Graph Normalizing Flows to Pre-image Free Machine Learning for Regression

Clément Glédel(✉), Benoît Gaüzère, and Paul Honeine

Univ Rouen Normandie, INSA Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS UR 4108, 76000 Rouen, France
clement.gledel@univ-rouen.fr

Abstract. In Machine Learning, data embedding is a fundamental aspect of creating nonlinear models. However, they often lack interpretability due to the limited access to the embedding space, also called latent space. As a result, it is highly desirable to represent, in the input space, elements from the embedding space. Nevertheless, obtaining the inverse embedding is a challenging task, and it involves solving the hard pre-image problem. This task becomes even more challenging when dealing with structured data like graphs, which are complex and discrete by nature. This article presents a novel approach for graph regression using Normalizing Flows (NFs), in order to avoid the pre-image problem. By creating a latent representation space using a NF, the method overcomes the difficulty of finding an inverse transformation. The approach aims at supervising the space generation process in order to create a space suitable for the specific regression task. Furthermore, any result obtained in the generated space can be translated into the input space through the application of the inverse transformation learned by the model. The effectiveness of our approach is demonstrated by using a NF model on different regression problems. We validate the ability of the method to efficiently handle both the pre-image generation and the regression task.

Keywords: Graph Normalizing Flows · Pre-image problem · Regression · Interpretability · Nonlinear embedding

1 Introduction

Graph machine learning generally operates by embedding graph data to a meaningful space known as the latent (or feature) space. This embedding can either be implicit, as in the case of kernel machines, or explicit through the use of nonlinear operations in deep neural networks or more classic graph embedding approaches [3,11]. While providing accurate prediction models for classification or regression tasks, such methods lack interpretability, and it may be interesting to invert the embedding and map the results back to the graph space to analyze the behavior of the model. This process is referred to as the pre-image problem. Several solutions for the pre-image problem have been proposed in the general

M. Vento et al. (Eds.): GbRPR 2023, LNCS 14121, pp. 92–101, 2023.
https://doi.org/10.1007/978-3-031-42795-4_9

case [1,9]. Moreover, some solutions have been proposed to solve the pre-image problem on graph data [2,10].

In this work, we propose to design an interpretable prediction model using a graph regression method that addresses the issue of the pre-image. The key idea is to define a nonlinear-embedding function that is invertible by design. The learned embedding space is designed to linearly organize the samples, leading to good regression performances by the application of any standard linear regression method in this space. Additionally, pre-images can be conveniently generated by applying the inverse mapping on any sample of interest from the latent space.

Our approach producing a reversible nonlinear-embedding function takes inspiration from recent advances in graph generative models [5–7], including Normalizing Flows (NFs) for graphs and molecular data [20]. By defining an invertible transformation from the complex distribution of input data to a simple distribution easy to manipulate, NFs can learn a latent space while guaranteeing the invertibility of the model. Our experimental results showcase the efficacy of our proposed methodology through the use of the MoFlow architecture [20], a graph normalizing flow (GraphNF) using coupling-layers to operate on graphs represented by a combination of a feature matrix and adjacency tensor. We conducted experiments on well-known molecular datasets to demonstrate the applicability of our approach in addressing graph regression tasks while producing a high-quality representation space free of the pre-image problem.

The paper is organized as follows: Sect. 2 gives an overview of NFs and GraphNFs. Our contributions are presented in Sect. 3, which is divided into three parts. We first revisit the NF to address a regression task, and then detail the regression model. Lastly, we introduce the pre-image generation operations. The experiments and conclusion follow in Sects. 4 and 5, respectively.

2 Normalizing Flow Preliminaries

Normalizing Flows (NFs) are generative models that learn an invertible transformation function Φ between two probability distributions: a complex data distribution $P_{\mathcal{X}}$ and another distribution $P_{\mathcal{Z}}$, often chosen as a simple Gaussian distribution represented in a latent space. This allows fast and efficient data generation by sampling from the Gaussian distribution in the latent space and using the inverse function Φ^{-1} to generate data in the input space $\mathcal{X}$. The relationship between the two probability densities in NFs is defined using the change of variable formula. Therefore, considering the input samples $x_1, x_2, \ldots, x_N \in \mathcal{X}$, the training is performed by maximizing the log-likelihood function

$$\log P_{\mathcal{X}}(X) = \sum_{i=1}^{N} \log P_{\mathcal{Z}}(\Phi(\mathbf{x}_i)) + \log \left| \det \left(\frac{\partial \Phi(\mathbf{x}_i)}{\partial \mathbf{x}_i} \right) \right|, \tag{1}$$

where the determinant of the Jacobian of the function Φ, evaluated at $\mathbf{x}_i$, indicates the degree of deformation between the two distributions. This expression represents the exact relationship between the distributions, which differs from

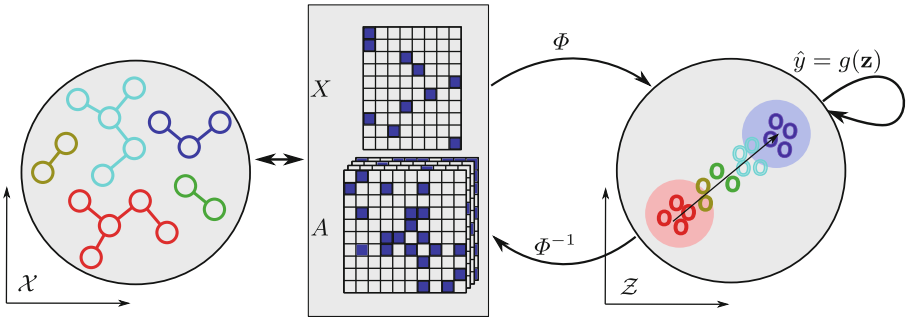

Fig. 1. NF adapted to a regression task, colors correspond to y values.

variational auto-encoders that rely on lower bounds. To define this relationship, the NF model should have an easy-to-invert structure with an easy-to-compute determinant. In order to enhance the model's expressiveness, the transformation Φ combines ℓ bijective functions, i.e., $\Phi = \Phi_\ell \circ \Phi_{\ell-1} \circ \cdots \circ \Phi_1$. This results in computing the Jacobian determinant of Φ as the product of the determinants of all Φ_i. Various NF architectures have been proposed in the literature to satisfy these constraints by defining a triangular Jacobian matrix whose determinant computation is very efficient [14].

Graph Normalizing Flows (GraphNFs) apply the concept of NFs to graph-structured data. While some GraphNFs generate graphs sequentially, such as GraphAF [19] based on a flow-based autoregressive model, a large number of GraphNFs generate graphs in a one-shot manner [8,15,16,20]. In the latter, the first attempt to design a graph neural network using NF structures was GNF [15] where the node features are updated using reversible message passing transformation based on coupling layers. GraphNVP [16] and MoFlow [20] are GraphNFs working in a one-shot manner and representing molecular graphs as a pair of node feature matrix and adjacency tensor. They are based on the use of affine-coupling layers to both the node feature matrix and the adjacency tensor. Specifically, MoFlow involves a modified version of Glow [13] – which is a convolutional NF for image data – to model the bonds and a new graph conditional flow to model the atoms given the bonds using Relational Graph Convolutional Network models. Moreover, to ensure the validity of the generated molecules, MoFlow applies a post-hoc correction step.

3 Proposed Approach

This article presents a graph regression approach based on the NF formalism, where the NF generated latent space $\mathcal{Z}$ is both relevant to the regression task at hand and, by design, does not suffer from the pre-image problem. The underlying idea, illustrated in Fig. 1, involves the three steps. First, we propose to supervise the learning of the NF function $\Phi : \mathcal{G} \to \mathcal{Z}$ to generate a distribution that

follows multiple Gaussian distributions linearly organized in $\mathcal{Z}$. Second, using the learned latent space $\mathcal{Z}$, we can perform straightforward operations (e.g., ridge regression) or more sophisticated algorithms to predict a quantitative value given a data. Finally, as our model is invertible, both the transformation Φ and its inverse Φ^{-1} are produced by our training algorithm. Therefore, it is possible to compute the pre-image of any point from the latent space $\mathcal{Z}$.

3.1 Regression-Based NF

Traditional NF models embed data in such a way that the distribution in the latent space follows a target probability density, typically a Gaussian distribution. However, this configuration does not permit data to be organized based on their quantitative values. In this section, we propose an adaptation of the NF objective function to embed data based on their quantitative values, thus, suitable for linear regression.

Consider a dataset $\mathcal{D} = \{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\}$ composed of input graph data denoted by $G_i \in \mathcal{G}$ and their corresponding quantitative labels denoted by $y_i \in Y \subset \mathbb{R}$. Specifically, we consider the case where every graph is partitioned into its corresponding feature matrix and adjacency tensor, i.e., $G = (\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e}$ where each graph is represented by n nodes of d dimensions and a set of edges characterized by e dimensions. Thus, the total dimension number is $D = n^2 \times e + n \times d$. In this paper, we consider using NF models to represent data, where each data point is represented by a two-part latent representation that corresponds to the features matrix and adjacency tensor. In particular, we concatenate the flattened representations of these two parts to obtain the representation of data in the latent space $\mathcal{Z} \subset \mathbb{R}^D$.

We constitute our latent space using Gaussian distributions, each parameterised by a mean $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$, namely

$$P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu}))}.$$

From this, we define the log-probability to belong to a Gaussian as

$$\log P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}\left(D\log(2\pi) + (\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})\right) - \log(\det(\boldsymbol{\Sigma})). \quad (2)$$

While common NFs rely on isotropic multivariate Gaussian distributions, thus using a parameterization of zero-valued $\boldsymbol{\mu}$ for all dimensions and the identity matrix as the covariance matrix $\boldsymbol{\Sigma}$, our approach aims at solving regression problems by the use of Gaussian distribution interpolations in $\mathcal{Z}$. The principle is to learn a distribution that spreads along a main axis by interpolating between two Gaussians, which are associated with the extreme quantitative values. Therefore, two Gaussian distributions are defined and parameterized by ($\boldsymbol{\mu}_1$, $\boldsymbol{\Sigma}_1$) and ($\boldsymbol{\mu}_2$, $\boldsymbol{\Sigma}_2$), which are respectively associated with the minimum and maximum values of Y. For the sake of simplicity, we use isotropic Gaussians, namely with covariance matrices $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \sigma^2 \mathbb{I}_D$, where $\mathbb{I}_D$ is the $D \times D$ identity matrix and $\sigma^2 \in \mathbb{R}$ represents the distribution variance.

To carry out the interpolation process, a belonging coefficient τ_{y_i} is assigned to each sample (G_i, y_i) based on its quantitative value computed with

$$\tau_{y_i} = \frac{y_i - \min(Y)}{\max(Y) - \min(Y)}. \tag{3}$$

The interpolated Gaussian mean $\boldsymbol{\mu}_{y_i}$ is computed using

$$\boldsymbol{\mu}_{y_i} = \tau_{y_i}\, \boldsymbol{\mu}_1 + (1 - \tau_{y_i})\, \boldsymbol{\mu}_2\,. \tag{4}$$

For the interpolation method to be useful, the 2 extreme Gaussian locations in $\mathcal{Z}$ represented by their means $(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ should be distinct and sufficiently separated. Thus, we propose to learn their means within the NF training. To achieve this, we incorporate an additional objective function into the NF objective function, aimed at maximizing the separability of the Gaussians.

Thus, the proposed NF loss function is composed of two terms. The first one applies the change of variable formula (1) to describe the change in density of a single sample. Specifically, for each sample G_i, the corresponding loss is

$$\mathcal{L}_{\text{nf}}(G_i, \boldsymbol{\mu}_{y_i}) = -\log P_{\mathcal{Z}}(\Phi(G_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) - \log \left| \det \left(\frac{\partial \Phi(G_i)}{\partial G_i} \right) \right|.$$

Here, $\log P_{\mathcal{Z}}(\Phi(G_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$ refers to (2), which uses the interpolated Gaussian parameters $(\boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$. The mean $\boldsymbol{\mu}_{y_i}$ is computed using (4), while the covariance matrix $\boldsymbol{\Sigma}_{y_i}$ is defined in the same way as the other Gaussian distributions, namely $\boldsymbol{\Sigma}_{y_i} = \sigma^2\, \mathbb{I}_D$. The second term promotes the separation between the two extreme Gaussians with $\mathcal{L}_\mu = -\log\left(1 + \| \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \|_2^2\right)$. Thus, the final loss function is

$$\mathcal{L}(G_i, y_i) = \mathcal{L}_{\text{nf}}(G_i, \boldsymbol{\mu}_{y_i}) + \beta \mathcal{L}_\mu, \tag{5}$$

with β corresponding to the tradeoff coefficient between the two terms.

Let Θ denotes the set of parameters of Φ, and let ω the set of optimizable parameters, i.e., $\omega = \{\Theta, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2\}$. To estimate the parameters in ω, we employ a stochastic gradient descent algorithm that minimizes the loss function (5) over a randomly selected batch I of the training dataset at each iteration, namely

$$\omega \leftarrow \omega - \eta \sum_{k \in I} \nabla_\omega \mathcal{L}(G_k, y_k), \tag{6}$$

where η is the learning rate.

3.2 Operating in $\mathcal{Z}$

Our approach allows a customized generation of a latent space $\mathcal{Z}$ where data are linearly organized. Therefore, a simple and efficient predictive model can be defined in $\mathcal{Z}$. We denote $g : \mathcal{Z} \to \mathbb{R}$ a linear predictive model. Since Φ is a nonlinear function, defining the linear predictive model g in $\mathcal{Z}$ is equivalent to defining a nonlinear predictive model $f : \mathcal{G} \to \mathbb{R}$, described as $f(G) = g(\Phi(G))$.

Let $g(\mathbf{z}) = \mathbf{z}^\top \boldsymbol{\varphi}^*$ be the predictive linear model $g : \mathcal{Z} \to \mathbb{R}$, where $\boldsymbol{\varphi}^* \in \mathcal{Z}$ are the optimal parameters to be estimated. Without losing generality, we consider a ridge regression in the latent space. This involves finding the best parameters by minimizing the regularized mean square error given by

$$\min_{\boldsymbol{\varphi}} \frac{1}{N} \sum_{i=1}^{N} \Big(y_i - \Phi(G_i)^\top \boldsymbol{\varphi} \Big)^2 + \lambda \, \|\boldsymbol{\varphi}\|_2^2, \tag{7}$$

where the importance of the regularization term is weighted by λ. The optimal solution vector $\boldsymbol{\varphi}^*$ for this regularized optimization problem is obtained by nullifying its gradient, resulting in

$$\boldsymbol{\varphi}^* = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbb{I}_D)^{-1} \, \mathbf{Z}^\top \, \mathbf{y} \, . \tag{8}$$

where

$$\mathbf{Z} = \begin{pmatrix} \Phi(G_1) & \cdots & \Phi(G_N) \end{pmatrix}^\top$$
$$\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^\top .$$

Then using (8) leads to the optimal predictive parameters in $\mathcal{Z}$ and we can therefore specify the nonlinear predictive model $f : \mathcal{G} \to \mathbb{R}$ by combining the transformation function Φ and the linear regression model g. Using this method, the quantitative value prediction for any graph $G \in \mathcal{G}$ is achieved by

$$f(G) = \Phi(G)^\top (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbb{I}_D)^{-1} \, \mathbf{Z}^\top \, \mathbf{y} \, .$$

3.3 Pre-imaging

The availability of Φ^{-1} allows the pre-image of any point of interest from the latent space to be computed, thus eliminating the pre-image problem. We propose a pre-image generation method to obtain new data given a quantitative label y. Indeed, our regression approach creates a linear relation between quantitative labels and positions in $\mathcal{Z}$. As the Gaussian distribution characterized by $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ is intentionally associated with the minimum quantitative label in Y, and the one defined by $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is associated to the maximum quantitative label in Y, we can determine the position of the mean $\boldsymbol{\mu}_y$ that corresponds to a quantitative label y by employing (4). Then, from the Gaussian distribution parameterized by $(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, it is possible to sample a point $\mathbf{z} \in \mathcal{Z}$ with $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \, \mathbb{I}_D)$ where σ^2 represents the variance of the Gaussian distributions chosen during the training of the model and obtain its pre-image in $\mathcal{G}$, namely

$$\hat{G} = \Phi^{-1}(\mathbf{z}).$$

4 Experiments

We evaluated our approach[1] in order to answer two separate questions:

Q1 Can the latent space produced by our regression method for graph data be considered effective, and are the representations generated suitable for regression-based objectives?
Q2 Does our model preserve its ability to efficiently generate pre-images ?

The analysis was conducted on three molecular regression datasets. In these datasets, the nodes encode the atoms of the molecule and are labeled by the chemical element of each atom. The edges encode the chemical bonds between atoms. The **QM7 dataset** is a quantum chemistry dataset composed of $7,165$ small organic molecules with up to 7 significant atoms. Its regression task consist in predicting the atomization energy of each molecule. The **ESOL** dataset is composed of $1,128$ molecular compounds, with a maximum of 55 nodes. As the used graph representation is sensitive to the size of the graphs, molecules with more than 22 atoms were filter out, thus reducing the dataset to $1,015$ different graphs. The prediction task consists of predicting the solubility measurement associated with each molecule. Finally, the **FREESOLV** dataset provides experimental and calculated information on the hydration free energies of 643 small molecules in water, with a maximum of 24 nodes. Similarly to the ESOL dataset, the dataset was reduced to 632 distinct graphs with a maximum size of 22 nodes.

We implemented our approach using **MoFlow** [20] and compared it to standard graph kernels, such as **Weisfeiler-Lehman** (WL) [18], **Shortest-Path** (SP) [4] and **Hadamard Code** (Hadcode) [12]. In addition, as our approach consists in applying ridge regression on the concatenation of the flattened representations of the graph in $\mathcal{Z}$, we also compared to simpler kernels working on the concatenation of the flattened representations of the graph in $\mathcal{G}$. We considered standard vector-based kernels: linear, RBF, Polynomial and a sigmoid [17]. Each predictive model was trained by minimizing the cost function described in (7). Finally, the capability of generating pre-images was compared with the approach outlined in [1], employed on previously mentioned kernel methods.

Each dataset was split to 90% for training and the remaining 10% for evaluation. To ensure a fair comparison, the kernels were fine-tuned and their parameters determined by cross-validation with a grid search where 10 values of λ ranging from 10^{-5} to 10^{2} were selected for sampling. In addition, for the simpler kernels, a logarithmic scale was used to sample 5 weighting values applied to the similarity measure used in the kernel ranging from 10^{-5} to 10^{3}. The power of the Polynomial kernel was varied over a set of candidate values: $1, 2, 3, 4$. For regression evaluations, the performance is measured by the R^2 score.

As described in previous sections, we converted the graph data into a combination of a node feature matrix and an adjacency tensor, namely $G = (\mathbf{X}, \mathbf{A}) \in \mathcal{G}$ with $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{A} \in \mathbb{R}^{n \times n \times e}$. The conducted experiments used labeled edges

[1] For sake of reproducibility, all experiments can be reproduced from the available GitHub repository https://github.com/clement-g28/nf-kernel.

Table 1. R^2 score (± std) on graph regression datasets

Method		Datasets		
		QM7	ESOL	FREESOLV
Standard Kernels	Linear	0.681 ± 0.001	0.555 ± 0.032	0.254 ± 0.114
	RBF	0.680 ± 0.002	0.558 ± 0.032	0.262 ± 0.113
	Polynomial	0.681 ± 0.001	0.566 ± 0.034	0.264 ± 0.102
	Sigmoid	0.673 ± 0.002	0.563 ± 0.037	0.255 ± 0.074
Graph Kernels	WL	0.490 ± 0.0	0.602 ± 0.0	0.895 ± 0.0
	SP	0.721 ± 0.0	0.531 ± 0.0	0.543 ± 0.0
	Hadcode	0.491 ± 0.0	0.573 ± 0.0	**0.901** ± 0.0
(Ours)	MoFlow	**0.730** ± **0.008**	**0.685** ± 0.039	0.754 ± 0.042

in each dataset, leading to the definition of e as the number of edge labels plus one label for the non-existent edge. The value of d was determined by the number of node labels along with one label for the non-existent node. In addition, since a graph composed of n nodes can be represented in $n!$ distinct ways using such a representation method, we trained our models by implementing a random permutation transformation of the input graph data. Additionally, to assess the permutation variability of our technique, all experiments were evaluated 10 times by employing random permutations. Consequently, the performance are reported as the mean performance and its standard deviation.

To answer **Q1**, Table 1 displays the average performance and standard deviation (std) on the regression datasets. These results show the good predictive performance of our method when employing linear ridge regression within the latent space $\mathcal{Z}$ for most datasets. However, best results on the FREESOLV dataset are achieved by the Weisfeiler-Lehman and Hadamard Code kernels. To understand this point, it is noteworthy that this dataset comprises a smaller number of distinct graphs, which are relatively larger in size when compared to other datasets like QM7. Moreover, the quantitative values in the FREESOLV dataset are non-uniformly distributed, ranging between -5.48 and 1.79, with a significant proportion (more than 97%) falling within the range of -2.57 to 1.79. As a result, the linear interpolation-based approach used in this study may not be the most suitable method for such datasets. However, our approach has the advantage over graph kernels that once the model has been learned, predictions can be made directly and simply while graph kernel methods can be computationally more expensive. Moreover, we can observe in the results non-zero standard deviation in our performances due to the nature of the used graph representations, as opposed to the graph kernels that are designed to be permutation invariant. Therefore, the question **Q1** can be answered positively in most cases, indicating that the nonlinear transformation Φ learned by our method is able of producing a good representation space for a regression task.

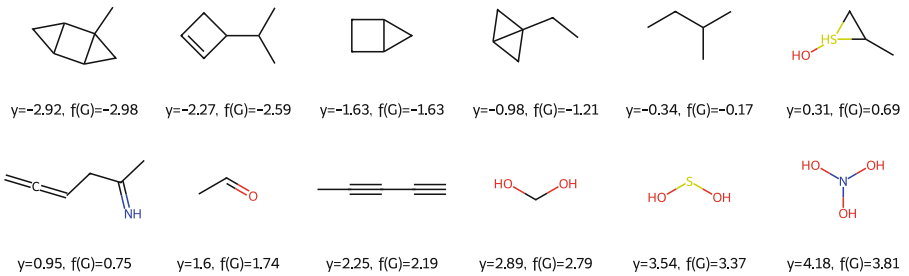

Fig. 2. The pre-images generated from 12 points in $\mathcal{Z}$ sampled uniformly between the $\min(Y)$ and $\max(Y)$.

To answer **Q2** and test the ability of our model to generate pre-images, we generated molecules from sampling points in $\mathcal{Z}$ using our model trained on the QM7 dataset. These points were sampled by interpolating 12 values of y between $\min(Y)$ and $\max(Y)$ as follows: For each value, we sampled a point in $\mathcal{Z}$ with $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \mathbb{I}_D)$. Corresponding pre-images were generated in $\mathcal{X}$ by applying the inverse transformation Φ^{-1} to the generated z. Figure 2 shows the obtained pre-images, as well as the sampled quantitative values y and the predicted value $f(G)$ using our learned prediction model. This experiment demonstrates that our model can generate meaningful pre-images of points in $\mathcal{Z}$ that are not a part of the dataset, hence answering positively to **Q2**.

5 Conclusion

Our paper presented a novel approach that overcomes the curse of the pre-image using NFs for a graph regression task. Our method generates a supervised space where linear regression can be efficiently operated, as demonstrated by the conducted experiments. The results indicated that the obtained latent space is efficient in solving graph regression problems using straightforward linear operations. Moreover, the method enabled interpretability by facilitating the transformation from the latent space to the input space and generating pre-images of specific points of interest.

Our approach contributes to the application of NFs in a specific task and has the potential to be adapted for other tasks such as classification. Although our method is sensitive to the permutation of the graph due to the used representation, it may be interesting to extend it to other types of graph representations, making it possible to achieve permutation invariance.

Acknowledgements. The authors acknowledge the support of the French *Agence Nationale de la Recherche* (ANR), under grant ANR-18-CE23-0014.

References

1. Bakır, G.H., Weston, J., Schölkopf, B.: Learning to find pre-images. Adv. Neural Inf. Process. Syst. **16**, 449–456 (2004)
2. Bakır, G.H., Zien, A., Tsuda, K.: Learning to find graph pre-images. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 253–261. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28649-3_31
3. Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., Honeine, P.: Analyzing the expressive power of graph neural networks in a spectral perspective. In: International Conference on Learning Representations, Vienna, Austria (2021)
4. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM 2005) (2005)
5. Bresson, X., Laurent, T.: A two-step graph convolutional decoder for molecule generation (2019). arXiv preprint arXiv:1906.03412
6. De Cao, N., Kipf, T.: Molgan: an implicit generative model for small molecular graphs (2018). arXiv preprint arXiv:1805.11973
7. Guo, X., Zhao, L.: A systematic survey on deep generative models for graph generation. IEEE Trans. Pattern Anal. Mach. Intell. **45**, 5370–5390 (2022)
8. Honda, S., Akita, H., Ishiguro, K., Nakanishi, T., Oono, K.: Graph residual flow for molecular graph generation (2019). arXiv preprint arXiv:1909.13521
9. Honeine, P., Richard, C.: Solving the pre-image problem in kernel machines: a direct method. In: 2009 IEEE International Workshop on Machine Learning for Signal Processing, pp. 1–6. IEEE (2009)
10. Jia, L., Gaüzère, B., Honeine, P.: A graph pre-image method based on graph edit distances. In: Proceedings of S+SSPR 2020 (2021)
11. Jia, L., Gaüzère, B., Honeine, P.: Graph kernels based on linear patterns: theoretical and experimental comparisons. Expert Syst. Appl. **189**, 116095 (2022)
12. Kataoka, T., Inokuchi, A.: Hadamard code graph kernels for classifying graphs. In: ICPRAM, pp. 24–32 (2016)
13. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1×1 convolutions. In: Advances in Neural Information Processing Systems, pp. 10215–10224 (2018)
14. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: an introduction and review of current methods. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 3964–3979 (2020)
15. Liu, J., Kumar, A., Ba, J., Kiros, J., Swersky, K.: Graph normalizing flows. Adv. Neural Inf. Process. Syst. **32** (2019)
16. Madhawa, K., Ishiguro, K., Nakago, K., Abe, M.: Graphnvp: an invertible flow model for generating molecular graphs (2019). arXiv preprint arXiv:1905.11600
17. Schölkopf, B., Smola, A.J., Bach, F., et al.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press, Cambridge (2002)
18. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. J. Mach. Learn. Res. **12**(9) (2011)
19. Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., Tang, J.: Graphaf: a flow-based autoregressive model for molecular graph generation (2020). arXiv preprint arXiv:2001.09382
20. Zang, C., Wang, F.: Moflow: an invertible flow model for generating molecular graphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 617–626 (2020)