Flatten Graphs as Sequences: Transformers are Scalable Graph Generators

Dexiong Chen
Max Planck Institute of Biochemistry
Martinsried, Germany
dchen@biochem.mpg.de

Markus Krimmel
Max Planck Institute of Biochemistry
Martinsried, Germany
krimmel@biochem.mpg.de

Karsten Borgwardt
Max Planck Institute of Biochemistry
Martinsried, Germany
borgwardt@biochem.mpg.de

Abstract

We introduce AUTOGRAPH, a scalable autoregressive model for attributed graph generation using decoder-only transformers. By flattening graphs into random sequences of tokens through a reversible process, AUTOGRAPH enables modeling graphs as sequences without relying on additional node features that are expensive to compute, in contrast to diffusion-based approaches. This results in sampling complexity and sequence lengths that scale optimally linearly with the number of edges, making it scalable and efficient for large, sparse graphs. A key success factor of AUTOGRAPH is that its sequence prefixes represent induced subgraphs, creating a direct link to sub-sentences in language modeling. Empirically, AUTOGRAPH achieves state-of-the-art performance on synthetic and molecular benchmarks, with up to 100x faster generation and 3x faster training than leading diffusion models. It also supports substructure-conditioned generation without fine-tuning and shows promising transferability, bridging language modeling and graph generation to lay the groundwork for graph foundation models.
Our code is available at https://github.com/BorgwardtLab/AutoGraph.

1 Introduction

Recent advancements in deep generative models have revolutionized various domains of artificial intelligence, demonstrating remarkable capabilities in generating complex data types such as images [55], natural language [7, 63, 64], and audio [17, 28]. These achievements have been primarily driven by the development of advanced architectures or methods such as transformers and diffusion models, alongside increasingly large-scale data resources. However, the generation of graph-structured data, which is fundamental to numerous scientific applications including drug discovery [66, 44], protein design [29], and program synthesis [5], remains a significant challenge. This disparity primarily stems from the inherent complexity of preserving structural validity, maintaining invariance properties within graphs, and achieving scalability in real-world graph generation tasks.

To this end, diffusion-based models have emerged as a promising direction for graph generation, demonstrating effectiveness in synthesizing both classic unattributed graphs and molecules [34, 66]. These approaches typically implement a denoising process in discrete graph space, simultaneously predicting edge connectivity and attributes. Yet, their practical applications are constrained by fundamental scalability limitations. The requirement for full adjacency matrix operations imposes quadratic memory complexity with respect to the number of nodes. Moreover, computing additional

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

node features in each denoising step, such as spectral features, often involving cubic complexity, further increases the computational overhead.

Autoregressive approaches represent an alternative paradigm, constructing graphs sequentially by generating nodes and edges in a step-by-step manner [43, 71]. These models have demonstrated strong performance in generating small to medium-sized graphs by leveraging their ability to maintain structural validity through the generation process. Nevertheless, these models face inherent limitations: their sequences are not composed of tokens and thereby require specialized architectures, primarily based on recurrent neural networks, to process their complex ad-hoc sequential representations, preventing them from directly leveraging the remarkable advances in large language models (LLMs). Moreover, these specialized architectures often struggle with long-range dependencies and global structural consistency, leading to significantly inferior performance compared to recent diffusion models [66, 35]. This representational and architectural constraint not only limits their scalability but also creates a growing performance gap as general-purpose LLMs continue to advance rapidly.

In light of these challenges, we introduce a novel paradigm that bridges the gap between graph generation and LLMs through a graph-to-sequence transformation. Our approach advances previous random walk-based methods by representing graphs as sequences of tokens while maintaining their topological properties. Instead of requiring specialized architectures or operating directly on graph structures, we propose a method to linearize graphs into random sequences that encode local connectivity patterns. This transformation enables direct utilization of language models for graph generation while achieving optimally linear complexity with respect to the number of edges in both computational and memory requirements. Our approach effectively addresses the limitations of both diffusion-based and autoregressive methods: it maintains structural validity while enabling efficient scaling to large graphs and leveraging the powerful capabilities of modern language models.

Our work presents several technical contributions to the field of graph generation. (1) We introduce the concept of segmented Eulerian neighborhood trails (SENTs), a specialized class of Eulerian trails that permit breaks and incorporate neighborhood information. We establish sufficient conditions under which they can be employed for effective graph generation. (2) We propose an efficient flattening algorithm that transforms graphs into sequences and vice versa by sampling these SENTs, enabling lossless sequence representation of graphs. (3) Our method, termed AUTOGRAPH, achieves state-of-the-art (SOTA) performance across diverse synthetic and molecular graph generation benchmarks, delivering a 100-fold generation and a 3-fold training speedup compared to diffusion-based models while maintaining the ability to scale to graphs of possibly immense size. (4) Additionally, AUTOGRAPH demonstrates strong transfer learning capabilities and supports substructure-conditioned generation without additional fine-tuning. Our work not only advances the field of graph generation but also opens new avenues for applying LLMs to graph-centric tasks, paving the way for building foundation models for graphs.

2 Methods

In this section, we present an approach to transforming graphs into sequences, enabling their modeling akin to natural language. Our method hinges on a specialized class of random trail segments that ensure complete graph coverage. We begin by introducing the concept of segmented Eulerian trails (SET) and demonstrate theoretically why this representation alone is insufficient for effective graph generation. Subsequently, we propose an extension of SET, namely the segmented Eulerian neighborhood trail (SENT), which additionally incorporates neighborhood information alongside the trails. We elucidate sufficient conditions for effective generation and develop an efficient sampling strategy to obtain such SENTs. The section concludes with extensions and discusses how to model the SENTs autoregressively using language models, thus bridging the gap between graph learning and language modeling paradigms. An overview of AUTOGRAPH is illustrated in Figure 1, and backgrounds and proofs are provided in Appendix C and D.

2.1 Segmented Eulerian Trail

To formalize our approach, we begin by introducing fundamental concepts in graph theory. Let V be a set of vertices and $E := V \times V$ a set of edges. A graph is defined as a tuple $G = (V_G, E_G)$, where $V_G \subseteq V$ is a finite set of vertices and $E_G \subseteq V_G \times V_G$ is the set of edges. For simplicity and without loss of generality, we restrict our attention to undirected graphs without isolated vertices, where each

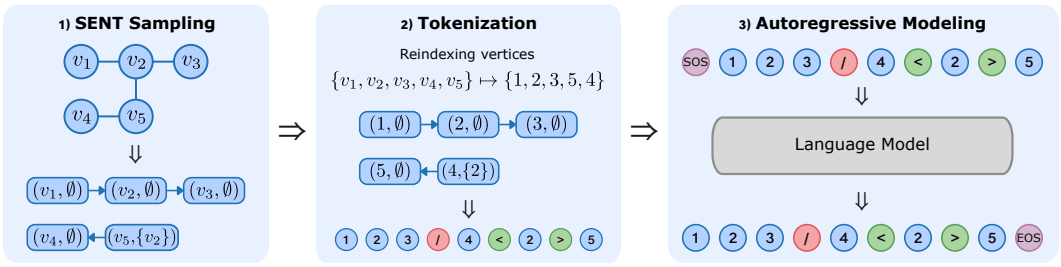

Figure 1: Overview of AUTOGRAPH: (1) We use Algorithm 1 to sample a SENT s from the input graph: $s = (s_1, s_2)$ with $s_1 = ((v_1, \emptyset), (v_2, \emptyset), (v_3, \emptyset))$ and $s_2 = ((v_5, \{v_2\}), (v_4, \emptyset))$. (2) We tokenize it by reindexing the vertices based on their first occurrence order in s and adding special tokens ('/' represents breakage between segments, '<' and '>' indicate the start and end of a neighborhood set). (3) We perform the next token prediction on the tokenized sequences using a decoder-only transformer or any language model.

edge is represented as an unordered pair (u, v) for $u, v \in V$. We begin by defining the concept of a trail in a graph:

Definition 2.1 (Walk and trail)**.** A walk is a sequence of nodes connected by edges in G and a trail is a walk in which all edges are distinct. Given a graph G, the set of trails in G is denoted as $\mathcal{T}_G$.

Next, we generalize the concept of trails beyond the context of a specific graph:

Definition 2.2 (Generalized trail)**.** A generalized trail of length k is defined as a sequence of nodes $w := (w_0, \ldots, w_k) \in V^{k+1}$ for $k \geq 0$ *s.t.* $(w_{i-1}, w_i) \neq (w_{j-1}, w_j), \forall i, j \in [k]$ and $i \neq j$.

The set of all generalized trails is denoted as $\mathcal{T}$, noting that $\mathcal{T}_G \subseteq \mathcal{T}$ for any G. For a generalized trail $w \in \mathcal{T}$, we define $V_w \subseteq V$ and $E_w \subseteq E$ as the sets of vertices and edges traversed by w, respectively, termed the *generated sets* of w. An *Eulerian trail* is a trail that visits every edge in a graph exactly once. Such trails are of particular interest as they capture the complete topology of the graph. However, the existence of an Eulerian trail depends on specific conditions related to vertex degrees and connectivity [3]. To generalize this concept to arbitrary graphs, we introduce the notion of trail segments:

Definition 2.3 (Segmented Eulerian trail (SET))**.** A segmented Eulerian trail (SET) in G is a sequence of trail segments such that each edge is visited exactly once across all segments, and segments do not need to be connected. Formally, a SET of size k in G is defined as $s := (s_1, \ldots, s_k)$ *s.t.* $s_i \in \mathcal{T}_G$, and the generated edge sets of its segments form a partition of E_G, *i.e.,* $\cup_{i=1}^{k} E_{s_i} = E_G$ and $E_{s_i} \cap E_{s_j} = \emptyset, \forall i, j \in [k], i \neq j$. Similarly, a SET (without relying on a specific graph) is defined as a sequence of generalized trails whose generated edge sets are disjoint.

The set of all SETs in G is denoted as $\mathcal{S}_G$, and the set of all SETs is denoted as $\mathcal{S}$. For a SET $s = (s_i)_{i=1}^{k}$, we define the *generated node and edge sets* as $V_s := \cup_{i=1}^{k} V_{s_i}$ and $E_s := \cup_{i=1}^{k} E_{s_i}$. The graph $G_s := (V_s, E_s)$ is termed *generated graph* of s. It is easy to show that s is a SET in G if $G_s \simeq G$. Moreover, SETs can be classified into equivalence classes based on graph isomorphism, as formalized below:

Definition 2.4 (SET isomorphism)**.** For any two SETs $s, t \in \mathcal{S}$, we say they are isomorphic $s \simeq t$ if there is a bijection $\pi : V_s \to V_t$ between their generated node sets and $\pi(s) = t$ where π applies elementwise to all nodes in s.

This isomorphism partitions $\mathcal{S}$ into equivalence classes. Moreover, we have the following relationship between SETs and graphs, relevant for our tokenization (Sec. 2.4):

Theorem 2.5. *For any SETs $s, t \in \mathcal{S}$, their generated graphs are isomorphic, i.e., $G_s \simeq G_t$, if $s \simeq t$. Conversely, if two graphs $G \simeq H$, then for any SET $s \in \mathcal{S}_G$, there exists a SET $t \in \mathcal{S}_H$ s.t. $s \simeq t$.*

While a SET in G fully characterizes its structure, we show below the prefixes of the SET do not necessarily describe the substructures of G, a critical property for effective autoregressive graph generation.

Definition 2.6 (Flattening)**.** The flattening of a sequence of sequences s is the concatenation of all its sequences, denoted as $\|s$.

Definition 2.7 (Prefix of a SET)**.** For $s \in \mathcal{S}$, we call t a prefix of s if $\|t$ is a prefix of $\|s$.

Lemma 2.8. *For any graph G and SET s in G, the generated graph of any prefix of s is a subgraph of G, but not necessarily an induced subgraph.*

This result motivates us to extend the definition of generalized trails to incorporate the full structural information of the *induced subgraphs*, rather than arbitrary subgraphs, to constrain the generation space better and address long-range dependency challenges. Without this extension, dependencies between neighboring nodes may span a long sequence of generation steps, making it more difficult for the model to learn such dependencies. Empirically, we show that SET fails to generate structurally valid graphs in Section 4.4.

2.2 Segmented Eulerian Neighborhood Trail

To make the prefixes of a SET encode richer information, we need to extend SET to contain neighborhood information in a graph. Thus, we consider the following definitions:

Definition 2.9 (Neighborhood sequence)**.** A neighborhood sequence is a sequence of tuples $w := (w_0, \ldots, w_k)$ where $w_i = (v_i, A_i)$ with a node $v_i \in V$ and a neighborhood set $A_i \subseteq V$, $\forall i \in \{0, \ldots, k\}$. w is called Hamiltonian if its node sequence $n(w) := (v_0, \ldots, v_k)$ has non-repeated elements. w is called *causal* if A_i only contains visited nodes, *i.e.,* $A_i \subseteq \{v_0, \ldots, v_{i-1}\}$ $\forall i \in [k]$.

Definition 2.10 (Neighborhood trail)**.** A neighborhood trail is a neighborhood sequence that satisfies two conditions. (i) $n(w)$ is a generalized trail. (ii) If we define the generated edge set of w_i as $E_{w_i} = \{(v_i, u) \mid u \in A_i\}$, the family $\{E_{n(w)}, E_{w_1}, \ldots, E_{w_k}\}$ is pairwise disjoint. Its union is called the generated edge set of w.

The set of all neighborhood trails is denoted by $\mathcal{T}^{\mathcal{N}}$. For any $w \in \mathcal{T}^{\mathcal{N}}$, we denote by $G_w := (V_w, E_w)$ the generated graph of w where $V_w := (\cup_{i=1}^{k} A_i) \cup V_{n(w)}$ is the generated node set and E_w is the generated edge set. Note that a generalized trail is a neighborhood trail with $A_i = \emptyset, \forall i$. We extend SET to incorporate neighborhood information:

Definition 2.11 (Segmented Eulerian neighborhood trail (SENT))**.** A segmented Eulerian neighborhood trail (SENT) of size k is a sequence of neighborhood trails $s := (s_1, \ldots, s_k)$ with pairwise disjoint generated edge sets, *i.e.,* $s_i \in \mathcal{T}^{\mathcal{N}}$ and $E_{s_i} \cap E_{s_j} = \emptyset, \forall i, j \in [k], i \neq j$.

Similarly to SETs, the generated graph of a SENT s is denoted by $G_s = (V_s, E_s)$. If a graph $G \simeq G_s$, we say that s is a SENT in G. We denote by $\mathcal{S}^{\mathcal{N}}$ and $\mathcal{S}_G^{\mathcal{N}}$ the set of SENTs and SENTs in G. Analogously to SETs, we define an isomorphism over $\mathcal{S}^{\mathcal{N}}$ and obtain the same relationship as in Thm. 2.5. A prefix of a SENT is defined similarly to that of a SET. We give below conditions to force generated graphs of prefixes of a SENT to be induced subgraphs.

Definition 2.12 (Causal SENT)**.** A SENT s is called causal if its flattening $\|s$ is causal.

Definition 2.13 (Hamiltonian and semi-hamiltonian SENT)**.** A SENT s is called Hamiltonian if its flattening $\|s$ is Hamiltonian. s is called semi-hamiltonian if s is Hamiltonian, or for any nodes visited more than once, their occurrences after the first time should be in a start tuple of a neighborhood trail and their associated neighborhood sets are empty.

Theorem 2.14. *For any causal SENT $s \in \mathcal{S}^{\mathcal{N}}$, the generated graph of any prefix t of s is an induced subgraph of G_s if and only if s is semi-hamiltonian. In this case, s is called subgraph-induced.*

Now let us find the conditions for a causal and Hamiltonian SENT. For any SENT s and a tuple $w := (v, A)$ in s, we denote by $V_s(w)$ the set of nodes visited by s before w, excluding the node linked to v through the trail if it exists. We have the following necessary and sufficient conditions:

Theorem 2.15. *For $s \in \mathcal{S}_G^{\mathcal{N}}$, s is causal and Hamiltonian if and only if every tuple $w := (v, A_v)$ in $\|s$ satisfies $A_v = \mathcal{N}_G(v) \cap V_s(w)$. In this case, every node is visited exactly once. Moreover, s is causal and semi-hamiltonian if and only if every tuple $w := (v, A_v)$ in s satisfies either $A_v = \mathcal{N}_G(v) \cap V_s(w)$ or $A_v = \emptyset$.*

This theorem offers a simple sufficient condition for subgraph-induced SENTs. We provide an implementation in the following through a random path sampling strategy.

2.3 Sampling Algorithm for SENT

Thm. 2.15 offers a simple strategy to sample a causal and Hamiltonian SENT: one needs to traverse the graph and choose the neighborhood set as all neighbors of the current node that have been visited. The traversing strategy could be achieved through a random path sampling or a depth-first search. In Algorithm 1, we provide a sampling strategy based on random path sampling with breaks.
Complexity analysis. The length of a SENT, including the sizes of neighborhood sets (in other words, tokenized SENT defined in Section 2.4), is bounded by the number of nodes and edges, as each node and edge can be visited exactly once. Therefore, both the time and space complexity of sampling a SENT from graph G are $\mathcal{O}(m)$ where m is the number of edges.

2.4 Tokenization of SENT

Previous works have explored related concepts of sequences in graphs. For example, You et al. [71] investigated causal Hamiltonian neighborhood sequences generated through breadth-first search, while Liao et al. [43], Goyal et al. [23] constructed SENT-like sequences using depth-first search. However, neither of these works interpreted these sequences as a language. Here, we present a method to bridge the gap between graph generation and language modeling.

Algorithm 1 Causal and Hamiltonian SENT Sampling

Input: $G = (V, E)$
Output: A SENT s in G

Set of unvisited nodes $U \leftarrow V$
$s \leftarrow []$
$v \leftarrow \texttt{RandomSample}(U)$; $U \leftarrow U \setminus \{v\}$
$t \leftarrow [(v, \emptyset)]$ ▷ first neighborhood trail
while $U \neq \emptyset$ **do**
  if $\mathcal{N}_G(v) \cap U = \emptyset$ **then** ▷ start a new trail
    $s.\texttt{append}(t)$
    $v \leftarrow \texttt{RandomSample}(U)$; $U \leftarrow U \setminus \{v\}$
    $A \leftarrow \mathcal{N}_G(v) \cap (V \setminus U)$
    $t \leftarrow [(v, A)]$
  else ▷ sample the next node in the trail
    $u \leftarrow \texttt{RandomSample}(\mathcal{N}_G(v) \cap U)$
    $U \leftarrow U \setminus \{u\}$
    $A \leftarrow (\mathcal{N}_G(u) \setminus \{v\}) \cap (V \setminus U)$
    $t.\texttt{append}((u, A))$
    $v \leftarrow u$

The tokenization process starts by mapping all isomorphic SENTs to the same sequence, by reindexing the vertices according to their first occurrence order within the sequence. Specifically, if we denote this ordering function for a SENT s by $\pi : V_s \to \{1, \ldots, |V_s|\}$, s is then replaced with its ordered representation $\pi(s)$. Thanks to the isomorphism property of SENT (Thm. 2.5), $\pi(s)$ generates a graph isomorphic to G_s while ensuring the obtained sequence is invariant to the node ordering of the input graph.

To convert an (ordered) SENT into a machine-readable sequence, we tokenize it into a sequence of indices using special tokens. These tokens include symbols such as '/' to indicate a breakage between segments, and '<' and '>' to mark the start and end of a neighborhood set. Specifically, for any $s := (s_1, \ldots, s_k) \in \mathcal{S}^{\mathcal{N}}$, we define the tokenization function $\texttt{Token}$ as follows:

$$\texttt{Token}(s) := \texttt{Token}(s_1) \parallel [\,/\,] \parallel \cdots \parallel [\,/\,] \parallel \texttt{Token}(s_k), \text{ where } \texttt{Token}(s_i) := \|_{w \in s_i} \texttt{Token}(w),$$

and for each tuple $w := (v, A)$ with the *sorted set* $A = \{u_1, \ldots, u_p\}$ (due to the reindexing by π), we define:

$$\texttt{Token}(w) := [v, <, u_1, \ldots, u_p, >]\,.$$

This process converts a SENT into a sequence of tokens that a language model can effectively model. Using an equivalent form, the resulting tokenization induces a *non-Markovian* random walk in the graph, incorporating additional virtual nodes labeled with the above special tokens (see Appendix C.2 for more details). *Language modeling of SENTs aims to learn the state transition probabilities.*

2.5 Extension to Attributed Graphs

Our method can be easily extended to graphs with categorical (or discretized) attributes by inserting node and edge attributes in an interleaved fashion into the tokenized SENT sequence. Specifically, let $L_{\text{node}}(v)$ and $L_{\text{edge}}(u, v)$ be the attributes of a node v and an edge (u, v) respectively. Using the same notation as above, we define for any $s_i := (w_1, \ldots, w_q) \in \mathcal{T}^{\mathcal{N}}$ with $w_i = (v_i, \cdot)$:

$$\texttt{Token}(s_i) := \texttt{Token}(w_1) \parallel [L_{\text{edge}}(v_1, v_2)] \parallel \texttt{Token}(w_2) \parallel \cdots \parallel \texttt{Token}(w_q),$$
$$\texttt{Token}(w) := [v, L_{\text{node}}(v), <, L_{\text{edge}}(v, u_1), u_1, \ldots, L_{\text{edge}}(v, u_p), u_p, >].$$

2.6 Autoregressive Modeling of Tokenized SENTs

The sampling and tokenization of SENTs in graphs allows for transforming graphs into sequences, which could be modeled by language models. Specifically, given a graph G represented as a SENT s, which consists of a sequence of tokens $(s_1, \ldots, s_n)$, a standard language modeling objective is to maximize the following log-likelihood:

$$p(s) = \sum_{i=1}^{n} \log p_\theta(s_i \mid s_1, \ldots, s_{i-1}), \tag{1}$$

where the conditional probability p_θ is modeled using a neural network with parameters θ. The architecture of the neural network can be any state-of-the-art sequence model.

3 Related Work

Autoregressive models for graph generation. Autoregressive models generate graphs by sequentially adding nodes and edges. GraphRNN [71] pioneered this approach by framing graph generation as a sequence prediction task, demonstrating the capacity of recurrent neural networks (RNNs) [15] to capture complex structures. DeepGMG [42] introduced a probabilistic policy framework for conditional generation, while GRAN [43] and BiGG [16] enhanced efficiency and scalability by generating multiple nodes and edges in parallel. ANFM [40] leverages filtration to improve efficiency.

Recent research has focused on optimizing the generation order. Chen et al. [12] highlighted that the ordering of node and edge additions impacts graph quality, and GraphARM [39] applied reinforcement learning to dynamically refine this order. Goyal et al. [23] incorporated logical constraints to improve domain-specific generation, and Bacciu et al. [1] proposed Bayesian reasoning to better capture graph dependencies. BwR [18] and GEEL [32] investigated node ordering based on optimized bandwidth.

These models, while efficient on synthetic datasets, do not explicitly represent graphs as token sequences, preventing direct application of LLM techniques. More significantly, their sequences are not guaranteed to be subgraph-induced (see Thm. 2.14). In contrast, our approach enables substructured-conditioned generation analogous to prompt-based generation in language models, establishing a more fundamental connection between graph and language modeling paradigms.

Other graph generative models. Other graph generative models include variational, GAN-based, flow-based, and diffusion-based approaches. GraphVAEs [38, 58] employ variational autoencoders to learn latent representations, effectively generating small graphs but struggling with more complex structures. GAN-based models, such as NetGAN [4] and SPECTRE [45], generate graphs by modeling graph descriptors like random walks and spectral features. Flow-based methods such as [57] have shown the ability to generate small molecular graphs.

Diffusion-based models iteratively refine noise into structured graphs through reverse diffusion steps. Continuous diffusion models [51, 34] adapt denoising diffusion probabilistic models for graph generation. To leverage graph sparsity and structure, discrete diffusion models [66, 39, 35, 69] have been developed. However, a key challenge for these models is the slow sampling process due to the long reverse diffusion chain. To mitigate this limitation, several efficient diffusion techniques have been proposed, including EDGE [13], HiGen [36], ESGG [2], and Pard [73].

Random walks for graph learning. Random walks have been widely used in graph learning due to their strong expressive power. GCKN [8] and RWGNN [50] utilize path and walk kernels to learn graph representations. Several recent works [30, 67, 70] explicitly integrate random walk sequences with positional encodings, inspiring subsequent methods such as CRaWL [62], NeuralWalker [10] and RWNN [37]. GraphGPT [74] leverages Eulerian paths to improve graph property prediction. Some graph transformers [48, 9] also leverage features based on random walks. Moreover, graph-to-sequence representations have been used to assist LLMs in understanding graphs [22, 11]. Our work explores sequence representations of graphs for graph generation, introducing a novel perspective on combining random walks and language modeling for scalable graph generation.

4 Experiments

In this section, we evaluate the performance of AUTOGRAPH on several graph generation benchmarks, including both small and large graphs, and synthetic and real-world molecular datasets. Our experi-

Table 1: Benchmarking AUTOGRAPH on Planar and SBM

	PLANAR GRAPHS $n_{\text{graphs}} = 128$, $\lvert V\rvert = 64$						STOCHASTIC BLOCK MODELS $n_{\text{graphs}} = 128$, $\lvert V\rvert_{\max} = 187$, $\lvert V\rvert_{\text{avg}} \approx 104$					
MODEL	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN
TRAINING SET	0.0002	0.0310	0.0005	0.0038	1.0	–	0.0008	0.0332	0.0255	0.0027	1.0	–
GRAPHRNN [71]	0.0049	0.2779	1.2543	0.0459	638.5	0.0	0.0055	0.0584	0.0785	0.0065	3.5	5.0
GRAN [43]	0.0007	0.0426	0.0009	0.0075	2.1	0.0	0.0113	0.0553	0.0540	0.0054	5.0	25.0
SPECTRE [45]	0.0005	0.0785	0.0012	0.0112	2.6	25.0	0.0015	0.0521	0.0412	0.0056	1.8	52.5
EDGE [1]	0.0761	0.3229	0.7737	0.0957	490.9	0.0	0.0279	0.1113	0.0854	0.0251	12.7	0.0
GRAPHGEN [23]	0.0328	0.2106	0.4236	0.0430	257.3	7.5	0.0550	0.0623	0.1189	0.0182	20.5	5.0
BIGG [16]	0.0007	0.0570	0.0367	0.0105	20.4	5.0	0.0012	0.0604	0.0667	0.0059	2.0	10.0
DIGRESS [66]	0.0007	0.0780	0.0079	0.0098	6.1	77.5	0.0018	**0.0485**	**0.0415**	0.0045	1.8	60.0
GRUM [35]	0.0005	**0.0353**	0.0009	**0.0062**	1.8	90.0	**0.0007**	0.0492	0.0448	0.0050	**1.5**	85.0
GEEL [32]	0.0039	0.0013	0.0062	0.0234	9.5	0.0	0.0106	0.0616	0.0023	0.0381	7.3	5.0
ESGG [2]	0.0005	0.0626	0.0017	0.0075	2.5	**95.0**	0.0119	0.0517	0.0669	0.0067	5.4	45.0
AUTOGRAPH	**0.0004**	0.0605	**0.0003**	0.0064	**1.5**	87.5	0.0077	0.0519	0.0439	**0.0040**	3.4	**92.5**

ments compare its performance to several SOTA methods and particularly focus on evaluating the following aspects: (1) We show its ability to generate relatively small graphs with a 100-fold inference speedup compared to diffusion-based models while maintaining or even improving structural validity. (2) We show its ability to scale to large graphs without loss of performance. (3) We demonstrate its effectiveness in generating real-world graphs with attributes with a focus on molecular generation, outperforming SOTA diffusion models. (4) We showcase its strong transfer capabilities and its ability to perform substructure-conditioned generation without any additional fine-tuning. Additional details on experimental settings and evaluation are provided in Appendix E.

Implementation details. We employ the LLaMA model with 12 layers and a hidden dimension of 768 as our sequence model backbone across all experiments, aligning with the architecture of GPT-2's smallest variant [54]. Although prior works have used smaller models, we argue that our approach still demonstrates better scalability and faster training and inference speeds compared to diffusion models. For inference, we adopt the commonly used top-k sampling strategy [21]. Our implementation leverages the Hugging Face framework [31], providing users with a flexible interface to experiment with SOTA language models for graph generation.

Evaluation. For fair comparison, we align our evaluation methodology with established practices from prior works [71, 45, 66]. Our evaluation compares generated samples against the test set using maximum mean discrepancy (MMD) [24], computed across multiple graph descriptors: node degree distributions (DEG.), clustering coefficients (CLUS.), orbit count statistics (ORBIT), and eigenvalue spectra (SPEC.). As a reference, we also compute MMDs between the training and test sets and report the average ratio between generated and training MMDs (RATIO) following Bergmeister et al. [2].

For synthetic datasets, we additionally assess model performance using the VUN metric, namely the proportion of generated graphs that are simultaneously valid, unique, and novel compared to the training graphs. Our efficiency analysis includes two measurements: inference speed, calculated as the per-graph generation time when producing 1024 graphs, and training speed, measured as the time required to achieve a VUN score of 75.0 for the Planar dataset and 60.0 for the SBM dataset. All efficiency measurements are performed on a single NVIDIA H100 GPU.

For molecular generation datasets, we strictly follow the evaluation metrics used in DiGress [66] and use the evaluation tools from the official codebase [53, 6]. More details are provided in App. E.2.

4.1 Comparison to State-of-the-Art Methods

We evaluate the performance of AUTOGRAPH compared to other SOTA graph generative models using the standard setting without pre-training. The comparison partners include GraphRNN [71], GRAN [43], SPECTRE [45], EDGE [13], GraphGen [23], BiGG [16], DiGress [66], GruM [35], GEEL [32], and ESGG [2].

Small synthetic graph generation. We first evaluate our method on the small synthetic graph datasets introduced by Martinkus et al. [45], including the Planar and SBM datasets. As shown in Table 1, AUTOGRAPH demonstrates competitive MMDs while ranking second-best and best in terms of VUN scores on the Planar and SBM datasets, respectively. Importantly, only GruM and AUTOGRAPH exhibit strong structural validity (VUN ≥ 80.0) on the SBM dataset. Previous state-of-the-art autoregressive models, particularly GEEL, completely fail on both datasets in terms

Table 2: Benchmarking AUTOGRAPH on Proteins and Point Clouds. OOM indicates out of memory. Note that for GEEL [32], we fail to reproduce their experiments on the Point Clouds dataset using their official codebase.

	PROTEINS $n_{\text{graphs}} = 587, \lvert V\rvert_{\max} = 500, \lvert V\rvert_{\text{avg}} \approx 258$					POINT CLOUDS $n_{\text{graphs}} = 26, \lvert V\rvert_{\max} = 5037, \lvert V\rvert_{\text{avg}} \approx 1332$				
MODEL	DEG.	CLUS.	ORBIT	SPEC.	RATIO	DEG.	CLUS.	ORBIT	SPEC.	RATIO
TRAINING SET	0.0003	0.0068	0.0032	0.0005	1.0	0.0000	0.1768	0.0049	0.0043	1.0
GRAPHRNN [71]	0.0040	0.1475	0.5851	0.0152	62.1	OOM	OOM	OOM	OOM	OOM
GRAN [43]	0.0479	0.1234	0.3458	0.0125	77.7	0.0201	0.4330	0.2625	0.0051	19.1
SPECTRE [45]	0.0056	0.0843	0.0267	0.0052	12.5	OOM	OOM	OOM	OOM	OOM
EDGE [1]	0.1863	0.3406	0.6786	0.1075	274.5	0.4441	0.3298	1.0730	0.4006	104.7
GRAPHGEN [23]	0.0159	0.1677	0.3789	0.0181	58.1	OOM	OOM	OOM	OOM	OOM
BIGG [16]	0.0070	0.1150	0.4696	0.0067	50.1	0.0994	0.6035	0.3633	0.1589	38.2
DIGRESS [66]	0.0041	0.0489	0.1286	0.0018	16.2	OOM	OOM	OOM	OOM	OOM
GRUM [35]	0.0019	0.0660	0.0345	0.0030	8.2	OOM	OOM	OOM	OOM	OOM
GEEL [32]	0.2110	0.3753	0.1768	0.1689	287.9	–	–	–	–	–
ESGG [2]	0.0030	0.0309	0.0047	0.0013	4.7	0.0139	0.5775	0.0780	0.0055	6.8
AUTOGRAPH	0.0004	0.0244	0.0056	0.0013	**2.3**	0.0307	0.3031	0.0167	0.0171	**3.0**

of VUN scores, as they largely memorize some subset of the training data. It is worth noting that the relatively low MMD ratio of AUTOGRAPH is expected, as we selected the best model based on the VUN score. More results with error bars are provided in App. F.1.

Additionally, we assess the training and inference times of AUTOGRAPH against representative models, including DiGress, GRAN, and ESGG. As presented in Table 3, AUTOGRAPH is approximately 3 times faster during training and 100 times faster during inference compared to diffusion-based models. This substantial speedup over diffusion-based models is even more pronounced than that observed in other data modalities such as images [61].

Large graph generation. To understand the scalability of AUTOGRAPH, we evaluate its performance on the Proteins and Point Clouds datasets used by Liao et al. [43]. The results, shown in Table 2, demonstrate that even when using a context window shorter than the longest sequence during training, AUTOGRAPH achieves MMD ratios comparable to those observed on the Planar and SBM datasets. Furthermore, AUTOGRAPH outperforms all existing methods in terms of MMD ratio, achieving a twofold or more improvement over the previous best model, ESGG. More significantly, while ESGG was specifically designed for generating unattributed graphs, AUTOGRAPH demonstrates versatility by being applicable to both unattributed and attributed graphs. Finally, previous state-of-the-art autoregressive models, such as GEEL, again fail to achieve competitive performance on these datasets.

Table 3: Time comparison of AUTOGRAPH to representative models. OOT indicates the model never reaches the target VUN.

DATASET	TIME	DIGRESS	GRAN	ESGG	AUTOGRAPH
PLANAR	TRAINING	25.9H	OOT	7.4H	**6.2H (4.2×)**
	INFERENCE	2.84S	0.03S	4.60S	**0.01S (284×)**
SBM	TRAINING	47.7H	OOT	OOT	**13.8H (3.5×)**
	INFERENCE	13.05S	**0.13S**	30.0S	**0.14S (93×)**

Molecular graph generation. We demonstrate the applicability of our method to generating real-world attributed graphs, such as molecular structures. We evaluate AUTOGRAPH on the same datasets used by DiGress [66], including QM9 (all atoms) [68], MOSES [53], and GuacaMol [6]. Following the data splits and experimental setup from DiGress, we benchmark AUTOGRAPH against a variety of SOTA models, including DiGress, VAE on SMILES [53], JT-VAE [33], GraphINVENT [47], NAGVAE [41], LSTM and MCTS [6]. On the QM9 dataset (Table 12), AUTOGRAPH outperforms DiGress across all metrics except uniqueness, showing its superiority for attributed graphs.

For the more challenging MOSES and GuacaMol datasets, AUTOGRAPH also demonstrates superior performance, achieving higher validity and improved distributional alignment as measured by metrics like FCD, as shown in Table 4. Notably, to our best knowledge, AUTOGRAPH is the first autoregressive model for graphs to surpass diffusion-based approaches on these datasets. It is worth mentioning that all metrics were computed using SMILES representations rather than molecular graphs. Due to the non-reversible nature of converting SMILES to graphs and back, where approximately 20% of molecules cannot be mapped back to their original SMILES [66], some discrepancies are introduced when calculating these metrics. Despite these challenges, AUTOGRAPH achieves validity and FCD scores comparable to SMILES-based methods.

Table 4: Benchmarking AUTOGRAPH on the molecular generation datasets, more results in App. F.2. AUTOGRAPH* was first pretrained on the PubChem-10M dataset [14].

	MOSES $n_{\text{graphs}} = 1.58\text{M}, \lvert V\rvert_{\max} = 27, \lvert V\rvert_{\text{avg}} \approx 22$					
MODEL - TYPE	VALID↑	UNIQUE↑	NOVEL↑	FILTERS↑	FCD↓	SNN↓
VAE - SMILES	97.7	99.8	69.5	99.7	0.57	0.58
JT-VAE - FRAGMENTS	100	100	99.9	97.8	1.00	0.53
GRAPHINVENT - GRAPH	96.4	99.8	–	95.0	1.22	0.54
DIGRESS - GRAPH	85.7	100	95.0	97.1	1.19	0.52
AUTOGRAPH - GRAPH	87.4	100	85.9	98.6	0.91	0.55

	GUACAMOL $n_{\text{graphs}} = 1.1\text{M}, \lvert V\rvert_{\max} = 88, \lvert V\rvert_{\text{avg}} \approx 28$				
MODEL - TYPE	VALID↑	UNIQUE↑	NOVEL↑	KL DIV↑	FCD↑
LSTM - SMILES	95.9	100	91.2	99.1	91.3
NAGVAE - GRAPH	92.7	95.5	100	38.4	0.9
MCTS - GRAPH	100	100	99.4	52.2	1.5
DIGRESS - GRAPH	85.2	100	99.9	92.9	68.0
AUTOGRAPH - GRAPH	91.6	100	97.7	97.5	79.2
AUTOGRAPH* - GRAPH	95.9	100	95.5	98.1	91.4

Table 5: Transfer performance on downstream tasks using AUTOGRAPH pre-trained on the NetworkX dataset. Red and green colors indicate relative decreases and increases, respectively, compared to AUTOGRAPH without pre-training.

DATASET	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN (IMPROVEMENT)
NETWORKX	0.0016	0.0073	0.0068	0.0020	–	–
PLANAR	0.0007	0.0811	0.0005	0.0061	2.2	95.0 (+7.5)
SBM	0.0099	0.0566	0.0854	0.0065	4.8	97.5 (+5)
PROTEINS	0.0002	0.0183	0.0038	0.0012	1.7	–
POINT CLOUDS	0.0154	0.2591	0.0076	0.0236	2.8	–

Furthermore, AUTOGRAPH demonstrates remarkable efficiency, with training times of less than one day on both datasets, compared to up to one week for DiGress [66]. This substantial reduction underscores AUTOGRAPH's practical advantages in large-scale molecular graph generation tasks.

4.2 Transfer Performance of AUTOGRAPH

We evaluate the transferability of AUTOGRAPH by pre-training it on a large dataset of synthetic graphs generated using NetworkX [26] and fine-tuning it on the unattributed graph datasets. Dataset and experimental details are provided in Appendix E. As shown in Table 5, the pre-trained model consistently outperforms the baseline on small synthetic datasets in terms of the VUN score, achieving near-perfect validity. On larger graph datasets, the pre-trained model also surpasses the baseline across MMD metrics, demonstrating its ability to generalize to more complex structures. However, on small synthetic datasets, the pre-trained model shows a slight decline in MMD metrics compared to the baseline. These findings highlight the potential of building foundation models for graph generation and underscore the need for more comprehensive benchmarks beyond synthetic datasets.

We also test the transferability of AUTOGRAPH on molecular graphs, by pre-training it on the PubChem-10M dataset [14] and fine-tuning on GuacaMol. The pre-trained model substantially outperforms the baseline, as shown in Table 4.

4.3 Substructure Conditioned Generation

We explore the ability of AUTOGRAPH to perform substructure-conditioned generation without requiring fine-tuning. Given a subgraph S (which could represent a functional motif of interest in drug discovery), we flatten the subgraph into a SENT sequence and condition the generation process on this sequence. This approach guarantees that the generated graph will contain S as an induced subgraph (Thm. 2.14). As a proof-of-concept, we follow the methodology of Vignac et al. [66], Maziarz et al. [46] and generate molecular graphs starting from a specific motif, called 1,4-Dihydroquinoline[1], using the model pre-trained on the GuacaMol dataset. Our results in Table 6 demonstrate that this approach maintains similar validity, uniqueness, and novelty to unconditional generation (Table 4). To further showcase the flexibility of this method, we test more extreme cases by replicating the same motif multiple times before performing the conditional generation. While validity decreases significantly when using an unrealistically large number of copies (*e.g.,* 5), the model still generates some visually plausible molecules (Appendix F.4), showing superior flexibility over Vignac et al. [66]. These results highlight the potential of AUTOGRAPH for important applications in drug discovery, particularly in motif scaffolding. Additional experiments on multiple but different motifs are provided in App. F.4.

Table 6: Motif scaffolding

# MOTIF COPIES	VALID	UNIQUE	NOVELTY
1	92.0	98.8	99.6
2	88.8	99.7	100.0
5	66.0	100.0	100.0

[1] `https://pubchem.ncbi.nlm.nih.gov/compound/1_4-Dihydroquinoline`

Table 7: Comparison of sequence model architectures on the Planar dataset.

ARCHITECTURE	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN
GPT-2	0.0004	0.0720	0.0010	0.0053	1.8	85.0
MAMBA	0.0002	0.0429	0.0014	0.0087	1.6	55.0
LLAMA	0.0005	0.0651	0.0005	0.0056	**1.6**	**90.0**

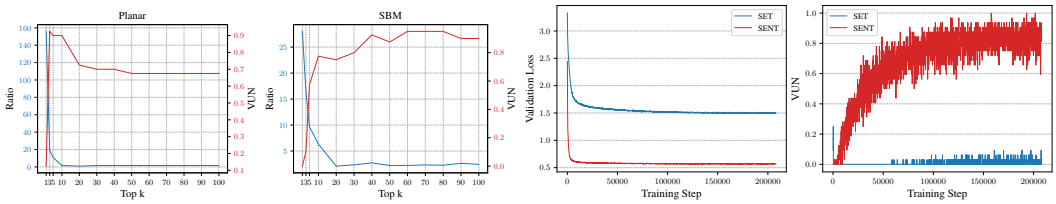

Figure 2: Ablation experiments. Left: the effect of top-k sampling on the Planar and SBM datasets. Right: the validation loss and VUN scores when using SET and SENT on the Planar dataset.

4.4 Ablation Experiments

In this study, we aim to understand the effectiveness of the key components in AUTOGRAPH.

Comparison of sequence model architectures. AUTOGRAPH provides a novel framework for evaluating the capability of current LLM architectures in graph generation and, more broadly, in structural reasoning tasks. In Table 7, we compare several state-of-the-art architectures on the Planar dataset, including GPT-2 [54], Mamba [25], and LLaMA [63]. While all models achieve comparable MMD ratios, transformer-based architectures, particularly LLaMA, demonstrate significantly better performance in terms of VUN scores compared to state-space models. These findings highlight the potential of AUTOGRAPH to serve as a valuable benchmark for assessing sequence/language models' capabilities in graph generation tasks.

Effect of top-k sampling. A key advantage of AUTOGRAPH over diffusion-based approaches is the flexibility to apply top-k sampling [21] during inference, which can improve generation quality. As shown on the left of Figure 2, a smaller k improves the VUN score on the Planar dataset, whereas it is not beneficial on the SBM dataset. In contrast, increasing k generally improves MMD ratios on both datasets. These results suggest that top-k sampling can be optimized based on dataset characteristics. In our experiments, we select the best k that maximizes the VUN score for small synthetic datasets and minimizes the validation MMD ratios for other datasets. This flexibility allows practitioners to select k based on specific performance criteria they aim to prioritize.

Comparison of SET and SENT. As discussed in Section 2, SENT is preferred over SET for graph generation, as incorporating neighborhood information is essential to ensure structural coherence. To empirically validate this, we compare the performance of SENT and SET on the Planar dataset and present the training curves on the right of Figure 2. Consistent with our theoretical analysis, SET fails to produce high-validity graphs, resulting in a VUN score close to zero, whereas SENT successfully generates valid planar graphs.

5 Conclusion

We proposed AUTOGRAPH, a scalable and efficient autoregressive model for attributed graph generation that handles large graphs while maintaining high quality. Our approach enables substructure-conditioned generation without additional fine-tuning and demonstrates promising transfer capabilities. Crucially, AUTOGRAPH establishes the first fundamental connection between graph and language modeling–where graphs are losslessly represented as token sequences, and prefixes in these sequences serve as meaningful patterns in both paradigms–representing a significant step toward applying language modeling techniques to graph generation and broader graph learning challenges.

Limitations. While AUTOGRAPH demonstrates strong scalability on current graph generation benchmarks, we acknowledge that the datasets used in our study remain relatively small-scale compared to those used in pre-training LLMs. To push the boundaries of more powerful graph generative models or eventually foundation models for graphs, we draw the community's attention to building more comprehensive graph generation benchmarks and well-curated pre-training datasets.

Acknowledgements

The authors thank Dr. Till Hendrik Schulz, Philip Hartout, and Błażej Banaszewski for their insightful discussions and valuable feedback on the manuscript.

References

[1] Davide Bacciu, Alessio Micheli, and Marco Podda. Edge-based sequential graph generation with recurrent neural networks. *Neurocomputing*, 416:177–189, 2020.

[2] Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[4] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International Conference on Machine Learning (ICML)*, pages 610–619, 2018.

[5] Marc Brockschmidt, Miltiadis Allamanis, Alexander L Gaunt, and Oleksandr Polozov. Generative code modeling with graphs. In *International Conference on Learning Representations (ICLR)*, 2019.

[6] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[8] Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *International Conference on Machine Learning (ICML)*, pages 1576–1586, 2020.

[9] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning (ICML)*, pages 3469–3489. PMLR, 2022.

[10] Dexiong Chen, Till Hendrik Schulz, and Karsten Borgwardt. Learning long range dependencies on graphs via random walks. *arXiv preprint arXiv:2406.03386*, 2024.

[11] Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. In *International Conference on Machine Learning (ICML)*, pages 7809–7823. PMLR, 2024.

[12] Xiaohui Chen, Xu Han, Jiajing Hu, Francisco Ruiz, and Liping Liu. Order matters: Probabilistic modeling of node sequence for graph generation. In *International Conference on Machine Learning (ICML)*, pages 1630–1639, 2021.

[13] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning (ICML)*, pages 4585–4610, 2023.

[14] Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

[15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning*, 2014.

[16] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International Conference on Machine Learning (ICML)*, pages 2302–2312, 2020.

[17] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[18] Nathaniel Lee Diamant, Alex M Tseng, Kangway V Chuang, Tommaso Biancalani, and Gabriele Scalia. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning (ICML)*. PMLR, 2023.

[19] Reinhard Diestel. *Graph Theory*. Electronic library of mathematics. Springer, 2005.

[20] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[21] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 889–898, 2018.

[22] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[23] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of The Web Conference*, pages 1253–1263, 2020.

[24] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13(1): 723–773, 2012.

[25] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[26] Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

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

[28] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 23802–23804, 2024.

[29] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[30] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In *International Conference on Machine Learning (ICML)*, 2018.

[31] Shashank Mohan Jain. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pages 51–67. Springer, 2022.

[32] Yunhui Jang, Seul Lee, and Sungsoo Ahn. A simple and scalable representation for graph generation. In *International Conference on Learning Representations (ICLR)*, 2024.

[33] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning (ICML)*, 2018.

[34] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning (ICML)*, pages 10362–10383, 2022.

[35] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *International Conference on Machine Learning (ICML)*. PMLR, 2024.

[36] Mahdi Karami. Higen: Hierarchical graph generative networks. In *International Conference on Learning Representations (ICLR)*, 2024.

[37] Jinwoo Kim, Olga Zaghen, Ayhan Suleymanzade, Youngmin Ryou, and Seunghoon Hong. Revisiting random walks for learning on graphs. *arXiv preprint arXiv:2407.01214*, 2024.

[38] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[39] Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In *International Conference on Machine Learning (ICML)*, pages 17391–17408, 2023.

[40] Markus Krimmel, Jenna Wiens, Karsten Borgwardt, and Dexiong Chen. Towards fast graph generation via autoregressive noisy filtration modeling. *arXiv preprint arXiv:2502.02415*, 2025.

[41] Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. Compressed graph representation for scalable molecular graph generation. *Journal of Cheminformatics*, 12:1–8, 2020.

[42] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[43] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[44] Jaechang Lim, Sang-Yeon Hwang, Seokhyun Moon, Seungsu Kim, and Woo Youn Kim. Scaffold-based molecular design with a graph generative model. *Chemical Science*, 11(4): 1153–1164, 2020.

[45] Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning (ICML)*, pages 15159–15179, 2022.

[46] Krzysztof Maziarz, Henry Richard Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR)*, 2022.

[47] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023, 2021.

[48] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

[49] Marion Neumann, Plinio Moreno, Laura Antanas, Roman Garnett, and Kristian Kersting. Graph kernels for object category prediction in task-dependent robot grasping. In *Online proceedings of the eleventh workshop on mining and learning with graphs*, pages 0–6. ACM Chicago, Illinois, USA, 2013.

[50] Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16211–16222, 2020.

[51] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4474–4484, 2020.

[52] Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten M. Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *International Conference on Learning Representations (ICLR)*, 2022.

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

[54] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[56] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[57] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations (ICLR)*, 2020.

[58] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422, 2018.

[59] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[60] Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. In *International Conference on Learning Representations (ICLR)*, 2022.

[61] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[62] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research (TMLR)*, 2023.

[63] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[64] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[66] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

[67] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974*, 2021.

[68] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[69] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[70] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and system co-design for efficient subgraph-based graph representation learning. *Proceedings of the VLDB Endowment*, 15(11):2788–2796, 2022.

[71] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning (ICML)*, 2018.

[72] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[73] Lingxiao Zhao, Xueying Ding, and Leman Akoglu. Pard: Permutation-invariant autoregressive diffusion for graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[74] Qifang Zhao, Weidong Ren, Tianyu Li, Xiaoxiao Xu, and Hong Liu. Graphgpt: Graph learning with generative pre-trained transformers. *arXiv preprint arXiv:2401.00529*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are fully justified in our theoretical and empirical results in Section 2 and 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Appendix 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The background for the theoretical results is provided in Section C.1. The full set of assumptions and complete proofs is provided in Appendix D.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experimental details, including dataset description, evaluation metrics, computing details, and hyperparameter choices, are provided in Appendix E. We will release the full code upon publication.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the full code and documentation upon publication.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details are provided in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: While reporting error bars is not yet standard in the field of graph generation, we report the error bars on small synthetic datasets in Appendix F.1 to mitigate the impact of small test sample size.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computing details are provided in Appendix E.3. Runtime to reproduce some experiments is given in Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both aspects in Appendix B.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work focuses on generative modeling for general graphs, but does not release ready-to-use models for real-world applications.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited the code, data, and models used in this study (see Appendix E.1).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new NetworkX dataset and its generation is fully documented in Appendix E.1. It conforms with the License of the NetworkX library.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

This appendix provides both theoretical and experimental materials. It is organized as follows: Section A provides additional background on sequence models. Section B provides a discussion about the broader impact of this work. Section C provides additional details and remarks on our method. Section D provides proofs for the theorems presented in the main manuscript. Section E provides experimental details. Section F provides additional quantitative and qualitative results.

A Background on Sequence Model Architectures

Our sequence model architectures are fully based on established natural language models. In particular, we consider three prominent models, including GPT-2 [54], LLaMA [63, 64], and Mamba [25] to demonstrate the effectiveness of our approach. Notably, our methodology is not restricted to these specific models; it can be applied to any sequence or language model.

GPT-2. GPT-2 represents one of the earliest large language models based on the transformer architecture [65]. The model employs pre-normalization with LayerNorm, the GeLU activation function, and absolute positional embeddings to encode token positions in sequences. These design choices laid the foundation for many subsequent models.

LLaMA. LLaMA [63, 64] builds upon the transformer framework with several key enhancements. It incorporates pre-normalization through RMSNorm [72] and employs the SwiGLU activation function [56]. Additionally, LLaMA replaces absolute positional embeddings with rotary positional embeddings [59], enabling better generalization to longer sequences.

Mamba. Mamba [25] is a state-space model (SSM) that maps input sequences to outputs using continuous-time dynamics. It introduces a selection mechanism that dynamically controls how input data flows into hidden states, making the model parameters adaptive to time and data. This innovation enables Mamba to achieve superior performance compared to other SSMs across various tasks.

B Broader Impacts

Our research focuses on advancing the algorithmic development of graph generative models, strongly emphasizing their responsible and ethical application in specialized fields. In domains such as drug discovery and synthetic biology, ensuring the trustworthiness and appropriate use of our methods is essential to prevent potential misuse. Through our experiments, we showcase the potential of our approach in these fields, underscoring its promise to deliver meaningful societal benefits while acknowledging the need to address potential risks.

C Additional Details about AUTOGRAPH

C.1 Background on Graph Theory

We provide additional background on graph theory necessary for the definitions and theories of SETs and SENTs. The background is largely based on Diestel [19].

We first give the formal definition of *graph isomorphism*:

Definition C.1 (Graph isomorphism)**.** An isomorphism of graphs G and H is a bijection between the vertex sets of G and H: $\pi : V_G \to V_H$ such that any two vertices u and v of G are adjacent in G if and only if $\pi(u)$ and $\pi(v)$ are adjacent in H, *i.e.,* $(u, v) \in E_G$ if and only if $(\pi(u), \pi(v)) \in E_H$.

Graph isomorphism is an equivalence relation on graphs and as such it partitions the class of all graphs into equivalence classes. A set of graphs isomorphic to each other is called an isomorphism

class of graphs. It is worth noting that our SENT isomorphism also partitions the class of all SENTs into equivalence classes in a similar fashion.

We also provide the formal definition of *induced subgraph*:

Definition C.2 (Induced subgraph)**.** An induced subgraph of a graph is another graph, formed from a subset of the vertices of the graph and all of the edges, from the original graph, connecting pairs of vertices in that subset. Formally, let $S \subseteq V_G$ be any subset of vertices of $G := (V_G, E_G)$. Then, the induced subgraph $G[S]$ is the graph whose vertex set is S and whose edge set consists of all of the edges in E_G that have endpoints in S. That is, for any two vertices $u, v \in S$, $(u, v) \in E_{G[S]}$ if and only if $(u, v) \in E_G$.

C.2 Remarks on Tokenized SENTs

In Section 2.4, we showed that an (ordered) SENT can be converted into a sequence of tokens. Here, we extend this idea by interpreting the tokenized sequence as a random walk on a slightly modified graph. We first introduce an alternative tokenization scheme that is equivalent to the one described earlier but offers enhanced interpretability. The proposed tokenization remains largely unchanged except for how tuples are handled. For each $w = (v, A)$ with $A = \{u_1, \ldots, u_p\}$, we now define

$$\texttt{Token}(w) := [v, <, u_1, <, u_2 \ldots, <, u_p, >] .$$

We now detail how to modify the original graph G: we introduce three virtual nodes, labeled /, $<$, and $>$ respectively. These virtual nodes are connected to all other virtual nodes and original nodes in the graph. This modification ensures that for any non-special token in the tokenized sequence, its subsequent token can either be one of its neighbors or one of the virtual nodes (/, $<$, or $>$). Consequently, each token has a direct connection to the node corresponding to the current token, and the language model amounts to learning the state transition functions for these random walks. Since these random walks are non-Markovian, this perspective further justifies our choice of using autoregressive models instead of one-step generative models. Furthermore, as random walks are random sequences on graphs, sampling random walks amounts to sampling from those random sequences.

C.3 Remarks on Model Inference

The model inference is straightforward following the same process as LLMs such as LLaMA [63, 64]. An alternative way is to enforce the semantic correctness of the generated sequences of tokens by adjusting the logits at a certain token to obey the semantic rule of the tokenization. For instance, the token '$>$' can only occur after a token '$<$' or no special tokens can appear right after /. We manually implemented these transition constraints and incorporated them into the inference. We compared this strategy with the constraint-free counterpart. Surprisingly, our experiments demonstrate that the constraint-free variant could always generate semantically correct tokenized SENTs and perform similarly to the one with the transition constraints. Therefore, we did not use any transition constraints during the inference in our experiments.

D Proofs

In this section, we provide proof for the theorems stated in the manuscript.

Theorem 2.5. *For any SETs $s, t \in \mathcal{S}$, their generated graphs are isomorphic, i.e., $G_s \simeq G_t$, if $s \simeq t$. Conversely, if two graphs $G \simeq H$, then for any SET $s \in \mathcal{S}_G$, there exists a SET $t \in \mathcal{S}_H$ s.t. $s \simeq t$.*

Proof of Theorem 2.5. By definition of the isomorphism between s and t, there exists a bijection $\pi : V_s \to V_t$ *s.t.* $\pi(s) = t$. Now if $u, v \in V_s$ are adjacent in G_s, *i.e.,* $(u, v) \in E_s$, then $(\pi(u), \pi(v))$ is an edge visited by $\pi(s) = t$, thus $(\pi(u), \pi(v)) \in E_t$. Similarly, the reverse is also true. Consequently, $G_s \simeq G_t$.

Now assume that $G \simeq H$ with an isomorphism π and $s \in \mathcal{S}_G$. It is easy to show that $\pi(s)$ is also a SET and its generated graph $G_{\pi(s)} = H$. By taking $t = \pi(s)$, we obtain the result. □

Lemma 2.8. *For any graph G and SET s in G, the generated graph of any prefix of s is a subgraph of G, but not necessarily an induced subgraph.*

Proof of Lemma 2.8. Assume that t is a prefix of s. Then $V_t \subseteq V_s = V_G$ and $E_t \subseteq E_s$. However, G_t is not necessarily an induced subgraph of G_s. We consider the following counter-example: $s = ((1,2,3,4,1,3))$, $V_s = \{1,2,3,4\}$, and $E_s = \{(1,2),(2,3),(3,4),(1,4),(1,3)\}$. Let $t = ((1,2,3,4,1))$. t is clearly a prefix of s, but its generated graph is not an induced subgraph of G_s as its generated edge set does contain $(1,3)$. □

Theorem 2.14. *For any causal SENT $s \in \mathcal{S}^{\mathcal{N}}$, the generated graph of any prefix t of s is an induced subgraph of G_s if and only if s is semi-hamiltonian. In this case, s is called subgraph-induced.*

Proof of Theorem 2.14. Let us first introduce some notations. We denote by R_s the sequence of the start tuples across all neighborhood trails in s, which is also a neighborhood sequence. By definition of semi-hamiltonian, the occurrences after the first time of a node in s should be in R_s. We denote by $n(s)$ the associated node sequence of SENT s, *i.e.,* $n(s) := n(\|s)$.

Let us first assume that s is semi-hamiltonian.

Assume that t is a prefix of s. It is easy to show that G_t is a subgraph of G_s. Now assume that $u, v \in V_t$ *s.t.* $(u,v) \in E_s$, we want to show that $(u,v) \in E_t$. There are two cases:

1) Assume that $u, v \in n(t)$. Since s is semi-hamiltonian, $n(s) \setminus n(t)$ either does not contain u or v, or even if one of them, say $u \in n(s) \setminus n(t)$, we have $u \in n(R_s)$ and its associated neighborhood set is empty. In both cases, the edge (u,v) does not belong to the generated edge set of the neighborhood subsequence after $\|t$. By the disjointness of the generated edge sets of s, it can only be included in the generated edge set of t, we thus have $(u,v) \in E_t$.

2) Assume that one of them, say $u \notin n(t)$. There exists a neighborhood set A in a tuple of $\|t$ such as $u \in A$. Since t is causal, we have $u \in n(t)$ which contradicts the assumption.

In all the above cases, we have $(u,v) \in E_t$.

Now let us assume that the generated graph of any prefix of s is an induced subgraph of G_s.

Let us prove that s is semi-hamiltonian by contradiction. Assume that there exist two tuples in $\|s$ with the same nodes $s_i = (v, A_i)$ and $s_j = (v, A_j)$ with $i < j$. There are two cases: 1) $s_j \notin R_s$. A tuple (u, A_u) exists one step before s_j in the same neighborhood trail. We consider the prefix t ending at (u, A_u). We have $v, u \in V_t$ and $(u,v) \in E_s$, but $(u,v) \notin E_t$, by the disjointness of s and since (u,v) is visited at s_j after t. 2) $s_j \notin R_s$ and $A_j \neq \emptyset$. Since s is causal, there exists $s_u := (u, A_u)$ before s_j *s.t.* $u \in A_j$. We consider the prefix t ending at exactly this tuple. We have $u, v \in V_t$ and $(u,v) \in E_s$, but $(u,v) \notin E_t$, by the disjointness of s and since (u,v) is an edge visited at (v, A_j) after t. □

Theorem 2.15. *For $s \in \mathcal{S}_G^{\mathcal{N}}$, s is causal and Hamiltonian if and only if every tuple $w := (v, A_v)$ in $\|s$ satisfies $A_v = \mathcal{N}_G(v) \cap V_s(w)$. In this case, every node is visited exactly once. Moreover, s is causal and semi-hamiltonian if and only if every tuple $w := (v, A_v)$ in s satisfies either $A_v = \mathcal{N}_G(v) \cap V_s(w)$ or $A_v = \emptyset$.*

Proof of Theorem 2.15. Let us first assume that for any tuple $w := (v, A_v)$ in $\|s$, $A_v = \mathcal{N}_G(v) \cap V_s(w)$. Since $A_v \subseteq V_s(w)$ which is a subset of the set of visited nodes, s is causal. Now we prove s is Hamiltonian by contradiction. Assume that there exist two tuples in $\|s$, $s_u := (u, A_u)$ and a later visited one $s_v := (v, A_v)$ *s.t.* $u = v$. Then, $A_v = \mathcal{N}_G(v) \cap V_s(s_v) = \mathcal{N}_G(u) \cap V_s(s_v)$ should contain the node visited before that is a neighbor of u (either through a trail or the neighborhood set of u), denoted by u'. Thus, the edge (u, u') has been visited twice, which contradicts the disjointness of s.

Assuming that $A_v = \mathcal{N}_G(v) \cap V_s(w)$ or $A_v = \emptyset$ for any tuple (v, A_v) in s, we can also prove s is semi-hamiltonian by contradiction. Assume that there exist two tuples in $\|s$, $s_u := (u, A_u)$ and a later visited one $s_v := (v, A_v)$ *s.t.* $u = v$ and $A_v \neq \emptyset$. Then, $A_v = \mathcal{N}_G(v) \cap V_s(s_v) = \mathcal{N}_G(u) \cap V_s(s_v)$ by assumption. And using the same argument as above, we have a contradiction.

Now assume that s is causal and Hamiltonian. Let us prove the other direction by contradiction. There exists a tuple $w := (v, A_v)$ in s *s.t.* $A_v \neq \mathcal{N}_G(v) \cap V_s(w)$. As s is causal, $A_v \subseteq V_s(w)$. $A_v \subseteq \mathcal{N}_G(v)$ as $s \in \mathcal{S}_G^{\mathcal{N}}$. Thus, $A_v \subset \mathcal{N}_G(v) \cap V_s(w)$, which means that there exists $u \in \mathcal{N}_G(v) \cap V_s(w)$ and $u \notin A_v$. Hence, $(u,v) \in E_G$ and u is visited before v. However, as $u \notin A_v$, $(u,v) \in E_G$, and s is Hamiltonian, there exists a tuple (u, A_u) in $\|s$ *s.t.* $v \in A_u$. By causality of s, v is visited before u, which contradicts the fact that s is Hamiltonian.

Assuming that s is causal and semi-hamiltonian. Let us prove the other direction by contradiction. There exists a tuple $w := (v, A_v)$ in s *s.t.* $A_v \neq \mathcal{N}_G(v) \cap V_s(w)$ and $A_v \neq \emptyset$. Using the same arguments as above, there exists $(u, v) \in E_G$, and u is visited before v. However, as $u \notin A_v$ and $(u, v) \in E_G$, s should visit the edge (u, v) at some point. Since s is semi-hamiltonian, if s visits again u, v they can only be the first nodes and their associated neighborhood sets are empty. Hence, there is no means for s to visit (u, v) after v, leading to a contradiction. □

E Experimental Details

E.1 Datasets

We provide details of the datasets used in our experiments. we adopt the standard train/validation/test splits provided in the original sources. The statistics about the datasets are summarized in Table 8.

Small synthetic graphs: Planar and SBM. Both of these datasets are from Martinkus et al. [45]. The Planar dataset consists of 200 planar graphs with 64 nodes each, generated via Delaunay triangulation on points uniformly sampled in the unit square. The SBM dataset contains 200 graphs comprising 2 to 5 communities, with each community having between 20 and 40 nodes. An edge is placed between two nodes with probability 0.3 if they belong to the same community, and 0.05 otherwise. We follow the same splits as Martinkus et al. [45].

Large graphs: Proteins and Point Clouds. The Proteins dataset includes graph representations (contact maps) of proteins from Dobson and Doig [20]. In these graphs, each node represents an amino acid, and an edge connects two nodes if their corresponding amino acids are within 6 angstroms of each other. We use the same data splits as Liao et al. [43]. The Point Clouds dataset, also from Liao et al. [43], consists of 41 point clouds of household objects [49]. As many of these graphs are disconnected, we retain only the largest connected component of each, following Bergmeister et al. [2], and again employ the splits used by Liao et al. [43].

QM9. The QM9 dataset, from Wu et al. [68], comprises small molecules with up to nine heavy atoms (carbon, oxygen, nitrogen, and fluorine). In this work, we adopt the more challenging setting proposed by Vignac et al. [66], where hydrogen atoms are modeled explicitly, and we follow the same data splits as in that reference.

MOSES and GuacaMol. The MOSES and GuacaMol datasets are obtained from the respective benchmark tools of Polykovskiy et al. [53] and Brown et al. [6]. Both consist of drug-like molecules, with those in GuacaMol typically being larger on average. For each dataset, we convert generated molecular graphs to SMILES using the code from Jo et al. [34], which permits partial charges. We employ the standard data splits provided by the corresponding benchmarks.

PubChem-10M. PubChem-10M is a subset of about 10M molecules from PubChem curated by Chithrananda et al. [14].

NetworkX. We generate the graphs using the generators from the NetworkX library[2] [26], categories including "Classic", "Lattice", "Small", "Random Graphs", "Geometric", "Trees", "Community", "Social Networks". We ensure that this dataset *does not contain any graphs in the downstream datasets*. The summary of the code for generating these graphs is provided in Table 9. Notably, the largest graph has up to 5999 nodes.

E.2 Evaluation Metrics

We follow Martinkus et al. [45] and Vignac et al. [66] in comparing our model's performance with other graph generative approaches. Specifically, we measure the maximum mean discrepancy (MMD) between the generated and test graphs for degree distribution, clustering coefficient, orbit counts, and spectrum. As a reference, we also compute these metrics on the training set and report the mean ratio across all properties as a global indicator of statistical discrepancy between the generated samples

[2] `https://networkx.org/documentation/stable/reference/generators.html`

Table 8: Dataset statistics

Dataset	n_{graphs}			$\lvert V\rvert_{\max}$	$\lvert V\rvert_{\text{avg}}$	$\lvert E\rvert_{\max}$	$\lvert E\rvert_{\text{avg}}$
	Train	Val	Test				
Unattributed Graphs							
Planar	128	32	40	64	64	181	178
SBM	128	32	40	187	104	1129	500
Proteins	587	147	184	500	258	1575	646
Point Clouds	26	7	8	5037	1332	10886	2971
Attributed Graphs							
QM9	97734	20042	13055	29	18	28	19
MOSES	1584663	176225	176074	27	22	31	23
GuacaMol	1118633	69926	209654	88	28	88	30
Pre-training unattributed graphs							
NetworkX	24957	2516	—	5999	459	5999	751

Table 9: Summary of the code for generating graphs in the NetworkX dataset

Generator	n_{graphs}	Python Code
Category: Classic		
balanced tree	10	`nx.balanced_tree(2, np.random.randint(4, 10))`
barbell graph	100	`nx.barbell_graph(np.random.randint(3, 31), np.random.randint(41))`
binomial tree	10	`nx.binomial_tree(np.random.randint(2, 9))`
complete graph	10	`nx.complete_graph(np.random.randint(3, 31))`
circular ladder graph	300	`nx.circular_ladder_graph(np.random.randint(10, 501))`
cycle graph	2000	`nx.cycle_graph(np.random.randint(10, 6001))`
Dorogovtsev Goltsev Mendes graph	5	`nx.dorogovtsev_goltsev_mendes_graph(np.random.randint(2, 7))`
ladder graph	500	`nx.ladder_graph(np.random.randint(10, 1001))`
lollipop graph	200	`nx.lollipop_graph(np.random.randint(3, 21), np.random.randint(10, 51))`
star graph	200	`nx.star_graph(np.random.randint(10, 501))`
turan graph	100	`nx.turan_graph(np.random.randint(10, 41), 2)`
wheel graph	100	`nx.wheel_graph(np.random.randint(10, 201))`
Category: Lattices		
grid 2d graph	400	`nx.grid_2d_graph(np.random.randint(5, 31), np.random.randint(5, 31))`
triangular lattice graph	400	`nx.triangular_lattice_graph(np.random.randint(5, 41), np.random.randint(5, 41))`
Category: Small		
all but the LCF graph	1 (each)	`nx.{method}()`
Category: Random graphs		
Erdos Renyi graph	4000	`nx.erdos_renyi_graph(np.random.randint(20, 101), 0.2)`
random regular graph	2000	`nx.random_regular_graph(np.random.randint(3, 11), np.random.choice([20,30,...,500]))`
Barabasi Albert graph	4000	`nx.barabasi_albert_graph(np.random.randint(20, 501), np.random.randint(2, 6))`
random lobster	4000	`nx.random_lobster(80, 0.7, 0.7)`
Category: Geometric		
random geometric graph	3000	`nx.random_geometric_graph(np.random.choice([20,30,...,100]), 0.3)`
Waxman graph	2000	`nx.waxman_graph(np.random.choice([50,100,150,...,300]))`
Category: Trees		
random unlabeled tree	1000	`nx.random_unlabeled_tree(np.random.randint(20, 501))`
Category: Community		
connected Caveman graph	300	`nx.connected_caveman_graph(np.random.randint(10, 101), np.random.randint(2, 5))`
Windmill graph	300	`nx.windmill_graph(np.random.randint(10, 101), np.random.randint(2, 5))`
Category: Social networks		
All social networks	1 (each)	`nx.{method}()`

and test samples. Note that for the Point Clouds dataset, which is defined by a k-nearest-neighbor structure, the degree MMD is always zero and is therefore excluded from the mean ratio. While we utilize these metrics to maintain consistency with previous research, we acknowledge their limitations, particularly regarding arbitrary kernel hyperparameter selection, as highlighted by O'Bray et al. [52], Thompson et al. [60]. In short, MMD measures the distributional similarity between generated and real graphs. Lower MMD scores indicate that the generated graphs' statistics (e.g., degree, clustering coefficients) more closely match the training data.

We additionally track uniqueness and novelty: *uniqueness* is the fraction of generated graphs that are not isomorphic to each other, and *novelty* is the fraction of generated graphs that are not isomorphic to any training graph.

Below, we describe additional metrics specific to each dataset.

Planar and SBM. Following Martinkus et al. [45], we report a *validity score* for synthetic datasets. For Planar graphs, it verifies whether the generated graphs remain planar; for SBM graphs, it measures how likely they are to be generated under the original SBM parameters. We integrate validity, novelty, and uniqueness into a single metric, VUN, which measures the fraction of generated graphs that are simultaneously valid, novel, and unique. In short, VUN assesses sample quality.

QM9. For QM9, we report the *validity*, *uniqueness*, and *novelty* defined for general molecules, as described in the following paragraph. We also report *atom stability* and *molecule stability* as defined by Hoogeboom et al. [27] and Vignac et al. [66].

MOSES and GuacaMol. Since MOSES [53] and GuacaMol [6] are benchmarking platforms, each comes with its own suite of metrics, which we use to evaluate our model. These include:

- **Validity**: Proportion of molecules passing basic valency checks.
- **Uniqueness**: Proportion of generated molecules with distinct SMILES strings (indicating non-isomorphic structures).
- **Novelty**: Proportion of generated molecules not present in the training set.
- **Filter score**: Proportion of molecules passing the same filters used to create the test set.
- **Fréchet ChemNet Distance (FCD)**: Similarity measure between generated and training sets based on learned neural embeddings.
- **SNN**: Similarity to the nearest neighbor, computed via Tanimoto distance.
- **Scaffold similarity**: Comparison of Bemis–Murcko scaffold frequencies.
- **KL divergence**: Differences in the distributions of various physicochemical descriptors.

E.3 Computing Details

We implemented our sequence models using the model hub of Hugging Face. Users can easily test their preferred sequence or language models using our code. Experiments were conducted on a shared computing cluster with various CPU and GPU configurations, including 16 NVIDIA H100 (80GB) GPUs. Each experiment was allocated resources on a single GPU, along with 8 CPUs and up to 48GB of system RAM. The run-time of each model was measured on a single NVIDIA H100 GPU.

E.4 Hyperparameters

Unlike prior studies that adjust model sizes across datasets, we maintain a consistent model architecture and size throughout all experiments, specifically using the small GPT configuration (768 hidden dimensions, 12 layers, 12 attention heads). Training hyperparameters are aligned with established practices from popular LLMs such as GPT-3 [7] and LLaMA [63]. We fix the context length to 2048 and use a batch size of 128 if possible, otherwise 64 for larger graphs. In particular, we employ the AdamW optimizer with a gradient clipping threshold of 1.0, a weight decay of 0.1, and a learning rate schedule with a linear warmup followed by cosine decay, peaking at 6e-4. The AdamW hyperparameters are set to $\beta = (0.9, 0.95)$. Due to the small dataset sizes of previous benchmarks, we tune the only training hyperparameter dropout in $\{0, 0.5\}$, and find the model achieves better validation loss with the value of 0.5 on the small synthetic datasets. Each model was trained for 200000, 400000, or 800000 iterations, depending on the dataset size.

Inference hyperparameters, including k (top-k sampling) and τ (temperature), are reported in Table 10 and analyzed in detail in Section F.5.

F Additional Results

F.1 Additional Results on Synthetic Datasets

Due to the small number of samples in the Planar and SBM datasets, we observe significant variances in evaluation metrics. In order to mitigate the impact of the small test samples on the evaluation, we use trained models to generate samples with 10 different seeds and report the average metrics and error bars for DiGress, ESGG, and our AUTOGRAPH. For DiGress and ESGG, we use either pretrained models released by the authors, if available, or our reproduced models using their officially released code repository. As shown in Table 11, the variances appear reasonable, and the conclusions remain the same.

Table 10: Inference hyperparameters for each dataset.

	W/O PRE-TRAINING		W/ PRE-TRAINING	
DATASET	TOP-k	TEMPERATURE τ	TOP-k	TEMPERATURE τ
PLANAR	10	1.0	30	0.9
SBM	60	1.0	150	1.0
PROTEINS	40	1.0	30	1.05
POINT CLOUDS	60	1.0	20	0.9
NETWORKX	120	1.0	—	—
QM9	5	1.0	—	—
MOSES	5	1.0	—	—
GUACAMOL	5	1.0	—	—

Table 11: Performances on the Planar and SBM datasets with error bars

(a) Planar

MODEL	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN
DIGRESS	0.0003±0.0002	0.0415±0.0081	0.0056±0.0028	0.0078±0.0010	3.8±1.4	79.0±6.0
ESGG	0.0006±0.0004	0.0434±0.0154	0.0101±0.0073	0.0091±0.0018	6.3±4.0	**90.5±5.6**
AUTOGRAPH	0.0004±0.0003	0.0533±0.0083	0.0005±0.0004	0.0066±0.0009	**1.6±0.5**	80.3±6.8

(b) SBM

MODEL	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN
DIGRESS	0.0013±0.0009	0.0501±0.0009	0.0393±0.0104	0.0053±0.0007	**1.6±0.3**	66.0±5.6
ESGG	0.0468±0.0096	0.0554±0.0013	0.0699±0.0051	0.0085±0.0011	15.3±3.1	16.0±4.7
AUTOGRAPH	0.0081±0.0051	0.0525±0.0015	0.0687±0.0138	0.0048±0.0011	3.9±1.6	**88.3±4.7**

F.2 Additional Results on molecular generation datasets

Due to space constraints, we provide results on the QM9 dataset in Table 12, and additional metrics on the MOSES dataset in Table 13. It is worth noting that all results on MOSES and GuacaMol were obtained using the benchmarking tools from the original works, which might rely on outdated packages. For instance, using the latest FCD package gives a FCD score of 94.7 for the pre-trained AUTOGRAPH.

Our results on all three benchmarks demonstrate the immense potential of AUTOGRAPH for molecular generation. The fact that AutoGraph, by learning from graph data alone, can achieve validity scores competitive with these specialized models is a strong demonstration of its learning capabilities. It successfully infers the complex, implicit rules of molecular construction. Furthermore, some models, such as MCTS a search-based method, that guarantee 100% validity do so at the cost of poor distributional similarity (e.g., a low FCD score). From a practical perspective, one can easily reject invalid graphs to achieve near-perfect validity while generating over-simplified and basic molecules that have low distributional similarity (FCD score) with the training data, which is of low practical interest. AutoGraph achieves a superior overall balance.

Table 12: Benchmarking AUTOGRAPH on the QM9 dataset

	QM9 WITH HYDROGEN ATOMS $n_{\text{graphs}} = 100\text{K}, \lvert V\rvert_{\max} = 29, \lvert V\rvert_{\text{avg}} \approx 18$				
MODEL	VALID↑	UNIQUE↑	NOVEL↑	ATOM STABLE↑	MOL STABLE↑
DIGRESS	95.4	**97.6**	33.4	98.1	79.8
AUTOGRAPH	**97.7**	96.7	**45.5**	**98.6**	**87.3**

Table 13: Benchmarking AUTOGRAPH on the MOSES dataset

		MOSES $n_{\text{graphs}} = 1.58\text{M}$, $\lvert V\rvert_{\max} = 27$, $\lvert V\rvert_{\text{avg}} \approx 22$						
MODEL	TYPE	VALID↑	UNIQUE↑	NOVEL↑	FILTERS↑	FCD↓	SNN↓	SCAF↑
VAE	SMILES	97.7	99.8	69.5	99.7	0.57	0.58	5.9
JT-VAE	FRAGMENTS	100	100	99.9	97.8	1.00	0.53	10
GRAPHINVENT	GRAPH	96.4	99.8	–	95.0	1.22	0.54	12.7
DIGRESS	GRAPH	85.7	100	95.0	97.1	1.19	0.52	14.8
AUTOGRAPH	GRAPH	87.4	100	85.9	98.6	0.91	0.55	10.2

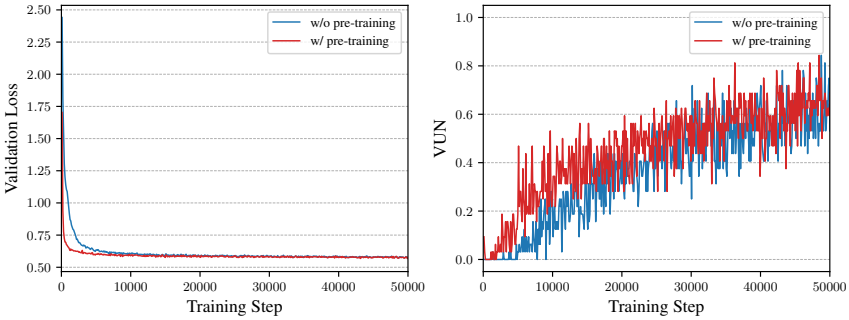

Figure 3: Comparison of AUTOGRAPH with and without pre-training on the Planar dataset with 50000 training steps. The model with pre-training converges clearly faster than the model without pre-training.

F.3 Transfer Performance of AUTOGRAPH

We provide here additional results for the transfer learning of AUTOGRAPH. We compare the training curves of AUTOGRAPH models with and without pre-training on the Planar datasets in Figure 3. The result suggests that the model with pre-training converges clearly faster.

For the transfer experiment on molecular generation, we first pre-train AUTOGRAPH on the PubChem-10M dataset [14], and then fine-tune it on the GuacaMol dataset. In this experiment, we use richer node attributes including atom types, total number of hydrogens, and formal charges.

F.4 Substructure Conditioned Generation

As presented in Section 4.3, we test more extreme cases by replicating the same motif multiple times before initiating the conditional generation. Figure 4, 5, and 6 demonstrate non-curated samples generated by AUTOGRAPH (trained on the GuacaMol dataset without any additional fine-tuning) conditioned on p copies of the same motif, where $p = 1, 2, 5$ respectively.

To further showcase the flexibility of AUTOGRAPH, we conduct the same experiments for two different motifs: 1,4-Dihydroquinoline[3] and 3-(Trifluoromethyl)aniline[4]. This is a very relevant problem in drug discovery, usually termed linker design. The validity, uniqueness, and novelty for 1024 samples are respectively 97.4, 81.4, and 99.9. Visual examples are given in Figure 7.

F.5 Additional Ablation Experiments

Impact of model size. Table 14 and Figure 8 compare the impact of model size. Larger models demonstrate better VUN scores. In this work, we use LLaMA-s in all our experiments to balance the trade-off between performance and speed.

[3] `https://pubchem.ncbi.nlm.nih.gov/compound/1_4-Dihydroquinoline`

[4] `https://pubchem.ncbi.nlm.nih.gov/compound/3-_Trifluoromethyl_aniline`

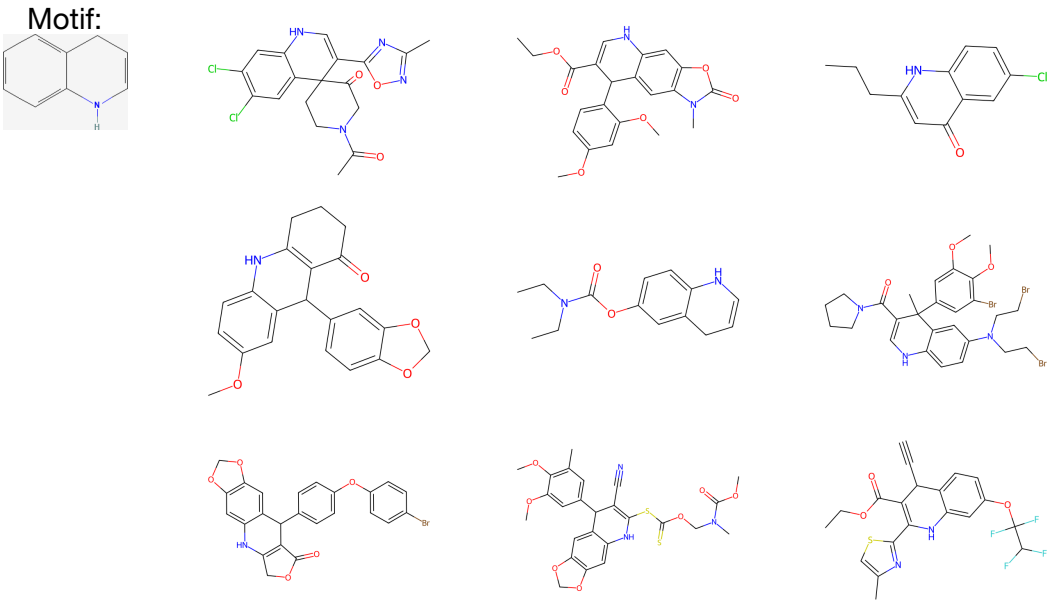

Figure 4: Substructure conditioned generation on one copy of the motif 1_4-Dihydroquinoline.

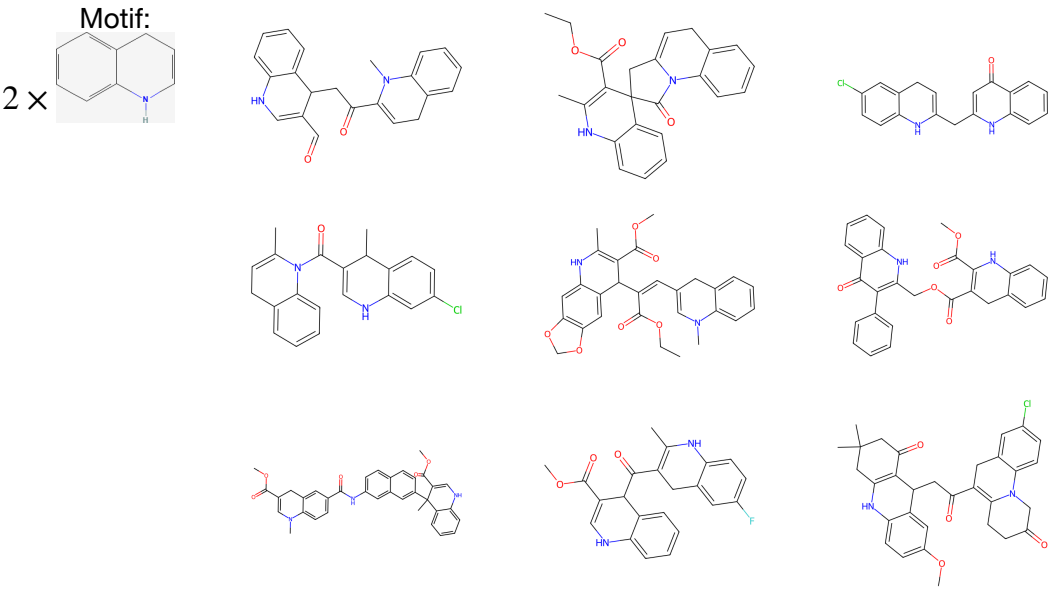

Figure 5: Substructure conditioned generation on two copies of the motif 1_4-Dihydroquinoline.

F.6 Visualization of Graphs Generated by AUTOGRAPH

F.6.1 Results without Pre-training

We provide visualization of non-curated samples generated by AUTOGRAPH without pre-training on all datasets in Figure 9, 10, 11, 12, 13, 14, and 15. The results on NetworkX are illustrated in Figure 16. Node colors in unattributed graphs represent the eigenvectors associated with the second-smallest eigenvalues of the graph Laplacian.

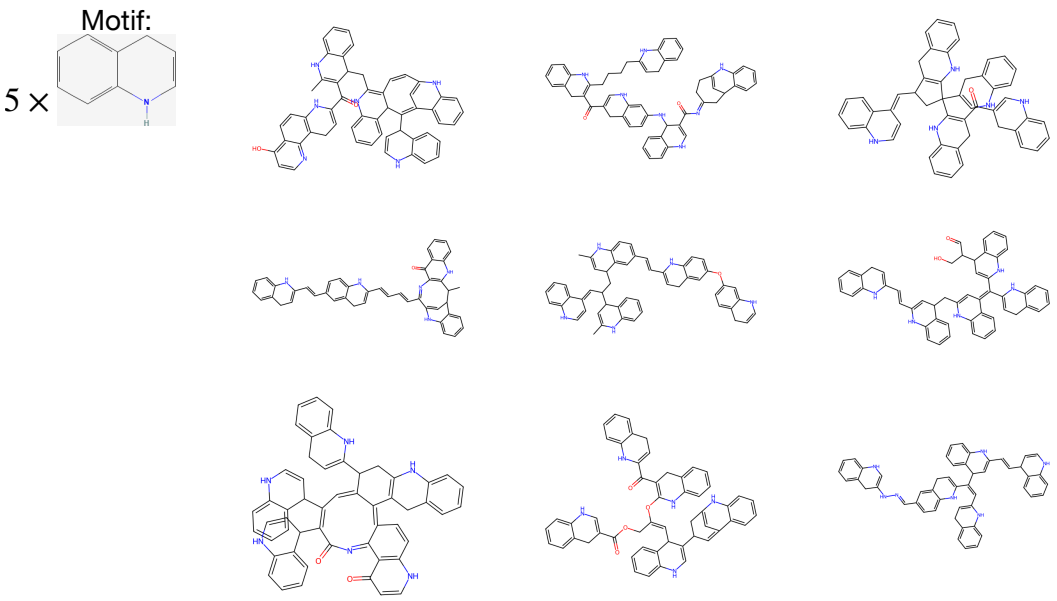

Figure 6: Substructure conditioned generation on five copies of the motif 1_4-Dihydroquinoline.

Table 14: Comparison of model size on the Planar dataset

MODEL	# PARAMS	CONFIGURATION	DEG.	CLUS.	ORBIT	SPEC.	RATIO	VUN
LLAMA-XS	25.2M	6 LAYERS, 512 DIMS	-0.0001	0.0570	0.0006	0.0063	**1.0**	60.0
LLAMA-S	113M	12 LAYERS, 768 DIMS	0.0005	0.0651	0.0005	0.0056	1.6	**90.0**
LLAMA-M	402M	24 LAYERS, 1024 DIMS	0.0001	0.0340	0.0013	0.0064	1.4	82.5

F.6.2 Results with pre-training

We provide visualization of non-curated samples generated by AUTOGRAPH with pre-training (on the NetworkX dataset) trained on the non-attributed datasets including Planar, SBM, Proteins, and Point Clouds, illustrated in Figure 17, 18, 19, 20.

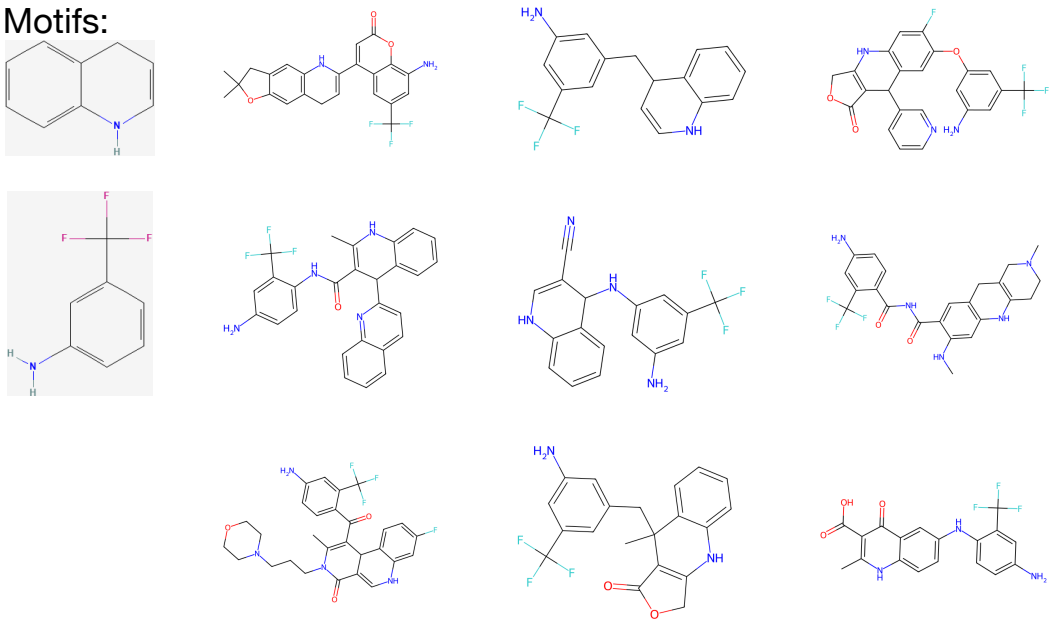

Figure 7: Substructure conditioned generation on two different motifs: 1_4-Dihydroquinoline and 3-(Trifluoromethyl)aniline.

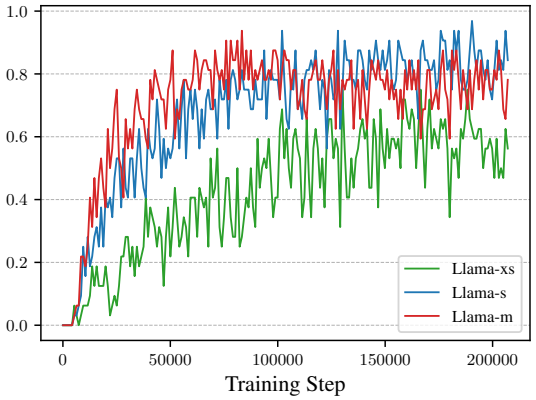

Figure 8: Comparison of model size on the Planar dataset: VUN score vs training steps. LLaMA-m appears to suffer from overfitting and LLaMA-xs appears to suffer from underfitting.

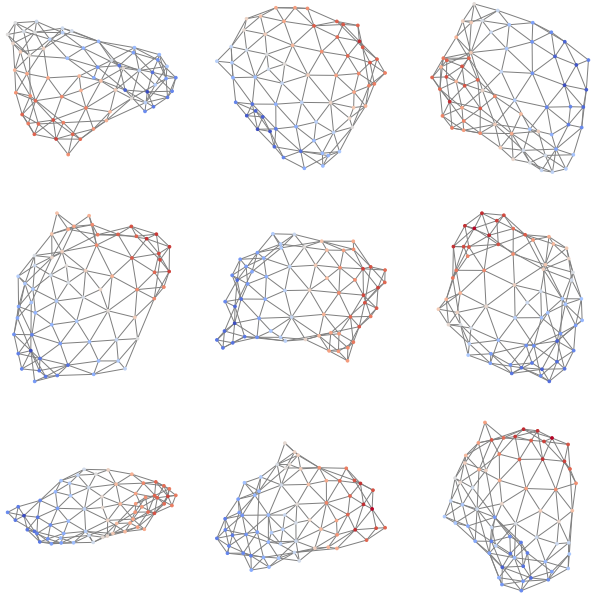

Figure 9: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the Planar dataset.

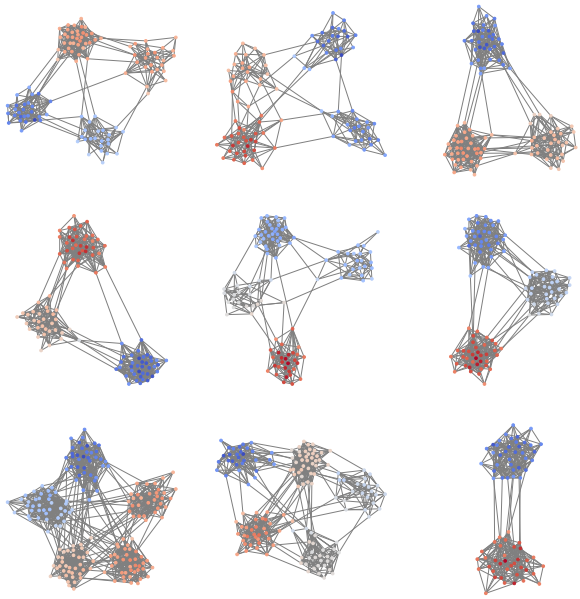

Figure 10: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the SBM dataset.

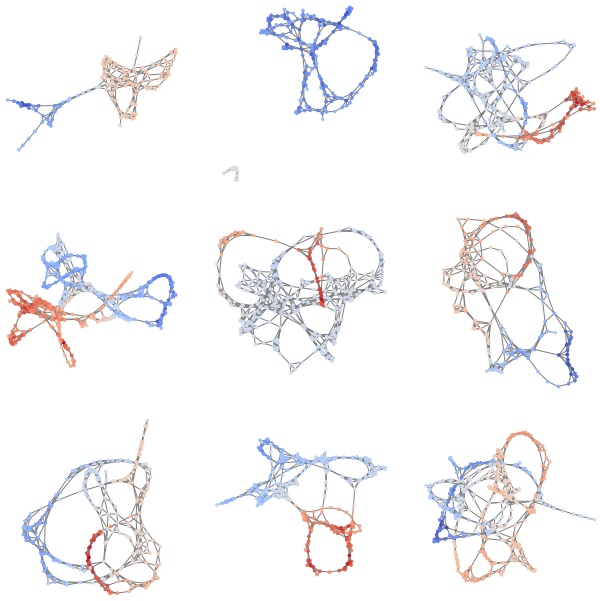

Figure 11: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the Proteins dataset.

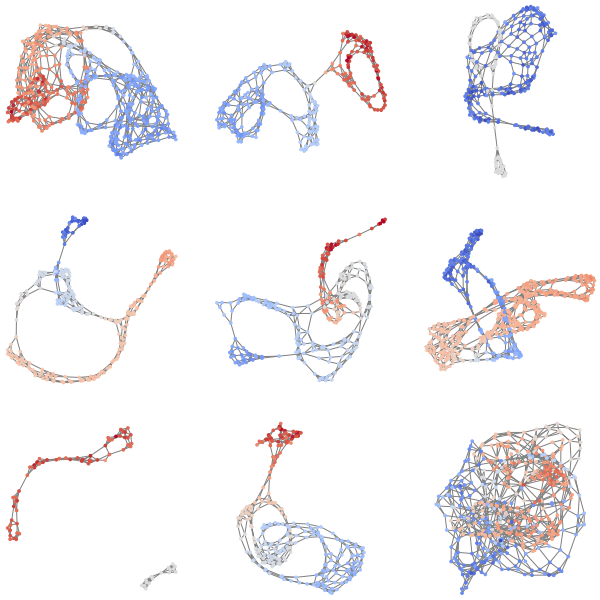

Figure 12: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the Point Clouds dataset.

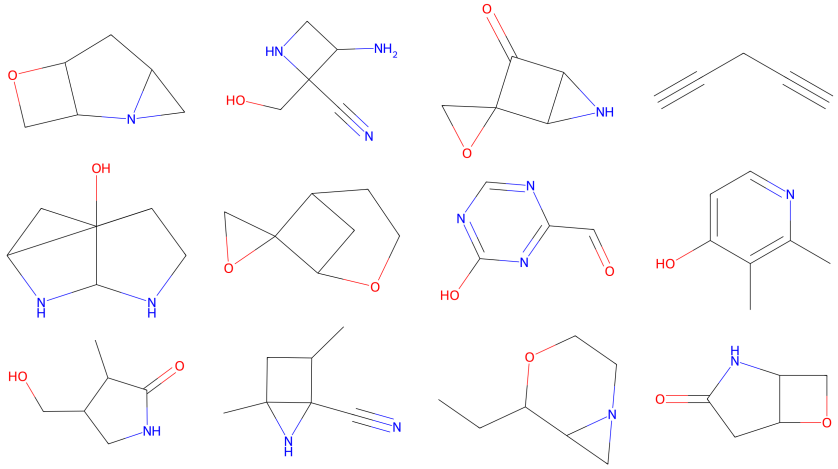

Figure 13: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the QM9 dataset.

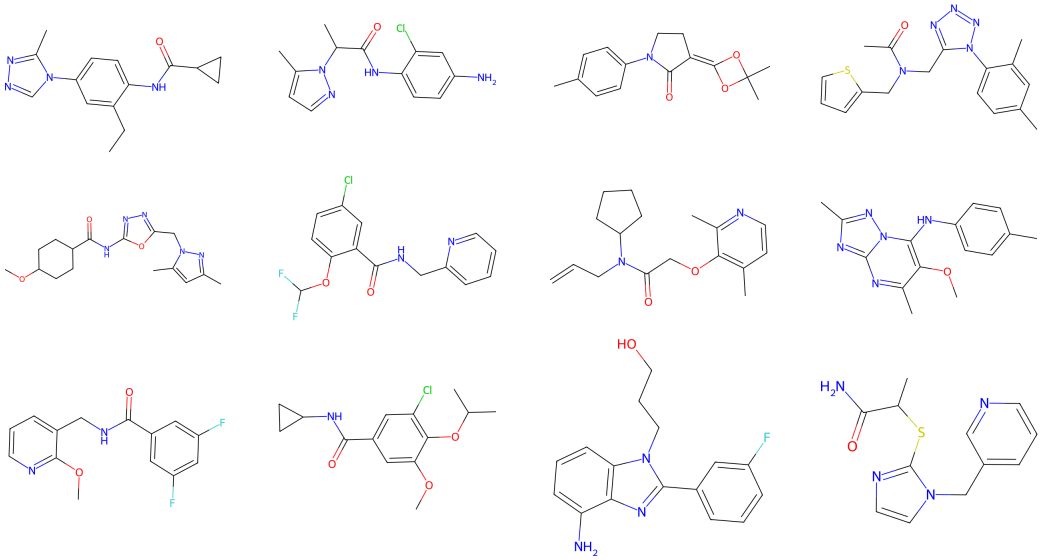

Figure 14: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the MOSES dataset.

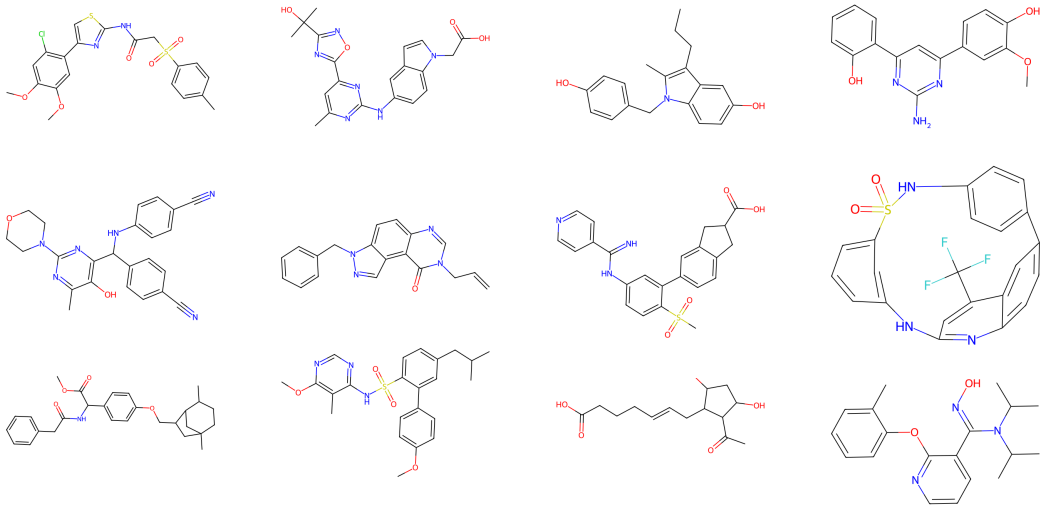

Figure 15: Non-curated samples generated by AUTOGRAPH (without pre-training) trained on the GuacaMol dataset.

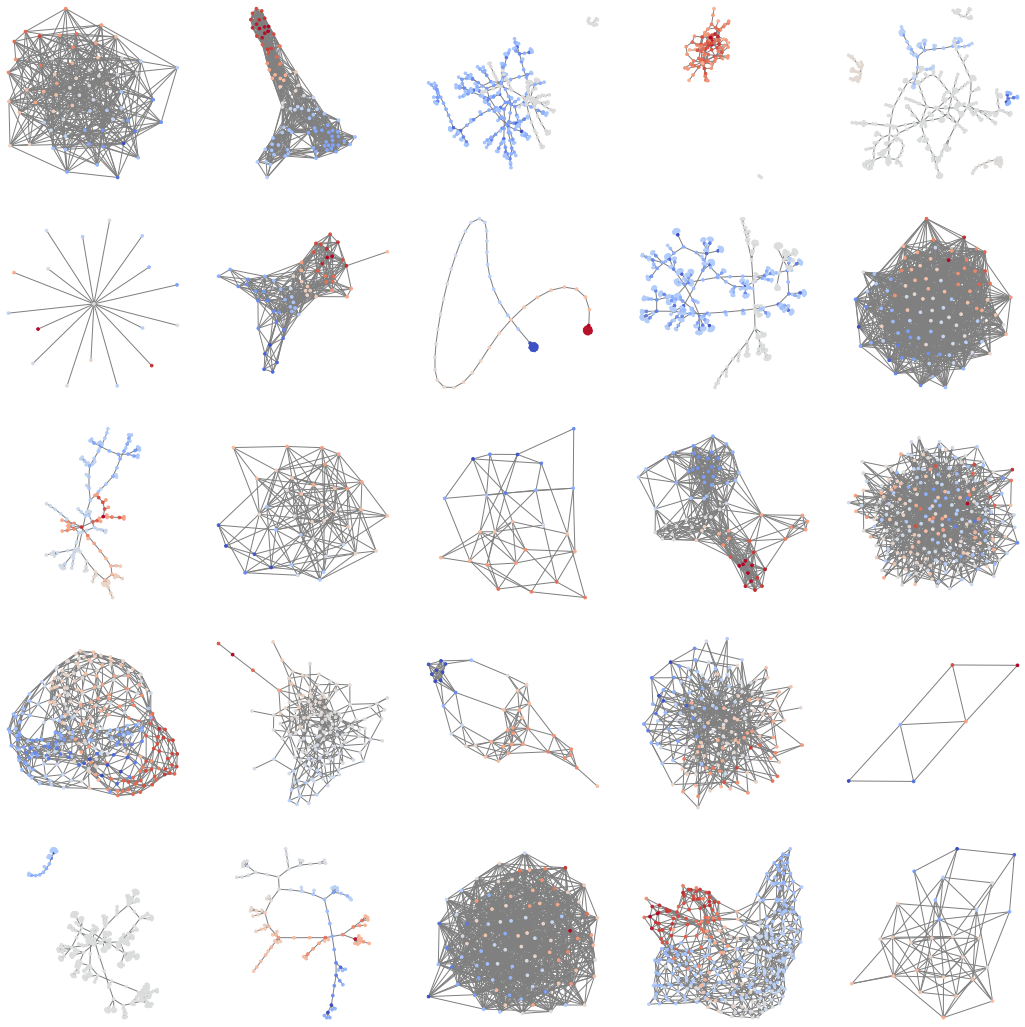

Figure 16: Non-curated samples generated by AUTOGRAPH trained on the NetworkX dataset.

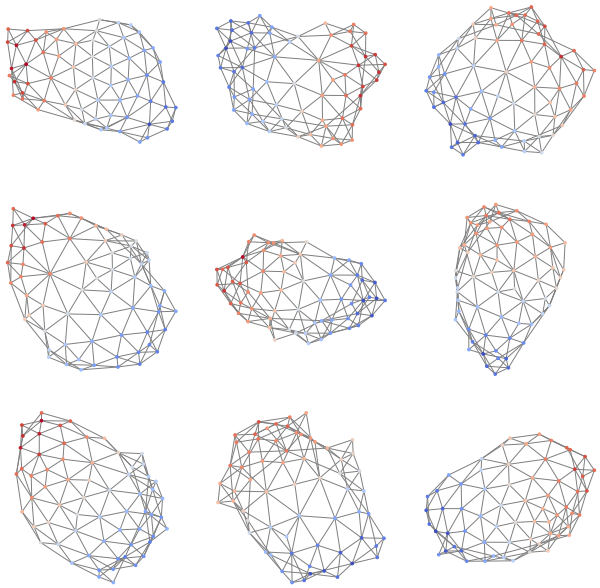

Figure 17: Non-curated samples generated by AUTOGRAPH (with pre-training on the NetworkX dataset) trained on the Planar dataset.

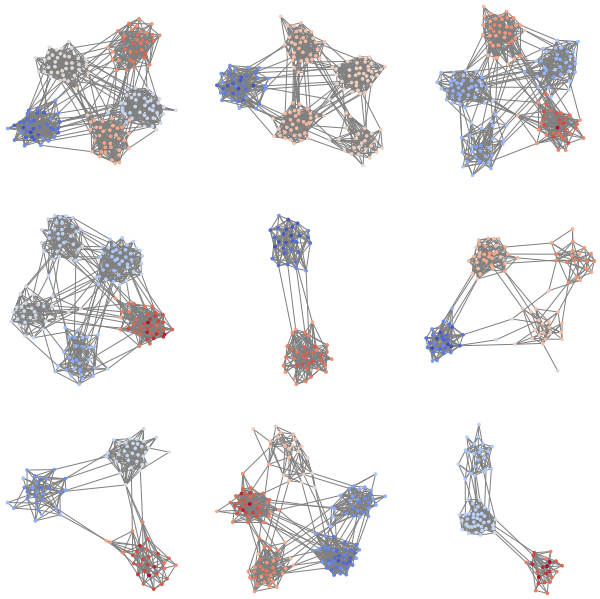

Figure 18: Non-curated samples generated by AUTOGRAPH (with pre-training on the NetworkX dataset) trained on the SBM dataset.

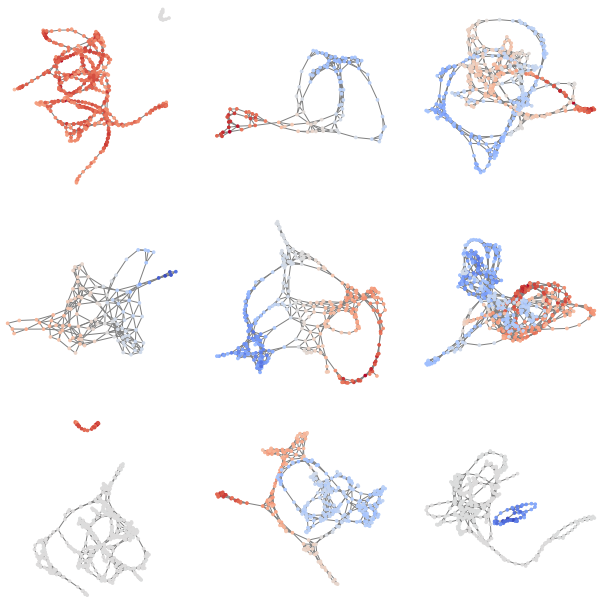

Figure 19: Non-curated samples generated by AUTOGRAPH (with pre-training on the NetworkX dataset) trained on the Proteins dataset.

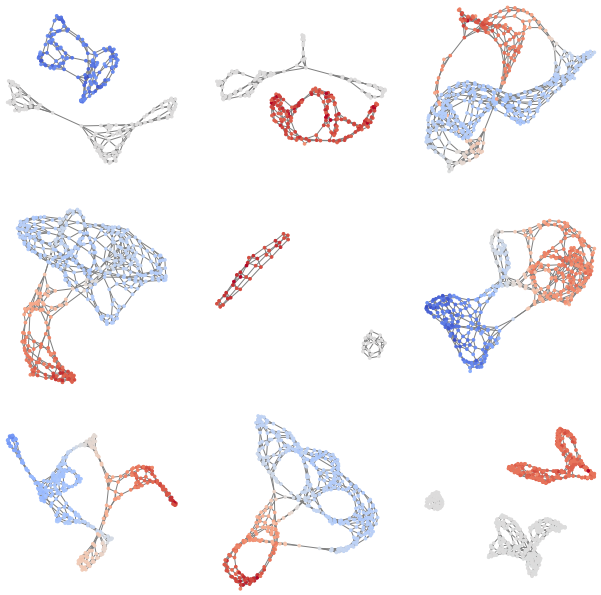

Figure 20: Non-curated samples generated by AUTOGRAPH (with pre-training on the NetworkX dataset) trained on the Point Clouds dataset.