GraViti: Graph-Level VAE with Relaxed Permutation Invariance

Anonymous authors
Paper under double-blind review

Abstract

We introduce GraViti, a transformer-based graph-level variational autoencoder that encodes entire graphs into single, fixed-dimensional latent vectors rather than per-node embeddings, yielding a graph-level latent space that supports smooth interpolation, property-guided search, and other downstream tasks beyond the reach of node-level approaches. GraViti achieves state-of-the-art reconstruction accuracy on large molecular graph datasets while offering competitive single-step generative performance. Our central finding concerns where permutation invariance is needed in such a model, and where it is not. The encoder remains permutation-invariant throughout, ensuring the model generalizes to graphs regardless of input order. The reconstruction loss, however, does not need to be: when training data is provided in a consistent node order, comparing predictions to targets directly in that order, without a graph-matching step, is sufficient. Relaxing invariance in the loss alone lets GraViti avoid the cubic-cost matching step required by prior graph-level autoencoders, reducing complexity to quadratic while improving reconstruction fidelity. We show the resulting latent space is chemically meaningful, supporting controlled molecular editing and property optimization, recovering established chemical trends, and enabling direct regression of graph-level properties from latent embeddings.

1 Introduction

Autoencoders (Bengio et al., 2013) are unsupervised learning models designed to learn compact representations of complex data. They consist of two components: an encoder that maps an input data point (such as an image, or in our case a graph) into a low-dimensional latent vector, and a decoder that reconstructs the original data from this latent code. The model is trained end-to-end by minimizing the difference between the input and its reconstruction. This training process forces the latent representation to capture the essential structure of the data while discarding redundancy and noise. As a result, autoencoders are widely used for tasks such as denoising, compression, and extracting meaningful features that can support downstream applications (Berahmand et al., 2024; Li et al., 2023). To enforce structure on the latent representations, variational autoencoders (Kingma & Welling, 2014) constrain the distribution of all data embeddings z to match a fixed prior (usually normal) through minimizing a KL-divergence term.

While autoencoders have been applied across various modalities (Vafaii et al., 2024), graph autoencoding remains challenging due to graphs' inherently discrete and non-euclidean nature. Most approaches covering graph-oriented AE and VAEs do so at node-level. The early approach by Kipf et al. (Kipf & Welling, 2016) encodes a graph of N nodes into N latent vectors, each encoding one node's information, and the decoder recovers the adjacency matrix. More recent methods still follow this approach (Zahirnia et al., 2022; Ketata et al., 2025; Rigoni et al., 2025), which significantly limits the ability to capture meaningful distances between graphs or manifold structure (Sun et al., 2020).

While graph-level autoencoders (which encode an entire graph in a single latent representation) remain under-explored, with PIGVAE (Winter et al., 2021) and GRALE (Krzakala et al., 2025) being notable exceptions which will be further discussed in this work. This

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

can be traced to several core challenges inherent to their design. First, the encoder must be permutation-invariant, since node order is arbitrary while graph structure is not; without this property, the latent vector risks encoding artifacts of node indexing rather than genuine topology. Second, the model must handle variable graph sizes: the encoder must accept graphs of arbitrary cardinality, and the decoder must expand a single latent vector into a graph whose size is not known in advance, a requirement that makes graph-level decoding substantially more complex than its node-level counterpart.

In this work, we introduce GraViti, a graph-level VAE that encodes entire graphs into single latent vectors using a graph-transformer backbone. Our contributions are threefold. First, we propose a scalable architecture that achieves state-of-the-art reconstruction accuracy on large graph datasets. Second, we show that the learned latent space is well structured: on molecular benchmarks, it supports interpolation, controlled editing, and property-driven optimization, recovers well-established chemical phenomena, and provides competitive single-step molecular generation. Third, we show through ablations that this improvement stems specifically from relaxing permutation invariance in the reconstruction loss, while keeping the encoder itself permutation-invariant. This relaxation only requires that the training data follows a consistent node ordering, not that the ordering rule itself be known, a property that holds for many real-world graph datasets.

2 Background and state of the art

Permutation-invariant encoding of size-varying graphs Early GNNs learn permutation-equivariant node representations via message passing with permutation-invariant aggregation (Kipf & Welling, 2017; Xu et al., 2019), including attention-based aggregators that allow neighbor-specific information transmission (Vaswani et al., 2017; Veličković et al., 2017; Rong et al., 2020).

Since message-passing GNNs have limited expressivity (Xu et al., 2019), graph transformers were introduced, letting all nodes attend to all others at the cost of a full attention matrix (Dwivedi & Bresson, 2021; Müller et al., 2024), with positional encodings now carrying the structural information (Kreuzer et al., 2021; Huang et al., 2025b). Hybrid approaches such as GraphGPS (Rampášek et al., 2022) and SAT (Chen et al., 2022) combine transformers with message passing to balance efficiency and flexibility. Both families operate directly on sets of nodes and edges, giving permutation-equivariant representations that handle graphs of arbitrary size without a fixed ordering, making them suitable graph encoders as in the original Graph VAE (Kipf & Welling, 2016). A permutation-invariant pooling operation then produces a graph-level representation, though often at the cost of discarding substantial structural information (Ying et al., 2018).

Size-varying and permutation-invariant decoding Graph (variational) autoencoders must reconstruct a graph of unknown size and determine the correspondence between reconstructed and input nodes to compute a reconstruction loss. Node-level approaches handle this naturally: the encoder produces one latent vector per node, and the decoder reconstructs adjacency and attributes in the same fixed order, so correspondence is implicit in node indexing. Common decoders include inner-product or MLP heads (Kipf & Welling, 2016), sometimes refined via message passing (Grover et al., 2019).

Graph-level autoencoders lack this implicit ordering and must instead resolve it explicitly. Graph matching (Simonovsky & Komodakis, 2018; Huang et al., 2025a) finds an optimal node alignment, at the cost of solving an expensive quadratic assignment problem. PIG-VAE (Winter et al., 2021) instead learns a relaxed permutation matrix by sorting node embeddings, avoiding the assignment problem but offering only limited alignment; GRALE (Krzakala et al., 2025) improves on this with a Sinkhorn-based matcher and an optimal-transport reconstruction loss, yielding a full doubly stochastic alignment matrix with better stability and expressivity.

Variable graph size is typically handled by padding to a fixed maximum, as in NLP transformers (Vaswani et al., 2017), often via a "dummy node" class (Krzakala et al., 2025; Ling et al., 2023), with the decoder predicting a maximal-size graph plus a node mask. This

has two drawbacks: it prevents generalization to larger graphs, and the dummy class biases node classification with an artificial label. PIGVAE instead broadcasts the graph representation across a sequence transformer, using sinusoidal positional encodings to differentiate duplicated tokens; this avoids both a maximum size and a dummy class, masking only non-existent nodes, with size predicted by an MLP from the latent.

Overall, existing methods trade off scalability against expressivity: node-level approaches scale well but yield incoherent graph-level latent spaces, while graph-level methods need a costly matching step. This motivates methods that learn meaningful graph-level representations without expensive alignment, while remaining expressive and scalable.

3 The GraViti architecture

Graph representation Let N denote the size of the graph (i.e., the number of nodes). In this work, we assume that graphs have discrete node classes and edge classes instead of continuous features (e.g. atom classes, color). A graph is represented by two objects. First is its node-class matrix $X \in \mathbb{R}^{N \times n_c}$ where n_c is the number of node classes and X_i is the one-hot encoding of the i-th node's class. Importantly, we do not introduce a "dummy node" class; all classes correspond to real nodes. Then $E \in \mathbb{R}^{N \times N \times n_e}$ is the edge class tensor, where n_e is the number of edge classes, including both actual edge types and an additional class representing "no edge".

To enrich the graph representation, we compute directly from the graph a global feature vector $y \in \mathbb{R}^{d_y}$ containing graph-level information. In practice, $d_y = n_c + n_e + 1$, and y encodes the count of each node and edge type, as well as the graph size. We also compute $P \in \mathbb{R}^{N \times d_P}$, the positional encodings for the nodes, obtained by concatenating Laplacian positional encodings and random walk positional encodings. These inject global structural information into the graph transformer at no extra cost.

Finally, when relevant for the field, we also exploit a matrix of node-specific features derived from domain knowledge $K \in \mathbb{R}^{N \times d_k}$. For molecular datasets, these include atomic properties such as atomic number, atomic mass, valence electrons, and electronegativity.

Since training involves batching, all graphs in a batch must share the same size. To achieve this, we pad all graphs to reach the size of the largest graph in the batch, extending X, P, K, and E accordingly. This padding has no semantic meaning, and the model is designed to ignore it entirely. The padded size of the input does not influence the decoder or model size in any way. The model is illustrated in figure 1; the letters in paragraph titles refer to the respective components on the figure.

Encoding node, edge, and global representation (A) Our encoder maps a triplet X, E, y (optionally P and K) to a mean and standard deviation $z_\mu, z_\sigma \in \mathbb{R}^{d_z}$ where d_z is the latent dimension. At first, the node representation matrix is expanded into X_{emb} by concatenating a learned shallow embedding of the node class, both positional encoding (Laplacian and random-walk) processed through a dedicated linear layer and optionally node attributes processed through a dedicated linear layer. Edge representations are obtained via a learned embedding layer of their discrete classes, producing E_{emb}. The global feature vector y is projected through a linear layer into y_{emb}.

All three tensors (X_{emb}, E_{emb}, y_{emb}) undergo layer normalization and are updated through multiple graph transformer layers, yielding X_{rep}, E_{rep}, and y_{rep}. This step uses the specific graph transformer architecture introduced in DiGress (Vignac et al., 2023) and updated in CatFlow (Eijkelboom et al., 2024). This design enables simultaneous updates of node, edge, and graph-level features while ensuring permutation-equivariance.

Encoder readout (B) Traditional readout functions for graph-level representations (such as sum, mean, or max pooling) are permutation-invariant at the cost of a large quantity of information. To ensure a highly expressive latent space, we adopt a sequence transformer-based readout mechanism. Design motivations as to why we maintain a permutation-invariant encoder are detailed in Appendix A.

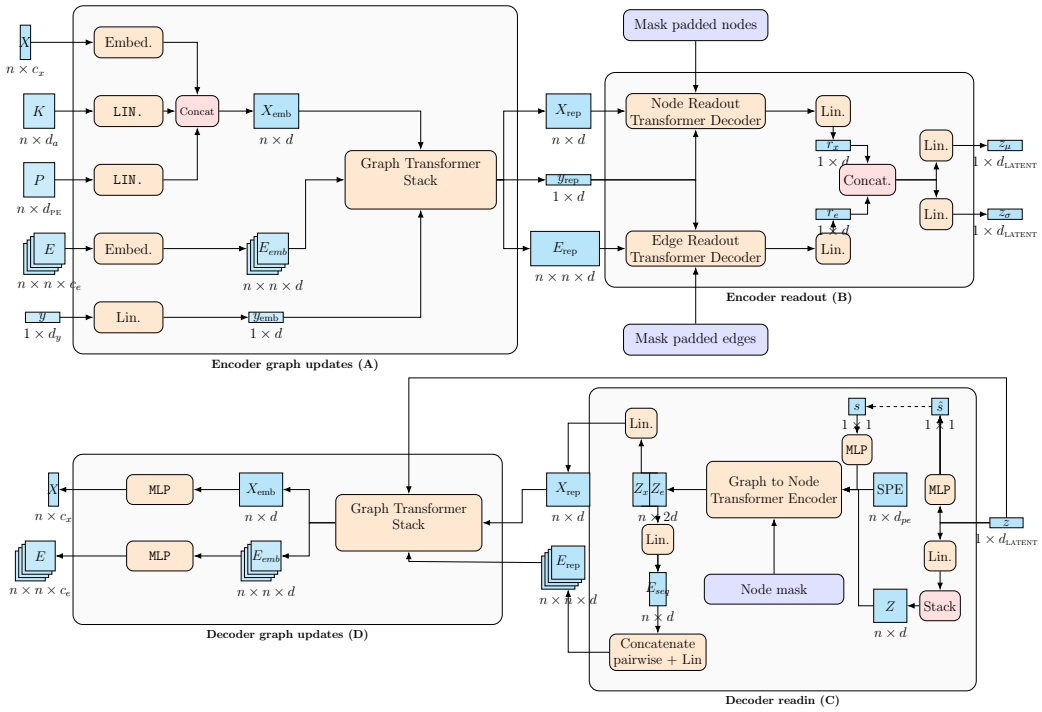


Figure 1: The architecture of our model. The data space is on the left, while the latent space is on the right. Encoder on top reads left to right, decoder reads right to left.

Unlike GRALE, which applies a transformer solely on node embeddings, our approach incorporates nodes, edges, and global features into the readout to prevent overloading node embeddings. Specifically, the readout consists of two parallel sequence-decoder transformers: one for nodes and one for edges. Both transformers ignore padded nodes and edges. y_{rep} serves as the target for both transformers, injecting global context into the readout.

Since positional encodings are omitted at this stage, the inherent permutation invariance of the sequence transformer guarantees that the output is independent of node ordering. Finally, node and edge outputs are normalized independently and projected through linear layers to produce the graph-level representation. This one can be either a single vector (classic autoencoder case) or a pair of same-sized vectors (in the variational case).

Graph size prediction (C) One crucial information required to reconstruct the graph is its size, which needs to be recovered from the latent representation. We make the design choice of disentangling this crucial structural information from the rest of the latent representation by forcing the model to encode it in only the first d_s dimensions of the latent space. The log-size $\hat{s}$ is then predicted by an MLP. In order for the decoding process to leverage the size information, we use another MLP to re-encode the log-size s into d_s dimensions that replace the corresponding latent features. During training, no gradient flows between $\hat{s}$ and s. The predicted log-size $\hat{s}$ is optimized via a smooth L1 loss against the ground-truth log-size, while the ground-truth value s is directly reinjected into the latent representation. This ensures that the corresponding latent dimensions encode size information exclusively, rather than relying on indirect supervision through masking.

Decoding graph-level into node-level representation (C) The first module of our decoder has to invert the readout function, i.e. obtain node representations from graph representations. Similarly to PIGVAE we first broadcast the graph-level information into a node-wise representation. This is done by simply duplicating our graph-level representation (i.e. the latent representation projected through a linear layer) N times. We therefore obtain a sequence of N identical vectors of dimension D. In order to break the symmetries

and allow the transformer to differentiate between these pseudo-tokens, we add sinusoidal positional encoding to the sequence. During training, N is determined as the ground truth size, which is also used for reinjection. During inference, N is $\texttt{round}(\exp(\hat{s}))$.

This sequence is then updated into another $N \times D$ sequence through a sequence transformer encoder, matching our readout. We then split this sequence width-wise into two sequences of size $N \times \frac{D}{2}$, a node-oriented sequence X_x and an edge-oriented sequence X_e. We create an edge-feature matrix ϕ_e by concatenating the rows of X_e in pairs: $\phi_{e,(i,j)} = X_{e,i}|X_{e,j}$; finally, symmetric edge-feature Φ_e matrix is obtained as $\Phi_{e,(i,j)} = \frac{1}{2}\left(\phi_{e,(i,j)} + \phi_{e,(j,i)}\right) \in \mathbb{R}^{n \times n \times D}$. In parallel, we create a node features matrix $\phi_x \in \mathbb{R}^{n \times D}$.

Updating and refining the graph (D) We update the obtained graph through another graph transformer, with similar architecture as that of the encoder. Finally, we apply an MLP to the nodes and one to the edges as a final prediction head. In our case, those are classifiers. We only reconstruct X and E, which is all that is needed for a graph. Also note that the size is predicted, and there is therefore no need to predict an extra dummy node class. The model is trained following the standard VAE objective, using a cross-entropy loss for all node and edge classes predictions. A KL term regularizes the latent variables toward a normal prior. Details of the loss formulation are provided in Appendix E.

4 Comparison to GRALE and PIGVAE

Loss function Both GRALE and PIGVAE leverage a permutation-invariant scheme for their loss functions, allowing to reconstruct the graph in any order while still obtaining a relevant reconstruction loss: PIGVAE predicts a permutation matrix obtained through a solver, while GRALE iterates upon this by using a Sinkhorn matcher to obtain a relaxed permutation matrix, and using the Any2Graph loss (Krzakala et al., 2024) on the soft assignment. While these approaches are elegant and well-suited to fully general graph settings, this incurs a significant computational overhead (at least cubic for both approaches). In contrast, our model relies on a simpler reconstruction objective made possible by the use of a consistent node ordering, allowing a quadratic complexity. In practice, this results improved reconstruction quality on the benchmarks we consider, especially as graph size increases.

Graph size flexibility GRALE decodes graphs by iteratively refining a learned node and edge representation of fixed maximal size. This mechanism inherently enforces an upper bound on the size of generated graphs. In contrast, our approach, like PIGVAE, does not impose such a limit: the graph-level latent is replicated to the desired number of nodes, and continuous sequence-based positional encodings are used to distinguish node-level embeddings. As a result, the model architecture is independent of graph size, although computational cost for full attention naturally scales with it.

Variational component Similar to PIGVAE, our method follows a fully variational formulation, enabling likelihood-based training and explicit regularization of the latent space. Unlike GRALE, which relies on a tokenized latent representation, we operate in a continuous latent space. In practice, this enables a well-structured latent, unlocking in particular generative and editing capabilities.

5 Experimental results

Experimental setting A prominent application domain for graph-level representation learning is computational chemistry, where molecules are naturally represented as attributed graphs of atoms and bonds. We consider two benchmark datasets: QM9 (Ramakrishnan et al., 2014), a dataset of approximately 130,000 molecules containing up to 9 heavy atoms, and PubChem (Kim et al., 2016). Following GRALE, we focus on the PubChem16 and PubChem32 subsets, comprising approximately 13 million and 74 million molecules with up to 16 and 32 heavy atoms, respectively. Both datasets are publicly provided in a canonical ordering established by proprietary softwares, as detailed in Appendix B alongside the dataset statistics. As is standard in molecular graph modeling, we remove explicit hydrogen atoms (Shi et al., 2020; Luo et al., 2021; Eijkelboom et al., 2024; Jo et al., 2022) and

Table 1: Percentage of perfectly reconstructed graphs and edit distance. Molecules are in kekulized form. *PIGVAE's results from (Krzakala et al., 2025).

Dataset	Metric	GRALE	PIGVAE*	GraViti (AE)	GraViti (VAE)
QM9NoHydro	Edit Dist.	**0.08**	N/A	0.11	0.10
	Accuracy	95.7	N/A	97.6	**97.8**
PubChem16	Edit Dist.	0.06	1.69*	**0.04**	**0.04**
	Accuracy	96.9	41.0*	**99.5**	**99.5**
PubChem32	Edit Dist.	0.70	2.53*	0.15	**0.14**
	Accuracy	73.3	24.9*	98.2	**98.4**
ASIA-BN	Edit Dist.	**0**	N/A	0	0
	Accuracy	**100**	N/A	100	100

Table 2: Training efficiency comparison between GRALE and GraViti on PubChem.

Dataset	Model	Params	Epochs	Accuracy (%)	Time (h)
PubChem16	GRALE	7M	5	94.6	44.0
	GraViti	7M	5	98.4	37.5
PubChem32	GRALE	35M	5	73.3	229.8
	GraViti	35M	5	97.5	186.5

study both the aromatic form and the kekulized form. The resulting graphs contain three bond types (resp. four) in kekulized form (resp. aromatic): single, double, triple (resp. and aromatic). An additional edge type is introduced to denote "no-edge". We also consider non-molecular graphs through the Bayesian Networks-Asia dataset, which consists in directed acyclic graphs. Training settings can be found in Appendix C.

Reconstruction performance Table 1 reports the graph reconstruction performance of our model using what we consider to be the primary evaluation metric of this study. We deliberately adopt a particularly strict reconstruction criterion, under which a graph is deemed correctly reconstructed only if all node types and all edge types are predicted without error. For molecules in aromatic form, this requirement is further extended to include the exact reconstruction of the hydrogen count and formal charge for every aromatic atom. The reported accuracy therefore corresponds to the proportion of graphs that are perfectly reconstructed. We also report the less strict edit distance (Wang et al., 2024) for each model. All models are trained until convergence.

We observe that our model outperforms the matching-based baselines on all molecular datasets. We also notice that, as the size of the graphs in the datasets increases, so does the performance gap, with our model beating the baselines by 25% on PubChem32. We attribute this success to the increasing difficulty of training a proper matching as graph size increases for GRALE. This demonstrates in particular that the canonical ordering provided in molecular datasets is sufficiently consistent to ensure good performance, especially as graphs get larger, rejoining results observed on readout functions (Buterez et al., 2022). Results in aromatic form are similar (see Appendix D). The ASIA-BN dataset was perfectly fitted by all models.

Performance and efficiency On top of the higher performance ceiling, we further demonstrate the efficiency gains obtained by leveraging a canonical ordering. To this end, we train two models of comparable size, one GRALE and one GraViti, for the same number of epochs. The results, reported in Table 2, show that eliminating the matching step leads to a substantial reduction in training time while simultaneously yielding higher reconstruction accuracy. These gains become even more pronounced as graph size increases.

Table 3: Architectural ablation study results. The baseline model is the VAE on PubChem16

Ablated component	Alternative	Acc.	Diff.	Ed. Dist.	Diff.
Baseline		99.5	0	0.04	0
Transformer Encoder	GNN-based Enc.	97.2	-2.3	0.15	+0.1
Transformer Decoder	MLP Decoder	99.3	-0.2	0.06	+0.01
Atom attributes Used	Not used	99.4	-0.1	0.05	+0.01
Laplacian + RW Pos. Enc.	Laplacian only	99.2	-0.3	0.05	+0.01
Laplacian + RW Pos. Enc.	RW only	99.5	0	0.04	0
Laplacian + RW Pos. Enc.	None	99.2	-0.3	0.06	+0.02
Global features	No global features	99.4	-0.1	0.05	+0.01

Table 4: Percentage of perfectly reconstructed graphs and edit distance for the Coloring dataset. Results for PIGVAE are reported from (Krzakala et al., 2025).

Dataset	Metric	GRALE	PIGVAE*	Ours (AE)	Ours (VAE)
Coloring no order.	Edit Dist.	**0.11**	2.13*	35.5	35.3
	Accuracy	**95.8**	85.3*	0	0
Coloring + order.	Edit Dist.	**0.06**	N/A	0.57	0.79
	Accuracy	**96.9**	N/A	80.5	76.3

Ablation studies To determine whether our gains stem from architectural choices or from the canonical ordering used in the loss, we ablate several components of the architecture (Table 3). The model proves remarkably robust to most modifications. The largest drop comes from replacing the transformer encoder with a message-passing GNN, though this variant still outperforms GRALE. Training with GRALE's matching-based loss instead, however, yields 96.5% accuracy, matching GRALE's 96.9%, which strongly suggests that the reconstruction objective accounts for most of the improvement.

Limitations of Canonical Ordering Dependance To illustrate the limits of our scope, we test on COLORING, a very general dataset with no natural node ordering, where our model should struggle. As expected, baseline reconstruction accuracy is near 0%. Imposing a generic, domain-agnostic WL-color-based node ordering (details in Appendix F) raises accuracy to 80.5%, while remaining significantly below matching-based models. Table 4 reports the full results alongside comparisons to matching-based baselines. This shows a middle ground between no ordering at all, where our model fails outright, and the well-informed, domain-specific canonical orderings available for molecules, where our model achieves state-of-the-art performance.

Interpolation We also demonstrate that the latent embedding enables smooth transition between two molecules. To illustrate this, we perform linear and rotative interpolation directly in latent space between the embeddings of two valid molecules. This produces a continuous trajectory of latent vectors, each of which we decode back into a molecular graph. Because the decoder may map nearby latent points to identical structures, we retain only the distinct molecules encountered along the interpolation. Example interpolations sequences are shown in Figure 2, with more in Appendix J.

Local Property Optimization We demonstrate that our latent embedding can be used not only to represent molecules but also to edit them in a controlled manner to improve a target property. To illustrate this capability, we perform black-box optimization directly in the latent space using CMA-ES (Hansen et al., 2019; Hansen & Ostermeier, 2001) to maximize logP (Klimoszek et al., 2024; Tsantili-Kakoulidou & Demopoulos, 2021), which quantifies the ratio of lipo- over hydro-philicity and is widely sought in drug design. Figure 3 shows representative optimization trajectories.

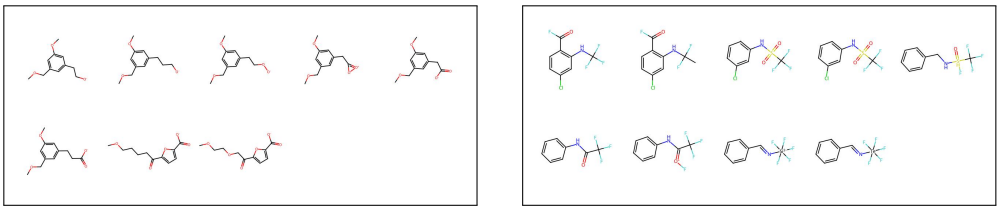


Figure 2: Interpolation between two pairs of molecules (only steps where the molecule is updated are shown). Starting molecules are top-left, progression is left to right, top to bottom.

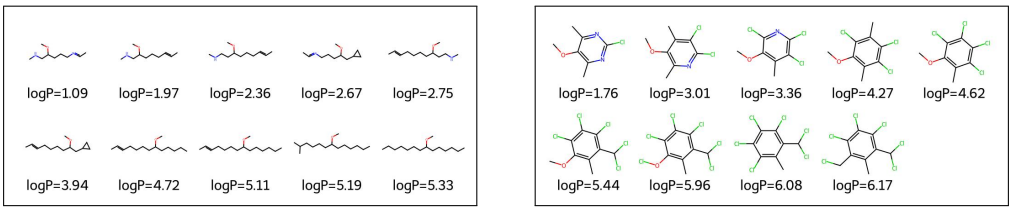


Figure 3: LogP maximization from 2 molecules. Starting from the embedding of the top-left molecule, the latent representation is optimized through CMA-ES to increase its logP (indicated below each molecule). Steps are displayed (left-to-right, top to bottom).

The results highlight that the latent space supports smooth and minimal structural edits that incrementally improve the objective. CMA-ES parameters enforce small steps between successive candidates, and we observe that the decoded molecules evolve gradually. We also notice the recovery of well-established chemical patterns during optimization. In the left example, the algorithm increases logP by progressively lengthening the carbon chain. This behavior aligns with a documented principle: longer alkyl chains are increasingly hydrophobic, raising logP (Mozrzymas, 2020; Yang et al., 2024; Xie et al., 2021; Smith & Tanford, 1973). On the right, the molecule develops a tetrachlorobenzene moiety, another structure associated with increased hydrophobicity (McPhedran et al., 2013). The fact that such trends emerge purely from latent-space navigation suggests that the model captures chemically meaningful directions that can be exploited for property-guided molecular editing.

We also note that VAE performs significantly better than AE on this task, with the latter often not finding a near improvement beyond the original embedding, showing the benefits of imposing structure on the latent. On a batch of 1024 molecules, average LogP improvement was 0.16 using AE, and 2.72 using VAE. Figure 4 in Appendix G illustrates the performance on individual samples alongside more examples.

Graph generation capability While molecular graph generation is not the main focus of this work, it arises naturally from the architecture of a VAE. In particular, molecules can be generated in a single decoder pass by sampling the latent prior. We report generation results mainly as a diagnostic of the learned latent space and decoder, rather than as an attempt to optimize a dedicated molecular generator or establish a new state of the art, especially as VAEs are generally not competitive compared to more sophisticated, multi-step approaches such as diffusion (Vignac et al., 2023) or flow matching (Cheng et al., 2024).

Table 5 reports performance on QM9 and PubChem16 in terms of validity, uniqueness, novelty, and Frechet ChemNet Distance (Preuer et al., 2018). Despite using no chemistry-specific constraints during decoding, the non-aromatic QM9 model achieves 93.18% validity and 98.48% uniqueness. These results are competitive with recent methods specifically designed for one-step generation. For example, one-step Categorical Flow Maps (Roos et al., 2026) reports 95.8% validity and 89.2% uniqueness with its ECLD objective, and 90.0% validity and 91.8% uniqueness with its CSD objective. Our model therefore reaches a comparable validity regime while producing a larger fraction of unique samples.

Table 5: Generation performance of GraViti on QM9 and PubChem16, both in aromatic (A) and kekulized (K) form. Baselines for QM9 are reported directly from (Roos et al., 2026). VAE means direct latent decoding. VUN is the ratio of valid, unique and novel molecules (product of the three metrics).

Dataset/Model	Steps	Valid. (%)	Uniq. (%)	Novelty (%)	VUN (%)	FCD
QM9(K)/CFM ECLD	1	95.8	89.2	N/A	85.5	2.16
QM9(K)/CFM CSD	1	90.0	91.8	N/A	82.6	2.14
QM9(K)/GraViti	VAE	93.2	98.5	N/A	91.8	2.88
QM9(A)/GraViti	VAE	92.0	99.0	N/A	91.1	3.00
PubChem16(K)/GraViti	VAE	68.1	100.	91.7	62.4	4.20
PubChem16(A)/GraViti	VAE	46.5	100.	98.1	45.6	3.57

Using explicit aromatic representation, we obtain 46.5% validity on PubChem16 and 92% on QM9. A large fraction of invalid molecules contain aromatic edges not forming closed aromatic cycles, reflecting the known difficulty of graph neural networks, including transformers, in representing cyclic structures without higher-order mechanisms (Zhu et al., 2023; Müller & Morris, 2024; Nikolentzos et al., 2026). Training the model on the corresponding kekulized representation improves validity to 68% on PubChem and 93% on QM9. Examples of generated molecules are shown in Appendix K.

We interpret these results as indicating that the latent space is locally well structured, enabling the generation of diverse and novel molecules in nearby regions, but globally insufficiently compact, leading to a significant fraction of invalid decodings when sampling more broadly. This observation is consistent with the literature on molecular VAEs (Kusner et al., 2017; Jin et al., 2018; Nguyen & Karolak, 2025; Hu et al., 2023), which has shown that generation validity is commonly low in the absence of additional domain-specific constraints (Bhadwal et al., 2025; Ochiai et al., 2023). Moreover, validity is known to be anti-correlated with uniqueness and novelty in molecular generative models (Mahmood et al., 2021). In order to further probe the latent generative capabilities, we train a simple, cheap flow model in our latent space, raising validity to 89.7% on PubChem16 in one step only (Appendix L).

Denoising One application of autoencoding is removing noise from a perturbed input. In Appendix I, we perturb molecules (up to 10 node and 10 edge labels are randomly changed), and show that we still recover valid molecules as output.

Regression We further show that chemical and graph-level properties can be predicted directly from the latent space, including molecular quantum properties on QM9 and BIC scores on Bayesian networks (Appendix H).

6 Conclusion and future work

We introduced `GraViti`, a transformer-based graph-level variational autoencoder that learns compact latent representations of entire graphs and achieves state-of-the-art reconstruction accuracy among graph-level VAEs on large graph datasets.

We demonstrate on molecular graphs that the latent space is coherent and chemically meaningful: it supports smooth interpolation, denoising, and property-guided optimization, even recovering known chemical principles. Beyond molecular tasks, we also obtain strong performance on downstream inference problems for Bayesian graphs.

Finally, we show that a permutation invariant objective is not always required; when a stable node ordering is available, relaxing this constraint yields substantial gains in reconstruction quality and scalability. Future work includes integrating domain-aware constraints to improve generation validity, exploring richer latent priors, and evaluating transfer to other graph domains such as circuits, proteins, and knowledge graphs. Overall, `GraViti` offers a practical foundation for graph-level representation learning and provides a usable latent space for downstream tasks.

LLM Usage In this work, we used generative AI tools for rephrasing and polishing the text, and assisting in code development.

We have **not** used generative AI tools for generating synthetic data, developing theoretical models or conceptual framework, formulating mathematical claims or any other task with required or recommended disclosure.

All AI-generated or AI-edited content was checked by humans. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement All used datasets are openly accessible. The code is provided as supplementary materials, which contains our architecture, as well as all scripts required to pretreat and split the data, and train and evaluate the models. All hyperparameters are detailed in appendix C.

References

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

Kamal Berahmand, Fatemeh Daneshfar, Elaheh Sadat Salehi, Yuefeng Li, and Yue Xu. Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, 57(2), February 2024. ISSN 1573-7462. doi: 10.1007/s10462-023-10662-6. URL http://dx.doi.org/10.1007/s10462-023-10662-6.

Arun Singh Bhadwal, Monika Kumari, and Anil Kumar. Pcf-vae: posterior collapse free variational autoencoder for de novo drug design. *Scientific Reports*, 15(1), October 2025. ISSN 2045-2322. doi: 10.1038/s41598-025-14285-5. URL http://dx.doi.org/10.1038/s41598-025-14285-5.

David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. Graph neural networks with adaptive readouts. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=yts7fLpWY9G.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *ArXiv*, abs/1805.11973, 2018. URL https://api.semanticscholar.org/CorpusID:44100802.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2022.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=5fybcQZ0g4.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 11735–11764. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/15b780350b302a1bf9a3bd273f5c15a4-Paper-Conference.pdf.

Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2434–2444. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/grover19a.html.

Volker D. Hähnke, Sunghwan Kim, and Evan E Bolton. Pubchem chemical structure standardization. *Journal of Cheminformatics*, 10(36), 2018. doi: 10.1186/s13321-018-0293-8. URL https://link.springer.com/article/10.1186/s13321-018-0293-8.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL https://doi.org/10.5281/zenodo.2559634.

Chao Hu, Song Li, Chenxing Yang, Jun Chen, Yi Xiong, Guisheng Fan, Hao Liu, and Liang Hong. ScaffoldGVAE: scaffold generation and hopping of drug molecules via a variational autoencoder based on multi-view graph neural networks. *J. Cheminform.*, 15 (1):91, October 2023.

Wei Huang, Hanchen Wang, Dong Wen, Shaozhen Ma, Wenjie Zhang, and Xuemin Lin. Towards unsupervised training of matching-based graph edit distance solver via preference-aware GAN. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=59HMRbpQ25.

Yinan Huang, Haoyu Peter Wang, and Pan Li. What are good positional encodings for directed graphs? In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=s4Wm71LFK4.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2323–2332. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/jin18a.html.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Mohamed Amine Ketata, Nicholas Gao, Johanna Sommer, Tom Wollschläger, and Stephan Günnemann. Lift your molecules: Molecular graph generation in latent euclidean space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uNomADvF3s.

Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. Pubchem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 01 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv951. URL https://doi.org/10.1093/nar/gkv951.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv:1611.07308 [cs, stat]*, November 2016. URL http://arxiv.org/abs/1611.07308. arXiv: 1611.07308.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Daria Klimoszek, Małgorzata Jeleń, Małgorzata Dołowy, and Beata Morak-Młodawska. Study of the lipophilicity and admet parameters of new anticancer diquinothiazines with pharmacophore substituents. *Pharmaceuticals*, 17(6):725, Jun 2024. ISSN 1424-8247. doi: 10.3390/ph17060725. URL http://dx.doi.org/10.3390/ph17060725.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21618–21629. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b4fd1d2cb085390fbbadae65e07876a7-Paper.pdf.

Paul Krzakala, Junjie Yang, Rémi Flamary, Florence d'Alché Buc, Charlotte Laclau, and Matthieu Labeau. Any2graph: Deep end-to-end supervised graph prediction with an optimal transport loss. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Paul Krzakala, Gabriel Melo, Charlotte Laclau, Florence d'Alché Buc, and Rémi Flamary. The quest for the GRAph level autoencoder (GRALE). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=V8HhrBSsEn.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1945–1954. JMLR.org, 2017.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 01 1988. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1988.tb01721.x. URL https://doi.org/10.1111/j.2517-6161.1988.tb01721.x.

Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2023.110176. URL https://www.sciencedirect.com/science/article/pii/S1568494623001941.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. doi: 10.1109/TPAMI.2018.2858826.

Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X. Liu, Chunming Wu, and Shouling Ji. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(2):799–813, 2023. doi: 10.1109/TNNLS.2021.3102234.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231749761.

Omar Mahmood, Elman Mansimov, Richard Bonneau, and Kyunghyun Cho. Masked graph modeling for molecule generation. *Nature Communications*, 12(1), May 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23415-2. URL http://dx.doi.org/10.1038/s41467-021-23415-2.

Kerry N. McPhedran, Rajesh Seth, and Ken G. Drouillard. Hydrophobic organic compound (hoc) partitioning behaviour to municipal wastewater colloidal organic carbon. *Water Research*, 47(7):2222–2230, 2013. ISSN 0043-1354. doi: https://doi.org/10.1016/j.watres.2013.01.039. URL https://www.sciencedirect.com/science/article/pii/S0043135413000699.

Anna Mozrzymas. On the hydrophobic chains effect on critical micelle concentration of cationic gemini surfactants using molecular connectivity indices. *Monatshefte für Chemie - Chemical Monthly*, 151(4):525–531, April 2020. ISSN 1434-4475. doi: 10.1007/s00706-020-02581-x. URL http://dx.doi.org/10.1007/s00706-020-02581-x.

Luis Müller and Christopher Morris. Aligning transformers with weisfeiler-leman. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=HhbqHBBrfZ.

Trieu Nguyen and Aleksandra Karolak. Transformer graph variational autoencoder for generative molecular design. *Biophysical Journal*, 124(22):3867–3875, 2025. ISSN 0006-3495. doi: https://doi.org/10.1016/j.bpj.2025.01.022. URL https://www.sciencedirect.com/science/article/pii/S0006349525000359.

Giannis Nikolentzos, Dimitrios Kelesis, and Michalis Vazirgiannis. On the theoretical expressive power of graph transformers for solving graph problems. *Neural Networks*, 194: 108112, 2026. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2025.108112. URL https://www.sciencedirect.com/science/article/pii/S089360802500992X.

Toshiki Ochiai, Tensei Inukai, Manato Akiyama, Kairi Furui, Masahito Ohue, Nobuaki Matsumori, Shinsuke Inuki, Motonari Uesugi, Toshiaki Sunazuka, Kazuya Kikuchi, Hideaki Kakeya, and Yasubumi Sakakibara. Variational autoencoder-based chemical latent space for large molecular structures with 3d complexity. *Communications Chemistry*, 6(1), November 2023. ISSN 2399-3669. doi: 10.1038/s42004-023-01054-6. URL http://dx.doi.org/10.1038/s42004-023-01054-6.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Frechet chemnet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9):1736–1741, 08 2018. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00234. URL https://doi.org/10.1021/acs.jcim.8b00234.

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), August 2014. ISSN 2052-4463. doi: 10.1038/sdata.2014.22. URL http://dx.doi.org/10.1038/sdata.2014.22.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Davide Rigoni, Nicolò Navarin, and Alessandro Sperduti. Rgcvae: relational graph conditioned variational autoencoder for molecule design. *Machine Learning*, 114(2), January 2025. ISSN 1573-0565. doi: 10.1007/s10994-024-06638-4. URL http://dx.doi.org/10.1007/s10994-024-06638-4.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12559–12571. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/94aef38441efa3380a3bed3faf1f9d5d-Paper.pdf.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL https://arxiv.org/abs/2602.12233.

Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 11 2012. ISSN 1549-9596. doi: 10.1021/ci300415d. URL https://doi.org/10.1021/ci300415d.

Marco Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v035i03.

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*, 2020. URL https://openreview.net/forum?id=S1esMkHYPr.

Martin Simonovsky and Nikos Komodakis. *GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders*, pp. 412–422. Springer International Publishing, 2018. ISBN 9783030014186. doi: 10.1007/978-3-030-01418-6_41. URL http://dx.doi.org/10.1007/978-3-030-01418-6_41.

R. Smith and C. Tanford. Hydrophobicity of long chain n-alkyl carboxylic acids, as measured by their distribution between heptane and aqueous solutions. *Proceedings of the National Academy of Sciences of the United States of America*, 70(2):289–293, February 1973. ISSN 0027-8424. doi: 10.1073/pnas.70.2.289.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

Anna Tsantili-Kakoulidou and Vassilis Demopoulos. Drug-like properties and fraction lipophilicity index as a combined metric. *ADMET and DMPK*, 9(3):177–190, October 2021. ISSN 1848-7718. doi: 10.5599/admet.1022. URL http://dx.doi.org/10.5599/admet.1022.

Hadi Vafaii, Dekel Galor, and Jacob L. Yates. Poisson variational autoencoder. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 44871–44906. Curran Associates, Inc., 2024. doi: 10.52202/079017-1426. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4f3cb9576dc99d62b80726690453716f-Paper-Conference.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations*, 2017.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *ICLR*. OpenReview.net, 2023. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2023.html#VignacKSWCF23.

Runzhong Wang, Ziao Guo, Wenzheng Pan, Jiale Ma, Yikai Zhang, Nan Yang, Qi Liu, Longxuan Wei, Hanxue Zhang, Chang Liu, Zetian Jiang, Xiaokang Yang, and Junchi Yan. Pygmtools: A python graph matching toolkit. *Journal of Machine Learning Research*, 25 (33):1–7, 2024. URL http://jmlr.org/papers/v25/23-0572.html.

Robin Winter, Frank Noe, and Djork-Arné Clevert. Permutation-invariant variational autoencoder for graph-level representation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9559–9573. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f3d7d38d24b740c95da2b03dc3a2333-Paper.pdf.

Zhang Xie, Zheng Li, Gang Lou, Qing Liang, Jiang-Xing Chen, Jianlong Kou, and Gui-Na Wei. Interface water-induced hydrophobic carbon chain unfolding in water. *Communications in Theoretical Physics*, 73(5):055602, mar 2021. doi: 10.1088/1572-9494/abe84e. URL https://doi.org/10.1088/1572-9494/abe84e.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Rong Yang, Xiaojuan Lai, Qiying Li, Xi Ding, Lei Wang, Xin Wen, and Yan Guo. Effect of hydrophobic monomers with different carbon chains on the structure–activity relationship of associating polyacrylamides. *Journal of Polymer Research*, 31(242), Jul 2024. doi: 10.21203/rs.3.rs-4393619/v1. URL http://dx.doi.org/10.21203/rs.3.rs-4393619/v1.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 4805–4815, Red Hook, NY, USA, 2018. Curran Associates Inc.

Kiarash Zahirnia, Oliver Schulte, Parmis Naddaf, and Ke Li. Micro and macro level graph modeling for graph variational auto-encoders. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30347–30361. Curran Associates, Inc., 2022. doi: 10.52202/068431-2200. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c400474e8a36d0812fdee52739288b12-Paper-Conference.pdf.

Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs, 2019. URL https://arxiv.org/abs/1904.11088.

Wenhao Zhu, Tianyu Wen, Guojie Song, Liang Wang, and Bo Zheng. On structural expressive power of graph transformers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 3628–3637, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599451. URL https://doi.org/10.1145/3580305.3599451.

A Design choices

A.1 Graph-level or Node-level

To motivate our use of a graph-level encoder (despite leveraging a canonical ordering), we implemented a node-level version of our VAE, where the latent space consists of per-atom embeddings. The architecture is the same as the main paper, but the readout and readin are removed. While this variant achieves higher reconstruction performance (99.94% accuracy, 0.004 edit distance), its generative performance collapses: only 17% of sampled molecules are valid, compared to 68% for our graph-level VAE.

This outcome is consistent with prior work showing that node-wise generative models struggle to enforce global chemical constraints. GraphVAE reports at best 57% validity on QM9 (Simonovsky & Komodakis, 2018) (with the exception of a NoGM reaching 81%, but due to a collapse on a mode, leading to only 24% uniqueness). MolGAN (Cao & Kipf, 2018) also report that no baseline exceeds 57% on QM9 (with the exception of NoGM), and uses an extensive reinforcement learning procedure specifically to improve validity. More recent analyses demonstrate that node-level likelihoods fundamentally fail to capture global structural dependencies, with graph-level modeling significantly improving validity: "It is known from previous work that node-level representations, while useful for many applications, do not capture enough graph structure to generate realistic graphs" (Zahirnia et al., 2022). In particular, the basic setting of sampling each node independently prevents sampling a globally coherent molecule with high probability.

Node-level embeddings also break down outside the generation task. Because their dimensionality depends on the number of atoms, they live in different latent spaces for different molecules, so there is no canonical way to define distances or interpolate between them. This makes latent-space editing, smooth interpolation, and property-guided traversal ill-posed: one cannot move "between" molecules when their latent representations have incompatible shapes. In the same way, optimization algorithms such as CMA-ES work on fixed-sized spaces. Graph-level embeddings avoid this entirely by mapping every molecule into a single fixed-dimensional latent space where distances, directions, and edits are well defined, regardless of size.

Table 6: Comparison between graph-level and node-level latent representations.

Model	Reconstruction Acc. (↑)	Edit Dist. (↓)	Validity (%, ↑)
Graph-level VAE	99.50	0.04	67
Node-level VAE	99.94	0.004	17

A.2 Permutation-invariant encoder

Since our loss function is permutation-sensitive, it could be assumed that our model would benefit from a permutation-sensitive encoder. We justify here the choice for a permutation-invariant encoder.

In order to test that theory, we experiment on PubChem16 replacing the RW- and Laplacian-positional encodings (both permutation equivariant) with a sinusoidal positional encoding (similar to input of the decoder, permutation-sensitive). The encoder, and therefore latent representation are thus a sequence-model, where different node orderings can result in different representations. Our results are presented in table 7.

Table 7: Comparison between a permutation-invariant and a permutation-sensitive encoder (PubChem16).

Model	Reconstruction Acc. (↑)	Edit Dist. (↓)
Default	99.50	0.04
Permutation-Sensitive	99.95	0.001

We observe that injecting the node-ordering information makes reconstruction easier for our model. We tested the robustness of our permutation-sensitive variant by evaluating its reconstruction capacity on molecules obtained by random node permutations of the input. None of the reconstructed graphs matched their input counterparts, showing that the learned representation is not only not permutation-equivariant, but more generally incapable of properly encoding any graph that is not provided in the canonical ordering. In particular, 65% of the reconstructed graphs gave invalid molecules (out of a sample of 2048 molecules).

However, this gain is conditional on the ordering convention being reliably available. In the case where the ordering exists as an external, callable function, this is not an issue, since input can be pre-treated to be in the right ordering. On the other hand, if the ordering is a property of the training dataset but not otherwise accessible (as is the case with QM9 and PubChem, which both use closed-source canonization) the permutation-invariant model can still train and generalize (since the canonical ordering is only required for loss computation and can be ignored at inference), but the permutation-sensitive variant cannot function properly.

B Statistics of the datasets

B.1 Statistics

Table 8: Statistics of the used datasets.

Name	# Graph	Avg # nodes	Avg # edges	Max Size	Node types
QM9NoHydro	130k	8.79	9.4	9	4
PubChem16	12.89M	13.87	14.19	16	31
PubChem32	74.34M	21.75	21.15	32	31
Coloring	320k	12.51	24.93	20	4
Asia (BN)	200k	8	8.05	8	8

B.2 Canonizations

All datasets below are publicly provided in canonized form, and were used as-is.

QM9 (Ramakrishnan et al., 2014) is a subset of a larger database, GDB-17. GDB-17[1] (Ruddigkeit et al., 2012) is stored as canonized SMILES strings.

The canonization of GDB-17's SMILES representations is documented in the primary enumeration paper (Ruddigkeit et al., 2012). The methods section states that all software was implemented using JChem libraries from the ChemAxon proprietary software, and that each generated molecule was converted to a unique SMILES string specifically to detect and remove duplicates before storage. Note that the paper uses the term "unique" rather than "canonical" when describing this step.

PubChem's canonization applies directly to atom ordering rather than only to an upstream SMILES string. PubChem's standardization pipeline invokes OpenEye's OECanonicalOrderAtoms as part of its aromaticity and kekulized form standardization[2] (Hähnke et al., 2018). The resulting canonical SMILES strings are parsed with RDKit's Chem.MolFromSmiles function, which preserves the input order rather than recomputing its own canonization, so the canonical ordering established by OpenEye is retained in the molecules we construct.

Neither OpenEye's OEChem nor ChemAxon's JChem are open source, so we do not have access to the specific canonization algorithms in either case. This is not an obstacle in our case thanks to the encoder being permutation invariant. As long as the training dataset's ordering remains consistent (as it is), the model functions well on any new molecule, regardless of ordering, so long as no permutation-sensitive loss is required.

[1] https://gdb.unibe.ch/downloads/

[2] https://docs.eyesopen.com/toolkits/python/oechemtk/OEChemFunctions/OECanonicalOrderAtoms.html

C Hyperparameters and training details

We provide in table 9 the hyperparameters used to train our models. Our models use a half-sinusoidal warmup curve for beta (KL-divergence) and a cosine annealing scheduler with warmup and restarts.

Table 9: Hyperparameters used for best models. All experiments were conducted on AMD MI210 GPUs.

Dataset	QM9	PubChem16	PubChem32	Coloring	Asia (BN)
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	2e-4	1e-4	1e-4	1e-4	1e-4
Batch size	64	64	256	64	64
Latent dimension	64	256	512	128	64
Encoder hidden size	128	512	512	256	64
Encoder input size	128	128	128	128	64
Graph transformer layers	2	5	7	3	1
Transformer heads	4	4	8	4	1
Beta (VAE)	1e-5	1e-4	1e-5	1e-4	1e-3
LR cycle length	200	200	800	300	30
Beta cycle length	400	200	800	200	30
Ratio negative edges	4	4	4	4	4
Edge loss weight	1.25	1.25	1.25	1	1
Train split	0.8	0.9736	0.9978	0.9375	0.9
Valid. split	0.1	0.003	0.0011	0.03125	0.05
Test. split	0.1	0.0234	0.0011	0.03125	0.05
GPUs (training)	1	1	2	1	1
GPUs (inference)	1	1	1	1	1

D Performance on molecules in aromatic form

Table 10: Percentage of perfectly reconstructed graphs and edit distance. Results from PIGVAE are reported from (Krzakala et al., 2025).

Dataset	Metric	GRALE	PIGVAE*	GraViti (AE)	GraViti (VAE)
QM9NoHydro	Edit Dist.	**0.05**	N/A	0.10	0.10
	Accuracy	97.2	N/A	97.5	**97.7**
PubChem16	Edit Dist.	0.06	1.69*	**0.05**	**0.06**
	Accuracy	96.8	41.0*	**99.3**	**99.3**
PubChem32	Edit Dist.	0.70	2.53*	0.17	0.18
	Accuracy	73.3	24.9*	98.0	97.3

E Loss function

E.1 Reconstruction Loss

The reconstruction loss is defined over nodes, edges and predicted size. Since both nodes and edges take discrete values (atom and bond types), we treat our graph prediction task as multiple classification tasks, one per node and edge. We use per-node and per-edge cross-entropy:

$$\mathcal{L}_{\texttt{CE}}(p, y) = -\sum_{i=1}^{C} y_i \log(p_i)$$

with p_i the predicted probability of class i and y a one-hot vector for true class. Despite facing a highly imbalanced distribution of classes, with some elements appearing orders of magnitude more often than others, we found that cross-entropy with inverse-frequency weights and focal loss (Lin et al., 2020) consistently underperformed naive cross-entropy. For a graph with N nodes, with y^n (resp. y^{n_1,n_2}) the one-hot encoding of classes for node n (resp. edge n_1, n_2) and p^n (resp. p^{n_1,n_2}) the predicted class probabilities for this node (resp. edge), we therefore have:

$$\mathcal{L}_{\texttt{node}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{\texttt{CE}}(p^i, y^i) \text{ and } \mathcal{L}_{\texttt{edge}} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathcal{L}_{\texttt{CE}}(p^{i,j}, y^{i,j})$$

In order to avoid the overwhelmingly frequent "no-edge" class from hurting the training, we actually sample a number of "no-edge" class equal to rN_e, with r a hyperparameter and N_e the number of actual edges.

A natural concern is that this loss is applied in the input node order, which makes it highly sensitive to permutations. As we show in the section 5, this inductive bias is beneficial for dataset with a consistent canonical ordering: enforcing a consistent ordering improves reconstruction fidelity and does not hinder the model's ability to represent graphs up to isomorphism. Finally, the error $\mathcal{L}_{\text{size}}$ on the predicted log-size $\hat{s}$ is simply the smooth $L1$ error with the ground truth log-size s. Our final reconstruction loss is a combination of those three losses:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{node}} + \eta\mathcal{L}_{\text{edge}} + \mathcal{L}_{\text{size}} \text{ with } \eta \text{ a hyperparameter.}$$

E.2 Variational loss

For the latent variables we assume a standard normal prior $\mathcal{N}(0, I)$. Given encoder outputs z_μ and z_σ, the variational term corresponds to the KL divergence between the approximate posterior $q(z \mid x) = \mathcal{N}(z_\mu, \text{diag}(z_\sigma^2))$ and the prior:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}\big(\mathcal{N}(z_\mu, \text{diag}(z_\sigma^2)) \,\|\, \mathcal{N}(0, I)\big) = -\frac{1}{2}\sum_{i=1}^{d}\left(1 + \log z_{\sigma,i}^2 - z_{\mu,i}^2 - z_{\sigma,i}^2\right).$$

The final training objective is therefore:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \beta\,\mathcal{L}_{\text{KL}},$$

where β optionally controls the strength of the variational term. In practice, β follows a warmup schedule, sinusoidally increasing until plateauing at its maximum value.

Extra loss terms for molecules In the case of molecules in aromatic form, the graph representation (atom-types/bond-type) does not contain all of the relevant information when using the aromatic representation. To alleviate this ambiguity, we also predict the formal charge and number of bonded hydrogen atoms for all atoms that are part of an aromatic ring using two extra prediction heads.

F Order enforced on COLORING

To demonstrate the importance of ordering for our model, we also run an ablation study on the COLORING dataset (Krzakala et al., 2024). This dataset contains general graphs with only node classes, and such that two nodes of the same color cannot be connected. As such, there is no inherent ordering of the node. To enforce ordering on a Coloring graph $G = (V, E)$, we replace the arbitrary node order with a deterministic WL-based order. Let c_v the sampled color label of node v. We initialize the WL labels as $h_v^{(0)} = c_v$. For $t = 1, 2, 3$, each node receives the signature

$$s_v^{(t)} = \left(h_v^{(t-1)}, sort\left(h_u^{(t-1)} : u \in N\left(v\right) \right) \right),$$

where $N(v)$ is the set of graph neighbors of v. The set of signatures is then compressed into integer labels $h_v^{(t)}$ by sorting the unique signatures lexicographically and assigning consecutive IDs. Thus, after three iterations, $h_v^{(3)}$ encodes the color and topology of the node's local neighborhood up to radius three, as distinguished by 1-WL refinement.

The final node order is obtained by sorting nodes lexicographically using the tuple

$$\left(h_v^{(3)}, d_r\left(v\right), d_a\left(v\right), deg\left(v\right), y_v, x_v \right),$$

where $d_r(v), d_a(v)$ are the unweighted shortest-path distances from selected root and anchor nodes, $deg(v)$ is the graph degree, and (x_v, y_v) is the geometric position of the node.

The root r is chosen as the smallest structurally identifiable node after WL refinement, i.e. the node minimizing

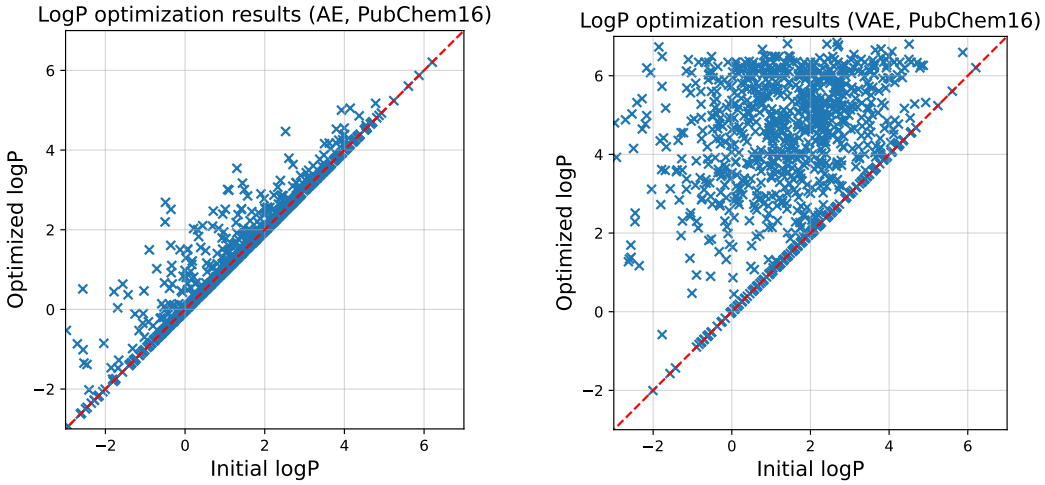
$$\left(h_v^{(3)}, deg\left(v\right), y_v, x_v \right).$$

The anchor a is chosen far from the root to further break symmetries by adding a second axis, as the node maximizing

$$\left(d_r\left(v\right), deg\left(v\right), -h_v^{(3)} \right).$$

G Latent optimization of LogP

We show below some extra examples for optimizing logP as described in section 5.



(a) AE optimization

(b) VAE optimization

Figure 4: Comparison of AE and VAE optimization performance on 128 random molecules. The structure in the VAE latent allows significantly better improvement.

CMA-ES is obtained from the `cma` package [3] run for 100 steps, with `sigma0` set to 0.25, 50 iterations, and a population size of 16.

[3] https://pypi.org/project/cma/

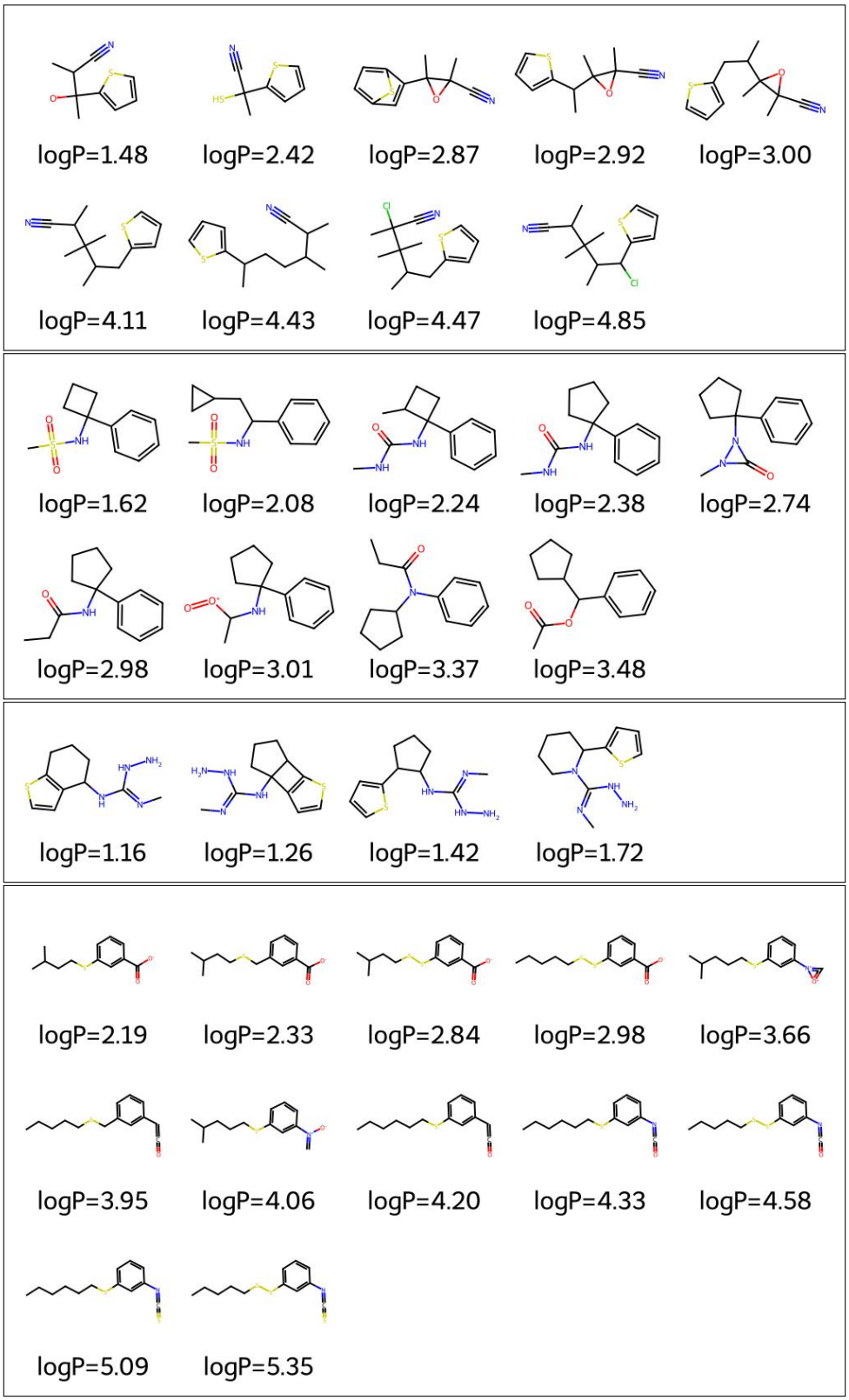


Figure 5: Extra logP optimization plots (PubChem16).

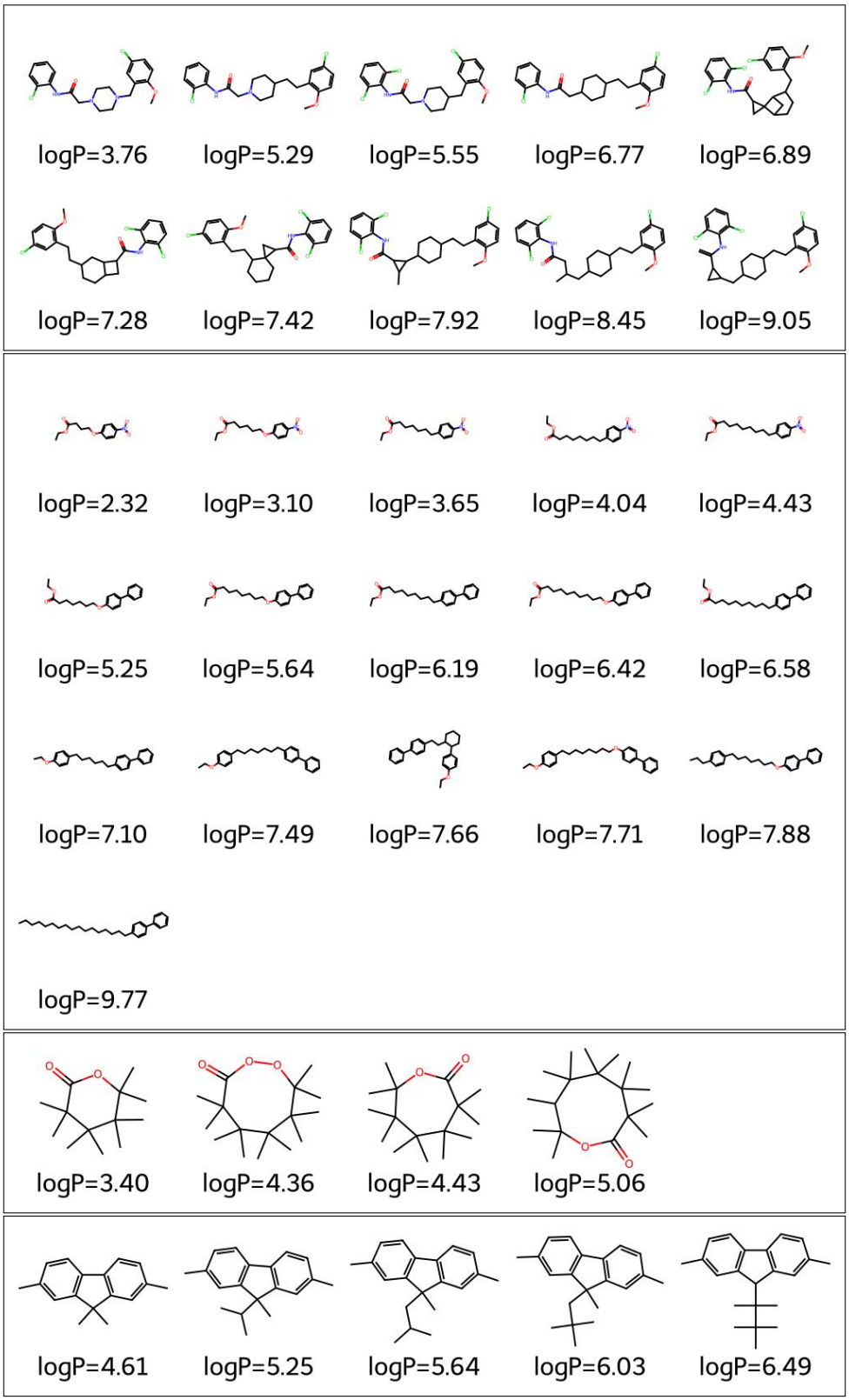


Figure 6: Extra logP optimization plots (PubChem32).

H Downstream Regression

To evaluate the chemical expressiveness and structural organization of our latent space, we test whether meaningful molecular information can be recovered directly from the embeddings. We embed the QM9 molecules using a model trained on PubChem. Using only the latent vectors as input, we train simple regressors to predict three key molecular properties: dipole moment, HOMO–LUMO gap, and U_0 the 0 K internal energy. These properties probe complementary aspects of molecular structure and chemical behavior. The results are reported in table 11.

Table 11: Downstream task: regression on molecular properties. MSE is reported (mean / standard dev. over 5 folds). Models are trained either on PubChem16 or 32, as indicated.

Model	Dipole moment	Homo-Lumo	Energy at 0 Kelvin
GRALE (PC16)	0.4029 ± 0.0071	0.1230 ± 0.0026	$\mathbf{0.0033 \pm 0.0010}$
GRALE (PC32)	0.3677 ± 0.0105	0.1484 ± 0.0033	0.0204 ± 0.0024
GraViti (VAE, PC16)	0.3219 ± 0.0042	0.0975 ± 0.0032	0.0056 ± 0.0001
GraViti (AE, PC16)	0.3322 ± 0.0016	0.0929 ± 0.0013	0.0035 ± 0.0002
GraViti (VAE, PC32)	$\mathbf{0.2937 \pm 0.0012}$	0.0861 ± 0.0030	0.0106 ± 0.0053
GraViti (AE, PC32)	0.3040 ± 0.0096	$\mathbf{0.0853 \pm 0.0017}$	0.0034 ± 0.0002

To further evaluate whether the latent space captures graph-level information beyond molecular structure, we consider a regression task on Bayesian networks. We use the BN dataset introduced in D-VAE (Zhang et al., 2019), consisting of 200,000 8-node Bayesian networks generated with the bnlearn package (Scutari, 2010). Each graph is assigned a Bayesian Information Criterion (BIC) score measuring how well the network structure fits the Asia dataset (Lauritzen & Spiegelhalter, 1988). We first train small models for reconstruction (where all models and baseline reach 100% accuracy), and using the encoder embeddings we train MLP regressors to predict the BIC score. The results are reported in table 12.

Table 12: Downstream task: regression on BIC score. RMSE and Pearson correlation coefficient are reported (mean and standard deviation). Models are trained on BN dataset.

Model	RMSE	Pearson's r
GRALE	0.0247 ± 0.0052	$\mathbf{0.9997 \pm 0.0001}$
GraViti (VAE)	0.0457 ± 0.0021	0.9990 ± 0.0001
GraViti (AE)	$\mathbf{0.0229 \pm 0.0038}$	$\mathbf{0.9997 \pm 0.0001}$

I Denoising

Autoencoders are effective tools for denoising structured data. To evaluate this capability, we sample 1024 graphs from PubChem16 and introduce controlled noise by randomly altering node and edge labels, which can include the creation or deletion of edges. We then forward these corrupted graphs through our model. Figure 8 reports the validity of molecules after the noising procedure and after reconstruction by the autoencoder. The results show a clear improvement in molecular validity once the noisy inputs are passed through the model, confirming its denoising ability, even with large numbers of edits.

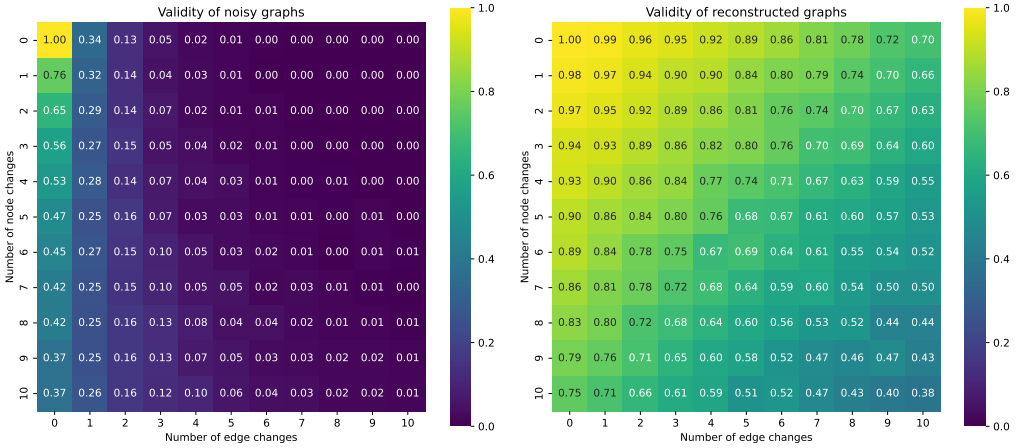


Figure 7: Denoising results: each cell reports the fraction of valid molecules (1024 molecules from PubChem16) after applying a number of random node and edge edits (left) and after reconstruction by the model (right). The AE consistently restores chemically valid structures even under heavy perturbations.

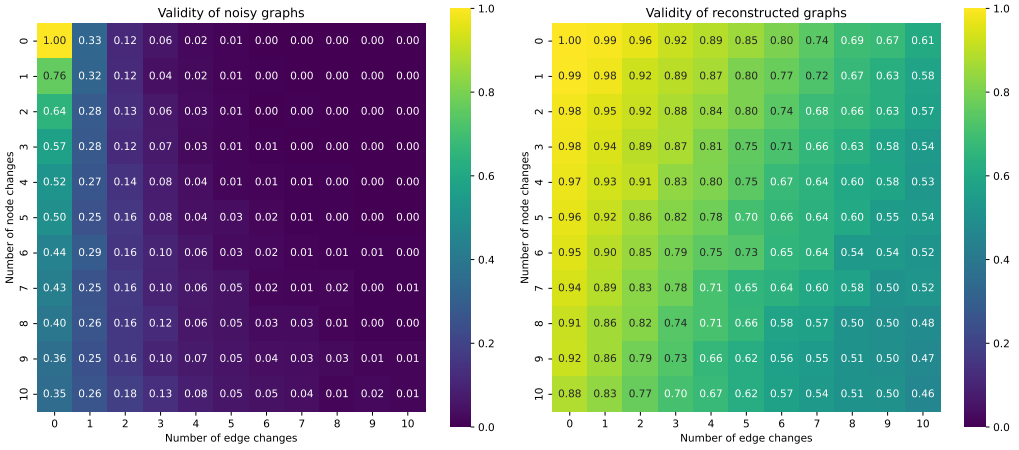
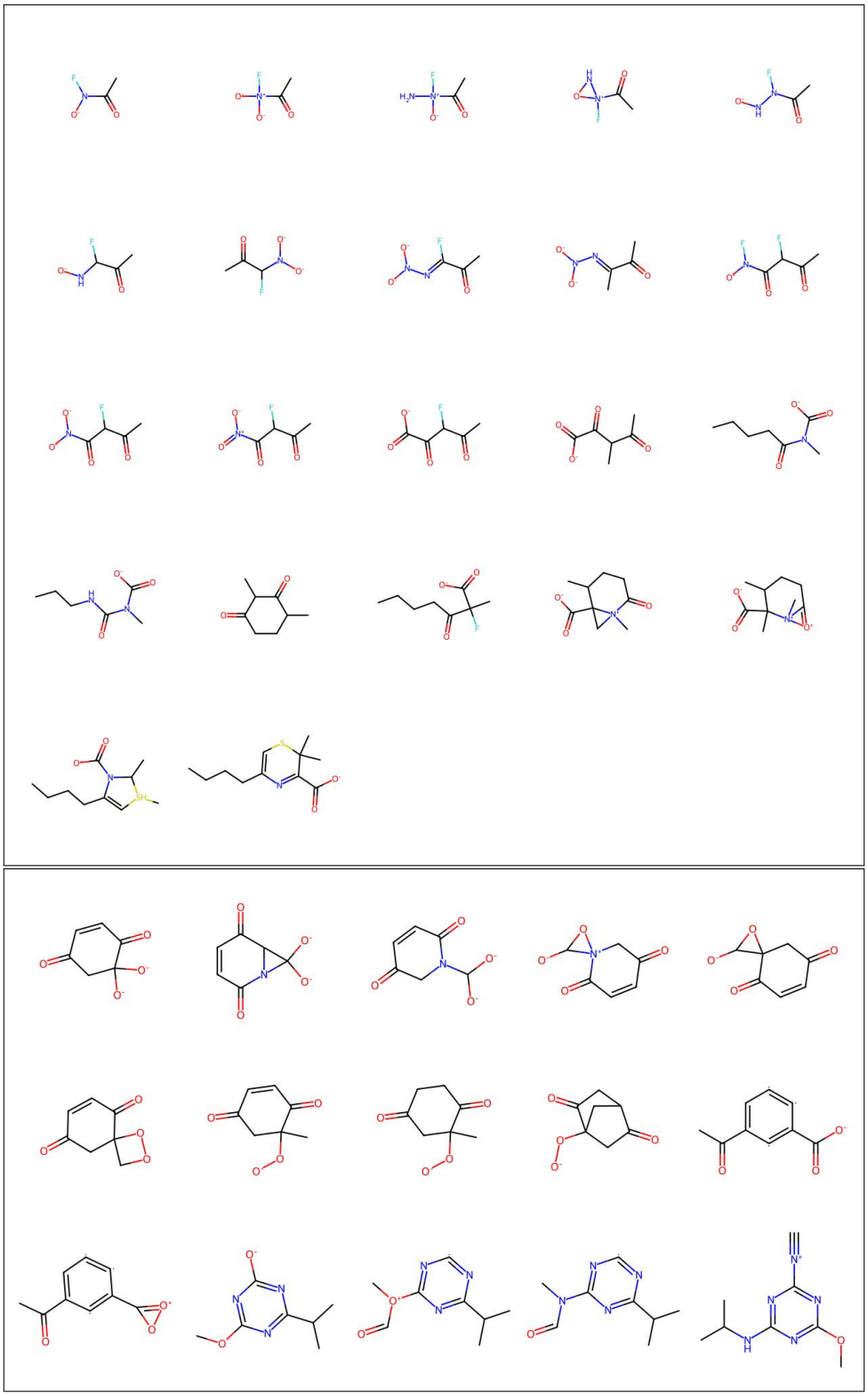


Figure 8: Denoising results: each cell reports the fraction of valid molecules (1024 molecules from PubChem16) after applying a number of random node and edge edits (left) and after reconstruction by the model (right). The VAE consistently restores chemically valid structures even under heavy perturbations.

Fig. 8 compares the denoising abilities of AE (top) and VAE (bottom), showing a slight advantage for the variational model.

J Molecule interpolation

We show below some extra examples for molecule interpolation described in Section 5.



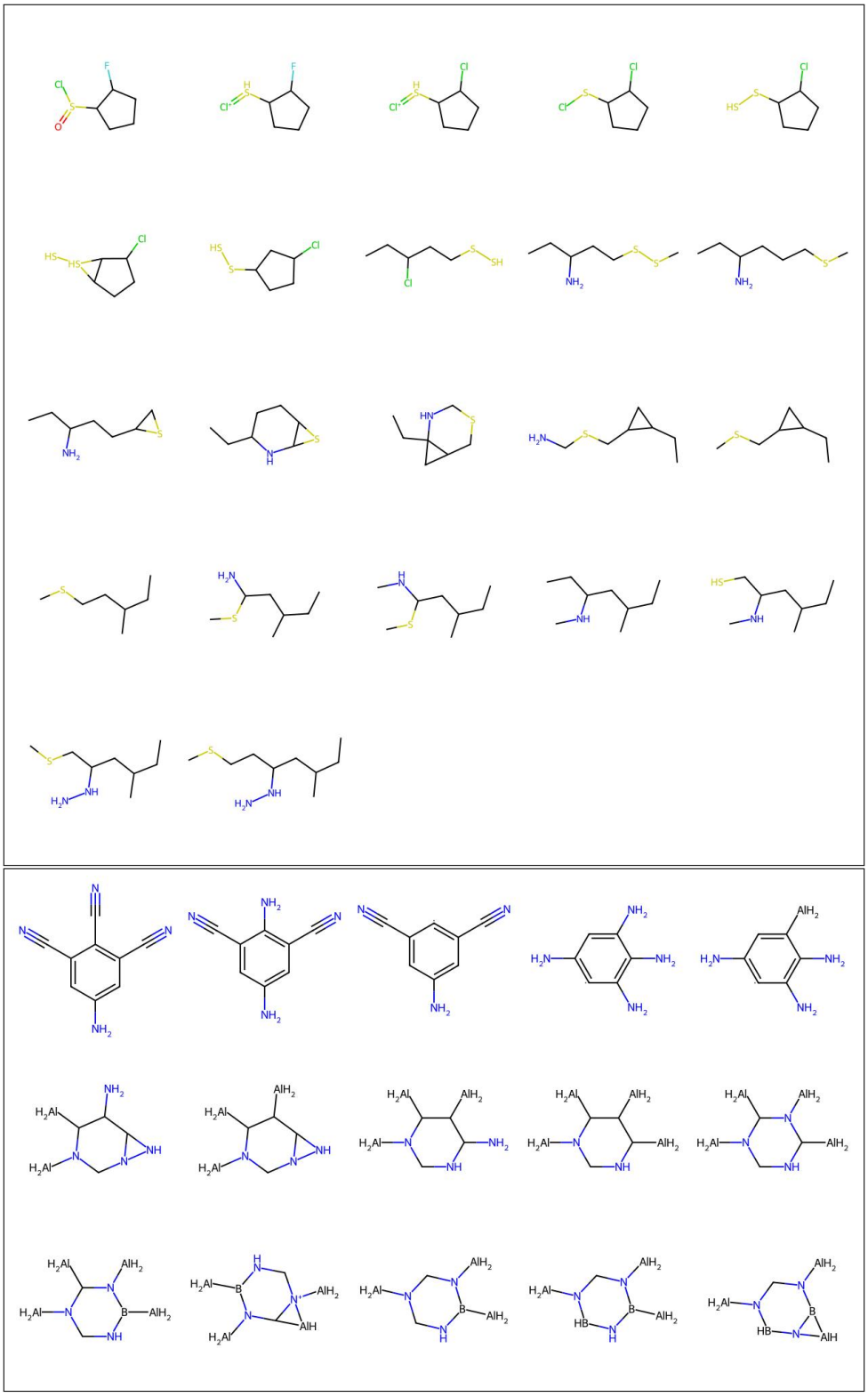


Figure 9: Extra molecule interpolation plots (PC16)

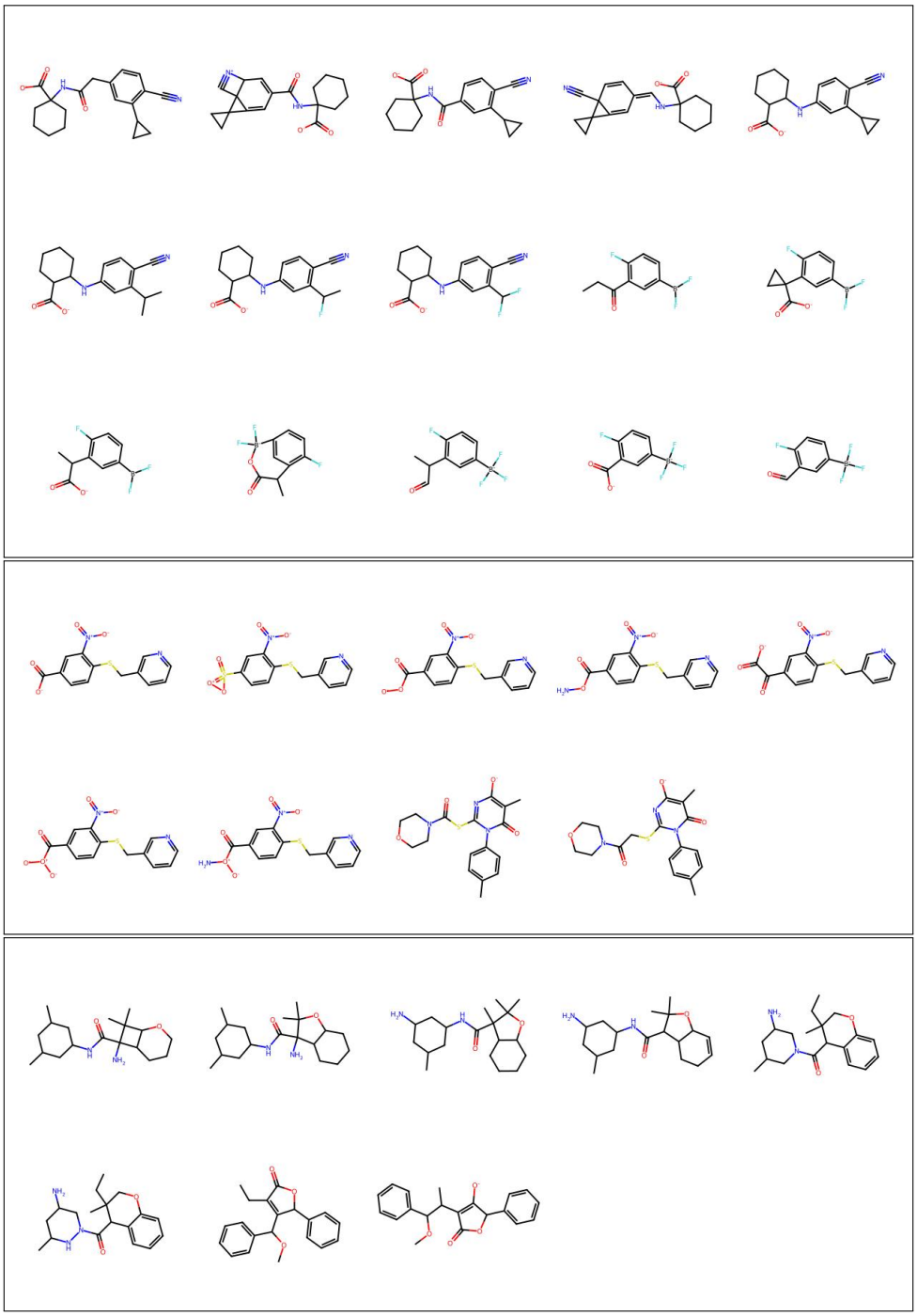


Figure 10: Extra molecule interpolation plots (PC32)

K Generated molecules samples

Figure 11 presents examples of molecules generated by our PubChem-trained model. Latent vectors are sampled from the standard normal prior and decoded into molecular graphs. We report only valid generations, that is, molecules for which all chemical constraints are satisfied.

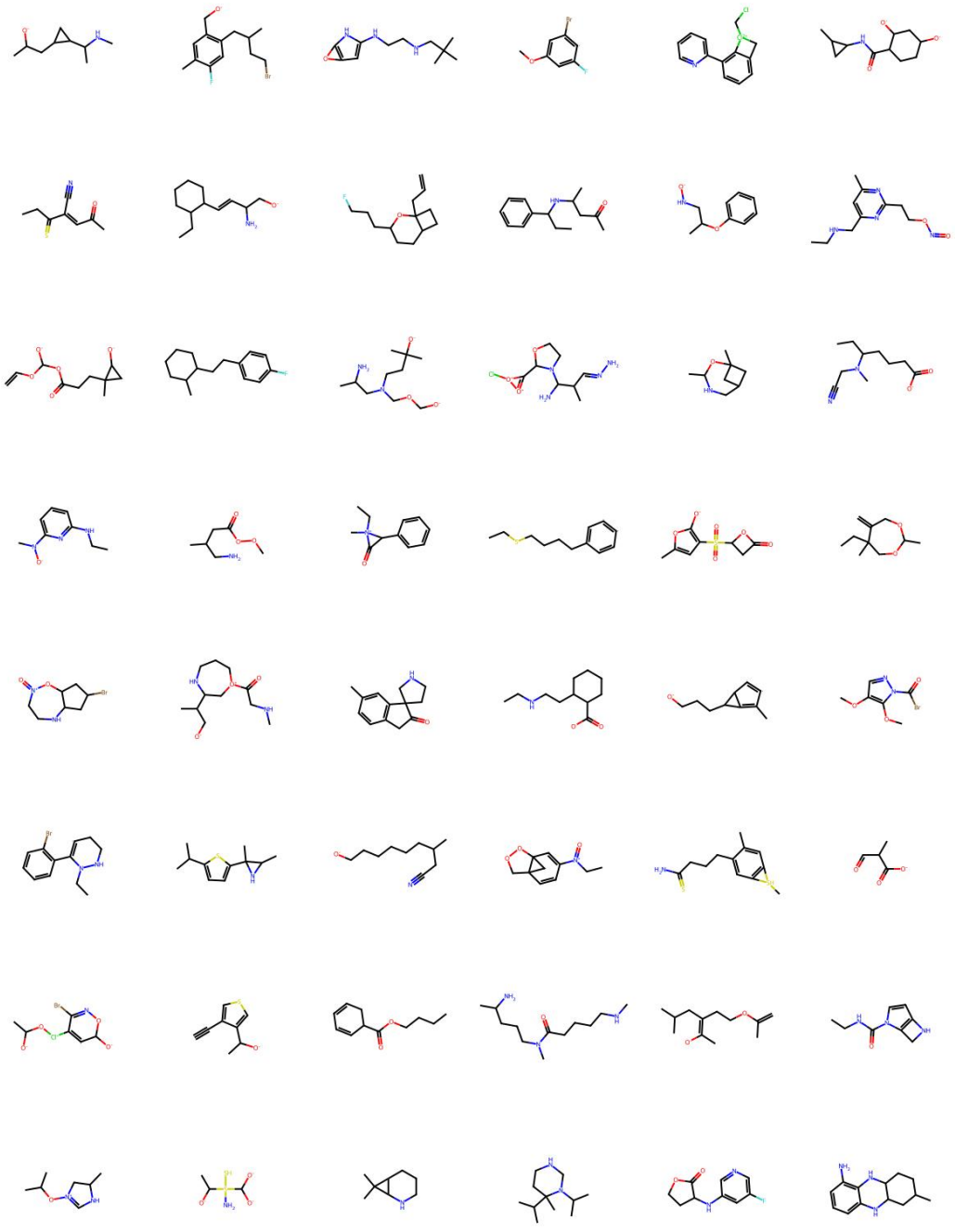


Figure 11: Samples of generated molecules (PubChem16)

L Latent generation

To further study the structure of our latent, we ran a simple flow matching algorithm to generate latents. The flow model is a simple MLP with FilM layers for time. Results are reported in table 13.

Table 13: Generation performance of a flow matching trained to reconstruct latent representations of graphs. The model for each dataset is the variational autoencoder trained on kekulized molecules. We provide results for different numbers of generation steps. Statistics are computed on batches of 1024 molecules, 10 trials. Baselines are reported directly from (Roos et al., 2026).

Dataset/Model	Steps	Valid. (%)	Uniq. (%)	Novelty (%)	FCD
QM9/GraViti	VAE	93.2	98.5	N/A	2.88
QM9/GraViti	1	94.1	99.2	N/A	1.66
QM9/CFM ECLD	1	95.8	89.2	N/A	2.16
QM9/CFM CSD	1	90.0	91.8	N/A	2.14
QM9/GraViti	2	95.6	99.8	N/A	0.75
QM9/CFM ECLD	2	97.0	96.6	N/A	0.52
QM9/CFM CSD	2	91.1	97.6	N/A	0.49
PubChem16/GraViti	VAE	68.1	98.5	98.6	4.76
PubChem16/GraViti	1	89.7	98.6	87.2	5.39
PubChem16/GraViti	2	85.9	100.0	97.1	2.74