MOLECULE GENERATION BY HETEROPHILOUS TRIPLE FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating molecules with desirable properties is key to domains like material design and drug discovery. The predominant approach is to encode molecular graphs using graph neural networks or their continuous-depth analogues. However, these methods often implicitly assume strong homophily (*i.e.*, affinity) between neighbours, overlooking repulsions between dissimilar atoms and making them vulnerable to oversmoothing. To address this, we introduce *HTFlows*. It uses multiple interactive flows to capture heterophily patterns in the molecular space and harnesses these (dis-)similarities in generation, consistently showing good performance on chemoinformatics benchmarks.

1 INTRODUCTION

Identifying molecular candidates with specific chemical properties is an integral task in important biochemistry domains such as material design and drug discovery. However, traditional methods rely on expensive exploratory experiments that involve time and resource-intensive investigations (Paul et al., 2010), hindered by the inherent discreteness of the search space and its vast combinatorial possibilities (Reymond et al., 2012; Polishchuk et al., 2013). Deep generative models can employ effective inductive biases to encode molecules and expedite the discovery process by narrowing down the search space; *e.g.*, they have recently shown significant potential for suggesting promising drug candidates *in silico* (Ingraham et al., 2019; Polykovskiy et al., 2020).

Molecules can be presented as input to a deep learning model in different formats. Initial works, *e.g.*, Kusner et al. (2017), Dai et al. (2018) posed molecular generation as an autoregressive problem, utilizing SMILES (short for 'Simplified Molecular-Input Line-Entry System'), *i.e.*, a unique sequence representation for molecules (Landrum et al., 2013). However, the mapping from molecules to SMILES is not continuous, so similar molecules can be assigned vastly different string representations. Graphs provide an elegant abstraction to encode the interactions between the atoms in a molecule, so powerful encoders based on graph neural networks (GNNs, Scarselli et al., 2009; Kipf & Welling, 2017; Veličković et al., 2018; Xu et al., 2019; Garg et al., 2020) have been adopted in recent years. A range of deep learning frameworks have been integrated with GNNs for molecule generation, including, adversarial models (De Cao & Kipf, 2018; You et al., 2018), diffusion models (Hoogeboom et al., 2022), energy-based models (Liu et al., 2021b), and Neural ODEs (Verma et al., 2022) and other flow-based models (Shi et al., 2019; Luo et al., 2021; Zang & Wang, 2020).

We seek to illuminate, and address, a key issue that has been overlooked while using GNNs in molecule generation settings. Standard GNNs employ local message-passing steps on each input graph to exchange information between nodes and their neighbours; implicitly assuming strong *homophily*, *i.e.*, tendency of nodes to connect with others that have similar labels or features. This assumption turns out to be reasonable in settings such as social (McPherson et al., 2001), regional planning (Gerber et al., 2013), and citation (Ciotti et al., 2016) networks. However, heterophilous graphs violate this assumption leading to sub-optimal performance (Zhu et al., 2020; 2021; Chien et al., 2021; Lim et al., 2021; Wang et al., 2023), owing to *oversmoothing* (Li et al., 2018) resulting from flattening of high-frequency information (Wu et al., 2023) by message-passing schemes.

We shed light on this issue with the standard QM9 data in Fig. 1. A conceptual way to characterize homophily is by examining the neighbours of each node. A fully *homophilous* molecule only has links between atoms of the same type (right), while a *heterophilous* molecule has links between different types (left). We observe that a major fraction of molecules in QM9 have scores in the range [0.4, 0.8].

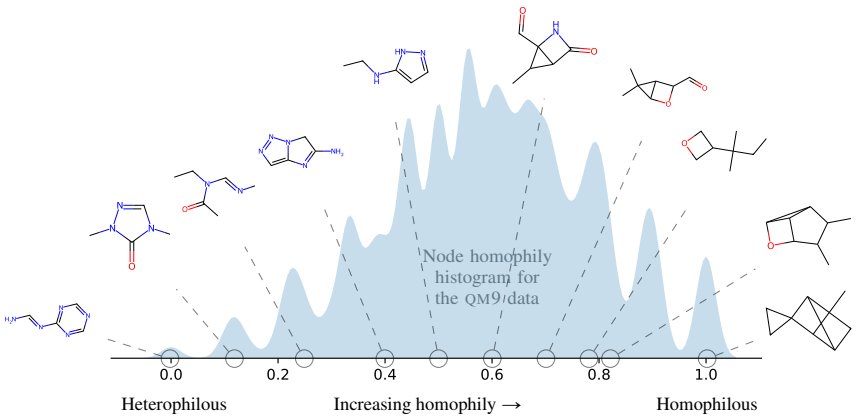


Figure 1: A simple way to characterize homophily is by studying the links of each node. A fully *homophilous* molecule only has links between atoms of the same type, while a *heterophilous* molecule has links between different types. Rather than counting links, HTFlow utilizes multiple interactive flows to estimate the propensity of a link to be homophilic/heterophilic in a given molecular context.

However, in practice, simply counting atom types is not expressive enough. Instead, the heterophily typically stems from more intricate properties of the molecules which need to be learned from data. We introduce HTFlows to carefully address and utilize the heterophily present in molecular data during generative modelling.

Our contributions In this paper, we introduce a novel framework for flow-based graph generation, likely the first molecular generation model that directly accounts for data heterophily. The proposed model comprises several interactive flows, designed to learn graph structures and node features across varying degrees of homophily and heterophily. Our key contributions are summarized below:

- **(Conceptual and technical)** we motivate the relevance of heterophily in molecular contexts, and propose a generative framework that encodes homophily/heterophily patterns;
- **(Methodological)** we design an invertible model with three co-evolving flows: a central flow interacts with heterophilous and homophilous flows to learn nuanced representations;
- **(Empirical)** we demonstrate the benefits of our method by benchmarking molecule generation on the QM9 and ZINC-250K data sets, evaluating with an extensive set of 14 different chemoinformatics metrics to analyze the actual chemical properties of the generated data.

Notable advantages of our model include achieving high validity without the need for additional validity checks in random generation experiments and successful optimization of target chemical properties in molecular searches. We now proceed to reviewing relevant related works.

2 Related work

Molecule representation and generation Early works in molecule generation (*e.g.*, Kusner et al., 2017; Guimaraes et al., 2017; Gómez-Bombarelli et al., 2018; Dai et al., 2018) primarily used sequence models to encode the SMILES strings (Weininger et al., 1989). Graphs afford more flexible modeling of interactions, so the field has gravitated towards representing molecules as (geometric) graphs and using powerful graph encoders, *e.g.*, based on graph neural networks (GNNs).

Variational autoencoders (VAEs, Kingma & Welling, 2014) provided a toolset for molecule generation with an encoder-decoder architecture, affording a latent encoding that can be optimized to search for molecules with specific properties. A prominent work, JT-VAE, showed benefits of viewing graphs as tree-like substructures obtained by including rings, in addition to the usual atoms labels, as part of the vocabulary (Jin et al., 2018). Other models such as Graph Convolutional Policy Network (You et al., 2018) and MolecularRNN (Popova et al., 2019; Shi et al., 2019; Luo et al., 2021) add atoms/bonds sequentially, and rely on rejection schemes to ensure the validity of the generated

molecules. Generative Adversarial Networks (GANs, Goodfellow et al., 2014) introduced added flexibility, as demonstrated by works such as De Cao & Kipf (2018) and You et al. (2018).

Flow-based models Normalizing flows enable exact likelihood estimation (see Papamakarios et al., 2021), so have recently gained prominence in the context of molecule generation (Kaushalya et al., 2019; Luo et al., 2021; Shi et al., 2019; Zang & Wang, 2020; Verma et al., 2022). These models learn invertible transformations to map data from a simpler base distribution to a more complex distribution over molecules. GraphAF (Shi et al., 2019) and GraphDF (Luo et al., 2021) keep the traditional sequential generation process, with GraphDF constraining the latent variables to be discrete. MoFlow (Zang & Wang, 2020) leverages a GLOW model (Kingma & Dhariwal, 2018) for structure generation with a conditional flow for assigning atom types.

More recently, there has been a shift towards incorporating prior knowledge and stronger inductive biases into deep learning models for molecule generation, thus allowing for more nuanced and accurate representations. ModFlow (Verma et al., 2022) builds a continuous normalizing flow with graph neural ODEs (Poli et al., 2019) assuming molecular structure is available, and use an E(3)-equivariant GNN (Satorras et al., 2021) to account for rotational and translational symmetries. EDM (Hoogeboom et al., 2022) generates molecules in 3D space through an equivariant diffusion model (Sohl-Dickstein et al., 2015; Song et al., 2021; Austin et al., 2021; Vignac et al., 2022) on the atom coordinates and categorical types. This relates to ongoing interest in guiding the generative process by controlling the inductive biases of the model. Such structure is perhaps more apparent in image generation (*e.g.*, Rissanen et al., 2023; Hoogeboom & Salimans, 2023), while in molecule modelling the prior knowledge needs to be included in more subtle ways, such as in the form of heterophily.

Heterophily Many previous studies analyze how heterophily influences GNN performance and design new methods to mitigate it (Zhu et al., 2020; Liu et al., 2021a; Yan et al., 2022; Ma et al., 2021). Some studies demonstrate deeper insights about how heterophily affects model expressiveness (Ma et al., 2021; Luan et al., 2022; Mao et al., 2023; Luan et al., 2023). However, most of these papers focus on node classification. However, molecular generation requires models to learn the data distribution by distinguishable graph embeddings. Heterophilic graphs lose distinguishability more from message-passing layers. We now address this issue with HTFlows.

3 Heterophilous Triple Flows

We propose a graph generative model leveraging normalizing flows and heterophily features in graph data. Our model is split into two main components: the bond flow and the atom flow. The bond flow focuses on learning the molecular structure, while the atom flow assigns specific atomic details to this topology.

3.1 Prerequisites: Normalizing Flows with Affine Coupling Layers

Normalizing flows offer a methodical approach to transform a simple distribution (like a Gaussian) into a complex one, matching the distribution of the target data. This is achieved by applying a chain of reversible and bijective transformations for distribution learning (Dinh et al., 2014). Given a flow $f = f_T \circ \cdots \circ f_1$, we initialize from a target distribution $\boldsymbol{z}_0 \sim p_{\boldsymbol{z}}$. The flow then undergoes a series of transformations to reach a Gaussian distribution $\boldsymbol{z}_T \sim \mathrm{N}(\mu, \sigma^2)$ through invertible functions: $\boldsymbol{z}_i = f_i(\boldsymbol{z}_{i-1}), i = 1, 2, \ldots, T$. The goal of normalizing flows is to minimize the difference between the learned distribution and the target distribution. This is typically quantified using the negative log-likelihood of the data. The flow learns the

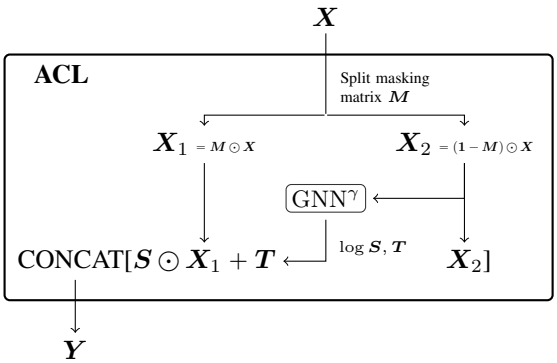


Figure 2: The affine coupling layer. The coupling is defined through a GNN and depends on the nature ($\gamma \in \{\text{hom.}, \text{cen.}, \text{het.}\}$) of the flow (see Sec. 3.2).

target distribution by minimizing the negative log-likelihood:

$$\mathcal{L} = -\log p_{\boldsymbol{z}}(\boldsymbol{z}_0) = -\log \mathrm{N}(\boldsymbol{z}_T \mid \mu, \sigma^2) - \log\det|\partial f/\partial \boldsymbol{z}_0|\,. \tag{1}$$

The power of normalizing flows lies in their bijectiveness. Each transformation is both reversible and maintains the 'volume' of the data distribution. This ensures that no information from the data is lost during these transformations. Thus, the transformed distribution can be 'pulled back' to the original space using the inverse of the transformation functions, providing a bridge between the simple Gaussian and the intricate target distribution. For this to work, the flow needs to be reversible, which we get back to in Sec. 3.5.

Affine coupling layers (ACLs) introduce reversible transformations to normalizing flows, ensuring efficient computation of the log-determinant of the Jacobian (Kingma & Dhariwal, 2018). Typically, the affine coupling layer, denoted by $\mathrm{ACL}^{(f,\boldsymbol{M})}$, contains a binary masking matrix $\boldsymbol{M} \in \{0,1\}^{m\times n}$ and **coupling function** f which determines the affine transformation parameters . Given an input $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, the input is split into $\boldsymbol{X}_1 = \boldsymbol{M} \odot \boldsymbol{X}$ and $\boldsymbol{X}_2 = (\boldsymbol{1} - \boldsymbol{M}) \odot \boldsymbol{X}$ by masking, where '$\odot$' denotes the element-wise Hadamard product. Here, $\boldsymbol{X}_1$ is the masked input that will undergo the transformation, and $\boldsymbol{X}_2$ is the part that provides parameters for this transformation via the coupling function and keeps invariant insdide the ACLs. The output is the concatenation of the transformed part and the fixed part as visualized in Fig. 2:

$$\mathrm{ACL}^{(f,\boldsymbol{M})}(\boldsymbol{X}) = \boldsymbol{M} \odot (\boldsymbol{S} \odot \boldsymbol{X}_1 + \boldsymbol{T}) + (\boldsymbol{1} - \boldsymbol{M}) \odot \boldsymbol{X}_2 \quad \text{such that} \quad \log \boldsymbol{S}, \boldsymbol{T} = f(\boldsymbol{X}_2). \tag{2}$$

The binary masking eensures that only part of the input is transformed, allowing the model to retain certain features while altering others, enabling the flow to capture intricate data distribution characteristics. This is key for enabling heterophily in the next sections.

3.2 Heterophilious Message Passing

Graph Neural Networks (GNNs) have emerged as a potent paradigm for learning from graph-structured data, where the challenges include diverse graph sizes and varying structures (Kipf & Welling, 2017; Veličković et al., 2018; Xu et al., 2019; Garg et al., 2020). Consider a graph $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. For these nodes and edges, we denote the corresponding node features as $\boldsymbol{X} = \{\boldsymbol{x}_v \in \mathbb{R}^{n_v} \mid v \in \mathcal{V}\}$ and edge features as $\boldsymbol{E} = \{\boldsymbol{e}_{uv} \in \mathbb{R}^{n_e} \mid u, v \in \mathcal{E}\}$. For each node $v \in \mathcal{V}$, its embedding at the k^{th} layer is represented as $\boldsymbol{h}_v^{(k)}$. These embeddings evolve through a sequence of transformations across K-deep GNN, by the message passing scheme (Hamilton, 2020):

$$\boldsymbol{m}_{uv}^{(k)} = \mathrm{MESSAGE}_{uv}^{(k)}\left(\boldsymbol{h}_u^{(k)}, e_{uv}\right), \qquad u \in \mathcal{N}(v), \quad k = 0, 1, \ldots, K, \tag{3}$$

$$\boldsymbol{h}_v^{(k+1)} = \mathrm{UPDATE}^{(k)}\left(\boldsymbol{h}_v^{(k)}, \boldsymbol{m}_{\mathcal{N}(v)}^{(k)}\right), \qquad k = 0, 1, \ldots, K. \tag{4}$$

Here $\mathcal{N}(v)$ denotes the neighbours set of node v. Both $\mathrm{UPDATE}^{(k)}$ and $\mathrm{MESSAGE}_{uv}^{(k)}$ are arbitrary differentiable functions. The set $\boldsymbol{m}_{\mathcal{N}(v)}^{(k)} = \{\boldsymbol{m}_{uv}^{(k)} \mid u \in \mathcal{N}(v)\}$ aggregates messages from all neighbours of v. Importantly, the function $\mathrm{UPDATE}^{(k)}$ needs to be permutation invariant on this message set $\boldsymbol{m}_{\mathcal{N}(v)}^{(k)}$ (*e.g.*, by operations like summation or taking the maximum). However, a naïve aggregation strategy will mix different messages and leads to the 'oversmoothing' problem.

Heterophilious GNNs Our method HTFlows encodes the heterophily assumption into the message passing sheme of the GNN. We denote the GNN^{γ} with heterophilious message passing scheme with an indicator $\gamma \in \{\text{cen.}, \text{hom.}, \text{het.}\}$ depending on the scheme being employed. These indicators specify the preference of the GNNs: whether they lean towards homophily (hom.), centrality (cen.), or heterophily (het.).

Referring to Eq. (4), the messages undergo a preprocessing step before being sent forward to the subsequent layer. This is given by:

$$\boldsymbol{m}_{\mathcal{N}(v)}^{(k)} = \{\alpha_{uv}^{(k)} \boldsymbol{m}_{uv}^{(k)} \mid u \in \mathcal{N}(v)\}, \tag{5}$$

where

$$\alpha_{uv}^{(k)} = \begin{cases} \mathcal{H}(u,v), & \text{if } \gamma = \text{hom.} \\ 1, & \text{if } \gamma = \text{cen.} \\ 1 - \mathcal{H}(u,v), & \text{if } \gamma = \text{het.} \end{cases} \tag{6}$$

where $\mathcal{H}$ denotes the homophily of the node (Pei et al., 2019). Yet, instead of traditional labels, in this context, the model aims to learn embeddings. Thus, in practice, we define the homophily or *attraction to similarity* between embeddings as the cosine similarity $\mathcal{H}(u, v) \triangleq S_{\cos}(\boldsymbol{h}_u^{(k)}, \boldsymbol{h}_v^{(k)})$ at the relevant layer.

3.3 Heterophilous training process

Given a molecule represented as a graph $G = (\boldsymbol{X}, \boldsymbol{B})$, the atom features are denoted by $\boldsymbol{X} \in \mathbb{R}^{n \times n_a}$ and the bond features by $\boldsymbol{B} \in \mathbb{R}^{n \times n \times n_b}$. The terms n_a and n_b represent the number of atom types and bond types, respectively. Specifically, $(\boldsymbol{X})_i$ denotes the one-hot encoded type of the i^{th} atom present in molecule G. Similarly, $(\boldsymbol{B})_{ij}$ denotes the one-hot encoding of the specific chemical bond between the i^{th} and j^{th} atom in G. Our model HTFlows maps the molecule G to embeddings $\boldsymbol{h}^{(a)}$ and $\boldsymbol{h}^{(b)}$ from the Gaussian distributions:

$$\boldsymbol{h}^{(a)} \sim p_a = \mathrm{N}(\mu_a, \sigma_a^2), \quad \boldsymbol{h}^{(b)} \sim p_b = \mathrm{N}(\mu_b, \sigma_b^2). \tag{7}$$

Bond flow The bond flow represented by $f_b = \mathrm{ACL}_{k_b}^b \circ \cdots \circ \mathrm{ACL}_1^b$ consists of a series of affine coupling layers with simple convolutional networks (CNNs) as coupling function: $\mathrm{ACL}_i^b = \mathrm{ACL}^{(\mathrm{CNN}_i, \boldsymbol{M}_i^b)}$, $i = 1, 2, \ldots, k_b$, where k_b denotes the number of layers and masking $(\boldsymbol{M}_i^b)_{jk} = \mathbf{1}_{\lceil 2k/n_b \rceil \equiv i(2)}$. Then bond embeddings $\boldsymbol{h}^{(b)} = \boldsymbol{B}_{k_b} = f_b(\boldsymbol{B}_0)$ emerge from the bond tensor $\boldsymbol{B}_0 = \boldsymbol{B}$:

$$\boldsymbol{B}_i = \mathrm{ACL}_i^b\left(\boldsymbol{B}_{i-1}\right), \quad i = 1, 2, \ldots, k_b. \tag{8}$$

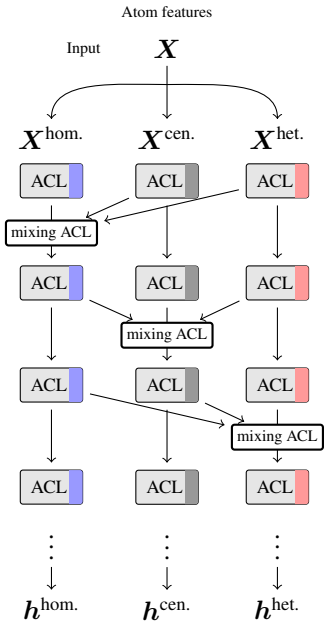


Figure 3: Heterophilous atom flow structure of HTFlows. The color of the ACL block refers to the indicators of GNN coupling functions: hom., cen., het.

Heterophilous atom flow The atom flow f_a contains three dependent normalizing flows of depth k_a. They are the central, homophilic, and heterophilic flows, associated with specific indicators labelled as $\Gamma = \{\text{cen.}, \text{hom.}, \text{het.}\}$. The corresponding affine coupling layers are built with heterophilious GNNs defined in Sec. 3.2 as coupling functions and masking $\boldsymbol{M}_i \in \{0, 1\}^{n \times n \times n_b}$

$$\mathrm{ACL}_{i,\gamma}^a = \mathrm{ACL}^{(\mathrm{GNN}_i^\gamma, \boldsymbol{M}_i)}, \quad i = 1, 2, \ldots, k_a, \quad \gamma \in \Gamma. \tag{9}$$

where $(\boldsymbol{M}_i)_{j,k,l} = \mathbf{1}_{j \equiv i(n_a)}$. All GNNs in this context derive their graph topology $(\mathcal{E}, \boldsymbol{E})$ from the bond tensor $\boldsymbol{B}$. The embeddings are initialized by the atom features: $\boldsymbol{X}_0^\gamma = \boldsymbol{X}, \gamma \in \Gamma$. With each layer, the embeddings undergo an update through the coupling layers:

$$\bar{\boldsymbol{X}}_i^\gamma = \mathrm{ACL}_{i,\gamma}^a\left(\boldsymbol{X}_{i-1}^\gamma \mid \boldsymbol{B}\right), \quad i = 1, 2, \ldots, k_a, \quad \gamma \in \Gamma. \tag{10}$$

Instead of constructing three separate flows, another sequence of 'mixing' affine coupling layers is introduced: $\mathrm{ACL}_i^{\text{mix.}} = \mathrm{ACL}^{(\mathrm{MLP}_i, \boldsymbol{M}_i^{\text{mix.}})}$ with MLP coupling functions. These layers serve the purpose of facilitating interactions between flows. By modulating the mask matrix $\boldsymbol{M}_i^{\text{mix.}} \in \{0, 1\}^{n \times 3n_a}$, the three flows engage in iterative interactions:

$$\boldsymbol{h}^{(a)} = \boldsymbol{h}_{k_a}^{(a)} = f_a(\boldsymbol{h}_0^{(a)} \mid \boldsymbol{B}), \quad \boldsymbol{h}_i^{(a)} = \mathrm{ACL}_i^{\text{mix.}}\left(\bar{\boldsymbol{h}}_i^{(a)}\right), \quad i = 1, 2, \ldots, k_a, \tag{11}$$

where the embeddings are concatenated from the three flows as $\boldsymbol{h}_i^{(a)} = \mathrm{concat}\left[\boldsymbol{X}_i^{\text{cen.}}, \boldsymbol{X}_i^{\text{hom.}}, \boldsymbol{X}_i^{\text{het.}}\right]$ and $\bar{\boldsymbol{h}}_i^{(a)} = \mathrm{concat}\left[\bar{\boldsymbol{X}}_i^{\text{cen.}}, \bar{\boldsymbol{X}}_i^{\text{hom.}}, \bar{\boldsymbol{X}}_i^{\text{het.}}\right]$, and the mask matrix $(\boldsymbol{M}_i^{\text{mix.}})_{jk} = \mathbf{1}_{\lceil k/n_a \rceil \equiv i(3)}$. A visual representation of the entire structure of the HTFlows model can be found in Fig. 3. For better undersatnding, we provide example reconstructions from intermediate layers in Fig. 4.

Loss The loss function combines the negative log-likelihoods (NLLs) from both the atom and bond flows: $\mathcal{L} = \mathcal{L}_a + \mathcal{L}_b$. Each NLL could be decomposed as shown in Eq. (1):

$$\mathcal{L}_b = -\log p_b\left(\boldsymbol{h}^{(b)}\right) - \log\det\left(\left|\frac{\partial \boldsymbol{h}^{(b)}}{\partial \boldsymbol{B}}\right|\right) = -\log p\left(\boldsymbol{h}^{(b)}\right) - \sum_{i=1}^{k_b} \log\det\left(\left|\frac{\partial \mathrm{ACL}_i^b\left(\boldsymbol{B}_{i-1}\right)}{\partial \boldsymbol{B}_{i-1}}\right|\right). \tag{12}$$

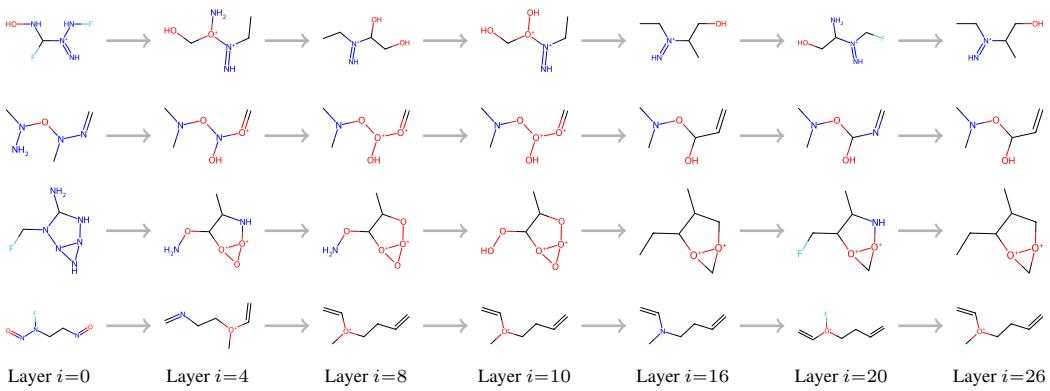


Figure 4: **Step-by-step generation (QM9).** Snapshots of reconstructed molecules when fixing the bond model and collecting node embeddings of the intermediate layers i.

Similarly, the loss $\mathcal{L}_a$ for the heterophilous atom flow can be constructed as:

$$\begin{aligned}\mathcal{L}_a &= -\log p\left(\boldsymbol{h}_{k_a}^{(a)}\right) - \log\det\left(\left|\frac{\partial \boldsymbol{h}^{(a)}}{\partial \boldsymbol{X}}\right|\right) \\ &= -\log p\left(\boldsymbol{h}_{k_a}^{(a)}\right) - \sum_{i=1}^{k_a}\left[\log\det\left(\left|\frac{\partial \mathrm{ACL}_i^{\text{mix.}}\left(\bar{\boldsymbol{h}}_i^{(a)}\right)}{\partial \bar{\boldsymbol{h}}_i^{(a)}}\right|\right) - \sum_{\gamma\in\Gamma}\log\det\left(\left|\frac{\partial \mathrm{ACL}_{i,\gamma}^{a}\left(\boldsymbol{X}_{i-1}^{\gamma}\right)}{\partial \boldsymbol{X}_{i-1}^{\gamma}}\right|\right)\right].\end{aligned} \tag{13}$$

3.4 Generation Process

Given a trained HTFlows model, with established atom flow f_{a*} and bond flow f_{b*}, the procedure for generating molecules is described as follows.

1. **Sampling Embeddings:** Start by randomly sampling embeddings $\boldsymbol{h}^{(a)}$ and $\boldsymbol{h}^{(b)}$ from a Gaussian distribution as expressed in Eq. (7).
2. **Obtaining the Bond Tensor:** The bond tensor $\boldsymbol{B}$ can be derived by applying the inverse of the bond flow f_{b*} to the sampled embedding $\boldsymbol{h}^{(b)}$. This is given as
$$\boldsymbol{B} = f_{b*}^{-1}(\boldsymbol{h}^{(b)}) = \left(\mathrm{ACL}_{1*}^{b}\right)^{-1} \circ \cdots \circ \left(\mathrm{ACL}_{k_b*}^{b}\right)^{-1}(\boldsymbol{h}^{(b)}). \tag{14}$$
3. **Recovering Graph Topology:** From the bond tensor $\boldsymbol{B}$, the graph topology $(\mathcal{E}, \boldsymbol{E})$ can be deduced. This topology is essential for the GNN-based affine coupling layers (ACLs) within the atom flow f_a.
4. **Generating Node Features:** With the bond tensor in place, node features can be produced by applying the inverse of the atom flow f_{a*} to the sampled atom embedding $\boldsymbol{h}^{(a)}$. This is given as
$$\boldsymbol{X} = f_{a*}^{-1}(\boldsymbol{h}^{(a)} \mid \boldsymbol{B}). \tag{15}$$
5. **Molecule Recovery:** Finally, a molecule, represented as G, can be reconstructed using the generated atom features $\boldsymbol{X}$ and bond tensor $\boldsymbol{B}$ from random embeddings $[\boldsymbol{h}^{(a)}, \boldsymbol{h}^{(b)}]$.

3.5 Reversibility of the Heterophilous Triple Flows

To ensure that the molecular embeddings and transformations produced by HTFlows can be inverted back, it is crucial to understand the reversibility of the processes. Both the atom and bond models of HTFlows rely on ACL blocks. As introduced in Sec. 3.1, these blocks are inherently reversible. This means they can forward process the input to produce an output and can also take that output to revert it back to the original input without loss of information. Besides the use of ACL blocks,

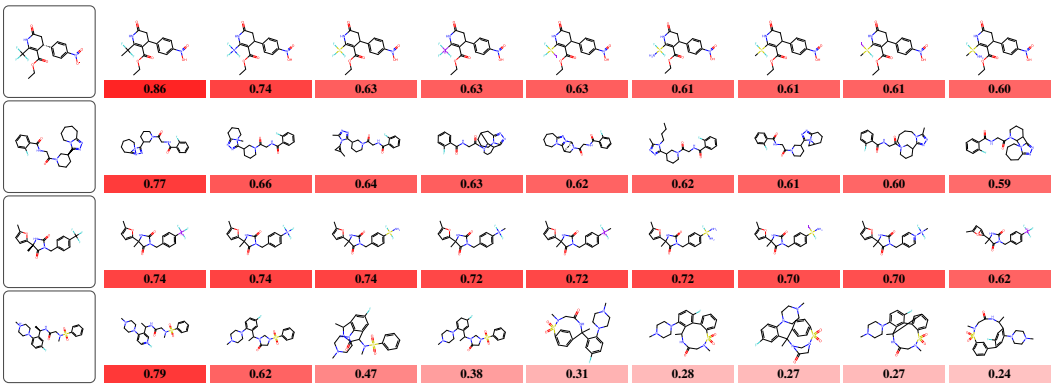


Figure 5: **Structured latent-space exploration (ZINC-250K).** Nearest neighbour search in the latent space with the seed molecules on the left and neigbours with the Tanimoto similarity (1 0) given for each molecule. For results on QM9, see Fig. A8 in the appendix.

the operations used within the model primarily leverage simple concatenation or splitting. These operations are straightforward and do not affect the overall reversibility of the processes. Given that the individual components (both atom and bond flows) are reversible and the operations performed on the data are straightforward, it is apparent that HTFlows as a whole is reversible. A formal proof on reversibility of ACL blocks and HTFlows is provided in App. B.

4 EXPERIMENTS

We demonstrate our model in a variety of common benchmarks tasks for molecule generation and modelling. First, provide an illustrative example of latent space exploration around seed molecules. Second, we provide results for molecule generation with benchmarks on a wide range of chemoinformatics metrics. Finally, we provide results for molecular property optimization.

Implementation The models were implemented in PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). In HTFlows, we used GNNs with 4 layers and flows that were $k_a = 27$(QM9) and $k_a = 38$(ZINC-250K) and $k_b = 10$ deep. We trained our models with the AdamW optimizer (Loshchilov & Hutter, 2019) for 500 epochs, with batch size 256 and learning rate 0.001. The final model selection was based on score comparison on a hold-out validation set. We select the best-performin model by the FCD score as suggested in Polykovskiy et al. (2020). Our models are trained on a cluster equipped with NVIDIA A100 GPUs. The training time for single models were 24 hours (QM9) and 56 hours (ZINC-250K).

Chemoinformatics metrics We compare methods through an extensive set of chemoinformatics metrics that perform both sanity checks (validity, uniqueness, and novelty) on the generated molecule corpus and quantify properties of the molecules: neighbour (SNN), fragment (Frag), and scaffold (scaf) similarity, internal diversity (IntDiv_1 and IntDiv_2), and Fréchet ChemNet distance (FCD). We also show score histograms for solubility (logP), syntetic accessibility (SA), drug-likeness (QED), and molecular weight. For computing the metrics, we use the MOSES benchamrking platform (Polykovskiy et al., 2020) and the RDKit open-source cheminformatics software (Landrum et al., 2013). The 'data' row in metrics is based on randomly sampled (1000 mols) for 10 times from data set. When we calculate the metrics we simulate 1000 molecules for 10 times and compare them to a hold-out reference set (20% of data, other 80% is used for training). Full details on the 14 metrics we use are included in App. C.

Data sets We consider two common molecule data sets: QM9 and ZINC-250K. The QM9 data set (Ramakrishnan et al., 2014) comprises $\sim$ 134k stable small organic molecules composed of atoms from the set {C, H, O, N, F}. These molecules have been processed into their kekulized forms with hydrogens removed using the RDkit software (Landrum et al., 2013). The ZINC-250K (Irwin et al., 2012) data contains $\sim$ 250k drug-like molecules, each with up to 38 atoms of 9 different types. Despite still relatively small (molecular weights ranging from 100 to 500), the molecules in the ZINC-250K data set are larger and more complicated than those in QM9.

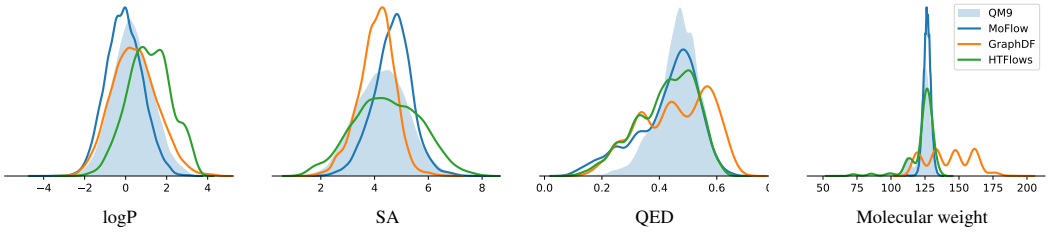


Figure 6: Chemoinformatics statistics for data (QM9) and generated molecules from HTFlows (ours), MoFlow, and GraphDF. We report histograms for the Octanol-water partition coefficient (logP), synthetic accessibility score (SA), quantitative estimation of drug-likeness (QED), and molecular weight.

Table 1: Chemoinformatics summary statistics for random generation on the QM9 data set. Full listing of all 14 metrics in Table A4. HTFlows performs well on all these summary metrics.

	FCD ↓	SNN ↑	Frag ↑	Scaf ↑	IntDiv_1 ↑	IntDiv_2 ↑
Data (QM9)	0.40	0.54	0.94	0.76	0.92	0.90
GraphDF	10.76	0.35	0.61	0.09	0.87	0.86
MoFlow	7.48	0.33	0.60	0.04	**0.92**	**0.90**
HTFlows	**5.63**	**0.36**	**0.71**	**0.23**	**0.92**	**0.90**

Visualizing the continuous latent space Similar to Zang & Wang (2020), we examine the learned latent space of our method on both QM9 and ZINC-250K. Results for ZINC-250K are presented in Fig. 5 and QM9 in the appendix (Fig. A8). Qualitatively, we note that latent space appears smooth and the molecules near the seed molecule resemble the input and have high Tanimoto similarity (Rogers & Hahn, 2010).

4.1 Molecule Generation

Baselines For random generation, we include baseline results for models that have pre-trained models available. We need access to the trained models, because few papers report chemoinformatics metrics beyond trivial sanity checks (validity, uniqueness, and novelty) that tend to be high (90%–100%) for most models. We compare to GraphDF (Luo et al., 2021) and MoFlow (Zang & Wang, 2020) which are current state-of-the-art (see Verma et al., 2022).

Results on QM9 For the QM9 data set, the main chemoinformatic summary statistics are given in Table 1 and the descriptive distributions in Fig. 6. The full listing of all 14 metrics is provided in Table A4 in the appendix. From Table 1, HTFlows achieves the lowest FCD, and achives highest (or on-par values with MoFlow) on SNN, Frag, and diversity. From the extended results in Table A4, it is clear that each model has its strengths, and the choice might depend on the specific requirements of a task. If one is looking for a model that produces a broad range of diverse molecules, HTFlows stands out as preferable.

Results on ZINC-250K For the ZINC-250K data set, the main chemoinformatic summary statistics are given in Table 2 and the descriptive distributions in Fig. 7. The full listing of all 14 metrics are provided in Table A5 in the appendix. While its performance on the ZINC-250K data set exhibits some variation, HTFlows still achieves the best internal diversity (IntDiv_1 and IntDiv_2) and has the most favorable molecular weight distribution. Notably, its validity score is lower than MoFlow, indicating some challenges in generating completely valid molecules in this context. Overall, HTFlow emerges as a robust and versatile molecular generation model, adept at balancing fidelity, diversity, and molecular properties.

4.2 Property Optimizaiton

In the property optimization task, models show their capability in finding novel molecules that optimize specific chemical properties not present in the training data set: a critical component for drug discovery. For our study, we focused on maximizing the QED property. We trained HTFlows on

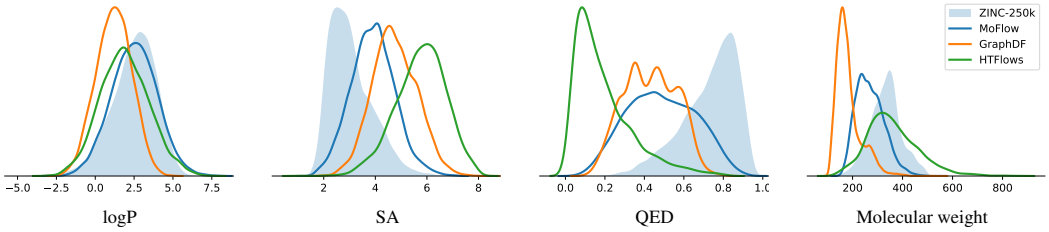


Figure 7: Chemoinformatics statistics for data (ZINC-250K) and generated molecules from HTFlows (ours), MoFlow, and GraphDF. Histograms for the Octanol-water partition coefficient (logP), synthetic accessibility score (SA), quantitative estimation of drug-likeness (QED), and molecular weight.

Table 2: Chemoinformatics summary statistics for random generation on the ZINC-250K data set. Full listing of all 14 metrics in Table A5. HTFlows performs well especially on diversity metrics.

	FCD ↓	SNN ↑	Frag ↑	Scaf ↑	IntDiv_1 ↑	IntDiv_2 ↑
Data (ZINC-250K)	1.44	0.51	1.00	0.28	0.87	0.86
GraphDF	34.30	0.23	0.35	0.00	0.88	0.87
MoFlow	**22.65**	**0.29**	**0.81**	**0.01**	0.88	0.86
HTFlows	27.90	0.22	0.57	0.00	**0.90**	**0.88**

ZINC-250K and evaluated its performance against other state-of-the-art models (Verma et al., 2022; Luo et al., 2021; Zang & Wang, 2020; Shi et al., 2019; Jin et al., 2018; You et al., 2018). The results, given in Table 3, show that the top three novel molecule candidates identified by HTFlows (that are not part of the ZINC-250K data set), exhibit QED values on par with those from ZINC-250K or other state-of-the-art methods. For details of the property optimization strategy and the top three molecules, see App. D.2.

5 Discussion and Conclusions

We have presented HTFlows, a novel approach to molecular generation by emphasizing heterophily patterns, countering the traditional oversmoothing vulnerability seen in existing graph neural network methodologies. By leveraging multiple interactive flows to discern (dis-)similarities between molecular entities, our method offers a more versatile representation of the intricate balance between molecular affinities and repulsions. The experiment results show HTFlows' ability to consistently generate molecules with high fidelity, diversity, and desired properties, marking it as a promising tool in the field of chemoinformatics and molecular design.

Table 3: Performance on molecule property optimization in terms of the best QED scores, scores taken from the corresponding papers (JTVAE score from Luo et al., 2021; Verma et al., 2022).

Method	1st	2nd	3rd
Dat(ZINC-250K)	0.948	0.948	0.948
JTVAE	0.925	0.911	0.910
GCPN	**0.948**	0.947	0.946
GraphAF	**0.948**	**0.948**	0.947
GraphDF	**0.948**	**0.948**	**0.948**
MoFlow	**0.948**	**0.948**	**0.948**
ModFlow	**0.948**	**0.948**	0.945
HTFlows	**0.948**	**0.948**	**0.948**

Based on the experiment results, it is noteworthy to draw parallels and distinctions between our model and MoFlow (Zang & Wang, 2020). While there are overarching similarities, our approach introduces several enhancements. Foremost, our atom model incorporates a heterophilous message-passing scheme within the coupling layers of the GNNs, and employs multiple interactive flows for dynamic information exchange. MoFlow's implementation uses an additional dimension to represent non-existent nodes, which, in practice, reduces the GNNs to MLPs. Furthermore, the masking matrix in MoFlow's ACL layers filters information predicated on node order in each graph, inadvertently making the model susceptible to isomorphic transformations. In contrast, our HTFlows model allows flexible-sized input graphs, avoids message exchange from the non-existed nodes, and is permutation-invariant to isomorphism.

Reproducibility statement The code and trained models will be made available under the MIT License on GitHub upon acceptance.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 17981–17993. Curran Associates, Inc., 2021.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations (ICLR)*, 2021.

Valerio Ciotti, Moreno Bonaventura, Vincenzo Nicosia, Pietro Panzarasa, and Vito Latora. Homophily and missing links in citation networks. *EPJ Data Science*, 5:1–14, 2016.

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

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

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

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 3419–3430. PMLR, 2020.

Elisabeth R Gerber, Adam Douglas Henry, and Mark Lubell. Political homophily and collaboration in regional planning networks. *American Journal of Political Science*, 57(3):598–610, 2013.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pp. 2672–2680. Curran Associates, Inc., 2014.

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

William L Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, 2020.

Emiel Hoogeboom and Tim Salimans. Blurring diffusion models. In *International Conference on Learning Representations (ICLR)*, 2023.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning (ICML)*, pp. 8867–8887. PMLR, 2022.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 15820–15831. Curran Associates, Inc., 2019.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7): 1757–1768, 2012.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning (ICML)*, pp. 2323–2332. PMLR, 2018.

Madhawa Kaushalya, Ishiguro Katushiko, Nakago Kosuke, and Abe Motoki. GraphNVP: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Durk P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 10236–10245. Curran Associates, Inc., 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning (ICML)*, pp. 1945–1954. PMLR, 2017.

Greg Landrum et al. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 8:31, 2013.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 20887–20902. Curran Associates, Inc., 2021.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10270–10276, 2021a.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. GraphEBM: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 1362–1375. Curran Associates, Inc., 2022.

Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification: Investigating the homophily principle on node distinguishability. *arXiv preprint arXiv:2304.14274*, 2023.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning (ICML)*, pp. 7192–7203. PMLR, 2021.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

Haitao Mao, Zhikai Chen, Wei Jin, Haoyu Han, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *arXiv preprint arXiv:2306.01323*, 2023.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 8026–8037. Curran Associates, Inc., 2019.

Steven M Paul, Daniel S Mytelka, Christopher T Dunwiddie, Charles C Persinger, Bernard H Munos, Stacy R Lindborg, and Aaron L Schacht. How to improve r&d productivity: The pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery*, 9(3):203–214, 2010.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of Computer-aided Molecular Design*, 27:675–679, 2013.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:565644, 2020.

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

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7, 2014.

Jean-Louis Reymond, Lars Ruddigkeit, Lorenz Blum, and Ruud Van Deursen. The enumeration of chemical space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):717–733, 2012.

Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. In *International Conference on Learning Representations (ICLR)*, 2023.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 9323–9332. PMLR, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations (ICLR)*, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

Yogesh Verma, Samuel Kaski, Markus Heinonen, and Vikas Garg. Modular flows: Differential molecular generation. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 12409–12421. Curran Associates, Inc., 2022.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: Allen-Cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

David Weininger, Arthur Weininger, and Joseph L Weininger. SMILES. 2. Algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 29(2): 97–101, 1989.

Lirong Wu, Haitao Lin, Bozhen Hu, Cheng Tan, Zhangyang Gao, Zicheng Liu, and Stan Z Li. Beyond homophily and homogeneity assumption: Relation-based frequency adaptive graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *IEEE International Conference on Data Mining (ICDM)*, pp. 1287–1292. IEEE, 2022.

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 (NeurIPS)*, pp. 6410–6421, 2018.

Chengxi Zang and Fei Wang. 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.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 7793–7804. Curran Associates, Inc., 2020.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11168–11176, 2021.

APPENDIX

This appendix is organized as follows. App. A presents details on heterophilious message passing in our model and compuational issues. App. B provides a formal proof to show that the proposed triple flow model is reversible. App. C summarizes and describes the metrics used in the experiments. App. D provides additional experiment results for molecule generation and the algorithm for property optimization together with detals on found candidate molecules.

A ALGORITHM DETAILS

A.1 THE BEHAVIOR OF HETEROPHILIOUS GNN CONTROLLED BY γ

In the code implementation, we choose a single-layer Graph Attention Network (GAT) as the base, but change the scaling of the message collected from neighbours based on the homophily during message passing.

Given node embeddings $\boldsymbol{h}_u, \boldsymbol{h}_v$ of node u, v, there is the attention factor $\beta_{u,v}$ calculated based on a softmax of the attributes of nodes and edge feature $\boldsymbol{e}_{uv}$:

$$\beta_{u,v} = \frac{\exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^\top[\boldsymbol{\Theta}\boldsymbol{h}_u \,\|\, \boldsymbol{\Theta}\boldsymbol{h}_v \,\|\, \boldsymbol{\Theta}_e \mathbf{e}_{i,j}]\right)\right)}{\sum_{w\in\mathcal{N}(u)\cup\{u\}} \exp\left(\text{LeakyReLU}\left(\boldsymbol{a}^\top[\boldsymbol{\Theta}\boldsymbol{h}_u \,\|\, \boldsymbol{\Theta}\boldsymbol{h}_w \,\|\, \boldsymbol{\Theta}_e \mathbf{e}_{v,w}]\right)\right)},$$

where $\theta = (\Theta, \Theta_e, \boldsymbol{a})$ are model parameters, Then the message collected from neighbours and updated to be

$$\boldsymbol{h}'_v = \alpha^\gamma_{v,v}\beta_{v,v}\boldsymbol{\Theta}\boldsymbol{h}_v + \sum_{u\in\mathcal{N}(v)} \alpha^\gamma_{u,v}\beta_{u,v}\boldsymbol{\Theta}\boldsymbol{h}_u,$$

where $\alpha^\gamma_{u,v}$ denotes the homophily factor, where

$$\alpha^\gamma_{u,v} = \begin{cases} 1, & \text{if } \gamma = \text{cen.} \\ \mathcal{H}(u,v), & \text{if } \gamma = \text{hom.} \\ 1-\mathcal{H}(u,v), & \text{if } \gamma = \text{het.}, \end{cases}$$

where $\mathcal{H}(u,v) \triangleq S_{\cos}(\boldsymbol{h}_u^{(k)}, \boldsymbol{h}_v^{(k)})$ is the cosine similarity.

In conclusion, given the input $\boldsymbol{X} = [\boldsymbol{x}_v]_{v\in V}$, edge attributes and edge index contained inside the edge tensor $\mathbf{E}$, the GNN gets the output $\boldsymbol{X}' = [\boldsymbol{x}'_v]_{v\in V}$

$$\text{GNN}^\gamma_\theta(\boldsymbol{X} \mid \mathbf{E}) = \boldsymbol{X}'.$$

A.2 COMPUTATIONAL CONSIDERATIONS

For the convenience on the calculation of the log-likelihood, every transformation of variables needs the calculation of a Jacobian matrix (*i.e.*, $\partial \boldsymbol{Z}^{(l+1)}/\partial \boldsymbol{Z}^{(l)}$). So all the complicated modules (*e.g.*, GNNs, MLPs) are all built inside the coupling structure (part of input is updated by the scaling matrix $\boldsymbol{S}$, and transformation matrix $\boldsymbol{T}$ depends on the other part of input).

B PROOF OF REVERSIBILITY

B.1 REVERSIBILITY OF THE ACL

Set up Assume an ACL defined in Sec. 3.1 contains coupling function f and masking matrix $\boldsymbol{M} \in \{0,1\}^{m\times n}$. Given input $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, the output $\boldsymbol{Y}$ is calculated as

$$\boldsymbol{Y} = \text{ACL}^{(f,\boldsymbol{M})}(\boldsymbol{X}) = \boldsymbol{M} \odot (\boldsymbol{S} \odot \boldsymbol{X}_1 + \boldsymbol{T}) + (\mathbf{1} - \boldsymbol{M})\boldsymbol{X}_2 \tag{16}$$

where $\log \boldsymbol{S}, \boldsymbol{T} = f(\boldsymbol{X}_2)$, and $\boldsymbol{X}_1, \boldsymbol{X}_2$ are the split from input by masking:

$$\boldsymbol{X}_1 = \boldsymbol{M} \odot \boldsymbol{X}, \quad \boldsymbol{X}_2 = (\mathbf{1} - \boldsymbol{M}) \odot \boldsymbol{X}.$$

We seek to recover $\boldsymbol{X}$ from the $f, \boldsymbol{M}$, and $\boldsymbol{Y}$.

Reversibility from output to input Since $\boldsymbol{M}$ is binary, we get

$$\begin{aligned}&\boldsymbol{M}\odot\boldsymbol{M}=\boldsymbol{M},\quad(\mathbf{1}-\boldsymbol{M})\odot(\mathbf{1}-\boldsymbol{M})=(\mathbf{1}-\boldsymbol{M}),\\&\boldsymbol{M}\odot(\mathbf{1}-\boldsymbol{M})=(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{M}=\mathbf{0}\end{aligned}$$

and

$$\boldsymbol{X}=(\boldsymbol{M}+(\mathbf{1}-\boldsymbol{M}))\odot\boldsymbol{X}=\boldsymbol{M}\odot\boldsymbol{X}+(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{X}=\boldsymbol{X}_1+\boldsymbol{X}_2.$$

By splitting the output $\boldsymbol{Y}$ to $\boldsymbol{Y}_1,\boldsymbol{Y}_2$ by masking matrix:

$$\boldsymbol{Y}_1=\boldsymbol{M}\odot\boldsymbol{Y},\quad\boldsymbol{Y}_2=(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{Y}.$$

Combining with Eq. (16), we know

$$\begin{aligned}\boldsymbol{Y}_1&=\boldsymbol{M}\odot\boldsymbol{Y}\\&=\boldsymbol{M}\odot(\boldsymbol{M}\odot(\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{T})+(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{X}_2)\\&=\boldsymbol{M}\odot(\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{T}),\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{Y}_2&=(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{Y}\\&=(\mathbf{1}-\boldsymbol{M})\odot(\boldsymbol{M}\odot(\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{T})+(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{X}_2)\\&=(\mathbf{1}-\boldsymbol{M})\odot(\boldsymbol{M}\odot(\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{T})+(\mathbf{1}-\boldsymbol{M})\odot(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{X})\\&=(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{X}=\boldsymbol{X}_2.\end{aligned}$$

Now the $\log\boldsymbol{S},\boldsymbol{T}=f(\boldsymbol{X}_2)=\boldsymbol{Y}_2$ are recovered by $\boldsymbol{Y}$. Notice that

$$\begin{aligned}\boldsymbol{M}\odot(\boldsymbol{Y}_1-\boldsymbol{T})\oslash\boldsymbol{S}&=\boldsymbol{M}\odot(\boldsymbol{M}\odot(\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{T})-\boldsymbol{T})\oslash\boldsymbol{S}\\&=(\boldsymbol{M}\odot\boldsymbol{S}\odot\boldsymbol{X}_1+\boldsymbol{M}\odot\boldsymbol{T}-\boldsymbol{M}\odot\boldsymbol{T})\oslash\boldsymbol{S}\\&=(\boldsymbol{M}\odot\boldsymbol{S}\odot\boldsymbol{X}_1)\oslash\boldsymbol{S}\\&=\boldsymbol{M}\odot\boldsymbol{X}_1\\&=\boldsymbol{M}\odot\boldsymbol{M}\odot\boldsymbol{X}\\&=\boldsymbol{M}\odot\boldsymbol{X}\\&=\boldsymbol{X}_1\quad\text{if}\quad(\boldsymbol{S})_{i,j}>0,\quad\forall i,j,\end{aligned}$$

where '$\oslash$' denotes element-wise division. Since we define $\boldsymbol{S}$ as the exponential of part of output from coupling function, the elements of $\boldsymbol{S}$ are all strictly positive. Then

$$\begin{aligned}\left(\mathrm{ACL}^{(f,\boldsymbol{M})}\right)^{-1}(\boldsymbol{Y})=\boldsymbol{X}&=\boldsymbol{X}_1+\boldsymbol{X}_2\\&=\boldsymbol{M}\odot(\boldsymbol{Y}_1-\boldsymbol{T})\oslash\boldsymbol{S}+\boldsymbol{Y}_2\\&=\boldsymbol{M}\odot(\boldsymbol{M}\odot\boldsymbol{Y}-\boldsymbol{T})\oslash\boldsymbol{S}+(\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{Y}.\end{aligned}\tag{17}$$

where $\log\boldsymbol{S},\boldsymbol{T}=f(\boldsymbol{X}_2)=f((\mathbf{1}-\boldsymbol{M})\odot\boldsymbol{Y}))$. Eq. (17) shows how the input is recovered from output, thus the ACL block is reversible.

B.2 Reversibility of the bond model

For the bond model $f_b=\mathrm{ACL}_{k_b}^b\circ\cdots\circ\mathrm{ACL}_1^b$, and since each $\mathrm{ACL}_i^b,\ i=1,\ldots,k_b$ is reversible, we can write $f_b^{-1}=\left(\mathrm{ACL}_1^b\right)^{-1}\circ\cdots\circ\left(\mathrm{ACL}_{k_b}^b\right)^{-1}$, which the reverse function of f_b.

B.3 Reversibility of the atom model

For atom model f_a, which includes all $\{\mathrm{ACL}_i^{\text{mix.}},\mathrm{ACL}_{i,\gamma}^a|i=1,\ldots,k_a,\gamma\in\Gamma\}$, we prove that each layer of f_a which maps $\boldsymbol{h}_{i-1}^{(a)}$ to $\boldsymbol{h}_i^{(a)}$ is reversible.

For $i\in\{1,\ldots,k_a\}$, given $\boldsymbol{h}_i^{(a)}=\mathrm{ACL}_i^{(\text{mix.})}(\bar{\boldsymbol{h}}_i^{(a)})$, the $\bar{\boldsymbol{h}}_i^{(a)}$ could be recovered by reversible $ACL_i^{(\text{mix.})}$. Since

$$\bar{\boldsymbol{h}}_i^{(a)}=\mathrm{concat}\left[\bar{\boldsymbol{X}}_i^{\text{cen.}},\bar{\boldsymbol{X}}_i^{\text{hom.}},\bar{\boldsymbol{X}}_i^{\text{het.}}\right]\tag{18}$$

$$=\mathrm{concat}\left[\mathrm{ACL}_{i,\text{cen.}}(\boldsymbol{X}_{i-1}^{\text{cen.}}),\mathrm{ACL}_{i,\text{hom.}}(\boldsymbol{X}_{i-1}^{\text{hom.}}),\mathrm{ACL}_{i,\text{het.}}(\boldsymbol{X}_{i-1}^{\text{het.}})\right],\tag{19}$$

then the $\boldsymbol{X}_{i-1}^{\text{cen.}},\boldsymbol{X}_{i-1}^{\text{hom.}},\boldsymbol{X}_{i-1}^{\text{het.}}$ can be recovered by reversible $\{ACL_{i,\gamma}^a\mid\gamma\in\Gamma\}$, thus $\boldsymbol{h}_{i-1}^{(a)}=\mathrm{concat}\left[\boldsymbol{X}_{i-1}^{\text{cen.}},\boldsymbol{X}_{i-1}^{\text{hom.}},\boldsymbol{X}_{i-1}^{\text{het.}}\right]$ is recovered.

C DESCRIPTION OF METRICS

For benchmarking, model selection, comparison, and explorative analysis, we use the following 14 metrics. The metrics are presented in detail in the work by Polykovskiy et al. (2020) that introduced the MOSES benchmarking platform. The metrics calculation makes heavy use of the RDKit open-source cheminformatics software (https://www.rdkit.org/). We briefly summarize the metrics below.

Sanity check metrics

1. **Validity** Fraction (in $[0, 1]$) of the molecules that produce valid SMILES representations. This is a sanity check for how well the model captures explicit chemical constraints such as proper valence. Higher values are better as a low value can indicate that the model does not capture properly chemical structure. We report numbers without *post hoc* validity corrections.
2. **Uniqueness** Fraction (in $[0, 1]$) of the molecules that are unique. This is a sanity check based on the SMILES string representation of the generated molecules. Higher values are better as a low value can indicate the model has collapsed and produces only a few typical molecules.
3. **Novelty** Fraction (in $[0, 1]$) of the generated molecules that are not present in the training set. Higher values are better as a low value can indicate overfitting to the training data set.

Summary statistics

4. **Similarity to a nearest neighbour (SNN)** The average Tanimoto similarity (Jaccard coefficient) in $[0, 1]$ between the generated molecules and their nearest neighbour in the reference data set. Higher is better: If the generated molecules are far from the reference set, similarity to the nearest neighbour will be low.
5. **Fragment similarity (Frag)** Measures similarity (in $[0, 1]$) of distributions of BRICS fragments (substructures) in the generated set vs. the original data set. If molecules in the two sets share many of the same fragments in similar proportions, the Frag metric will be close to 1 (higher better).
6. **Scaffold similarity (Scaf)** Measures similarity (in $[0, 1]$) of distributions of Bemis–Murcko scaffolds (molecule ring structures, linker fragments, and carbonyl groups) in the generated set vs. the original data set. This metric is calculated similarily as the Fragment similarity metric by counting substructure presence in the data, and they can be high even if the data sets do not contain the same molecules.
7. **Internal diversity (IntDiv_1)** Measure (in $[0, 1]$) of the chemical diversity within the generated set of molecules. Higher values are better and signal higher diversity in the generated set of moleculers. Low values can signal mode collapse.
8. **Internal diversity (IntDiv_2)** Measure (in $[0, 1]$) of the chemical diversity within the generated set of molecules. The interpretation is similar to IntDiv_1, but with stronger penalization of the Tanimoto similarity in calculating the diversity.
9. **Filters** This metric is specific to the MOSES benchmarking platrofm (see Polykovskiy et al., 2020). It gives the fraction (in $[0, 1]$) of generated molecules that passes filters applied during data set construction. In practice, these filters may filter out chemically valid molecules that have fragments that are not of interest in the MOSES data set (filtered with medicinal chemistry filters). Thus, this metric is not of primary interest for us, but gives a view on match with the MOSES data set.
10. **Fréchet ChemNet distance (FCD)** Analogous to the Frechét Inception Distance (FID) used in image generation, FCD compares feature distributions of real and generated molecules using a pre-trained model (ChemNet). Lower values are better.

Descriptive distributions

11. **Octanol-water partition coefficient (logP)** A logarithmic measure of the relationship between lipophilicity (fat solubility) and hydrophilicity (water solubility) of a set of molecules.

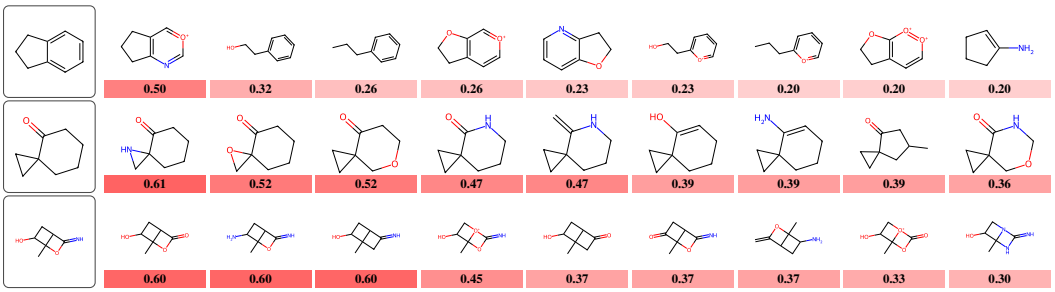


Figure A8: **Structured latent-space exploration (QM9).** Nearest neighbour search in the latent space with the seed molecule on the left and neigbours with the Tanimoto similarity (1 0) given for each molecule.

For large values a substance is more soluble in fat-like solvents such as n-octanol, and for small values more soluble in water. We report both histograms of logP and a summary statistic in terms of the Wasserstein distance between the generated and reference distributions (smaller better).

12. **Synthetic accessibility score (SA)** A metric that estimates how easily a chemical molecule can be synthesized. It provides a quantitative value indicating the relative difficulty or ease of synthesizing a molecule, with a lower SA score suggesting that a molecule is more easily synthesized, and a higher score suggesting greater complexity or difficulty. We report both histograms of SA and a summary statistic in terms of the Wasserstein distance between the generated and reference distributions (smaller better).
13. **Quantitative estimation of drug-likeness (QED)** A metric designed to provide a quantitative measure of how 'drug-like' a molecule is. It essentially refers to the likelihood that a molecule possesses properties consistent with most known drugs, estimated based on a variety of molecular descriptors. We report both histograms of QED and a summary statistic in terms of the Wasserstein distance between the generated and reference distributions (smaller better).
14. **Molecular weight (Weight)** The sum of atomic weights in a molecule. We report both histograms of molecular weights and a summary statistic in terms of the Wasserstein distance between the generated and reference distributions (smaller better).

D Experiment details

D.1 Further results

We provide further results for structured latent-space exploration (only ZINC-250K included in the main paper). Example explorations for QM9 are shown in Fig. A8.

We include full listings of all 14 metrics (description of metrics in App. C) considered in the random generation tasks for QM9 and ZINC-250K. The values are listed in Tables A4 and A5, respectively. Additionally, we also visualize the node homophily (in the 'neighbour-counting' sense as in Fig. 1) for both QM9 and ZINC-250K together with the estimated node homophily histograms (see Fig. A9) from the generation output from the different models. Even if our model, considers homophily/heterophily in learned embedding sense, the histograms show structure even for node homophily—though with an additional mode for strong heterophily, which shows for both HTFlows and MoFlow.

D.2 Property optimization

Algorithm Given a pretrained HTFlows f, and training set $\mathcal{D}$ contains molecule and property label pairs $\{G, y\}$. Now we introduce an extra simple MLP g_θ, trained on the dataset to be $g_{\theta*}$ by optimizing the parameters:

$$\theta^* = \arg\min_{\theta} \underset{(G,y)\in\mathcal{D}}{\text{MSEloss.}}(g_\theta(f(G)), y) \tag{20}$$

Table A4: Full benchmark metrics for random generation using QM9 (reporting mean±std).

	Validity ↑	Uniqueness ↑	Novelty ↑	SNN ↑	Frag ↑	Scaf ↑	IntDiv_1 ↑
Data (QM9)	1.00±0.00	1.00±0.00	0.62±0.02	0.54±0.00	0.94±0.01	0.76±0.03	0.92±0.00
GraphDF	-	**1.00**±0.00	0.98±0.00	0.35±0.00	0.61±0.01	0.09±0.07	0.87±0.00
MoFlow	**0.94**±0.01	**1.00**±0.00	**0.99**±0.00	0.33±0.00	0.60±0.03	0.04±0.03	**0.92**±0.00
HTFlows	0.83±0.01	**1.00**±0.00	0.95±0.01	**0.36**±0.01	**0.71**±0.04	**0.23**±0.05	**0.92**±0.00

	IntDiv_2 ↑	Filters ↑	FCD ↓	logP ↓	SA ↓	QED ↓	Weight ↓
Data (QM9)	0.90±0.00	0.64±0.02	0.40±0.02	0.04±0.01	0.03±0.01	0.00±0.00	0.32±0.08
GraphDF	0.86±0.00	**0.69**±0.02	10.76±0.21	**0.16**±0.03	**0.27**±0.02	0.05±0.00	19.72±0.54
MoFlow	**0.90**±0.00	0.55±0.02	7.48±0.23	0.38±0.02	0.41±0.02	**0.04**±0.00	3.74±0.09
HTFlows	**0.90**±0.00	0.39±0.02	**5.63**±0.15	0.42±0.06	0.49±0.04	0.07±0.00	**2.97**±0.31

Table A5: Full benchmark metrics for random generation using ZINC-250K (reporting mean±std).

	Validity ↑	Uniqueness ↑	Novelty ↑	SNN ↑	Frag ↑	Scaf ↑	IntDiv_1 ↑
Data (ZINC-250K)	1.00±0.00	1.00±0.00	0.02±0.00	0.51±0.00	1.00±0.00	0.28±0.02	0.87±0.00
GraphDF	-	**1.00**±0.00	**1.00**±0.00	0.23±0.00	0.35±0.01	0.00±0.00	0.88±0.00
MoFlow	**0.70**±0.01	**1.00**±0.00	**1.00**±0.00	**0.29**±0.00	**0.81**±0.01	**0.01**±0.00	0.88±0.00
HTFlows	0.46±0.02	**1.00**±0.00	**1.00**±0.00	0.22±0.00	0.57±0.03	0.00±0.00	**0.90**±0.00

	IntDiv_2 ↑	Filters ↑	FCD ↓	logP ↓	SA ↓	QED ↓	Weight ↓
Data (ZINC-250K)	0.86±0.00	0.59±0.01	1.44±0.01	0.05±0.01	0.03±0.01	0.01±0.00	2.18±0.39
GraphDF	0.87±0.00	**0.54**±0.01	34.30±0.30	1.28±0.03	1.70±0.03	0.30±0.00	149.27±1.55
MoFlow	0.86±0.00	0.53±0.02	**22.65**±0.40	**0.14**±0.03	**0.85**±0.04	**0.24**±0.01	61.83±3.00
HTFlows	**0.88**±0.00	0.22±0.02	27.90±0.23	0.96±0.07	2.07±0.05	0.44±0.01	**16.51**±2.85

Then we find molecule candidates $\{G_i\}_{i=1}^k$ with top-k properties in the data set $\mathcal{D}$ are chosen. New embeddings are explored by optimizing the predict label by $g_{\theta*}$ starting from these candidates:

$$\boldsymbol{h}_{i,j} = \delta \frac{\partial g_{\theta*}}{\partial \boldsymbol{h}}(\boldsymbol{h}_{i,j-1}) + \boldsymbol{h}_{i,j-1}, \quad j = 1, \ldots, N, \quad \boldsymbol{h}_{i,0} = f(G_i), \quad i = 1, \ldots, k,$$

where δ denotes the search step length, and N is the number of iterations. These embeddings could be recovered to be molecule set:

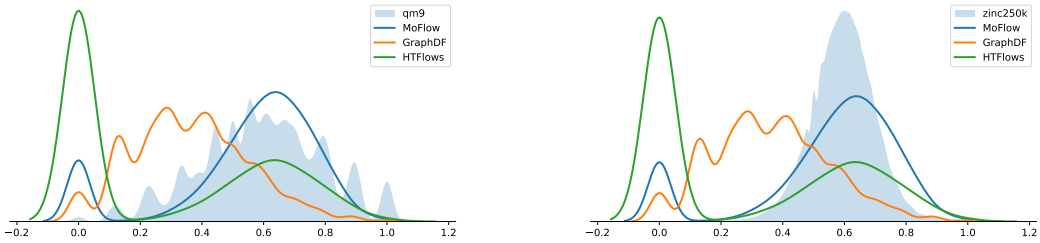
$$\mathcal{D}' = \{f^{-1}(\boldsymbol{h}_{ij})\}_{i=1,\ldots,k, \quad j=1,\ldots,N}.$$

Finally, $\mathcal{D}'\backslash\mathcal{D}$ gives the novel molecule sets with related high target properties.

Generation results In our experiments, the g_θ is a simple 3-layer MLP with 16 hidden nodes, the dataset $\mathcal{D}$ is ZINC-250K, and target property y is QED. And $\mathcal{D}'\backslash\mathcal{D}$ provides 17 molecules with QED score 0.948. The Top-3 QED score and molecular SMILES are listed below:

1. QED = 0.948442, `CC(C)N1N=CC2=NC(c3ccc(-c4ccccn4)cc3)NC21`
2. QED = 0.948190, `O=C(NCC1COc2ccccc2O1)c1ccccc1Cl`
3. QED = 0.948051, `Cc1ccc(C(CO)C2CS(=O)(=O)c3ccccc32)cc1`

Baselines The baselines scores of GCPN (You et al., 2018), GraphAF (Shi et al., 2019), GraphDF (Luo et al., 2021), MoFlow Zang & Wang (2020) and ModFlow (Verma et al., 2022) are acquired from the corresponding papers. The score of JTVAE (Jin et al., 2018) is acquired from Zang & Wang (2020); Verma et al. (2022).



QM9 ZINC-250K

Figure A9: Node homophily distribution of generated molecules.