SPARSE GNNS MAKE STRONG BACKBONES FOR DISCRETE GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Leading discrete graph generators use Graph Transformers or dense message-passing networks as their denoising backbone, processing all N^2 node pairs at every layer despite typically sparse output graphs. We present SGen (**S**parse-backbone Graph **Gen**eration), which restricts intermediate message passing to the current non-null edges and predicts the full adjacency matrix only at the output through a bilinear head. We interpret SGen through conditional link prediction: a Bayes analysis decomposes edge denoising into structural prediction and current-edge information, and further bounds the error when SGen approximates this target. Integrated into DiGress and SimGFM, SGen achieves competitive generation quality on molecular and structural benchmarks, with metric-dependent trade-offs. On Tree and Planar, it delivers approximately 2.8–$3.9\times$ training speedups and 86%–92% lower peak training memory than the corresponding Graph Transformer backbones. Architecture-matched ablations show that sparse communication itself reduces runtime and memory, with the operator and model size held fixed. Reusing shared representations across terminal output blocks further enables SGen to extend an existing graph generation method to Proteins and Point Clouds with up to 5,037 nodes, without any algorithm specifically designed for large graphs.

1 INTRODUCTION

Dense backbones dominate graph generation. Iterative-refinement models have become the dominant paradigm for discrete graph generation. Since DiGress (Vignac et al., 2022) introduced a Graph Transformer (GT) backbone for discrete denoising diffusion, a series of methods—including, CatFlow (Eijkelboom et al., 2024), COMETH (Siraudin et al., 2024), DeFoG (Qin et al., 2024) and SimGFM (Luo et al., 2026)—have pushed generation quality forward through better noise schedules, flow-matching objectives, and sampling strategies. These advances focus almost exclusively on the *generative process*; the neural backbone that denoises at each step has received comparatively little attention and remains dense in every case. Both GT-based and GNN-based backbones—from early score-based models like EDP-GNN (Niu et al., 2020) and GDSS (Jo et al., 2022) to the recent GenGNN (Revolinsky et al., 2026), which replaces attention with message passing but still operates on the complete graph—maintain $O(N^2)$ per-layer cost.

Dense backbones bottleneck graph generation. This design is often treated as a necessary consequence of the task: because the model must ultimately predict edge types for all N^2 node pairs, it appears that intermediate layers should also exchange information across all pairs. In practice, the cost is substantial. On 64-node structural benchmarks, GT-based backbones consume up to 18 GB of peak GPU memory, limiting batch sizes and making training slow.

Can sparse backbones support graph generation? On the discriminative side, the GNN$^+$ line of work (Luo et al., 2024; 2025) has shown that sparse GNNs augmented with modern training recipes—residual connections, normalization, feed-forward blocks, and positional encodings—can match Graph Transformers on node and graph classification while communicating only along existing edges. Whether this finding transfers to the generative setting, where the model must reason about *absent* edges, has remained an open question. In this work, we explore whether sparse message passing can replace dense backbones for discrete graph generation. We present SGen (**S**parse-backbone Graph **Gen**eration based on GNN$^+$), an architecture that performs message passing only on a sparse communication graph derived from the noisy edge state, and recovers the full $N \times N$ edge prediction

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.

at a single output layer through a bilinear head. A pooling-based global state, updated at each layer and broadcast via learned affine modulation, provides graph-wide communication without cross-attention. Even at the highest noise level, the communication graph covers only a small fraction of all node pairs—an order of magnitude sparser than the complete graph—while the output head still predicts every node pair.

A Bayes view explains the sparse-to-dense design. We interpret SGen through conditional link prediction. Under independent categorical corruption, Bayes-optimal edge denoising combines structural prediction from the rest of the noisy graph with information from the current edge state. Our analysis shows that the exact current-state correction does not amplify average conditional KL error, and bounds denoising error when a neural network approximates the resulting logits. Local link-prediction theory and latent-role models further connect this view to sparse encoding and bilinear decoding. These results characterize SGen as a learnable approximation to the Bayes-optimal edge denoiser (Section 3.5 and Appendix E).

Sparse backbones achieve competitive quality at a fraction of the cost. We integrate SGen as a drop-in backbone replacement into both discrete diffusion and discrete flow matching methods and evaluate on six benchmarks spanning molecular and structural graph generation. Across these settings, the sparse backbone achieves competitive generation quality with metric-dependent trade-offs while being substantially cheaper: on QM9, SGen improves Validity and Uniqueness over the GT baseline with $4\times$ fewer parameters; on QM9H, Validity improves from 98.4% to 99.6%; on Tree, DiGress + GNN$^+$ improves V.U.N. from 90.0% to 97.0%, while SimGFM + GNN$^+$ matches its GT counterpart at 99.5%. On Tree and Planar, training is approximately 2.8–$3.9\times$ faster with 86%–92% lower peak training memory, and sampling is 3.3–$5.6\times$ faster. Matched complete/sparse GNN$^+$ ablations isolate the time and memory savings from sparse communication under the same operator, hidden widths, conditioning, and output head (Table 16).

SGen scales to larger graphs without a dedicated algorithm. By reusing shared node representations across terminal output blocks, SGen can extend an existing graph generation method to larger graphs without an algorithm specifically designed for large graphs. Experiments with the DiGress integration on Proteins and Point Clouds reach up to 500 and 5,037 nodes, respectively (Section 4.4). Each output block reuses the same sparse encoding instead of adding candidate edges to the backbone or rerunning it, reducing simultaneous output activations while preserving the original generation process (Section 3.6).

Contributions. We summarize our main contributions as follows:

- We introduce SGen, a backbone that decouples sparse communication from full node-pair prediction for discrete graph generation.
- We derive a Bayes decomposition of edge denoising through conditional link prediction and bound the transfer of structural-prediction and logit-approximation errors.
- We reuse sparse node representations across terminal output blocks, reducing simultaneous output activations and extending DiGress to larger graphs while preserving its generative process.
- Across discrete diffusion and discrete flow matching, we demonstrate competitive generation quality with metric-dependent trade-offs and use architecture-matched controls to isolate the efficiency gains from sparse communication.

2 Related Work

Backbones for iterative generation. DiGress (Vignac et al., 2022) established an edge-aware Graph Transformer (GT) for discrete graph diffusion. CatFlow, COMETH, DeFoG, and SimGFM developed different process and sampling designs (Eijkelboom et al., 2024; Siraudin et al., 2024; Qin et al., 2024; Luo et al., 2026). GenGNN (Revolinsky et al., 2026) draws on the GNN$^+$ design philosophy (Luo et al., 2024; 2025), but computes gates over all valid pairs and retains dense edge updates. Replacing attention therefore leaves quadratic per-layer pair computation. SGen instead restricts intermediate propagation and edge states to current non-null edges, deferring absent-pair prediction to the terminal head. Our complete/sparse GNN$^+$ controls match layers, widths, conditioning, and output head to isolate this support-level distinction.

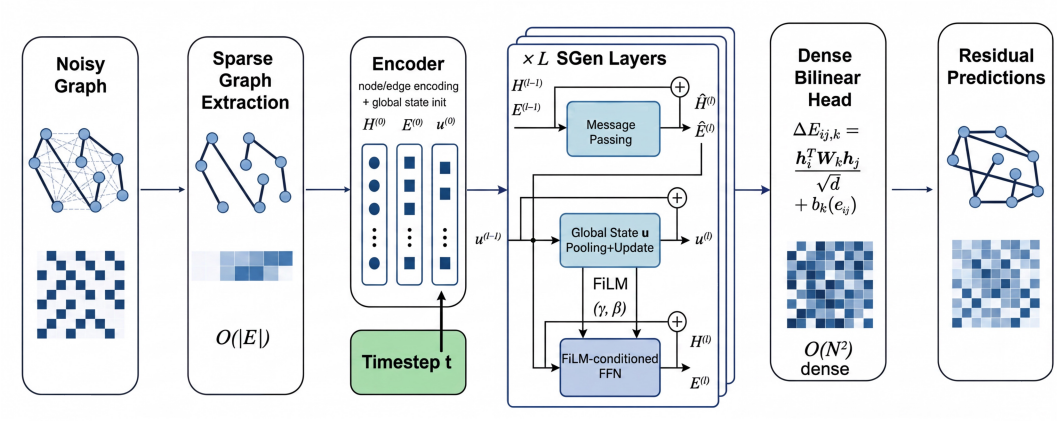

Figure 1: Architecture of SGen. The noisy graph G_t is converted to a sparse communication graph. The backbone processes this sparse representation through L GNN$^+$ layers (SGen layers), each consisting of an edge-aware sparse message-passing operator, a global state update via pooling, and a conditioned FFN. A single dense bilinear head at the output reconstructs predictions for all N^2 node pairs. The $O(|\mathcal{E}_t|)$ label describes sparse edge storage and propagation; dense-state extraction may require an $O(N^2)$ scan.

Sparse and scalable generation. SparseDiff (Qin et al., 2023) uses marginal categorical transitions, an efficient sparse noising implementation, query-edge training, and chunked inference. Query edges participate in its message passing, and each inference chunk invokes the denoiser. SGen instead encodes the current non-null support once per state and shares those representations across terminal blocks. Thus the distinction is not the marginal transition family alone. Degree-guided generation (Chen et al., 2023), local expansion (Bergmeister et al., 2023), and autoregressive graph representations (Liao et al., 2019; Dai et al., 2020; Chen et al., 2025) provide other routes to scale. Their whole-system results complement our controlled backbone comparisons.

Link prediction and shared representations. Enclosing-subgraph methods characterize local link-prediction structure (Zhang & Chen, 2018), while graph auto-encoders combine shared node representations with pair decoders (Kipf & Welling, 2016). We connect these ideas to generation by decomposing Bayes-optimal edge denoising into structural prediction and current-state evidence, then transferring approximation bounds to conditional denoising error. Since shared embeddings can lose joint pair information (Zhang et al., 2021), this compression remains part of the neural approximation error.

3 Method

We present SGen (Figure 1), a sparse GNN$^+$ backbone for discrete graph generation. Given a noisy graph $G_t = (X_t, E_t)$ and timestep t, the model: (1) extracts a sparse communication graph from E_t, (2) processes it through L GNN$^+$ layers with global conditioning, and (3) reconstructs dense edge predictions via a bilinear head.

3.1 Sparse Communication Graph Construction

For a sampled categorical state $G_t = (X_t, E_t)$, $E_t \in \mathbb{R}^{N\times N\times K_E}$ is one-hot. SGen retains the current non-null edges as its communication support:

$$\mathcal{E}_t = \{(i,j) : i \neq j,\ \arg\max_k E_{t,ij,k} \neq k_{\text{null}}\}. \tag{1}$$

The support is rebuilt after every framework update. It is derived from the sampled noisy graph, rather than fixed to clean edges or obtained by discretizing arbitrary soft probabilities.

3.2 GLOBAL-CONDITIONED GNN$^+$ BACKBONE

The encoder embeds node categories, active-edge categories, and auxiliary features including RRWP (Ma et al., 2023). A global state $u^{(0)} = \phi_u([t, y_t, y_{\text{extra}}])$ encodes the timestep and graph conditions. Each of the L layers applies three stages:

Sparse message passing. An edge-aware GatedGCN$^+$ operator (Bresson & Laurent, 2017; Luo et al., 2024; 2025) updates active-edge features and aggregates their messages into node features. No hidden edge state is propagated for absent pairs.

Global update. Mean pooling of the updated node and active-edge features feeds an MLP that updates u. This gives every node access to graph-wide information without all-pair intermediate communication.

Conditioned FFNs. Node and active-edge feed-forward outputs are modulated by the updated u through FiLM, $\text{Mod}(h, u) = \gamma(u) \odot h + \beta(u)$ (Perez et al., 2018). The shared conditioning leaves local inputs node- or edge-specific. Global conditioning enters here, rather than at the message-passing inputs. All five residual branches use learnable scalar ReZero gates initialized to zero (Bachlechner et al., 2021).

The full encoder and layer equations are in Appendix H; the GatedGCN$^+$ operator is specified in Appendix B.

3.3 DENSE PAIRWISE RECONSTRUCTION HEAD

After L layers of sparse message passing, the model must predict edge types for *all* node pairs, including those not present in the sparse communication graph. We accomplish this through a bilinear pairwise decoder.

Sparse node embeddings are first scattered back to dense format $H \in \mathbb{R}^{B\times N\times d}$. For each node pair (i, j) and edge class k, the edge logits are:

$$\Delta E_{ij,k} = \frac{h_i^\top W_k h_j}{\sqrt{d}} + b_k(e_{ij}), \tag{2}$$

where $W_k \in \mathbb{R}^{d\times d}$ is a per-class bilinear weight and $b_k(e_{ij})$ is an edge-specific bias. For node pairs in the sparse communication graph, b_k is computed from the learned edge representation; for absent pairs, a learnable null-edge embedding provides the default bias.

Node type logits are computed via an MLP: $\Delta X_i = \text{MLP}_X(h_i)$.

3.4 RESIDUAL PREDICTION AND FRAMEWORK INTEGRATION

Following standard practice, we predict residuals ($\hat{X} = X_t + \Delta X_\theta$, $\hat{E} = E_t + \Delta E_\theta$) and enforce symmetry and zero self-loops for undirected graphs.

SGen is a drop-in backbone replacement for different discrete generation frameworks. We integrate it into: (1) **DiGress** (Vignac et al., 2022), a discrete diffusion model with Markov chain corruption; and (2) **SimGFM** (Luo et al., 2026), a simplified discrete flow matching framework that uses a vf-denoiser velocity field (Gat et al., 2024), endpoint-focused polynomial scheduler. In both cases, we replace only the neural backbone while preserving the generative process.

Training and sampling interface. SGen returns $Z = (\hat{X}, \hat{E})$, unnormalized logits for clean node categories and the edge categories of every node pair. These outputs retain the original generator's loss and sampling interface.

Algorithm 1 shows how this interface is used. During training, the original algorithm corrupts a training graph over all node pairs. SGen extracts the non-null communication edges, computes logits Z, and passes them to the original loss with the clean graph as its target.

During generation, the original algorithm starts from its noise prior. Softmax converts Z into clean-category probabilities for each node and pair. The original sampling rule, UPDATE, uses these probabilities to construct and sample the next-state distribution. SGen then rebuilds its communication edges from the new graph. A null pair can therefore become an edge: sparse intermediate computation does not restrict the generator's candidate pairs.

Algorithm 1 SGen's logit interface

Training
$G_t \leftarrow$ original noising of G_{data}
$Z \leftarrow \text{SGEN}(G_t, t)$ ▷ logits
Train on Z with the original loss
Generation
$G \leftarrow$ original noise prior
for $t \to s$ in the original schedule **do**
  $Z \leftarrow \text{SGEN}(G, t)$ ▷ logits
  $G \leftarrow \text{Update}(G, \text{softmax}(Z), t, s)$
end for

We provide an analysis of why sparse-to-dense works (receptive field coverage, global state as virtual node, expressiveness considerations) in Appendix E.

3.5 A BAYES VIEW THROUGH CONDITIONAL LINK PREDICTION

Theorem 3.1 (From link prediction to denoising). *At fixed N, t, let Y_{ij} and $A_{ij,t}$ be the clean and current categorical edge states, including k_{null}, and $C_{ij,t}$ the remaining noisy graph, excluding both directions of (i, j). Assume $\Pr(A_{ij,t} = a \mid Y_{ij} = k, C_{ij,t}, t) = K_t(k, a)$ for a row-stochastic channel. Write $r_t(k) = \Pr(Y_{ij} = k \mid C_{ij,t}, t)$ and $m_t(a) = \sum_k r_t(k) K_t(k, a)$. For $m_t(a) > 0$,*

$$\eta_{t,a}(k) := \Pr(Y_{ij} = k \mid C_{ij,t}, A_{ij,t} = a, t) = \frac{r_t(k) K_t(k, a)}{\sum_b r_t(b) K_t(b, a)}. \tag{3}$$

For any strictly positive link predictor $\widehat{r}_t$, let $\widehat{\eta}_{t,a}$ be its correction by the same channel. Then

$$\mathbb{E}_{a \sim m_t} D_{\text{KL}}(\eta_{t,a} \| \widehat{\eta}_{t,a}) \le D_{\text{KL}}(r_t \| \widehat{r}_t). \tag{4}$$

The proof and its consequences for SGen's learned logits, local approximation, and bilinear decoding are in Appendices E.1–E.4.

3.6 SCALING TO LARGER GRAPHS

SGen reuses shared node embeddings across terminal blocks of b rows against all N columns, covering every pair with the same weights and bias rule. DiGress+SGen reaches 500-node Proteins and 5,037-node Point Clouds graphs (Section 4.4).

Theorem 3.2 (Terminal blocking and output cost). *At fixed feature and category dimensions in the unit-cost model, SGen's neural forward cost is $O(L(N + |\mathcal{E}_t|) + N^2)$. Its $O(N^2)$ terminal head matches the $\Omega(N^2)$ work required to explicitly emit all-pair scores. Sequentially consuming and releasing b-row blocks, $1 \le b \le N$, uses $O(bN)$ live terminal pairwise activations. In exact arithmetic, globally normalized additive losses preserve loss and gradient under fixed-parameter accumulation, complete shared-backbone differentiation, and counting node terms once. Sampling preserves the transition distribution when blocks retain the original category probabilities, transition rule, and undirected-edge conventions. Graph-state, auxiliary-feature, and backbone storage are separate; auxiliary-feature computation is excluded from the work bound.*

Appendix E.5 gives the execution assumptions and proof.

4 EXPERIMENTS

We test generation quality, resource savings from sparse communication, and scaling through terminal representation reuse. Evaluation protocols, run counts, and baseline sources are in Appendix G. Code and configurations are available at `https://anonymous.4open.science/r/SGen/`.

Table 1: Comparison on QM9 (without explicit H) and QM9H (with explicit H). DeFoG, DiGress, and DiGress + GNN$^+$ use 500 sampling steps; SimGFM and SimGFM + GNN$^+$ use 10 steps on QM9 and 200 steps on QM9H. GNN$^+$ uses **2.0M** parameters in both settings, vs. **8.2M** (QM9) and **6.4M** (QM9H) for the DiGress/SimGFM GT. **Bold**: column best per dataset, excluding training data; underline: + GNN$^+$ strictly improves on its matching dense backbone. GNN$^+$ denotes our SGen backbone (§3.2).

Dataset	Model	Params	Valid↑	Rel. Valid↑	Unique↑	FCD↓
QM9	Training set	–	99.3	99.5	99.2	0.03
	SPECTRE (Martinkus et al., 2022)	–	87.3	–	35.7	–
	GraphNVP (Madhawa et al., 2019)	–	83.1	–	99.2	–
	GDSS (Jo et al., 2022)	–	95.7	–	98.5	2.90
	GRAPHARM (Kong et al., 2023)	–	90.3	–	95.6	1.22
	DisCo (Xu et al., 2024)	–	99.3	–	–	–
	GruM (Jo et al., 2023)	–	99.2	–	96.7	**0.11**
	CatFlow (Eijkelboom et al., 2024)	–	99.8	–	**100.0**	0.44
	COMETH (Siraudin et al., 2024)	–	99.6	–	96.8	0.25
	SID (Boget, 2025)	–	99.7	–	–	0.50
	CID (Boget, 2025)	–	**99.9**	–	–	1.76
	DeFoG (Qin et al., 2024)	–	99.3	99.4	96.3	0.12
	DiGress (Vignac et al., 2022)	8.2M	99.0	–	96.2	–
	DiGress + GenGNN (dense) (Revolinsky et al., 2026)	–	99.49 ± 0.36	–	–	–
	DiGress + GNN$^+$ (Ours)	**2.0M**	99.5 ± 0.0	99.5 ± 0.1	96.8 ± 0.0	0.18 ± 0.01
	SimGFM (Luo et al., 2026)	8.2M	99.5 ± 0.0	**99.7** ± 0.0	95.0 ± 0.2	0.92 ± 0.00
	SimGFM + GNN$^+$ (Ours)	**2.0M**	99.4 ± 0.1	99.6 ± 0.0	96.1 ± 0.0	0.53 ± 0.01
QM9H	Training set	–	97.8	98.9	99.9	0.01
	DeFoG (Qin et al., 2024)	–	98.0	98.8	96.7	0.05
	AutoGraph (Chen et al., 2025)	–	97.7	–	96.7	–
	DiGress (Vignac et al., 2022)	6.4M	95.4	–	97.6	–
	DiGress + GNN$^+$ (Ours)	**2.0M**	95.9 ± 0.0	97.8 ± 0.1	98.1 ± 0.0	0.67 ± 0.00
	SimGFM (Luo et al., 2026)	6.4M	98.4 ± 0.1	99.2 ± 0.0	97.0 ± 0.3	**0.04** ± 0.00
	SimGFM + GNN$^+$ (Ours)	**2.0M**	**99.6** ± 0.0	**99.7** ± 0.1	**98.3** ± 0.0	0.89 ± 0.00

4.1 Experimental Setup

We evaluate molecular generation on QM9/QM9H, GuacaMol, and MOSES; structural generation on Planar, Tree, and SBM; and large-graph generation on Proteins and Point Clouds. Within DiGress and SimGFM, backbone replacements retain the framework's objective and sampling rule. Comparisons include the original Graph Transformer (GT), dense GenGNN, and published generators under the corresponding benchmark protocols. Table 8 lists dataset-specific model and training settings; evaluation protocols and baseline sources are in Appendices G.1 and G.2.

4.2 Main Results: Sparse GNNs vs. Dense Backbones

Baseline values follow the corresponding literature (Vignac et al., 2022; Luo et al., 2026; Qin et al., 2024; Revolinsky et al., 2026); dashes denote unavailable metrics.

Molecular quality. SGen uses 2.0M parameters versus 8.2M on QM9 and 6.4M on QM9H for GT (Table 1). DiGress+SGen improves Validity and Uniqueness on both datasets. At matched SimGFM budgets, QM9 FCD decreases from 0.92 to 0.53, Uniqueness increases from 95.0% to 96.1%, and Validity decreases by 0.1 percentage point. On QM9H, Validity rises from 98.4% to 99.6% and Uniqueness from 97.0% to 98.3%, while FCD worsens from 0.04 to 0.89. Comparisons with SparseDiff are reported in Table 11.

Structural and larger molecular benchmarks. DiGress+SGen improves V.U.N. from 77.5/90.0/60.0% to 86/97.0/80.5% on Planar/Tree/SBM (Table 2). SimGFM+SGen matches GT on Tree (99.5%) and has lower V.U.N. on Planar (99.5% versus 100%) and SBM (87.0% versus 90.5%). Table 15 adds source-reported SparseDiff structural results. On GuacaMol and MOSES, both integrations improve the GuacaMol FCD similarity score and MOSES FCD distance, while some Validity, Novelty, and scaffold metrics favor GT. For example, SimGFM MOSES scaffold similarity falls from 17.3 to 13.7. GuacaMol and MOSES results are reported in Table 12, with SGen standard deviations and additional baselines in Tables 13 and 14.

Table 2: Generation quality on structural benchmarks (Planar, Tree, SBM). V.U.N. is the percentage of generated graphs that are valid, unique, and novel (↑); Ratio = average MMD ratio (↓, closer to 1 is better). **Bold** = best in column, underline = second-best (Train set excluded). The DiGress + GenGNN (dense) row is reproduced from Revolinsky et al. (2026).

	Planar (N=64)		Tree (N=64)		SBM	
Model	V.U.N.↑	Ratio↓	V.U.N.↑	Ratio↓	V.U.N.↑	Ratio↓
Train set	100	1.0	100	1.0	85.9	1.0
GraphRNN (You et al., 2018)	0.0	490.2	0.0	607.0	5.0	14.7
GRAN (Liao et al., 2019)	0.0	2.0	0.0	607.0	25.0	9.7
BiGG (Dai et al., 2020)	5.0	16.0	75.0	5.2	10.0	11.9
GraphGen (Goyal et al., 2020)	7.5	210.3	95.0	33.2	5.0	48.8
AUTOGRAPH (Chen et al., 2025)	87.5	**1.5**	–	–	**92.5**	3.4
EDGE (Chen et al., 2023)	0.0	431.4	0.0	850.7	0.0	51.4
BwR (EDP-GNN) (Diamant et al., 2023; Niu et al., 2020)	0.0	251.9	0.0	11.4	7.5	38.6
HSpectre (Bergmeister et al., 2023)	95.0	2.1	**100.0**	4.0	75.0	10.5
GruM (Jo et al., 2023)	90.0	1.8	–	–	85.0	**1.1**
DisCo (Xu et al., 2024)	83.6	–	–	–	66.2	–
Cometh (Siraudin et al., 2024)	92.5	–	–	–	77.0	–
Cometh-PC (Siraudin et al., 2024)	99.5	–	–	–	–	–
CatFlow (Eijkelboom et al., 2024)	80.0	–	–	–	85.0	–
DeFoG (Qin et al., 2024)	99.5	1.6	96.5	1.6	90.0	4.9
DiGress (Vignac et al., 2022)	77.5	5.1	90.0	1.6	60.0	1.7
DiGress + GenGNN (dense) (Revolinsky et al., 2026)	88 ± 7.5	3.9 ± 0.5	89 ± 7.2	**1.4** ± 0.3	79.5 ± 5.3	4.2 ± 0.8
DiGress + GNN$^+$ (Ours)	86 ± 2.5	3.7 ± 2.3	97.0 ± 1.9	1.6 ± 0.4	80.5 ± 7.0	3.1 ± 0.4
SimGFM (Luo et al., 2026)	**100.0** ± 0.0	9.3 ± 2.6	99.5 ± 1.0	1.5 ± 0.2	90.5 ± 4.0	3.2 ± 0.5
SimGFM + GNN$^+$ (Ours)	99.5 ± 1.0	6.6 ± 2.4	99.5 ± 1.0	1.5 ± 0.5	87.0 ± 6.0	3.4 ± 0.8

Table 3: Computational cost on one RTX 3090 (mean ± std). Training: s/recorded epoch; sampling: seconds for 40 structural graphs, 10,000 QM9/QM9-H/GuacaMol molecules, or 25,000 MOSES molecules. Peak Mem: PyTorch allocator GB. Parentheses: raw GT-relative speedups or memory reductions; halve DiGress Tree/Planar training speedups. Matched-batch protocols: Appendix G.6.

			Training				Sampling			
Dataset	Backbone	Params	Time (s)		Peak Mem (GB)		Time (s)		Peak Mem (GB)	
SimGFM										
Tree	GT	8.9M	1.05 ±0.00		18.4		26.28 ±0.07		1.3	
	SGen	7.3M	**0.38** ±0.04	(2.8×)	**1.5**	(−92%)	**7.09** ±0.05	(3.7×)	**0.2**	(−85%)
Planar	GT	8.9M	1.04 ±0.01		18.2		25.52 ±0.00		1.1	
	SGen	8.0M	**0.32** ±0.04	(3.3×)	**2.6**	(−86%)	**7.41** ±0.05	(3.4×)	**0.3**	(−73%)
DiGress										
Tree	GT	8.9M	2.55 ±0.02		18.1		207.02 ±0.65		1.1	
	SGen	7.3M	**0.33** ±0.04	(7.8×)	**1.5**	(−92%)	**36.73** ±0.70	(5.6×)	**0.2**	(−82%)
Planar	GT	8.9M	2.56 ±0.04		18.1		213.56 ±1.28		1.1	
	SGen	8.0M	**0.34** ±0.05	(7.5×)	**2.6**	(−86%)	**64.25** ±4.03	(3.3×)	**0.3**	(−73%)
QM9	GT	8.2M	34.60 ±0.39		7.0		619.13 ±26.55		1.1	
	SGen	2.0M	21.34 ±0.68	(1.6×)	1.5	(−79%)	110.10 ±0.75	(5.6×)	0.6	(−45%)
QM9-H	GT	6.4M	OOM		OOM		2882.13 ±136.02		5.2	
	SGen	2.0M	37.01 ±0.42	–	1.7	–	332.84 ±2.28	(8.7×)	1.4	(−73%)
GuacaMol	GT	16.3M	10547.25 ±518.57		22.0		19646.67 ±379.73		1.7	
	SGen	9.7M	6557.40 ±38.58	(1.6×)	0.8	(−96%)	3194.94 ±10.86	(6.1×)	0.3	(−82%)
MOSES	GT	16.3M	3345.27 ±32.52		17.5		11131.62 ±283.77		2.2	
	SGen	9.7M	1241.03 ±72.32	(2.7×)	2.6	(−85%)	1156.31 ±0.91	(9.6×)	0.5	(−77%)

Efficiency. On Tree and Planar, SGen gives 2.8–3.9× effective training and 3.3–5.6× sampling speedups, with 86%–92% lower peak training memory. Under DiGress, matched-batch molecular speedups are 1.6–2.7× for training and 5.6–9.6× for sampling where both backbones run (Table 3). Table 17 extends the Tree/Planar resource comparison to dense GNN$^+$, GenGNN, and SparseDiff. Epoch normalization and measurement protocols are in Appendix G.6.

4.3 Ablation Studies

Backbone component ablations. Table 4 removes one component at a time on QM9 under both frameworks. In DiGress/SimGFM, freezing the global state increases FCD by 0.12/0.10, while

Table 4: Ablation study on QM9. Each row removes one component from SGen. Δ columns report the change relative to the full model within each framework. Mean $\pm$ std over 3 runs. "$-$Global u update" freezes the pooling-based global state (Eq. 22) at $u^{(0)}$; "$-$Residual prediction" removes the output residual $\hat{E} = E_t + \Delta E_\theta$; "$-$FiLM conditioning" removes the affine modulation (Eq. 23).

	DiGress				*SimGFM*			
Ablation	FCD ↓	ΔFCD	Valid ↑	ΔValid	FCD ↓	ΔFCD	Valid ↑	ΔValid
SGen	0.18 ± 0.01	—	99.5 ± 0.0	—	0.53 ± 0.01	—	99.4 ± 0.1	—
$-$RRWP	0.20 ± 0.01	$+0.02$	99.4 ± 0.0	-0.1	0.51 ± 0.02	-0.02	99.3 ± 0.0	-0.1
$-$Global u update	0.30 ± 0.02	$+0.12$	99.4 ± 0.0	-0.1	0.63 ± 0.02	$+0.10$	99.3 ± 0.0	-0.1
$-$Residual prediction	0.24 ± 0.00	$+0.06$	99.4 ± 0.0	-0.1	0.58 ± 0.02	$+0.05$	99.4 ± 0.1	$+0.0$
$-$FiLM conditioning	0.47 ± 0.01	$+0.29$	68.3 ± 0.3	-31.2	1.66 ± 0.04	$+1.13$	40.9 ± 0.4	-58.5

Table 5: Architecture controls on QM9 under SimGFM (mean $\pm$ std). Dense GNN$^+$ and SGen share layers, widths, conditioning, gates, and head; only communication support differs.

Backbone	FCD ↓	Valid ↑	Rel. val. ↑	Unique ↑
GT (8.2M)	$0.92_{\pm 0.00}$	**99.5**$_{\pm 0.0}$	**99.7**$_{\pm 0.0}$	$95.0_{\pm 0.2}$
Dense GNN$^+$	**0.41**$_{\pm 0.03}$	$99.3_{\pm 0.1}$	$99.5_{\pm 0.1}$	$95.8_{\pm 0.2}$
SGen	$\underline{0.53}_{\pm 0.01}$	$\underline{99.4}_{\pm 0.1}$	$\underline{99.6}_{\pm 0.0}$	**96.1**$_{\pm 0.0}$
GT + ReZero	$0.81_{\pm 0.01}$	**99.5**$_{\pm 0.1}$	$\underline{99.6}_{\pm 0.1}$	$95.2_{\pm 0.0}$
GT (2.0M)	$0.91_{\pm 0.05}$	$97.9_{\pm 0.1}$	$98.8_{\pm 0.0}$	$\underline{95.9}_{\pm 0.2}$

Table 6: Input-only conditioning retains timestep and graph conditions at the encoders, while removing global-state updates and layerwise FiLM. Full metrics: Table 18.

Dataset	Metric	SGen	Input-only
QM9	FCD ↓	$0.53_{\pm 0.01}$	$0.58_{\pm 0.01}$
QM9-H	FCD ↓	$0.89_{\pm 0.00}$	$0.91_{\pm 0.02}$
Planar	V.U.N. ↑	$99.5_{\pm 1.0}$	$95.5_{\pm 2.5}$
Tree	V.U.N. ↑	$99.5_{\pm 1.0}$	$98.5_{\pm 3.0}$

removing output residuals increases it by $0.06/0.05$; Validity changes by at most 0.1 percentage point. Removing RRWP changes FCD by $+0.02/-0.02$. Removing FiLM reduces Validity from $99.5/99.4\%$ to $68.3/40.9\%$. The input-only replacement below retains the original timestep and graph conditions while changing how they enter the backbone, distinguishing the conditioning inputs from the layerwise modulation mechanism.

Operator and model-size controls. To test whether the quality differences can be explained by the GNN$^+$ operator, residual initialization, or parameter count rather than communication support, we compare SGen with dense GNN$^+$, GT with ReZero, and a parameter-matched GT under SimGFM. On QM9, SGen achieves FCD 0.53 ± 0.01, compared with 0.92 ± 0.00 for GT, 0.81 ± 0.01 for GT with ReZero, and 0.91 ± 0.05 for a parameter-matched 2.0M GT (Table 5). Complete GNN$^+$ achieves lower FCD, 0.41 ± 0.03, while SGen has higher Validity and Uniqueness. The matched communication-efficiency control in Table 16 separately quantifies the cost of retaining complete communication.

Conditioning and component tests. Under SimGFM, an input-only variant concatenates the original timestep and graph conditions to node and edge encoder inputs, replacing dynamic global updates and layerwise FiLM while keeping hidden widths, layers, and the head fixed. It retains effective generation on four benchmarks (Table 6). FCD increases from $0.53/0.89$ to $0.58/0.91$ on QM9/QM9-H, and Tree V.U.N. falls from 99.5 to 98.5. On Planar, V.U.N. decreases from 99.5 to 95.5, while Ratio improves from 6.6 to 5.3. These results show that input-level conditioning can support the sparse backbone without dynamic global updates or layerwise FiLM. Implementation details and complete replacement metrics are in Appendix G.7 and Table 18.

Support and high-noise training. Measured communication support remains sparse at all five tested noise levels on Tree and Planar (Table 9 in Appendix C). Under the same high-noise training weighting, Tree V.U.N. decreases to 78.0 for GT and 81.5 for SGen from 99.5 for both. SGen's Ratio remains 1.5, while GT's increases to 2.3 (Table 19).

Table 7: Generation on Proteins and Point Clouds; all metrics are lower-is-better. Training splits: Proteins, 587 graphs, $N_{\text{avg}} \approx 258$, $N_{\max} = 500$; Point Clouds, 26 graphs, $N_{\text{avg}} \approx 1{,}332$, $N_{\max} = 5{,}037$. SGen: five-seed mean and population standard deviation. MMD standard deviations appear below means. Dashes denote unavailable source values. Baseline sources, dataset-specific Ratio definitions, and OOM/OOT: Appendix G.8.

	Proteins					Point Clouds				
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	0.0040	0.1475	0.5851	0.0152	62.1			OOM		
GRAN	0.0479	0.1234	0.3458	0.0125	77.7	0.0201	0.4330	0.2625	**0.0051**	19.1
SPECTRE	0.0056	0.0843	0.0267	0.0052	12.5			OOM		
EDGE	0.1863	0.3406	0.6786	0.1075	274.5	0.4441	0.3298	1.0730	0.4006	104.7
GraphGen	0.0159	0.1677	0.3789	0.0181	58.1			OOM		
BiGG	0.0070	0.1150	0.4696	0.0067	50.1	0.0994	0.6035	0.3633	0.1589	38.2
DiGress	0.0041	0.0489	0.1286	0.0018	16.2			OOM		
GruM	0.0019	0.0660	0.0345	0.0030	8.2			OOM		
GEEL	0.2110	0.3753	0.1768	0.1689	287.9	–	–	–	–	–
ESGG	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**
SparseDiff	0.0015 ±0.0003	0.0335 ±0.0033	0.0053 ±0.0078	0.0014 ±0.0002	$3.6_{\pm 1.1}$	OOM / OOT				
DiGress + SGen	0.0008 ±0.0002	0.0268 ±0.0030	0.0116 ±0.0037	**0.0012** ±0.0001	$3.1_{\pm 0.5}$	0.0782 ±0.0050	0.8476 ±0.1280	0.2645 ±0.0374	0.0419 ±0.0018	$22.8_{\pm 2.8}$

4.4 Scaling to Larger Graphs

Sparse intermediate propagation removes the layerwise all-pair bottleneck, but materializing the terminal prediction can still dominate memory as N grows. We therefore test whether terminal representation reuse carries the same sparse-to-dense design to large graphs without altering the underlying generator. Using the DiGress integration, we evaluate Proteins and Point Clouds with up to 500 and 5,037 nodes, respectively, following AutoGraph's data splits, preprocessing, evaluator, and Ratio definitions (Chen et al., 2025). SGen results use five independent seeds, and detailed run settings and metric definitions are given in Appendix G.8.

On Proteins, SGen achieves Ratio 3.1 ± 0.5, compared with 16.2 for DiGress, 3.6 ± 1.1 for SparseDiff, and 2.3 for AutoGraph (Table 7). On Point Clouds, its Ratio is 22.8 ± 2.8, lower than BiGG and EDGE and higher than GRAN, ESGG, and AutoGraph. The SGen integration thus reaches the 5,037-node scale through sparse encoding and terminal representation reuse while retaining the original DiGress corruption process, objective, and sampling rule.

5 Conclusion

SGen decouples sparse intermediate communication from all-pair edge prediction in discrete graph generators. Unlike dense message-passing backbones, it changes the communication support, not only the operator. A conditional link-prediction analysis interprets this design by separating structural context from current-edge evidence and relating logit approximation to denoising error. Reusing shared representations across terminal blocks extends DiGress to larger graphs without changing the generation rules; matched controls quantify resource savings and quality trade-offs.

Limitations. Quality depends on the dataset and metric, and specialized generators retain lower Point Clouds Ratios. Shared node embeddings and pooled global context may lose pair-specific information. Terminal blocking reduces simultaneous output activations; the retained dense graph states still require quadratic storage. A block-size resource sweep and numerical checks of blocked execution remain unreported; the current results do not establish performance beyond the evaluated models and scales.

AI USE STATEMENT

Generative AI tools were used for language editing and formatting of tables, figures, and LaTeX. All methods, theoretical analyses, and experimental results were developed and produced by the authors, who take full responsibility for the content of this paper.

REFERENCES

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence*, pp. 1352–1361. PMLR, 2021.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. *arXiv preprint arXiv:2312.11529*, 2023.

Yoann Boget. Simple and critical iterative denoising: A recasting of discrete diffusion in graph generation. *arXiv preprint arXiv:2503.21592*, 2025.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

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.

Dexiong Chen, Markus Krimmel, and Karsten Borgwardt. Flatten graphs as sequences: Transformers are scalable graph generators. *arXiv preprint arXiv:2502.02216*, 2025.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pp. 2302–2312. PMLR, 2020.

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*, pp. 7939–7959. PMLR, 2023.

Floor Eijkelboom, Grigory Bartosh, Christian A Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of the web conference 2020*, pp. 1253–1263, 2020.

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*, pp. 10362–10383. PMLR, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. URL https://arxiv.org/abs/1611.07308.

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*, pp. 17391–17408. PMLR, 2023.

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. *Advances in neural information processing systems*, 32, 2019.

Chunyu Luo, Yuankai Luo, Xiao-Ming Wu, and Lei Shi. Simgfm: Simplifying discrete flow matching for graph generation, 2026. URL https://openreview.net/forum?id=eCftYujQkK.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems*, 37:97650–97669, 2024.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 41290–41310, 2025. URL https://proceedings.mlr.press/v267/luo25h.html.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pp. 23321–23337. PMLR, 2023.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in neural information processing systems*, volume 32, 2019.

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.

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*, pp. 4474–4484. PMLR, 2020.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Yiming Qin, Clement Vignac, and Pascal Frossard. Sparse training of discrete diffusion models for graph generation. *arXiv preprint arXiv:2311.02142*, 2023.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*, 2024.

Jay Revolinsky, Harry Shomer, and Jiliang Tang. Are expressive encoders necessary for discrete graph generation? *arXiv preprint arXiv:2603.08825*, 2026. Version 1: https://arxiv.org/abs/2603.08825v1.

Antoine Siraudin, Fragkiskos D Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. *arXiv preprint arXiv:2406.06449*, 2024.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

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.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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. *Advances in Neural Information Processing Systems*, 37:79704–79740, 2024.

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*, pp. 5708–5717. PMLR, 2018.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4be49c79f233b4f4070794825c323733-Abstract.html.

A HYPERPARAMETER DETAILS

In Table 8, *marg.* samples categories from empirical training-set marginals; *absorb.* puts all probability mass on one absorbing category. QM9, QM9-H, Planar, Tree, GuacaMol, and MOSES use marginal noise; SBM uses absorbing noise. These original benchmark settings do not specify the additional large-graph models and runs.

The standard benchmark (dataset $\times$ framework) pairs in Table 8 share the backbone design. Only hidden dimension, dropout, batch size, training duration, diffusion/sampling steps, and RRWP steps vary by dataset. GNN switches (FiLM placement, ReZero gates, norm placement, etc.) remain fixed across runs; the table lists only dataset-dependent settings.

Table 8: Hyperparameters for SGen across datasets and frameworks. Items not listed (FiLM placement, ReZero gates, norm placement, optimizer, learning rate, weight decay, gradient clip, EMA decay, head type, etc.) are held fixed across all runs.

	QM9	QM9-H	Planar	Tree	SBM	GuacaMol	MOSES
Shared: DiGress / SimGFM							
Hidden dim d (node)	128	128	256	256	256	256	256
Edge dim $(0.25d)$	32	32	64	64	64	64	64
Global dim $(0.5d)$	64	64	128	128	128	128	128
Message-passing layers L	9	10	10	10	8	12	12
FFN multiplier	2.0	2.0	2.0	1.5	2.0	2.0	2.0
RRWP steps	20	20	20	20	32	32	32
Noise distribution	marg.	marg.	marg.	marg.	absorb.	marg.	marg.
Batch size	1024	512	64	64	64	128	512
DiGress							
Diffusion steps T	500	500	1000	1000	1000	500	500
SimGFM							
Sampling steps	10	200	200	200	200	200	200

B GATEDGCN$^+$ INSTANTIATION OF STAGE 1

Stage 1 (Section 3.2) uses the standard $(\mathrm{MSG}, \mathrm{AGG}, \mathrm{UPDATE})$ template (Eqs. 18–19). Any edge-aware GNN$^+$ operator (Luo et al., 2024; 2025) fits this template. All experiments use *GatedGCN*$^+$ (Bresson & Laurent, 2017; Luo et al., 2025), defined below.

Message and edge update. For each edge $(i, j) \in \mathcal{E}_t$,

$$e'_{ij} = D_x x_i + E_x x_j + C_e e_{ij}, \qquad \sigma_{ij} = \mathrm{sigmoid}(e'_{ij}), \tag{5}$$

where C_e, D_x, E_x are learned linear projections and σ_{ij} is a per-feature sigmoid edge gate. The pre-aggregation message and the edge update are

$$m_{ij} = \sigma_{ij} \odot B_x x_j, \qquad \hat{e}_{ij} = \mathrm{proj}(e'_{ij}), \tag{6}$$

with learned B_x and a linear projection proj.

Aggregation and node update. Aggregation is gate-normalized mean pooling, and the node update is residual:

$$\mathrm{AGG}\big(\{m_{ij}\}\big) = \frac{\sum_{j\in\mathcal{N}(i)} m_{ij}}{\sum_{j\in\mathcal{N}(i)} \sigma_{ij} + \epsilon}, \qquad \hat{x}_i = A_x x_i + \mathrm{AGG}\big(\{m_{ij}\}\big), \tag{7}$$

with learned A_x and a small constant ϵ for numerical stability.

Mapping to the abstract template. Eqs. (5–7) realize the abstract operators of Stage 1 as $\mathrm{MSG}^\ell(x_i, x_j, e_{ij}) = \sigma_{ij} \odot B_x x_j$, AGG^ℓ as gate-normalized mean, $\mathrm{UPDATE}^\ell_X(x_i, a_i) = A_x x_i + a_i$, and $\mathrm{UPDATE}^\ell_E(e_{ij}, m_{ij}) = \mathrm{proj}(e'_{ij})$. Activation, dropout, and ReZero gates (Eqs. 20–21) process $(\hat{x}_i, \hat{e}_{ij})$; Stages 2 and 3 (global update and conditioned FFN) are unchanged.

C Communication Graph Sparsity on Larger Graphs

The sparse-to-dense design reduces intermediate work only if the communication support extracted from the current noisy state remains sparse throughout generation. We therefore measure $|\mathcal{E}_t|$ at five noise levels on Tree and Planar under the same marginal-noise construction used in the experiments. This isolates the support that controls sparse propagation before the all-pair output head.

Table 9: Communication-graph size on Planar and Tree ($N = 64$) at five noise levels, averaged over 200 independent samples. $|\mathcal{E}_t|$ counts *directed* edges: each undirected edge $\{i, j\}$ contributes (i, j) and (j, i), one aggregation per direction. The dense baseline is $N(N-1) = 4032$. We follow the flow-matching convention that t=1 is the clean data graph and t=0 is a sample from the limit distribution; intermediate t interpolates the per-pair Bernoulli probability. Statistics are reported at the hop-1 receptive-field setting used in the Planar/Tree experiments.

	Planar ($N = 64$)		Tree ($N = 64$)	
Noise t	Avg. $\lvert\mathcal{E}_t\rvert$	Density	Avg. $\lvert\mathcal{E}_t\rvert$	Density
$t = 0.0$	353.63 ± 24.63	8.77%	125.65 ± 14.21	3.12%
$t = 0.2$	355.23 ± 24.92	8.81%	127.31 ± 15.34	3.16%
$t = 0.5$	356.89 ± 21.18	8.85%	128.36 ± 11.64	3.18%
$t = 0.8$	354.65 ± 16.00	8.80%	126.57 ± 9.88	3.14%
$t = 1.0$	355.25 ± 3.41	8.81%	126.00 ± 0.00	3.12%
Complete directed baseline: $N(N-1) = 4032$.				

Means are pinned by the marginal transition. The per-row averages are nearly flat in t because the forward process induces the marginal $E_t = t\,E_1 + (1-t)\,\pi$, and the limit distribution is set to the empirical edge density $\pi_1 = \rho_{\text{data}}$. Hence

$$\mathbb{E}\big[\,|\mathcal{E}_t|\,\big] \;=\; N(N-1)\,\big[\,t\,\rho_{\text{data}} + (1-t)\,\pi_1\,\big] \;=\; N(N-1)\,\rho_{\text{data}},$$

which is independent of t. The nominal densities are $1/32 \approx 3.125\%$ for Tree and approximately 8.82% for Planar; the measured means fluctuate around these values. This expectation uses matching data and noise marginals at fixed N; finite-sample means need not coincide exactly.

Standard deviations match a Bernoulli sanity check. At the noise-only endpoint, unordered pairs are independent Bernoulli draws with the noise marginal. Since each undirected edge contributes two directed messages, the following approximation applies there, with $\pi_t = \rho_{\text{data}}$:

$$\text{std}\big(\,|\mathcal{E}_t|\,\big) \;\approx\; \sqrt{2\,N(N-1)\,\pi_t\,(1-\pi_t)}.$$

At $t = 0$, with $\pi_0 = \rho_{\text{data}}$, this evaluates to 25.5 for Planar (observed 24.63) and 15.6 for Tree (observed 14.21). These values provide a noise-endpoint sanity check, not a test excluding finite-sample variation. At intermediate times, pair probabilities depend on the clean graph, and unconditional variation also includes variation across sampled clean graphs.

Two distinct sources of residual variance at $t = 1$. At $t = 1$ the noise term vanishes and the only remaining source of variation is the dataset's intrinsic edge-count distribution. Tree graphs on $N = 64$ nodes contain exactly $N-1 = 63$ undirected edges by definition, so $|\mathcal{E}_1| = 126$ deterministically and $\text{std} = 0$. This follows from trees being connected and acyclic: every N-node tree has $N-1$ edges, so intrinsic edge-count variance is zero. The forward process at $t = 1$ is the identity; the std at $t = 1$ therefore remains 0 with either 200 or 200,000 samples. Planar graphs do not fix the edge count (planarity permits up to $3N-6 = 186$ undirected edges), and the within-dataset variation accounts for the residual $\text{std} \approx 3.41$ observed at $t = 1$. The endpoint calculations distinguish noise variation from dataset variation; they do not by themselves derive the full intermediate-time variance curve.

Sparsity is preserved across all noise levels. At every t in the table the communication graph is approximately $11\times$ (Planar) and $32\times$ (Tree) sparser than the complete directed graph at $N = 64$. This reduces the number of pairwise messages, while realized speedups also depend on node operations, kernel overhead, auxiliary features, and the all-pair head. A gain increasing with N requires a corresponding sparsity scaling assumption; fixed density alone does not imply it.

Sampling protocol. For each dataset we pool the train, validation, and test splits into a single set of 200 unique graphs at $N = 64$. For every (dataset, t) pair we then draw 200 samples with replacement from this pool under a fixed random seed, with each sample independently noised. The standard deviations reported above therefore aggregate over 200 independent noise realisations per cell.

D Molecular Quality Metrics: Definitions

We use the following molecular quality metrics:

- **Validity**: Percentage of generated molecules that pass RDKit sanitization checks.
- **Uniqueness**: Percentage of valid molecules with unique canonical SMILES strings.
- **Novelty**: Percentage of unique valid molecules not present in the training set.
- **FCD (Fréchet ChemNet Distance)** (Preuer et al., 2018): Distance between generated and reference molecule distributions in pretrained ChemNet activation space. QM9, QM9H, and MOSES report this distance (lower is better); GuacaMol reports a normalized FCD similarity score (higher is better).
- **Atom stability**: Percentage of generated atoms with the expected valency for their type.
- **Bond stability**: Percentage of generated bonds with a chemically valid type for the connected atom types.
- **SNN (Similarity to Nearest Neighbor)**: Average Tanimoto similarity between each generated molecule and its nearest neighbor in the reference set. Higher means greater nearest-reference similarity, not necessarily coverage or diversity.
- **Frag (Fragment similarity)**: Cosine similarity of generated and reference fragment-frequency vectors, measuring similarity in chemical fragments.
- **Scaff (Scaffold similarity)**: Cosine similarity of generated and reference scaffold-frequency vectors, measuring similarity in core structures.

E Conditional Link Prediction Analysis

SGen replaces dense $O(LN^2)$ propagation with L sparse layers and one all-pair head, costing $O(L(N + |\mathcal{E}_t|) + N^2)$ at fixed hidden dimensions, including node updates but excluding auxiliary-feature computation. We first derive a Bayes reduction from conditional edge denoising to link prediction, then connect approximation guarantees to locality and latent-role structure. These conditional results explain the factorisation without claiming universal expressivity or automatic optimization success.

The formal analysis is followed by architectural observations on information flow, bilinear collisions, and task-dependent expressiveness. These distinguish the ideal predictor from its shared neural implementation.

E.1 Bayes decomposition and error transfer

Fix N, t and one unordered pair, and abbreviate $Y = Y_{ij}$, $A = A_{ij,t}$, and $C = C_{ij,t}$. Here C contains noisy node categories and all other unordered edge categories. Auxiliary features computed using the target edge are not added to C: the exclusion defines a conditional probability, not an extra encoder pass. The assumption is $\Pr(A = a \mid Y = k, C, t) = K_t(k, a)$, with a row-stochastic channel K_t. Independent coordinate-wise corruption implies this property without requiring independent edges in the clean data distribution.

Proof of Theorem 3.1. Conditioning throughout on C, t, Bayes' rule applied to $\Pr(Y = k, A = a) = r_t(k)K_t(k, a)$ gives Eq. (3) whenever $m_t(a) > 0$. For the error bound, define joint distributions

$$P(k, a) = r_t(k)K_t(k, a), \qquad \widehat{P}(k, a) = \widehat{r}_t(k)K_t(k, a),$$

and $\widehat{m}_t(a) = \sum_k \widehat{r}_t(k) K_t(k, a)$. The shared channel gives $D_{\mathrm{KL}}(P\|\widehat{P}) = D_{\mathrm{KL}}(r_t\|\widehat{r}_t)$. Applying the KL chain rule in the other order yields the identity

$$D_{\mathrm{KL}}(r_t\|\widehat{r}_t) = D_{\mathrm{KL}}(m_t\|\widehat{m}_t) + \mathbb{E}_{a\sim m_t} D_{\mathrm{KL}}(\eta_{t,a}\|\widehat{\eta}_{t,a}), \tag{8}$$

which implies Eq. (4) by nonnegativity of KL. Strict positivity of $\widehat{r}_t$ ensures that every observed state with $m_t(a) > 0$ also has $\widehat{m}_t(a) > 0$. Zero-probability observations contribute nothing to the expectation. □

Averaging over contexts, times, and pairs under the training distribution preserves the bound. For ordinary categorical cross-entropy, each expected KL is the corresponding predictor's excess risk over its Bayes optimum. The result transfers probability-prediction guarantees on the same noisy-context distribution, not ranking metrics such as AUC. It is an average over current-edge observations, not a pointwise contraction for each observed category.

Implications for SGen. For positive probabilities, the ideal logits are $\log r_t(k) + \log K_t(k, a)$: contextual structure plus current-edge evidence. SGen's shared sparse encoder and all-pair decoder approximate these logits. Lemma E.1 bounds conditional KL error by $\varepsilon^2/2$ when the finite ideal and learned logits differ by at most ε in maximum norm up to a common shift. Proposition E.2 separates model, local-truncation, and neural approximation errors. These bounds depend on approximation accuracy; the bilinear term, edge bias, and input residual jointly approximate the ideal logit sum, with no requirement that each module recover a separate Bayes term. Appendices E.2–E.4 develop the null-edge interpretation, local bound, and latent-role instance.

E.2 Current-state correction and neural approximation

For the marginal categorical channel (Vignac et al., 2022), write

$$K_t(k, a) = \alpha_t \mathbf{1}\{k = a\} + (1 - \alpha_t)\pi_a, \qquad \kappa_t(a) = \log\left(1 + \frac{\alpha_t}{(1 - \alpha_t)\pi_a}\right). \tag{9}$$

Here α_t denotes retained clean signal, independently of the framework's time direction, and π is the noise-category distribution. For $0 \le \alpha_t < 1$ and $\pi_a > 0$, $\log K_t(k, a)$ equals $\kappa_t(a)\mathbf{1}\{k = a\}$ up to a class-independent constant. Consequently,

$$\eta_{t,a} = \mathrm{softmax}(\log r_t + \kappa_t(a)\,\mathrm{onehot}(a))\,. \tag{10}$$

In particular, for $a = k_{\mathrm{null}}$ and $k \neq k_{\mathrm{null}}$ with positive structural probabilities,

$$\log \frac{\eta_{t,a}(k)}{\eta_{t,a}(k_{\mathrm{null}})} = \log \frac{r_t(k)}{r_t(k_{\mathrm{null}})} - \kappa_t(k_{\mathrm{null}}). \tag{11}$$

All currently absent pairs share the same direct observation correction at fixed t; their different predictions must come from structural context. This motivates a shared null-edge treatment without discarding the information in observing a null edge.

The probability formula (3) and the KL identity also cover absorbing channels. Equation (10) is only used where its denominator is positive; at deterministic endpoints or other zero-probability transitions, the probability formula determines the supported categories instead of finite logits.

Relation to the implemented head. The actual logits are $\widehat{E}_{ij,k} = E_{t,ij,k} + h_i^\top W_k h_j/\sqrt{d} + b_k(e_{ij})$, with the symmetry operation from the method applied for undirected edges. These terms jointly approximate the ideal denoiser: node and edge embeddings may both encode the current observation and its context. In particular, the unit input residual is not the time-dependent coefficient $\kappa_t(a)$ by itself. No analytic correction is added to the implementation, and the original training objective is unchanged. The following bound makes the approximation interpretation precise.

Lemma E.1 (Stability of categorical logits)**.** *For finite logits v, w and $p = \mathrm{softmax}(v)$, $q = \mathrm{softmax}(w)$, if $\inf_{c\in\mathbb{R}} \|w - v - c\mathbf{1}\|_\infty \le \varepsilon$, then $D_{\mathrm{KL}}(p\|q) \le \varepsilon^2/2$.*

Proof. Choose the common shift so that $\delta_k = w_k - v_k - c \in [-\varepsilon, \varepsilon]$. Then

$$D_{\mathrm{KL}}(p\|q) = \log \mathbb{E}_{k\sim p} e^{\delta_k} - \mathbb{E}_{k\sim p}\delta_k.$$

For $f(s) = \log \mathbb{E}_{k\sim p} e^{s\delta_k}$, $f''(s)$ is the variance of δ under the exponentially tilted distribution. Since δ has range at most 2ε, $f''(s) \le \varepsilon^2$. Thus $f(1) - f(0) - f'(0) = \int_0^1 (1 - s) f''(s)\, ds \le \varepsilon^2/2$. □

This bounds the loss of approximating the complete denoising logits; it does not assume that separate network modules recover separate Bayes terms. The finite-logit statement applies to positive probabilities. With zero-probability categories it can instead be applied on a common support, provided the predictor enforces that support.

E.3 Local link-prediction theory and denoising error

The SEAL analysis of Zhang & Chen (2018, Theorem 2) bounds the tail of geometrically decaying link-prediction scores using a target pair's enclosing subgraph. To connect that score approximation to denoising, we explicitly assume a calibrated structural-logit approximation below; locality of a ranking heuristic alone would not suffice.

Proposition E.2 (Local approximation transfer)**.** *At fixed N, t, suppose structural scores have the expansion*

$$s_k(C) = \sum_{\ell=1}^{\infty} \xi^\ell f_{k,\ell}(C), \quad |f_{k,\ell}(C)| \le M\lambda^\ell, \quad 0 < \xi < 1, \quad 0 < \xi\lambda < 1.$$

Assume r_t is positive and $\inf_c \|\log r_t - s(C) - c\mathbf{1}\|_\infty \le \varepsilon_{\text{model}}$. For each radius R, assume that the first $J(R)$ terms are computable from the R-hop enclosing subgraph of (i, j) in C. Let s_R be that truncated sum, $\widehat{r}_{t,R} = \mathrm{softmax}(s_R)$, and $\widehat{\eta}_{t,a,R}$ its Bayes correction. Then

$$\mathbb{E}_{a\sim m_t} D_{\mathrm{KL}}(\eta_{t,a} \| \widehat{\eta}_{t,a,R}) \le \frac{1}{2}\left(\varepsilon_{\text{model}} + T_R\right)^2, \qquad T_R = \frac{M(\xi\lambda)^{J(R)+1}}{1-\xi\lambda}. \tag{12}$$

If all relevant probabilities are positive and the network's full logits approximate $s_R + \log K_t(\cdot, a)$ within ε_{NN} in the same shift-invariant norm, then

$$D_{\mathrm{KL}}(\eta_{t,a} \| p_{\theta,ij,t}) \le \frac{1}{2}\left(\varepsilon_{\text{model}} + T_R + \varepsilon_{\text{NN}}\right)^2, \quad p_{\theta,ij,t} = \mathrm{softmax}(\widehat{E}_{ij}). \tag{13}$$

Proof. The geometric tail gives $\|s - s_R\|_\infty \le \sum_{\ell > J(R)} M(\xi\lambda)^\ell = T_R$. The triangle inequality and Lemma E.1 bound $D_{\mathrm{KL}}(r_t \| \widehat{r}_{t,R})$ by $(\varepsilon_{\text{model}} + T_R)^2/2$. Theorem 3.1 transfers this to Eq. (12). For Eq. (13), adding the same finite $\log K_t(\cdot, a)$ preserves the structural-logit difference; another triangle inequality adds ε_{NN}, and the lemma applies to the complete logits. □

When $J(R)$ grows linearly with R, the truncation term decays exponentially, leaving model and neural approximation errors. This is a sufficient condition on the noisy-context posterior, not an assertion that arbitrary graph distributions satisfy it. The enclosing subgraph is pair-specific, whereas SGen reuses node embeddings. As emphasized by the labeling-trick analysis (Zhang et al., 2021), shared node representations need not preserve all joint pair structure. That compression, together with fitting error, belongs in ε_{NN}; increasing L alone does not eliminate it. The global state and positional features provide additional context without changing this distinction.

E.4 A latent-role model for bilinear decoding

Latent-feature link predictors combine a shared encoder with a pair decoder, as in graph auto-encoders (Kipf & Welling, 2016). A categorical stochastic block model gives an exact instance of the bilinear form. Let each node have role $z_i \in \{1, \dots, q\}$, and suppose edges are independent given all roles, with $\Pr(Y_{ij} = k \mid z) = B_{z_i z_j, k} > 0$ and $\sum_k B_{ab,k} = 1$. Assume any node observations depend only on the roles. For an undirected model, $B_{ab,k} = B_{ba,k}$. With $h_i = \mathrm{onehot}(z_i)$ and $(W_k)_{ab} = \sqrt{q} \log B_{ab,k}$, Bayes' rule gives

$$\Pr(Y_{ij} = k \mid z, G_t) = \mathrm{softmax}_k\left(\frac{h_i^\top W_k h_j}{\sqrt{q}} + \log K_t(k, A_{ij,t})\right). \tag{14}$$

Indeed, conditioning on the roles makes the other noisy observations irrelevant to this edge, while the bilinear term is exactly $\log B_{z_i z_j, k}$. A dimension $d \ge q$ realizes the same construction by zero-padding the embeddings and adjusting the scale.

Independent corruption preserves the role-based form:

$$B^{(t)}_{uv,a} = \sum_k B_{uv,k} K_t(k, a) = \alpha_t B_{uv,a} + (1 - \alpha_t)\pi_a \tag{15}$$

for the marginal channel. Thus shared latent roles can support both link prediction and conditional graph denoising. When roles are unobserved, however,

$$r_t(k) = \sum_{u,v} B_{uv,k} \Pr(z_i = u, z_j = v \mid C, t),$$

which involves a joint role posterior, not necessarily a product of two node-wise posteriors. Equation (14) is therefore an exact expressivity example given roles, not a role-recovery guarantee for the GNN. Approximating that structural inference is part of the neural task.

Scope for graph generation. These results characterize conditional edge prediction on categorical states, without factorizing the clean graph distribution. They apply to diffusion or discrete flow-matching paths when the stated channel assumption holds. A general soft probability tensor is not covered merely by taking its argmax. Control of a final generated graph distribution additionally depends on the framework's sampling dynamics; the results do not replace that analysis.

E.5 Terminal blocking: execution conditions and cost

Theorem 3.2 concerns the same SGen model evaluated with different terminal schedules. Fix the current graph, model parameters, and shared node and edge representations. A block contains at most b rows against all N columns and applies the same bilinear weights, pair-specific bias, input residual, masks, and symmetry operation as the full head. This evaluates every candidate pair, including absent and cross-component pairs. For undirected logits, both directional contributions must be combined before a pair is consumed. Terminal activations are released after each block has contributed to the loss, gradient, or next-state update.

Training execution. Let $\mathcal{P}$ contain the pair entries used by the original loss, including its masking and directed- or undirected-pair convention. Suppose the loss is additive over these entries, with the original global normalization Z:

$$\mathcal{L}(\theta) = \mathcal{L}_X(\theta) + \frac{1}{Z} \sum_{(i,j)\in\mathcal{P}} \ell_{ij}(\theta) = \mathcal{L}_X(\theta) + \sum_s \mathcal{L}_s(\theta),$$
$$\mathcal{L}_s(\theta) = \frac{1}{Z} \sum_{(i,j)\in\mathcal{P}_s} \ell_{ij}(\theta). \tag{16}$$

The sets $\mathcal{P}_s$ partition $\mathcal{P}$, and each block uses the same Z, rather than its own number of entries. Gradients accumulate at fixed parameters before one optimizer step, with node loss $\mathcal{L}_X$ counted once. Every block loss must differentiate through its shared backbone computations. These computations can be retained or consistently recomputed; recomputation must preserve any stochastic realization used to define the shared representations. Detaching the embeddings without propagating their accumulated gradients would violate this condition.

Sampling execution. All blocks use the same current graph and shared representations. They retain the original category probabilities, transition rule, masks, and coupling of the two directions of an undirected edge. A partially constructed next graph must not replace the current graph while remaining blocks are evaluated. Under these conditions, blocking changes the evaluation schedule and preserves the transition kernel. Floating-point accumulation and random-number order can differ, so equal seeds need not give bitwise-identical outputs.

Proof of Theorem 3.2. At fixed feature and category dimensions, a sparse layer performs $O(N)$ node/global work and $O(|\mathcal{E}_t|)$ edge work. Summing over L layers gives $O(L(N + |\mathcal{E}_t|))$. The terminal bilinear head performs constant work per candidate pair, for $O(N^2)$ work. An undirected graph has $N(N-1)/2$ candidate pairs; explicitly emitting a fixed-size category-score vector for

each requires $\Omega(N^2)$ work in the standard unit-cost model. Thus the head matches the output-size lower bound in N.

A block holds at most bN pair outputs and their fixed-width terminal intermediates. Sequentially consuming and releasing these gives $O(bN)$ live terminal pairwise activations while covering the same quadratic number of pairs. Every block computes the same scores because its shared representations, parameters, and pairwise operations are unchanged. Equation (16) therefore preserves the loss; differentiating this finite sum gives $\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}_X + \sum_s \nabla_\theta \mathcal{L}_s$. Complete shared-backbone differentiation and one optimizer step preserve the full gradient in exact arithmetic. Finally, unchanged categorical probabilities and the same transition and symmetry rules define the same conditional next-state distribution under the sampling execution above. □

Scope of the resource bounds. The activation bound applies to terminal pairwise computations. Current and next graph states, auxiliary features, and backbone activations retain their own storage costs. A dense-state support scan may take $O(N^2)$ work, and auxiliary-feature computation is accounted for separately. The forward-work bound does not include extra recomputation used to reduce training memory. Optimality concerns terminal output work in N at fixed dimensions, not end-to-end runtime, total memory, or generation quality.

Dense intermediate communication costs $O(LN^2)$ at the same fixed dimensions. Blocking a dense graph transformer or complete-graph GNN's final head leaves its intermediate pairwise states intact; releasing those states requires rescheduling with recomputation or changing the architecture. SGen removes this repeated layerwise pair computation before blocking the terminal output, enabling the large-graph extension while preserving the original generation process.

E.6 Information flow through sparse layers

Node embeddings receive information through sparse propagation on the noisy graph and a global broadcast channel formed by pooling and re-injection.

Reachability on the noisy graph. With L stacked layers, each node's receptive field on the noisy communication graph $\mathcal{E}_t$ covers all nodes within graph distance L. Two qualifications are immediate. First, the relevant distance is measured on the noisy graph $\mathcal{E}_t$, not on the clean graph: at high noise level the noisy graph may fragment, and pure-propagation paths between two clean-graph neighbours may be absent. Second, L measures only the existence of a propagation path, not the magnitude of the signal carried by it. In the implemented gate-normalized aggregation (Eq. 7), a path's influence depends on learned gates, projections, normalization, and nonlinearities. Reachability alone therefore gives neither a uniform signal-strength bound nor an approximation guarantee. Residual paths retain direct access to a node's previous representation.

Global broadcast channel. The global state $u^{(\ell)}$ pools node and edge features (Eq. 22) and modulates each node through FiLM (Eq. 23) in the FFN (Stage 3), after message passing. Two properties follow:

- Every node reads $u^{(\ell)}$ at every layer. Any function of the pooled summaries reaches every node in $O(1)$ propagation hops on the augmented computation graph, regardless of clean-graph diameter or fragmentation of $\mathcal{E}_t$.
- The channel uses a fixed-width shared summary. It can lose task-relevant distinctions, but its dimension alone is not a finite-bit information-capacity bound.

The channels are complementary: pooled context is broadcast in a constant number of computational hops, while sparse paths retain node-specific local context over L layers.

Coverage of pairwise predictions. The target for (i, j) may depend on local structure around i and j, graph-level summaries, or pair-specific long-range correlations expressible as neither. Sparse propagation carries the first kind of information; the global channel carries the second. With no dedicated channel for the third, the model must approximate it through information jointly encodable in h_i, h_j, and e_{ij}. We make no claim about its magnitude in any specific data distribution.

E.7 EXPRESSIVENESS OF THE BILINEAR READOUT

Before symmetrization, the dense head computes residual logit increments

$$\Delta E_{ij,k} \;=\; \frac{h_i^\top W_k h_j}{\sqrt{d}} \;+\; b_k(e_{ij}),$$

where b_k is a learned function of the edge feature e_{ij}. We make two algebraic observations on its capacity.

Collision constraint. If $h_i = h_j$, then $h_i^\top W_k h_a = h_j^\top W_k h_a$ for every node a and every class k. The two logit increments $\Delta E_{ia,k}$ and $\Delta E_{ja,k}$ therefore differ only through the edge-feature term:

$$\Delta E_{ia,k} - \Delta E_{ja,k} \;=\; b_k(e_{ia}) - b_k(e_{ja}).$$

If $h_i = h_j$ *and* $e_{ia} = e_{ja}$, the residual logit increments coincide. Thus, distinguishability of these increments depends jointly on the node-embedding pipeline and edge-feature stream. The final logits also coincide if the current-state residuals and the reverse-direction contributions used in symmetrization coincide.

Absent pairs. For $(i, j) \in \mathcal{E}_t$, the edge-feature residual $b_k(e_{ij})$ uses that pair's learned edge representation. For $(i, j) \notin \mathcal{E}_t$, all pairs share a learned null-edge embedding, so b_k becomes a class-dependent, pair-independent constant. Apart from this constant, absent-pair predictions depend entirely on $h_i^\top W_k h_j$. Recovering true edges removed by forward noise therefore relies on information encoded by the node-embedding pipeline alone. Both the node representations and the bilinear readout constrain which pairwise dependencies can be represented.

E.8 GENERATION VERSUS DISCRIMINATION

Without distinguishing node features, the message-passing backbone is upper-bounded by 1-Weisfeiler–Leman refinement (Xu et al., 2018). Our inputs include atom type, charge, and RRWP positional encodings (Ma et al., 2023); the featureless bound therefore does not tightly characterise the deployed system. We neither formally characterise its expressiveness class nor claim a specific separation.

For both classification and denoising, a representation collision matters when the corresponding conditional targets differ; classification does not require distinct labels for every graph isomorphism class. Here the relevant target is the conditional edge posterior $\eta_{t,a}$. The approximation bounds concern preserving its logits, not distinguishing every non-isomorphic graph.

E.9 WHEN THE ANALYSIS BREAKS

The design rationale breaks down under the following conditions.

1. **Long distances on the noisy communication graph.** A dependency beyond L noisy-graph hops cannot arrive through pure sparse propagation within L layers. Auxiliary features and the global state may carry some of it. Increasing L expands reachability within a connected component but does not by itself ensure accurate representation.
2. **Fragmentation of the noisy graph.** When $\mathcal{E}_t$ is disconnected, pure sparse propagation cannot cross components. The shared global channel and all-pair head remain available, but their existence does not guarantee that cross-component correlations are represented.
3. **Pair-specific information not summarisable through u.** If the denoising target depends on pairwise correlations that are neither local around the pair nor encodable in a $\dim u$-dimensional graph summary — for example, edge labels determined by global combinatorial constraints such as graph colouring or hard constraint-satisfaction structures — the architecture has no dedicated channel for such information. Whether a given application falls into this regime is beyond the scope of this analysis.
4. **Distributions concentrated on backbone collisions.** If the data distribution places significant mass on graphs that the backbone cannot distinguish through its refinement procedure,

and assigns different label distributions to graphs within the same collision class, no readout operating on the resulting node embeddings can recover the distinction. Recovering such distinctions requires additional discriminating information or a richer representation, rather than only a different readout of identical inputs.

These are failure modes of the architectural factorisation, not predictions about any specific dataset.

F BROADER IMPACT

Our measurements show lower runtime and memory use in the reported settings. Energy consumption and environmental impact were not measured. More efficient graph generation could accelerate molecular design for drug discovery and materials science. As with other generative models, more efficient molecule generation carries a dual-use risk: it could be misused to design harmful substances. We believe the benefits of more accessible and efficient molecular generation outweigh these risks, particularly given that existing tools already enable similar capabilities.

G Additional Results and Evaluation Details

These comparisons distinguish controlled backbone experiments, adapted baselines, and published results. Dashes denote unavailable source values, not zero or failure; OOM/OOT denote reported failures. SGen large-graph results and adapted QM9-no-H comparisons use five independent runs. Other entries retain reported uncertainty without inferring run counts.

G.1 Standard-Benchmark Setup

Datasets. We use: (1) **QM9** (Wu et al., 2018) in two settings—*QM9* (heavy atoms only, 4 atom types) and *QM9H* (with explicit hydrogens, 5 atom types)—each containing ∼134K small molecules with up to 9 heavy atoms; (2) **Planar** (Martinkus et al., 2022) (200 planar graphs, 64 nodes); (3) **Tree** (Bergmeister et al., 2023) (100 tree graphs, 64 nodes); (4) **SBM** (Martinkus et al., 2022) (200 stochastic block model graphs, 44–187 nodes); (5) **GuacaMol** (Brown et al., 2019) (large-scale drug-like molecules); (6) **MOSES** (Polykovskiy et al., 2020) (large-scale molecular generation benchmark). We further evaluate large-graph generation on Proteins and Point Clouds (Section 4.4).

Baselines. We compare against: (1) the original **Graph Transformer (GT)** backbone used in DiGress and SimGFM/DeFoG; (2) **PPGN** backbone (Maron et al., 2019) (where applicable); (3) **GenGNN (dense)** (Revolinsky et al., 2026); (4) the full benchmark suite (Luo et al., 2026), including GraphRNN, GRAN, BiGG, GraphGen, AUTOGRAPH, GraphNVP, SPECTRE, GDSS, DiGress, GruM, CatFlow, DisCo, COMETH, GRAPHARM, SID/CID, DeFoG, and SimGFM at different sampling budgets. Note that GenGNN (dense) still operates on full communication graphs, whereas SGen keeps intermediate message passing sparse and only densifies at the output head. Additional SparseDiff comparisons are reported in Tables 11 and 15; the QM9 adaptation study is presented in Appendix G.3.

Standard-benchmark architecture. SGen uses GatedGCN$^+$ layers with global state conditioning and FFN, pre-normalization (LayerNorm), sparse communication graph, RRWP positional encodings, and a bilinear-with-edge-bias head. Hidden dimension d and the number of message-passing layers L vary by dataset and are shared across the two frameworks (DiGress and SimGFM); see Table 8 in Appendix A for exact values. Dimension ratios for node:edge:global features are fixed at $1 : 0.25 : 0.5$, reducing edge and global parameters to mitigate overfitting.

Training. We use AdamW optimizer with learning rate 10^{-4}, weight decay 10^{-12}, gradient clipping at 20, EMA decay 0.999, and train for 1000 epochs on QM9 and synthetic datasets, 500 epochs on GuacaMol and MOSES. Standard quality experiments run on single NVIDIA A100 GPUs; resource comparisons use a single NVIDIA RTX 3090 GPU. Controlled backbone comparisons retain the base framework's optimizer, objective, and sampling budget. We use a cosine learning rate schedule with 50-epoch linear warmup. Resource-measurement protocols and large-graph settings are detailed in Appendices G.6 and G.8.

G.2 Metric Suites and Baseline Inclusion

Benchmark protocols define the metric suites (Brown et al., 2019; Polykovskiy et al., 2020): QM9/QM9-H uses Validity, Relaxed Validity, Uniqueness, and FCD distance; GuacaMol uses Validity, V.U., V.U.N., KL score, and FCD score; MOSES uses Validity, Uniqueness, Novelty, Filters, FCD distance, SNN, and scaffold similarity. Each dataset's suite is fixed across methods.

Appendix D defines the common molecular metrics. Relaxed Validity permits valid formal-charge assignments before sanitization. GuacaMol's V.U. and V.U.N. are joint proportions of all generated samples. KL compares physicochemical-property distributions, Filters uses MOSES medicinal-chemistry filters, and the scaffold metric uses Bemis–Murcko scaffolds.

We include baselines under compatible data and evaluation protocols.

G.3 Supplementary QM9 Adaptation Study

We adapt GraphGen, AutoGraph, and the attributed-graph extension of BiGG to QM9 without explicit hydrogens. The public implementations used for these methods support node and edge labels. We use the same data split, molecular conversion, and evaluator as SGen. The GraphRNN and GRAN configurations considered here generate unlabeled binary topologies; evaluating them under this molecular protocol would require extending their outputs to atom types and categorical bond types.

Table 10 reports our adaptation results over five independent runs, with both SGen integrations as references. AutoGraph's published all-atom results in Table 1 use a separate setting with explicit hydrogens.

Table 10: Supplementary QM9 adaptation study conducted in this work. The three adapted baselines use the same data split, molecular conversion, and evaluator as SGen, without explicit hydrogens. Entries are mean ± standard deviation over five independent runs; SGen rows provide reference results.

Model	Valid ↑	Rel. valid ↑	Unique ↑	FCD ↓
GraphGen (adapted)	$71.1_{\pm 0.4}$	$84.0_{\pm 0.4}$	$81.8_{\pm 0.4}$	$2.13_{\pm 0.07}$
BiGG (adapted)	$75.5_{\pm 0.6}$	$82.9_{\pm 0.7}$	$\mathbf{97.5}_{\pm 0.1}$	$1.18_{\pm 0.05}$
AutoGraph (adapted)	$99.0_{\pm 0.1}$	$99.3_{\pm 0.1}$	$\underline{96.9}_{\pm 0.1}$	$\mathbf{0.10}_{\pm 0.01}$
DiGress + SGen	$\mathbf{99.5}_{\pm 0.0}$	$\underline{99.5}_{\pm 0.1}$	$96.8_{\pm 0.0}$	$\underline{0.18}_{\pm 0.01}$
SimGFM + SGen	$\underline{99.4}_{\pm 0.1}$	$\mathbf{99.6}_{\pm 0.0}$	$96.1_{\pm 0.0}$	$0.53_{\pm 0.01}$

Among these configurations, adapted AutoGraph has the lowest FCD (0.10), and adapted BiGG has the highest Uniqueness (97.5%). DiGress+SGen and SimGFM+SGen have higher Validity (99.5% and 99.4%) than the adapted baselines. This study compares complete generators in one molecular setting; the matched backbone controls in Section 4.3 examine the effects of communication support and architecture.

G.4 Molecular Quality and Reported Uncertainty

Table 11: Molecular comparison with SparseDiff. Dashes denote unreported values. Generation processes and sampling budgets differ across methods.

Dataset	Method	Valid ↑	Unique ↑	FCD ↓
QM9	SparseDiff	$99.2_{\pm 0.1}$	$\underline{96.4}_{\pm 0.1}$	$\mathbf{0.12}_{\pm 0.00}$
	DiGress + SGen	$\mathbf{99.5}_{\pm 0.0}$	$\mathbf{96.8}_{\pm 0.0}$	$\underline{0.18}_{\pm 0.01}$
	SimGFM + SGen	$\underline{99.4}_{\pm 0.1}$	$96.1_{\pm 0.0}$	$0.53_{\pm 0.01}$
QM9-H	SparseDiff	$\underline{97.9}_{\pm 0.13}$	$97.4_{\pm 0.10}$	–
	DiGress + SGen	$95.9_{\pm 0.0}$	$\underline{98.1}_{\pm 0.0}$	$\mathbf{0.67}_{\pm 0.00}$
	SimGFM + SGen	$\mathbf{99.6}_{\pm 0.0}$	$\mathbf{98.3}_{\pm 0.0}$	$0.89_{\pm 0.00}$

The main text characterizes SGen's quality as competitive but metric-dependent. We therefore expand the molecular comparison to complete metric suites, available uncertainty, and additional protocol-compatible baselines before interpreting those trade-offs.

Tables 13 and 14 give reported SGen standard deviations and available AutoGraph and SparseDiff results. Table 12 retains original point estimates; missing error bars do not imply zero variance. SimGFM values come from this study's comparison record and are not relabeled as independently verified results from another public version.

Converting AutoGraph's GuacaMol Validity, Uniqueness, and Novelty gives joint proportions V.U.$=91.6$ and V.U.N.$=89.5$. SparseDiff's percentage SNN is divided by 100 for the $[0, 1]$ scale. On MOSES, both SGen variants have higher Validity, Filters, and SNN and lower FCD distance than SparseDiff, while SparseDiff has higher Novelty and scaffold similarity. On QM9, SparseDiff has lower FCD than DiGress+SGen, whereas the latter has higher Validity and Uniqueness. Thus, the full metric suite is needed to describe these comparisons.

Table 12: Comparison on GuacaMol and MOSES. DiGress, DiGress + GNN$^+$, and DeFoG use 500 sampling steps; SimGFM and SimGFM + GNN$^+$ use 200 steps. GNN$^+$ uses **9.7M** parameters, vs. **16.3M** for the DiGress/SimGFM GT. **Bold**: column best, excluding training data; underline: + GNN$^+$ strictly improves on its matching dense backbone.

GuacaMol

Method	Params	Val.↑	V.U.↑	V.U.N.↑	KL↑	FCD score↑
Training set	–	100.0	100.0	0.0	99.9	92.8
DisCo (Xu et al., 2024)	–	86.6	86.6	86.5	92.6	59.7
COMETH (Siraudin et al., 2024)	–	98.9	98.9	97.6	96.7	72.7
DeFoG (Qin et al., 2024)	–	99.0	99.0	97.9	**97.7**	73.8
DiGress (Vignac et al., 2022)	16.3M	85.2	85.2	85.1	92.9	68.0
DiGress + GNN$^+$ (Ours)	**9.7M**	95.3	95.2	94.5	95.1	77.6
SimGFM (Luo et al., 2026)	16.3M	**99.2**	**99.2**	**98.8**	94.0	80.8
SimGFM + GNN$^+$ (Ours)	**9.7M**	99.1	99.1	98.2	96.2	**84.4**

MOSES

Method	Params	Val.↑	Uniq.↑	Nov.↑	Filt.↑	FCD distance↓	SNN↑	Scaf↑
Training set	–	100.0	100.0	0.0	100.0	0.01	0.64	99.1
DisCo (Xu et al., 2024)	–	88.3	**100.0**	**97.7**	95.6	1.44	0.50	15.1
COMETH (Siraudin et al., 2024)	–	90.5	99.9	92.6	99.1	1.27	0.54	16.0
DeFoG (Qin et al., 2024)	–	**92.8**	99.9	92.1	98.9	1.95	0.55	14.4
DiGress (Vignac et al., 2022)	16.3M	85.7	**100.0**	95.0	97.1	1.19	0.52	14.8
DiGress + GNN$^+$ (Ours)	**9.7M**	86.4	**100.0**	94.9	97.7	1.13	0.55	14.2
SimGFM (Luo et al., 2026)	16.3M	89.4	**100.0**	94.6	98.9	1.08	0.55	**17.3**
SimGFM + GNN$^+$ (Ours)	**9.7M**	89.0	**100.0**	93.1	**99.2**	**0.96**	**0.56**	13.7

Table 13: GuacaMol results with reported uncertainty. AutoGraph values are source-reported. V.U. and V.U.N. are joint proportions of all generated samples; higher values are better for every column.

Model	Valid	V.U.	V.U.N.	KL score	FCD score
AutoGraph	91.6	91.6	89.5	**97.5**	79.2
DiGress + SGen	$95.3_{\pm 0.1}$	$95.2_{\pm 0.1}$	$94.5_{\pm 0.1}$	$95.1_{\pm 0.1}$	$77.6_{\pm 0.2}$
SimGFM + SGen	$\mathbf{99.1}_{\pm 0.1}$	$\mathbf{99.1}_{\pm 0.1}$	$\mathbf{98.2}_{\pm 0.1}$	$96.2_{\pm 0.1}$	$\mathbf{84.4}_{\pm 0.3}$

Table 14: MOSES results with reported uncertainty. AutoGraph values are source-reported. FCD is a distance; other metrics retain the benchmark directions. SNN is on the $[0, 1]$ scale.

Model	Valid ↑	Unique ↑	Novel ↑	Filters ↑	FCD ↓	SNN ↑	Scaf. ↑
AutoGraph	87.4	**100.0**	85.9	98.6	**0.91**	0.55	10.2
SparseDiff	$84.7_{\pm 0.2}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{95.1}_{\pm 0.1}$	$97.0_{\pm 0.2}$	$1.28_{\pm 0.01}$	$0.52_{\pm 0.00}$	$\mathbf{15.5}_{\pm 1.3}$
DiGress + SGen	$86.4_{\pm 0.2}$	$\mathbf{100.0}_{\pm 0.0}$	$94.9_{\pm 0.5}$	$97.7_{\pm 0.6}$	$1.13_{\pm 0.03}$	$0.55_{\pm 0.00}$	$14.2_{\pm 1.1}$
SimGFM + SGen	$\mathbf{89.0}_{\pm 0.2}$	$\mathbf{100.0}_{\pm 0.0}$	$93.1_{\pm 0.1}$	$\mathbf{99.2}_{\pm 0.1}$	$0.96_{\pm 0.02}$	$\mathbf{0.56}_{\pm 0.00}$	$13.7_{\pm 0.9}$

G.5 Structural Quality and Architecture Controls

The structural evidence addresses two separate questions: whole-system comparisons test whether SGen remains competitive beyond molecular data, while architecture controls test whether observed differences arise from communication support rather than the GNN operator or parameter count. We keep these evidence types distinct below.

Table 15 reports source results for Planar and SBM. SparseDiff's Ratio uses only normalized Degree, Clustering, and Orbit MMDs (Qin et al., 2023); we do not recompute all methods on a common descriptor set. Both SGen variants have higher reported V.U.N. The original GenGNN comparison retains the cited version. These whole-system comparisons differ from the matched complete/sparse GNN$^+$ control (Table 5).

The ReZero control replaces only GT residual gates; GT already uses FiLM. Parameter matching reduces node, edge, global, and FFN widths, preserving layers, dense communication, objective, and training settings. Dense GNN$^+$ results are retained where its FCD is better than SGen's.

Table 15: Source-reported structural results. SparseDiff's Ratio averages Degree, Clustering, and Orbit MMD ratios only. Values retain their source evaluation protocols.

Dataset	Method	V.U.N. ↑	Ratio ↓
Planar	SparseDiff	$85.0_{\pm 9.0}$	$\mathbf{1.2}_{\pm 0.4}$
	DiGress + SGen	$\underline{86.0}_{\pm 2.5}$	$\underline{3.7}_{\pm 2.3}$
	SimGFM + SGen	$\mathbf{99.5}_{\pm 1.0}$	$6.6_{\pm 2.4}$
SBM	SparseDiff	$75.0_{\pm 10.0}$	$\mathbf{1.7}_{\pm 0.5}$
	DiGress + SGen	$\underline{80.5}_{\pm 7.0}$	$\underline{3.1}_{\pm 0.4}$
	SimGFM + SGen	$\mathbf{87.0}_{\pm 6.0}$	$3.4_{\pm 0.8}$

G.6 Resource Measurements and Timing Conventions

To attribute measured savings to sparse communication rather than batch-size or implementation differences, we pair architecture-matched complete/sparse controls with batch-matched system measurements.

Tables 16, 17, and 3 use one RTX 3090. Complete and sparse GNN$^+$ share the SparseTensor implementation, layers, hidden widths, conditioning, and output head, and within-dataset settings; storing all directed pairs in SparseTensor still gives complete communication. GenGNN and SparseDiff also change the operator or generation process, making them system comparisons rather than isolated sparsity controls. SGen/GenGNN batch sizes are matched; SGen's memory saving is not used to enlarge its batch.

Structural training times use recorded epochs; sampling times cover 40 graphs. DiGress GT processes each sample twice per recorded epoch, so its raw GT-to-SGen training ratio requires division by two for per-sample comparison. Sampling times are unaffected; other methods do not automatically receive this correction. Peak memory is measured by the PyTorch allocator and reported in GB.

Table 16: Same-architecture communication control on one RTX 3090. Training time: s/recorded epoch; sampling time: s/40 graphs. Peak: PyTorch allocator memory (GB). GT epoch accounting: Appendix G.6.

		Training		Sampling	
Dataset	Backbone	Time	Peak	Time	Peak
Tree	GT	$2.55_{\pm 0.02}$	18.1	$207.02_{\pm 0.65}$	1.1
	Dense GNN$^+$	$0.86_{\pm 0.02}$	18.4	$96.83_{\pm 0.79}$	1.3
	SGen	$0.33_{\pm 0.04}$	1.5	$36.73_{\pm 0.70}$	0.2
Planar	GT	$2.56_{\pm 0.04}$	18.1	$213.56_{\pm 1.28}$	1.1
	Dense GNN$^+$	$0.86_{\pm 0.00}$	19.2	$93.90_{\pm 0.19}$	1.3
	SGen	$0.34_{\pm 0.05}$	2.6	$64.25_{\pm 4.03}$	0.3

Changing complete communication to the current non-null support reduces Tree/Planar peak training memory from $18.4/19.2$ to $1.5/2.6$ GB and training time from 0.86 to $0.33/0.34$ seconds per recorded epoch (Table 16). This matched comparison isolates the resource savings from sparse communication.

Table 17: Structural efficiency on one RTX 3090. Time: s/recorded epoch for training, s/40 graphs for sampling. Peak Mem is measured by the PyTorch allocator in GB (Appendix G.6).

			Training		Sampling	
Dataset	Backbone	Params	Time	Peak Mem	Time	Peak Mem
Tree	GT	8.9M	$2.55_{\pm 0.02}$	18.1	$207.02_{\pm 0.65}$	1.1
	Dense GNN^{+}	7.3M	$0.86_{\pm 0.02}$	18.4	$96.83_{\pm 0.79}$	1.3
	SGen	7.3M	$0.33_{\pm 0.04}$	1.5	$36.73_{\pm 0.70}$	0.2
	GenGNN	4.8M	$0.44_{\pm 0.01}$	6.7	$61.55_{\pm 0.13}$	0.5
	SparseDiff	20.7M	$0.93_{\pm 0.02}$	12.2	$280.17_{\pm 0.60}$	0.6
Planar	GT	8.9M	$2.56_{\pm 0.04}$	18.1	$213.56_{\pm 1.28}$	1.1
	Dense GNN^{+}	8.0M	$0.86_{\pm 0.00}$	19.2	$93.90_{\pm 0.19}$	1.3
	SGen	8.0M	$0.34_{\pm 0.05}$	2.6	$64.25_{\pm 4.03}$	0.3
	GenGNN	4.8M	$0.43_{\pm 0.00}$	6.7	$61.81_{\pm 0.05}$	0.5
	SparseDiff	20.7M	$0.96_{\pm 0.00}$	12.8	$293.10_{\pm 0.01}$	0.7

Matched GT/SGen train/sample batches are QM9 $1024/2048$, QM9-H $512/1024$, GuacaMol $32/64$, and MOSES $256/512$. Sampling produces 10,000 graphs for QM9, QM9-H, and GuacaMol and 25,000 for MOSES. With no preprocessing duplication, molecular times need no division-by-two correction. These resource settings differ from the quality-model hyperparameters (Table 8). Measured speedups exclude potential gains from larger SGen batches.

G.7 Conditioning and High-Noise Ablations

Input-only implementation. For the SimGFM replacement in Section 4.3, only the node and edge encoders' input dimensions change. Their output dimensions, subsequent hidden widths, GNN layers, and output head retain the original configuration. Table 18 reports the complete metrics corresponding to the main-text summary in Table 6.

Table 18: Conditioning-preserving ablation under SimGFM. Input-only concatenates initial timestep and graph conditions at the encoders and removes dynamic global updates and FiLM. Hidden widths and the head are unchanged. Dashes denote inapplicable metrics.

Dataset	Variant	Valid/V.U.N. ↑	Rel. valid ↑	Unique ↑	FCD ↓	Ratio ↓
QM9	SGen	$99.4_{\pm 0.1}$	$99.6_{\pm 0.0}$	$96.1_{\pm 0.0}$	$0.53_{\pm 0.01}$	–
	Input-only	$99.4_{\pm 0.0}$	$99.7_{\pm 0.0}$	$96.2_{\pm 0.1}$	$0.58_{\pm 0.01}$	–
QM9-H	SGen	$99.6_{\pm 0.0}$	$99.7_{\pm 0.1}$	$98.3_{\pm 0.0}$	$0.89_{\pm 0.00}$	–
	Input-only	$98.8_{\pm 0.0}$	$99.3_{\pm 0.0}$	$98.0_{\pm 0.1}$	$0.91_{\pm 0.02}$	–
Planar	SGen	$99.5_{\pm 1.0}$	–	–	–	$6.6_{\pm 2.4}$
	Input-only	$95.5_{\pm 2.5}$	–	–	–	$5.3_{\pm 1.3}$
Tree	SGen	$99.5_{\pm 1.0}$	–	–	–	$1.5_{\pm 0.5}$
	Input-only	$98.5_{\pm 3.0}$	–	–	–	$1.7_{\pm 0.4}$

Communication-support measurements on 64-node Tree and Planar graphs at five noise levels are reported separately in Appendix C.

High-noise training protocol. On Tree, both GT and SGen use the same SimGFM training-time distribution $t = u^2$, $u \sim \mathcal{U}(0, 1)$, where $t = 0$ is the noise endpoint. Each backbone retains its original architecture, noisy-state conditional, and sampler. This changes the weighting of training states while holding the remaining configuration fixed. Table 19 gives the results interpreted in Section 4.3.

Table 19: Tree generation after biasing training toward high noise. SimGFM with $t = u^2$, $u \sim \mathcal{U}(0, 1)$; $t = 0$ is the noise endpoint. The noisy-state conditional, model and sampler are unchanged. GT and SGen use the same training-time distribution.

Backbone	Time distribution	V.U.N. ↑	Δ V.U.N.	Ratio ↓	Δ Ratio
GT	Standard	$99.5_{\pm 1.0}$	–	$1.5_{\pm 0.2}$	–
GT	$t = u^2$	$78.0_{\pm 3.3}$	-21.5	$2.3_{\pm 1.1}$	+0.8
SGen	Standard	$99.5_{\pm 1.0}$	–	$1.5_{\pm 0.5}$	–
SGen	$t = u^2$	$81.5_{\pm 3.0}$	-18.0	$1.5_{\pm 0.3}$	0.0

G.8 Large-Graph Protocol, Sources, and Failure Statuses

Table 7 retains the five metrics and baselines of AutoGraph (Chen et al., 2025), Table 2, adding Proteins components from SparseDiff (Qin et al., 2023), Appendix C.3, Table 7. SparseDiff uses 10^{-3} units for Degree/Spectrum and 10^{-2} for Clustering/Orbit; we convert means and standard deviations to absolute MMDs. Its five samplings are not five independent training seeds. These literature values are not new hardware-matched reruns.

Proteins Ratio averages normalized Degree, Clustering, Orbit, and Spectrum MMDs, as specified in SparseDiff's Table 1. Its three-descriptor Ratio applies to Planar/SBM (Table 2). We retain its Proteins Ratio 3.6 ± 1.1, without reconstructing it from rounded component means. Point Clouds excludes Degree because its train/test reference MMD is zero. SGen uses AutoGraph splits, preprocessing, evaluator, and Ratio definition with five independent seeds. All eight component MMDs give mean and population standard deviation; Ratio statistics use the five per-seed Ratios, not mean MMDs.

SparseDiff's Point Clouds OOM/OOT entry distinguishes the transferred default configuration (OOM) from a reduced-query run (OOT), without implying failure for every configuration. GEEL retains source dashes for unavailable Point Clouds values. A separate dense GenGNN run reported OOM on Point Clouds; it does not replace any published baseline row. Device-specific failure thresholds and a block-size timing/memory sweep remain unreported.

Large-graph experiments use DiGress+SGen; SimGFM results on molecules and smaller structural graphs do not establish large-graph SimGFM performance. Under the conditions in Section 3.6, terminal blocking preserves predictions, additive loss, and transition distribution for the same SGen model, not equivalence to a different dense backbone.

Large-graph run settings. Both datasets use DiGress with a 12-layer GNNPlus backbone, hidden width 384, and 20-step RRWP features. Proteins uses training batch size 32, and its evaluated checkpoint has 233,000 training updates. Point Clouds uses batch size 1 with 26 gradient-accumulation steps, giving effective batch size 26. Sampling uses batch size 1 and 1,000 diffusion steps on one RTX 3090 with 24 GB memory. The streaming output head evaluates blocks of 128 rows in FP32.

Remaining implementation checks. No block-size resource sweep or numerical blocked/unblocked logits, loss, and gradient comparison is reported. The mathematical execution conditions in Section 3.6 and the five-seed quality results are separate from these unreported implementation checks.

H Backbone and Framework Details

The standard-benchmark implementation uses LayerNorm pre-normalization, with node:edge:global hidden dimensions in the ratio $1 : 0.25 : 0.5$.

Sections 3.1 and 3.2 summarize the sparse support and layer sequence. We give the full encoder and layer equations below. Appendix B specifies the GatedGCN$^+$ operator.

All layer operations below use the communication support defined in Eq. (1).

H.1 Encoder and Layer Equations

The backbone consists of an encoder, L stacked GNN$^+$ layers, and produces updated node and edge representations. We describe the encoder and layer structure.

Encoder. Node features X_t, edge features E_t, and auxiliary features (e.g., RRWP positional encodings (Ma et al., 2023)) are projected to hidden dimensions through learned embeddings:

$$x_i^{(0)} = \phi_X(X_{t,i}), \quad e_{ij}^{(0)} = \phi_E(E_{t,ij}), \quad u^{(0)} = \phi_u([t, y_t, y_{\text{extra}}]), \tag{17}$$

where $u^{(0)}$ is the initial graph-level global state incorporating the diffusion timestep t and any global conditioning.

GNN$^+$ Layer with Global Conditioning. Each layer ℓ updates node, edge, and global representations through three stages:

Stage 1: Sparse message passing. Following the GNN$^+$ design philosophy (Luo et al., 2024; 2025), each layer ℓ updates node and edge representations via the standard edge-aware message-passing template (Gilmer et al., 2017): for each edge $(i, j) \in \mathcal{E}_t$,

$$m_{ij} = \text{MSG}^\ell\big(x_i,\, x_j,\, e_{ij}\big), \quad \hat{e}_{ij} = \text{UPDATE}_E^\ell\big(e_{ij},\, m_{ij}\big), \tag{18}$$

$$\hat{x}_i = \text{UPDATE}_X^\ell\Big(x_i,\ \text{AGG}^\ell\big(\{m_{ij} : j \in \mathcal{N}(i)\}\big)\Big), \tag{19}$$

where MSG, AGG, and UPDATE can be any edge-conditioned GNN$^+$ operator. We instantiate them with **GatedGCN$^+$** (Bresson & Laurent, 2017; Luo et al., 2025); explicit equations are given in Appendix B.

After activation and dropout, the message-passing outputs are combined with the layer inputs through learnable ReZero residual gates:

$$\tilde{x}_i = x_i + \alpha_{\text{mp}}^x \cdot \hat{x}_i, \tag{20}$$

$$\tilde{e}_{ij} = e_{ij} + \alpha_{\text{mp}}^e \cdot \hat{e}_{ij}, \tag{21}$$

where $\alpha_{\text{mp}}^x, \alpha_{\text{mp}}^e$ are learnable scalar coefficients initialized to zero (Bachlechner et al., 2021). Global conditioning by u is deferred to the FFN stage (Stage 3) and is not applied to the message-passing inputs.

Stage 2: Global state update via pooling. After message passing, the global state is updated using pooled node and edge representations:

$$u^{(\ell+1)} = u^{(\ell)} + \alpha_u \cdot f_u\big(W_u u^{(\ell)} + W_x\, \text{Pool}_X(\tilde{X}) + W_e\, \text{Pool}_E(\tilde{E})\big), \tag{22}$$

where Pool_X and Pool_E are mean pooling operators, f_u is a two-layer MLP, and α_u is a learnable scalar gating coefficient initialized to zero (ReZero (Bachlechner et al., 2021)). This pooling-based update provides a lightweight global information channel: the model aggregates graph-wide statistics without requiring dense pairwise interactions, then broadcasts this information back to local computations in the next stage.

Stage 3: Conditioned FFN. Both node and edge representations pass through feed-forward networks modulated by the updated global state via FiLM (Perez et al., 2018) affine conditioning:

$$\text{Mod}(h, u) = \gamma(u) \odot h + \beta(u), \tag{23}$$

where γ, β are learned linear functions of u. The node and edge updates are then

$$x_i^{(\ell+1)} = \tilde{x}_i + \alpha_x \cdot \text{Mod}\big(\text{FFN}_X(\tilde{x}_i),\ u^{(\ell+1)}\big), \tag{24}$$

$$e_{ij}^{(\ell+1)} = \tilde{e}_{ij} + \alpha_e \cdot \text{Mod}\big(\text{FFN}_E(\tilde{e}_{ij}),\ u^{(\ell+1)}\big), \tag{25}$$

where α_x, α_e are learnable ReZero gating coefficients (initialized to zero). This mechanism allows the diffusion timestep and graph-level information to continuously modulate local feature transformations on both nodes and edges, providing global-to-local conditioning rather than pairwise attention. Local inputs remain node- or edge-specific despite sharing the pooled context.

All five residual connections in a layer (two from message passing, one from the global update, two from the FFN sub-blocks) use learnable scalar gates initialized to zero, following the initialization strategy of Bachlechner et al. (2021). This initialization starts the residual branches at zero and lets their contributions be learned.

Conditioning inputs. The global encoder receives an embedded timestep. In unconditional experiments, y_t is empty. On the standard benchmarks, y_{extra} includes normalized node count and 3–6-node cycle statistics, with normalized molecular weight added for molecules. A noisy QM9-no-H molecule containing C, C, and O, for example, supplies node-count ratio $3/9$ and weight ratio $(12 + 12 + 16)/150 = 40/150$. For this QM9 setting, these features and the timestep embedding produce a 64-dimensional initial global state.

Framework integration. Residual outputs and the DiGress/SimGFM integrations are described in Section 3.4. Algorithm 1 shows how logits enter the original training loss and clean-category probabilities enter the original sampling update.