FINE-GRAINED GRAPH GENERATION WITH LATENT MIXTURE SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Controlled graph generation requires producing graphs whose structures satisfy given topological attributes, instead of merely matching an unconditional graph distribution. Yet conditional latent-variable models can suffer from a mismatch between graph-conditioned latent representations used during training and attribute-conditioned representations used at inference. We introduce TOPOGEN, a conditional stochastic graph generator for fine-grained structural control. Its central component is a latent mixture scheduler that gradually transitions training from graph-conditioned training to attribute-only generation by interpolating latent distributions. This schedule reduces the mismatch between posterior-based reconstruction during training and prior-based conditional generation at inference time, enabling TOPOGEN to generate graphs from attributes alone. The approach allows improving *joint constraint satisfaction* without sacrificing diversity among valid outputs. Across five graph benchmark collections, TOPOGEN achieves the lowest average paired attribute error on all datasets and the best average structural fidelity across spectral and edit-distance metrics. Ablations show that latent scheduling and distribution alignment are important for controllable generation.

1 INTRODUCTION

Graph generation is a fundamental task in machine learning for modeling real-world networks such as molecular structures, social networks, and citation networks. Traditional models focus on producing graphs that match broad structural patterns, such as degree distributions (Erdös & Rényi, 1959; Barabási & Albert, 1999; You et al., 2018). In many applications, however, generation must be *controlled*: the use specifies desired graph-level properties, and the model must generate graphs that satisfy those properties (Zahirnia et al., 2024; Martinkus et al., 2022). This type of fine-grained control is crucial in drug discovery (generating new molecules that satisfy certain chemical properties) (Jin et al., 2018; 2020; Luo et al., 2021; Popova et al., 2019; Shi et al., 2020; Liu et al., 2021; Zang & Wang, 2020; De Cao & Kipf, 2018), synthetic material design (Wang et al., 2022; Sanchez-Lengeling & Aspuru-Guzik, 2018), social and information networks (Pitas, 2016; Zhou et al., 2020; Zeno et al., 2021), knowledge graphs (Melnyk et al., 2022; Zhou et al., 2023; Cao et al., 2023), and programming languages (generating program graphs from source codes) (Allamanis et al., 2018).

Existing controlled graph generation methods provide useful but often limited forms of control. EDGE Chen et al. (2023) is a discrete diffusion model that explicitly focuses on node degrees. DiGress Vignac et al. (2023) also builds on discrete diffusion techniques but incorporate properties such as planarity or acyclicity. Spectre Martinkus et al. (2022) is a generative adversarial network that uses spectral information to guide graph generation. GenStat Zahirnia et al. (2024) is a variational autoencoder that learns latent adjacency representations from attributes such as edge counts, triangle counts, and k-hop neighbor histograms. Other works condition on class labels (Yang et al., 2019; Ommi et al., 2022) or molecular properties (Liu et al., 2024; Mercatali et al., 2024). Despite this progress, fine-grained simultaneous control over multiple structural attributes is still a challenge, especially when the model must generate diverse graphs from attributes alone.

We propose TOPOGEN, a conditional stochastic graph generator. During training, TOPOGEN uses both a graph and its topological attributes for better latent space alignment and improved decoder tuning. At inference time, it discards the graph encoder and generates graphs using only the desired attributes. The main **technical contribution** is a latent mixture scheduler (MIXTURE-SCHEDULER)

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

that transitions decoder training from graph-conditioned posterior latents to attribute-conditioned prior latents by interpolating Gaussian distributions using shared noise. At the end of training, the decoder is trained on pure attribute-prior samples, matching the inference-time distribution and mitigatigating the mismatch between training-time reconstruction and attribute-only generation.

We evaluate TOPOGEN against existing models on several controlled graph generation benchmarks. Our key findings are as follows: (1) joint graph-attribute integration improves generation quality, aligning graphs with specific attribute constraints more effectively than prior models, (2) gradual transitioning from graph-posterior to attribute-prior information during training improves controlled graph generation, and (3) jointly conditioning on multiple fine-grained control attributes improves generation precision, suggesting that different structural constraints provide complementary guidance for graph generation.

2 CONTROLLED GRAPH GENERATION

Problem Definition Let $\mathbf{c} \in \mathbb{R}^m$ denote a vector of fine-grained topological attributes of a target graph G. Our goal is to generate a graph $\hat{G}$ whose structure satisfies the attributes $\mathbf{c}$.[1]

Solution Overview We formulate controlled graph generation as conditional stochastic generative modeling. As illustrated in Figure 1, during *training*, each example consists of a graph G= (V, E) and its corresponding attribute vector $\mathbf{c}$. A graph neural network encoder maps $G$to a diagonal Gaussian graph posterior distribution $q_\phi(\mathbf{Z_G} \mid G)$, while the attribute encoder maps $\mathbf{c}$ to an attribute-conditioned Gaussian prior $p_\theta(\mathbf{Z_c} \mid \mathbf{c})$. MIXTURE-SCHEDULER constructs a training latent $\mathbf{Z}$ by progressively interpolating the means and standard deviations of these two distributions using shared Gaussian noise, gradually exposing the graph decoder to attribute-prior samples during training. A graph decoder then learns the likelihood distribution $p_\psi(G \mid \mathbf{Z})$ from this latent representation. An explicit squared 2-Wasserstein alignment term encourages the graph-conditioned and attribute-conditioned distributions to occupy compatible regions of latent space. At *inference* time, the graph encoder is discarded and graphs are generated using only attribute conditioned prior $p_\theta(\mathbf{Z_c} \mid \mathbf{c})$, which is the same latent distribution used toward the end of training.

Topological Properties as Control Attributes We use graph-level structural attributes for explicit and precise control over the graph generation process. These include **number of nodes & edges**: define the scale of the target graph; **number of local bridges**: number of edges that are not part of a triangle in the graph, these "bridges" transfer information between different graph regions; **graph density**: fraction of possible edges in the graph; **edge connectivity**: minimum number of edges that must be removed to disconnect the graph; **node connectivity**: minimum number of nodes that must be removed to disconnect the graph; **number of maximal cliques**: number of maximal complete subgraphs in the graph; **graph diameter**: maximum shortest path distance between any pair of nodes in a graph; **treewidth min degree**: an estimate quantifying how far the graph is from a tree-like structure; **closeness centrality**: average distance of a node to all other nodes in its corresponding connected component, averaged across all nodes; **clustering coefficient**: fraction of triangles within a node's immediate neighbors, averaged across all nodes; and **transitivity**: fraction of all possible triangles present in a graph.

Importance These attributes allow users to specify desired graph structures directly, instead of relying on coarse labels or post-hoc filtering. They enable controlled synthetic benchmarks, targeted data augmentation, and stress-testing of graph models. For example, one can generate communication networks with specified robustness (node/edge connectivity, local bridges), contact networks with tunable clustering, or graph structures with prespecified connectivity and clique structure.

2.1 GRAPH ENCODING THROUGH MIXTURE SCHEDULING

Preprocessing Each training example consists of a graph $G = (V, E)$ and its corresponding control-attribute vector $\mathbf{c}$. Graph attributes are standardized using the training-set mean and standard deviation. For simplicity, we use $\mathbf{c}$ to denote the standardized attribute vector below.

[1] In this work we focus on undirected graphs to maintain generality. Extensions to node labels, edge labels, or typed graphs can be incorporated by adding categorical node and edge likelihood terms to the decoder.

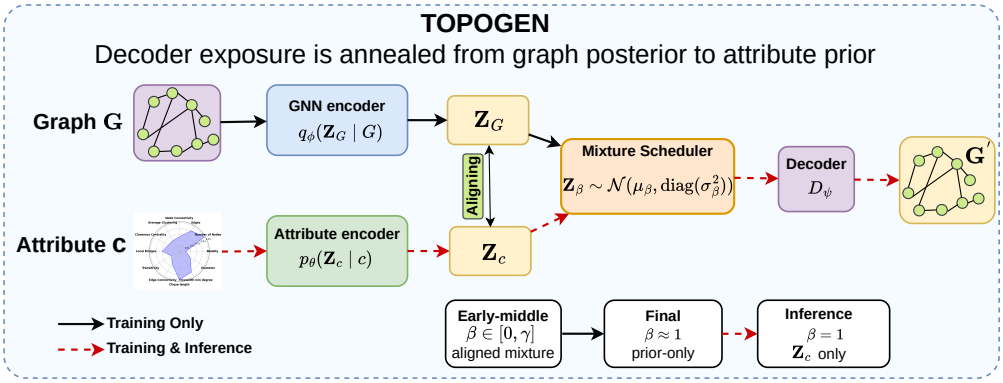


Figure 1: TOPOGEN uses a GNN encoder and an attribute encoder during training. The latent mixture scheduler interpolates diagonal Gaussian parameters according to equation 4, creating a bridge from graph-conditioned reconstruction to attribute-prior generation. During inference, TOPOGEN uses only the attribute encoder and graph decoder.

Graph Encoder TOPOGEN encodes the input graph G using a Graph Neural Network (GNN) encoder and parameterizes a graph-conditioned diagonal Gaussian distribution,

$$q_\phi(Z_G \mid G) = \mathcal{N}\big(\mu_G,\, \mathrm{diag}(\sigma_G^2)\big)\,, \qquad \mu_G = h_\phi^\mu(G), \quad \sigma_G = \exp\big(h_\phi^s(G)\big)\,, \tag{1}$$

where h_ϕ^μ and h_ϕ^s denote neural projection heads for the mean and log-standard-deviation respectively. The diagonal covariance provides a stochastic graph representation while allowing efficient reparameterized sampling. We refer to $q_\phi(Z_G \mid G)$ as the graph-conditioned latent distribution.

Attribute Encoder The attribute encoder in Figure 1 maps the normalized control attribute $\mathbf{c}$, to the mean of an attribute-conditioned Gaussian prior p_θ:

$$p_\theta(Z_c \mid \mathbf{c}) = \mathcal{N}(\mu_c, I)\,, \qquad \mu_c = f_\theta(\mathbf{c}), \tag{2}$$

where $f(\mathbf{c})$ is a non-linear transformation of the attribute vector from a feed forward neural network to capture interaction between the features of the attribute representation, and has mean μ_c and unit standard deviation. Thus, $\mathbf{Z}_G$ captures graph-specific structural information, while $\mathbf{Z}_c$ captures the latent representation induced by the requested control attributes.

We couple the two diagonal Gaussian distributions using a shared reparameterization noise $\epsilon \sim \mathcal{N}(0, I)$, and construct:

$$Z_G = \mu_G + \sigma_G \odot \epsilon, \qquad Z_c = \mu_c + \epsilon. \tag{3}$$

Mixture Scheduler At normalized training time $t = e/E$, where e is a training epoch and E is the total number of epochs, MIXTURE-SCHEDULER progressively transitions the latent distribution form the graph-conditioned posterior towards the attribute-conditioned prior. For a prior-inclusion coefficient $\beta \in [0, 1]$, the scheduled latent is:

$$Z_\beta = (1 - \beta(t))\mu_G + \beta(t)\mu_c + [(1 - \beta(t))\sigma_G + \beta(t)\mathbf{1}] \odot \epsilon. \tag{4}$$

Consequently,

$$Z_\beta \sim \mathcal{N}\Big((1 - \beta(t))\mu_G + \beta(t)\mu_c,\, \mathrm{diag}\Big([(1 - \beta(t))\sigma_G + \beta(t)\mathbf{1}]^2\Big)\Big)\,. \tag{5}$$

At $\beta = 0$, the decoder receives a sample from the graph-conditioned distribution. At $\beta = 1$, it receives a sample from the attribute-conditioned prior. For diagonal Gaussian distributions, this parameter interpolation is the 2-Wasserstein geodesic between $q_\phi(Z_G \mid G)$ and $p_\theta(Z_c \mid \mathbf{c})$. Therefore, the scheduler defines a valid latent distribution at every training step, and progressively interpolates both the mean and standard deviation of the graph posterior toward those of the attribute prior.

We define the prior inclusion factor:

$$\beta(t) = \min\Big(\gamma, (1 - (1 - \beta(0))(1 - t))^{\frac{1}{\alpha}}\Big)\,, \tag{6}$$

where $\gamma \in [0, 1]$ controls the maximum prior inclusion during the scheduled phase, $\beta(0)$ is the initial inclusion value, and $\alpha > 0$ controls the growth rate during the mixed phase; larger α delays prior inclusion, while smaller α introduces the prior earlier.

Training proceeds in two phases. During the scheduled phase, $\beta(t)$ increases according to (6) and is capped by γ, allowing the decoder to transition gradually from the graph-conditioned posterior toward the attribute-conditioned prior. In the final phase, we set $\beta(t) = 1$, so that

$$Z_\beta = \mu_c + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{7}$$

and the decoder is trained exclusively using samples from $p_\theta(Z_c \mid \mathbf{c})$, matching the inference setting.

2.2 Attribute-Guided Graph Generation

Graph Generation We model graph generation using a Bernoulli distribution (Murphy, 2012). Given the latent representation Z_β, the graph decoder maps Z_β to $N_{\max}$ latent node representations $\{\mathbf{h}_i\}_{i=1}^{N_{\max}}$, where $N_{\max}$ is the maximum number of node. For each node pair (i, j), the decoder computes a bilinear score and corresponding Bernoulli probability as:

$$p_{ij} = \sigma\left(\mathbf{h}_i^\top \mathbf{W} \mathbf{h}_j\right), \qquad E_{ij} \sim \text{Bernoulli}(p_{ij}), \tag{8}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\mathbf{h}_i$ and $\mathbf{h}_j$ are the latent representations of nodes i and j respectively, and $\mathbf{W}$ is a learned bilinear edge-scoring matrix. The graph decoder predicts edge probabilities over $N_{\max}$ node, providing a fixed generation space for graphs of varying effective size. Following edge sampling, isolated nodes are discarded by retaining the largest connected component of the generated graph. At inference time, generation relies exclusively on the attribute-conditioned prior $p_\theta(Z_c \mid \mathbf{c})$, consistent with the prior-only stage used during the final phase of training.

Attribute Decoder The attribute decoder reconstructs the control vector from $\mathbf{Z}_c$:

$$\widehat{\mathbf{c}} = D_{\text{att}}(\mathbf{Z}_c). \tag{9}$$

We implement D_{att} as a feedforward neural network and train it by minimizing the mean squared error (MSE) between the ground truth attribute ($\mathbf{c}$) and predicted attribute vectors $\widehat{\mathbf{c}}$. This auxiliary loss encourages $\mathbf{Z}_c$ to preserve the requested graph-attribute information, see the third term in (11).

Wasserstein Latent Alignment We align the graph and attribute latent distributions using the exact squared 2-Wasserstein distance between diagonal Gaussians. For a mini-batch of size B, let $\boldsymbol{\mu}_G, \boldsymbol{\mu}_c, \boldsymbol{\sigma}_G \in \mathbb{R}^{B \times d}$ denote the stacked graph means, attribute means, and graph standard deviations. Since the attribute prior has unit standard deviation, the batch alignment loss is:

$$\mathcal{L}_W = \frac{1}{B} \sum_{i=1}^{B} \left[\left\| \mu_G^{(i)} - \mu_c^{(i)} \right\|_2^2 + \left\| \sigma_G^{(i)} - \mathbf{1} \right\|_2^2 \right]. \tag{10}$$

This is the closed-form squared 2-Wasserstein distance for diagonal Gaussian distributions, averaged over the batch.

Training Objective For each training pair $(G, \mathbf{c})$, the graph encoder and attribute encoder parameterize the diagonal Gaussian distributions $q_\phi(Z_G \mid G)$ and $p_\theta(Z_c \mid \mathbf{c})$, respectively. Using a shared noise variable $\epsilon \sim \mathcal{N}(0, I)$, MIXTURE-SCHEDULER constructs the scheduled latent representation Z_β according to (4), and TOPOGEN minimizes:

$$\mathcal{J}(\phi, \theta, \psi) = -\frac{1}{B} \sum_{i=1}^{B} \log p_\psi \left(G^{(i)} \mid Z_\beta^{(i)} \right) + \lambda_d \mathcal{L}_W + \lambda_c \frac{1}{B} \sum_{i=1}^{B} \left\| \mathbf{c}^{(i)} - D_{\text{att}}\left(Z_c^{(i)} \right) \right\|_2^2, \tag{11}$$

where B is the minibatch size. The first term trains the decoder to generate graphs consistent with the training graph under the scheduled latent distribution. The second term explicitly aligns the graph-conditioned and attribute-conditioned Gaussian distributions. The third term encourages the attribute prior to retain information about the requested control attributes. The coefficients λ_d and λ_c balance these terms.

We note that the objective in (11) is a regularized stochastic autoencoding objective, not a standard variational evidence lower bound. The role of the Wasserstein term is explicit distribution alignment, and the role of the scheduler is controlled decoder exposure to a sequence of well-defined latent distributions ending at the inference prior.

Inference Process At inference time, only the desired attribute vector $\mathbf{c}$ is required. We compute:

$$\mu_c = f_\theta(\mathbf{c}), \qquad Z_c = \mu_c + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{12}$$

which is exactly the $\beta = 1$ latent distribution used in the final training phase. The graph decoder then samples edges from $p_\psi(G \mid Z_c$. Multiple draws of ϵ and decoder node noise yield diverse graphs for the same requested attributes.

3 EXPERIMENTS

We evaluate TOPOGEN on controlled graph generation across citation, lexical, molecular-topology, and synthetic graph benchmarks. Our experiments are designed to answer five questions: (i) do generated graphs satisfy the requested structural attributes? (ii) do generated graphs remain structurally close to the target graph beyond attribute-level agreement? (iii) does the model generate novel and diverse graphs rather than memorizing training examples? (iv) can the model generalize to control attributes that are out of the training attribute distribution? and (v) which components of TOPOGEN are responsible for controllability and training–inference calibration?

Datasets **Ogbn-arxiv** (Hu et al., 2020) is a large citation network of computer science papers from arXiv, with nodes as papers and edges represent citations among papers. Each node carries an embedding derived from its title and abstract. **Citeseer** (Kipf & Welling, 2017) is a citation network of scientific articles, where nodes are papers and edges indicate citations. **MUTAG** (Morris et al., 2020) is a molecular graph dataset where each graph represents a chemical compound labeled by mutagenic effect on gram negative bacteria. **MOLBACE** (Hu et al., 2020) is a molecular graph dataset where each graph represents a chemical compound. We use MUTAG and MOLBACE as *unlabeled* topology benchmarks; chemical validity and atom/bond labels are outside the scope of this work. **WordNet** (Miller, 1995) is a lexical database of English, where words are connected by semantic relationships. We construct four WordNet graphs using hypernyms, hyponyms, meronyms, and holonyms relations. We extract k-hop ego-subgraphs with $k \in \{2, 3\}$ around distinct source nodes. The train, validation and test splits are constructed after subgraph extraction, with each extracted subgraph assigned to exactly one split. Thus, no subgraph instance appears in more than one split. Dataset statistics and additional details are provided in Table 7, Appendix A.1.

Evaluation Metrics We evaluate controlled graph generation using several metrics focused on attribute controllability, structural fidelity, novelty, and diversity. **Mean Absolute Difference (MAD↓)** measures the average absolute error between the attributes of a generated graph and the requested target attributes, averaged over all attributes and test instances. To prevent attributes with large numerical ranges from dominating the aggregate score, attributes are normalized using training-set statistics before aggregation. MAD is our primary controllability metric. **Spectral Difference (SD↓)** (Jo et al., 2024) compares the sorted Laplacian spectra of a generated graph and a reference graph. It captures global structural properties such as connectivity, diffusion dynamics, and community structure, while being invariant to node ordering and small local perturbations. For graphs of different sizes, we zero-pad the shorter eigenvalue sequence and report average spectral difference, $SD = 1/n \times ||\lambda_{gold} - \lambda_{pred}||_2$, where n is the aligned spectrum length. **Graph Edit Distance (GED↓)** Sanfeliu & Fu (1983) measures the minimum number of node and edge edit operations required to transform one graph into another. GED captures fine-grained structural differences but is NP-hard to compute exactly Zeng et al. (2009). We therefore use it primarily for graphs where computation is tractable. As with SD, GED is a useful reference metric but can penalize valid generated graphs that satisfy the same attributes while being non-isomorphic to the held-out reference.[2] **Novelty (↑)** measures the fraction of generated graphs that are not isomorphic to any training graph. We use a fast candidate filter and exact isomorphism checks (Foggia, 2001) using NetworkX (Hagberg et al., 2008) for candidate matches. **Diversity (↑)** under fixed controls measures whether the model captures one-to-many nature of conditional graph generation (for many attribute vectors, multiple non-isomorphic graphs can satisfy the same structural constraints). For

[2]We note that Maximum Mean Discrepancy (**MMD↓**) (You et al., 2018; Vignac et al., 2023) is widely used to assess the quality of graph generative models. However, its diagnostic value for controlled generation has been questioned (O'Bray et al., 2021), as MMD measures distribution-level similarity rather than per-graph adherence to the specified target attribute vector. Therefore, we do not use MMD for evaluation.

a fixed attribute vector $\mathbf{c}$, we sample $K = 5$ graphs and compare pair-wise structural diversity: $\mathrm{Div}_d(\mathbf{c}) = \frac{2}{K(K-1)} \sum_{i<j} d(\widehat{G}_i, \widehat{G}_j)$, where d is SD or GED when tractable.

Baselines We compare TOPOGEN against graph generation methods. For controlled baselines, we incorporate the same control attributes into model training (details in Appendix A.4). Unconditional baselines are included to quantify the benefits of explicit attribute conditioning. **Erdős–Rényi** (Erdös & Rényi, 1959) generates a random graph with the same number of nodes as the target graph, connecting node pairs independently with probability p. This tests whether matching the number of nodes alone is sufficient for fine-grained control. **GraphRNN** (You et al., 2018) generates graph sequentially using node-level and edge-level recurrent networks over a breadth-first node ordering. **EDGE** (Chen et al., 2023) is a diffusion-based graph generative that iteratively corrupts and reconstructs graphs through iterative denoising. **GenStat** (Zahirnia et al., 2024) learns latent adjacency representations conditioned on graph level statistics, and decodes it to generate graphs satisfying target attribute. **DiGress** (Vignac et al., 2023) performs discrete denoising diffusion over categorical node and edge attributes, augmented with graph-theoretic features. **GruM** (Jo et al., 2024) generates graphs using mixture of endpoint-conditioned diffusion processes designed to capture graph topology.

3.1 MAIN RESULTS

Structural Attribute Satisfaction Table 1 reports performance across all datasets. TOPOGEN achieves the lowest average MAD among all methods, reducing the average attribute error from to 3.32. It obtains the best MAD on WordNet, Citeseer, Ogbn-Arxiv, and MUTAG, and remains competitive on MOLBACE. The Erdős-Rényi baseline performs poorly despite preserving the number of nodes, confirming that graph size alone is insufficient for satisfying higher-order attributes. GraphRNN improves over random generation but is substantially worse than TOPOGEN in average MAD, indicating the importance of explicit attribute conditioning. Among controlled baselines, GenStat and EDGE are competitive on selected datasets, especially MOLBACE, but TOPOGEN gives the best average controllability across domains. Attribute-wise errors are reported in Appendix A.6.

Structural Fidelity We evaluate whether generated graphs remain structurally close to the held-out graph associated with each control vector. Table 1 shows that no single method dominates every reference-similarity metric on every dataset. This is expected: the conditioning attributes are not unique identifiers of a graph, so a generated graph can satisfy the requested attributes while being different from the particular reference instance. **On SD**, TOPOGEN achieves the best score on Citeseer and MUTAG and remains competitive on the remaining datasets. GenStat and GraphRNN obtain strong SD on some datasets, indicating that they can preserve certain spectral properties, but they do so with substantially higher attribute error on average. Thus, TOPOGEN provides a favorable controllability–fidelity tradeoff: it gives the best attribute satisfaction while maintaining reasonable global structural similarity. **On GED**, TOPOGEN is competitive with sequential and diffusion baselines, with particularly strong performance on MOLBACE. GED can penalize valid alternative graphs that satisfy the same controls but differ from the source instance. We thus interpret GED as a reference similarity score instead of a complete measure of conditional generation quality.

Novelty of Generated Graphs A controlled generator should not satisfy attributes by memorizing training graphs. We therefore evaluate whether TOPOGEN generates graph structures that are distinct from those observed during training. For each held-out test instance, we use its attribute vector $\mathbf{c}$ as the control input and sample a graph from the attribute-conditioned prior. We then compute the percentage of generated graphs that are not isomorphic to any graph in the training. Graphs for which the isomorphism check exceeds 10 minutes are excluded from the analysis. As shown in Table 2, TOPOGEN generates highly novel graphs, with novelty ranging from 94.00% on WordNet to 100% on Citeseer, MUTAG, and MOLBACE. These results show that TOPOGEN learns a stochastic conditional graph distribution rather than retrieving training examples.

Diversity under Fixed Control Because many non-isomorphic graphs can share the same attribute vector, diversity under a fixed control vector is essential. To test whether TOPOGEN can capture this one-to-many mapping, we sample five graphs from the same control attribute vector at

Table 1: Controlled graph generation results. MAD (↓) measures paired attribute-control error and is our primary controllability metric. Spectral Difference (SD, ↓) and Graph Edit Distance (GED, ↓) measure reference-based structural similarity to the held-out source graph. Lower is better for all metrics. GED is reported only where tractable.

Metric	Model	WordNet	Citeseer	Ogbn-Arxiv	MUTAG	MOLBACE	Avg.
MAD (↓)	**Erdős-Rényi**	5.99	6.95	7.16	6.18	11.05	7.46
	GraphRNN	3.26	5.05	4.80	1.71	3.81	3.73
	GenStat	4.11	5.34	5.53	4.14	**3.05**	4.43
	EDGE	3.91	4.97	5.52	2.62	3.07	4.02
	DiGress	5.23	6.63	6.67	5.39	9.96	6.78
	GruM	3.75	5.37	5.42	3.80	10.8	5.83
	TopoGen	**2.54**	**4.38**	**4.65**	**1.60**	3.46	**3.32**
SD (↓)	**Erdős-Rényi**	0.98	0.89	1.01	0.66	0.49	0.80
	GraphRNN	**0.31**	0.42	**0.46**	0.21	0.13	0.31
	GenStat	0.32	0.40	0.47	0.21	**0.11**	**0.30**
	EDGE	0.32	0.44	0.47	0.27	0.14	0.33
	DiGress	0.37	0.85	0.73	0.88	0.91	0.75
	GruM	0.40	0.43	0.50	0.52	0.69	0.51
	TopoGen	0.60	**0.30**	0.62	**0.20**	0.17	0.37
GED (↓)	**Erdős-Rényi**	34.24	46.37	49.13	36.00	66.32	46.41
	GraphRNN	32.58	54.83	52.43	15.77	41.52	**39.43**
	GenStat	37.01	58.69	63.77	34.66	61.32	51.09
	EDGE	35.16	59.58	59.70	29.77	72.93	51.43
	DiGress	31.32	45.16	**47.24**	32.66	61.64	43.60
	GruM	29.68	55.06	51.87	27.00	65.27	45.78
	TopoGen	**28.72**	62.28	50.92	**17.00**	**40.62**	39.9

inference time and compare the resulting structures. Table 9, Appendix A.3 shows that TopoGen generates diverse graphs with low SD error. This demonstrates that the model does not collapse to a single canonical graph for each attribute vector, but can generates distinct outputs that are faithful to the same control specification.

OOD Control Attributes We evaluate whether TopoGen can respond to control vectors outside the training attribute distribution. We therefore construct an attribute stress test by sampling 25 Barabási–Albert graphs (Barabási & Albert, 1999), computing their structural attributes, and using these attributes as out-of-distribution control inputs for models trained on each dataset. Table 3 reports SD error under this OOD setting. Across training datasets, TopoGen maintains low SD error, with the average value of 0.48. These results suggest that TopoGen learns a meaningful mapping from structural attributes to graph topology. This experiment should be interpreted as an attribute-level stress test instead of a full domain-transfer benchmark, since the OOD attributes are generated from a synthetic graph family.

Synthetic SBM Evaluation Following prior work (Vignac et al., 2023; Jo et al., 2024), we also evaluate on stochastic block model (SBM) graphs (Holland et al., 1983). SBM graphs test whether the generator can preserve community structure. As shown in Table 4, TopoGen achieves the lowest SD error compared with strong graph generation baselines, further supporting its ability to generate graphs with realistic global organization under structural constraints.

Qualitative Analysis Table 5 shows representative generated graphs from TopoGen across all datasets. The generated graphs reasonably match the target structures (see SD values below each generated graph), especially in preserving global shape and local connectivity patterns. Additional visualizations generated by other baseline models and TopoGen are provided in Appendix A.5, Table 10.

Table 2: Novelty (%)

Dataset	Novelty (%)
Ogbn-Arxiv	98.6
Citeseer	100.0
MOLBACE	100.0
MUTAG	100.0
WordNet	94.0

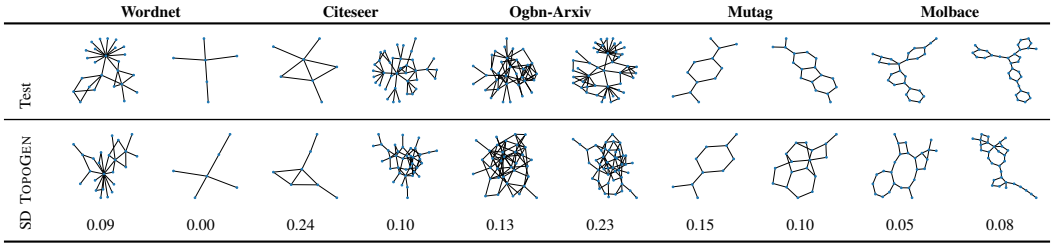
Table 3: OOD Error (SD ↓)

Model Trained on	SD
Citeseer	0.33
MOLBACE	0.53
MUTAG	0.83
Ogbn-Arxiv	0.53
WordNet	0.21

Table 4: SD (↓) on SBM graphs

Model	SD
GenStat	1.11
DiGress	1.47
GruM	1.23
TopoGen	**1.04**

Table 5: Graph visualization across datasets. Examples are taken from test splits of datasets.



3.2 Model Introspection

We conduct ablations to isolate the effect of TopoGen's design choices: the control attributes, the Wasserstein alignment loss, the attribute reconstruction objective, use of node/edge counts, and adaptive Mixture-Scheduler.

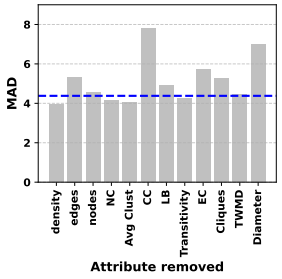


Figure 2: Removing one control attribute increases error; the blue line corresponds to using all attributes.

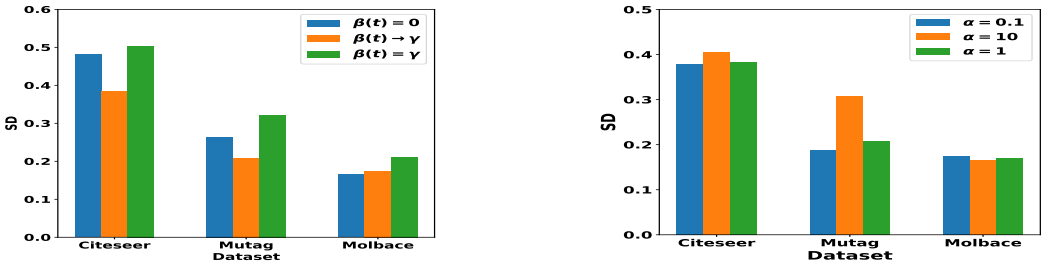
Contribution of Control Attributes Figure 2 shows the effect of removing one control attribute at a time. Removing closeness centrality, edge connectivity or diameter results in larger errors than removing simpler ones such as number of nodes or average clustering. This indicates that fine-grained structural attributes provide complementary constraints and improve controlled generation. Here, NC denotes node connectivity, EC edge connectivity, Avg Clust average clustering, LB local bridges, TWMD treewidth-min-degree, and Clique number of cliques.

Component Ablation Table 6 studies the contribution of the main components of TopoGen by removing each one in isolation. The full model achieves the lowest MAD on Citeseer and MUTAG and remains competitive on MOLBACE, indicating that the proposed components is beneficial for controllable generation. Removing the Wasserstein alignment term increases MAD on all datasets, showing that explicit alignment between the two latent distributions improves controllability. Removing the attribute reconstruction objective substantially degrades performance on Citeseer and MUTAG, indicating that explicitly preserving control information in the attribute space is important for accurate conditioning. Removing Mixture-Scheduler also increases error, particularly on Citeseer and MOLBACE, confirming that gradual interpolation between the graph-conditioned posterior and the attribute-conditioned prior is beneficial. The prior-only and posterior-only variants consistently underperform the full model on most datasets, showing that neither source alone is sufficient: graph-conditioned latents are useful for learning structure early in training, while attribute-prior latents are necessary for inference-time controllability.

Table 6: Component ablation of TopoGen. Bold marks the largest MAD degradation.

	Citeseer	MUTAG	MOLBACE
TopoGen	4.38	1.60	3.46
w/o Wasserstein alignment	4.87	1.72	3.78
w/o Attribute Reconstruction	**7.86**	2.52	2.92
w/o Mixture-Scheduler	7.54	1.76	**4.06**
w/o G (prior only)	6.75	2.45	2.94
w/o $\mathbf{c}$ (posterior only)	7.02	**2.93**	3.40

Does gradual inclusion of the prior distribution p_θ help? We compare three settings: posterior-only decoding with $\beta(t) = 0$, gradual inclusion of the attribute prior with $\beta(t) \to \gamma$, and constant inclusion with $\beta(t) = \gamma$. Figure 3(a) shows that gradual inclusion substantially reduces SD on



(a) Inclusion factors effect on generation error.

(b) Effect of α on generation error.

Figure 3: Ablation Analysis

Citeseer and MUTAG compared with posterior-only, while MOLBACE shows comparable performance. In contrast, constant inclusion consistently yields higher error. These results suggest that progressively introducing p_θ is generally more effective than exposing the decoder to a fixed prior contribution throughout the scheduled phase.

How does the rate of inclusion affect model's performance? Figure 3(b) shows the rate of prior inclusion (α) matters. Slower inclusion ($\alpha = 0.1$) achieves the lowest SD on Citeseer and MUTAG, whereas MOLBACE shows only small differences across settings and slightly favors faster inclusion. In fact, overly rapid prior exposure can destabilize reconstruction before the two latent distributions are aligned, whereas gradual inclusion yields better structural fidelity.

How much of the prior should be included? Figure 4 varies the maximum prior-inclusion rate γ from 0 to 1; we note that γ controls only the scheduled phase—the final training phase still uses $\beta = 1$ to match the attribute-only inference setting. The results show that the optimal amount of prior information is dataset dependent, and that full prior inclusion during the scheduled phase is generally not necessary. Moderate values of γ often achieve lower SD, suggesting that retaining graph-posterior information while gradually incorporating the attribute prior provides a better training signal than relying predominantly on either source. We therefore treat γ as a dataset-specific hyperparameter.

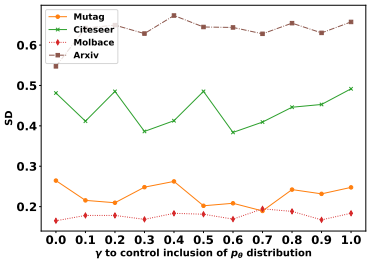


Figure 4: Maximum inclusion rate γ vs. SD error.

Limitations TopoGen currently focuses on undirected and unlabeled graph topology. Molecular datasets are used only as topology benchmarks, so atom types, bond types, valence constraints, and chemical validity are not modeled. Although the GNN encoder allows scalability, exact edge decoding can still be quadratic in the number of generated nodes; our experiments therefore generate extracted local subgraphs rather than entire networks such as Ogbn-Arxiv in one pass. Finally, arbitrary user-specified attributes may be mutually infeasible or may correspond to many non-isomorphic valid graphs, so TopoGen should be understood as learning a conditional distribution near the feasible attribute manifold observed during training but not as providing hard guarantees for every possible control vector.

4 Conclusion and Future Work

We introduced TopoGen, a conditional stochastic graph generator for fine-grained topological control. It aligns graph-conditioned and attribute-conditioned latent distributions and uses a mixture scheduler to gradually transition training from posterior-rich reconstruction to prior-only conditional generation. It enables precise control, even without explicit node and edge counts supervision. Across multiple graph benchmarks, TopoGen improves attribute satisfaction while maintaining competitive structural fidelity, novelty, and diversity. Future work can extend TopoGen to labeled and temporal graphs, chemically valid molecular generation, knowledge graph generation, as well as settings with noisy, incomplete, and potentially infeasible user-provided control settings.

AI use statement We used LLMs to assist with drafting and editing text; we retain responsibility for the originality, accuracy, and integrity of the final manuscript, including its technical claims, citations, experimental results, and any AI-assisted material.

Ethics statement This work studies controlled graph generation using citation, lexical, molecular-topology, and synthetic benchmarks that are publicly available. The molecular experiments concern unlabeled topology and do not establish chemical validity. Potential downstream risks include reproducing structural biases. We make no claims of fairness or suitability for safety-critical applications. Downstream use should include validation of requested constraints and application-specific requirements.

Reproducibility statement We will release the code and data to support reproduction and further investigation of the proposed method.

REFERENCES

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/pdf?id=BJOFETxR-.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999. URL https://www.science.org/doi/full/10.1126/science.286.5439.509.

Pengfei Cao, Yupu Hao, Yubo Chen, Kang Liu, Jiexin Xu, Huaijun Li, Xiaojian Jiang, and Jun Zhao. Event ontology completion with hierarchical structure evolution networks. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 306–320. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.21.

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*, pp. 4585–4610. PMLR, 2023. URL https://proceedings.mlr.press/v202/chen23k/chen23k.pdf.

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018. URL https://arxiv.org/pdf/1805.11973.

P Erdös and A Rényi. 2017-10-20t13:47:06.000+0200. *Publicationes Mathematicae Debrecen*, 6: 290–297, 1959. URL https://publi.math.unideb.hu/load_doi.php?pdoi=10_5486_PMD_1959_6_3_4_12.

Pasquale Foggia. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop...*, 2001.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. URL https://www.researchgate.net/publication/236407765_Exploring_Network_Structure_Dynamics_and_Function_Using_NetworkX.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018. URL http://proceedings.mlr.press/v80/jin18a/jin18a.pdf.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020. URL https://proceedings.mlr.press/v119/jin20a/jin20a.pdf.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/pdf?id=cZTFxktg23s.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph diffusion transformers for multi-conditional molecular generation. *Advances in Neural Information Processing Systems*, 37:8065–8092, 2024.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. GraphEBM: Molecular graph generation with energy-based models. In *Energy Based Models Workshop - ICLR 2021*, 2021. URL https://openreview.net/forum?id=Gc51PtL_zYw.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pp. 7192–7203. PMLR, 2021. URL https://proceedings.mlr.press/v139/luo21a.html.

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*, pp. 15159–15179. PMLR, 2022. URL https://proceedings.mlr.press/v162/martinkus22a/martinkus22a.pdf.

Igor Melnyk, Pierre Dognin, and Payel Das. Knowledge graph generation from text. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1610–1622, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.116. URL https://aclanthology.org/2022.findings-emnlp.116.

Giangiacomo Mercatali, Yogesh Verma, Andre Freitas, and Vikas Garg. Diffusion twigs with loop guidance for conditional graph generation. *Advances in Neural Information Processing Systems*, 37:137741–137767, 2024.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995. URL https://dl.acm.org/doi/pdf/10.1145/219717.219748.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020. URL https://arxiv.org/abs/2007.08663.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. *International Conference on Learning Representations, ICLR*, 2021.

Yassaman Ommi, Matin Yousefabadi, Faezeh Faez, Amirmojtaba Sabour, Mahdieh Soleymani Baghshah, and Hamid R Rabiee. Ccgg: A deep autoregressive model for class-conditional graph generation. In *Companion Proceedings of the Web Conference 2022*, pp. 1092–1098, 2022.

Ioannis Pitas. *Graph-based social media analysis*. CRC Press, 2016.

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

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018. URL https://www.science.org/doi/10.1126/science.aat2663.

Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, 1983. doi: 10.1109/TSMC.1983.6313167.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/pdf?id=S1esMkHYPr.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/pdf?id=UaAD-Nu86WX.

Shiyu Wang, Xiaojie Guo, and Liang Zhao. Deep generative model for periodic graphs. *Advances in Neural Information Processing Systems*, 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e89e8f84626197942b36a82e524c2529-Paper-Conference.pdf.

Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. *Advances in neural information processing systems*, 32, 2019.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/you18a.html.

Kiarash Zahirnia, Yaochen Hu, Mark Coates, and Oliver Schulte. Neural graph generation from graph statistics. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/72153267883fbcafdb6e4662382696c5-Paper-Conference.pdf.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 617–626, 2020. URL https://dl.acm.org/doi/pdf/10.1145/3394486.3403104.

Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

Giselle Zeno, Timothy La Fond, and Jennifer Neville. Dymond: Dynamic motif-nodes network generative model. In *Proceedings of the Web Conference 2021*, pp. 718–729, 2021. URL https://arxiv.org/pdf/2308.00770.

Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 401–411, 2020. URL https://dl.acm.org/doi/pdf/10.1145/3394486.3403082.

Wentao Zhou, Jun Zhao, Tao Gui, Qi Zhang, and Xuanjing Huang. Inductive relation inference of knowledge graph enhanced by ontology information. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pp. 6491–6502. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.findings-emnlp.431.

A APPENDIX

A.1 DATASET STATISTICS

The train/validation/test splits are constructed after subgraph extraction, with each extracted subgraph assigned to only one split. Since each subgraph is centered on a distinct source node, there is no overlap in source nodes between the data splits. We note that although neighboring nodes may appear in multiple node-centered subgraphs, we verified that no identical extracted subgraphs appear across splits. We use the extracted subgraphs to divide the train/val/test splits.

Table 7: Dataset statistics in terms of number of graphs.

Dataset	Train	Val	Test
WordNet	52,675	2,926	2,927
Citeseer	1,406	78	79
Arxiv	47,538	2,641	2,641
MUTAG	169	10	9
MOLBACE	1,323	74	74

A.2 EFFECT OF GRAPH SIZE

The GNN encoder is not tied to a fixed node ordering and can process sparse graphs of varying size. The main computational bottleneck is dense edge decoding, which scales quadratically with the number of generated nodes. To study this effect, we vary $N_{\max}$ from 50 to 200 and report spectral difference in Table 8. As expected, generation becomes more difficult as graph size increases because the number of possible edges and valid structural configurations grows rapidly. Nevertheless, the model remains trainable up to 200 nodes in this stress test.

Table 8: Generation performance degrades as the target number of nodes increases.

#Nodes	SD(↓)
50	0.40
60	0.60
80	0.74
100	0.82
200	0.83

A.3 GRAPH DIVERSITY

For many attribute vectors, multiple non-isomorphic graphs can satisfy the same structural constraints. To test whether TOPOGEN can capture this one-to-many mapping, we sample five graphs from the same control attribute vector at inference time and compare the resulting structures. Table 9 shows five sample graphs generated during inference compared with the ground truth graph. Furthermore, we sample K=10 graphs for each test condition on Citeseer. Across the full test set, the generated graphs achieve a mean pairwise spectral distance of 0.39, while maintaining a MAD of 4.29 . The SD and MAD are close to the value reported in Table 1, indicating that TOPOGEN generates structurally diverse samples without violating the required controls.

A.4 SETTINGS

Following previous works (De Cao & Kipf, 2018; Zahirnia et al., 2024), we set the maximum number of nodes to $V = 50$ in experiments. This threshold is used for fair comparison with baselines and is appropriate, given the common practice of sampling 1-2 hop subgraphs for nodes. We set the number of hops to $k = 2$ for all datasets except for Citeseer, for which we use $k = 3$ due to its smaller size. In addition, we extract graph attributes using Networkx (Hagberg et al., 2008). We consider a maximum number of 200 training iterations for Citeseer and 100 iterations for other

Table 9: Each row shows the five generated graphs given the same attribute with the ground truth graph and SD error variance measuring how close the generated graphs with ground truth.

Generated Graphs						SD Variance
Ground Truth	Sample 0 SD: 0.2739	Sample 1 SD: 0.3047	Sample 2 SD: 0.2324	Sample 3 SD: 0.2584	Sample 4 SD: 0.3087	0.0008
Ground Truth	Sample 0 SD: 0.2564	Sample 1 SD: 0.3560	Sample 2 SD: 0.3482	Sample 3 SD: 0.2122	Sample 4 SD: 0.2972	0.0030
Ground Truth	Sample 0 SD: 0.2564	Sample 1 SD: 0.1714	Sample 2 SD: 0.3106	Sample 3 SD: 0.2630	Sample 4 SD: 0.3340	0.0031
Ground Truth	Sample 0 SD: 0.2700	Sample 1 SD: 0.1720	Sample 2 SD: 0.3410	Sample 3 SD: 0.2268	Sample 4 SD: 0.2256	0.0032
Ground Truth	Sample 0 SD: 0.4792	Sample 1 SD: 0.5736	Sample 2 SD: 0.4538	Sample 3 SD: 0.2697	Sample 4 SD: 0.5693	0.0122

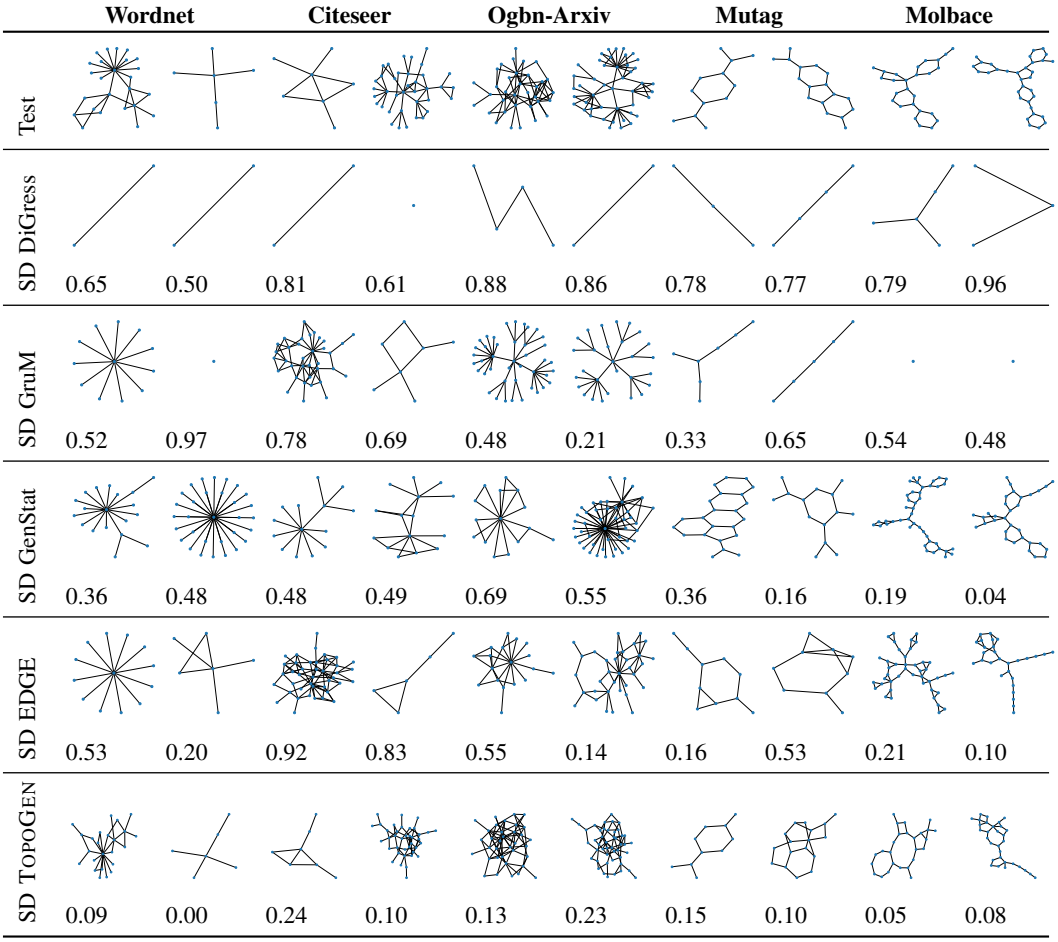
datasets, which is sufficiently large for convergence. All graphs are considered undirected graphs. For the graph encoder, we use a two-layer GCN with a hidden dimension of 128. In the absence of node attributes, all nodes are initialized with a learnable 25-dimensional embedding, and the resulting node representations are aggregated using mean pooling to obtain a graph-level representation; no node-index embeddings or order-dependent positional embeddings are used. A linear projection then parameterizes the mean and log-standard deviation of the graph-conditioned latent distribution. The graph decoder maps the latent representation to $N_{\max}$ node representations and predicts pairwise edge logits using a learned bilinear scoring function. We consider a batch-size of 1,028 and run all our experiments on a single A100 40GB GPU. During latent interpolation, the graph posterior and attribute prior are sampled using a shared reparameterization noise $\epsilon \sim \mathcal{N}(0, I)$, consistent with the distribution-level interpolation used by MIXTURE-SCHEDULER.

To incorporate our control attributes into the baseline models, we replace their default conditioning variables, such as eigenvalues or class labels, with the same graph-level control attributes used by TOPOGEN.

A.5 GRAPH VISUALIZATION

Table 10 shows examples of different graphs generated by baselines across datasets.

Table 10: Graph visualization across datasets. Examples are taken from test splits of datasets.



A.6 Attribute-Wise Error Analysis

Table 11 shows the absolute mean error between the ground truth value and predicted value for each attribute. We use mean absolute difference (MAD↓) metric for evaluation. MAD computes the absolute difference between the attributes of predicted graphs and their corresponding target graphs. We average these differences for each dataset. Abbreviations NC (node connectivity), EC (edge connectivity), TWMD (tree width min degree), Avg Clust (average clustering), LB (number of local bridge), Clique (number of cliques).

Table 11: Performance of TopoGen for each attribute across all datasets. We report the mean absolute Difference from target attributes; lower is better (MAD↓ is the average over attributes). Here, NC = Node Connectivity, CC = Closeness Centrality, LB = Number of local bridge, EC = edge connectivity, and TWMD = treewidth min degree

Attributes	Citeseer	WordNet	Mutag	Molbace	Ogbn-Arxiv
Density	0.05	0.10	0.04	0.02	0.11
Edges	21.96	11.80	6.11	16.58	25.05
Nodes	4.68	1.27	2.00	2.68	1.27
NC	0.04	0.04	0.00	0.01	0.37
Avg Clust	0.15	0.38	0.10	0.16	0.19
CC	0.06	0.05	0.10	0.11	0.08
LB	5.46	10.55	3.56	9.20	7.65
Transitivity	0.14	0.25	0.12	0.16	0.17
EC	0.04	0.05	0.00	0.03	0.39
Cliques	15.81	3.23	2.67	5.42	15.25
TWMD	3.37	1.83	1.22	2.22	4.16
Diameter	0.90	0.95	3.33	6.31	0.92
MAD	4.39	2.54	1.60	3.57	4.63