QM9 Molecular Skeleton Generation

Exploiting a Simpler Yet Powerful Denoiser Architecture Comparable to Graph Transformers

https://github.com/gusah1104/networkML_project_QM9

Berta Céspedes Sarrias Hyunmo Kang

***Abstract*—This study evaluates a simplified denoising process as an alternative to the DiGress model's graph transformer (GT) based denoiser [1]. Despite the GT's effectiveness, it suffers from limited interpretability and scalability. Our method utilizes a message-passing algorithm to aggregate node features and employs a multi-layer perceptron to predict edge probabilities. We find the node aggregating layer is important for performance. Furthermore, our experiments on the QM9 dataset for molecular skeleton generation reveal that our simplified denoiser achieves comparable results to those of the graph transformer-based denoiser, when enhanced with additional spectral and cycle features. Despite not critical in this task, modeling diffusion with marginal noise in the forward process results in slightly better performance.**

I. Introduction

Graph-based generative models has become an active area of research in various domains. Particularly, they have attracted considerable attention in the biomedical field, in areas such as drug discovery [2]–[6], and medical imaging [7], [8].

A. State-of-the-art

Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are widely used for modeling complex data. GANs use a generator to create synthetic graphs and a discriminator to distinguish them from real data, while VAEs use an encoder to map data into a latent space and a decoder to reconstruct it. Both face issues with scalability, oversimplification, and training instability.

GraphRNN models graphs sequentially using two RNNs: one for nodes and another for edges. However, GraphRNN can struggle with long-term dependencies, leading to potential issues in generating large and complex graphs. It may also face difficulties in capturing intricate graph structures due to its sequential nature.

Recent advances have highlighted diffusion models, which address the limitations of GANs, VAEs, and GraphRNN. Diffusion models add noise to data in the forward process and remove it in the reverse process, generating accurate new samples. These models offer superior scalability, performance, and training stability, effectively capturing complex data patterns [9], [10].

Diffusion models have achieved success in graph generation, which is the task at hand in this project. There has been promising results in areas such as molecule generation [5], social network analysis [11], and recommendation systems [12].

Continuous diffusion models have been implemented, where graphs are embedded in a continuous space, and Gaussian noise is added to the node features and graph adjacency matrix. This decreases graph sparsity, and capturing the structural properties of the data, such as connectivity, becomes more challenging in the denoising stage.

B. Discrete diffusion for graph:

This project implements discrete diffusion following the DiGress implementation [1], [13], which overcomes the limitations of the continuous diffusion framework. Instead of Gaussian, the noise model is Markovian, where each successive graph state is derived independently. Iteratively, noise is added to an initial graph G^0 until reaching a fully corrupted graph G^T, thus obtaining a set of graphs $\{G^0, \ldots, G^T\}$. Our implementation diffuses each edge separately, according the the transition matrix:

$$\left[\mathbf{Q}_E^t\right]_{ij} = q\left(e^t = j \mid e^{t-1} = i\right) \tag{1}$$

G^T is obtained by sampling from a categorical distribution defined by:

$$q\left(G^t \mid G^{t-1}\right) = \mathbf{E}^{t-1}\mathbf{Q}_E^t \quad \text{and} \quad q\left(G^t \mid G\right) = \mathbf{E}\overline{\boldsymbol{Q}}_E^t \tag{2}$$

where $\overline{\boldsymbol{Q}}_E^t = \boldsymbol{Q}_E^1 \ldots \boldsymbol{Q}_E^t$. Note that we can also improve sample quality by preserving the marginal distributions of edge types during diffusion, to maintain sparsity. Marginal transitions have shown to improve over uniform transitions experimentally. [1]. The definition of the transition matrix can be seen below.

$$\mathbf{Q}_E^t = \alpha^t \mathbf{I} + \beta^t \mathbf{1}\mathbf{m}_E' \tag{3}$$

where α^t follows the cosine schedule $\overline{\alpha}^t = \cos\left(0.5\pi(t/(T+s))/(1+s)\right)^2$ and transitions from 1 to 0 with time t, and is used as a coefficient to the identity matrix. Moreover, $\overline{\beta}^t = 1 - \overline{\alpha}^t$, thus transitioning from 0 to 1. Then, $\mathbf{1} \in \{1\}^a$, and $\mathbf{m}_E' \in \mathbb{R}^b$ is a transposed row vector of marginal distributions of edge types. Thus, the probability of transitioning from state i to state j, which in the case of unattributed graphs will be having an existing edge or not, is proportional to the marginal probability of one of these categories in the training set. This is significant because preserving the marginal distributions of node and edge types during diffusion allows graph characteristics such as connectivity and graph topology to be maintained.

As previously described, the second stage of the process leverages the denoising neural network, for which multiple implementations are available. In DiGress, a graph transformer network is trained to reverse the diffusion process. This method first aggregates node features using self-attention, while edges are processed using a Feature-wise Linear Modulation (FiLM) layer. Then, the architecture iteratively applies this graph transformer layer coupled with a Multi-layer Perceptron (MLP) layer. Despite being powerful, this method lacks interpretibility and scalibity. We propose a simpler denoiser architecture where node attributes are aggregated using a message passing algorithm, followed by an MLP layer to predict edge probability.

C. Problem Formulation:

In the evolving field of graph-based neural networks, achieving optimal peformance with simpler models remains essential. For this study, all node and edge features have been removed from a subset of the QM9 dataset. The aim lies on investigating the effectiveness of graph transformers and other architectures. The following questions are posed in the context of the unattributed QM9 dataset.

- Can similar performance to graph transformers be achieved using a simpler model?
- Which components of the denoiser -specifically, the node attribute layer or the edge predictions layer- are critical to achieve this performance?
- What additional features can be added to enhance the model's performance and expressivity?
- Does the injection of marginal noise in this simpler setting yield better results?

II. Methods

A. Dataset

The dataset employed is QM9 [14], a widely used benchmark for prediction and modeling of molecular properties and structure. A subset of 345 small organic molecules is utilized, where all graphs have 9 nodes. All node and edge features are removed for this study. The training set is composed of 70% of the data (250 graphs), while 30% (95 graphs) is separated for testing. 5 different seeds are used to ensure robustness.

B. Forward diffusion process - Noise injection:

Two different noise distributions are tested. Firstly, the model is trained adding uniform noise. Thus, the transition probability is [0.5,0.5], for existing and non-exisiting edges respectively. Secondly, marginal noise is calculated from the edge distribution in the training set, and the transition matrix was changed to approximately [0.7,0.3], depending on the choice of training set.

C. Backward diffusion process - Choice of denoising model:

Two pipelines are employed for the backward process.

- **Graph Transformers:** This approach improves upon the above mentioned, by utilizing edges directly, and thus avoiding the limitation of relying solely on node features. The implementation is as described by [1].

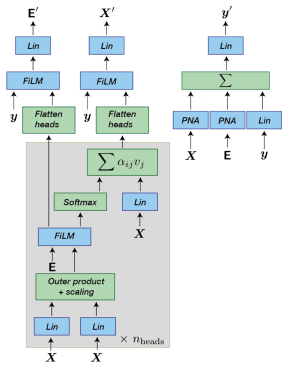

Fig. 1: Self attention module in graph transformer. Figure adopted from [1].

- **Simple Denoiser:** This model aggregates node features with a message passing algorithm, and then predicts edge probability based on the features of node pairs, using a multi-layer perceptron. The architectures compared are Graph Attention Network (GAT), which is deemed to be the most powerful, Graph Isomorphism Network (GIN), and Graph Convolutional Network (GCN).

D. Additional Features Added:

- **Cycles:** Message passing algorithm cannot capture cycles (closed loop paths that start and end at the same vertex without repeating any edges or vertices). For this reason, we added the number of k-cycles for each node prior to message passing. [1] For each node, we calculated its involvement in cycles ranging from three to five edges. Additionally, at the graph level, we computed the presence of cycles spanning from three to six edges.
- **Eigenfeatures:** They are derived from the spectral decomposition of a graph´s adjacency matrix, and capture structural properties of the graph. [1] We begin by calculating graph-level features related to the eigenvalues of the graph Laplacian, such as the number of connected components (indicated by the multiplicity of eigenvalue 0) and the first five nonzero eigenvalues. Then, we put node-level features based on the graph eigenvectors, including an estimation of the largest connected component using the eigenvectors corresponding to eigenvalue 0 and the first two eigenvectors associated with nonzero eigenvalues.

E. Evaluation of generated graphs

The Erdös–Rényi (ER) model is used as a baseline model to compare with our discrete diffusion model. The following metrics are used to test performance.

- **Clustering Coefficient & Degree:** We calculated MMD of clustering coefficient and degrees. Note that we report ratio of MMD between train/generated and train/test.
- **Eigenvalue distribution:** We conducted a Kolmogorov-Smirnov test to compare the distributions of the top 5 eigenvalues between generated set and test set.
- **Validity:** A validity function is defined according to 3 criteria. First, connected components should be 1, second, the edge degree should be bounded to 4, and third, the generated graph should be a planar graph. We report the ratio of valid graphs out of the total number of generated graph.
- **Uniqueness:** This metric measures the proportion of distinct, non-isomorphic graphs out of the set of generated graphs. This ensures that graphs do not share the exact same structure or connectivity pattern.
- **Novelty:** We report the ratio of novel graphs, i.e. graphs that are not in the training set out of all the generated graphs.

F. Training Scheme:

The cross-entropy loss function is used to measure the discrepancy between the predicted edges and the true edges. Optimization is carried out using the Adam optimizer. With no additional features the learning rate is set to 1×10^{-4}. When using additional features the learning rate is set to 1×10^{-3}. We generated 100 graphs in each case after training.

III. Results: Simple Denoiser Model

A. No additional node features

- **Learning Curve:** In this setting, no additional node features are given. Node attributes are inferred exclusively from the adjacency matrix, using 3 different types of GNNs. Following this, a MLP layer is implemented to to predict edge probability between nodes.

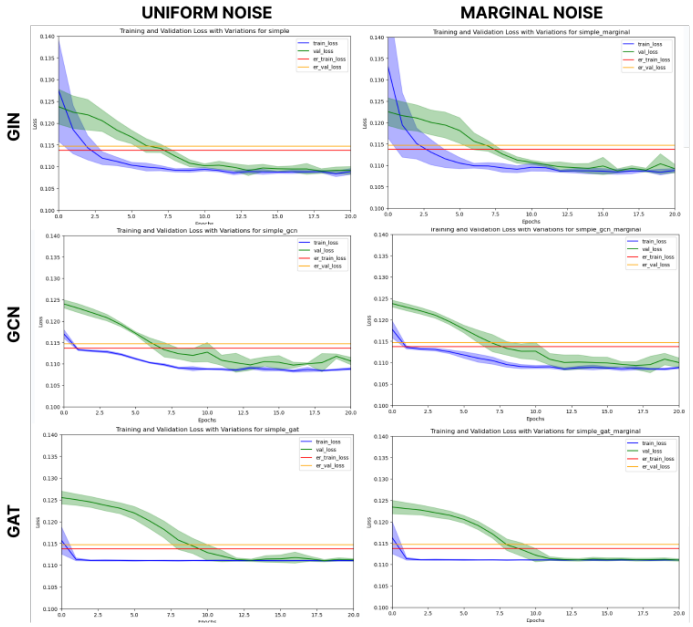

Fig. 2: Loss trend over epochs using uniform noise (left) and marginal noise (right) for forward process. 3 different message passing GNNs are used to aggregate node attributes. From top to bottom: GIN, GCN and GAT. Results are averaged over 5 different seeds, and line width shows standard deviation from the mean. Note that for all three cases, training loss does not get smaller than 0.11.

- **Evaluation:** See Table I

	C.C.↓	Degree↓	E-value KS↓	Valid↑	Unique↑	Novel↑
ER	6.80	2.74	0.15	0.27	1.00	1.00
GCN Unif.	21.25	7.11	0.15	0.03	1.00	1.00
GIN Unif.	11.71	3.87	0.18	0.16	1.00	1.00
GAT Unif.	8.00	4.69	0.14	0.24	1.00	1.00

TABLE I: Evaluation for generated graphs with no features added. ER is used as the graph generation baseline. C.C. stands for clustering coefficient. For C.C and degree, the MMD is shown. E-value KS stands for eigenvalue distribution Kolmogorov-Smirnov statistics. For these three metrics, lower values are best. For the rest, the greater the better.

It is important to note that further results not included in this report demonstrate that changing the edge predicting layer from MLP into a simple linear network does not result in a significant change in the learning curve. Thus, the limitation in expressivity resides in the node attribute layer rather than the MLP layer for predicting edges. This motivates the use of additional features, such as eigenfeatures, cycles, etc., prior to processing with a GNN.

B. Adding eigenfeatures & cycles:

- **Learning Curve:** Cycle features and eigenfeatures are added for the results in this section. Then, an MLP layer is used to predict edge probability. Loss when training with features is lower (0.09) than without (0.11) as seen in Fig 2 and Fig 3, respectively.

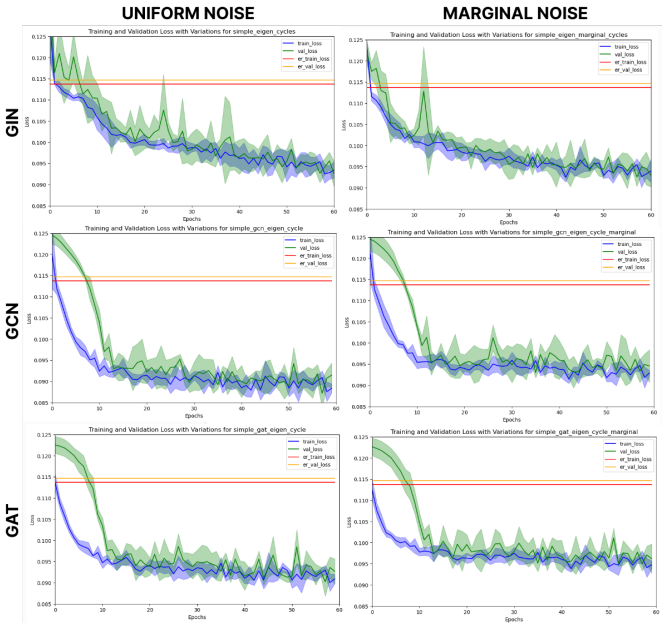

Fig. 3: Loss trend over epochs using uniform noise (left) and marginal noise (right). Additional features added for backward process: eigenfeatures, cycles. 3 different message passing GNNs are used to aggregate node attributes. From top to bottom: GIN, GCN and GAT. Results are averaged over 5 different seeds, and line width shows standard deviation from the mean.

- **Evaluation:** See Table IV

	C.C.	Degree	E-value KS	Valid	Unique	Novel
ER	6.80	2.74	0.15	0.27	1.00	1.00
GIN	5.30	5.10	0.10	0.31	1.00	1.00
GIN marg	5.61	5.41	0.11	0.33	1.00	1.00
GCN	9.15	36.35	0.09	0.26	1.00	1.00
GCN marg	7.30	5.61	0.09	0.34	1.00	1.00
GAT	4.36	15.81	0.12	0.35	1.00	1.00
GAT marg	5.12	4.26	0.10	0.36	1.00	1.00

TABLE II: Evaluation metrics for generated graphs adding eigenfeatures & cycles. Abbreviations and interpretation guidelines are provided in the caption of Table I. Note that by adding additional features, simple denoiser model performed better than erodos-reniyi model in terms of degree MMD, eigenvalue KS score, validenss.

By incorporating eigenfeatures and cycles, not only training loss decrease from 0.11 to 0.09, but also evaluation measure for generated graphs imporved. For instance, without adding features simple model was worse than Erdos-reniyi model, where MMD of clustering coefficient, degree and Kolomogrov-smirov statistics of eigenvalue distribution was higher. Morever validness was even worse. However, adding eigenfeature overcame this issue. Also note that for GCN and GAT, marginal noise showed significantly better MMD than uniform noise.

IV. Result: Graph Transformer Denoiser

Improving on the simple denoiser, the Graph Transformer model deals with edge data directly, not being bottlenecked by solely relying on node attributes. Two cases are shown: without additional features and using additional features.

- **Learning Curve:** Learning curves in Fig 4 do not appear different. However, when computing evalution metrics for the generated graphs, distinction can be seen (refer to Table III. The training loss for graph transformer is lower than for the simple denoiser, with a lower bound of 0.08 compared to 0.09. This emphasizes the expresiveness of graph transformers.

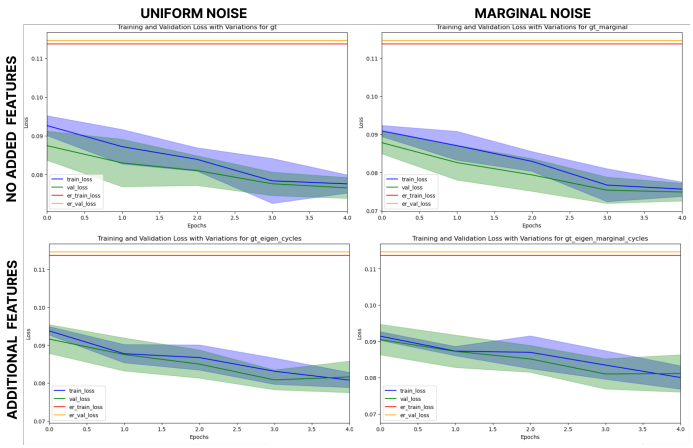

Fig. 4: Loss trend over epochs using uniform noise (left) and marginal noise (right) for forward process. Top row corresponds to training using no addtional features. Bottom row shows using both eignefeatures and cycles.

- **Evaluation:**

	C.C.	Degree	E-value KS	Valid	Unique	Novel
ER	6.80	2.74	0.15	0.27	1.00	1.00
No feat., Unif.	10.31	11.33	0.17	0.21	0.95	0.95
No feat., Marg.	8.80	7.03	0.16	0.23	0.95	0.95
Add feat., Unif.	4.41	3.49	0.10	0.38	1.00	1.00
Add feat., Marg	4.47	6.65	0.10	0.36	1.00	1.00

TABLE III: Evaluation metrics for generated graphs using the graph transformer denoiser. The results included show the performance with and without additional features when using uniform and marginal noise. Abbreviations and interpretation guidelines are provided in the caption of Table I.

V. Discussion

Looking at data, we can answer the 4 questions formulated in the introduction.

- **Performance of simple denoiser:** In the evaluation of the simple denoiser including additional features, GAT outperforms GCN and GIN across most statistical measures(without addtional features, only difference was that GIN obtains better eigenvalue Kolmogorov-Smirnov statistics). Interestingly, GAT with eigenfeatures and cycle-features reaches a close performance to the graph transformer denoiser. These findings suggest that, at least for the simplified QM9 dataset, giving additional features addresses the bottleneck of solely relying on node attributes.
- **Bottleneck of performance:** Switching the edge prediction layer from MLP into a simple linear layer had minimal impact on the learning curve. This indicates that the node feature aggregation layer is important for the model's expressivity, specially for the simple denoiser's performance. Moreover, GAT achieved better evaluation metrics for graph generation than GCN and GIN.

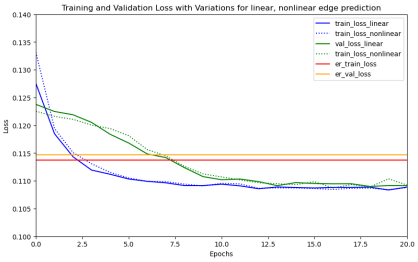

Fig. 5: Learning curve of simple denoiser using GIN as node aggregator. The node attribute layer remains fixed. Dotted line corresponds to using MLP as the edge prediction layer, while the solid line uses a simple linear layer. The loss converges to similar values in both configurations. This suggests expressivity is indeed bottlenecked in the node aggregation layer. Evaluation metrics also demonstrated comparable results between the two.

- **Choice of additional features:** The simplicity of the dataset limited the options for integrating additional features. We selected eigenfeatures as they can encapsulate long-range dependencies that are beyond the capabilities of message-passing algorithms. Cycle features are also incorporated since this is not addressed by message-passing techniques either. [15]. Including these features brings the performance of the simple model on par with that of a graph transformer. Validity and eigenvalue Kolmogorov-Smirnov statistics significantly improved when integrating new features.
- **Effect of Marginal Noise:** Due to the absence of node and edge features, the effect of marginal noise in forward diffusion is considered negligible. Training loss exhibited comparable results between using marginal noise than using uniform noise. However, the MMD of clustering coefficients and degrees is notablby improved in the simple denoiser when employing marginal noise. Thus, although the effect of marginal noise is not deemed critical, as also noted in [1], these findings suggest that marginal noise is preferable to uniform noise for enhancing model performance.

VI. Conclusion

We propose a simpler denoiser model that offers better interpretability and scalability compared to SOTA graph transformers. Our findings indicate that the node attribute layer is crucial for performance. This is highlighted by comparative assessments, which shows GAT outperforms GIN and GCN. Moreover, changing the edge prediction layer does not affect performance. To address the limitations of the node attribute layer, we incorporate spectral and cycle features in it, achieving performance comparable to graph transformer denoisers.

	C.C.↓	Degree↓	E-value KS↓	Valid↑	Unique↑	Novel↑
ER	6.80	2.74	0.15	0.27	1.00	1.00
Add feat., Simple, Marg.	5.12	4.26	0.10	0.36	1.00	1.00
Add feat., GT, Marg.	4.47	6.65	0.10	0.36	1.00	1.00

TABLE IV: Evaluation metrics for generated graphs adding eigenfeatures & cycles. Abbreviations and interpretation guidelines are provided in the caption of Table I. Note that by adding additional features, simple denoiser model performed better than erodos-reniyi model in terms of degree MMD, eigenvalue KS score, validenss.

REFERENCES

[1] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard, "Digress: Discrete denoising diffusion for graph generation," *arXiv preprint arXiv:2209.14734*, 2022.

[2] L. Huang, T. Xu, Y. Yu, P. Zhao, X. Chen, J. Han, Z. Xie, H. Li, W. Zhong, K.-C. Wong *et al.*, "A dual diffusion model enables 3d molecule generation and lead optimization based on target pockets," *Nature Communications*, vol. 15, no. 1, p. 2657, 2024.

[3] B. Hu, A. Saragadam, A. Layton, and H. Chen, "Synthetic data from diffusion models improve drug discovery prediction," *arXiv preprint arXiv:2405.03799*, 2024.

[4] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," *arXiv preprint arXiv:2203.02923*, 2022.

[5] L. Huang, H. Zhang, T. Xu, and K.-C. Wong, "Mdm: Molecular diffusion model for 3d molecule generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 5105–5112.

[6] J. Torge, C. Harris, S. V. Mathis, and P. Lio, "Diffhopp: A graph diffusion model for novel drug design via scaffold hopping," *arXiv preprint arXiv:2308.07416*, 2023.

[7] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, and D. Merhof, "Diffusion models in medical imaging: A comprehensive survey," *Medical Image Analysis*, p. 102846, 2023.

[8] H. Jiang, M. Imran, L. Ma, T. Zhang, Y. Zhou, M. Liang, K. Gong, and W. Shao, "Fast-ddpm: Fast denoising diffusion probabilistic models for medical image-to-image generation," *arXiv e-prints*, pp. arXiv–2405, 2024.

[9] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[10] K. Deja, A. Kuzina, T. Trzcinski, and J. Tomczak, "On analyzing generative and denoising capabilities of diffusion-based deep generative models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26 218–26 229, 2022.

[11] P. Kumar and A. Sinha, "Information diffusion modeling and analysis for socially interacting networks," *Social Network Analysis and Mining*, vol. 11, no. 1, p. 11, 2021.

[12] Y. Jiang, Y. Yang, L. Xia, and C. Huang, "Diffkg: Knowledge graph diffusion model for recommendation," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 313–321.

[13] M. Madeira, D. Thanou, and P. Frossard, "Tertiary lymphoid structures generation through graph-based diffusion," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2023, pp. 37–53.

[14] H. Yu, M. Liu, Y. Luo, A. Strasser, X. Qian, X. Qian, and S. Ji, "Qh9: A quantum hamiltonian prediction benchmark for qm9 molecules," 2024.

[15] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?" 2020.