arXiv:1905.10310v1 [cs.LG] 24 May 2019

Likelihood-Free Inference and Generation of Molecular Graphs

Sebastian Pölsterl **Christian Wachinger**
Artificial Intelligence in Medical Imaging (AI-Med)
Department of Child and Adolescent Psychiatry
Ludwig Maximilian Universität, Munich, Germany
{sebastian.poelsterl,christian.wachinger}@med.uni-muenchen.de

Abstract

Recent methods for generating novel molecules use graph representations of molecules and employ various forms of graph convolutional neural networks for inference. However, training requires solving an expensive graph isomorphism problem, which previous approaches do not address or solve only approximately. In this work, we propose LF-MolGAN, a likelihood-free approach for *de novo* molecule generation that avoids explicitly computing a reconstruction loss. Our approach extends generative adversarial networks by including an adversarial cycle-consistency loss to implicitly enforce the reconstruction property. To capture properties unique to molecules, such as valence, we extend Graph Isomorphism Network to multi-graphs. To quantify the performance of models, we propose to compute the distance between distributions of physicochemical properties with the 1-Wasserstein distance. We demonstrate that LF-MolGAN more accurately learns the distribution over the space of molecules than all baselines. Moreover, it can be utilized for drug discovery by efficiently searching the space of molecules using molecules' continuous latent representation.

1 Introduction

Deep generative models have been proven successful in generating high-quality samples in the domain of images, audio, and text, but it was only recently when models have been developed for *de novo* chemical design [1, 2]. The goal of *de novo* chemical design is to map desirable properties of molecules, such as a drug being active against a certain biological target, to the space of molecules. This process – called inverse Quantitative Structure-Activity Relationship (QSAR) – is extremely challenging due to the vast size of the chemical space. Polishchuk et al. [3] estimated the number of realistic drug-like molecules to be in the order of 10^{33}. Searching this space efficiently is often hindered by the discrete nature of molecules, which prevents the use of gradient-based optimization. Thus, obtaining a continuous and differentiable representation of molecules is a desirable goal that could ease drug discovery. For *de novo* generation of molecules, it is important to produce chemically valid molecules that comply with the valence of atoms, i.e., how many electron pairs an atom of a particular type can share. For instance, carbon has a valence of four and can form at most four single bonds. Therefore, any mapping from the continuous latent space of a model to the space of molecules should result in a chemically valid molecule.

The current state of the art deep learning models are adversarial or variational autoencoders (AAE, VAE) that represent molecules as graphs and rely on graph convolutional neural networks (GCNs) [4–12]. The main obstacle is in defining a suitable reconstruction loss, which is challenging when inputs and outputs are graphs. Because there is no canonical form of a graph's adjacency matrix, two graphs can be identical despite having different adjacency matrices. Before the reconstruction

Preprint. Under review.

loss can be computed, correspondences between nodes of the target and reconstructed graph need to be established, which requires solving a computationally expensive graph isomorphism problem. Existing graph-based VAEs have addressed this problem by either traversing nodes in a fixed order [5, 8, 10] or employing graph matching algorithms [11] to approximate the reconstruction loss.

We propose LF-MolGAN, a likelihood-free Generative Adversarial Network for inference and generation of molecular graphs (see fig. 1). This is the first time that an inference and generative model of molecular graphs can be trained without computing a reconstruction loss. Our model consists of an encoder (inference model) and a decoder (generator) that are trained by implicitly imposing the reconstructing property via cycle-consistency, thus, avoiding the need to solve a computationally prohibitive graph isomorphism problem. To learn from graph-structered data, we base our encoder on the recently proposed Graph Isomorphism Network [13], which we extend to multi-graphs, and employ the Gumbel-softmax trick [14, 15] to generate discrete molecular graphs. Finally, we explicitly incorporate domain knowledge such that generated graphs represent valid chemical structures. We will show that this enables us to perform efficient nearest neighbor search in the space of molecules.

In addition,we performed an extensive suite of benchmarks to accurately determine the strengths and weaknesses of models. We argue that summary statistics such as the percentage of valid, unique, and novel molecules used in previous studies, are poor proxies to determine whether models are generating chemically meaningful molecules. In our experiments, we instead compare the distributions of 10 chemical properties and demonstrate that our proposed method is able to more accurately learn a distribution over the space of molecules than previous approaches.

2 Related Work

Graphs, where nodes represent atoms, and edges chemical bonds, are a natural representation of molecules, which has been explored in [4–12]. Most methods rely on graph convolutional neural networks (GCNs) for inference, which can efficiently learn from the non-Euclidean structure of graphs [4, 7, 8, 10–12]. Molecular graphs can be generated sequentially, adding single atoms and bonds using an RNN-based architecture [5–8, 10, 12], or in a single step [4, 9, 11]. Sequential generation has the advantage that partially generated graphs can be checked, e.g., for valence violations [8, 10]. Molecules can also be represented as strings using SMILES encoding, for which previous work relied on recurrent neural networks for inference and generation [1, 2, 16–23]. However, producing valid SMILES strings is challenging, because models need to learn the underlying grammar of SMILES. Therefore, a considerable portion of generated SMILES tend to be invalid (15-80%) [16, 18, 19, 22, 23] – unless constraints are built into the model [2, 7, 17, 21]. The biggest downside of the SMILES representation is that it does not capture molecular similarity: substituting a single character can alter the underlying molecule structure significantly or invalidate it. Therefore, partially generated SMILES strings cannot be validated and transitions in the latent space of such models may lead to abrupt changes in molecule structure [5].

With respect to the generative model, most previous work use either VAEs or adversarial learning. VAEs and AAEs take a molecule representation as input and project it via an inference model (encoder) to a latent space, which is subsequently transformed by a decoder to produce a molecule representation [1, 2, 5, 8, 10, 11, 16, 17, 19, 24]. New molecules can be generated by drawing points from a simple prior distribution over the latent space (usually Gaussian) and feeding it to the decoder. AAEs [16, 24] perform variational inference by adversarial learning, using a discriminator as in GANs, which allows using a more complex prior distribution over the latent space. Both VAEs and AAEs are trained to minimize a reconstruction loss, which is expensive to compute. Standard GANs for molecule generation lack an encoder and a reconstruction loss, and are trained via a two-player game between a generator and discriminator [4, 18, 22]. The generator transforms a simple input distribution into a distribution in the space of molecules, such that the discriminator is unable to distinguish between real molecules in the training data and generated molecules. Due the lack of an inference model, such models cannot be used for efficient neighborhood search in the latent space.

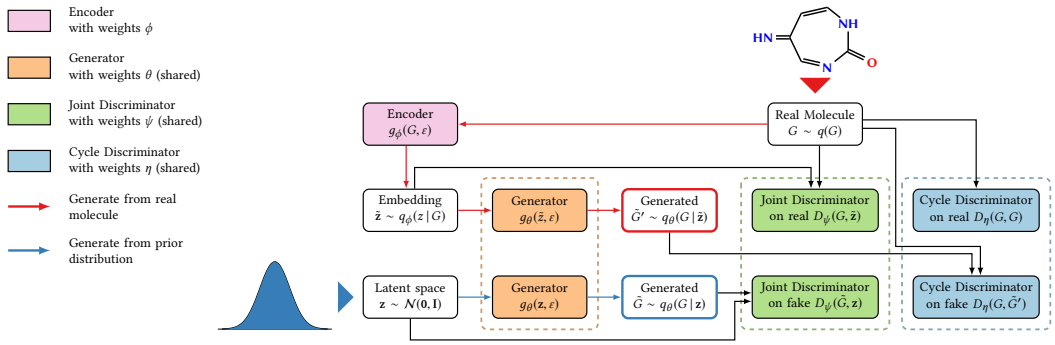


Figure 1: Overview of the proposed model. Boxes with identical background color represent neural networks that share their weights. The joint discriminator plays a similar role as the discriminator in standard GANs. The cycle discriminator enforces the reconstruction property without explicitly computing a reconstruction loss. Molecules can be generated by transforming a sample from a simple prior distribution (blue path), or by embedding a real molecule into the latent space and reconstructing its latent representation (red path).

3 Methods

We are seeking a generative model that learns a distribution in the space of molecules that enables us to generate novel molecules and to efficiently search the neighborhood of existing molecules. Here, we represent molecules as graphs. Our model, LF-MolGAN, has a GCN-based encoder that projects a graph into the latent space and a decoder that outputs a one-hot representation of atoms and an adjacency matrix defining atomic bonds. Hence, graph generation is performed in a single step, which allows considering global properties of a molecule. An undirected multi-graph $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ is defined by its vertices $\mathcal{V} = \{v_1, \dots, v_n\}$, relation types $\mathcal{R} = \{r_1, \dots, r_m\}$, and typed edges (relations) $\mathcal{E} = \{(v_i, r_k, v_j) \,|\, v_i, v_j \in \mathcal{V}, r_k \in \mathcal{R}\}$. Here, we only allow vertices to be connected by at most one type of edge. We represent a multi-graph G by its adjacency tensor $\mathbf{A} \in \{0, 1\}^{n \times n \times m}$, where A_{ijk} is one if $(v_i, r_k, v_j) \in \mathcal{E}$ and zero otherwise, and the node feature matrix $\mathbf{X} = (\mathbf{x}_{v_1}, \dots, \mathbf{x}_{v_n})^\top \in \mathbb{R}^{n \times d}$, where d is the number of features describing each node. Here, vertices are atoms, $\mathcal{R}$ is the set of bonds considered (single, double, triple), and node feature vectors $\mathbf{x}_{v_i}$ are one-hot encoded atom types (carbon, oxygen, ...), with d representing the total number of atom types. We do not consider hydrogen atoms explicitly, but implicitly add hydrogens to match an atom's valence.

3.1 Adversarially Learned Inference

We first describe Adversarially Learned Inference with Conditional Entropy (ALICE) [25], which is at the core of LF-MolGAN. It allows training our encoder-decoder model fully adversarially and implicitly enforces the reconstruction property without the need to compute a reconstruction loss (see fig. 1 for an overview). Training is performed by matching joint distributions over graphs $G \in \mathcal{G}$ and latent variables $\mathbf{z} \in \mathcal{Z}$. We denote by $q(G)$ the marginal data distribution, from which we have samples, by $\tilde{\mathbf{z}}$ a latent representation produced by the encoder, and by $\tilde{G}$ a generated graph. The *encoder* generative model over the latent variables is $q_\phi(z \,|\, G)$ with parameters ϕ, and the *decoder* generative model over graphs is $q_\theta(G \,|\, z)$, parametrized by θ. Putting everything together, we obtain the *encoder joint distribution* $q_\phi(G, z) = q(G) q_\phi(z \,|\, G)$, and the *decoder joint distribution* $p_\theta(G, z) = p_z(z) q_\theta(G \,|\, z)$. The objective of Adversarially Learned Inference (ALI) [26] is to match the two joint distributions by playing an adversarial game. A discriminator network D_ψ with parameters ψ is trained to distinguish samples $(G, \tilde{\mathbf{z}}) \sim q_\phi(G, z)$ from $(\tilde{G}, \mathbf{z}) \sim p_\theta(G, z)$:

$$\begin{aligned} \min_{\theta, \phi} \max_{\psi} \quad & \mathbb{E}_{G \sim q(G),\, \tilde{\mathbf{z}} \sim q_\phi(z \,|\, G)} [\log \sigma(D_\psi(G, \tilde{\mathbf{z}}))] \\ & + \mathbb{E}_{\tilde{G} \sim q_\theta(G \,|\, z),\, \mathbf{z} \sim p(z)} [\log(1 - \sigma(D_\psi(\tilde{G}, \mathbf{z})))]. \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Drawing samples $\tilde{\mathbf{z}}$ and $\tilde{G}$ is made possible by specifying the encoder q_ϕ and decoder q_θ as neural networks using the change of variable technique: $\tilde{\mathbf{z}} = g_\phi(G, \varepsilon)$,

$\tilde{G} = g_\theta(\mathbf{z}, \varepsilon)$, where ε is some random source of noise. This allows gradients to propagate from the discriminator to the encoder and decoder.

ALICE [25] extends this idea by including a cycle-consistency constraint via an additional adversarial loss. This encourages encoder and decoder networks to mimic the reconstruction property without explicitly computing a reconstruction loss. Therefore, we do not need to solve a computationally demanding graph isomorphism problem. To this end, a second discriminator network with parameters η is trained to distinguish a real graph G from its reconstructed graph $\tilde{G}'$:

$$\begin{aligned} \min_{\theta,\phi} \max_{\eta} \quad & \mathbb{E}_{G\sim q(G)}[\log \sigma(D_\eta(G, G))] \\ & + \mathbb{E}_{\tilde{G}'\sim q_\theta(G\,|\,\tilde{z}),\ \tilde{\mathbf{z}}\sim q_\phi(z\,|\,G)}[\log(1 - \sigma(D_\eta(G, \tilde{G}')))]. \end{aligned} \tag{2}$$

3.2 Generator

The generator network $g_\theta(\mathbf{z}, \varepsilon)$ takes a point $\mathbf{z}$ from latent space, and outputs a discrete-valued and symmetric graph adjacency tensor $\mathbf{A}$ and a discrete-valued node feature matrix $\mathbf{X}$. We use an MLP architecture as in [4], consisting of three hidden layers with 128, 256, 512 units and $\tanh$ activation, respectively. We extend $\mathbf{A}$ and $\mathbf{X}$ to explicitly model the absence of edges and nodes by introducing a separate ghost-edge type and ghost-node type. This will enable us to encourage the generator to produce chemically valid molecular graphs as described below. Thus, we define $\tilde{\mathbf{A}} \in \{0,1\}^{n\times n\times(m+1)}$ and $\tilde{\mathbf{X}} \in \{0,1\}^{n\times(d+1)}$. Each vector $\tilde{\mathbf{A}}_{ij\bullet}$, representing generated edges between nodes i and j, needs to be a member of the simplex $\Delta^m = \{(y_0, y_1, \ldots, y_m)\,|\,y_k \in \{0,1\}, \sum_{k=0}^m y_k = 1\}$, because only none or a single edge between i and j is allowed. Here, we use the zero element to represent the absence of an edge. Similarly, each generated node feature vector $\tilde{\mathbf{x}}_{v_i}$ needs to be a member of the simplex Δ^d, where the zero element represents ghost nodes.

Gumbel-softmax Trick. The generator is a neural network with two outputs, $\mathrm{MLP}_A(\mathbf{z}) \in \mathbb{R}^{n\times n\times(m+1)}$ and $\mathrm{MLP}_X(\mathbf{z}) \in \mathbb{R}^{n\times(d+1)}$, which are created by linearly projecting hidden units into a $n^2(m+1)$ and $n(d+1)$ dimensional space, respectively. Next, continuous outputs need to be transformed into discrete values according to the rules above to obtain tensors $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{X}}$ representing a generated graph. Since a simple argmax operation is non-differentiable, we employ the Gumbel-softmax trick [14, 15], which uses reparameterization to obtain a continuous relaxation of discrete states. Thus, we obtain an approximately discrete adjacency tensor $\tilde{\mathbf{A}}$ from $\mathrm{MLP}_A(\mathbf{z})$, and feature matrix $\tilde{\mathbf{X}}$ from $\mathrm{MLP}_X(\mathbf{z})$.

Node Connectivity and Valence Constraints. While this allows us to generate graphs with varying number of nodes, the generator could in principle generate graphs consisting of two or more separate connected components. In addition, generating molecules where atoms have the correct number of shared electron pairs (valence) is an important aspect the generator needs to consider, otherwise generated graphs would represent invalid molecules. Finally, we want to prohibit edges between any pair of ghost nodes. All of these issues can be addressed by incorporating regularization terms proposed in [9]. Multiple connected components can be avoided by generating graphs that have a path between every pair of non-ghost nodes. Using the generated tensor $\tilde{\mathbf{A}}$, which explicitly accounts for ghost edges, the number of paths between nodes i and j is given by

$$\tilde{\mathbf{B}}_{ij} = I(i = j) + \sum_{k=1}^{m}\sum_{p=1}^{n-1}\left(\tilde{\mathbf{A}}^p\right)_{ijk}. \tag{3}$$

The regularizer comprises two terms, the first term encourages non-ghost nodes i and j to be connected by a path, and the second term that a ghost node and non-ghost node remain disconnected:

$$\frac{\mu}{n^2}\sum_{i,j}\left[1 - (\tilde{\mathbf{x}}_{v_i})_0\right]\left[1 - \left(\tilde{\mathbf{x}}_{v_j}\right)_0\right]\left[1 - \tilde{\mathbf{B}}_{ij}\right] + \frac{\mu}{n^2}\left(\tilde{\mathbf{x}}_{v_i}\right)_0\left(\tilde{\mathbf{x}}_{v_j}\right)_0\tilde{\mathbf{B}}_{ij}, \tag{4}$$

where $\mu > 0$ is a hyper-parameter, and $(\tilde{\mathbf{x}}_{v_i})_0 > 0$ indicates that the i-th node is a ghost node.

To ensure atoms have valid valence, we enforce an upper bound – hydrogen atoms are modeled implicitly – on the number of edges of a node, depending on its type (e.g. four for carbon). Let $\mathbf{u} = (u_0, u_1, \ldots, u_d)^\top$ be a vector indicating the maximum capacity (number of bonding electron

pairs) a node of a given type can have, where $u_0 = 0$ denotes the capacity of ghost nodes. The vector $\mathbf{b} = (b_0, b_{r_1}, \ldots, b_{r_m})$ denotes the capacity for each edge type with $b_0 = 0$ representing ghost edges. The actual capacity of a node v_i can be computed by $c_{v_i} = \sum_{j \neq i} \mathbf{b}^\top \tilde{\mathbf{A}}_{ij\bullet}$. If c_{v_i} exceeds the value in $\mathbf{u}$ corresponding to the node type of v_i, the generator incurs a penalty. The valence penalty with hyper-parameter $\nu > 0$ is defined as

$$\frac{\nu}{n} \sum_{i=1}^{n} \max(0, c_{v_i} - \mathbf{u}^\top \tilde{\mathbf{x}}_{v_i}). \tag{5}$$

3.3 Encoder and Discriminator

The encoder network $g_\phi(G, \varepsilon)$, and the two discriminator networks $D_\psi(G, \mathbf{z})$ and $D_\eta(G_1, G_2)$ are closely related, because they all take graphs as input. First, we extract node-level descriptors by stacking several GCN layers. Next, node descriptors are aggregated to obtain a graph-level descriptor, which forms the input to a series of fully-connected layers. Here, inputs are multi-graphs with m edge types, which we model by extending the Graph Isomorphism Network (GIN) architecture [13] to multi-graphs. Let $\mathbf{h}_{v_i}^{(l+1)}$ denote the descriptor of node v_i after the l-th GIN layer, with $\mathbf{h}_{v_i}^{(0)} = \mathbf{x}_{v_i}$, then node descriptors get updated as follows:

$$\mathbf{h}_{v_i}^{(l+1)} = \tanh\left[\sum_{k=1}^{m} \mathrm{MLP}_{r_k}^{(l)}\left((1 + \epsilon^{(l)})\mathbf{h}_{v_i}^{(l)} + \sum_{u \in \mathcal{N}_{r_k}(v_i)} \mathbf{h}_u^{(l)}\right)\right], \tag{6}$$

where $\epsilon^{(l)} \in \mathbb{R}$ is a learnable weight, and $\mathcal{N}_{r_k}(v_i) = \{u \,|\, (u, r_k, v_i) \in \mathcal{E}\}$. Next, graph-level node aggregation is performed. We use skip connections [27] to aggregate node-level descriptors from all L GIN layers and soft attention [28] to allow the network to learn which node descriptors to use. The graph-level descriptor $\mathbf{h}_G$ is defined as

$$\mathbf{h}_{v_i}^{c} = \mathrm{CONCAT}(\mathbf{x}_{v_i}, \mathbf{h}_{v_i}^{(1)} \ldots, \mathbf{h}_{v_i}^{(L)}), \tag{7}$$

$$\mathbf{h}_{v_i}^{c'} = \tanh(\mathbf{W}_1 \mathbf{h}_{v_i}^{c} + \mathbf{b}_1), \qquad \mathbf{h}_G = \textstyle\sum_{v \in \mathcal{V}} \sigma(\mathbf{W}_2 \mathbf{h}_v^{c'} + \mathbf{b}_2) \odot \mathbf{h}_v^{c'}, \tag{8}$$

where $\mathbf{W}$ and $\mathbf{b}$ are parameters to be learned. Graph-level descriptors can be abstracted further by adding an additional MLP on top, yielding $\mathbf{h}'_G$. The discriminator network $D_\eta(G_1, G_2)$ contains two GIN-based towers to extract graph-level descriptors $\mathbf{h}'_{G_1}$ and $\mathbf{h}'_{G_2}$, which are combined by component-wise multiplication and fed to a 2-layer MLP. Network $D_\psi(G, \mathbf{z})$ has a noise vector as second input, which is the input to an MLP whose output is concatenated with $\mathbf{h}'_G$ from above and linearly projected to form $\log[\sigma(D_\psi(G, \mathbf{z}))]$. The encoder network $g_\phi(G, \varepsilon)$ has the same architecture as $D_\psi(G, \mathbf{z})$, except that its output matches the dimensionality of $\mathbf{z}$. Finally, we employ the 1-Lipschitz constraint in [29] to constrain discriminators D_ψ and D_η to be approximately 1-Lipschtiz continuous.

4 Experiments

In our experiments, we use molecules from the QM9 dataset [30] with at most 9 heavy atoms. We consider $d = 4$ node types (atoms C, N, O, F), and $m = 3$ edge types (single, double, and triple bonds). After removal of molecules with non-zero formal charge, we retained 131 941 molecules, which we split into 80% for training, 10% for validation, and 10% for testing.

We extensively compare LF-MolGAN against three state-of-the-art VAEs: NeVAE [10] and CGVAE [8] are graph-based VAEs with validity constraints, while GrammarVAE [2] uses the SMILES representation. We also compare against MolGAN [4], which is a Wasserstein GAN without inference network, reconstruction loss, node connectivity, or valence constraints. Finally, we include a *random graph generation model*, which only enforces valence constraints during generation, similar to CGVAE and NeVAE, but selects node types ($\mathbf{X}$) and edges ($\mathbf{A}$) randomly. Note that generated graphs can have multiple connected components if valence constraints cannot be satisfied otherwise; we consider these to be invalid. For NeVAE, we used our own implementation, for the remaining methods, we used the author's publicly available code to train and evaluate on the same set of molecules as our proposed approach. Implementation details are described in the supplement.

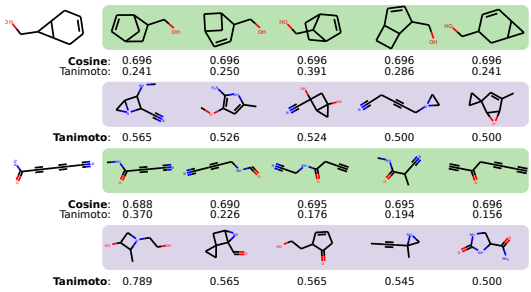


Figure 2: Nearest neighbors with respect to the molecule in the first column in embedding space (green rows) and according to Tanimoto similarity (purple rows).

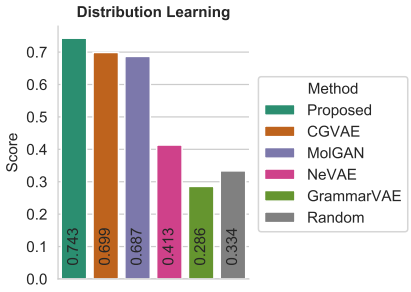


Figure 3: Comparison with respect to the proposed aggregate evaluation metric for distribution learning. See figs. 5 and A.1 for differences between individual distributions.

4.1 Latent Space

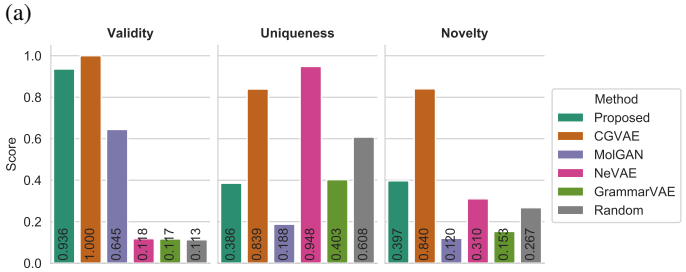
In the first experiment, we investigated properties of the encoder and the associated latent space. We projected molecules of the test set into the latent space, and performed a k nearest neighbor search to find the closest latent representation of a molecule in the training set in terms of cosine distance. We compared this against the nearest neighbors by Tanimoto similarity of ECFP4 fingerprints [31]. We found that the two approaches lead to quite different sets of nearest neighbors, as depicted in fig. 2. Nearest neighbors based on molecules' latent representation usually differ by small substructures. For instance, the query and fifth nearest neighbor in the first row of fig. 2 differ by the location of the side chain, whereas the ring structure is shared. On the other hand, the topology of nearest neighbors by Tanimoto similarity (second row) differs considerably from the query. Moreover, all but one nearest neighbor contain nitrogen atoms, which are absent from the query. In the second example (last two rows), the nearest neighbors in latent space are all linear structures with triple bonds that differ in the number and location of functional groups, whereas all nearest neighbors by Tanimoto similarity contain ring structures and only one contains a triple bond. Additional experiments with respect to interpolation in the latent space can be found in the supplement.

4.2 Molecule Generation

Next, we evaluated the quality of generated molecules. We generated molecules from $N = 10\,000$ latent vectors, sampled from a multivariate standard Gaussian and employed the metrics available in the benchmark suite proposed in [32]: *Validity* is the percentage of valid molecules, i.e., the molecular graph has a single connected component and all nodes have the correct valence. *Uniqueness* is the percentage of unique molecules within a set of N randomly sampled valid molecules. *Novelty* is the percentage of molecules not in the training data within a set of N unique randomly sampled molecules. Validity and novelty are calculated with respect to the set of all valid and all unique molecules, which we obtain by repeated sampling (up to 10 times). Thus, scores of models with limited validity/uniqueness will be penalized. Note that previous work defined uniqueness and novelty as the percentage with respect to all valid molecules, which is hard to interpret, because models with low validity would have high novelty. Hence, scores reported here are considerably lower compared to previously reported numbers.

Figure 4a shows that LF-MolGAN generates molecules with high validity (93.6%) and is only outperformed by CGVAE, which by design is constrained to only generate valid molecules. Moreover, LF-MolGAN ranks second in novelty and third in uniqueness; we will investigate the reason for this difference in detail in the next section on distribution learning. Graph-based NeVAE always generates molecules with correct valence, but often (88.2%) generates graphs with multiple connected components, which we regard as invalid. Its set of valid generated molecules has the highest uniqueness. We can also observe that graph-based models outperform the SMILES-based GrammarVAE, which is prone to generate invalid SMILES representation of molecules, which is a known problem [16, 18, 19, 22, 23]. MolGAN without inference network is struggling to generate molecules with valid valence and only learned one particular mode of the distribution, which we will discuss in more detail in the next section. It is inferior to LF-MolGAN in all categories, in particular with respect to novelty and uniqueness of generated molecules.

(a)



(b)

	Reconstructed (%)		Generated (%)	
Ghost-node bond	4	0.03	2	0.02
Valency	152	1.15	124	0.94
Split graph	541	4.10	680	5.16
Valid	12 487	94.71	12 378	93.89

Figure 4: (a) Overview of simple molecule generation statistics. (b) Frequency of errors of LF-MolGAN on test set.

Next, we inspected the reason for generated molecules being invalid to assess whether imposed node connectivity and valence constraints of our model are effective. Molecules can be generated by (a) reconstructing the latent representation of a real molecule in the test set, or (b) by decoding a random latent representation draw from an isotropic Gaussian (see fig. 1). Figure 4b reveals that most errors are due to multiple disconnected graphs being generated (4-5%). While individual components do represent valid molecules, we treat them as erroneous molecular graphs. In these instances, 95-96% of graphs have 2 connected components and the remainder has 3. We did not observe a clear preference towards the size of generated connected components. Only about 1% of molecules have atoms with improper valence and less than 0.1% of graphs have atomic bonds between ghost nodes. Therefore, we conclude that the valence constraint is highly effective.

Finally, we turn to the random graph generation model. It achieves a relatively high uniqueness of 60.8%, which ranks third, and has a higher novelty than MolGAN and GrammarVAE. Many generated graphs have multiple connected components, which yields a low validity. The fact that the random model cannot be clearly distinguished from the remaining models, indicates that validity, uniqueness, and novelty do not accurately capture what we are really interested in: *Can we generate chemically meaningful molecules with similar properties as in the training data?* We will address this issue by proposing a more appropriate set of evaluation metrics, which we will discuss next.

4.3 Distribution Learning

While the simple overall statistics in the previous section can be useful rough indicators, they ignore the physicochemical properties of generated molecules and do not capture to which extent a model is able to estimate the distribution of molecules from the training data. Therefore, we compare the distribution of 10 descriptors d_k used in [32] in terms of 1-Wasserstein distance. One descriptor, *internal similarity*, is a measure of diversity that is defined as the maximum Tanimoto similarity with respect to all other molecules in the dataset – using the binary Extended Connectivity Molecular Fingerprints with diameter 4 (ECFP4) [31]. The remaining descriptors represent physicochemical properties of molecules, such as molecular weight (see fig. 5 for a full list).

We approximate the distribution on the training data and the generated data using histograms $\mathbf{h}_k^{\text{train}}$ and $\mathbf{h}_k^{\text{gen}}$, and define the ground distance $\mathbf{C}_{ij}(k)$ between the edges of the i-th and j-th bin as the Euclidean distance, normalized by the minimum of the standard deviation of d_k on the training and the generated data. The 1-Wasserstein distance, also called Earth Mover's Distance (EMD), is defined as $\text{EMD}(\mathbf{h}_k^{\text{train}}, \mathbf{h}_k^{\text{gen}}) = \min_{\mathbf{P}\in\mathbf{U}(\mathbf{h}_k^{\text{train}}, \mathbf{h}_k^{\text{gen}})} \langle \mathbf{C}(k), \mathbf{P} \rangle$, where $\mathbf{U}(\mathbf{a}, \mathbf{b})$ is the set of coupling matrices with $\mathbf{P}\mathbb{1} = \mathbf{a}$ and $\mathbf{P}^\top \mathbb{1} = \mathbf{b}$ [33]. As overall measure of how well properties of generated molecules match those of molecules in the training data, we compute $\text{mean}\{\exp[-\text{EMD}(\mathbf{h}_k^{\text{train}}, \mathbf{h}_k^{\text{gen}})]\}_{k=1,\dots,10}$, which is summarized in fig. 3 (a perfect model would obtain a score of 1). Distributions of individual descriptors are depicted in fig. 5 and fig. A.1 of the supplement.

First of all, we want to highlight that using the proposed overall metric (see fig. 3), we can easily identify the random model, which is not obvious from the simple summary statistics in fig. 4. In particular, from fig. 4 we could have concluded that NeVAE is only marginally better than the random model. The proposed scheme clearly demonstrates that NeVAE is superior to the random model (overall score 0.431 vs 0.334). When considering differences between individual descriptors (see fig. A.1d), we can see that randomly generated molecules are not meaningful due to higher number of hydrogen acceptors, molecular weight, molecular complexity, and polar surface area.

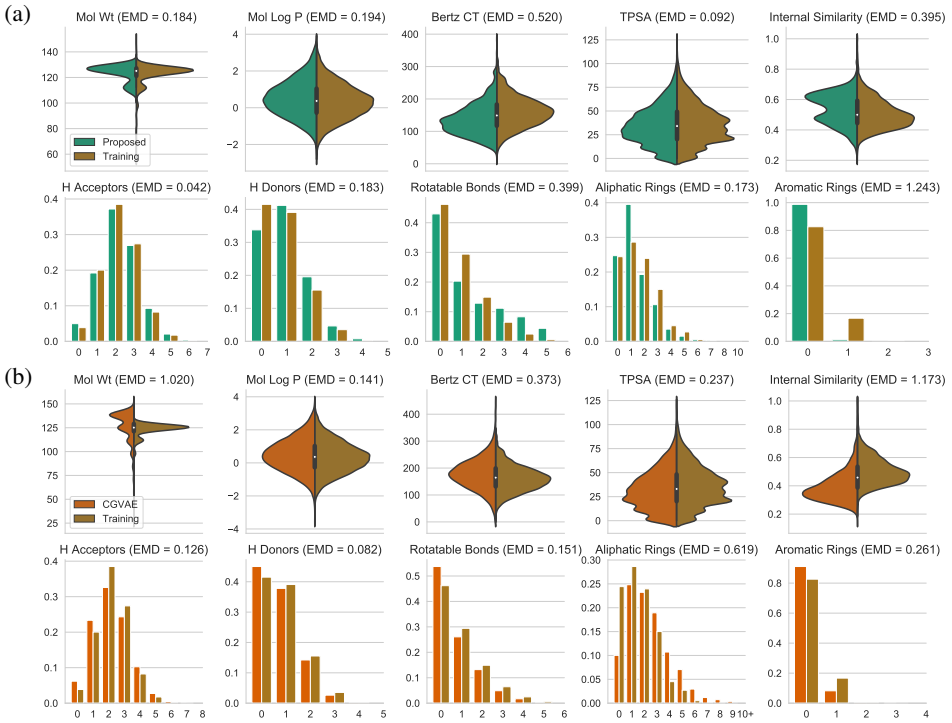


Figure 5: Distribution of 10 descriptors of molecules in the training data and of 10,000 unique molecules generated by (a) LF-MolGAN, and (b) CGVAE [8]. Mol Wt: molecular weight. Mol Log P: water-octanol partition coefficient. Bertz CT: molecular complexity index. TPSA: molecular polar surface area. H Acceptors (Donors): Number of hydrogen acceptors (donors).

Figure 3 also shows that LF-MolGAN achieved the best overall score. Molecular weight and the internal similarity of generated molecules best match that of the training data by a large margin (EMD = 0.17 and 0.40, see fig. 5a). The highest difference is due to the number of aromatic rings, which are underrepresented in generated molecules (EMD = 1.24). CGVAE is overall the best performing VAE (see fig. 5b). However, it notably produces molecules with larger weight (EMD = 1.02) and lower internal similarity (EMD = 1.17). The latter explains its high novelty value in fig. 4a: by creating a more diverse set of molecules than the training data represents, a high percentage of generated molecules is novel and unique. The trade-off between internal similarity and novelty/uniqueness also explains the results in fig. 4a: LF-MolGAN traded matching internal similarity for lower novelty/uniqueness. Molecules generated by NeVAE (see fig. A.1a) have issues similar to CGVAE: they have larger weight (EMD = 1.73) and lower internal similarity (EMD = 2.02), which benefits uniqueness. GrammarVAE in fig. A.1b is far from capturing the data distribution, because it tends to generate long SMILES strings corresponding to heavy molecules (EMD = 5.32 for molecular weight). With respect to MolGAN without inference network (see fig. A.1c): LF-MolGAN has a slight advantage, which is mostly due to hydrogen acceptors (0.04 vs 0.38) and TPSA (0.10 vs 0.37), but is also worse for some properties such as hydrogen donors (0.18 vs 0.07). An interesting detail can be derived from the distribution of molecular weight: MolGAN has problems generating molecules with intermediate to low molecular weight (< 115 g/mol), which highlights a common problem with GANs, where only one mode of the distribution is learned by the model (mode collapse). In contrast, LF-MolGAN can capture this mode, but still misses the smaller mode with molecular weight below 100 g/mol (see fig. 5a). This mode contains only 2344 molecules (2.2%), which makes it challenging – for any model – to capture. This demonstrates that our proposed set of metrics can reveal valuable insights that would have been missed when solely relying on validity, novelty, and uniqueness for evaluation.

5 Conclusion

We formulated generation and inference of molecular graphs as a likelihood-free adversarial learning task. Compared to previous approaches, it allows training without explicitly computing a reconstructing loss, which would require solving an expensive graph isomorphism problem. Moreover, we argued that the common validation metrics validity, novelty, and uniqueness are insufficient to properly assess the performance of algorithms for molecule generation, because they ignore physicochemical properties of generated molecules. Instead, we proposed to compute the 1-Wasserstein distance between distributions of physicochemical properties of molecules. We showed that the proposed LF-MolGAN allows efficiently exploring the space of molecules via molecules' continuous latent representation, and that it more accurately represents the distribution over the space of molecules than previous methods.

Acknowledgments

This research was partially supported by the Bavarian State Ministry of Education, Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research.

References

[1] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, et al. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Central Science* 4.2 (2018), pp. 268–276.

[2] Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. "Grammar Variational Autoencoder". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. 2017, pp. 1945–1954.

[3] P. G. Polishchuk, T. I. Madzhidov, and A. Varnek. "Estimation of the size of drug-like chemical space based on GDB-17 data". In: *Journal of Computer-Aided Molecular Design* 27.8 (2013), pp. 675–679.

[4] Nicola De Cao and Thomas Kipf. "MolGAN: An implicit generative model for small molecular graphs". In: (2018). arXiv: 1805.11973.

[5] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Junction Tree Variational Autoencoder for Molecular Graph Generation". In: (2018). arXiv: 1802.04364.

[6] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. "Learning Deep Generative Models of Graphs". In: (2018). arXiv: 1803.03324.

[7] Yibo Li, Liangren Zhang, and Zhenming Liu. "Multi-objective de novo drug design with conditional graph generative model". In: *Journal of Cheminformatics* 10.1 (2018), p. 33.

[8] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. "Constrained Graph Variational Autoencoders for Molecule Design". In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 7806–7815.

[9] Tengfei Ma, Jie Chen, and Cao Xiao. "Constrained Generation of Semantically Valid Graphs via Regularizing Variational Autoencoders". In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 7113–7124.

[10] Bidisha Samanta, Abir De, Niloy Ganguly, and Manuel Gomez-Rodriguez. "Designing Random Graph Models Using Variational Autoencoders With Applications to Chemical Design". In: *33rd AAAI Conference on Artificial Intelligence*. 2018. arXiv: 1802.05283.

[11] Martin Simonovsky and Nikos Komodakis. "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders". In: (2018). arXiv: 1802.03480.

[12] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. "Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation". In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 6412–6422.

[13] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *7th International Conference on Learning Representations*. 2019.

[14] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *5th International Conference on Learning Representations*. 2017. arXiv: 1611.01144.

[15] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *5th International Conference on Learning Representations*. 2017.

[16] Thomas Blaschke, Marcus Olivecrona, Ola Engkvist, Jürgen Bajorath, and Hongming Chen. "Application of Generative Autoencoder in De Novo Molecular Design". In: *Molecular Informatics* 37.1-2 (2018), p. 1700123.

[17] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. "Syntax-Directed Variational Autoencoder for Structured Data". In: *6th International Conference on Learning Representations*. 2018. arXiv: 1802.08786.

[18] 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". In: (2017). arXiv: 1705.10843.

[19] Jaechang Lim, Seongok Ryu, Jin Woo Kim, and Woo Youn Kim. "Molecular generative model based on conditional variational autoencoder for de novo molecular design". In: (2018). arXiv: 1806.05805.

[20] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. "Molecular de-novo design through deep reinforcement learning". In: *Journal of Cheminformatics* 9.1 (2017), p. 48.

[21] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. "Deep reinforcement learning for de novo drug design". In: *Science Advances* 4.7 (2018).

[22] Evgeny Putin, Arip Asadulaev, Quentin Vanhaelen, et al. "Adversarial Threshold Neural Computer for Molecular de Novo Design". In: *Molecular Pharmaceutics* 15.10 (2018), pp. 4386–4397.

[23] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. "Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks". In: *ACS Central Science* 4.1 (2018), pp. 120–131.

[24] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. "druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico". In: *Molecular Pharmaceutics* 14.9 (2017), pp. 3098–3104.

[25] Chunyuan Li, Hao Liu, Changyou Chen, et al. "ALICE: Towards Understanding Adversarial Learning for Joint Distribution Matching". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 5495–5503.

[26] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, et al. "Adversarially learned inference". In: *5th International Conference on Learning Representations* (2017). arXiv: 1606.00704.

[27] Keyulu Xu, Chengtao Li, Yonglong Tian, et al. "Representation Learning on Graphs with Jumping Knowledge Networks". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 5453–5462.

[28] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. "Gated Graph Sequence Neural Networks". In: *4th International Conference on Learning Representations*. 2016. arXiv: 1511.05493.

[29] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. "Improved Training of Wasserstein GANs". In: (2017). arXiv: 1704.00028.

[30] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. "Quantum chemistry structures and properties of 134 kilo molecules". In: *Scientific Data* 1 (2014).

[31] David Rogers and Mathew Hahn. "Extended-Connectivity Fingerprints". In: *Journal of Chemical Information and Modeling* 50.5 (2010), pp. 742–754.

[32] Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. "GuacaMol: Benchmarking Models for de Novo Molecular Design". In: *Journal of Chemical Information and Modeling* 59.3 (2019), pp. 1096–1108.

[33] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40.2 (2000), pp. 99–121.

[34] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations*. 2015. arXiv: 1412.6980.

[35] Greg Landrum, Brian Kelley, Paolo Tosco, et al. *Rdkit/Rdkit: 2018_09_1 (Q3 2018) Release*. 2018. DOI: 10.5281/zenodo.1468109.

[36] R'emi Flamary and Nicolas Courty. *POT Python Optimal Transport library*. 2017. URL: https://github.com/rflamary/POT.

A Additional Results

A.1 Distribution Learning

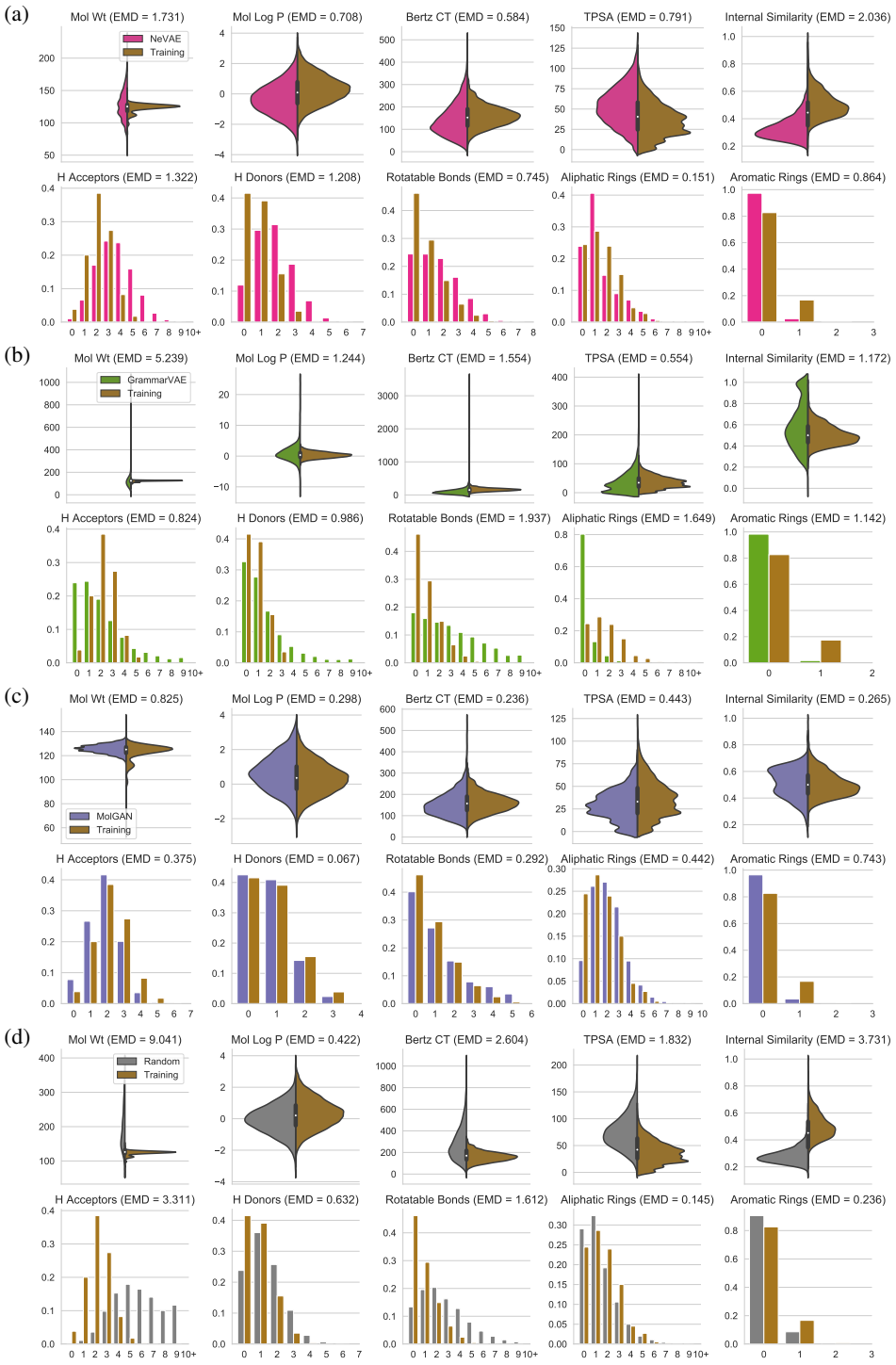


Figure A.1: Distribution of 10 descriptors of molecules in the training data and of 10,000 unique molecules generated by (a) NeVAE [10], (b) GrammarVAE [2], (c) MolGAN [4], and (d) randomly.

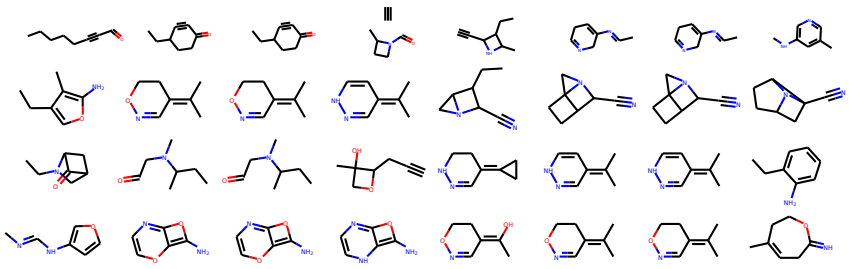


Figure A.2: Each row is a one-dimensional interpolation in latent space. The left and right most molecule are from the QM9 data, the remaining molecules lie on the line from the latent representation of the left most to the right most molecule.

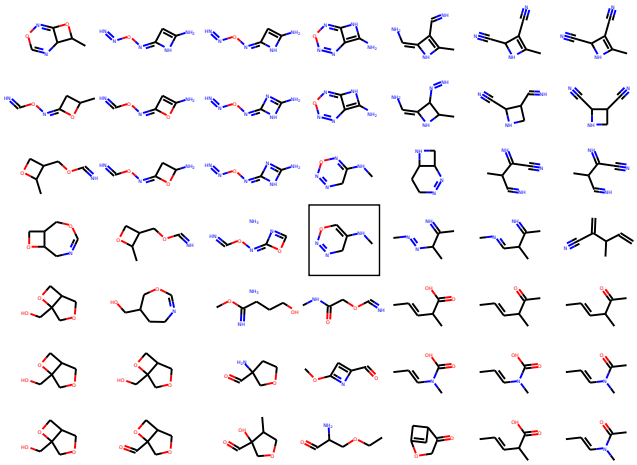


Figure A.3: Interpolation along two axes of the embedding space. The highlighted molecule in the middle is the query.

A.2 Latent Space Interpolation

In the first experiment, we interpolated between two molecules by computing their respective latent representation $\tilde{\mathbf{z}}_A$ and $\tilde{\mathbf{z}}_B$ and reconstructing molecules from latent codes on the line between $\tilde{\mathbf{z}}_A$ and $\tilde{\mathbf{z}}_B$. The results are depicted in fig. A.2. Interpolation between molecules appears smooth, but we can also see examples of latent representations corresponding to graphs with multiple connected components (top row), which we excluded from our analyses above, but included here for illustration.

Next, we investigated whether we can assign meaning to the different dimensions of the latent space. To this end, we projected a query molecule into the latent space and moved in equidistant steps in latent space, each time reconstructing the corresponding molecule. While we found it generally difficult to assign meaning to latent dimensions, we did find some dimensions that allowed interpretation. Figure A.3 illustrates the changes by moving along two axes of the latent representation. From left to right, the number of double and triple bonds tend to increase, whereas from top to bottom, the frequency of nitrogen atoms decreases and that of oxygen atoms increases. Therefore, the top right contains molecules rich in triple bonds and nitrogen, whereas the lower left contains oxygen-rich molecules with single bonds only.

B Implementation Details

GIN We implemented the sum over the descriptors of neighboring nodes in the GIN layer (6) as a dense matrix product between the adjacency matrix $\mathbf{A}_{\bullet,\bullet,r_k}$ and the matrix $(\mathbf{h}_{v_1}^{(l)}, \ldots, \mathbf{h}_{v_n}^{(l)})^\top$, which has a complexity of $O(n^2 d)$. Clearly, this is a bottleneck for large graphs. However, due to the sparsity of the adjacency matrix, we can be employ sparse matrix computation. In the first layer,

where descriptors are one-hot encoded node types, the computation requires $O(\bar{e})$ operations, with $\bar{e}$ being the average number of edges between nodes. After the second layer, descriptors will be dense and we have to compute the product between a sparse and a dense matrix, which requires $O(\bar{e}d)$ operations. To the best of our knowledge, most deep learning frameworks only support sparse computation on rank-2 matrices, therefore additional engineering would be required to incorporate the batch dimension for efficient sparse matrix computation.

Architecture If not mentioned otherwise, tanh is used as activation function. Encoders and discriminators have $L = 2$ GIN-layers with 128 units for each of the $m = 3$ edge types. The GIN-MLP in (6) has one linear layer. The two linear layers used for soft-attention and subsequent graph-pooling in (7) have 128 units each. The MLP taking the graph-pooled descriptor $\mathbf{h}_G$ and outputting $\mathbf{h}'_G$ has two fully-connected layers with 128 and 64 units, respectively. The cycle-discriminator $D_\eta(G_1, G_2)$ uses a 2-layer MLP with 64 units each to combine graph-level feature descriptors $\mathbf{h}'_{G_1}$ and $\mathbf{h}'_{G_2}$. Networks $D_\psi(G, \mathbf{z})$, $g_\phi(G, \varepsilon)$ have a noise vector as second input, which is the input to one fully-connected layer with 256 units. The hyper-parameter τ used in the Gumbel-softmax [14, 15] is set to $\tau = 1$. In total, our model consists of $927\,242$ weights that need to be optimized during training. We trained for 200 epochs using the Adam optimizer [34] with $\beta_1 = 0.5$, $\beta_2 = 0.9$, and initial learning rate 0.001, which is lowered to 0.0005 after 60 epochs, and 0.0001 after 120 epochs. The regularization weights μ and ν in (4) and (5) were tuned manually and set to 0.005 and 0.05. In practice, we compute a smoothed version of $\tilde{\mathbf{B}}_{ij}$ in (3) using $s(x) = \sigma(a(x - \frac{1}{2}))$ with $a = 100$, which improves numerical stability. The weight of the gradient penalty [29] of the discriminators was set to 10. Weights θ and ϕ of the encoder and decoder get updated at the same time, as are the weights ψ and η of both discriminators. Updates are performed sequentially, with a 5:1 ratio of discriminator to encoder/decoder updates. To stabilize training and encourage exploration, we added a discount term proportional to the variance of the predictions of the discriminator $D_\psi(\tilde{G}, \mathbf{z})$. The term was weighted by -0.2 and added to the remaining losses when updating weights θ and ϕ of the encoder and decoder.

Random Graph Generation For the random graph generation model, we used a similar setup as the generator used in NeVAE [10]. We first randomly select the atom types for each node in the graph to create $\mathbf{X}$, and then sequentially draw edges (v_i, r_k, v_j) such that the valence of atoms is always correct. We denote by $\delta(i, j \,|\, \mathcal{E}_l) \in \{0; 1\}$ whether nodes v_i and v_j are disconnected in the graph specified by the set of edges $\mathcal{E}_l$, and by $\delta(r_k \,|\, v_i, v_j, \mathcal{E}_l) \in \{0; 1\}$ whether nodes v_i and v_j can be connected by an edge of type r_k without violating valence constraints. The first mask is used to ensure only a single bond between atoms is formed, and the second mask to satisfy valence constraints. Independent random noise drawn from $\mathcal{N}(0, 1)$ is denoted by ε.

The probability of the i-th node feature vector $\mathbf{x}_{v_i}$ encoding the atom type is given by

$$P(\mathbf{x}_{v_i} = \mathbf{e}_j) = \frac{\exp \varepsilon_j}{\sum_{j'=1}^{d} \exp \varepsilon_{j'}},$$

where $\mathbf{e}_j$ is the one-hot encoding of the j-th atom type ($j = 1, \ldots, d$). Edges are added sequentially: in the l-th step, the probability of an edge (v_i, r_k, v_j), conditional on the previously added edges $\mathcal{E}_{l-1}$ is given by

$$\begin{aligned}
P((v_i, r_k, v_j) \,|\, \mathcal{E}_{l-1}) &= P(v_i, v_j \,|\, \mathcal{E}_{l-1}) P(r_k \,|\, v_i, v_j, \mathcal{E}_{l-1}), \\
P(v_i, v_j \,|\, \mathcal{E}_{l-1}) &= \frac{\delta(i, j \,|\, \mathcal{E}_{l-1}) \exp(\varepsilon_{i,j})}{\sum_{i',j'} \delta(i', j' \,|\, \mathcal{E}_{l-1}) \exp(\varepsilon_{i',j'})}, \\
P(r_k \,|\, v_i, v_j, \mathcal{E}_{l-1}) &= \frac{\delta(r_k \,|\, v_i, v_j, \mathcal{E}_{l-1}) \exp(\varepsilon_{r_k})}{\sum_{k'=1}^{r} \delta(r_{k'} \,|\, v_i, v_j, \mathcal{E}_{l-1}) \exp(\varepsilon_{r_{k'}})}.
\end{aligned}$$

We repeatedly sample edges until no additional valid edges can be added. Therefore, generated random molecular graphs will always have correct valences, but can have multiple connected components if valence constraints cannot be satisfied otherwise; we consider these samples to be invalid.

Software We implemented our model using TensorFlow version 1.10.0 and performed training on a NVIDIA Quadro P6000. We used RDKit [35] version 2018.09.1.0 for computing molecular descriptors. Validity of generated molecular graphs was assessed by RDKit's `SanitizeMol` function.

To evaluate how well models estimate the distribution of molecules from the training data, we used the benchmark suite GuacaMol version 0.3.2 [32] and the Python Optimal Transport (POT) library version 0.5.1 [36].