* [DKB14, DSDB16] consistent à obtenir une forme particulière de la Jacobienne en divisant l'entrée x de dimension d en deux parties de taille égale $d/2$. Nous noterons $x_{1:d/2}$ et $x_{d/2+1:d}$ celles-ci. Ainsi, f_i est définie par $f_i(x) = z$ avec :

$$\begin{aligned} z_{1:d/2} &= x_{1:d/2} \\ z_{d/2+1:d} &= x_{d/2+1:d} \odot \exp(s(x_{1:d/2})) + t(x_{1:d/2}) \end{aligned}$$

où s et t sont des fonctions non linéaires sous forme de réseaux de neurones et $\odot$ est le produit de Hadamard.

Grâce à la fonction identité appliquée sur la première partie du vecteur, chaque fonction f_i est facilement inversible par $f_i^{-1}(z) = x$ avec :

$$\begin{aligned} x_{1:d/2} &= z_{1:d/2} \\ x_{d/2+1:d} &= (z_{d/2+1:d} - t(x_{1:d/2})) \oslash \exp(s(x_{1:d/2})) \end{aligned}$$

où $\oslash$ est la division de Hadamard.

Ainsi, l'utilisation d'une telle structure permet aux transformations f_i d'obtenir une matrice Jacobienne triangulaire inférieure [DKB14, DSDB16]. Cette particularité permet de calculer facilement le déterminant

de la Jacobienne de f nécessaire à la transformation décrite par l'Eq. (1) :

$$\det\left(\frac{\partial f(x)}{\partial x}\right) = \prod_{i=1}^{\ell} \det\left(\frac{\partial f_i(x)}{\partial x_i}\right)$$

Il existe de nombreux modèles utilisant les *affine coupling layers*, un des plus connus étant Glow [KD18] qui applique celle-ci sur des données de type image. Une telle approche est efficace à condition de se restreindre à une architecture inversible et permettant un calcul facile du déterminant.

Les Continuous Normalizing Flows (CNF) se libèrent de ces contraintes en obtenant f grâce à la résolution d'équations différentielles ordinaires (ODE) [CRBD18]. Ils décrivent les NF comme des transformations continues et définissent $z(x,t)$ comme une trajectoire $z : \mathbb{R}^d \times t \to \mathbb{R}^d$ où $t \in [t_0, t_1]$ correspond au temps, $z(x,t_0) = z$ avec $z \in \mathcal{Z}$ et $z(x,t_1) = x$ avec $x \in \mathcal{X}$. Ils utilisent la formule de changement instantané de variable [CRBD18] pour remplacer le calcul de déterminant de la Jacobienne par l'opération de trace de la Jacobienne. Les CNF permettent des estimations de densité performantes mais ont un coût. En effet, l'utilisation de solveur d'ODE est nécessaire et peut être coûteux selon le nombre d'évaluations requis [CRBD18]. De plus, le calcul de trace sans Jacobienne peut être difficile et représente un défi à résoudre. FFJORD [GCB+18] est un CNF introduisant l'utilisation de l'estimateur de l'opérateur de trace de Hutchinson [Hut89]. Ceci facilite le calcul de trace et permet une extension plus directe à des données de plus grande dimension. Plus récemment, l'utilisation du concept de Transport Optimal (OT) a été proposée par RNODE [FJNO20] et OT-Flow [OWFLR21] permettant des trajectoires minimales et sans intersection. De plus, OT-Flow calcule exactement la trace de la Jacobienne avec une complexité comparable aux estimateurs permettant une meilleure convergence. Cette approche apporte une réduction du nombre d'étapes sans perte de performance.

Dans cet article, nous proposons l'association de ces modèles génératifs aux méthodes à noyaux pour la résolution du problème de pré-image.

3 Le problème de pré-image

Pour $\mathcal{X}$ l'espace des données avec $x_i, x_j \in \mathcal{X}$, soit $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ un noyau semi-défini positif [STC+04]. Ainsi, il existe un unique RKHS $\mathcal{H}$ engendré par une fonction implicite Φ associée au noyau avec $\kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}}$ où $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ correspond au produit scalaire dans $\mathcal{H}$. L'astuce du noyau consiste à remplacer l'intégralité des accès aux données d'apprentissage par des appels à la fonction noyau. Ainsi, la méthode d'apprentissage, généralement défini comme une méthode linéaire, peut opérer implicitement dans l'espace $\mathcal{H}$ associé au noyau, et transcrire une décision non linéaire dans l'espace initial.

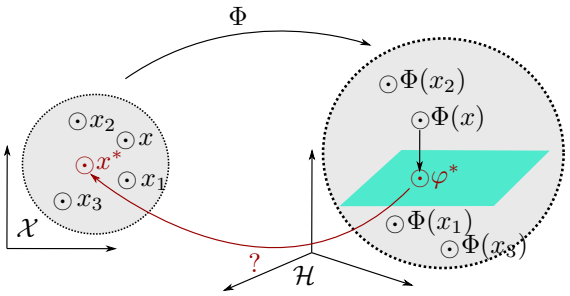


FIGURE 1 – Illustration du problème de pré-image des méthodes de noyaux, qui consiste à trouver $x^* \in \mathcal{X}$ dont l'image dans $\mathcal{H}$ est la plus proche possible de φ^*.

En reconnaissance de formes, notamment en débruitage et compression, il est d'un grand intérêt de revenir à l'espace initial, afin de pouvoir extraire les résultats et les interpréter dans l'espace des données, comme c'est le cas par exemple en apprentissage de dictionnaires non linéaires [ZH17, TDY+20]. Un aperçu du problème de pré-image est illustré dans la Figure 1. Ainsi, le but est d'obtenir la pré-image $x^* \in \mathcal{X}$ d'un élément $\varphi^* \in \mathcal{H}$. Une approche triviale serait simplement d'appliquer la transformation inverse de Φ à φ^*, c'est-à-dire $x^* = \Phi^{-1}(\varphi^*)$. Toutefois, la fonction Φ de plongement est généralement implicite, et l'espace $\mathcal{H}$ engendré par le noyau peut être de très grande dimension. Ainsi, la résolution exacte de ce problème n'est généralement pas possible [HR09]. Par conséquent, nous définissons une version relaxée de ce problème. Le problème de pré-image consiste à trouver $x^* \in \mathcal{X}$ qui, après projection dans $\mathcal{H}$, soit le plus proche possible de φ^*, c'est-à-dire

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} \|\Phi(x) - \varphi^*\|^2_{\mathcal{H}}, \tag{3}$$

où $\|\cdot\|_{\mathcal{H}}$ désigne la norme dans $\mathcal{H}$.

De façon générale, le théorème de représentation [STC+04] permet de définir φ^* comme une combinaison linéaire des projections des données d'entraî-

nement dans $\mathcal{H}$, désignées par $x_1, x_2, \ldots, x_N$:

$$\varphi^* = \sum_{i=1}^{N} \alpha_i \Phi(x_i), \tag{4}$$

où α_i correspond au facteur d'importance de l'échantillon x_i. Dans ce cas, la distance quadratique à minimiser pour le problème de pré-image (Eq. (3)) peut se réécrire :

$$\begin{aligned} d^2_{\mathcal{H}} &= \|\Phi(x) - \varphi^*\|^2_{\mathcal{H}} \\ &= \langle \Phi(x) - \varphi^*, \Phi(x) - \varphi^* \rangle_{\mathcal{H}} \\ &= \kappa(x,x) - 2\sum_{i=1}^{N} \alpha_i \kappa(x, x_i) + \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \kappa(x_i, x_j). \end{aligned} \tag{5}$$

4 Approche proposée

Notre approche a pour but de générer une pré-image x^* qui satisfait l'Eq. (3). Afin de trouver une potentielle pré-image, une première solution triviale consiste à sélectionner de l'ensemble d'entraînement la donnée x_d minimisant (5), c'est-à-dire

$$x_d = \underset{x \in \{x_1, \cdots, x_N\}}{\operatorname{argmin}} \|\Phi(x) - \varphi^*\|^2_{\mathcal{H}}. \tag{6}$$

Cependant, cette méthode nous restreint aux données présentes dans le jeu de données.

Pour palier cela, nous proposons d'utiliser une méthode générative associée à un espace latent $\mathcal{Z}$ à partir duquel la génération de données de $\mathcal{X}$ est possible. Un tel espace permet la correspondance entre chaque $z_i \in \mathcal{Z}$ avec chaque $x_i \in \mathcal{X}$ et donc chaque $\Phi(x_i) \in \mathcal{H}$. Avec cela, résoudre le problème de pré-image consiste à sélectionner un élément de z telle que l'élément x associé minimise l'Eq. (3).

Ainsi, nous utilisons comme méthode générative un NF, noté f. Comme décrit en section 2, les NF permettent d'apprendre des fonctions inversibles entre l'espace $\mathcal{X}$ et un espace latent $\mathcal{Z}$. De par le caractère inversible de f, la génération d'une nouvelle donnée $x \in \mathcal{X}$ se fait directement en calculant $f^{-1}(z)$, pour $z \in \mathcal{Z}$. Une fois la méthode générative explicitée, il reste à définir l'élément $z \in \mathcal{Z}$ à partir duquel la pré-image sera générée. La sélection de celui-ci peut se faire d'une première façon en calculant l'équivalent de l'Eq. (4) dans $\mathcal{Z}$ suivant $z = \sum_{i=1}^{N} \alpha_i z_i$.

Toutefois, sachant que l'espace $\mathcal{Z}$ généré par le NF est généralement non aligné avec l'espace $\mathcal{H}$ associé au noyau, nous proposons une méthode permettant d'améliorer l'estimation du point z^* permettant la génération de la pré-image. Pour cela, nous utilisons une fonction d'alignement $g_\Theta : \mathcal{Z} \to \mathcal{H}'$ dont l'espace engendré est isométrique à $\mathcal{H}$. Celle-ci est modélisée par un réseau de neurones paramétré par Θ que l'on souhaite déterminer. Une fois g_Θ entraîné, celui-ci permet d'optimiser un point $z \in \mathcal{Z}$, pour que sa pré-image par f^{-1} dans $\mathcal{X}$ soit le plus proche possible de la pré-image de φ^* par Φ^{-1}.

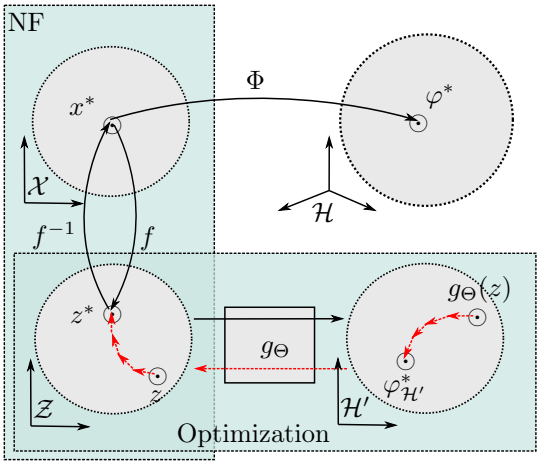


FIGURE 2 – Schéma de l'approche proposée décrivant la relation de l'espace initial $\mathcal{X}$ au noyau, le bloc génératif avec NF ainsi que le bloc d'optimisation de $z \in \mathcal{Z}$ par le modèle d'alignement g_Θ.

Un aperçu de l'approche est présenté dans la Figure 2. Le schéma décrit les différentes relations entre les éléments de notre méthode :

- Φ la fonction implicite de plongement associée au noyau κ représente le lien entre l'espace des données $\mathcal{X}$ et le RKHS $\mathcal{H}$, voir section 3.
- Le bloc NF associe l'espace des données $\mathcal{X}$ à l'espace du NF $\mathcal{Z}$ permettant la projection de $z \in \mathcal{Z}$ vers $x \in \mathcal{X}$ par f^{-1} et inversement, voir section 4.1.
- Enfin, le bloc d'optimisation aligne l'espace du NF $\mathcal{Z}$ à l'espace RKHS $\mathcal{H}$, générant ainsi un espace $\mathcal{H}'$, voir sections 4.2 et 4.3 pour plus de détails.

4.1 Modèle génératif

Nous proposons d'utiliser un NF $f : \mathcal{X} \to \mathcal{Z}$ comme modèle génératif de notre approche, permettant la définition d'un espace latent $\mathcal{Z}$ de notre espace de données $\mathcal{X}$. Nous souhaitons avoir une approche générique

indépendante de l'espace $\mathcal{X}$ utilisé. Cependant, celui-ci peut être différent selon le domaine ou le problème posé. En effet, il n'est pas rare de travailler avec des espaces $\mathcal{X}$ de données euclidiennes, séquentielles, textuelles ou de séries temporelles. De plus, nous pouvons aussi être confrontés à des espaces discrets lorsque l'on utilise, par exemple, des données de type graphe. De la même manière, il existe des noyaux différents dépendant du type d'espace $\mathcal{X}$, comme par exemple les noyaux de graphes [JGH22].

Un NF est un modèle permettant la génération d'un espace continu $\mathcal{Z}$ suivant une distribution simple telle qu'une loi normale. Ainsi, en utilisant un NF travaillant sur l'espace $\mathcal{X}$ en question, nous pouvons générer un espace $\mathcal{Z}$ continu, simple et interprétable quelque soit l'espace qu'il transforme. Ainsi, notre approche vise à avoir la liberté du choix d'utiliser tout type d'architecture de NF (Glow [KD18], FFJORD [GCB+18], OT-Flow [OWFLR21], etc.) en fonction de la nature de l'espace $\mathcal{X}$. Pour ce faire, aucune modification n'est apportée au modèle de NF utilisé, et son entraînement est en tout point similaire à ce qui est décrit dans l'état de l'art.

4.2 Modèle d'alignement

L'utilisation du NF permet une génération simple et efficace de données dans $\mathcal{X}$ à partir d'éléments de $\mathcal{Z}$. Cependant, étant donné que l'espace $\mathcal{Z}$ est généralement non aligné avec $\mathcal{H}$, la transposition de la définition d'un élément $\varphi^* \in \mathcal{H}$ sous forme d'une combinaison linéaire des projections des données d'entraînements (Eq. (4)) dans l'espace $\mathcal{Z}$ par $z = \sum_{i=1}^{N} \alpha_i z_i$ peut être incohérent et ne pas répondre à la problématique initiale.

Pour cela, nous proposons l'apprentissage d'un modèle d'alignement g_Θ qui transforme l'espace $\mathcal{Z}$ en un espace "aligné" avec l'espace $\mathcal{H}$. Soit $\mathcal{H}'$ cet espace, nous avons alors :

$$g_\Theta \colon \mathcal{Z} \to \mathcal{H}'.$$

Par alignement, nous souhaitons que l'espace $\mathcal{H}'$ détienne des relations inter-données similaires à celles dans l'espace du noyau $\mathcal{H}$, permettant ainsi une transposition plus cohérente de φ^* (Eq. (4)) dans $\mathcal{H}'$. Plus spécifiquement, nous souhaitons une isométrie entre les deux espaces, c'est-à-dire que les produits scalaires soient similaires dans les deux espaces. À partir des données d'entraînement, $x_1, x_2, ...x_N$, et de leurs images par f dans $\mathcal{Z}$, $z_1, z_2, \cdots, z_N$, l'alignement est défini par

$$\langle g_\Theta(z_i), g_\Theta(z_j)\rangle_{\mathcal{H}'} \approx \kappa(x_i, x_j), \forall i, j \in \{1, 2, \ldots, N\}. \tag{7}$$

où $\langle \cdot, \cdot \rangle_{\mathcal{H}'}$ désigne le produit scalaire dans $\mathcal{H}'$.

Ainsi, pour l'entraînement de g_Θ, nous proposons la minimisation de la perte suivante, pour tous les $i, j = 1, 2, \ldots, N$:

$$\mathcal{L}_e(\Theta, i, j) = \left(\langle g_\Theta(z_i), g_\Theta(z_j)\rangle_{\mathcal{H}'} - \kappa(x_i, x_j)\right)^2 \tag{8}$$

Nous pouvons alors définir la fonction de coût à minimiser comme la moyenne des pertes, introduites par l'Eq. (8), pour chaque paire de données d'entraînement :

$$J_e(\Theta) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{L}_e(\Theta, i, j) \tag{9}$$

Le modèle g_Θ doit être une fonction différentiable paramétrée par des paramètres Θ. Celui-ci doit être défini sur l'espace latent $\mathcal{Z}$. En effet, l'architecture du réseau de neurone g_Θ peut être différente selon le NF et l'espace des données $\mathcal{X}$. Ainsi, si nous travaillons sur des données structurées de type image et si le NF choisi a la propriété de préserver cette structure dans $\mathcal{Z}$, e.g. Glow [KD18], alors il peut être préférable d'utiliser un réseau convolutionnel de type ResNet [HZRS16]. Sinon, un perceptron multicouches (MLP) peut être utilisé. L'entraînement de celui-ci repose sur une stratégie de rétropropagation du gradient calculé sur le coût précédent. Ainsi les poids $\theta^{(j)} \in \Theta$ sont optimisés de la manière suivante :

$$\theta^{(j)}_{\text{new}} = \theta^{(j)}_{\text{old}} - \eta \frac{\partial J_e}{\partial \theta^{(j)}_{\text{old}}}$$

avec η le taux d'apprentissage prédéfini.

Une telle méthode d'entraînement ne nous impose aucune contrainte sur l'espace $\mathcal{H}'$ et sa dimension. Nous décidons d'utiliser un espace $\mathcal{H}'$ de même nature que l'espace $\mathcal{Z}$ du NF, ce qui permet d'avoir une autre interprétation de g_Θ. Ainsi, la fonction g_Θ peut être considérée comme une fonction de déformation de $\mathcal{Z}$ pour l'aligner à $\mathcal{H}$.

4.3 Optimisation de la pré-image

Comme introduit section 4, nous cherchons, à partir de z, l'élément z^* qui après génération dans l'espace $\mathcal{X}$, ait une projection dans $\mathcal{H}$ égale à φ^*, i.e $\Phi(x^*) = \varphi^*$. Ainsi, le modèle d'alignement g_Θ a pour but d'optimiser un élément $z \in \mathcal{Z}$ vers z^* permettant la génération de la meilleure pré-image $x^* = f^{-1}(z^*)$. Le schéma présenté dans la Figure 2 permet une visualisation de

la méthode d'optimisation. Après initialisation d'un z, nous utilisons g_Θ pour optimiser itérativement z vers z^*. Sachant l'Eq. (5), nous pouvons définir le problème à minimiser permettant d'obtention de la pré-image désirée de la manière suivante :

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} \ \kappa(x,x) - 2\sum_{i=1}^{N} \alpha_i \kappa(x, x_i). \tag{10}$$

La résolution directe de ce problème d'optimisation [HR11] est contraint par plusieurs limitations, notamment le noyau doit être différentiable et l'espace d'optimisation continu. Cependant, en général ce n'est pas le cas comme par exemple avec les graphes [JGH21].

Pour pallier à ce problème, le modèle g_Θ permet d'avoir une isométrie entre les espaces $\mathcal{H}$ et $\mathcal{H}'$ selon l'Eq. (7). Sachant cela, nous pouvons remplacer les termes dépendants de x en utilisant g_Θ. Par conséquent, nous pouvons transposer notre problème dans $\mathcal{H}'$ et reformuler le problème de l'Eq.(10) sous la forme d'une fonction de coût à minimiser de la façon suivante :

$$J_o = \langle g_\Theta(z), g_\Theta(z) \rangle_{\mathcal{H}'} - 2\sum_{i=1}^{N} \alpha_i \langle g_\Theta(z), g_\Theta(z_i) \rangle_{\mathcal{H}'}. \tag{11}$$

Nous souhaitons agir sur l'entrée z qui est notre cible à optimiser. Pour cela, nous nous sommes inspiré par la littérature sur l'optimisation de propriété de molécule faite dans [GBWD$^+$18, KPHL17, JBJ18, DTD$^+$18, YLY$^+$18, PSOI19]. Le modèle d'alignement étant une fonction différentiable, nous pouvons ainsi optimiser les paramètres de z en rétropropagant le gradient calculé sur le coût (Eq. (11)) :

$$z_{\text{new}}^{(j)} = z_{\text{old}}^{(j)} - \eta \frac{\partial J_o}{\partial z_{\text{old}}^{(j)}},$$

avec η le taux d'apprentissage défini et $z^{(j)} \in z$ représentant le paramètre j de z à optimiser. De cette façon, nous pouvons optimiser z vers z^* de manière à obtenir une meilleure pré-image avec $f^{-1}(z^*)$.

5 Évaluation

5.1 Configuration

Afin d'évaluer notre approche, nous utilisons le jeu de données MNIST [Den12]. Ce jeu de données est constitué de 60 000 images de chiffres manuscrits, chaque image étant de taille 28×28 pixels. Ce jeu de données standard nous permettra d'évaluer la pertinence de notre approche de génération de pré-image.

Espace	Glow	FFJORD	OT-Flow
$\mathcal{Z}$	0,1828	0,0452	0,1306
$\mathcal{H}'$	8,62e-5	0,0001	6,20e-5

TABLE 1 – Évaluation de l'alignement des espaces latent $\mathcal{Z}$ (Eq. (12)) et générés $\mathcal{H}'$ avec g_Θ (Eq. (9))

À des fins d'expérimentations, nous utilisons un noyau gaussien classique $\kappa(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2)$, avec $\sigma = 2500$. Celui-ci obtient 96% de bonne classification sur un sous-ensemble du jeu de données MNIST composé de 1000 images par classe, démontrant ainsi la pertinence du noyau. L'évaluation du noyau se fait avec un ensemble de validation constitué de 10% des données de chaque classe, soit 100 images par classe. Nous noterons toutefois que n'importe quel noyau semi défini positif pourra être employé.

L'architecture de NF à utiliser étant libre, voir section 4.1, nous avons choisi d'utiliser Glow [KD18], FFJORD [GCB$^+$18] et OT-Flow [OWFLR21], étant des architectures pouvant travailler sur des images. Comme spécifié dans la section 4.2, l'architecture du modèle d'alignement g_Θ est différente selon le NF utilisé. En effet, Glow produisant des sorties à multiples échelles tout en gardant la structure des images (e.g. couche, largeur, hauteur), nous avons choisi une architecture de réseau résiduel (ResNet) avec convolution. Pour FFJORD et OT-Flow, de simples perceptrons multicouches (MLP) constitués de deux couches cachées ont été utilisés.

5.2 Alignement

Chaque modèle d'alignement est entraîné comme indiqué dans la section 4.2. Ainsi, afin d'estimer la pertinence de nos modèles, nous évaluons l'alignement de l'espace avant optimisation $\mathcal{Z}$ et après optimisation $\mathcal{H}'$. L'évaluation de $\mathcal{Z}$ est obtenu avec :

$$\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\langle z_i, z_j \rangle_{\mathcal{Z}} - \kappa(x_i, x_j))^2 \tag{12}$$

Ensuite, l'alignement de l'espace après optimisation $\mathcal{H}'$ est calculé avec le coût J_e suivant l'Eq. (9). Les résultats sont visibles dans la Table 1. En effet, nous pouvons constater que nos modèles g_Θ permettent l'ajustement des espaces de NF $\mathcal{Z}$ à $\mathcal{H}$ par la génération de l'espace $\mathcal{H}'$.

5.3 Génération

Afin d'évaluer notre approche à générer des pré-images cohérentes pour un problème donné, nous nous plaçons dans le cas où φ^* est le centroïde de k échantillons dans l'espace $\mathcal{H}$, comme c'est souvent le cas en filtrage ou en approximation [JGH21], par exemple avec les k plus proches voisins. Ainsi, pour chaque expérimentation, nous tirons aléatoirement un ensemble de k plus proches voisins dans $\mathcal{H}$. Par conséquent nous posons :

$$\alpha_i = \begin{cases} 1/k, & \text{si } \Phi(x_i) \in k \text{ plus proches voisins} \\ 0, & \text{sinon} \end{cases}$$

Dans un second temps, comme indiqué dans la section 4, une première estimation triviale de la pré-image est effectuée en sélectionnant la meilleure donnée parmi l'ensemble des données du jeu de données (Eq. (6)). Ainsi, nous effectuons une recherche exhaustive dans l'ensemble d'entraînement de la donnée minimisant l'Eq. (5). La pré-image issue du jeu de données correspondante, notée x_d, représente à la fois la meilleure pré-image présente dans notre dataset et notre point de comparaison.

5.3.1 Génération de la pré-image

Ensuite, nous souhaitons savoir si la pré-image générée depuis $\mathcal{Z}$ permet d'obtenir une distance $d^2_{\mathcal{H}}$ dans $\mathcal{H}$ plus faible et donc une meilleure pré-image que x_d. Pour cela, nous générons tout d'abord une pré-image notée x_{nf}, sans tenir compte d'une possible différence de relation inter-données dans $\mathcal{Z}$ et $\mathcal{H}$ comme indiqué dans la section 4. Celle-ci est obtenue avec $z_{\text{nf}} = \sum_{i=1}^{N} \alpha_i z_i$, puis en générant la pré-image $x_{\text{nf}} = f^{-1}(z_{\text{nf}})$. Figure 4 nous permet de comparer x_d avec les x_{nf} générés pour chaque modèle en termes de $d^2_{\mathcal{H}}$ (Eq. (5)). Pour plus de précision, ces valeurs sont indiquées dans Table 2. Les résultats sont obtenus en moyennant 100 simulations pour chaque k avec $2 \leq k \leq 15$. Ceux-ci mettent en évidence de très bons résultats avec le modèle d'OT-Flow, où les $d^2_{\mathcal{H}}$ calculés avec x_{nf} sont très inférieurs à ceux calculés avec x_d. Pour la génération avec Glow, nous pouvons constater une amélioration de la pré-image dès lors où l'on utilise un $k > 4$. Enfin, les pré-images obtenues avec le modèle FFJORD, quant à eux, ne correspondent pas au noyau utilisé, ceci étant mis en évidence par les hautes valeurs de $d^2_{\mathcal{H}}$ obtenues. De tels résultats permettent d'évaluer la ressemblance entre $\mathcal{Z}$ et $\mathcal{H}$. En effet, l'espace $\mathcal{H}$ généré par κ est donc très similaire à l'espace $\mathcal{Z}$ généré par OT-Flow, assez ressemblant à celui de Glow, et très loin de celui de FFJORD. De plus, nous constatons que le nombre de k plus proches voisins à peu d'influence sur la génération de la pré-image.

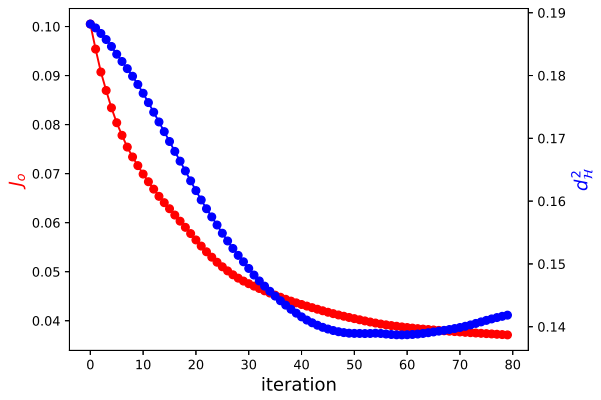
		Glow		FFJORD		OT-Flow	
kppv	x_d	x_{nf}	x_{opti}	x_{nf}	x_{opti}	x_{nf}	x_{opti}
2	0,1451	0,1748	0,1727	0,8215	0,7222	0,0304	0,0300
3	0,1673	0,1895	0,1865	0,8310	0,7473	0,0372	0,0367
4	0,1869	0,1889	0,1858	0,8562	0,7675	0,0442	0,0439
5	0,1956	0,1854	0,1824	0,8270	0,7473	0,0480	0,0475
6	0,2017	0,1883	0,1845	0,7841	0,6977	0,0512	0,0505
7	0,2029	0,1848	0,1817	0,8080	0,7221	0,0480	0,0476
8	0,1950	0,1775	0,1742	0,7738	0,6941	0,0462	0,0459
9	0,2131	0,1871	0,1831	0,8053	0,7145	0,0534	0,0525
10	0,2073	0,1826	0,1794	0,7856	0,6980	0,0533	0,0526
11	0,2172	0,1888	0,1858	0,8165	0,7257	0,0542	0,0537
12	0,2116	0,1819	0,1784	0,7643	0,6931	0,0541	0,0539
13	0,2098	0,1818	0,1777	0,8023	0,7105	0,0525	0,0519
14	0,2111	0,1769	0,1737	0,7710	0,6894	0,0505	0,0499
15	0,2136	0,1783	0,1748	0,7731	0,6861	0,0564	0,0553

TABLE 2 – Moyenne sur 100 simulations de Monte Carlo de $d^2_{\mathcal{H}}$ en fonction du nombre de k plus proches voisins sur les images du chiffre 3 de MNIST

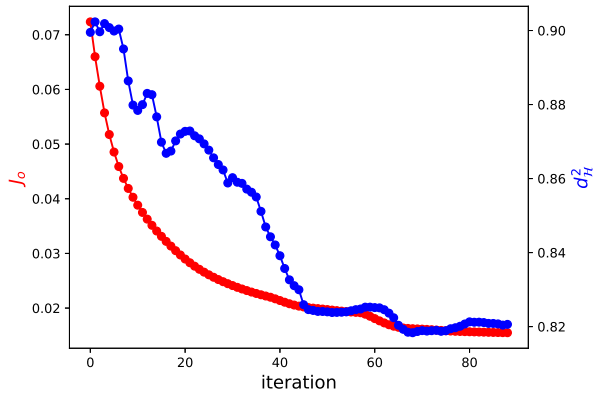
5.3.2 Optimisation de la pré-image

Enfin, nous souhaitons connaître dans quelle mesure, en terme de $d^2_{\mathcal{H}}$ (Eq. (5)), notre procédure d'optimisation, vue dans les sections 4.2 et 4.3, permet d'améliorer la qualité de la pré-image. Pour cela, nous utilisons z_{nf} comme entrée à optimiser et évaluons une nouvelle valeur suivant l'Eq. (11). Des aperçus d'optimisation comparant l'évolution des pertes à minimiser J_o (Eq. (11)) avec le calcul de $d^2_{\mathcal{H}}$ (Eq. (5)) à chaque itération est visible dans la Figure 3. La pré-image atteignant la plus faible valeur de $d^2_{\mathcal{H}}$ est notée x_{opti}.

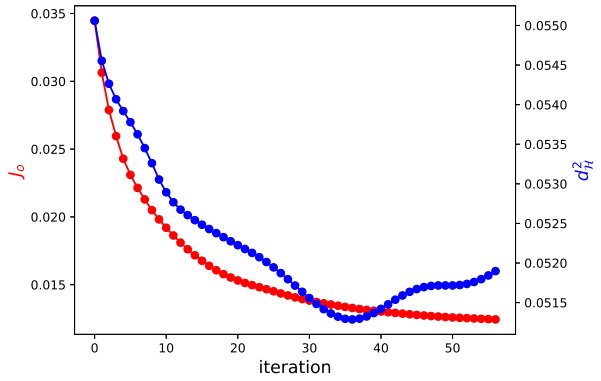
Après avoir généré x_{nf} et x_{opti} pour chacun des trois modèles, nous pouvons afficher leurs valeurs de $d^2_{\mathcal{H}}$ associées, voir Figure 4. Ainsi, nous pouvons voir que pour OT-Flow et Glow, les résultats de $d^2_{\mathcal{H}}$ avec optimisation ne sont pas ou peu éloignés des valeurs sans optimisation, voir Table 2. Dans le cas de FFJORD, l'optimisation fonctionne mieux mais x_{opti} reste une moins bonne pré-image que x_d. En effet, meilleures sont les pré-images sans optimisation, plus faible est le gain après optimisation. Enfin, nous constatons que le nombre de k plus proches voisins n'a pas d'influence sur les résultats d'optimisation.



(a) Glow



(b) FFJORD



(c) OT-Flow

FIGURE 3 – Évolution des pertes à minimiser J_o (Eq. (11), en rouge) avec le calcul de $d^2_{\mathcal{H}}$ (Eq. (5), en bleu) à chaque itération d'optimisation. Ces courbes représentent l'optimisation pour 1 simulation avec $k = 8$ pour chacun des modèles d'alignement associés aux NF Glow, FFJORD et OT-Flow

6 Conclusion et perspectives

Dans cet article, nous avons proposé une approche utilisant les NF pour la résolution du problème de pré-image lié aux méthodes à noyaux. Nous avons proposé une méthode d'alignement permettant de rechercher dans l'espace latent associé au NF une donnée qui, après génération, est une meilleure pré-image. Les résultats expérimentaux ont montré que certains NF, tel que OT-Flow, permettent une génération de pré-image très performante.

Étant donné que les méthodes à noyaux détiennent des RKHS qui peuvent être différents de celui d'un NF, voir le modèle FFJORD dans l'expérimentation précédente, nous proposons d'orienter l'apprentissage d'un NF pour qu'il corresponde à l'espace engendré par un noyau.

7 Remerciements

Ce travail est soutenu par le projet APi « Apprivoiser la pré-image » (ANR-18-CE23-0014).

Références

[CRBD18] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv :1806.07366*, 2018.

[Den12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6) :141–142, 2012.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice : Non-linear independent components estimation. *arXiv preprint arXiv :1410.8516*, 2014.

[DSDB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv :1605.08803*, 2016.

[DTD+18] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv :1802.08786*, 2018.

[FJNO20] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode : the world of jacobian and kinetic regularization.

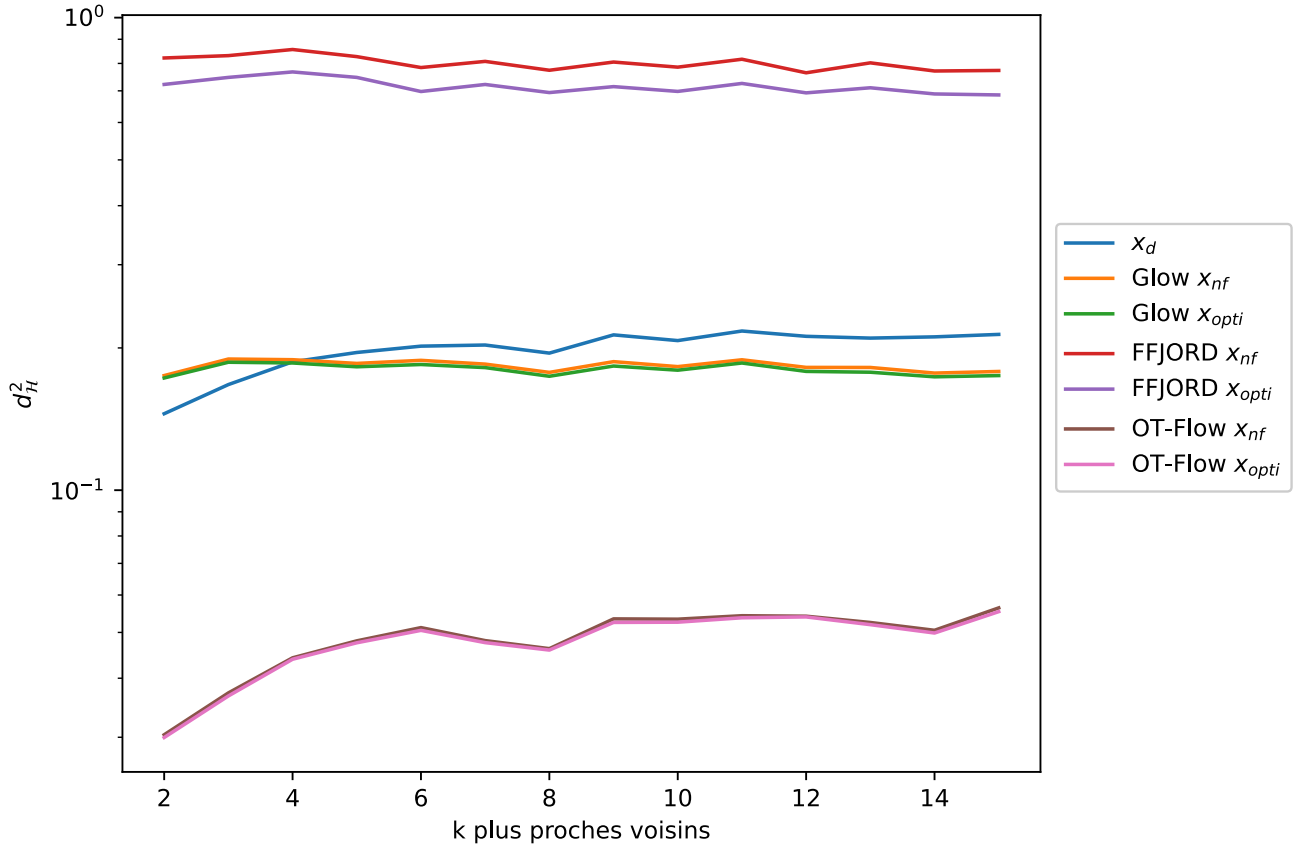


FIGURE 4 – Moyenne sur 100 simulations de Monte Carlo de $d^2_{\mathcal{H}}$ en fonction du nombre k de plus proches voisins sur les images du chiffre 3 de MNIST.

In *International conference on machine learning*, pages 3154–3164. PMLR, 2020.

[GBWD+18] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2) :268–276, 2018.

[GCB+18] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord : Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv :1810.01367*, 2018.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[Gra13] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv :1308.0850*, 2013.

[HR09] Paul Honeine and Cédric Richard. Solving the pre-image problem in kernel machines : A direct method. In *2009 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2009.

[HR11] Paul Honeine and Cédric Richard. Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine*, 28(2) :77 – 88, March 2011.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural*

computation, 9(8) :1735–1780, 1997.

[Hut89] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3) :1059–1076, 1989.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JBJ18] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv :1802.04364*, 2018.

[JGH21] Linlin Jia, Benoit Gaüzère, and Paul Honeine. A graph pre-image method based on graph edit distances. In *Proceedings of IAPR Joint International Workshops on Statistical techniques in Pattern Recognition (SPR 2020) and Structural and Syntactic Pattern Recognition (SSPR 2020).*, 2021.

[JGH22] Linlin Jia, Benoit Gaüzère, and Paul Honeine. Graph kernels based on linear patterns : theoretical and experimental comparisons. *Expert Systems With Applications*, 189 :116095, March 2022.

[KD18] Durk P Kingma and Prafulla Dhariwal. Glow : Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[KPB20] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows : An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11) :3964–3979, 2020.

[KPHL17] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv :1703.01925*, 2017.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013.

[KW19] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv :1906.02691*, 2019.

[OWFLR21] Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow : Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[PSOI19] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn : Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv :1905.13372*, 2019.

[STC+04] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[TDY+20] Thao Tran Thi Phuong, Ahlame Douzal, Saeed Varasteh Yazdi, Paul Honeine, and Patrick Gallinari. Interpretable time series kernel analytics by pre-image estimation. *Artificial Intelligence*, 286 :103342, September 2020.

[YLY+18] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018.

[ZH17] Fei Zhu and Paul Honeine. Online kernel nonnegative matrix factorization. *Signal Processing*, 131 :143 – 153, February 2017.