GraphCG: Unsupervised Discovery of Steerable Factors in Graphs

Shengchao Liu[1,2] **Chengpeng Wang**[3] **Weili Nie**[4] **Hanchen Wang**[5] **Jiarui Lu**[1,2]
Bolei Zhou[6] **Jian Tang**[1,7,8]
[1]Mila [2]Université de Montréal [3]University of Illinois Urbana-Champaign
[4]Nvidia Research [5]University of Cambridge [6]University of California, Los Angeles
[7]HEC Montréal [8]CIFAR AI Chair
liusheng@mila.quebec

Abstract

Deep generative models have been extensively explored recently, especially for the graph data such as molecular graphs and point clouds. Yet, much less investigation has been carried out on understanding the learned latent space of deep graph generative models. Such understandings can open up a unified perspective and provide guidelines for essential tasks like controllable generation. In this paper, we first examine the representation space of the recent deep generative model trained for graph data, observing that the learned representation space is not perfectly disentangled. Based on this observation, we then propose an unsupervised method called GraphCG, which is model-agnostic and task-agnostic for discovering steerable factors in graph data. Specifically, GraphCG learns the semantic-rich directions via maximizing the corresponding mutual information, where the edited graph along the same direction will possess certain steerable factors. We conduct experiments on two types of graph data, molecular graphs and point clouds. Both the quantitative and qualitative results show the effectiveness of GraphCG for discovering steerable factors. The code will be public in the near future.

1 Introduction

Graph is a general format for many real-world data. For instance, molecules can be modeled as a graph [8, 12] where the atoms and bonds are nodes and edges respectively. Processing point clouds as graphs is also a popular strategy [49, 54], where points are viewed as nodes and edges based on nearest neighbors. Many existing works on deep generative models (DGMs) focus on modeling the graph data and improving the synthesis quality. However, understanding DGMs on graph and their learned representations has been much less explored, which may hinder the development of important applications like the controllable generation and the discovery of interpretable data structure.

The graph controllable generation task refers to modifying the steerable factors of graph so as to obtain graphs with desired properties [7, 39]. This is an important task in many applications, but traditional methods (*e.g.*, manual editing) possess certain inherent limitations under certain circumstances. A classic example is molecule editing, which aims at modifying the substructures of molecules [35] that may relate to some key tactics in drug discovery like functional group change [11] and scaffold hopping [1, 20]. This is a routine task in pharmaceutical company, yet, relying on domain experts for manual editing can be subjective or biased [7, 13]. Different to previous works, this paper aims to explore the unsupervised graph editing with DGMs. It can act as a good complementary module to conventional methods and bring many crucial benefits: (1) It enables the efficient graph editing in the large-scale setting. (2) It alleviates the requirements for extensive domain knowledge for factor change labeling. (3) It provides another perspective for editing preference, which reduces biases from the domain experts.

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

One core property relevant to unsupervised graph editing using DGMs is the disentanglement. While there does not exist a widely-accepted definition of disentanglement, the key intuition [33] is that a disentangled representation should separate the distinct, informative, and steerable factors of variations in the data. Thus, the controllable generation task would become trivial with the disentangled DGMs as the backbone. Such a disentanglement assumption has been widely used in generative modeling on the image data, *e.g.*, β-VAE [17] learns disentangled representation by forcing the representation to be close to an isotropic unit Gaussian. However, it may introduce extra constraints on the formulations and expressiveness of DGMs [17, 43, 9, 56].

For graph data, one crucial question remains: *is the latent representation space learned from DGMs on graph data disentangled?* For image generation, one finding [33] shows that without inductive bias, the latent space learned by VAEs is not guaranteed to be disentangled. However, the disentanglement property of graph DGMs is much less explored. In Sec. 3, we first study the latent space of DGMs on two typical graph data (molecular graphs and point clouds), and empirically illustrate that the learned space is not perfectly disentangled. This observation then drives us to the second question: *Given a not perfectly disentangled latent space, is there a flexible framework enabling the graph controllable generation in an unsupervised manner?* To tackle this, we propose a model-agnostic and task-agnostic framework called GraphCG for unsupervised controllable generation. GraphCG has two main phases, as illustrated in Fig. 1. During the training phase (Fig. 1(a)), GraphCG starts with the assumption that the steerable directions can be learned by maximizing the mutual information (MI) among the semantic directions. We formulate GraphCG with an energy-based model (EBM), which provides a large family of solutions. Then for the test phase, with the learned semantic directions, we can carry out the editing task by moving along the direction with certain step sizes. As the example illustrated in Fig. 1(b), the molecular structure (hydroxyl group) changes consistently along the editing sequence. For evaluation, we visually verify the learned semantic directions on both types of data. Further for the molecular graph, we propose a novel evaluation metric called sequence monotonic ratio (SMR) to measure the output sequences.

We summarize our contributions as follows: (1) We conduct an empirical study on the disentanglement property of three pretrained deep generative models (DGMs) on two types of graph data, molecular graphs and point clouds. We find that the latent space of these pretrained graph DGMs is not perfectly disentangled. (2) We propose a **model-agnostic and task-agnostic** method called GraphCG for the unsupervised graph controllable generation. GraphCG aims at learning the semantic directions by maximizing their corresponding mutual information, and its outputs are sequences of graphs. (3) We evaluate the proposed methods on two types of graph data, molecular graphs and point clouds. The experimental results show the clear improvement (up to 3 times better in SMC) over the baselines.

2 Background and Problem Formulation

Graph and deep generative models (DGMs). Each graph data (including nodes and edges) is denoted as $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is the data space, and the DGMs are modeling the data distribution, *i.e.*, $p(\boldsymbol{x})$. Our proposed graph editing method (GraphCG) is model-agnostic, so we also briefly introduce the mainstream DGMs as below. Variational auto-encoder (VAE) [27, 17] measures a variational lower bound of $p(\boldsymbol{x})$ by introducing a proposal distribution; flow-based model [6, 42] constructs revertible encoding functions such that the data distribution can be deterministically mapped to a prior distribution. Notice that these mainstream DGMs, either explicitly or implicitly, contain an encoder ($f(\cdot)$) and a decoder ($g(\cdot)$) function as:

$$\boldsymbol{z} = f(\boldsymbol{x}), \qquad \boldsymbol{x}' = g(\boldsymbol{z}), \tag{1}$$

where $\boldsymbol{z} \in \mathcal{Z}$ is the latent representation, $\mathcal{Z}$ is the latent space, and $\boldsymbol{x}'$ is the reconstructed output graph. Notice that in the literature [47, 48], people also call latent representation as latent codes, and in what follows, we will be using these two terms interchangeably.

Semantic direction and step size. In the latent space $\mathcal{Z}$, we assume there exist D semantically meaningful direction vectors, *i.e.*, $\boldsymbol{d}_i$ with $i \in \{0, 1, \ldots, D-1\}$. There is also a scalar variable, step size α, which controls the degree to edit the sampled data with desired steerable factors (as will be introduced below), and we follow the literature [47] on taking $\alpha \in [-3, 3]$. Each direction corresponds to one or multiple factors, such that by editing the latent vector $\boldsymbol{z}$ with direction $\boldsymbol{d}_i$ and step size α, the reconstructed graph will be augmented with the desired factors, leading to certain structural changes.

Steerable factors. The steerable factors are attributes to the DGMs and they refer to the semantic information that we can explicitly learn from the pretrained DGMs. In this work, we will be studying

the steerable factors on the graph data, which is data- and task-specific. Yet, there is one category of factors that is commonly shared among all the graph data: the **structure** information. Concretely, the steerable factors can be the functional groups or fragments in molecular graphs and shapes or sizes in point clouds. In Appendix C, we provide a comprehensive description of these steerable factors.

Problem formulation: graph controllable generation. The goal here is: given a pretrained DGM (*i.e.*, the encoder and decoder are fixed), we want to learn the most semantically rich directions ($\boldsymbol{d}_i$) in the latent space $\mathcal{Z}$. Then for each latent code $\boldsymbol{z}$, with the i-th semantic direction and a step size α, we can get an edited latent vector $\bar{\boldsymbol{z}}_{i,\alpha}$ and edited data $\bar{\boldsymbol{x}}'$, as:

$$\boldsymbol{z} = f(\boldsymbol{x}), \qquad \bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}_i, \alpha), \qquad \bar{\boldsymbol{x}}' = g(\bar{\boldsymbol{z}}_{i,\alpha}), \tag{2}$$

where $\boldsymbol{d}_i$ and $h(\cdot)$ are the edit direction and edit function that we want to learn. We expect that $\bar{\boldsymbol{z}}_{i,\alpha}$ can inherently possess certain steerable factors, which can then be reflected in the graph structure of $\bar{\boldsymbol{x}}'$.

Energy-based model (EBM). EBM is a flexible framework for distribution modeling:

$$p(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{A} = \frac{\exp(-E(\boldsymbol{x}))}{\int_{\boldsymbol{x}} \exp(-E(\boldsymbol{x}))d\boldsymbol{x}}, \tag{3}$$

where $E(\cdot)$ is the energy function and A is the partition function. In EBM, the bottleneck is the estimation of partition function A: it is commonly intractable due to the high cardinality of $\mathcal{X}$. Various methods have been proposed to handle this issue, including but not limited to contrastive divergence [18], noise-contrastive estimation [14, 3], and score matching [21, 50, 52].

3 Disentanglement of Latent Representation

In this section, we quantify the degree of disentanglement of the deep generative models (DGMs) for graph data. In specific, we adopt six disentanglement scores, and the observed low disentanglement scores show that the attributes in latent space are not well disentangled.

The key intuition [33] behind disentanglement is that a disentangled representation space should separate the distinct, informative, and steerable factors of variations in the data. In other words, each latent dimension of the disentangled representation correspond to one or multiple factors, and the change of the disentangled dimension can lead to the change in the factors accordingly. This is an appealing attribute for the general controllable generation task, since the straightforward changes on the latent dimensions can result in the change of corresponding properties of the data.

Is the latent space learned in the graph deep generative models disentangled? For image generation, one previous finding [33] shows that without inductive bias, the representation learned by VAEs is not perfectly disentangled. We want to verify if this claim is also valid for the mainstream DGMs on graphs. We conduct the following experiment for verification.

Steerable factors and experiments on disentanglement measure. There has been a series of works [17, 26, 9, 43, 33] measuring the disentanglement of the latent space in DGMs. The high-level idea is that, given the latent representation from a pretrained DGM, we want to explore how predictive it is to certain steerable factors.

First we need to consider the steerable factors in graph. The extraction of steerable factors requires the domain knowledge. For instance, in molecular graphs, we can extract some special substructures named fragments or functional groups. These substructures can be treated as steerable factors since they are the key components of the molecules and are closely related to certain molecular properties [46]. We use RDKit [29] to extract 9 most distinguishable fragments as steerable factors for disentanglement measurement. For point clouds, we use PCL tool [44] to extract 75 VFH descriptors [45] as steerable factors, which depicts the geometries and viewpoints accordingly.

Then for the disentanglement measure, we consider six metrics on two data types with three backbone models, and results are are shown in Table 1. All the metric values range from 0 to 1, and the higher the value, the more disentangled the DGM is. So we can observe that generally, these graph DGMs are not perfectly disentangled.

Table 1: The six disentanglement metrics on three pretrained DGMs and two graph types. All measures range from 0 to 1, and higher scores mean more disentangled representation.

Graph Type	Model	Dataset	BetaVAE [17] ↑	FactorVAE [26] ↑	MIG [4] ↑	DCI [9] ↑	Modularity [43] ↑	SAP [28] ↑
Molecular Graph	MoFlow	ZINC250K	0.260	0.175	0.031	0.953	0.620	0.009
	HierVAE	ChEMBL	0.178	0.165	0.022	0.114	0.606	0.026
Point Cloud	PointFlow	Airplane	0.022	0.025	0.029	0.160	0.745	0.022

4 GraphCG Method

Following the formulation in Sec. 2, given a fixed pretrained graph generative model, the goal is to discover semantically meaningful directions where we can do interpolation in the latent space. More concretely, we assume there are D semantic directions in the latent space $\mathcal{Z}$, and each direction corresponds to multiple steerable factors. Moving in the latent space along a single semantic direction with a step size α can result in the change of the certain steerable factors in the graph data accordingly, *i.e.*, the change of graph structures. In Sec. 3, the experiment shows that the learned representation space in graph DGMs is not perfectly disentangled. Without the disentanglement assumption, the graph controllable generation task becomes more challenging.

This naturally raises the next research question: *given a not well-disentangled representation space, is there a flexible way to do the graph data editing?* The answer is positive. We propose GraphCG, a flexible model-agnostic and task-agnostic framework to learn the semantic directions in an unsupervised manner. It starts with the assumption that the latent representations edited with the same semantic direction and step size should possess similar information (corresponding to the factors) to certain degree, thus by maximizing the mutual information them, we can learn the most semantic-rich directions. Then we formulate this editing task as a density estimation problem with the energy-based model (EBM). As introduced in Sec. 2, EBM covers a broad range of solutions, and we further propose GraphCG-NCE by adopting the noise-contrastive estimation (NCE).

4.1 GraphCG with Mutual Information

The mutual information (MI) measures the non-linear dependency between variables. To adapt it here, we assume that maximizing the MI between edited data points with different editing conditions (directions and step sizes) can maximize the shared information within each semantic direction and step size, while diversifying the semantic information among different conditions. The pipeline is as follows.

We first sample two codes in the latent space, $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$. Then we pick up the i-th semantic direction and one step size α to obtain the edited latent points in $\mathcal{Z}$. The corresponding points are as

$$\bar{\boldsymbol{z}}^u_{i,\alpha} = h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha), \qquad \bar{\boldsymbol{z}}^v_{i,\alpha} = h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha). \tag{4}$$

Under our assumption, we expect that these two edited points can share certain information with respect to the steerable factors. Thus, we want to maximize the MI between $\bar{\boldsymbol{z}}^u_{i,\alpha}$ and $\bar{\boldsymbol{z}}^v_{i,\alpha}$, which is equivalent to estimating the following objective [32, 31]:

$$\mathcal{L}_{\text{MI}}(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}) = \frac{1}{2}\mathbb{E}_{p(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha})}\big[\log p(\bar{\boldsymbol{z}}^u_{i,\alpha}|\bar{\boldsymbol{z}}^v_{i,\alpha}) + \log p(\bar{\boldsymbol{z}}^v_{i,\alpha}|\bar{\boldsymbol{z}}^u_{i,\alpha})\big]. \tag{5}$$

Till this step, we have transformed the graph data editing task into the summation of two conditional log-likelihoods estimation problem.

4.2 GraphCG with Energy-Based Model

Following Eq. (5), maximizing the MI between $I\big(\bar{\boldsymbol{z}}^u_{i,\alpha}; \bar{\boldsymbol{z}}^v_{i,\alpha}\big)$ is equivalent to estimating the summation of two conditional log-likelihoods. We then model them using the energy-based model (EBM). Because these two views are in the mirroring direction, we may as well take one for illustration. For example, for the first conditional log-likelihood, we can model it with EBM as:

$$p(\bar{\boldsymbol{z}}^u_{i,\alpha}|\bar{\boldsymbol{z}}^v_{i,\alpha}) = \frac{\exp(-E(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}))}{\int \exp(-E(\bar{\boldsymbol{z}}^{u'}_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}))d\bar{\boldsymbol{z}}^{u'}_{i,\alpha}} = \frac{\exp(f(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}))}{A_{ij}}, \tag{6}$$

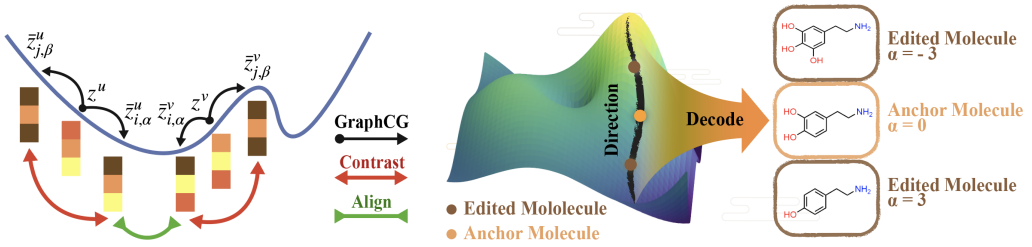
where $E(\cdot)$ is the energy function, A_{ij} is the intractable partition function, and $f(\cdot)$ is the negative energy. The (negative) energy function can be quite flexible, and here we use the dot-product:

$$f(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}) = \langle h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha), h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha)\rangle, \tag{7}$$

where $h(\cdot)$ is the editing function introduced in Eq. (2). Similarly for the other conditional log-likelihood term, and the objective becomes:

$$\mathcal{L}_{\text{GraphCG}} = \mathbb{E}\big[\log \frac{\exp(f(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha}))}{A_{ij}} + \log \frac{\exp(f(\bar{\boldsymbol{z}}^v_{i,\alpha}, \bar{\boldsymbol{z}}^u_{i,\alpha}))}{A_{ji}}\big]. \tag{8}$$

With Eq. (8), we are able to learn the semantically meaningful direction vectors. We name this unsupervised graph controllable generation framework as GraphCG. In specific, GraphCG utilizes EBM for estimation, which yields a wide family of solutions. Next we will introduce an intuitive solution.



(a) Training phase of GraphCG. (b) Test phase of GraphCG.

Figure 1: (a) the training phase. Given two latent codes $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$, we edit the four latent representations along i-th and j-th direction with step size α and β respectively. The goal of GraphCG is to align the positive pair ($\bar{\boldsymbol{z}}^u_{i,\alpha}$ and $\bar{\boldsymbol{z}}^v_{i,\alpha}$), and contrast them with $\bar{\boldsymbol{z}}^u_{j,\beta}$ and $\bar{\boldsymbol{z}}^v_{j,\beta}$ respectively. (b) the test phase. We will first sample an anchor molecule, and adopt the learned directions in the training phase for editing. With step size $\alpha \in [-3, 3]$, we can generate a sequence of molecules. Specifically, after decoding, there is a functional group change shown up: the number of hydroxyl groups decreases along the sequence in the decoded molecules.

4.3 GraphCG with Noise Contrastive Estimation

We solve Eq. (8) using the noise contrastive estimation (NCE) [14]. The high-level idea of NCE is to introduce a noise distribution and transforms the density estimation into a binary classification problem: it aims at distinguishing if the data comes from the noise distribution or from the true distribution. NCE for solving EBM has been widely discussed [51], and we adopt it as GraphCG-NCE:

$$\begin{aligned}\mathcal{L}_{\text{GraphCG-NCE}} = -\Big(&\mathbb{E}_{p_n(\bar{\boldsymbol{z}}^u_{j,\beta}|\bar{\boldsymbol{z}}^v_{i,\alpha})}\big[\log\big(1-\sigma(f(\bar{\boldsymbol{z}}^u_{j,\beta},\bar{\boldsymbol{z}}^v_{i,\alpha}))\big)\big] + \mathbb{E}_{p_{\text{data}}(\bar{\boldsymbol{z}}^u_{i,\alpha}|\bar{\boldsymbol{z}}^v_{i,\alpha}))}[\log\sigma(f(\bar{\boldsymbol{z}}^u_{i,\alpha},\bar{\boldsymbol{z}}^v_{i,\alpha}))] \\ &+ \mathbb{E}_{p_n(\bar{\boldsymbol{z}}^v_{j,\beta}|\bar{\boldsymbol{z}}^u_{i,\alpha})}\big[\log\big(1-\sigma(f(\bar{\boldsymbol{z}}^v_{j,\beta},\bar{\boldsymbol{z}}^u_{i,\alpha}))\big)\big] + \mathbb{E}_{p_{\text{data}}(\bar{\boldsymbol{z}}^v_{i,\alpha}|\bar{\boldsymbol{z}}^u_{i,\alpha}))}[\log\sigma(f(\bar{\boldsymbol{z}}^v_{i,\alpha},\bar{\boldsymbol{z}}^u_{i,\alpha}))]\Big),\end{aligned} \tag{9}$$

where p_{data} is the data distribution and p_n is the noise distribution. The latent vectors, $\boldsymbol{z}^u, \boldsymbol{z}^v$, are given, and the noise distribution is based on the semantic directions and step sizes. Notice that the noise distribution can be very flexible, and we use a uniform random sampling in this paper. The objective Eq. (9) is for one latent code pair, and we will take the expectation of it over the dataset. Besides, we would like to consider extra similarity and sparsity constraints as:

$$\mathcal{L}_{\text{sim}} = \mathbb{E}_{i,j}[\text{sim}(d_i, d_j)], \qquad \mathcal{L}_{\text{sparsity}} = \frac{1}{D}\sum_{i=1}^{D}|d_i|, \tag{10}$$

where $\text{sim}(\cdot)$ is the similarity function between two latent representations, and we take the dot product. The similarity term is to minimize the similarity between semantic directions, and the sparsity term is to sparsify each semantic direction. Both regularization terms target at making the learned semantic directions more sparse and diverse. Thus, the final objective function is:

$$\mathcal{L} = c_1 \cdot \mathbb{E}_{u,v}[\mathcal{L}_{\text{GraphCG-NCE}}] + c_2 \cdot \mathcal{L}_{\text{sim}} + c_3 \cdot \mathcal{L}_{\text{sparsity}}, \tag{11}$$

and c_1, c_2, c_3 are coefficients. This pipeline is illustrated in Fig. 1.

Positive and Negative Views. GraphCG-NCE is essentially a contrastive learning method, *i.e.*, we define the positive and negative pairs, and the goal is to align the positives and contrast the negatives. As described in Eq. (9), the positive pairs $(\boldsymbol{z}^u, \boldsymbol{z}^v)$ are latent codes moving with the same semantic direction and step size, while the negative pairs are the edited latent codes with different semantic directions and/or step sizes. We consider two options in designing the positive views. (1) **Perturbation (GraphCG-P)** is that for each latent variable $\boldsymbol{z} \in \mathcal{Z}$, we apply 2 perturbations (*e.g.*, adding Gaussian noise) on $\boldsymbol{z}$ to get 2 perturbed latent codes as $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$ respectively. (2) **Random sampling (GraphCG-R)** is that we randomly encode two data points from the empirical data distribution as $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$ respectively. For negative pairs, we simply sample latent codes with different directions and different step sizes.

Semantic Direction Modeling. To model the semantic direction, we first randomly draw one *direction basis vector* for each direction, denoted as $\boldsymbol{e}_i$. Then the corresponding direction vector is modeled using $\boldsymbol{d}_i = \text{MLP}(\boldsymbol{e}_i)$, where $\text{MLP}(\cdot)$ is the multi-layer perceptron network.

Design of Editing Function. Given the semantic direction and two views, the next task is to design the editing function in Eq. (2). Our proposed GraphCG is flexible, and the editing function is the key design in the energy function Eq. (7). Thus, we consider both the linear and non-linear cases as:

$$\bar{\boldsymbol{z}}_i = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \qquad \bar{\boldsymbol{z}}_i = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i + \text{MLP}(\boldsymbol{z} \oplus \boldsymbol{d}_i \oplus [\alpha]). \tag{12}$$

4.4 Implementations

The training algorithm of GraphCGis in Algorithm 1, where the goal is to learn semantically meaningful direction vectors and an editing function in the latent space. Then we will need to manually pick up the direction vectors corresponding to certain factors for editing with some post-training evaluation metrics. Finally for the test phase, we can use the pretrained DGM and selected direction for editing, as described in Eq. (2). The detailed algorithm is illustrated in Algorithm 2. Next, we briefly discuss the differences to some other related methods.

Algorithm 1 Training Phase of GraphCG

Input: Given a pretrained generative model, $f(\cdot)$ and $g(\cdot)$.
Output: Learned direction vector $\boldsymbol{d}_i$ and function $h(\cdot)$.
Select two data points, $\boldsymbol{z}^u, \boldsymbol{z}^v \in \mathcal{Z}$.
for each step size α and each direction i **do**
  Set $\bar{\boldsymbol{z}}^u_{i,\alpha} = h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha)$.
  Set $\bar{\boldsymbol{z}}^v_{i,\alpha} = h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha)$.
  Assign positive to pair $(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha})$.
  for step size $\beta \neq \alpha$ and direction $j \neq i$ **do**
    Set $\bar{\boldsymbol{z}}^u_{j,\beta} = h(\boldsymbol{z}^u, \boldsymbol{d}_j, \beta)$.
    Set $\bar{\boldsymbol{z}}^v_{j,\beta} = h(\boldsymbol{z}^v, \boldsymbol{d}_j, \beta)$.
    Assign negative to pair $(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{j,\beta})$.
    Assign negative to pair $(\bar{\boldsymbol{z}}^u_{j,\beta}, \bar{\boldsymbol{z}}^v_{i,\alpha})$.
  end for
  Do SGD w.r.t. GraphCG in Eq. (11).
end for

Algorithm 2 Test Phase of GraphCG

Input: Given a pre-trained generative model ($f(\cdot)$ and $g(\cdot)$), a learned direction vector $\boldsymbol{d}$, (optional) input molecule $\boldsymbol{x}$.
Output: A sequence of edited graphs.
Do sampling (or encoding if $\boldsymbol{x}$ is given) to get a latent code $\boldsymbol{z}$.
for step size $\alpha \in [-3, 3]$ **do**
  Do graph edit in the latent space to get $\bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}, \alpha)$.
  Decode to the graph space with $\bar{\boldsymbol{x}}' = g(\bar{\boldsymbol{z}}_{i,\alpha})$.
end for
Output is thus a sequence of edited graphs, $\{\bar{\boldsymbol{x}}'\}$.

NCE and Contrastive Representation Learning. GraphCG-NCE is using EBM-NCE, which is essentially a contrastive learning method, and another dominant contrastive loss is the InfoNCE [38]. We list the main differences below. (1) EBM-NCE is equivalent to InfoNCE, and both are essentially doing the same thing: align the positive pairs and contrast the negative pairs. (2) EBM-NCE [16, 32] has been found to outperform InfoNCE on the general graph data. (3) What we want to build up here is a flexible framework, as in EBM or GraphCG. Specifically, EBM provides a more general framework by designing the energy functions.

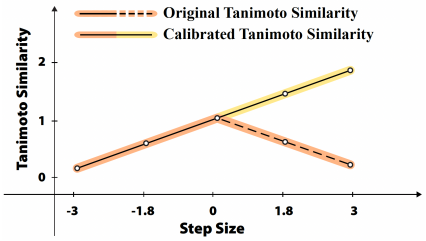
5 Experiments

5.1 Graph Data: Molecular Graphs

Backbone DGMs. We consider two state-of-the-art DGMs for molecular graph generation. MoFlow [60] is a flow-based generative model on molecules which adopts an invertible mapping between the input molecular graphs and a latent prior. HierVAE [24] is a hierarchical VAE model which encodes and decodes molecule atoms and motifs in a hierarchical manner. Besides, the pretrained checkpoints are also provided, on ZINC250K [22] and ChEMBL [34] dataset respectively.

Editing Sequences and Anchor Molecule. As discussed in Sec. 4, the output of the inference in GraphCG is a sequence of edited molecules with the i-th direction, $\{\bar{\boldsymbol{x}}'\}_i$. We first randomly generate a molecule using the backbone DGMs, and we name such molecule as the anchor molecule, $\bar{\boldsymbol{x}}^*$. Then we will take 21 step sizes from -3 to 3, with interval 0.3, to obtain a sequence of 21 molecules following Eq. (2). Note that the edited molecule with step size 0 under the linear editing function is the same as the anchor molecule, *i.e.*, $\bar{\boldsymbol{x}}^*$.

Change of Structure Factors and Evaluation Metrics. We are interested in the change of the graph structure (the steerable factors) along the output sequence edited with the i-th direction. To evaluate the structure change, we apply the Tanimoto similarity between each output molecule and the anchor molecule. Besides, for the ease of evaluating the monotonicity, we utilize a transformation (on the Tanimoto similarity) of output molecules with positive step sizes by taking the deduction from 2. We call this calibrated Tanimoto similarity sequence (CTS). Further, we propose a metric called Sequence Monotonic Ratio (SMR), $\phi_{\text{SMR}}(\cdot, \gamma, \tau)$. It measures the monotonic ratio of M generated sequences with two arguments: the diversity threshold γ constrains the minimum number of distinct molecules, and the ratio tolerance threshold τ controls the non-monotonic tolerance ratio along each sequence.



$$\phi_{\text{SMR}}\big(\{s(x)\},\gamma,\tau\big) = \begin{cases} 1, & \text{len}\big(\text{set}\big(\{s(x)\}\big)\big) \geq \gamma \\ & \quad \wedge \text{monotonic}_\tau\big(\{s(x)\}\big) \\ 0, & \text{otherwise} \end{cases}, \tag{13}$$

$$\phi_{\text{SMR}}(\gamma,\tau)_i = \frac{1}{M}\sum_{m=1}^{M} \phi_{\text{SMR}}\big(\{s(\bar{\boldsymbol{x}}')\}_i^m,\gamma,\tau\big), \tag{14}$$

$$\text{top-K}(\gamma,\tau) = \frac{1}{K}\sum_{i\in\text{top-K directions}} \Big(\phi_{\text{SMR}}(\gamma,\tau)_i\Big). \tag{15}$$

Figure 2: This shows the sequence monotonic ratio (SMR) on calibrated Tanimoto similarity (CTS). Eqs. (13) and (14) are the SMR on each sequence and each direction respectively, where M is the number of generated sequences for the i-th direction and $\{s(\bar{\boldsymbol{x}}')\}_i^m$ is the CTS of the m-th generated sequence with the i-th direction. Eq. (15) is the average of top-K SMR on D directions. More details are in Appendix E.

Table 2: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity w.r.t. the top-1 and top-3 directions. The best performances are marked in **bold**.

Model	Dataset		Tanimoto top-1				Tanimoto top-3			
		diversity γ	3		4		3		4	
		tolerance τ	0	0.2	0	0.2	0	0.2	0	0.2
MoFlow	ZINC250k	Random	23.0	25.0	12.0	15.0	22.0	24.0	11.0	13.7
		Variance	24.0	28.0	12.0	16.0	20.0	25.0	10.0	15.0
		SeFa	4.0	4.0	0.0	0.0	3.3	3.3	0.0	0.0
		DisCo	7.0	14.0	2.0	8.0	5.3	11.7	2.0	7.7
		GraphCG-P	**32.0**	**34.0**	**16.0**	**18.0**	**29.0**	**31.0**	**13.7**	**16.3**
		GraphCG-R	25.0	26.0	11.0	14.0	22.0	24.3	10.3	13.3
HierVAE	ChEMBL	Random	14.0	45.0	14.0	43.0	10.0	42.3	9.3	41.7
		Variance	23.0	59.0	19.0	55.0	18.3	52.7	15.7	50.3
		SeFa	4.0	41.0	4.0	41.0	2.3	36.0	2.3	36.0
		GraphCG-P	40.0	**73.0**	**32.0**	**65.0**	36.0	**64.3**	26.3	**57.7**
		GraphCG-R	**42.0**	67.0	30.0	55.0	**38.0**	62.3	**28.7**	53.7

Evaluating the Diversity of Semantic Directions. To better illustrate that GraphCG is able to learn multiple directions with various semantic information, we also consider taking the average of top-K SMR w.r.t. directions to reveal that all the best K directions are semantically meaningful, as in Eq. (15).

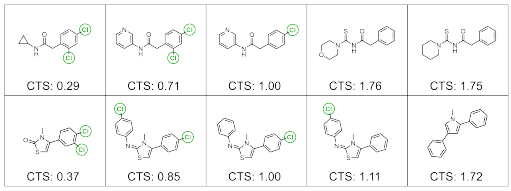
Baselines. For baselines, we consider four unsupervised editing methods. (1) The first is Random. It randomly samples a normalized random vector in the latent space as the semantic direction. (2) The second one is Variance. We analyze the variance on each dimension of the latent space, and select the highest one as the semantic direction. (3) The third one is SeFa [47]. (4) The last one is DisCo [41]. It requires the backbone DGMs to be end-to-end and is infeasible for HierVAE.

Quantitative results. We take $D = 10$ to train GraphCG, and the optimal results on 100 sampled sequences are reported in Table 2. We can observe that GraphCG can show consistently better structure change with both top-1 and top-3 directions.

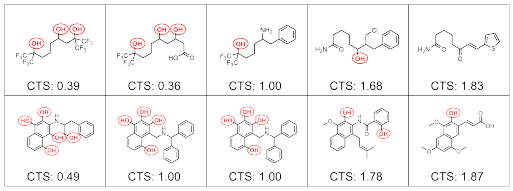
Analysis on steerable factors in molecules: functional group change. For visualization, we sample 8 molecular graph paths along 4 selected directions in Fig. 3, and the backbone DGM is HierVAE pretrained on ChEMBL. The CTS holds good monotonic trend in these sequences. Each direction shows certain unique changes in the steerable factors in molecules. In Fig. 3(a) and Fig. 3(b), the number of halogen atoms and hydroxyl groups (in alcohols and phenols) in the molecules decrease from left to right, respectively. In Fig. 3(c), the number of amides in the molecules increases along the path. Because amides are polar functional groups, the topological polar surface area (tPSA) of the molecules also increase accordingly, which is a key molecular property for the prediction of drug transport properties, *e.g.*, permeability [10]. In Fig. 3(d), the flexible chain length, marked by the number of ethylene (CH_2CH_2) units, increases from left to right. Since the number of rotatable bonds (NRB) measures the molecular flexibility, it also increases accordingly [53].

5.2 Graph Data: Point Clouds

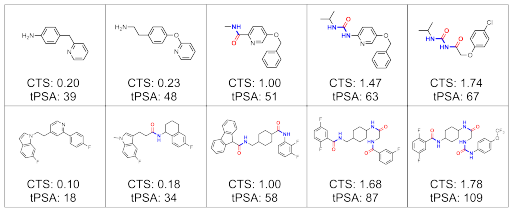
Backbone DGMs. We consider one of the latest DGMs on point clouds, PointFlow [57]. It is using the normalizing flow model for estimating the 3D point clouds distribution. Then we consider PointFlow pretrained on three datasets in ShapeNet [2]: Airplane, Car, and Chair.



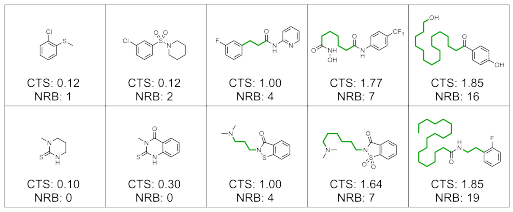
(a) Steerable factor: number of halogens.



(b) Steerable factor: number of hydroxyls.

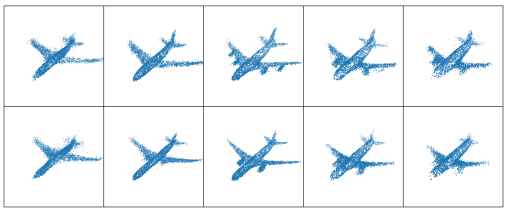


(c) Steerable factor: number of amides.

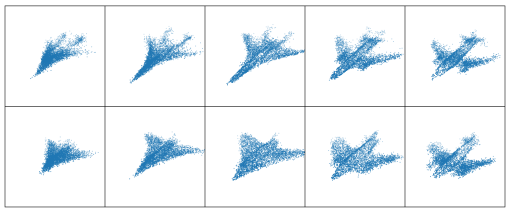


(d) Steerable factor: chain length.

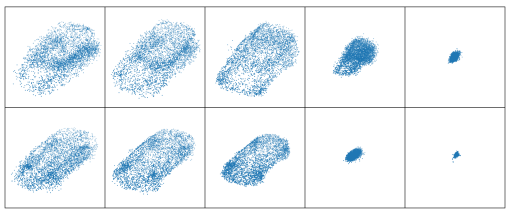
Figure 3: GraphCG for molecular graph editing. We visualize the output molecules and CTS on four directions with two sequences each, where each sequence consists of five steps. The center point is the anchor molecule, and the other four points correspond to step size with -3, -1.8, 1.8, and 3 respectively. Figs. 3(a) to 3(c) show how functional groups in the molecules can be viewed as the steerable factors as they change along the sequence, such as halogen atoms, hydroxyl groups and amides. Fig. 3(d) illustrates the effect on the steerable factor on the length of flexible chains in the molecules. Notably, certain properties change with molecular structures accordingly, like topological polar surface area (tPSA) and number of rotatable bonds (NRB).



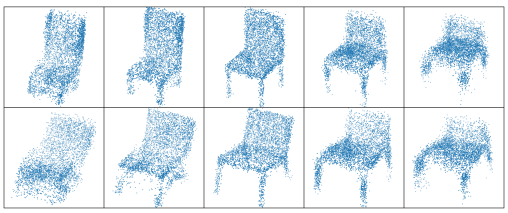
(a) Steerable factor: engine.



(b) Steerable factor: engine.



(c) Steerable factor: size.



(d) Steerable factor: leg height.

Figure 4: GraphCG for point clouds editing. We show four editing sequences, where each sequence consists of five points. The center point is the anchor data, *i.e.*, with step size 0. The other four data points correspond to step size with -3, -1.8, 1.8, and 3, respectively. Fig. 4(a) and Fig. 4(b) correspond to the same semantic direction, and they are showing how to steer the factor engine: the number of engines will be decreased and increased with the negative (left) and positive (right) step size respectively. Similarly, Figs. 4(c) and 4(d) illustrate the effect on the steerable factors on the car size and the chair leg height.

Analysis on steerable factors in point clouds: shape change. To train GraphCG, we take $D = 10$ directions, and we sample 8 point cloud paths along 3 directions for visualization in Fig. 4. More results are in Appendix F. It is observed that GraphCG can steer the shape of the point clouds, *e.g.*, the size of cars and the height of chair legs. We also find it interesting that GraphCG can steer more finger-trained factors, like modifying the number jet engines of airplanes in Figs. 4(a) and 4(b).

6 Conclusion

In this work, we are interested in unsupervised graph editing. It is a well-motivated task for various real-world applications, and we discuss two mainstream data types: molecular graphs and point clouds. We start with exploring the latent space of mainstream deep generative models and propose GraphCG, a model-agnostic and task-agnostic unsupervised method for graph data editing. The key

component of GraphCG is EBM, and we take the GraphCG-NCE as the solution for now. For the future work, we may as well extend it to more advanced solutions like denoising diffusion model [19].

References

[1] Hans-Joachim Böhm, Alexander Flohr, and Martin Stahl. Scaffold hopping. *Drug discovery today: Technologies*, 1(3):217–224, 2004.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015.

[3] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, et al. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv:2003.06060*, 2020.

[4] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*, 2018.

[5] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv:2004.05718*, 2020.

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

[7] Jürgen Drews. Drug discovery: a historical perspective. *Science*, 287(5460):1960–1964, 2000.

[8] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, et al. Convolutional networks on graphs for learning molecular fingerprints. *arXiv:1509.09292*, 2015.

[9] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *ICLR*, 2018.

[10] Peter Ertl, Bernhard Rohde, and Paul Selzer. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *Journal of medicinal chemistry*, 43(20):3714–3717, 2000.

[11] Peter Ertl, Eva Altmann, and Jeffrey M McKenna. The most common functional groups in bioactive molecules and how their popularity has evolved over time. *Journal of medicinal chemistry*, 63(15): 8408–8418, 2020.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[13] Laurent Gomez. Decision making in medicinal chemistry: The power of our intuition. *ACS Medicinal Chemistry Letters*, 9(10):956–958, 2018.

[14] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

[15] Erik Härkönen, Aaron Hertzman, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *NeurIPS*, 2020.

[16] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, et al. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[18] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[20] Ye Hu, Dagmar Stumpfe, and Jurgen Bajorath. Recent advances in scaffold hopping: miniperspective. *Journal of medicinal chemistry*, 60(4):1238–1246, 2017.

[21] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[22] John J Irwin and Brian K Shoichet. Zinc: a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

[23] Ali Jahanian, Lucy Chai, and Phillip Isola. On the" steerability" of generative adversarial networks. In *ICLR*, 2019.

[24] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *ICML*, 2020.

[25] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[26] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *ICML*, 2018.

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

[28] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *ICLR*, 2018.

[29] Greg Landrum et al. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

[30] Shengchao Liu, Mehmet Furkan Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *arXiv:1806.09206*, 2018.

[31] Shengchao Liu, Hongyu Guo, and Jian Tang. Molecular geometry pretraining with se (3)-invariant denoising distance matching. *arXiv preprint arXiv:2206.13602*, 2022.

[32] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, et al. Pre-training molecular graph representation with 3d geometry. In *ICLR*, 2022.

[33] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, et al. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

[34] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

[35] Zlatko Mihalić and Nenad Trinajstić. A graph-theoretical approach to structure-property relationships. *Journal of Chemical Education*, 69(9):701, 1992.

[36] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. In *NeurIPS*, 2021.

[37] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[38] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[39] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. Shift-map image editing. In *ICCV*, 2009.

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[41] Xuanchi Ren, Tao Yang, Yuwang Wang, and Wenjun Zeng. Do generative models know disentanglement? contrastive learning is all you need. *arXiv:2102.10543*, 2021.

[42] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[43] Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *NeurIPS*, 2018.

[44] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.

[45] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *IROS*, 2010.

[46] Paul G. Seybold, Michael May, and Ujjvala A. Bagal. Molecular structure: Property relationships. *Journal of Chemical Education*, 64(7):575, 1987.

[47] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021.

[48] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE TPAMI*, 2020.

[49] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *CVPR*, 2020.

[50] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv:1907.05600*, 2019.

[51] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv:2101.03288*, 2021.

[52] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, et al. Score-based generative modeling through stochastic differential equations. *arXiv:2011.13456*, 2020.

[53] Daniel F Veber, Stephen R Johnson, Hung-Yuan Cheng, Brian R Smith, Keith W Ward, and Kenneth D Kopple. Molecular properties that influence the oral bioavailability of drug candidates. *Journal of medicinal chemistry*, 45(12):2615–2623, 2002.

[54] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, et al. Dynamic graph cnn for learning on point clouds. *ACM ToG*, 2020.

[55] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. Retrieval-based controllable molecule generation. *arXiv preprint arXiv:2208.11126*, 2022.

[56] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *CVPR*, 2021.

[57] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, et al. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019.

[58] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[59] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[60] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *KDD*, 2020.

A Graph Data

A.1 Molecules

Molecules can be naturally represented as the 2D molecular graphs, where atoms and bonds are nodes and edges in the graph, respectively. For the recent years, graph representation learning has been extensively explored on the molecular graph [8, 12, 30, 58, 5]. Based on the graph representation of molecules, a variety of work have been done for molecule generation. The state-of-the-art ones include MoFlow [60] and HierVAE [24].

A.2 Point Clouds

A point cloud is represented as a set of points, where each point P_i is described by a vector of 3D Euclidean coordinates possibly with extra channels (e.g., colors, surface normals and returned laser intensity). In 2017, Qi et. al [40] designed a deep learning framework called PointNet that directly consumes unordered point sets as inputs and can be used for various tasks such as classification and segmentation. For the generative models on point clouds, we consider one of the latest work, PointFlow [57].

B Related Work

Image editing and image controllable generation Many existing works on controllable generation with DGMs mainly focus on the image data. With the assumption that the learned latent space already include rich semantic information [25, 23, 48, 15], the question then becomes how to identify semantic-rich directions from the latent space of DGMs. Depending on whether or using supervised signals to discover the semantic directions, existing works can be divided into two settings: *supervised* and *unsupervised*.

The *supervised* setting relies on the supervised signals to learn the pre-defined semantic-rich directions. For instance, InterfaceGAN [48] identifies the semantic directions in the latent space via linear models that recognize semantic boundary among data. LACE [36] uses energy-based models to learn the joint distribution of data and properties (*i.e.*, semantics) and formulate the sampling process as to solve an ordinary differential equation.

As supervised signals usually require domain knowledge and laborious annotations, latest works are more focused on the *unsupervised* setting, either model-specific or data-specific. Specifically, GANSpace [15] applies PCA on the intermediate layers of GANs (instead of latent space) for learning the semantic-rich directions. SeFa [47] exploits the pretrained GANs and extracts the semantic-rich directions by using PCA on the backbone layers. Nevertheless, as both methods are specifically designed for StyleGAN [25], it is nontrivial to generalize them to other DGMs. A more recent work DisCo [41] employs a different pipeline: it trains a new encoder after reconstruction, and maps the generated images to another latent space for editing. However, training a new encoder introduces extra complexities.

Graph editing and graph controllable generation This is an emerging field with many downstream applications [7, 49]. However, existing works are mainly focusing on the supervised setting. For example, conventional approaches, such as genetic algorithms (GAs), edit the molecule graphs in the data space via hand-crafted heuristics with the guidance of moleular property predictors. More recent learning based methods perform the latent direction discovery, either by training a classifier on the latent space of DGMs on the graph data [24] or by learning to retrieve exemplar samples from a retrieval database for guidance [55]. Other works fine-tune a pre-trained graph DGM using the supervised property annotations as rewards, resulting a controllable DGM specifically for the considered task [37, 59]. To the best of our knowledge, our work is the first to explore the unsupervised graph editing in the unsupervised manner. Besides, different from many previous approches that may only work for molecule graphs or point cloud graphs, our method is generic and thus can be applied to various graph modalities.

C Analysis Experiments on Disentanglement

In Sec. 3, we conduct three analysis experiments to conclude that the representation space is not perfectly disentangled in such the setting. In this section, we provide more details and complementary information of these experiments.

C.1 Steerable Factors

As mentioned in Sec. 3, we consider the measuring the disentanglement with respect to three types of structured data: molecular graphs and point clouds. Recall that we need to define steerable factors first, so as to measure the disentanglement.

Table 3: The six mainstream disentanglement metrics on five DGMs and three data types. All measures range from 0 to 1, and higher scores mean more disentangled representation. MoFlow and HierVAE are for molecular graphs, PointFlow is for point clouds.

Data Type	Model	Dataset	BetaVAE ↑	FactorVAE ↑	MIG ↑	DCI ↑	Modularity ↑	SAP ↑
Molecular Graphs	MolFlow	QM9	0.251	0.165	0.038	0.727	0.599	0.017
		ZINC250k	0.264	0.175	0.030	0.958	0.620	0.009
	HierVAE	QM9	0.165	0.135	0.044	0.241	0.626	0.076
		ChEMBL	0.159	0.130	0.023	0.113	0.604	0.026
Point Clouds	PointFlow	Airplane	0.022	0.025	0.029	0.160	0.745	0.022
		Chair	0.019	0.014	0.032	0.149	0.721	0.019
		Car	0.019	0.023	0.021	0.120	0.713	0.021

Molecular Graph For molecular graphs, we treat the important substructures (a.k.a., motifs or fragments) as factors, and they are extracted using RDKit. We consider the existence of the following 9 motifs as the binary factor labels:

- aliphatic hydroxyl groups.
- aliphatic hydroxyl groups excluding tert-OH.
- aromatic nitrogens.
- aromatic amines.
- Tertiary amines.
- Secondary amines.
- amides.
- ether oxygens (including phenoxy).
- nitriles.

Point Clouds For point clouds, we adopt the viewpoint feature histogram (VFH) [45] implemented in PCL [44]. There are 308 bins in total, where each bin is a histogram of the angles that viewpoint direction makes with each normal. VFH has been widely used as point cloud descriptors, and here we binarize it into factors with:

- We collect the shared non-zero bins from all three datasets (Airplane, Car, and Chair), and ignore the bins where the values distribution are highly skewed. This can give us 75 bins.
- Then for each of these selected bins, we use the median value as the threshold to generate the binary factor labels.

C.2 Disentanglement Measures

We follow the [33] on considering the following six disentanglement measures:

- β-VAE [17] evaluates the prediction accuracy of a linear classifier for the index of a fixed factor of variation.
- FactorVAE [26] addresses the limitations (i.e. corner case) of β-VAE via introducing a majority voting classifier on a different feature vector.
- MIG [4] measures the normalized difference between the highest and second highest mutual information between latent dimensions and each steerable factor.
- DCI [9] disentanglement score measures the average difference from one of the entropy of probability that a latent dimension is useful for predicting a steerable factor (computed by the relative importance score).
- Modularity [43] measures whether each latent dimension is dependent on at most one single steerable factor. It computes the average normalized squared difference over the highest and second highest mutual information between each steerable factor and each latent dimension.
- SAP [28] calculates the R^2 score of the linear models trained to predict each steerable factor from each latent dimension.

Recall that all six disentanglement measures range from 0 to 1, and higher value means the corresponding space is more disentangled. The results can be found in Table 3. We can conclude that all the values are indeed low on all datasets and models, revealing that DGMs are not well-disentangled in general.

D Implementation and Experiment Details

Editing function. For the editing function, we consider both linear (Eqs. (16) and (17)) and non-linear (Eq. (18)) cases as below, *i.e.*, for $\bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}_i, \alpha)$:

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \quad \boldsymbol{d}_i = \text{norm} \circ \text{Linear}(\boldsymbol{e}_i), \tag{16}$$

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \quad \boldsymbol{d}_i = \text{sqrt} \circ \text{norm} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{e}_i), \tag{17}$$

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i + \text{norm} \circ \text{Linear} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{z} \oplus \boldsymbol{d}_i \oplus [\alpha]), \quad \boldsymbol{d}_i = \text{norm} \circ \text{Linear} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{e}_i), \tag{18}$$

where $\circ$ means the composition of two functions.

Objective function. The objective function is given as:

$$\mathcal{L} = c_1 \cdot \mathbb{E}_{u,v}[\mathcal{L}_{\text{GraphCG-NCE}}] + c_2 \cdot \mathcal{L}_{\text{sim}} + c_3 \cdot \mathcal{L}_{\text{sparsity}}, \tag{19}$$

where $\mathcal{L}_{\text{GraphCG-NCE}}$ is the MI estimation defined in Eq. (9), $\mathcal{L}_{\text{sim}}$ is the direction similarity defined in Eq. (10), and $\mathcal{L}_{\text{sparsity}}$. c_1, c_2 and c_3 are three coefficients accordingly.

Hyperparameters. We list the key hyperparameters in Table 4, and all the results are evaluated on 100 sampled sequences. We also want to highlight that DisCo is an unstable baseline, in the sense that once we add more training data (*e.g.*, from 100 to 500) or more training epochs (*e.g.*, from 1 epoch to 5 epochs), the model will collapse, with the nan loss. Thus, here we are reporting the most reasonable results for DisCo, *i.e.*, 100 training data with 1 epoch.

Table 4: Hyperparameter specifications.

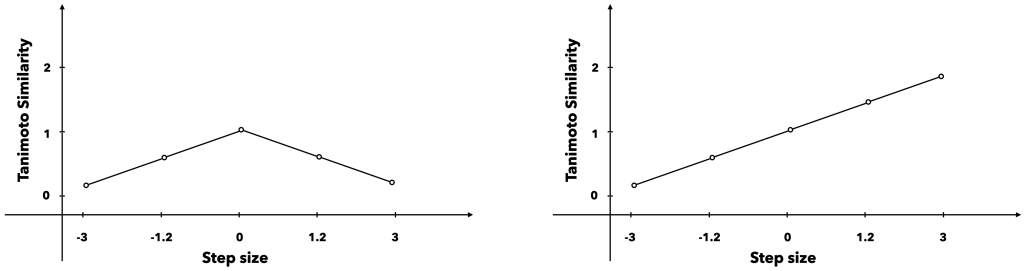
	Hyperparameter	Value
Random	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
Variance	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
SeFa	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
DisCo	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100}
	epochs	{1}
GraphCG	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
	epochs	{20, 100}
	coefficient c_1	{0, 1}
	coefficient c_2	{0, 1}
	coefficient c_3	{0, 1}

Hardware. We use V100 GPU cards, and each job (w.r.t. different hyperparameters) for GraphCG can be finished within 3 hours.

E Results: Molecular Graph

E.1 Evaluation Metrics

Change of Structure Factors and Calibrated Tanimoto Similarity (CTS). We are interested in the change of the graph structure (the steerable factors) along the output sequence edited with the i-th direction. To evaluate the structure change, we apply the Tanimoto similarity between each output molecule and the anchor molecule, as shown in Fig. 5(a). Besides, for the ease of evaluating the monotonicity, we utilize a transformation (on the Tanimoto similarity) of output molecules with positive step size by taking the deduction from 2. We call this calibrated Tanimoto similarity sequence (CTS), *i.e.*, $\{s(\bar{\boldsymbol{x}}')\}_i$, as shown in Fig. 5(b).



(a) Tanimoto Similarity Sequence (b) Calibrated Tanimoto Similarity Sequence

Figure 5: Fig. 5(a) is the original Tanimoto similarity sequence w.r.t. the anchor molecule, *i.e.*, step size with 0 in the figure. Yet, this is not easy to compute the monotonicity. We thus propose the calibrated Tanimoto similarity sequence, by taking the deduction from 2 for output molecules with positive step size, as shown in Fig. 5(b).

Sequence Monotonic Ratio (SMR). For evaluation, we propose a metric called Sequence Monotonic Ratio (SMR), $\phi_{\text{SMR}}(\gamma, \tau)_i$. It measures the monotonic ratio of M generated sequences edited with the i-th direction. It has two arguments: the diversity threshold γ constrains the minimum number of distinct molecules, and the tolerance threshold τ controls the non-monotonic tolerance ratio along each sequence.

In specific, for each learned semantic direction i, we will generate M sequences of edited molecules, and the calibrated Tanimoto similarity for each sequence is marked as $\{s(\bar{\boldsymbol{x}}')\}_i^m$. Then we can define the SMR on each direction as:

$$\begin{aligned}
&\phi_{\text{SMR}}(\gamma, \tau)_i = \frac{1}{M} \sum_{m=1}^{M} \phi_{\text{SMR}}\big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\big), \\
&\phi_{\text{SMR}}\big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\big) = \begin{cases} 1, & \text{len}\big(\text{set}\big\{s(\bar{\boldsymbol{x}}')\}_i^m\big)\big) \geq \gamma \wedge \ \text{monotonic}_\tau\big(\{s(\bar{\boldsymbol{x}}')\}_i^m\big) \\ 0, & \text{otherwise} \end{cases}.
\end{aligned} \tag{20}$$

Evaluating the Diversity of Semantic Directions. SMR can evaluate all the output sequences generated by one direction. To better illustrate that GraphCG is able to learn multiple directions with various semantic information, we also consider taking the average of top-K SMR w.r.t. directions to reveal that all the best K directions are semantically meaningful, as in Eq. (21):

$$\text{top-K}(\gamma, \tau) = \frac{1}{K} \sum_{i \in \text{top-K directions}} \Big(\phi_{\text{SMR}}(\gamma, \tau)_i\Big). \tag{21}$$

E.2 Results on Molecular Structures

Next we would like to show the comprehensive SMR on the CTS results with respect to different backbone models, as in Tables 5 and 6.

Table 5: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity (CST) w.r.t. the top-1, top-2, and top-3 directions. The backbone model is the pretrained MoFlow on ZINC250k.

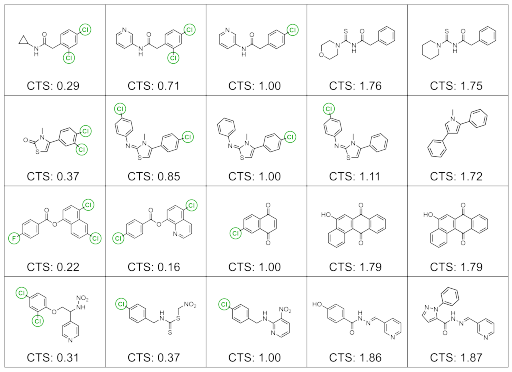
Edit Method	top-K	**Tanimoto top-1**						**Tanimoto top-2**						**Tanimoto top-3**					
	γ	2		3		4		2		3		4		2		3		4	
	τ	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2
Random		35.0	36.0	23.0	25.0	12.0	15.0	34.5	36.0	22.5	25.0	11.5	14.0	34.0	36.0	22.0	24.0	11.0	13.7
Variance		32.0	36.0	24.0	28.0	12.0	16.0	31.5	35.5	21.0	26.5	10.5	16.0	30.3	35.3	20.0	25.0	10.0	15.0
SeFa		23.0	23.0	4.0	4.0	0.0	0.0	19.0	19.0	4.0	4.0	0.0	0.0	17.3	17.3	3.3	3.3	0.0	0.0
DisCo		8.0	15.0	7.0	14.0	2.0	8.0	7.5	13.5	6.0	12.5	2.0	8.0	7.0	13.0	5.3	11.7	2.0	7.7
GraphCG-P	Eq. (16)	39.0	40.0	27.0	28.0	15.0	18.0	38.5	40.0	26.0	28.0	15.0	18.0	37.0	39.0	24.7	27.3	14.3	17.3
	Eq. (17)	35.0	37.0	19.0	22.0	8.0	11.0	33.5	36.5	18.5	21.5	7.0	10.5	31.7	34.7	17.7	20.7	6.3	9.3
	Eq. (18)	44.0	46.0	32.0	34.0	16.0	18.0	42.5	44.5	30.0	32.0	15.0	17.5	41.7	44.0	29.0	31.0	13.7	16.3
GraphCG-R	Eq. (16)	37.0	42.0	25.0	26.0	11.0	14.0	37.0	40.0	23.0	25.5	11.0	13.5	36.3	39.0	22.0	24.3	10.3	13.3
	Eq. (17)	35.0	37.0	19.0	22.0	8.0	11.0	33.5	36.5	18.5	21.5	7.0	10.5	31.7	34.7	17.7	20.7	6.3	9.3
	Eq. (18)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 6: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity (CST) w.r.t. the top-1, top-2, and top-3 directions. The backbone model is the pretrained HierVAE on ChEMBL.

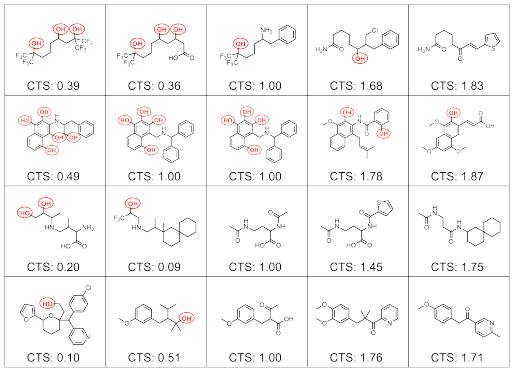
Edit Method	top-K	**Tanimoto top-1**						**Tanimoto top-2**						**Tanimoto top-3**					
	γ	2		3		4		2		3		4		2		3		4	
	τ	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2
Random		14.0	45.0	14.0	45.0	14.0	43.0	11.0	43.5	11.0	43.5	11.0	42.5	10.0	42.3	10.0	42.3	9.3	41.7
Variance		28.0	64.0	23.0	59.0	19.0	55.0	22.5	59.5	19.5	57.0	17.5	55.0	20.3	54.3	18.3	52.7	15.7	50.3
SeFa		4.0	41.0	4.0	41.0	4.0	41.0	3.0	41.0	3.0	41.0	3.0	41.0	2.3	36.0	2.3	36.0	2.3	36.0
GraphCG-P	Eq. (16)	23.0	61.0	19.0	57.0	15.0	53.0	19.0	59.0	16.5	56.5	13.5	53.0	17.0	55.3	15.3	53.7	12.7	50.7
	Eq. (17)	62.0	77.0	40.0	73.0	32.0	65.0	60.5	74.0	38.5	67.5	29.0	61.0	59.3	70.3	36.0	64.3	26.3	57.7
	Eq. (18)	29.0	71.0	28.0	70.0	27.0	69.0	22.0	62.0	21.5	61.5	20.5	61.0	18.7	57.7	18.3	57.3	17.7	56.7
GraphCG-R	Eq. (16)	16.0	56.0	16.0	56.0	15.0	55.0	13.5	48.0	13.5	48.0	12.0	47.0	11.7	44.7	11.7	44.7	10.7	43.3
	Eq. (17)	61.0	74.0	42.0	67.0	30.0	55.0	59.0	69.5	40.0	64.0	29.5	54.5	57.7	67.7	38.0	62.3	28.7	53.7
	Eq. (18)	25.0	57.0	24.0	57.0	21.0	55.0	20.0	55.0	19.5	54.5	17.0	52.0	17.3	52.7	17.0	52.3	15.0	50.0

E.3 Visualization

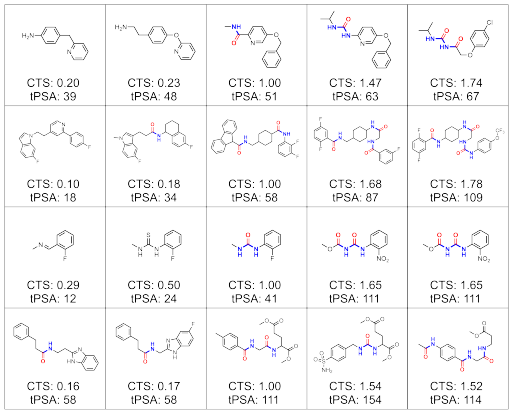
For a more comprehensive visualization of the steerable factors in molecular graphs, we demonstrate 16 molecular graph paths along the 4 selected directions in Fig. 6, and the backbone DGM is HierVAE pretrained on ChEMBL. The CTS holds good monotonic trend in all these sequences. Each direction shows certain unique changes in the molecular structures, *i.e.*, the steerable factors in molecules. Some structural changes are reflected in molecular properties. We expand all the details below. In Fig. 6(a) and Fig. 6(b), the number of halogen atoms and hydroxyl groups (in alcohols and phenols) in the molecules decrease from left to right, respectively. In Fig. 6(c), the number of amides in the molecules increases along the path. As a result, the topological polar surface area (tPSA) of the molecules increase accordingly, which is a key molecular property for the prediction of drug transport properties, *e.g.*, permeability [10]. In Fig. 3(d), the flexible chain length, marked by the number of ethylene (CH_2CH_2) units, increases from left to right. Since the number of rotatable bonds (NRB) measures the molecular flexibility, it also increases accordingly [53].



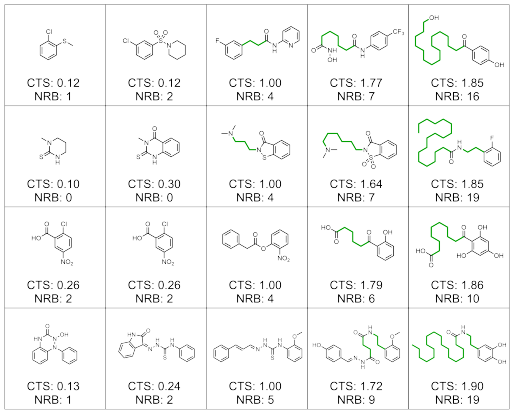
(a) Steerable factor: number of halogens.



(b) Steerable factor: number of hydroxyls.



(c) Steerable factor: number of amides.



(d) Steerable factor: chain length.

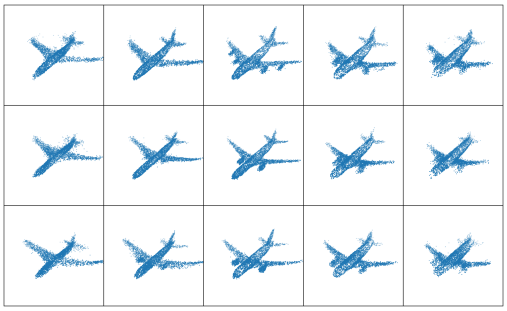
Figure 6: GraphCG for molecular graph editing. We visualize the output molecules and CTS on four directions with four sequences each, where each sequence consists of five steps. The center point is the anchor molecule, and the other four points correspond to step size with -3, -1.8, 1.8, and 3 respectively. Fig. 6(a) to Fig. 6(c) show how functional groups in the molecules can be viewed as the steerable factors as they change along the path, such as halogen atoms, hydroxyl groups and amides. Fig. 6(d) illustrates the effect on the steerable factor on the length of flexible chains in the molecules. Notebly, certain properties change together with molecular structures, like topological polar surface area (tPSA) and number of rotatable bonds (NRB).

F Results: Point Clouds

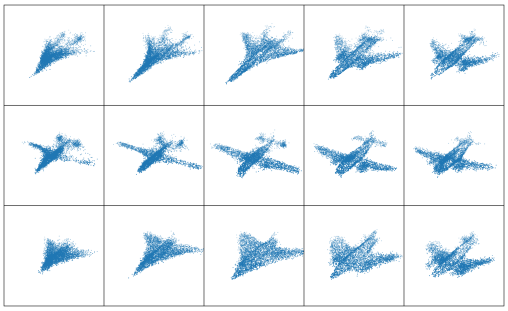
Here we compare two editing functions, *i.e.*, the key function design in GraphCG. We provide the visualizations for the linear editing function in Appendix F.1, and non-linear editing function in Eq. (18).

First we want to highlight that all the samples are generated randomly. In the linear case, we can observe that the shape of the airplanes, cars, and chairs, are steerable using GraphCG. We also find it interesting that GraphCG can steer more finger-trained factors, like modifying the airplane engines. However, in the non-linear case, the diversity of the edited data is smaller. This can been observed from the middle columns in Appendix F.2. We will leave this for future exploration.

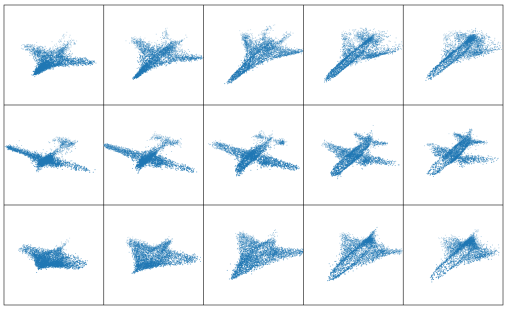
F.1 Linear Editing Function



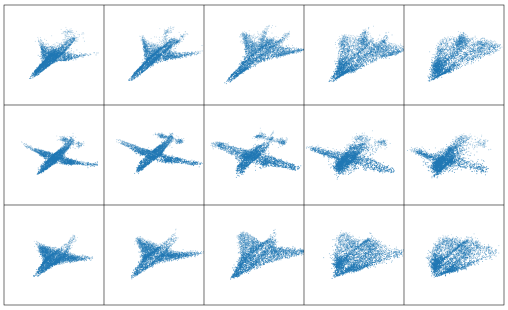
(a) Steerable factor: engine (direction 1).



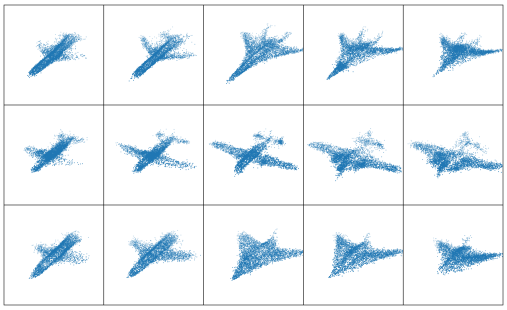
(b) Steerable factor: engine (direction 1).



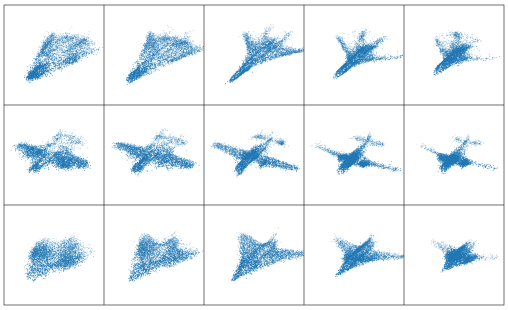
(c) Steerable factor: fuselage length (direction 2).



(d) Steerable factor: wing size (direction 3).

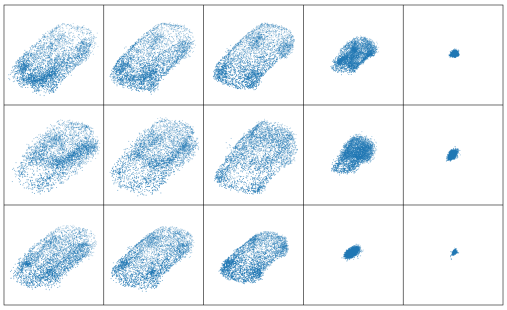


(e) Steerable factor: wing shape(direction 4).

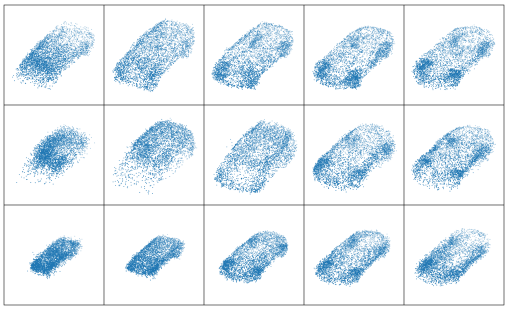


(f) Steerable factor: wing thickness (direction 5).

Figure 7: GraphCG for point clouds (Airplane) editing. It can successfully reflect these steerable factors: engine, fuselage length, wing size, wing shape, and wing thickness.

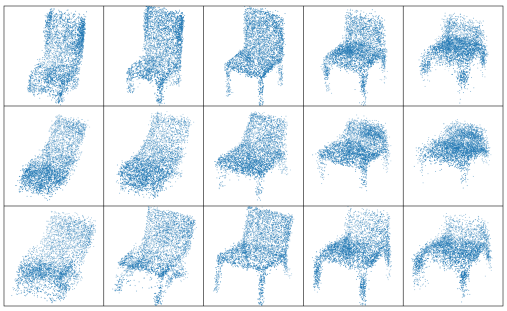


(a) Steerable factor: size (direction 1).

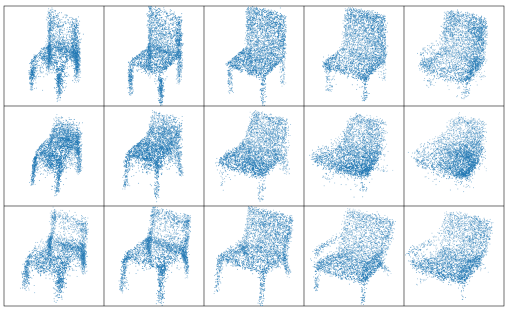


(b) Steerable factor: size (direction 2).

Figure 8: GraphCG for point clouds (Car) editing. The steerable factors on this dataset are not obvious, and here we only plot the car size editable with two directions.



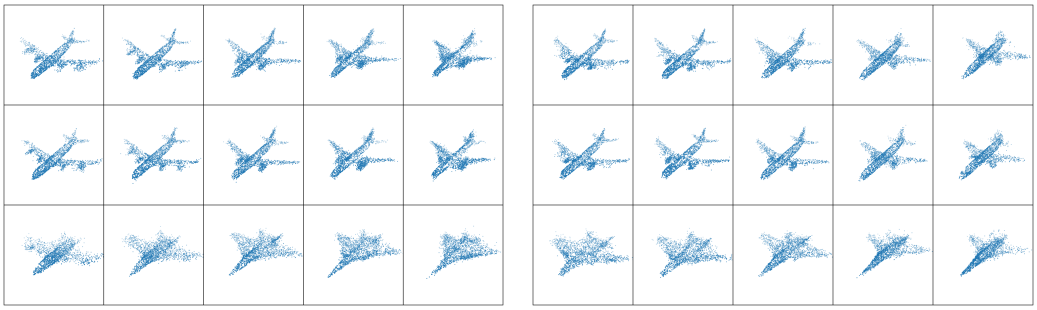
(a) Steerable factor: leg height (direction 1).



(b) Steerable factor: seat size (direction 2).

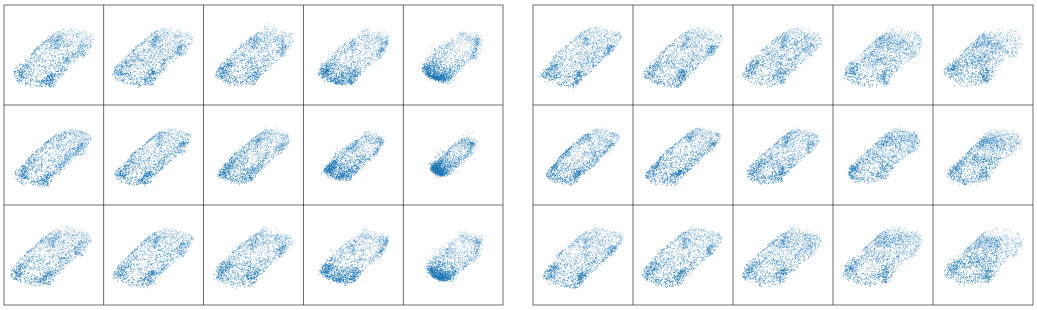
Figure 9: GraphCG for point clouds (Chair) editing. It can successfully reflect these steerable factors: leg height, and seat size.

F.2 Non-linear Editing Function



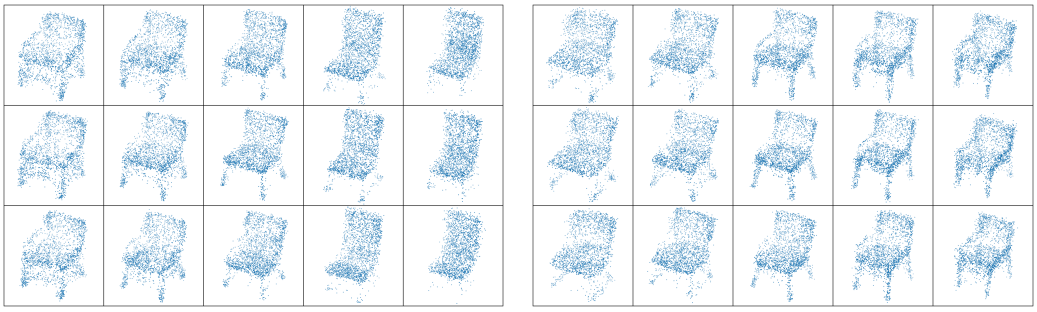
(a) Steerable factor: wing shape (direction 1). (b) Steerable factor: wing length (direction 2).

Figure 10: GraphCG for point clouds (Airplane) editing. It can successfully reflect these steerable factors: wing shape and wing length.



(a) Steerable factor: size (direction 1). (b) No obvious steerable factor (direction 2).

Figure 11: GraphCG for point clouds (Car) editing. The steerable factors on this dataset are not obvious, and here we only plot the car size editable with one directions.



(a) Steerable factor: leg height (direction 1). (b) Steerable factor: seat size (direction 2).

Figure 12: GraphCG for point clouds (Chair) editing. It can successfully reflect these steerable factors: leg height, and seat size.