Generating Directed Graphs with Dual Attention and Asymmetric Encoding

Alba Carballo-Castro[*], **Manuel Madeira, Yiming Qin, Dorina Thanou, Pascal Frossard**
LTS4, EPFL, Lausanne, Switzerland

Abstract

Directed graphs naturally model systems with asymmetric, ordered relationships, essential to applications in biology, transportation, social networks, and visual understanding. Generating such graphs enables tasks such as simulation, data augmentation and novel instance discovery, yet it remains underexplored. A key challenge lies in modeling edge directionality, which greatly enlarges the dependency space and makes the underlying distribution harder to learn. Addressing this requires more expressive models that are sensitive to directional topologies. We propose DIRECTO, the first generative model for directed graphs built on the discrete flow matching framework. Our approach combines: (i) principled positional encodings tailored to asymmetric pairwise relations, (ii) a dual-attention mechanism capturing both incoming and outgoing dependencies, and (iii) a robust, discrete generative framework. Our method performs strongly in diverse settings and even competes with specialized models for particular classes, such as directed acyclic graphs, highlighting the effectiveness and generality of our approach, and establishing a solid foundation for future research in directed graph generation.

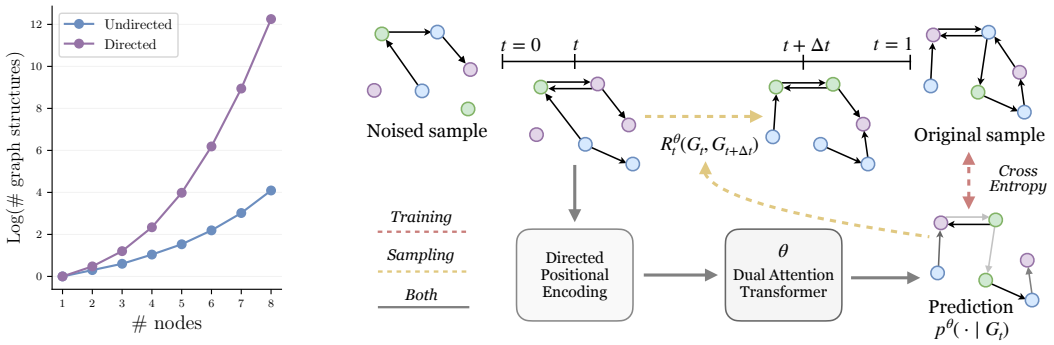
1 Introduction

Directed graphs (digraphs) naturally model systems with asymmetric relationships, capturing essential structures such as flows, dependencies, and hierarchies that arise in many real-world applications. This makes digraphs particularly well-suited for problems in diverse domains including biology [34, 60, 67], transportation [11], social dynamics [55], and, more recently, image and video understanding [9, 54], where structured, directional representations are critical for interpretation and reasoning. Consequently, generating digraphs is central to tasks such as such as simulation, data augmentation and novel instance discovery in domains characterized by directional structure.

Graph generative models offer a data-driven approach to producing plausible and diverse samples, with applications in drug discovery [44, 64], finance [37], social network modeling [61], and medicine [47]. Despite their broad applicability and strong performance, the majority of existing generative models focus on undirected graphs [38, 43, 63, 57, 70, 16, 52]. Digraphs have received comparatively less attention, with recent work proposing auto-regressive models for DAGs [72, 36] and, more recently, for general digraphs [33]. We identify a key factor limiting progress: at the modeling level, the directed setting is more challenging than its undirected counterpart, as edge directionality greatly enlarges the learnable space (see Figure 1a: e.g., for graphs with 4 nodes, there are 218 possible digraphs, compared to only 11 undirected ones). This is further amplified in settings with domain-specific structural constraints, such as *directed acyclic graphs* (DAGs).

[*]Correspondence to alba.carballocastro@epfl.ch

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Graph Machine Learning.



(a) Comparison on # structures.

(b) Overview of DIRECTO.

Figure 1: (a) The learnable space increases drastically with the number of nodes for digraphs compared to undirected graphs [21], highlighting challenges in extending graph generative models to directed structures. (b) Overview of our generative model for directed graphs. During training, the *dual attention transformer* (denoising network) parameterized by θ, and enhanced with *asymmetric positional encodings*, learns to reverse predictions using cross-entropy loss. During inference, we compute the *rate matrix* $R_t^\theta(G_t, G_{t+\Delta t})$ based on the model prediction $p^\theta(\cdot \mid G_t)$, which governs the evolution of the generative process over finite intervals Δt.

To address this, we introduce DIRECTO[2], the first iterative refinement-based generative model for directed graphs. Our approach brings together three core components of expressive graph generative modeling: (i) informative input features, (ii) an expressive neural architecture, and (iii) a robust generative framework. For input features, we find that, upon comprehensive evaluation, positional encodings specifically designed for graphs with an asymmetric adjacency matrix, are more successful than directionality-agnostic baselines. For the architecture, we employ a dual attention mechanism that performs cross-attention between edge features and their reversed counterparts, allowing to capture differently source-to-target and target-to-source information flows. Finally, these components are integrated within a robust discrete-state flow matching framework that enables efficient generation with state-of-the-art performance. Figure 1b gives an overview of our method and its components.

We evaluate the generative performance of DIRECTO on multiple datasets using distributional metrics tailored to the directed setting. Across all cases, it consistently delivers strong performance, underscoring its effectiveness on structurally diverse graphs. Extensive ablations further show that the proposed dual-attention mechanism is critical for modeling directional dependencies, while positional encodings enhance overall generation quality.

Our main contributions are summarized as follows:

(i) We propose DIRECTO, the first flow-based generative model specifically tailored for directed graphs, combining a direction-aware dual attention block with positional encoding tailored to capture directionally asymmetric structural properties.

(ii) We conduct an extensive empirical analysis demonstrating that DIRECTO achieves state-of-the-art performance on structurally diverse synthetic and real-world graphs.

(ii) We perform extensive ablations to highlight the criticality of our proposed components.

2 DIRECTO: Discrete Flow Matching for Directed Graphs

In this section, we introduce DIRECTO, the first flow-based digraph generative model. We begin by outlining the overall generative framework, followed by the two directionality-aware components that form the core of DIRECTO: asymmetric positional encoding and dual attention.

[2]Code available at: https://github.com/acarballocastro/DIRECTO

2.1 Directed Graph Generation via Discrete Flow Matching

Notation We denote by $G = \left(x^{(1:n:N)}, e^{(1:i\neq j:N)}\right)$ directed graphs with N nodes. The set of nodes is denoted by $\{x^{(n)}\}_{n=1}^{N}$, and the set of directed edges by $\{e^{(i,j)}\}_{1\leq i\neq j\leq N}$. Both nodes and edges are categorical variables, where $x^{(n)} \in \{1, \ldots, X\}$ and $e^{(i,j)} \in \{1, \ldots, E\}$.

Problem statement: We build on the Discrete Flow Matching setting (see Appendix I) and extend it to the directed graph setting. Inspired by standard practice in iterative refinement for undirected graphs [63, 70, 57, 52], we assume that every ordered pair of distinct nodes corresponds to a directed edge belonging to one of several possible classes, including a class representing the absence of an edge. We provide the complete training and sampling algorithms in Appendix D.

To address the combinatorial complexity of asymmetric adjacency matrices in digraphs (Figure 1a), we exploit a key strength of DFM: the decoupling of training and sampling. This enables post-training optimizations such as time-adaptive schedules and custom CTMC rate matrices [52], which we extend to directed graphs. While these strategies improve performance, they remain insufficient to capture the unique structural properties of digraphs (see Section 3). We therefore introduce architectural components explicitly tailored for directionality-aware graph generation.

2.2 Asymetric positional encoding

Our generative framework employs a denoising GNN, which struggles to capture global patterns due to message-passing locality [68, 45]. To address this, we augment it with positional encodings (PEs) that provide structural information beyond local neighborhoods [4, 6]. While common PEs like Laplacian eigenvectors or shortest-path distances work well for undirected graphs [63, 52], they ignore directionality. To address this limitation, we adopt recent direction-aware PEs [19, 25], appending them to node and edge features to capture the asymmetric dependencies of directed graphs.

Magnetic Laplacian To retain directional information while preserving desirable spectral properties, the authors in [19] propose the *Magnetic Laplacian* (MagLap), which introduces a complex-valued phase encoding into the adjacency matrix. It uses the symmetrized adjacency matrix $\boldsymbol{A}_s$ and the symmetrized degree matrix $\boldsymbol{D}_s$ and is given by:

$$\boldsymbol{L}_{\text{dir}}^{(q)} = \boldsymbol{D}_s - (\boldsymbol{A}_s \odot \exp(i\Theta^{(q)})) \tag{1}$$

where $\odot$ denotes the element-wise (Hadamard) product, i is the imaginary unit, and the phase matrix $\Theta^{(q)}$ is defined as $\Theta_{u,v}^{(q)} = 2\pi q(\boldsymbol{A}_{u,v} - \boldsymbol{A}_{v,u})$. The parameter $q \geq 0$ controls the strength of the phase shift (resulting in the classical combinatorial Laplacian when $q = 0$). We leverage the resulting complex-valued eigenvectors by concatenating the real and complex parts of the eigenvectors to the node features and the eigenvalues to the global graph features, thus injecting direction-aware structural signals at both levels of representation.

Multi-q Magnetic Laplacian Recent work by [25] leverages the Magnetic Laplacian with Q distinct complex potentials $q_1, \ldots, q_Q$, to recover a more informative bidirectional walk profile that better captures asymmetries in connectivity. Similarly to the vanilla method, we incorporate these by concatenating the first k eigenvalues (real and imaginary parts) of each Laplacian to the global features, and the k eigenvectors to the node features.

Directed Relative Random Walk Probabilities (RRWP) In [19], the authors define the directed transition matrix $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{D}_{\text{out}}^{-1}$, based on the out-degree matrix $\boldsymbol{D}_{\text{out}}^{-1}$, and propose modeling reverse random walks with $\boldsymbol{R} = \boldsymbol{A}^{\top}\boldsymbol{D}_{\text{in}}^{-1}$, where $\boldsymbol{D}_{\text{in}}$ is the in-degree matrix. To ensure valid probabilities, we follow their preprocessing step of adding self-loops to sink nodes. The resulting positional encoding concatenates forward and reverse k-step transition probabilities:

$$\text{RRWP}(G) = [\boldsymbol{I}, \boldsymbol{T}, \boldsymbol{T}^2, \ldots, \boldsymbol{T}^{K-1}, \boldsymbol{I}, \boldsymbol{R}, \boldsymbol{R}^2, \ldots, \boldsymbol{R}^{K-1}], \tag{2}$$

where off-diagonal entries enrich edge features and diagonal terms augment node features. To counter the concentration of random walks on sinks at large k, we also incorporate *Personalized PageRank (PPR)* features, given by $\text{PPR} = p_r(\boldsymbol{I} - (1 - p_r)\boldsymbol{T})^{-1}$ with restart probability p_r.

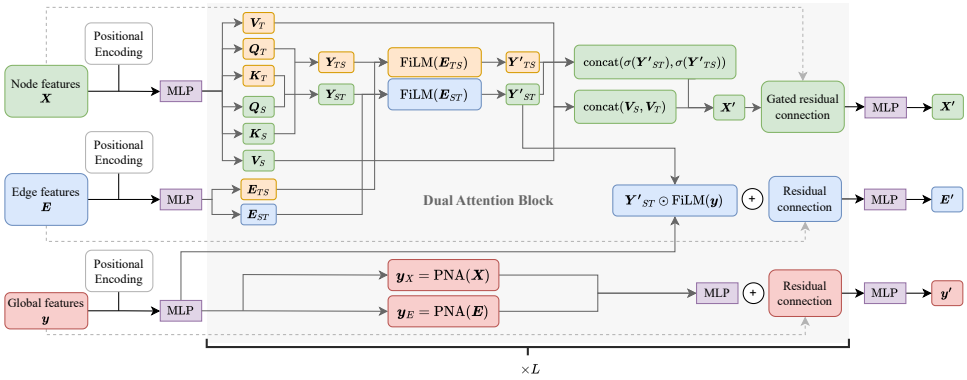


Figure 2: Network architecture of DIRECTO. We stack L dual attention layers that account for both source-to-target and target-to-source information via cross-attention mechanisms. $\boldsymbol{X}$, $\boldsymbol{E}$ and $\boldsymbol{y}$ denote the stacked input node, edge, and global features. $\boldsymbol{X}'$, $\boldsymbol{E}'$ and $\boldsymbol{y}'$ are the output of the model, i.e., predicted clean node and edge distribution, and graph feature. FiLM [50] and PNA [12] pooling layers are incorporated to enable flexible modulation between node, edge, and graph-level features.

2.3 Graph transformer with dual attention

Graph Transformers [15, 53] effectively model node interactions by leveraging the adjacency matrix. In directed graphs, this matrix encodes only source-to-target flow, neglecting reverse influences. To capture bidirectional dependencies and integrate structural signals across node $\boldsymbol{X}$, edge $\boldsymbol{E}$, and global $\boldsymbol{y}$ features, we design a Transformer with L dual-attention layers (Figure 2). This block introduces (i) an attention-based scheme that models both forward and reverse information flow, and (ii) modulation layers that fuse signals across different levels of granularity, ensuring direction-aware generation.

Bidirectional information flow via dual attention aggregation To model directional dependencies, we use a cross-attention mechanism between source-to-target edge features $\boldsymbol{E}_{\text{ST}}$ and their reversed counterparts $\boldsymbol{E}_{\text{TS}}$, which enables the model to reason bidirectionally across edges. By explicitly attending to both directions, our architecture better captures the complex, reciprocal relationships inherent in directed graphs, which builds on ideas from [65]. Concretely, we compute two directional attention maps between role-specific node projections:

$$\boldsymbol{Y}_{\text{ST}}[i,j] = \frac{\boldsymbol{Q}_{\text{S}}[i] \cdot \boldsymbol{K}_{\text{T}}[j]}{\sqrt{d_q}}, \quad \boldsymbol{Y}_{\text{TS}}[i,j] = \frac{\boldsymbol{Q}_{\text{T}}[i] \cdot \boldsymbol{K}_{\text{S}}[j]}{\sqrt{d_q}}, \tag{3}$$

where $\boldsymbol{Q}_{\text{S}}$, $\boldsymbol{Q}_{\text{T}}$, $\boldsymbol{K}_{\text{S}}$, and $\boldsymbol{K}_{\text{T}}$ are the role-specific projections for source and target nodes, respectively, and d_q is the query feature dimension. These attention weights are modulated using edge features through a *Feature-wise Linear Modulation* (FiLM) [50] layer: given the edge features $\boldsymbol{E}$, the attention matrix $\boldsymbol{E}_{\text{attn}}$, and the learnable weights $\boldsymbol{W}^1_{\text{FiLM}}$, $\boldsymbol{W}^2_{\text{FiLM}}$, it computes

$$\text{FiLM}(\boldsymbol{E}, \boldsymbol{E}_{\text{attn}}) = \boldsymbol{E}\boldsymbol{W}^1_{\text{FiLM}} + (\boldsymbol{E}\boldsymbol{W}^2_{\text{FiLM}}) \odot \boldsymbol{E}_{\text{attn}} + \boldsymbol{E}_{\text{attn}}, \tag{4}$$

where $\odot$ denotes element-wise multiplication. We produce updated attention maps $\boldsymbol{Y}'_{\text{ST}}$ and $\boldsymbol{Y}'_{\text{TS}}$ and use $\boldsymbol{Y}'_{\text{ST}}$ to update the edge features, combining local edge attributes with global graph information.

To consolidate directional information, we introduce an *attention aggregation mechanism*. Instead of treating the attentions from the two directions independently, we concatenate the modulated attention maps and apply a single softmax operation to obtain unified attention weights:

$$\boldsymbol{A}_{\text{aggr}} = \text{softmax}(\text{concat}(\boldsymbol{Y}'_{\text{ST}}, \boldsymbol{Y}'_{\text{TS}})) \in \mathbb{R}^{n \times 2n}, \quad \boldsymbol{V}^{\top}_{\text{aggr}} = \text{concat}(\boldsymbol{V}^{\top}_{\text{S}}, \boldsymbol{V}^{\top}_{\text{T}}) \in \mathbb{R}^{2n \times h}, \tag{5}$$

where h is the hidden dimension and the concatenation is performed along the node dimensions. These unified weights are used to aggregate the value vectors $\boldsymbol{V}_{\text{S}}$ and $\boldsymbol{V}_{\text{T}}$ through weighted summation:

$$\boldsymbol{X}' = \boldsymbol{A}_{\text{aggr}}\boldsymbol{V}_{\text{aggr}} \in \mathbb{R}^{n \times h}. \tag{6}$$

By applying a joint softmax, the model is able to assign asymmetric importance to the source-to-target and target-to-source directions. The node features are updated using a gated residual connection, which combines the original and updated features by learning a gate that controls how much of the new information should be integrated. For full technical details see Appendix A.

Table 1: Directed graph generation performance for different configurations of DIRECTO. Results are the mean ± standard deviation across five sampling runs. We considered MagLap with $Q = 10$ for the synthetic datasets and $Q = 5$ for the real-world ones due to the computational complexity in increasing Q. OOT indicates that the model could not be run within a reasonable timeframe.

	ER-DAG		SBM		TPU Tiles		Visual Genome	
Model	**Ratio** ↓	**V.U.N.** ↑	**Ratio** ↓	**V.U.N.** ↑	**Ratio** ↓	**V.U.N.** ↑	**Ratio** ↓	**V.U.N.** ↑
Training set	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
MLE	15.1 ± 0.2	0.0 ± 0.0	11.6 ± 0.2	0.0 ± 0.0	149.8 ± 0.7	24.7 ± 0.0	17.0 ± 0.6	0 ± 0.0
D-VAE	106.6 ± 5.4	0.0 ± 0.0	-	-	OOT	OOT	-	-
LayerDAG	4.2 ± 3.2	21.5 ± 2.7	-	-	413.6 ± 70.1	**98.5** ± 3.0	-	-
DiGress	1.9 ± 0.3	34.0 ± 4.1	3.9 ± 0.9	41.5 ± 5.1	57.5 ± 1.7	70.9 ± 3.4	17.0 ± 0.6	0.3 ± 0.6
DeFoG	**1.3** ± 0.1	67.5 ± 1.6	21.4 ± 1.6	13.5 ± 5.4	63.7 ± 2.6	72.0 ± 2.4	10.6 ± 0.8	39.6 ± 2.8
DIRECTO-DD RRWP	1.4 ± 0.3	79.0 ± 3.7	1.7 ± 0.4	81.5 ± 3.2	61.0 ± 2.9	76.8 ± 1.9	15.3 ± 0.8	72.7 ± 3.9
DIRECTO-DD MagLap	1.5 ± 0.2	85.0 ± 9.2	1.5 ± 0.4	95.5 ± 3.7	64.3 ± 5.3	77.0 ± 7.0	7.6 ± 0.7	61.9 ± 4.4
DIRECTO RRWP	1.7 ± 0.1	**94.0** ± 1.0	**1.4** ± 0.2	87.0 ± 5.1	75.4 ± 8.1	77.0 ± 2.9	12.8 ± 0.6	**83.8** ± 4.3
DIRECTO MagLap	**1.3** ± 0.2	92.0 ± 3.7	2.0 ± 0.3	**96.5** ± 2.5	**44.0** ± 7.1	80.5 ± 4.6	**6.2** ± 0.5	67.0 ± 4.3

Multi-scale information modulation for graph denoising Predicting clean node and edge types from noisy inputs requires combining local interactions with global structure. Following common practices in standard graph generation architectures [63, 57, 52], we incorporated FiLM layers (see Equation 4) to integrate global graph-level signals into the construction of edge representations. To complement this, *Principal Neighbourhood Aggregation* (PNA) [12] layers aggregate multi-scale neighborhood information via pooling operations. Given the node features $\boldsymbol{X} \in \mathbb{R}^{n \times h}$ and a learnable weight matrix $\boldsymbol{W}_{\text{PNA}}$, PNA computes:

$$\text{PNA}(\boldsymbol{X}) = \text{concat}\left(\texttt{max}(\boldsymbol{X}), \texttt{min}(\boldsymbol{X}), \texttt{mean}(\boldsymbol{X}), \texttt{std}(\boldsymbol{X})\right) \boldsymbol{W}_{\text{PNA}} \in \mathbb{R}^{4h}, \tag{7}$$

with concatenation performed along the feature dimension. We use PNA layers to update global graph representations based on node-level information at each attention pass. Together, these components enable expressive and scalable integration of local and global features, allowing the model to capture higher-order structure critical for accurate graph denoising.

3 Experiments

In this section, we first evaluate the flexibility of our method to generate directed graphs. Then, we analyze the impact of our architectural improvements on generative performance.

Datasets We sample digraphs from four distributions: Erdős–Rényi (binomial model) both general and DAGs variants; Price's model, a directed analogue of the Barabási–Albert model that produces DAGs; and a directed version of the stochastic block model (SBM) dataset [43]. For real-world datasets, we employ the TPU Tiles dataset [51] and the Visual Genome dataset [32], two widely adopted benchmarks for Neural Architecture Search and Scene Graph understanding, respectively. Further details are available in Appendix E.

Metrics We extend evaluation protocols from undirected [43, 5] and directed [33] graph generation to the digraph setting. We can divide our metrics in two groups: structural distributional alignment (Ratio) and Validity, Uniqueness, and Novelty (V.U.N.). Distributional alignment is measured via Maximum Mean Discrepancy (MMD) over degree distributions, clustering, spectral, and wavelet features (using the directed Laplacian [10]). Results are reported as ratios of generated to training statistics, averaged for comparability.

For synthetic datasets, *validity* is assessed via statistical tests of adherence to the generative distribution and, for DAG datasets, by ensuring acyclicity. On the TPU Tiles dataset, we measure acyclicity, while for Visual Genome we check typed structural constraints (e.g., edges go from objects to attributes and relationships, and from relationships to objects). To measure generative diversity while avoiding memorization, we compute *uniqueness* (fraction of non-isomorphic generated graphs) and *novelty* (fraction of generated graphs not in the training set). The V.U.N. ratio reports the proportion of samples that are simultaneously valid, unique, and novel. Further details as well as dataset statistics can be found in Appendix F.

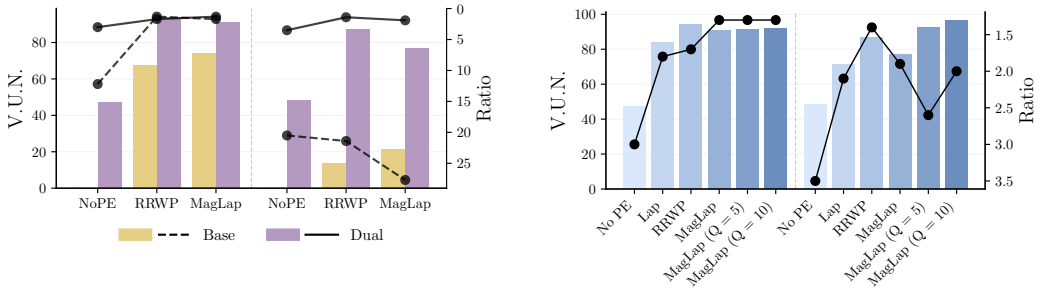


Figure 3: Ablation results for dual attention (left) and positional encodings (right). Each plot shows results on the ER-DAG (left bars/lines) and SBM (right bars/lines) datasets. Better performance corresponds to V.U.N. bars and Ratio lines appearing closer to the top of each subplot.

Baselines In the general directed setting, we compare DIRECTO to a Maximum Likelihood Estimation (MLE) baseline over node count, node types, and edge types. For DAGs, we also include the two publicly available baselines: D-VAE[72] and LAYERDAG[36]. See Appendix G.1 for details. The architectural improvements in Secs. 2.2 and 2.3 are agnostic to the underlying refinement framework. To illustrate this, we extend DIRECTO to a discrete diffusion backbone (DIRECTO-DD), described in Appendix I.3. We further include DEFOG[52] and DIGRESS[63], adapting them to directed graphs by removing edge symmetrization. For each experiment, we highlight the **best result** and second-best method. Further details on the experimental setup can be found in Appendix G.

3.1 Directed Graph Generation Performance

Table 1 reports the results on ER-DAG, SBM and the real-world datasets. DIRECTO consistently achieves superior performance in generating directed graphs across both synthetic and real-world datasets. In synthetic experiments on ER-DAG and SBM, DIRECTO variants demonstrated a strong ability to capture the target distributions, as evidenced by their low MMD Ratio scores and high V.U.N.. Notably, while LAYERDAG enforces acyclicity by design, it fails to capture the ER distribution, as evidenced by its low V.U.N. score (21.5%; see Table 14 for details). Similarly, on the real-world datasets TPU Tiles and Visual Genome, DIRECTO proved its versatility by generating graphs with more realistic structural properties and achieving the highest V.U.N. scores among most baselines, indicating a superior ability to produce diverse and structurally faithful graphs. The exception is LayerDAG, which in the case of TPU Tiles benefits from enforcing acyclicity, the only validity constraint evaluated in this case. Detailed results are reported in Appendix H.

3.2 Dual Attention Ablation

We now analyze the influence of the dual attention mechanism on the performance of DIRECTO. In Figure 3, we compare the V.U.N. metric and the Ratio score across both synthetic datasets, highlighting the impact of dual attention on generation quality and graph realism. We observe that the dual attention mechanism consistently improves generative performance, regardless of the positional encoding employed. Notably, even in the absence of any positional encoding ("No PE"), dual attention still achieves non-zero V.U.N., highlighting its capacity to independently capture directionality-relevant information. Appendix H.3 includes the full ablation tables, both for DIRECTO and DIRECTO-DD.

3.3 Positional Encodings Ablation

We evaluate the sensitivity of DIRECTO to different positional encodings, including a direction-agnostic baseline ("Lap"). In Figure 3, we observe consistent trends across datasets: integrating positional encodings improves V.U.N. and Ratio. Moreover, direction-aware encodings outperform agnostic ones, supporting our modeling hypothesis. Among the encodings tested, Directed RRWP achieves the best overall performance on ER-DAG, and MultiMagLapPE with $Q = 10$ on SBM. Nevertheless, we remark that RRWP demonstrates clear superiority in both scalability and runtime efficiency (see Appendix G.3), and that dual attention still proves to be the key component (see Figure 3). The complete tables for both DIRECTO and DIRECTO-DD can be found in Appendix H.4.

4 Related work

Graph generative methods Early graph generative models include *auto-regressive* methods [71, 38], which sequentially grow graphs but require node ordering. *One-shot* models such as VAEs [30, 56, 26, 42, 62], GANs [13, 31, 43], and normalizing flows [29, 66, 39, 41] generate full graphs in a single pass, avoiding predefined node order.Graph diffusion models extend this ideas, with initial approaches consisting of adaptations of continuous state-space discrete-time diffusion frameworks [58, 23, 59]. These include works such as EDP-GNN [48], DGSM [40], and GeoDiff [69]. Later, the discrete state-space framework [3] was adopted by DiGress [63] and MCD [20]. Methods that operate in continuous time [7] were also introduced, including GDSS [28] and GruM [27] in continuous state-spaces and DisCo [70] and Cometh [57] for discrete state-spaces. Recent approaches combine the sequential modeling of auto-regressive methods with the global denoising of diffusion [73]. Finally, DeFoG [52] achieves state-of-the-art performance by leveraging discrete flow matching [8, 18].

Directed graph generation Works in this area focus solely on DAG generation, using autoencoders [72], autoregressive models [35], or DAG-specific diffusion [36], all requiring topological ordering. Other approaches [2, 1] use ideas from discrete diffusion but are restricted to Neural Architecture search and, therefore, to DAGs. Finally, the work in [33] is the first to propose a general directed graph generation method using an auto-regressive approach. Unfortunately, to the best of our knowledge, its implementation is not publicly available, limiting reproducibility and evaluation.

5 Conclusion

We propose DIRECTO, a novel discrete flow matching-based method for effective directed graph generation. We combine directionality-aware positional encodings with a dual-attention mechanism that explicitly handles source-to-target and target-to-source dependencies. Empirical results demonstrate that our model significantly outperforms existing baselines, successfully learning to generate directed structures and preserve critical properties, such as acyclicity, without the need for explicit constraints. Looking forward, scalability, a deeper understanding of the role of permutation equivariance, and further guidance of the generative process remain promising directions of research. We refer the reader to Appendix K for a detailed discussion of the broader impact and limitations of this work.

References

[1] Sohyun An, Hayeon Lee, Jaehyeong Jo, Seanie Lee, and Sung Ju Hwang. DiffusionNAG: predictor-guided neural architecture generation with diffusion models. *International Conference on Learning Representations*, 2024. 7

[2] Rohan Asthana, Joschua Conrad, Youssef Dawoud, Maurits Ortmanns, and Vasileios Belagiannis. Multi-conditioned graph diffusion for neural architecture search. *Transactions on Machine Learning Research*, 2024. 7

[3] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 7, 33

[4] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. *International Conference on Machine Learning*, 139:748–758, 2021. 3

[5] Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. *International Conference on Learning Representations*, 2024. 5

[6] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 3

[7] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022. 7

[8] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *International Conference on Machine Learning*, 213:5453 – 5512, 2024. 7, 16, 32

[9] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, 2023. 1

[10] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005. 5, 21

[11] Anna Concas, Caterina Fenu, Lothar Reichel, Giuseppe Rodriguez, and Yunzi Zhang. Chained structure of directed graphs with applications to social and transportation networks. *Applied Network Science*, 7(1), 2022. 1

[12] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020. 4, 5, 14

[13] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018. 7

[14] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149(3683):510–515, 1965. 18

[15] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021. 4, 32

[16] Floor Eijkelboom, Grigory Bartosh, Christian Andersson 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. 1

[17] Paul Erdös and Alfred Rényi. On random graphs I. *Publ. math. debrecen*, 6(290-297):18, 1959. 18

[18] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024. 7, 32

[19] Simon Geisler, Yujia Li, Daniel J Mankowitz, Ali Taylan Cemgil, Stephan Günnemann, and Cosmin Paduraru. Transformers meet directed graphs. *International Conference on Machine Learning*, 447:11144–11172, 2023. 3, 15

[20] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanael Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *NeurIPS Workshop on New Frontiers in Graph Learning*, 2022. 7

[21] Frank Harary and Edgar M. Palmer. *Graphical Enumeration*. Academic Press, 1973. 2

[22] Leonhard Held and Daniel Sabanés Bové. *Applied Statistical Inference: Likelihood and Bayes*. Springer Publishing Company, Incorporated, 2013. 20

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 7

[24] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. 18

[25] Yinan Huang, Haoyu Wang, and Pan Li. What are good positional encodings for directed graphs? *International Conference on Learning Representations*, 2025. 3, 15

[26] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *International Conference on Machine Learning*, pages 2323–2332, 2018. 7

[27] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *International Conference on Machine Learning*, 900:22371 – 22405, 2024. 7

[28] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. *International Conference on Machine Learning*, pages 10362–10383, 2022. 7

[29] Madhawa Kaushalya, Ishiguro Katushiko, Nakago Kosuke, and Abe Motoki. GraphNVP: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019. 7

[30] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016. 7

[31] Igor Krawczuk, Pedro Abranches, Andreas Loukas, and Volkan Cevher. GG-GAN: A geometric graph generative adversarial network. *OpenReview*, 2021. 7

[32] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2017. 5, 19

[33] Marc T. Law, Karsten Kreis, and Haggai Maron. Directed graph generation with heat kernels. *Transactions on Machine Learning Research*, 2025. 1, 5, 7, 21

[34] Chun Li, Nannan Tang, and Jun Wang. Directed graphs of dna sequences and their numerical characterization. *Journal of Theoretical Biology*, 241(2):173–177, 2006. 1

[35] Muchen Li, Jeffrey Yunfan Liu, Leonid Sigal, and Renjie Liao. GraphPNAS: learning distribution of good neural architectures via deep graph generative models. *arXiv preprint arXiv:2211.15155*, 2022. 7

[36] Mufei Li, Viraj Shitole, Eli Chien, Changhai Man, Zhaodong Wang, Srinivas Sridharan, Ying Zhang, Tushar Krishna, and Pan Li. LayerDAG: A layerwise autoregressive diffusion model for directed acyclic graph generation. *International Conference on Learning Representations*, 2025. 1, 6, 7, 18, 22

[37] Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. Diga: Guided diffusion model for graph recovery in anti-money laundering. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 4404–4413, 2023. 1

[38] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in Neural Information Processing Systems*, 383:4255 – 4265, 2019. 1, 7

[39] Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. *International Conference on Learning Representations*, 2021. 7

[40] Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. *Advances in Neural Information Processing Systems*, 34:19784–19795, 2021. 7

[41] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. *International Conference on Machine Learning*, 139:7192–7203, 2021. 7

[42] Changsheng Ma and Xiangliang Zhang. GF-VAE: A flow-based variational autoencoder for molecule generation. *ACM International Conference on Information & Knowledge Management*, page 1181–1190, 2021. 7

[43] Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. *International Conference on Machine Learning*, 162:15159–15179, 2022. 1, 5, 7, 21

[44] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph Networks for Molecular Design. *Machine Learning: Science and Technology*, 2020. 1

[45] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *AAAI Conference on Artificial Intelligence*, 2019. 3

[46] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *International Conference on Machine Learning*, 574:6840 – 6851, 2021. 33

[47] Giannis Nikolentzos, Michalis Vazirgiannis, Christos Xypolopoulos, Markus Lingman, and Erik G. Brandt. Synthetic electronic health records generated with variational graph autoencoders. *Digital Medicine*, 6(1):83, 2023. 1

[48] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484, 2020. 7

[49] Tiago P. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1), 2014. 20

[50] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *AAAI Conference on Artificial Intelligence*, 32-1:3942 – 3951, 2018. 4

[51] Phitchaya Mangpo Phothilimthana, Sami Abu-El-Haija, Kaidi Cao, Bahare Fatemi, Michael Burrows, Charith Mendis, and Bryan Perozzi. TpuGraphs: A performance prediction dataset on large tensor computational graphs. *Advances on Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2023. 5, 18

[52] Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. *International Conference on Machine Learning*, 2025. 1, 3, 5, 6, 7, 16, 23, 29, 31, 32

[53] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022. 4

[54] Ivan Rodin, Antonino Furnari, Kyle Min, Subarna Tripathi, and Giovanni Maria Farinella. Action scene graphs for long-form understanding of egocentric videos. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 18622–18632, 2023. 1

[55] Christoph Schweimer, Christine Gfrerer, Florian Lugstein, David Pape, Jan A Velimsky, Robert Elsässer, and Bernhard C Geiger. Generating simple directed social network graphs for information spreading. *ACM Web Conference 2022*, pages 1475–1485, 2022. 1

[56] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018. 7

[57] Antoine Siraudin, Fragkiskos D Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. *arXiv preprint arXiv:2406.06449*, 2024. 1, 3, 5, 7, 31

[58] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*, 37:2256–2265, 2015. 7

[59] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *International Conference on Learning Representations*, 2021. 7

[60] Martha Takane, Saúl Bernal-González, Jesús Mauro-Moreno, Gustavo García-López, Bruno Méndez-Ambrosio, and Francisco F De-Miguel. Directed graph theory for the analysis of biological regulatory networks. *bioRxiv preprint bioRxiv:2023.10.02.560622*, 2023. 1

[61] Jui-Yi Tsai, Ya-Wen Teng, Ho Chiok Yew, De-Nian Yang, and Lydia Y. Chen. CDGraph: Dual conditional social graph synthesizing via diffusion model. *arXiv preprint arXiv:2311.01729*, 2023. 1

[62] Clement Vignac and Pascal Frossard. Top-N: Equivariant set and graph generation without exchangeability. *International Conference on Learning Representations*, 2022. 7

[63] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. *International Conference on Learning Representations*, 2023. 1, 3, 5, 6, 7, 22, 31, 33

[64] Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. MiDi: Mixed graph and 3d denoising diffusion for molecule generation. *European Conference on Machine Learning*, pages 560 – 576, 2023. 1

[65] Qitong Wang, Georgios Kollias, Vasileios Kalantzis, Naoki Abe, and Mohammed J Zaki. Directed graph transformers. *Transactions on Machine Learning Research*, 2024. 4

[66] Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. *International Conference on Artificial Intelligence and Statistics*, 130:37–45, 2021. 7

[67] Pi-Jing Wei, Ziqiang Guo, Zhen Gao, Zheng Ding, Rui-Fen Cao, Yansen Su, and Chun-Hou Zheng. Inference of gene regulatory networks based on directed graph convolutional networks. *Briefings in Bioinformatics*, 25(4), 2024. 1

[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations*, 2019. 3

[69] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *International Conference on Learning Representations*, 2022. 7

[70] 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 on Neural Information Processing Systems*, 2531:79704 – 79740, 2024. 1, 3, 7, 31

[71] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. *International Conference on Machine Learning*, 80:5694–5703, 2018. 7

[72] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-VAE: A variational autoencoder for directed acyclic graphs. *Advances in Neural Information Processing Systems*, 142:1588–1600, 2019. 1, 6, 7, 22

[73] Lingxiao Zhao, Xueying Ding, and Leman Akoglu. Pard: Permutation-invariant autoregressive diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:7156–7184, 2024. 7

Appendix

Table of Contents

A Dual Attention **13**

B Positional Encodings **14**

C Sampling Optimization in DIRECTO **16**

D DIRECTO Training and Sampling Algorithms **17**

E Dataset descriptions **17**
E.1 Synthetic datasets 18
E.2 TPU Tiles 18
E.3 Visual Genome 19

F Further details on evaluation metrics **19**
F.1 Validity metrics 20
F.2 Uniqueness and novelty 21
F.3 Maximum Mean Discrepancy metrics 21

G Experimental details **22**
G.1 Details on baselines 22
G.2 Training setup 23
G.3 Resources and runtime 24

H Additional results **24**
H.1 Synthetic datasets 25
H.2 Real-world datasets 25
H.3 The role of dual attention 26
H.4 The role of positional encodings 26
H.5 ER vs DAG performance 28
H.6 Scalability experiments 28
H.7 Impact of sampling optimization using discrete flow matching 30

I Iterative Refinement Methods for Graph Generation **31**
I.1 Graph iterative refinement methods 31
I.2 Discrete flow matching for graph generation 32
I.3 Discrete diffusion for graph generation 33

J Visualizations **34**

K Impact statement and limitations **40**

A Dual Attention

In this section, we provide a detailed description of the dual attention mechanism used in our model, which effectively integrates both target-to-source and source-to-target information to enhance the graph generation process.

We start with the stacked tensors of node $\boldsymbol{X} \in \mathbb{R}^{B\times N\times d_x}$, edge $\boldsymbol{E} \in \mathbb{R}^{B\times N\times N\times d_e}$ and global $\boldsymbol{y} \in \mathbb{R}^{B\times d_y}$ features, where B is the batch size, N is the number of nodes of the biggest graph in the batch, and d_x, d_e, and d_y are the node, edge, and global feature dimensions respectively. To handle different node sizes in a batch, we use a binary node mask $\boldsymbol{M} \in \{0,1\}^{B\times N\times 1}$ that accounts for the presence of nodes in each graph.

First, we begin by projecting the queries, keys, and values for the source and the target directions. These are then projected and reshaped into n_h heads of dimension $d_q = d_x/n_h$, thus obtaining for each node:

$$\boldsymbol{Q}_{\text{S}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{Q}}^{\text{S}}\boldsymbol{X}), \quad \boldsymbol{K}_{\text{S}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{K}}^{\text{S}}\boldsymbol{X}), \quad \boldsymbol{V}_{\text{S}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{V}}^{\text{S}}\boldsymbol{X}), \tag{8}$$

$$\boldsymbol{Q}_{\text{T}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{Q}}^{\text{T}}\boldsymbol{X}), \quad \boldsymbol{K}_{\text{T}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{K}}^{\text{T}}\boldsymbol{X}), \quad \boldsymbol{V}_{\text{T}} = \text{reshape}(\boldsymbol{M} \odot \boldsymbol{W}_{\text{V}}^{\text{T}}\boldsymbol{X}), \tag{9}$$

where $\boldsymbol{W}_{\text{Q}}^{\text{S}}, \boldsymbol{W}_{\text{K}}^{\text{S}}, \boldsymbol{W}_{\text{V}}^{\text{S}}, \boldsymbol{W}_{\text{Q}}^{\text{T}}, \boldsymbol{W}_{\text{K}}^{\text{T}}, \boldsymbol{W}_{\text{V}}^{\text{T}}$ are the learned linear projections. The final shape of each of the elements after the reshaping is $B \times N \times n_h \times d_q$.

Then we compute the dual attention, both from source to target and from target to source:

$$\boldsymbol{Y}_{\text{ST}}[i,j] = \frac{\boldsymbol{Q}_{\text{S}}[i] \cdot \boldsymbol{K}_{\text{T}}[j]}{\sqrt{d_q}}, \quad \boldsymbol{Y}_{\text{TS}}[i,j] = \frac{\boldsymbol{Q}_{\text{T}}[i] \cdot \boldsymbol{K}_{\text{S}}[j]}{\sqrt{d_q}}. \tag{10}$$

with i, j the index nodes and each $\boldsymbol{Y} \in \mathbb{R}^{B\times N\times N\times n_h\times d_q}$.

To incorporate edge features into the attention mechanism, we apply FiLM-style modulation using additive and multiplicative projections $\boldsymbol{W}_{\text{mul}}, \boldsymbol{W}_{\text{add}}$:

$$\boldsymbol{E}_{\text{mul}}^{\text{S}} = \text{reshape}(\boldsymbol{W}_{\text{mul}}^{\text{S}}(\boldsymbol{E})), \qquad \boldsymbol{E}_{\text{add}}^{\text{S}} = \text{reshape}(\boldsymbol{W}_{\text{add}}^{\text{S}}(\boldsymbol{E})), \tag{11}$$

$$\boldsymbol{E}_{\text{mul}}^{\text{T}} = \text{reshape}(\boldsymbol{W}_{\text{mul}}^{\text{T}}(\boldsymbol{E}^{\top})), \qquad \boldsymbol{E}_{\text{add}}^{\text{T}} = \text{reshape}(\boldsymbol{W}_{\text{add}}^{\text{T}}(\boldsymbol{E}^{\top})), \tag{12}$$

where we reshape the edge features to $\boldsymbol{E} \in \mathbb{R}^{B\times N\times N\times n_h\times d_q}$. This is done so that they can be added or multiplied to the attention scores $\boldsymbol{Y}_{\text{ST}}$ and $\boldsymbol{Y}_{\text{TS}}$ to modify them:

$$\boldsymbol{Y}_{\text{ST}} \leftarrow \boldsymbol{Y}_{\text{ST}} \cdot (\boldsymbol{E}_{\text{mul}}^{\text{S}} + 1) + \boldsymbol{E}_{\text{add}}^{\text{S}}, \tag{13}$$

$$\boldsymbol{Y}_{\text{TS}} \leftarrow \boldsymbol{Y}_{\text{TS}} \cdot (\boldsymbol{E}_{\text{mul}}^{\text{T}} + 1) + \boldsymbol{E}_{\text{add}}^{\text{T}}. \tag{14}$$

Edge Feature Update To get the final update of the edge features, the attention-weighted edge representations are flattened and modulated by global features $\boldsymbol{y}$ using FiLM with additive and multiplicative projections $\boldsymbol{W}_{\text{add}}^{E}$ and $\boldsymbol{W}_{\text{mul}}^{E}$:

$$\boldsymbol{E'} = \boldsymbol{W}_{\text{add}}^{y}(\boldsymbol{y}) + (\boldsymbol{W}_{\text{mul}}^{y}(\boldsymbol{y}) + 1) \odot \text{flatten}(\boldsymbol{Y}_{\text{ST}}), \tag{15}$$

where we flatten over the last dimension. We use $\boldsymbol{Y}_{\text{ST}}$ as it captures the directional flow from source to target, avoiding ambiguity in the directional influence, and still effectively encoding the influence of source nodes on their targets. Then, the result is followed by an output projection and a residual connection

$$\boldsymbol{E'} = \boldsymbol{W}_{\text{out}}^{E}(\boldsymbol{E'}), \quad \boldsymbol{E'} \leftarrow \boldsymbol{E} + \boldsymbol{E'}, \tag{16}$$

resulting in the updated $\boldsymbol{E'} \in \mathbb{R}^{B\times N\times N\times d_e}$.

Node Feature Update To perform the node feature update, first we concatenate the dual attention scores and apply softmax:

$$\boldsymbol{A}_{\text{aggr}} = \text{softmax}\left(\text{concat}(\boldsymbol{Y}_{\text{ST}}, \boldsymbol{Y}_{\text{TS}})\right) \in \mathbb{R}^{B\times N\times 2N\times n_h\times d_e}, \tag{17}$$

$$\boldsymbol{V}_{\text{aggr}} = \text{concat}(\boldsymbol{V}_{\text{T}}, \boldsymbol{V}_{\text{T}}) \in \mathbb{R}^{B\times 2N\times n_h\times d_x}, \tag{18}$$

with $\text{concat}()$ being the concatenation over the third dimension for $\boldsymbol{A}_{\text{aggr}}$ and over the second dimension for $\boldsymbol{V}_{\text{aggr}}$.

After this, the unified weights are aggregated through weighted summation and the result flattened over the second dimension to produce:

$$\boldsymbol{X}_{\text{aggr}} = \text{attn}_{\text{aggr}} \boldsymbol{V}_{\text{aggr}} \in \mathbb{R}^{B \times N \times n_h \times d_x}. \tag{19}$$

We then FiLM-modulate the attended features with the global vector $\boldsymbol{y}$ with additive and multiplicative projections $\boldsymbol{W}^X_{\text{add}}$ and $\boldsymbol{W}^X_{\text{mul}}$ to produce:

$$\boldsymbol{X}_{\text{mod}} = \boldsymbol{W}^X_{\text{add}}(\boldsymbol{y}) + (\boldsymbol{W}^X_{\text{mul}}(\boldsymbol{y}) + 1) \odot \boldsymbol{X}_{\text{aggr}}, \tag{20}$$

To allow the model to adaptively balance between preserving the original node features and incorporating the updated ones, we introduce a learnable gating mechanism $\boldsymbol{g}X \in [0, 1]$:

$$\boldsymbol{g}_X = \sigma(\boldsymbol{W}^X_{\text{gate}} \odot \text{concat}(\boldsymbol{X}, \boldsymbol{V}_{\text{aggr}})), \tag{21}$$

followed by a gated residual connection:

$$\boldsymbol{X'} = (1 + \boldsymbol{g}_X) \odot \boldsymbol{X} + (1 - \boldsymbol{g}_X) \odot \boldsymbol{X}_{\text{mod}}, \tag{22}$$

where the weights assigned to $\boldsymbol{X}$ and $\boldsymbol{X}_{\text{mod}}$ sum up to 2, ensuring that the overall magnitude remains consistent with the ungated case.

Finally, the result is followed by an output projection:

$$\boldsymbol{X'} = \boldsymbol{W}^X_{\text{out}}(\boldsymbol{X'}), \tag{23}$$

resulting in the updated $\boldsymbol{X'} \in \mathbb{R}^{B \times N \times d_x}$.

Global Features Update The updated global vector is computed by aggregating from the node $\boldsymbol{X}$ and edge $\boldsymbol{E}$ projections:

$$\boldsymbol{y'} = \boldsymbol{y} + W_y(\boldsymbol{y}) + X_{\to y}(\boldsymbol{X}) + E_{\to y}(\boldsymbol{E}), \tag{24}$$

where $X_{\to y}$ and $E_{\to y}$ are PNA layers [12] that given node features $\boldsymbol{X} \in \mathbb{R}^{B \times N \times d_x}$ or edge features $\boldsymbol{E} \in \mathbb{R}^{B \times N \times N \times d_e}$ and a learnable weight matrix $\boldsymbol{W}_{\text{PNA}}$, computes

$$\text{PNA}(\boldsymbol{X}) = \text{concat}\left(\texttt{max}(\boldsymbol{X}), \texttt{min}(\boldsymbol{X}), \texttt{mean}(\boldsymbol{X}), \texttt{std}(\boldsymbol{X})\right) \boldsymbol{W}_{\text{PNA}} \in \mathbb{R}^{4d_x}, \tag{25}$$

$$\text{PNA}(\boldsymbol{E}) = \text{concat}\left(\texttt{max}(\boldsymbol{E}), \texttt{min}(\boldsymbol{E}), \texttt{mean}(\boldsymbol{E}), \texttt{std}(\boldsymbol{E})\right) \boldsymbol{W}_{\text{PNA}} \in \mathbb{R}^{4d_e}, \tag{26}$$

where the different operations are performed over the features and then concatenation is performed along the feature dimension. Both results then pass through a linear layer resulting in $X_{\to y}(\boldsymbol{X}), E_{\to y}(\boldsymbol{E}) \in \mathbb{R}^{d_y}$.

Once again, we use an output projection

$$\boldsymbol{y'} = \boldsymbol{W}^y_{\text{out}}(\boldsymbol{y'}), \tag{27}$$

which results in the updated global features $\boldsymbol{y'} \in \mathbb{R}^{B \times d_y}$.

This dual attention block is applied L times within the graph transformer.

B Positional Encodings

To effectively capture the structural properties of directed graphs, we explore a range of positional encodings (PEs) that can be incorporated into the transformer architecture. These encodings provide nodes with a sense of position and orientation within the digraph, enabling the model to better reason about directionality. Below, we detail the specific PEs evaluated in our work.

Laplacian (Lap) Traditionally in the undirected case, the eigenvectors of the combinatorial Laplacian are widely used to encode graph structure, relying on the spectral decomposition $\boldsymbol{L} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^{-1}$ where $\boldsymbol{L}$ is the combinatorial Laplacian, $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues, and $\boldsymbol{\Gamma}$ is the matrix of eigenvectors. The unnormalized and normalized Laplacians are defined as

$$\boldsymbol{L}_U = \boldsymbol{D} - \boldsymbol{A}_s, \quad \boldsymbol{L}_N = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}_s\boldsymbol{D}^{-1/2}, \tag{28}$$

where $\boldsymbol{A}_s$ is the adjacency matrix (symmetrized when working with digraphs) and $\boldsymbol{D}$ is the degree matrix. However, when dealing with directed graphs, this symmetrization discards directionality information, as it enforces $\boldsymbol{A}_s =^{\top}_s$, but it is necessary to guarantee that $\boldsymbol{L}_U$ and $\boldsymbol{L}_N$ are symmetric and positive semi-definite.

Magnetic Laplacian (MagLap) To retain directional information while preserving desirable spectral properties, authors in [19] propose the *Magnetic Laplacian*, which introduces a complex-valued phase encoding into the adjacency matrix. Specifically, the (unnormalized and normalized) Magnetic Laplacians are given by:

$$\boldsymbol{L}^{(q)}_{\text{dir},U} = \boldsymbol{D} - (\boldsymbol{A}_s \odot \exp(i\Theta^{(q)})), \quad \boldsymbol{L}^{(q)}_{\text{dir},N} = \boldsymbol{I} - \left((\boldsymbol{D}^{-1/2}\boldsymbol{A}_s\boldsymbol{D}^{-1/2}) \odot \exp(i\Theta^{(q)})\right), \quad (29)$$

where $\odot$ denotes the element-wise (Hadamard) product, i is the imaginary unit, and the phase matrix $\Theta^{(q)}$ is defined as:

$$\Theta^{(q)}_{u,v} = 2\pi q(\boldsymbol{A}_{u,v} - \boldsymbol{A}_{v,u}). \quad (30)$$

The potential parameter $q \geq 0$ controls the strength of the phase shift. For $q = 0$, the Magnetic Laplacian reduces to the classical combinatorial Laplacian. The resulting complex-valued eigenvectors can effectively encode directionality patterns in the graph. We leverage the information by concatenating the real and complex parts of the eigenvectors to the node features, and the eigenvalues to the global graph features.

Multi-q Magnetic Laplacian Recent work by [25] extends the Magnetic Laplacian by introducing a multi-q formulation, which considers as positional encoding stacking together the eigenvalues and eigenvectors of Q distinct Magnetic Laplacians with potentials $q_1, \ldots, q_Q$. This approach enables the recovery of the bidirectional walk profile, a generalization of walk counting in undirected graphs that captures a broader range of directional relationships.

By incorporating information from multiple potentials, the multi-q formulation enhances the representational power for directed structures. In particular, they show that the method is robust to the chosen potentials q chosen. In our model, we leverage this information by concatenating the first k eigenvectors (real and imaginary parts) of each of the Q magnetic Laplacians to the global features, and the corresponding first k eigenvalues to the node features, providing direction-aware structural signals to both levels of representation. As a drawback, this positional encoding results in higher computational costs, as it is necessary to compute several eigenvalue decompositions and, in addition, results in a higher dimensional positional encoding to be processed by the denoising network.

Directed Relative Random Walk Probabilities (RRWP) For undirected graphs, RRWP encodes the likelihood of arriving from one node to another through k-step random walks. This is usually expressed via a transition matrix $\boldsymbol{T}^k$, with $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{D}^{-1}$, that encodes the transition probabilities.

For directed graphs, it is possible to consider $\boldsymbol{T} = \boldsymbol{A}\boldsymbol{D}^{-1}_{out}$, where we take into account the degree matrix of the outgoing edges $\boldsymbol{D}^{-1}_{out}$ [19]. The K-step transition probabilities are then captured by powers of $\boldsymbol{T}$, producing the sequence $[\boldsymbol{I}, \boldsymbol{T}, \boldsymbol{T}^2, \ldots, \boldsymbol{T}^{K-1}]$.

Additionally, in the directed setting, authors in [19] suggest it is also informative to model reverse random walks, starting from incoming edges. For this, we can define the reverse transition matrix $\boldsymbol{R} = \boldsymbol{A}^\top \boldsymbol{D}^{-1}_{\text{in}}$, where $\boldsymbol{D}_{\text{in}}$ is the in-degree matrix. This would represent the likelihood of arriving at a given edge starting from another through k-step random walks. Analogously, we compute $[\boldsymbol{I}, \boldsymbol{R}, \boldsymbol{R}^2, \ldots, \boldsymbol{R}^{K-1}]$.

To have our final positional encoding, following [19], we ensure that graphs are not nilpotent and that the probabilities encoded in the matrices sum up to 1 by adding self-loops to sink nodes during pre-processing. Finally, the full positional encoding concatenates both forward and reverse walk features for a pre-defined K, yielding:

$$\text{RRPW}(G) = [\boldsymbol{I}, \boldsymbol{T}, \boldsymbol{T}^2, \ldots, \boldsymbol{T}^{K-1}, \boldsymbol{I}, \boldsymbol{R}, \boldsymbol{R}^2, \ldots, \boldsymbol{R}^{K-1}]. \quad (31)$$

We concatenate the PE to the edge features, and additionally concatenate the diagonal elements of each matrix to the node features to further infuse them with the information.

Additionally, to mitigate that for large k the probabilities tend to converge to sink nodes, we optionally incorporate *Personalized PageRank (PPR)* features. The PPR matrix is defined in closed form as:

$$\text{PPR} = p_r(\boldsymbol{I} - (1 - p_r)\boldsymbol{T})^{-1}, \quad (32)$$

where p_r denotes the restart probability.

C Sampling Optimization in DIRECTO

The decoupling of training and sampling in discrete flow matching enables principled, training-free modifications to the sampling procedure, offering a flexible design space to enhance generation quality across digraph distributions [52]. In this section, we detail the three core parameters that govern sampling behavior: (i) time distortion functions, (ii) target guidance intensity, and (iii) stochasticity strength. Each plays a distinct role in shaping the denoising trajectory and contributes to different aspects of generation quality and structural fidelity.

Time Distortion Functions A central part for sampling optimization lies in the application of *time distortion functions*. These functions transform the time variable $t \in [0, 1]$ through a bijective, monotonic mapping $f(t)$, yielding a distorted time variable $t' = f(t)$. While used in both training and sampling, their objectives differ. In training, time distortions skew the sampling distribution over time, concentrating the model's learning capacity on specific denoising intervals. In sampling, they induce variable step sizes along the denoising trajectory, with finer resolution where structural sensitivity is highest (e.g., late-stage edits near $t = 1$).

Formally, the probability density function (PDF) of the transformed time variable t' is:

$$\phi_{t'}(t') = \phi_t(t) \left| \frac{d}{dt'} f^{-1}(t') \right|, \quad \text{where } \phi_t(t) = 1 \text{ for } t \in [0, 1]. \tag{33}$$

We consider the same five representative time distortion functions from [52] that yield diverse distributions for t':

- **Identity**: $f(t) = t$ — Uniform time density, baseline behavior.
- **Polydec**: $f(t) = 2t - t^2$ — Increasing density over time, emphasizing late-stage denoising.
- **Cos**: $f(t) = \frac{1-\cos(\pi t)}{2}$ — Concentrates density at both boundaries.
- **Revcos**: $f(t) = 2t - \frac{1-\cos(\pi t)}{2}$ — Peaks at intermediate times.
- **Polyinc**: $f(t) = t^2$ — Decreasing density over time, emphasizes early steps.

In the sampling phase, the induced variable step sizes, particularly from functions like *polydec*, allow for finer resolution near the clean data, where structural coherence is most fragile. Empirically, these functions are selected based on dataset-specific properties to improve fidelity and structural constraints without need for retraining.

Target Guidance Another axis for improving sampling efficiency is the modification of the *conditional rate matrix* R_t to better guide the generation trajectory toward the clean target graph z_1. This is achieved by incorporating an additive guidance term:

$$R_t(z_t, z_{t+\Delta t} \mid z_1) = R_t^*(z_t, z_{t+\Delta t} \mid z_1) + \omega \cdot \frac{\delta(z_{t+\Delta t}, z_1)}{\mathcal{Z}_t^{>0} \cdot p_{t|1}(z_t \mid z_1)}. \tag{34}$$

Here, $\omega \in \mathbb{R}_+$ controls the strength of the guidance, $\mathcal{Z}_t^{>0}$ is a discrete variable with Z possible positive values, and $\delta(\cdot, \cdot)$ is the Kronecker delta. Intuitively, this formulation prioritizes transitions that directly align with the clean graph.

Stochasticity Control The final sampling-time parameter is the *stochasticity coefficient* η. This parameter scales an auxiliary rate matrix R_t^{DB} that satisfies the detailed balance condition [8] and adds it to the optimal rate matrix R_t^*:

$$R_t^\eta = R_t^* + \eta \cdot R_t^{\text{DB}}. \tag{35}$$

The role of η is to regulate trajectory stochasticity: higher values promote broader exploration during denoising, potentially correcting suboptimal states. Setting $\eta = 0$ recovers the deterministic path prescribed by R_t^*, minimizing the expected number of transitions.

Sampling Optimization Together, these sampling parameters constitute a flexible, principled toolkit for tailoring the sampling procedure to diverse graph domains and structural requirements. In our case, we optimize these parameters at sampling time to improve the performance of DIRECTO. In particular, we individually optimize each of them, and perform the final sampling using the best performing parameters in terms of V.U.N. and ratio performance. A study of the effect of these parameters on sampling performance can be found in H.7.

D DIRECTO Training and Sampling Algorithms

This appendix provides detailed pseudocode for the training and sampling procedures used in the DIRECTO framework. Alg. 1 outlines the training loop, where a digraph is progressively noised and a denoising model is optimized to reconstruct the original graph.

We consider the original data distribution $\boldsymbol{p}_1$ as well as the time distribution $\mathcal{T}$ which in our case is set to be the uniform distribution. In addition, f^θ refers to the denoising network, in our case the graph transformer with dual attention.

Algorithm 1 DIRECTO Training

Input: Graph dataset $\{G^1, \dots, G^M\} \sim \boldsymbol{p}_1$
while f_θ not converged **do**
  Sample $G \sim \boldsymbol{p}_1$
  Sample $t \sim \mathcal{T}$
  Iteratively sample $G_t \sim \boldsymbol{p}_{t|1}(G_t|G)$ ▷ Noising process
  $h \leftarrow \mathrm{PosEnc}(G_t)$ ▷ Positional encoding
  $\boldsymbol{p}^\theta_{1|t}(\cdot|G_t) \leftarrow f_\theta(G_t, h, t)$ ▷ Denoising prediction with dual attention
  loss $\leftarrow \mathrm{CE}_\lambda(G, \boldsymbol{p}^\theta_{1|t}(\cdot|G_t))$
end while

Alg. 2 describes the generative sampling process in which new digraphs are synthesized by iteratively denoising from an initial random digraph structure. Again, we consider $\boldsymbol{p}_0 = \boldsymbol{p}_{noise}$ the predefined noise distribution, f^θ the denoising graph transformer with dual attention, $\boldsymbol{p}^\theta_{1|t}$ the denoising predictions, and $\boldsymbol{p}^\theta_{t+\Delta t|t}$ as the update rule, see Equation (42).

Algorithm 2 DIRECTO Sampling

Input: # graphs to sample S
for $i = 1$ to S **do**
  Sample N from train set ▷ # nodes
  Sample $G_0 \sim \boldsymbol{p}_0(G_0)$
  for $t = 0$ to $1 - \Delta t$ with step Δt **do**
    $h \leftarrow \mathrm{PosEnc}(G_t)$ ▷ Positional encoding
    $\boldsymbol{p}^\theta_{1|t}(\cdot|G_t) \leftarrow f_\theta(G_t, h, t)$ ▷ Denoising prediction with dual attention
    $G_{t+\Delta t} \sim \boldsymbol{p}^\theta_{t+\Delta t|t}(G_{t+\Delta t}|G_t)$ ▷ Update rule
  end for
  Store G_1 as G^i
end for

E Dataset descriptions

In this section, we provide further detailed information about the datasets described in Section **??** and used for the experiments in Section 3.

E.1 Synthetic datasets

We outline the mathematical formulation of each graph generation strategy used in our experiments. All graphs are generated as adjacency matrices $\boldsymbol{A} \in \{0,1\}^{n\times n}$ (where $\boldsymbol{A}_{ij} = 1$ denotes a directed edge from node i to node j), and preprocessed to adapt to the structure required by the model.

For all datasets, we generate a total of 200 graphs, split as: 128 train, 32 validation, 40 test. A study on the effect of the dataset size on the quality of the generations can be found in Appendix H.6. Furthermore, the dataset statistics can be found in Table 2.

Table 2: Synthetic dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	#Train	#Val	#Test
Erdos-Renyi (ER)	20	80	46	223	3670	1446	128	32	40
SBM	44	175	106	340	3008	1426	128	32	40
ER (DAG)	20	80	49	103	1892	762	128	32	40
Price (DAG)	64	64	64	132	246	197	128	32	40

Erdős-Renyi [17] We generate graphs with a variable number of nodes between $n = 20$ and $n = 80$. For a number of nodes n, edges are sampled independently with a fixed probability $p = 0.6$. The adjacency matrix $\boldsymbol{A}$ is generated by:

$$\boldsymbol{A}_{ij} \sim \text{Bernoulli}(p), \quad \forall i \neq j$$

To generate a second dataset of DAGs, we enforce a lower-triangular structure (i.e., edges only from higher-indexed to lower-indexed nodes), which results in graphs generated with a probability $p = 0.3$.

Price's model [14] This is the directed version of the Barabási-Albert or preferential attachment model, which simulates the growth of citation networks. We build DAGs sequentially for a fixed number of nodes $N = 64$. Each new node i forms $m = \log_2(N)$ edges to existing nodes j with probability proportional to their degree:

$$\mathbb{P}(i \to j) \propto \deg(j).$$

Since the number of nodes is fixed to $N = 64$, mean out-degree results in $m = 6$. In practice, we implement this "bag" of nodes that replicates existing connections. For each new node i, we sample m destination nodes from the bag. Then, for each selected node j, we set $\boldsymbol{A}_{ij} = 1$ and add i and the selected nodes to the bag. This ensures a directed acyclic graph (DAG) due to the order of node addition.

Stochastic Block Model [24] We create K communities with sizes $\{n_1, n_2, \ldots, n_K\}$. We set the number of communities between $K = 2$ and $K = 5$, and the number of nodes per community between 20 and 40. The probability of an edge between any two nodes depends on their community assignments

$$\boldsymbol{A}_{ij} \sim \text{Bernoulli}(P_{z_i z_j})$$

where $z_i \in \{1, \ldots, K\}$ is the block assignment of node i and $P \in [0,1]^{K\times K}$ is a matrix of intra- and inter-community connection probabilities. In particular, we use

$$P_{kk} = p_{\text{intra}} = 0.3, \quad P_{kl} = p_{\text{inter}} = 0.05 \text{ for } k \neq l$$

to generate a block-structured, directed graph.

E.2 TPU Tiles

In this dataset introduced in [51], each DAG represents a computation within a machine learning workload, such as a training epoch or an inference step. Each of the 12602 data points includes a computational graph, a specific compilation configuration, and the execution time of the graph when compiled with that configuration. The dataset features different model architectures, such as ResNets, EfficientNets, Masked R-CNNs, and Transformers, with graphs of up to ~ 400 nodes.

We adopt the processing and splits from [36]. Details on dataset statistics for the three different splits can be found in Table 3.

Table 3: TPU Tiles dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Train	2	394	41	1	711	43	5040
Validation	2	113	41	1	123	43	630
Test	2	154	41	1	249	44	631

E.3 Visual Genome

The Visual Genome dataset [32] aims at bridging computer vision and natural language understanding by providing richly annotated images. It comprises over 100,000 images annotated with object labels, attributes, relationships, which allows to capture the presence of objects but also their attributes and interactions. It has become a standard benchmark for tasks such as scene graph generation, visual question answering, and grounded language understanding.

To build the directed graph dataset, we set 3 types of nodes: objects, attributes, and relationships, with the actual node class being the text label. Then, we link them via directed edges according to the visual rules provided in the original data and taking into account that:

1. **Objects** have outgoing edges to attributes and relationships and incoming edges from relationships.
2. **Relationships** have outgoing edges to objects and incoming edges from objects.
3. **Attributes** have no outgoing edges and incoming edges from objects.

Once we built the digraphs, we select a relevant subset from this raw dataset by keeping graphs with 20 to 40 nodes, and with a minimum of 25 edges per graph. Then, for each graph, we only consider objects, attributes, and relationships that are in the top-20 in terms of prevalence in the full dataset, ending with 60 node classes. We randomly split the graphs into 203 train, 52 validation and 63 test graphs, with detailed statistics of each split available in Table 4. The total number of cyclic graph in the dataset is 119 ($\sim$37.5% of the digraphs).

Table 4: Visual Genome dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Global	21	40	34	25	47	28	317
Train	21	40	33	25	47	28	203
Validation	24	40	34	25	40	28	51
Test	21	40	34	25	36	28	63

The selected top-20 objects, relationships, and attributes are:

- **Objects:** `['window', 'tree', 'man', 'shirt', 'wall', 'person', 'building', 'ground', 'sign', 'light', 'sky', 'head', 'leaf', 'leg', 'hand', 'pole', 'grass', 'hair', 'car', 'cloud']`
- **Relationships:** `['on', 'has', 'in', 'of', 'wearing', 'with', 'behind', 'holding', 'on top of', 'on a', 'near', 'next to', 'has a', 'on', 'under', 'by', 'of a', 'wears', 'above', 'sitting on']`
- **Attributes:** `['white', 'black', 'blue', 'green', 'red', 'brown', 'yellow', 'small', 'large', 'wooden', 'gray', 'silver', 'metal', 'orange', 'grey', 'tall', 'long', 'dark', 'pink', 'clear']`

F Further details on evaluation metrics

In this section, we detail the statistical procedures used to evaluate the graphs generated by our model. We detail how we compute the validity metrics for the different datasets, as well as how we make the validity and uniqueness computation more efficient.

F.1 Validity metrics

Erdős-Renyi Given a graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges, we want to determine whether the graph likely originates from an Erdős–Rényi model $G(n, p)$ with edge probability p using a Wald test [22]. For that, we follow the following steps distinguishing the cases in which the graphs are DAGs or not:

1. **Empirical edge probability:** we compute the number of possible edges

$$m_{\max} = \begin{cases} \frac{n(n-1)}{2} & \text{if the graph acyclic} \\ n(n-1) & \text{if the graph is not acyclic} \end{cases}$$

   and then estimate the empirical probability of edge presence $\hat{p} = \frac{m}{m_{\max}}$.

2. **Wald test statistic:** the Wald statistic tests the null hypothesis $H_0 : \hat{p} = p$, where p is the expected edge probability:

$$W = \frac{(\hat{p} - p)^2}{\hat{p}(1 - \hat{p}) + \varepsilon}$$

   and where we add a small regularization $\varepsilon = 10^{-6}$ to prevent division by zero.

3. **p-value computation**: assuming the null hypothesis, the Wald statistic W asymptotically follows a Chi-squared distribution with 1 degree of freedom:

$$p\text{-value} = 1 - F_{\chi^2}(W;\, df = 1)$$

   where F_{χ^2} is the cumulative distribution function (CDF) of the Chi-squared distribution.

In the rare limit case where graphs contain only one node, we consider that the graph does not follow the distribution.

Stochastic Block Model To assess whether a graph $G = (V, E)$ conforms to a Stochastic Block Model (SBM), we recover the block structure and probability parameters. The test then computes Wald statistics for intra- and inter-block edge probabilities.

1. **Block assignment via model inference:** Given the adjacency matrix $A \in \{0, 1\}^{n \times n}$ of a graph G, we use an algorithm [49] to infer the stochastic block model (block assignment) from a given network:

$$\text{Infer } z : V \to \{1, \ldots, B\} \quad \text{using } \min \mathcal{L}(G, z)$$

   where B is the number of non-empty blocks found and $\mathcal{L}$ is the description length. The model can be further refined using Markov-Chain Monte Carlo (MCMC) for T timesteps.

2. **Estimating intra- and inter-block probabilities:** let N_i be number of nodes in block i and E_{ij} the number of edges between block i and block j recovered by the algorithm in the previous step. Then the estimated probabilities for directed SBM graphs are:

$$\hat{p}_{\text{intra},i} = \frac{E_{ii}}{N_i(N_i - 1) + \varepsilon}$$

$$\hat{p}_{\text{inter},ij} = \frac{E_{ij}}{N_i N_j + \varepsilon}, \quad i \neq j$$

   where $\varepsilon = 10^{-6}$ is a small regularization constant.

3. **Wald test statistic:** we compare the empirical estimates to expected values p_{intra} and p_{inter} using a Wald statistic:

$$W_{ii} = \frac{(\hat{p}_{\text{intra},i} - p_{\text{intra}})^2}{\hat{p}_{\text{intra},i}(1 - \hat{p}_{\text{intra},i}) + \varepsilon}$$

$$W_{ij} = \frac{(\hat{p}_{\text{inter},ij} - p_{\text{inter}})^2}{\hat{p}_{\text{inter},ij}(1 - \hat{p}_{\text{inter},ij}) + \varepsilon}, \quad i \neq j$$

4. p**-value computation**: assuming the null hypothesis (i.e., estimated probabilities match expected ones), each Wald statistic follows a Chi-squared distribution with 1 degree of freedom:
$$p_{ij} = 1 - F_{\chi^2}(W_{ij};\, df = 1),$$
and therefore the overall p-value can be computed as the mean of all p_{ij}
$$p\text{-value} = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} p_{ij}$$

Price's model To assess whether a graph follows a Price's model, we use a nonparametric two-sample Kolmogorov–Smirnov (KS) test:

1. **Computing degree distribution:** let $D = \{d_1, d_2, \ldots, d_n\}$ be the degree sequence of the nodes in G. To reduce noise and ensure a more stable distribution, ignore nodes with degree ≤ 1 and end with the distribution $D' = \{d_i \in D \mid d_i > 1\}$.
2. **Synthetic graph generation:** we construct a synthetic graph G_{BA} using the Barabási–Albert model with the same number of nodes: $G_{\text{BA}} \sim \text{BA}(n, m)$, where $n = |V|$ is the number of nodes and $m = 6$ is the number of edges each new node adds during attachment, fixed to the same value as in our data generation. The degree sequence of the synthetic graph is D_{BA}.
3. **Kolmogorov-Smirnov Two-Sample test:** we perform a two-sample Kolmogorov–Smirnov (KS) test to compare the empirical distribution functions of D' and D_{BA}
$$p\text{-value} = 1 - \text{KS}_{2\ sampled}(D', D_{\text{BA}})$$
which evaluates the null hypothesis: $H_0 : D' \sim D_{\text{BA}}$.

TPU Tiles For the TPU Tiles dataset, we report the percentage of valid Directed Acyclic Graphs (DAGs) as our primary validity metric, reflecting our focus on accurately reconstructing acyclic computational graphs.

Visual Genome As seen in Appendix E.3, by construction, object nodes can have outgoing edges to relationship or attribute nodes but only receive incoming edges from relationship nodes. Relationship nodes only have incoming edges from object nodes and send outgoing edges to objects. Finally, attributes can only receive edges from object nodes and do not have any outgoing edges. Therefore, to measure digraph validity we evaluate that the generated graphs verify these constraints.

F.2 Uniqueness and novelty

To evaluate the quality and novelty of generated graphs, we employ the usual metrics based on graph isomorphism. In particular we measure the fraction of isomorphic graphs from the sampled set to the train set (uniqueness) and the fraction of graphs from the sampled set that are not isomorphic to any other in the same sampled set (novelty).

F.3 Maximum Mean Discrepancy metrics

For the different metrics, we adapt previous work from [43, 33]. We propose to measure both out-degree and in-degree distributions, as well as clustering, spectre and walvelet. To measure the *clustering* coefficient, we compute the distribution of directed local clustering coefficients using `networkx` function `clustering()`, which supports digraphs. For the spectral features (*spectre* and *wavelet*), we derive the spectral features from the directed Laplacian [10]:
$$\boldsymbol{L} = \boldsymbol{I} - \frac{1}{2}\left(\boldsymbol{\Phi}^{\frac{1}{2}}\boldsymbol{P}\boldsymbol{\Phi}^{-\frac{1}{2}} + \boldsymbol{\Phi}^{-\frac{1}{2}}\boldsymbol{P}\boldsymbol{\Phi}^{\frac{1}{2}}\right), \tag{36}$$
where $\boldsymbol{P}$ is the random walk transition matrix and $\boldsymbol{\Phi}$ is the diagonal matrix of the Perron vector of $\boldsymbol{P}$. This choice simplifies computations compared to alternatives such as the Magnetic Laplacian.

In addition, the *orbit* metric that was also computed in [43] has not been adapted to the directed setting as it relies on the Orbit Counting Algorithm (ORCA) which counts graphlets in networks but is only implemented in the undirected setting. Therefore, adapting this metric to the directed setting remains an open challenge.

G Experimental details

G.1 Details on baselines

Maximum Likelihood Estimation (MLE) We define a simple baseline by estimating empirical probabilities from the training dataset via maximum likelihood estimation. The following distributions are computed:

1. **Number of Nodes Distribution:** Let n denote the number of nodes in a graph. The empirical distribution over n is given by:
$$P(n) = \frac{\text{Number of training graphs with } n \text{ nodes}}{\text{Total number of training graphs}}.$$
2. **Node Class Distribution:** Let c be a node class. The empirical node class distribution is:
$$P(v = c) = \frac{\text{Number of nodes of class } c}{\text{Total number of nodes in all training graphs}}.$$
3. **Edge Type Distribution Conditioned on Node Pairs:** Let (v_i, v_j) be a pair of nodes such that v_i has class c_a and v_j has class c_b. Let $e_{ij} \in \{1, \dots, K\}$ denote the edge type between them (with K total edge types). The empirical conditional distribution is:
$$P(e_{ij} = k \mid v_i = c_a, v_j = c_b) = \frac{\text{Number of edges of type } k \text{ between } (c_a, c_b)}{\sum_{k'=1}^{K} \text{Number of edges of type } k' \text{ between } (c_a, c_b)}.$$

These distributions are stored and later used to construct graphs by first sampling the number of nodes, then sampling node types independently, and finally sampling edge types between each node pair according to the conditional edge distribution.

D-VAE [72] This autoregressive method encodes and decodes directed acyclic graphs (DAGs) using an asynchronous message passing scheme. Unlike standard graph neural networks (GNNs), which apply simultaneous message passing across all nodes and updates them all at once, D-VAE updates nodes sequentially, allowing it to capture the computational flow within the graph rather than just its structure. During generation, the DAG is built one node at a time in topological order, with edges always directed toward newly added nodes ensuring the resulting graph remains acyclic. To enable this process, input graphs must be topologically sorted before being processed by the model.

To reproduce this baseline, we adapted the code available in their GitHub and arranged both the TPU Tiles and ER-DAG datasets to match the input format expected by the original model. We used the same hyperparameter configurations as suggested in the original paper to ensure consistency in training, which happened for 500 epochs. After generation, we converted the output DAGs back into the required format for compatibility with our evaluation metrics.

For ER-DAG, we tested two different learning rates 10^{-4} and 10^{-5}, and ended using the model with 10^{-5}, which resulted in better V.U.N. results with a training time of $\sim$3min per epoch. For TPU Tiles, due to the large size of the dataset and some of the graphs in it, it was impossible to perform a training epoch in a reasonable time, as each of them was predicted to take $\sim$12h.

LayerDAG [36] This autoregressive diffusion model generates DAGs by converting them into sequences of bipartite graphs, effectively enabling a layer-wise tokenization suitable for autoregressive generation. At each step, a layer-wise diffusion process captures dependencies between nodes that are not directly comparable (i.e., not connected by a path). As with other autoregressive DAG models, the input graphs must be topologically sorted to ensure correct processing.

To reproduce this baseline, we adopted the implementation from GitHub adapted the ER-DAG data to the correct format ordering the graphs topologically, as TPU Tiles was already in the correct format. We kept the hyperparameters from the default configurations, including the number of epochs for each of the elements of the model. After generation, we converted the output DAGs back into the required format for compatibility with our evaluation metrics.

DiGress [63] To benchmark against DiGress, we consider its directed version by simply removing the symmetrization operations. This corresponds to Directo-DD without dual attention and using the Laplacian positional encoding.

DeFoG [52] To benchmark against DEFOG, we consider its directed version by again removing the symmetrization operations. In this case this corresponds to DIRECTO without dual attention and using the RRWP Positional encoding (without PPR).

G.2 Training setup

This section details the training hyperparameters used across all experiments presented in the main text. Unless otherwise specified, the values reported below were used uniformly across all positional encoding variants and ablation settings. In particular, we report the choices for the training, for the positional encodings employed, and for the sampling.

Training The training hyperparameters used in our experiments are summarized in Table 5. These settings were consistently applied across all training runs for both DIRECTO and DIRECTO-DD (with discrete diffusion). We trained each model for up to 10,000 epochs, stopping the runs based on validation performance to account for variations in convergence behavior across datasets.

Table 5: Training and model hyperparameters used in all experiments.

Hyperparameter	**Value**
Training Settings	
Learning rate	0.0002
Weight decay	1×10^{-12}
Optimizer	AdamW
Model Settings	
Initial distribution	Marginal
Train distortion	Identity
Number of diffusion steps	500
Noise schedule	Cosine
Number of layers	5
Hidden MLP dimensions	X: 256, E: 128, y: 128
Transformer hidden dimensions	d_x: 256, d_e: 64, d_y: 64, n_{head}: 8
Feedforward dims	dim_{ff_X}: 256, dim_{ff_E}: 128, dim_{ff_y}: 128
Training loss weights (λ_{train})	[5, 0]

Positional encodings For RRWP, we set the walk length parameter to $K = 20$. For MagLap with $Q = 5$, we used equidistant potentials $\boldsymbol{q}_5 = (0, 0.1, 0.2, 0.3, 0.4)$. For $Q = 10$, we selected again 10 equidistant potentials $\boldsymbol{q}_{10} = (0.01, \dots, 0.1)$. In all cases, we then kept the $k = 10$ first eigenvalues and eigenvectors (padding with null values whenever the graphs had less than 10 nodes).

Sampling To perform sampling optimization, we conducted a targeted search over the three key hyperparameters: the time distortion function, the target guidance factor (ω), and the stochasticity coefficient (η). Each hyperparameter was varied independently, while the others were held at their default values (identity distortion and $\omega = \eta = 0$). For computational efficiency, all searches were performed using 100 sampling steps, and each configuration was evaluated using five independent sampling runs to ensure robustness. An exception was made for the TPU Tiles dataset, where only a single sampling run was performed per configuration due to the large size of the dataset and some individual graphs.

Based on the outcome of this optimization, we selected the best-performing configurations for each combination of dataset and positional encoding, balancing the trade-off between V.U.N and ratio. The optimal configurations of the results reported in the main paper are summarized in Table 6. For the final evaluation, we performed five sampling runs per configuration using 1000 sampling steps. In all cases, the number of generated graphs matched the size of the test set (40 for synthetic datasets, 63 for Visual Genome), except for TPU Tiles, where we again limited the number of sampled digraphs to 40 due to computational constraints.

For DIRECTO-DD we performed 500 sampling steps in all the different model configurations.

Table 6: Optimal sampling hyperparameters for each dataset and positional encoding variant.

Encoding	Distortion	ω	η
ER-DAG			
Baseline	Polydec	2	10
RRWP	Polydec	0.1	25
MagLap	Polydec	0	100
SBM			
Baseline	Revcos	0.3	5
RRWP	Polyinc	0.01	10
MagLap	Polydec	0.01	10
TPU Tiles			
Baseline	Polydec	0.01	25
RRWP	Polyinc	0.3	10
MagLap	Polydec	0.4	50
Visual Genome			
Baseline	Cos	0.05	0
RRWP	Polydec	0.3	100
MagLap	Polydec	0.1	10

G.3 Resources and runtime

All experiments were conducted on a single NVIDIA A100-SXM4-80GB GPU. Table 7 summarizes the runtime of the training and sampling stages across the different datasets. These timings reflect the training time until best performance and the average sampling time per sample, for all the configurations reported in the main results for DIRECTO.

Table 7: Training and sampling runtimes for each dataset and positional encoding variant.

Encoding	Training graphs	Training time (h)	Graphs sampled	Sampling time (min/sample)
ER-DAG				
Baseline	128	13.6	40	0.125
RRWP	128	13.5	40	0.3
MagLap	128	14.4	40	0.8
SBM				
Baseline	128	23.7	40	0.3
RRWP	128	13.7	40	0.5
MagLap	128	21	40	2.1
TPU Tiles				
Baseline	5040	10	40	1.1
RRWP	5040	21	40	1.1
MagLap	5040	49	40	6.7
Visual Genome				
Baseline	203	7.5	63	0.9
RRWP	203	9	63	1.1
MagLap	203	10	63	1.3

H Additional results

In this section, we present additional results that offer deeper insights into our model's performance and design choices. Specifically, we analyze the impact of the dual attention mechanism and the choice of positional encoding, compare performance on ER and DAG accuracy metrics, examine how dataset size influences generation quality, and evaluate the effect of the sampling optimization step in discrete flow matching.

H.1 Synthetic datasets

Table 8 reports the results on ER-DAG, SBM and the real-world datasets. In the synthetic setting, DIRECTO consistently achieves the best trade-off between sample quality and validity. For ER-DAG, it achieves the highest validity score using directed RRWP as positional encoding, with a V.U.N. ratio of 94%. Notably, while LAYERDAG enforces acyclicity by design, it fails to capture the ER distribution, as evidenced by its low V.U.N. score. For SBM, DIRECTO also successfully captures the distribution and consistently outperforms the baselines, achieving up to 95.5% in V.U.N. ratio and low average graph statistics ratios.

Table 8: Directed graph generation performance for different configurations of DIRECTO. Results are the mean ± standard deviation across five sampling runs. We considered $Q = 10$ for MagLap.

Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
	Erdős-Renyi Directed Acyclic Graph (ER-DAG)									
Training set	0.0113	0.0103	0.0355	0.0038	0.0024	1.0	99.2	100	0.0	0.0
MLE	0.0083 ± 0.0004	0.0089 ± 0.0003	0.1318 ± 0.0077	0.0823 ± 0.0013	0.1162 ± 0.0018	15.1 ± 0.2	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
D-VAE	0.6158 ± 0.0160	0.6246 ± 0.0103	1.0509 ± 0.00304	0.6160 ± 0.0315	0.5432 ± 0.0389	106.6 ± 5.4	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
LAYERDAG	0.0750 ± 0.0697	0.1773 ± 0.0722	0.1842 ± 0.0463	0.0159 ± 0.0120	0.0218 ± 0.0160	4.2 ± 3.2	21.5 ± 2.7	99.9 ± 0.0	100 ± 0.0	21.5 ± 2.7
DIGRESS	0.0138 ± 0.0035	0.0143 ± 0.0050	0.1074 ± 0.0090	0.0073 ± 0.0017	0.0042 ± 0.0011	1.9 ± 0.3	34.0 ± 4.1	100 ± 0.0	100 ± 0.0	34.0 ± 4.1
DEFOG	0.0109 ± 0.0019	0.0108 ± 0.0017	0.0540 ± 0.0112	0.0061 ± 0.0008	0.0030 ± 0.0005	**1.3** ± 0.1	67.5 ± 1.6	100 ± 0.0	100 ± 0.0	67.5 ± 1.6
DIRECTO-DD RRWP	0.0142 ± 0.0040	0.0129 ± 0.0034	0.0408 ± 0.0053	0.0055 ± 0.0010	0.0048 ± 0.0014	<u>1.4</u> ± 0.3	79.0 ± 3.7	100 ± 0.0	100 ± 0.0	79.0 ± 3.7
DIRECTO-DD MagLap	0.0145 ± 0.0040	0.0134 ± 0.0033	0.0582 ± 0.0083	0.0063 ± 0.0015	0.0034 ± 0.0011	1.5 ± 0.2	85.0 ± 9.2	100 ± 0.0	100 ± 0.0	85.0 ± 9.2
DIRECTO RRWP	0.0107 ± 0.0019	0.0104 ± 0.0012	0.1209 ± 0.0194	0.0054 ± 0.0005	0.0043 ± 0.0008	1.7 ± 0.1	94.0 ± 1.0	100 ± 0.0	100 ± 0.0	**94.0** ± 1.0
DIRECTO MagLap	0.0117 ± 0.0014	0.0110 ± 0.0015	0.0711 ± 0.0120	0.0055 ± 0.0016	0.0026 ± 0.0004	**1.3** ± 0.2	92.0 ± 3.7	100 ± 0.0	100 ± 0.0	<u>92.0</u> ± 3.7
	Stochastic Block Model (SBM)									
Training set	0.0031	0.0031	0.0274	0.0027	0.0011	1.0	97.7	100	0.0	0.0
MLE	0.0020 ± 0.0002	0.0020 ± 0.0002	0.1973 ± 0.0162	0.0087 ± 0.0003	0.0508 ± 0.0009	11.6 ± 0.2	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
DIGRESS	0.0037 ± 0.0018	0.0038 ± 0.0018	0.0722 ± 0.0098	0.0045 ± 0.0004	0.0142 ± 0.0039	3.9 ± 0.9	41.5 ± 5.1	100 ± 0.0	100 ± 0.0	41.5 ± 5.1
DEFOG	0.0036 ± 0.0009	0.0034 ± 0.0008	0.1276 ± 0.0065	0.0345 ± 0.0028	0.0961 ± 0.0071	21.4 ± 1.6	13.5 ± 5.4	100 ± 0.0	100 ± 0.0	13.5 ± 5.4
DIRECTO-DD RRWP	0.0036 ± 0.0019	0.0036 ± 0.0018	0.0592 ± 0.0027	0.0038 ± 0.0007	0.0027 ± 0.0009	1.7 ± 0.4	81.5 ± 3.2	100 ± 0.0	100 ± 0.0	81.5 ± 3.2
DIRECTO-DD MagLap	0.0037 ± 0.0019	0.0038 ± 0.0018	0.0450 ± 0.0037	0.0038 ± 0.0006	0.0021 ± 0.0009	<u>1.5</u> ± 0.4	95.5 ± 3.7	100 ± 0.0	100 ± 0.0	<u>95.5</u> ± 3.7
DIRECTO RRWP	0.0038 ± 0.0011	0.0037 ± 0.0012	0.0492 ± 0.0041	0.0035 ± 0.0007	0.0015 ± 0.0004	**1.4** ± 0.2	87.0 ± 5.1	100 ± 0.0	100 ± 0.0	87.0 ± 5.1
DIRECTO MagLap	0.0039 ± 0.0012	0.0038 ± 0.0010	0.0654 ± 0.0052	0.0038 ± 0.0003	0.0039 ± 0.0008	2.0 ± 0.3	96.5 ± 2.5	100 ± 0.0	100 ± 0.0	**96.5** ± 2.5

H.2 Real-world datasets

Table 9 shows the performance for the two real-world datasets. In TPU tiles, all DIRECTO variants significantly outperform DAG generation baselines in terms of structure-aware metrics. The ratio scores for DIRECTO are consistently and substantially lower than those observed in MLE, and especially LayerDAG. This indicates a more realistic distribution. Again, DIRECTO variants achieve higher V.U.N. than all baselines, with the exception of LAYERDAG, which benefits from enforcing acyclicity, the only validity constraint evaluated in this case. For Visual Genome, results reinforce the versatility of DIRECTO. DIRECTO RRWP achieves the highest V.U.N. scores, indicating a superior ability to generate diverse and novel graphs while maintaining structural fidelity.

Table 9: Directed graph generation performance on real-world graphs for different configurations of DIRECTO. Results are presented as mean ± standard deviation across five sampling runs. We considered $Q = 5$ for the MagLap variants. OOT indicates that the model could not be run within a reasonable timeframe. $^{(*)}$ indicates that isomorphism tests occasionally timed out, due to the large size of some graphs in this dataset; such cases were excluded from the uniqueness computation.

Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
	TPU Tiles									
Training set	0.0003	0.0003	0.0007	0.0006	0.0002	1.0	100	100	0.0	0.0
MLE	0.0354 ± 0.0005	0.0878 ± 0.0014	0.0141 ± 0.0006	0.0689 ± 0.0013	0.0407 ± 0.0004	149.8 ± 0.7	24.7 ± 0.0	99.9 ± 0.0	100 ± 0.0	24.7 ± 0.0
D-VAE	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT
LAYERDAG	0.1933 ± 0.0905	0.2225 ± 0.0395	0.1512 ± 0.0522	0.0501 ± 0.0206	0.0765 ± 0.0251	413.6 ± 70.1	100 ± 0.0	99.5 ± 1.0	98.5 ± 3.0	**98.5** ± 3.0
DIGRESS	0.0084 ± 0.0009	0.0726 ± 0.0019	0.0020 ± 0.0006	0.0033 ± 0.0003	0.0018 ± 0.0003	<u>57.5</u> ± 1.7	86.1 ± 2.3	87.1$^{(*)}$ ± 5.8	99.4 ± 0.6	70.9 ± 3.4
DEFOG	0.0099 ± 0.0012	0.0794 ± 0.0023	0.0042 ± 0.0027	0.0040 ± 0.0005	0.0017 ± 0.0003	63.7 ± 2.6	86.3 ± 2.2	87.7$^{(*)}$ ± 5.4	93.3 ± 2.5	72.0 ± 2.4
DIRECTO-DD RRWP	0.0093 ± 0.0010	0.0767 ± 0.0035	0.0015 ± 0.0013	0.0045 ± 0.0007	0.0018 ± 0.0005	61.0 ± 2.9	86.5 ± 1.9	90.3 ± 0.8	99.6 ± 0.5	76.8 ± 1.9
DIRECTO-DD MagLap	0.0115 ± 0.0035	0.0703 ± 0.0033	0.0103 ± 0.0021	0.0076 ± 0.0007	0.0043 ± 0.0014	64.3 ± 5.3	86.5 ± 5.1	90.5 ± 3.3	100 ± 0.0	77.0 ± 7.0
DIRECTO RRWP	0.0133 ± 0.0025	0.0859 ± 0.0072	0.0136 ± 0.0075	0.0086 ± 0.0010	0.0038 ± 0.0009	75.4 ± 8.1	97.0 ± 1.0	81.8$^{(*)}$ ± 4.5	96.5 ± 1.2	77.0 ± 2.9
DIRECTO MagLap	0.0039 ± 0.0017	0.0376 ± 0.0051	0.0211 ± 0.0117	0.0126 ± 0.0022	0.0062 ± 0.0009	**44.0** ± 7.1	90.5 ± 3.3	90.5 ± 4.6	97.5 ± 3.2	<u>80.5</u> ± 4.6
	Visual Genome									
Training set	0.0018	0.0030	0.0000	0.0072	0.0036	1.0	100	100	0.0	0.0
MLE	0.0607 ± 0.0036	0.0474 ± 0.0023	0.5342 ± 0.0830	0.0535 ± 0.0015	0.0399 ± 0.0017	17.0 ± 0.6	0 ± 0.0	100 ± 0.0	100 ± 0.0	0 ± 0.0
DIGRESS	0.0654 ± 0.0044	0.0389 ± 0.0032	0.0008 ± 0.0008	0.0416 ± 0.0017	0.0225 ± 0.0007	17.0 ± 0.6	0.3 ± 0.6	100 ± 0.0	100 ± 0.0	0.3 ± 0.6
DEFOG	0.0447 ± 0.0044	0.0396 ± 0.0028	0.0002 ± 0.0002	0.0122 ± 0.0020	0.0089 ± 0.0011	10.6 ± 0.8	39.6 ± 2.8	100 ± 0.0	100 ± 0.0	39.6 ± 2.8
DIRECTO-DD RRWP	0.0424 ± 0.0043	0.0413 ± 0.0037	0.0000 ± 0.0000	0.0132 ± 0.0053	0.0074 ± 0.0008	15.3 ± 0.8	72.7 ± 3.9	100 ± 0.0	100 ± 0.0	<u>72.7</u> ± 3.9
DIRECTO-DD MagLap	0.0245 ± 0.0022	0.0359 ± 0.0036	0.0000 ± 0.0000	0.0163 ± 0.0026	0.0086 ± 0.0011	<u>7.6</u> ± 0.7	86.5 ± 5.1	100 ± 0.0	100 ± 0.0	61.9 ± 4.4
DIRECTO RRWP	0.0494 ± 0.0021	0.0425 ± 0.0029	0.0000 ± 0.0000	0.0226 ± 0.0075	0.0228 ± 0.0058	12.8 ± 0.6	86.0 ± 4.5	98.4 ± 1.7	99.3 ± 0.7	**83.8** ± 4.3
DIRECTO MagLap	0.0180 ± 0.0024	0.0302 ± 0.0040	0.0000 ± 0.0000	0.0142 ± 0.0021	0.0099 ± 0.0013	**6.2** ± 0.5	67.6 ± 3.6	99.7 ± 0.6	99.7 ± 0.6	67.0 ± 4.3

H.3 The role of dual attention

To assess the impact of the dual-attention mechanism in our model, we conduct an ablation study in which we remove the cross-attention between edge features and their transposes. This mechanism is designed to capture both source-to-target and target-to-source interactions, which are critical for modeling directional dependencies in directed graphs. By disabling dual attention, we isolate its contribution to generation quality, particularly in terms of validity, expressiveness, and generalization. We compare the full model with its ablated variant across model combinations to quantify the importance of this component.

DIRECTO Table 10 presents ablation results highlighting the impact of the dual attention mechanism in our method. The results are presented for two datasets: ER-DAG and SBM.

Table 10: Directed Graph Generation performance across different transformer architectures using discrete flow matching.

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
ER-DAG	No PE	0.0078 ± 0.0008	0.0078 ± 0.0010	0.1293 ± 0.0024	0.0735 ± 0.0023	0.0874 ± 0.0030	12.2 ± 0.4	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
	No PE - Dual	0.0109 ± 0.0012	0.0110 ± 0.0012	0.0807 ± 0.0147	0.0073 ± 0.0007	0.0215 ± 0.0017	3.0 ± 0.2	47.0 ± 12.1	100 ± 0.0	100 ± 0.0	47.0 ± 12.1
	RRWP	0.0109 ± 0.0019	0.0108 ± 0.0017	0.0540 ± 0.0112	0.0061 ± 0.0008	0.0030 ± 0.0005	1.3 ± 0.1	67.5 ± 1.6	100 ± 0.0	100 ± 0.0	67.5 ± 1.6
	RRWP - Dual	0.0107 ± 0.0019	0.0104 ± 0.0012	0.1209 ± 0.0194	0.0054 ± 0.0005	0.0043 ± 0.0008	1.7 ± 0.1	94.0 ± 1.0	100 ± 0.0	100 ± 0.0	94.0 ± 1.0
	MagLap	0.0113 ± 0.0024	0.0110 ± 0.0015	0.1196 ± 0.0107	0.0061 ± 0.0011	0.0029 ± 0.0006	1.7 ± 0.2	74.0 ± 4.4	100 ± 0.0	100 ± 0.0	74.0 ± 4.4
	MagLap - Dual	0.0110 ± 0.0017	0.0116 ± 0.0021	0.0509 ± 0.0017	0.0062 ± 0.0014	0.0027 ± 0.0005	1.3 ± 0.2	91.0 ± 2.5	100 ± 0.0	100 ± 0.0	91.0 ± 2.5
SBM	No PE	0.0038 ± 0.0021	0.0040 ± 0.0020	0.1912 ± 0.0178	0.0975 ± 0.0028	0.0961 ± 0.0071	20.5 ± 2.0	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
	No PE - Dual	0.0038 ± 0.0011	0.0037 ± 0.0010	0.0638 ± 0.0059	0.0037 ± 0.0003	0.0127 ± 0.0018	3.5 ± 0.4	48.5 ± 6.4	100 ± 0.0	100 ± 0.0	48.5 ± 6.4
	RRWP	0.0036 ± 0.0009	0.0034 ± 0.0008	0.1276 ± 0.0065	0.0345 ± 0.0028	0.0961 ± 0.0071	21.4 ± 1.6	13.5 ± 5.4	100 ± 0.0	100 ± 0.0	13.5 ± 5.4
	RRWP - Dual	0.0038 ± 0.0011	0.0037 ± 0.0012	0.0492 ± 0.0041	0.0035 ± 0.0007	0.0015 ± 0.0004	1.4 ± 0.2	87.0 ± 5.1	100 ± 0.0	100 ± 0.0	87.0 ± 5.1
	MagLap	0.0033 ± 0.0008	0.0034 ± 0.0008	0.1356 ± 0.0118	0.0481 ± 0.0022	0.1250 ± 0.0017	27.7 ± 4.3	21.5 ± 2.0	100 ± 0.0	100 ± 0.0	21.5 ± 2.0
	MagLap - Dual	0.0039 ± 0.0011	0.0036 ± 0.0010	0.0653 ± 0.0026	0.0033 ± 0.0003	0.0038 ± 0.0010	1.9 ± 0.3	77.0 ± 7.6	100 ± 0.0	100 ± 0.0	77.0 ± 7.6

In particular, we observe that, across both datasets, the inclusion of the dual attention mechanism consistently improves the validity of the generated graphs and reduces structural discrepancies as measured by clustering, spectral, and wavelet distances. In the ER-DAG setting, models with dual attention (e.g., RRWP-Dual and MagLap-Dual) significantly outperform their non-dual counterparts in validity—achieving up to 94% validity with RRWP-Dual and 91% with MagLap-Dual, compared to just 67.5% and 74%, respectively. These models also demonstrate better structural fidelity, particularly in clustering and spectral metrics. A similar trend is observed in the SBM dataset, where dual attention boosts validity (e.g., RRWP-Dual: 87%, MagLap-Dual: 77%). These results demonstrate how incorporating bidirectional information flow into the attention mechanism contributes to improved model performance and provides insight into the effectiveness of this architectural component.

DIRECTO-DD Table 11 shows a second ablation assessing the contribution of the dual attention mechanism within the discrete diffusion framework. In this case, we see a similar pattern to that observed under discrete flow matching: the incorporation of dual attention leads to consistent gains in graph validity across both ER-DAG and SBM datasets. Models augmented with dual attention not only produce more valid graphs but also exhibit improved alignment with structural statistics such as clustering, spectre, and wavelet.

Table 11: Directed Graph Generation performance across different transformer architectures using discrete diffusion.

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
ER-DAG	No PE	0.0094 ± 0.0024	0.0095 ± 0.0023	0.1221 ± 0.0122	0.0814 ± 0.0030	0.1094 ± 0.0066	14.4 ± 0.8	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
	No PE - Dual	0.0149 ± 0.0037	0.0138 ± 0.0035	0.1256 ± 0.0088	0.0089 ± 0.0018	0.0106 ± 0.0014	2.6 ± 0.3	53.0 ± 2.9	100 ± 0.0	100 ± 0.0	53.0 ± 2.9
	RRWP	0.0147 ± 0.0038	0.0139 ± 0.0038	0.0985 ± 0.0154	0.0067 ± 0.0009	0.0038 ± 0.0009	2.1 ± 0.3	42.0 ± 1.0	100 ± 0.0	100 ± 0.0	42.0 ± 1.0
	RRWP - Dual	0.0142 ± 0.0040	0.0129 ± 0.0034	0.0408 ± 0.0053	0.0055 ± 0.0010	0.0048 ± 0.0014	1.4 ± 0.3	79.0 ± 3.7	100 ± 0.0	100 ± 0.0	79.0 ± 3.7
	MagLap	0.0144 ± 0.0040	0.0135 ± 0.0038	0.0792 ± 0.0126	0.0071 ± 0.0012	0.0036 ± 0.0011	1.6 ± 0.3	59.0 ± 6.8	100 ± 0.0	100 ± 0.0	59.0 ± 6.8
	MagLap - Dual	0.0142 ± 0.0044	0.0135 ± 0.0034	0.0438 ± 0.0071	0.0057 ± 0.0013	0.0046 ± 0.0010	1.4 ± 0.3	69.0 ± 6.4	100 ± 0.0	100 ± 0.0	69.0 ± 6.4
SBM	No PE	0.0038 ± 0.0021	0.0040 ± 0.0020	0.1912 ± 0.0177	0.0120 ± 0.0019	0.0975 ± 0.0081	20.5 ± 2.0	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
	No PE - Dual	0.0037 ± 0.0020	0.0039 ± 0.0019	0.0721 ± 0.0059	0.0055 ± 0.0007	0.0403 ± 0.0067	8.8 ± 1.3	35.0 ± 9.1	100 ± 0.0	100 ± 0.0	35.0 ± 9.1
	RRWP	0.0038 ± 0.0019	0.0038 ± 0.0018	0.1085 ± 0.0205	0.0088 ± 0.0009	0.0359 ± 0.0043	8.5 ± 0.3	4.0 ± 2.0	100 ± 0.0	100 ± 0.0	4.0 ± 2.0
	RRWP - Dual	0.0036 ± 0.0019	0.0036 ± 0.0018	0.0592 ± 0.0027	0.0038 ± 0.0007	0.0027 ± 0.0009	1.7 ± 0.4	81.5 ± 3.2	100 ± 0.0	100 ± 0.0	81.5 ± 3.2
	MagLap	0.0038 ± 0.0019	0.0039 ± 0.0019	0.0186 ± 0.0046	0.0043 ± 0.0006	0.0543 ± 0.0072	11.3 ± 1.6	10.0 ± 4.2	100 ± 0.0	100 ± 0.0	10.0 ± 4.2
	MagLap - Dual	0.0035 ± 0.0018	0.0036 ± 0.0018	0.0670 ± 0.0032	0.0045 ± 0.0008	0.0033 ± 0.0013	1.9 ± 0.4	66.5 ± 4.1	100 ± 0.0	100 ± 0.0	66.5 ± 4.1

H.4 The role of positional encodings

To evaluate the importance of positional encodings in our model, we perform an ablation study comparing different options. Directed graphs lack a canonical node ordering, making positional

information crucial for capturing structural context. We test several variants, including no positional encoding, encodings that do not take into accounts directed information, and then different directed positional encodings. This analysis isolates how each encoding contributes to the model's ability to generate valid and semantically meaningful graphs. We report performance across synthetic datasets for both discrete diffusion and flow matching.

DIRECTO Table 12 presents an ablation study analyzing the impact of various positional encoding (PE) strategies on directed graph generation performance under the discrete flow matching framework. This evaluation highlights the critical role of positional information in enabling models to capture the asymmetric and hierarchical structure inherent to directed graphs.

Table 12: Directed Graph Generation performance across different positional encodings using discrete Flow Matching.

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
ER-DAG	No PE	0.0109 ± 0.0012	0.0110 ± 0.0012	0.0807 ± 0.0147	0.0073 ± 0.0007	0.0215 ± 0.0017	3.0 ± 0.2	47.0 ± 12.1	100 ± 0.0	100 ± 0.0	47.0 ± 12.1
	Lap	0.0119 ± 0.0020	0.0115 ± 0.0025	0.1272 ± 0.0224	0.0047 ± 0.0009	0.0048 ± 0.0005	1.8 ± 0.0	84.0 ± 4.9	100 ± 0.0	100 ± 0.0	84.0 ± 4.9
	RRWP	0.0107 ± 0.0019	0.0104 ± 0.0012	0.1209 ± 0.0194	0.0054 ± 0.0005	0.0043 ± 0.0008	1.7 ± 0.1	94.0 ± 1.0	100 ± 0.0	100 ± 0.0	94.0 ± 1.0
	MagLap	0.0110 ± 0.0017	0.0116 ± 0.0021	0.0509 ± 0.0017	0.0062 ± 0.0014	0.0027 ± 0.0005	1.3 ± 0.2	91.0 ± 2.5	100 ± 0.0	100 ± 0.0	91.0 ± 2.5
	MagLap ($Q = 5$)	0.0112 ± 0.0015	0.0107 ± 0.0022	0.0527 ± 0.0059	0.0056 ± 0.0009	0.0032 ± 0.0006	1.3 ± 0.2	91.5 ± 2.5	100 ± 0.0	100 ± 0.0	91.5 ± 2.5
	MagLap ($Q = 10$)	0.0117 ± 0.0014	0.0110 ± 0.0015	0.0711 ± 0.0120	0.0055 ± 0.0016	0.0026 ± 0.0004	1.3 ± 0.2	92.0 ± 3.7	100 ± 0.0	100 ± 0.0	92.0 ± 3.7
SBM	No PE	0.0038 ± 0.0011	0.0037 ± 0.0010	0.0638 ± 0.0059	0.0037 ± 0.0003	0.0127 ± 0.0018	3.5 ± 0.4	48.5 ± 6.4	100 ± 0.0	100 ± 0.0	48.5 ± 6.4
	Lap	0.0040 ± 0.0012	0.0038 ± 0.0011	0.0552 ± 0.0042	0.0048 ± 0.000g	0.0044 ± 0.0008	2.1 ± 0.3	71.5 ± 4.1	100 ± 0.0	100 ± 0.0	71.5 ± 4.1
	RRWP	0.0038 ± 0.0011	0.0037 ± 0.0012	0.0492 ± 0.0041	0.0035 ± 0.0007	0.0015 ± 0.0004	1.4 ± 0.2	87.0 ± 5.1	100 ± 0.0	100 ± 0.0	87.0 ± 5.1
	MagLap	0.0039 ± 0.0011	0.0036 ± 0.0010	0.0653 ± 0.0026	0.0033 ± 0.0003	0.0038 ± 0.0010	1.9 ± 0.3	77.0 ± 7.6	100 ± 0.0	100 ± 0.0	77.0 ± 7.6
	MagLap ($Q = 5$)	0.0039 ± 0.0010	0.0038 ± 0.0011	0.0748 ± 0.0028	0.0041 ± 0.0005	0.0070 ± 0.0015	2.6 ± 0.2	92.5 ± 2.2	100 ± 0.0	100 ± 0.0	92.5 ± 2.2
	MagLap ($Q = 10$)	0.0039 ± 0.0012	0.0038 ± 0.0010	0.0654 ± 0.0052	0.0038 ± 0.0003	0.0039 ± 0.0008	2.0 ± 0.3	96.5 ± 2.5	100 ± 0.0	100 ± 0.0	96.5 ± 2.5

Overall, the inclusion of any form of positional encoding significantly improves generation validity compared to the baseline with no PE. Among the methods tested, directed encodings like RRWP and magnetic Laplacian-based variants demonstrate particularly strong performance. These encodings consistently enhance graph validity and reduce MMD ratio in both datasets. Notably, the MagLap variant with multiple potentials ($Q = 10$) achieves the best results in terms of validity and structural fidelity, although resulting in a higher computational cost.

Interestingly, although classical Laplacian encodings (Lap) also play a role in increasing the V.U.N. ratio, they obtain worse results than the directed counterparts, emphasizing that for directed generation tasks, the choice of positional encoding is not merely a design detail but a relevant architectural decision that affects model success.

DIRECTO-DD Table 13 presents an ablation study assessing the influence of various positional encodings within the discrete diffusion framework across four synthetic datasets: SBM, ER-DAG, ER, and Price's model.

Table 13: Directed Graph Generation performance across different positional encodings using discrete diffusion.

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
ER-DAG	No PE	0.0149 ± 0.0037	0.0138 ± 0.0035	0.1256 ± 0.0088	0.0089 ± 0.0018	0.0106 ± 0.0014	2.6 ± 0.3	53.0 ± 2.9	100 ± 0.0	100 ± 0.0	53.0 ± 2.9
	Lap	0.0138 ± 0.0038	0.0144 ± 0.0042	0.0462 ± 0.0071	0.0055 ± 0.0011	0.0052 ± 0.0015	1.5 ± 0.3	67.0 ± 5.3	100 ± 0.0	100 ± 0.0	67.0 ± 5.3
	RRWP	0.0142 ± 0.0040	0.0129 ± 0.0034	0.0408 ± 0.0053	0.0055 ± 0.0010	0.0048 ± 0.0014	1.4 ± 0.3	79.0 ± 3.7	100 ± 0.0	100 ± 0.0	79.0 ± 3.7
	MagLap	0.0142 ± 0.0044	0.0135 ± 0.0034	0.0438 ± 0.0071	0.0057 ± 0.0013	0.0046 ± 0.0010	1.4 ± 0.3	69.0 ± 6.4	100 ± 0.0	100 ± 0.0	69.0 ± 6.4
	MagLap ($Q = 5$)	0.0135 ± 0.0032	0.0140 ± 0.0045	0.0596 ± 0.0081	0.0061 ± 0.0007	0.0037 ± 0.0017	1.4 ± 0.2	78.5 ± 2.5	100 ± 0.0	100 ± 0.0	78.5 ± 2.5
	MagLap ($Q = 10$)	0.0145 ± 0.0040	0.0134 ± 0.0033	0.582 ± 0.0083	0.0063 ± 0.0015	0.0034 ± 0.0011	1.5 ± 0.2	85.0 ± 9.2	100 ± 0.0	100 ± 0.0	85.0 ± 9.2
SBM	No PE	0.0037 ± 0.0020	0.0039 ± 0.0019	0.0721 ± 0.0059	0.0055 ± 0.0007	0.0403 ± 0.0067	8.8 ± 1.3	35.0 ± 9.1	100 ± 0.0	100 ± 0.0	35.0 ± 9.1
	Lap	0.0037 ± 0.0020	0.0038 ± 0.0017	0.0531 ± 0.0042	0.0038 ± 0.0006	0.0106 ± 0.0009	1.4 ± 0.4	81.0 ± 3.7	100 ± 0.0	100 ± 0.0	81.0 ± 3.7
	RRWP	0.0036 ± 0.0019	0.0036 ± 0.0018	0.0592 ± 0.0027	0.0038 ± 0.0007	0.0027 ± 0.0009	1.7 ± 0.4	81.5 ± 3.2	100 ± 0.0	100 ± 0.0	81.5 ± 3.2
	MagLap	0.0035 ± 0.0018	0.0036 ± 0.0018	0.0670 ± 0.0032	0.0045 ± 0.0008	0.0033 ± 0.0013	1.9 ± 0.4	66.5 ± 4.1	100 ± 0.0	100 ± 0.0	66.5 ± 4.1
	MagLap ($Q = 5$)	0.0039 ± 0.0021	0.0038 ± 0.0018	0.0504 ± 0.0011	0.0047 ± 0.0006	0.0065 ± 0.0018	2.4 ± 0.4	92.0 ± 1.9	100 ± 0.0	100 ± 0.0	92.0 ± 1.9
	MagLap ($Q = 10$)	0.0037 ± 0.0019	0.0038 ± 0.0018	0.0450 ± 0.0037	0.0038 ± 0.0006	0.0021 ± 0.0009	1.5 ± 0.4	95.5 ± 3.7	100 ± 0.0	100 ± 0.0	95.5 ± 3.7
ER	No PE	0.0039 ± 0.0022	0.0038 ± 0.0023	0.0486 ± 0.0039	0.0039 ± 0.0009	0.0030 ± 0.0019	2.8 ± 1.4	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
	Lap	0.0022 ± 0.0014	0.0021 ± 0.0014	0.0479 ± 0.0058	0.0037 ± 0.0013	0.0017 ± 0.0010	1.8 ± 0.9	99.5 ± 1.0	100 ± 0.0	100 ± 0.0	99.5 ± 1.0
	RRWP	0.0021 ± 0.0013	0.0021 ± 0.0013	0.0505 ± 0.0037	0.0033 ± 0.0013	0.0018 ± 0.0010	1.8 ± 0.8	99.5 ± 1.0	100 ± 0.0	100 ± 0.0	99.5 ± 1.0
	MagLap	0.0021 ± 0.0013	0.0021 ± 0.0013	0.0515 ± 0.0055	0.0029 ± 0.0004	0.0019 ± 0.0011	1.8 ± 0.8	99.0 ± 1.2	100 ± 0.0	100 ± 0.0	99.0 ± 1.2
	MagLap ($Q = 5$)	0.0022 ± 0.0014	0.0021 ± 0.0014	0.0456 ± 0.0068	0.0029 ± 0.0006	0.0017 ± 0.0010	1.8 ± 0.9	100 ± 0.0	100 ± 0.0	100 ± 0.0	100 ± 0.0
Price	No PE	0.0149 ± 0.0020	0.0011 ± 0.0002	0.0524 ± 0.0030	0.0041 ± 0.0004	0.0017 ± 0.0000	2.9 ± 0.1	92.5 ± 1.6	100 ± 0.0	100 ± 0.0	92.5 ± 1.6
	Lap	0.0219 ± 0.0030	0.0010 ± 0.0002	0.0725 ± 0.0016	0.0476 ± 0.0034	0.0124 ± 0.0020	16.2 ± 2.0	99.0 ± 1.2	100 ± 0.0	100 ± 0.0	99.0 ± 1.2
	RRWP	0.0231 ± 0.0034	0.0010 ± 0.0002	0.0837 ± 0.0022	0.0619 ± 0.0048	0.0123 ± 0.0027	17.0 ± 2.8	99.0 ± 1.2	100 ± 0.0	100 ± 0.0	99.0 ± 1.2
	MagLap	0.0232 ± 0.0013	0.0012 ± 0.0001	0.0752 ± 0.0028	0.0531 ± 0.0059	0.0126 ± 0.0019	16.8 ± 2.1	97.5 ± 2.2	100 ± 0.0	100 ± 0.0	97.5 ± 2.2
	MagLap ($Q = 5$)	0.0191 ± 0.0031	0.0010 ± 0.0001	0.0653 ± 0.0118	0.0239 ± 0.0023	0.0044 ± 0.0016	6.8 ± 1.7	99.5 ± 1.0	100 ± 0.0	100 ± 0.0	99.5 ± 1.0

We observe that that ER and Price graphs, due to their simplicity and highly regular generative processes, pose a relatively easy modeling challenge. As a result, performance is consistently high across all variants, including the baseline with no positional encoding. In contrast, the SBM and ER-DAG datasets present more intricate structural patterns, which amplify the benefits of informed

positional encodings. In these cases, clear performance gaps emerge between the baseline, non-directed encodings (e.g., Lap), and directed-aware methods such as RRWP and MagLapPE.

These differences highlight the importance of encoding directionality to faithfully capture the distribution of more structurally diverse directed graphs. For this reason, in the main paper we focus our synthetic evaluations on SBM and ER-DAG, where the modeling challenge is sufficiently rich to reveal the comparative strengths of different modeling strategies.

H.5 ER vs DAG performance

To evaluate the structural quality of generated synthetic directed acyclic graphs, we defined a composite validity metric that captures two key properties: (i) the percentage of generated graphs that are Directed Acyclic Graphs (DAGs), and (ii) the percentage that follow the target structural distribution (e.g. Erdős-Renyi). The validity score is computed as the proportion of generated graphs that satisfy both criteria. For completeness, we also report each component individually to disentangle the contributions of acyclicity and distributional alignment.

Table 14: ER vs DAG accuracy

Model	% DAG	% ER	Valid
Training set	100	99.2	99.2
MLE	0.0 ± 0.0	97.0 ± 2.1	0.0 ± 0.0
D-VAE	100 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
LayerDAG	100 ± 0.0	21.5 ± 2.7	21.5 ± 2.7
DiGress	35.0 ± 4.2	98.5 ± 2.0	34.0 ± 4.1
DeFoG	21.5 ± 2.7	100 ± 0.0	21.5 ± 2.7
Directo-DD RRWP	79.0 ± 3.7	100 ± 0.0	79.0 ± 3.7
Directo-DD MagLap ($Q = 10$)	86.0 ± 9.4	99 ± 1.2	85.0 ± 10.2
Directo RRWP	94.0 ± 1.0	100 ± 0.0	94.0 ± 1.0
Directo MagLap ($Q = 10$)	99.5 ± 1.0	92.5 ± 2.7	92.0 ± 3.7

As shown in Table 14, baselines such as MLE and LayerDAG exhibit a stark imbalance: while LayerDAG guarantees acyclicity, both fail to model the target ER distribution, resulting in low overall validity. On the other hand, MLE closely aligns with the distribution but fails to generate acyclic graphs.

In contrast, diffusion- and flow matching–based methods paired with dual attention and relevant directed positional encodings achieve strong performance on both fronts, demonstrating their ability to generate structurally valid DAGs that also align with the underlying data distribution. This highlights the limitations of purely autoregressive or heuristic DAG-specific approaches when used in isolation.

H.6 Scalability experiments

In this section, we investigate the scalability of our method with respect to three key axes: dataset size, parameter efficiency, and graph size. These experiments are designed to disentangle the contributions of data availability, architectural choices, and input complexity to the overall generative performance, providing a clearer understanding of the trade-offs between accuracy, efficiency, and computational cost in our framework.

Effect of dataset size To evaluate the effect on the number of graphs available at training time, we conduct an ablation study using the synthetic ER-DAG dataset in two configurations: the standard size and a variant with 10× more training, test, and validation samples. This setup allows us to assess whether the method encodings benefit from increased data. We compare the different positional encodings for discrete diffusion using dual attention, with results in Table 15.

Table 15: Effect of the dataset size (using Directo-DD).

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
Standard	Train set	0.0113	0.0103	0.0355	0.0038	0.0024	1.0	99.2	100	0.0	0.0
	Lap	0.0138 ± 0.0038	0.0144 ± 0.0042	0.0462 ± 0.0071	0.0055 ± 0.0011	0.0052 ± 0.0015	1.5 ± 0.3	67.0 ± 5.3	100 ± 0.0	100 ± 0.0	67.0 ± 5.3
	RRWP	0.0142 ± 0.0040	0.0129 ± 0.0034	0.0408 ± 0.0053	0.0055 ± 0.0010	0.0048 ± 0.0014	1.4 ± 0.3	79.0 ± 3.7	100 ± 0.0	100 ± 0.0	79.0 ± 3.7
	MagLap	0.0142 ± 0.0044	0.0135 ± 0.0034	0.0438 ± 0.0071	0.0057 ± 0.0013	0.0046 ± 0.0010	1.4 ± 0.3	69.0 ± 6.4	100 ± 0.0	100 ± 0.0	69.0 ± 6.4
	MagLap ($Q = 5$)	0.0135 ± 0.0032	0.0140 ± 0.0045	0.0596 ± 0.0081	0.0061 ± 0.0007	0.0037 ± 0.0017	1.4 ± 0.2	78.5 ± 2.5	100 ± 0.0	100 ± 0.0	78.5 ± 2.5
x 10	Train set	0.0002	0.0003	0.0021	0.0004	0.0000	1.0	99.9	100	0.0	0.0
	Lap	0.00035 ± 0.0007	0.0032 ± 0.0008	0.0261 ± 0.0025	0.0024 ± 0.0003	0.0006 ± 0.0001	10.5 ± 1.3	85.5 ± 2.4	100 ± 0.0	100 ± 0.0	85.5 ± 2.4
	RRWP	0.0030 ± 0.0008	0.0031 ± 0.0006	0.0277 ± 0.0075	0.0024 ± 0.0004	0.0006 ± 0.0002	10.1 ± 2.2	94.0 ± 4.4	100 ± 0.0	100 ± 0.0	94.0 ± 4.4
	MagLap	0.0030 ± 0.0006	0.0033 ± 0.0005	0.0305 ± 0.0051	0.0025 ± 0.0005	0.0007 ± 0.0003	14.5 ± 2.4	89.5 ± 3.7	100 ± 0.0	100 ± 0.0	89.5 ± 3.7
	MagLap ($Q = 5$)	0.0031 ± 0.0008	0.0028 ± 0.0006	0.0237 ± 0.0028	0.0022 ± 0.0006	0.0008 ± 0.0001	9.6 ± 1.6	93.5 ± 2.5	100 ± 0.0	100 ± 0.0	93.5 ± 2.5

Overall, we see that increasing the dataset size significantly improves generation performance across all positional encoding variants, confirming that additional training data helps models better capture structural patterns. However, this performance gain comes at the cost of substantially longer training times and greater computational demand ($\sim$12h for the standard dataset vs $\sim$32h for the bigger version). These trade-offs motivate our focus on architectural innovations such as employing discrete flow matching as more efficient alternatives to scaling purely through data and compute.

Effect of model size (parameter efficiency) We next evaluate parameter efficiency by contrasting our dual-attention mechanism with an alternative approach that scales model capacity by doubling the number of parameters. Dual attention increases the parameter space by introducing two adjacency-based attention heads, each specialized for handling edge directionality. To ablate for model size, we compare against a baseline in which the architecture is widened to match the parameter count, but without dual-attention.

Table 16: Directed Graph Generation performance for the dual attention architecture versus doubling the depth of the network without using the dual attention mechanism.

Dataset	Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
ER-DAG	RRWP - Double	0.0129 ± 0.0020	0.0130 ± 0.0015	0.1741 ± 0.0200	0.0107 ± 0.0007	0.0020 ± 0.0005	4.9 ± 0.2	72.0 ± 9.8	100 ± 0.0	100 ± 0.0	72.0 ± 9.8
	RRWP - Dual	0.0107 ± 0.0019	0.0104 ± 0.0012	0.1209 ± 0.0194	0.0054 ± 0.0005	0.0043 ± 0.0008	**1.7 ± 0.1**	94.0 ± 1.0	100 ± 0.0	100 ± 0.0	**94.0 ± 1.0**
	MagLap - Double	0.0131 ± 0.0019	0.0118 ± 0.0022	0.1237 ± 0.0217	0.0095 ± 0.0030	0.0020 ± 0.0008	4.3 ± 0.7	80.0 ± 6.3	100 ± 0.0	100 ± 0.0	80.0 ± 6.3
	MagLap - Dual	0.0110 ± 0.0017	0.0116 ± 0.0021	0.0509 ± 0.0017	0.0062 ± 0.0014	0.0027 ± 0.0005	**1.3 ± 0.2**	91.0 ± 2.5	100 ± 0.0	100 ± 0.0	**91.0 ± 2.5**
SBM	RRWP - Double	0.0083 ± 0.0024	0.0081 ± 0.0025	0.1789 ± 0.0107	0.0084 ± 0.0014	0.0084 ± 0.0067	27.1 ± 2.3	0.0 ± 0.0	100 ± 0.0	100 ± 0.0	0.0 ± 0.0
	RRWP - Dual	0.0038 ± 0.0011	0.0037 ± 0.0012	0.0492 ± 0.0041	0.0035 ± 0.0007	0.0015 ± 0.0004	**1.4 ± 0.2**	87.0 ± 5.1	100 ± 0.0	100 ± 0.0	**87.0 ± 5.1**
	MagLap - Double	0.0081 ± 0.0025	0.0079 ± 0.0020	0.1799 ± 0.0097	0.0307 ± 0.0052	0.0723 ± 0.0156	24.8 ± 4.6	6.0 ± 4.9	100 ± 0.0	100 ± 0.0	6.0 ± 4.9
	MagLap - Dual	0.0039 ± 0.0011	0.0036 ± 0.0010	0.0653 ± 0.0026	0.0033 ± 0.0003	0.0038 ± 0.0010	**1.9 ± 0.3**	77.0 ± 7.6	100 ± 0.0	100 ± 0.0	**77.0 ± 7.6**

As shown in Table 16, this baseline fails to recover the correct edge distribution, particularly in more challenging regimes such as SBM graphs. In contrast, the dual-attention model achieves substantially higher fidelity, demonstrating that our architecture leverages parameters more effectively by embedding structural inductive bias rather than merely increasing model scale.

Effect of node size We finally investigate how the method scales with increasing graph size by conducting experiments on ER-DAG datasets ranging from 80–150 up to 200–250 nodes (see Table 17 for dataset statistics). Due to the poor scalability of MultMagLap encodings, we restrict this ablation to RRWP and MagLap positional encodings, with the largest graphs (200–250 nodes) evaluated only under RRWP. Training and sampling times, together with generation quality, are reported in Table 18. Importantly, the computational overhead introduced by dual attention remains bounded by a constant factor of two and does not affect the overall asymptotic complexity, which matches that of standard graph transformers.

Table 17: Dataset statistics for the synthetic experiments with larger graph size (nodes and edges).

Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges
80	150	117	1866	6699	4170
150	200	173	6643	11851	9046
200	250	225	11982	18645	15148

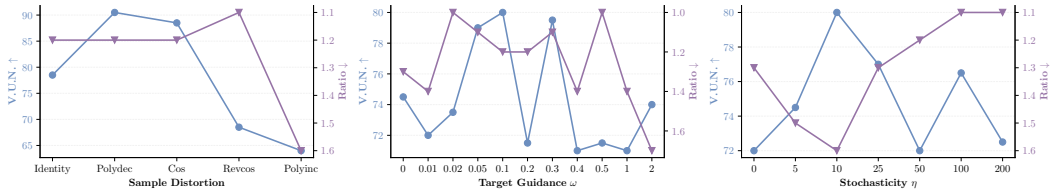
In terms of positional encodings, MagLap exhibits clear scalability limitations, both in runtime and performance, whereas RRWP features scale considerably better, consistent with prior findings (e.g., Appendix G.4 of DeFoG [52]). Regarding generative performance, DIRECTO demonstrates strong distributional fidelity across node sizes, as reflected in the Ratio and distributional validity scores. However, ensuring strict acyclicity becomes increasingly challenging at scale: a single misplaced edge can violate the constraint entirely, and the risk grows quadratically with the number of nodes. This highlights the inherent difficulty of enforcing global constraints in large directed graphs, motivating future work on more robust inductive biases for acyclicity preservation.

Table 18: Scaling performance of DIRECTO on ER-DAG datasets of increasing node size.

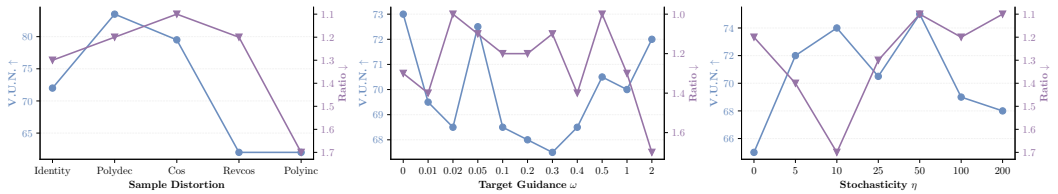
Model	Out-degree ↓	In-degree ↓	Clustering ↓	Spectre ↓	Wavelet ↓	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑	Train (h)	Min / sample
80-150 (RRWP)	0.0045 ± 0.0001	0.0044 ± 0.0008	0.3233 ± 0.0676	0.0010 ± 0.0003	0.0003 ± 0.0001	3.7 ± 0.5	53.5 ± 3.0	100 ± 0.0	100 ± 0.0	53.5 ± 3.0	22	0.285
80-150 (MagLap)	0.0040 ± 0.0005	0.0044 ± 0.0005	0.3382 ± 0.0361	0.0015 ± 0.0002	0.0007 ± 0.0002	4.2 ± 0.3	45.0 ± 7.1	100 ± 0.0	100 ± 0.0	45.0 ± 7.1	26	0.940
150-200 (RRWP)	0.0034 ± 0.0002	0.0031 ± 0.0002	0.1570 ± 0.0496	0.003 ± 0.0000	0.0000 ± 0.0000	1.7 ± 0.3	60.0 ± 5.9	100 ± 0.0	100 ± 0.0	60.0 ± 5.9	28	0.555
150-200 (MagLap)	0.0031 ± 0.0005	0.0031 ± 0.0004	0.6691 ± 0.0558	0.0004 ± 0.0000	0.0001 ± 0.0001	3.2 ± 0.4	34.5 ± 1.9	100 ± 0.0	100 ± 0.0	34.5 ± 1.9	38	1.365
200-250 (RRWP)	0.0034 ± 0.0002	0.0033 ± 0.0003	0.0411 ± 0.0178	0.0003 ± 0.0000	0.0000 ± 0.0000	1.9 ± 0.3	2.0 ± 2.4	100 ± 0.0	100 ± 0.0	2.0 ± 2.4	40	0.645

H.7 Impact of sampling optimization using discrete flow matching

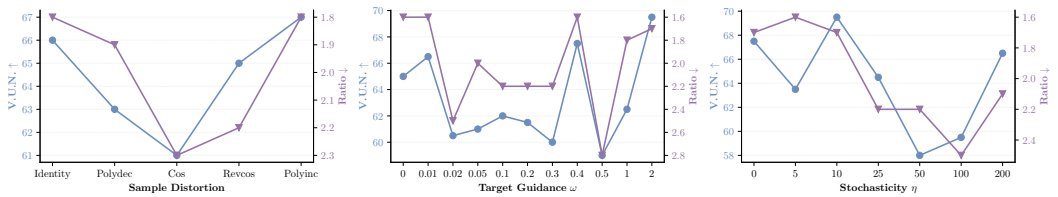
To better understand the influence of the three sampling hyperparameters discussed in Appendix C: time distortion, stochasticity coefficient (η), and target guidance factor (ω) on generative performance, we analyze their effect on the V.U.N. and the structural ratio. Specifically, we present performance curves for the four primary datasets reported in the main results table: SBM, RRWP, TPU Tiles, and Visual Genome. For each dataset, we evaluate the impact of these parameters under two positional encoding strategies: RRWP and MagLap. We report the results for 100 sampling steps and 5 different runs (except for TPU Tiles where only 1 run was performed due to computational constraints).



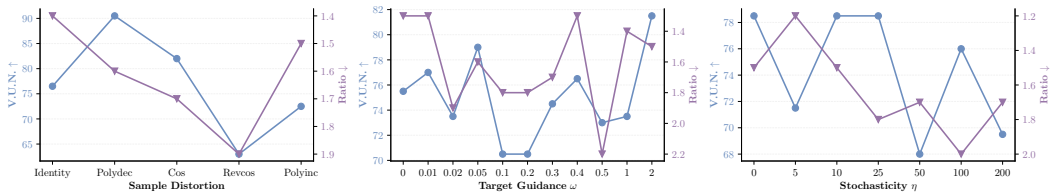
(a) ER-DAG dataset with RRWP positional encoding



(b) ER-DAG dataset with MagLap ($Q = 10$) positional encoding



(c) SBM dataset with RRWP positional encoding

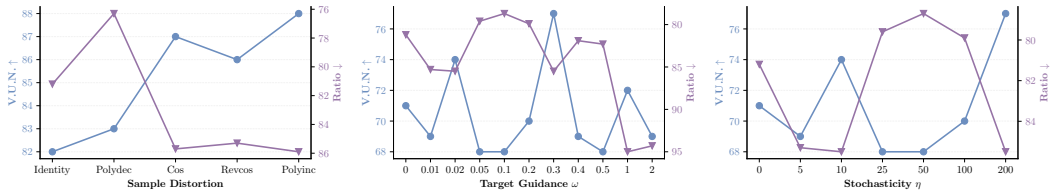


(d) SBM dataset with MagLap ($Q = 10$) positional encoding

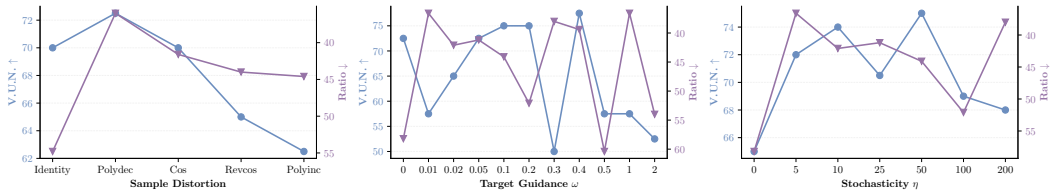
Figure 4: Sampling optimization curves for the synthetic datasets with 100 sampling steps and 5 sampling runs. We represent V.U.N. (blue) and MMD ratio (purple) and optimize for best trade-off for each of the three parameters individually.

The results for the synthetic datasets in Figure 4 illustrate that the sampling hyperparameters meaningfully influence the generative performance. We observe that different values can lead to notable variations in both V.U.N. and the structural ratio. However, a higher V.U.N. does not necessarily correspond to a lower ratio. While certain configurations tend to perform well across both RRWP and MagLap positional encodings, the improvements are generally modest, suggesting that optimal settings may vary slightly depending on the dataset and positional encoding strategy used.

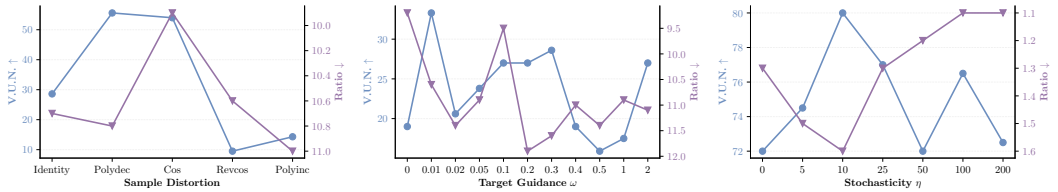
A similar pattern is observed in the results for the real-world datasets (TPU Tiles and Visual Genome), as shown in Figure 5. The sampling hyperparameters continue to affect generation performance, and there is no clear one-to-one relationship between V.U.N. and ratio. As with the synthetic datasets, some hyperparameter combinations show slightly more consistent behavior across positional encodings, but overall, the best hyperparameters combination remain dependent on the dataset and positional encoding used.



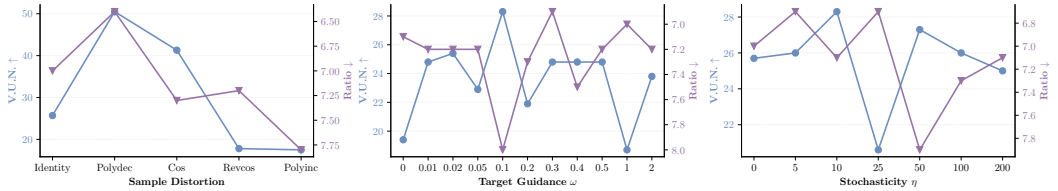
(a) TPU Tiles dataset with RRWP positional encoding



(b) TPU Tiles dataset with MagLap ($Q = 10$) positional encoding



(c) Visual Genome dataset with RRWP positional encoding



(d) Visual Genome dataset with MagLap ($Q = 10$) positional encoding

Figure 5: Sampling optimization curves for the synthetic datasets with 100 sampling steps and 5 sampling runs. We represent V.U.N. (blue) and MMD ratio (purple) and optimize for best trade-off for each of the three parameters individually.

I Iterative Refinement Methods for Graph Generation

I.1 Graph iterative refinement methods

Graph generative models aim to learn the underlying probability distribution over graphs that produced a given dataset, enabling the generation of new samples that preserve the structural patterns and properties of the training graphs.

Within this field, discrete state-space iterative refinement approaches [63, 70, 57, 52] have emerged as a powerful framework for capturing the intricate dependencies that govern target graph distributions. These methods achieve state-of-the-art performance by naturally aligning with the structure of graphs: they operate directly in discrete state spaces, matching the inherent discreteness of adjacency matrices, and respect node permutation equivariances due to their one-shot prediction formulation. These characteristics make iterative refinement approaches particularly well-suited for digraph generation.

These methods comprise two main processes: a noising process and a denoising process. For consistency, we define $t = 0$ as corresponding to fully noised graphs and $t = 1$ to clean graphs. The noising process runs from $t = 1$ to $t = 0$, progressively corrupting the original graphs, while the denoising process learns how to reverse this trajectory, running from $t = 0$ to $t = 1$.

Denoising is performed using a neural network parametrized by θ,

$$\mathbf{p}^{\theta}_{1|t}(\cdot|G_t) = \left(\left(p^{\theta,(n)}_{1|t}(x^{(n)}_1 \mid G_t) \right)_{1 \le n \le N}, \left(p^{\theta,(i,j)}_{1|t}(e^{(i,j)}_1 \mid G_t) \right)_{1 \le i \ne j \le N} \right), \tag{37}$$

which predicts categorical distributions over nodes and edges given a noised graph G_t. In practice, $\mathbf{p}^{\theta}_{1|t}(\cdot|G_t)$ is parameterized using a graph transformer such as [15], which allows for expressive modeling of complex graph structures. The network is trained using a cross-entropy loss applied independently to each node and each edge:

$$\mathcal{L} = \mathbb{E}_{t,G_1,G_t}\left[-\sum_n \log\left(p^{\theta,(n)}_{1|t}\left(x_1^{(n)} \mid G_t\right)\right) - \lambda \sum_{i\neq j} \log\left(p^{\theta,(i,j)}_{1|t}\left(e_1^{(i,j)} \mid G_t\right)\right)\right], \tag{38}$$

where the expectation is taken over time t sampled from a predefined distribution over $[0,1]$ (e.g., uniform); $G_1 \sim p_1(G_1)$ is a clean graph from the dataset; and $G_t \sim p_t(G_t|G_1)$ is its noised version at time t. The hyperparameter $\lambda \in \mathbb{R}^+$ controls the relative weighting between node and edge reconstruction losses. Once the denoising network is trained, different graph iterative refinement methods vary in how they leverage the predicted $\mathbf{p}^{\theta}_{1|t}(\cdot|G_t)$ to progressively recover the clean graphs.

I.2 Discrete flow matching for graph generation

Among graph iterative refinement methods, Discrete Flow Matching (DFM) [8, 18] has recently emerged as a particularly powerful framework. By decoupling the training and sampling processes, DFM enables a broader and more flexible design space, which has been shown to improve generative performance. This framework has proven particularly effective for undirected graph generation [52].

The noising process in DFM is defined as a linear interpolation between the data distribution and a pre-specified noise distribution. This interpolation is applied independently to each variable, corresponding to each node and each edge in the case of graphs:

$$p^X_{t|1}(x_t^{(n)} \mid x_1^{(n)}) = t\,\delta(x_t^{(n)}, x_1^{(n)}) + (1-t)\,p^X_{\text{noise}}(x_t^{(n)}), \tag{39}$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta and p^X_{noise} is a reference distribution over node categories. A similar construction is used for edges. Therefore, the complete noising process corresponds to independently noising each node and edge through $p^X_{t|1}$ and $p^E_{t|1}$, respectively.

The denoising process aims to reverse the noising trajectory, running from $t=0$ to $t=1$. It leverages a denoising neural network, parametrized by θ, which is trained to predict the clean node $\boldsymbol{p}^{\theta,(n)}_{1|t}$ and edge $\boldsymbol{p}^{\theta,(i,j)}_{1|t}$ distributions given a noisy graph, aggregated as:

$$\boldsymbol{p}^{\theta}_{1|t}(\cdot|G_t) = \left(\left(\boldsymbol{p}^{\theta,(n)}_{1|t}(x_1^{(n)} \mid G_t)\right)_{1\leq n\leq N}, \left(\boldsymbol{p}^{\theta,(i,j)}_{1|t}(e_1^{(i,j)} \mid G_t)\right)_{1\leq i\neq j\leq N}\right). \tag{40}$$

This network is trained using a cross-entropy loss independently to each node and each edge:

$$\mathcal{L} = \mathbb{E}_{t,G_1,G_t}\left[-\sum_n \log\left(p^{\theta,(n)}_{1|t}\left(x_1^{(n)} \mid G_t\right)\right) - \lambda \sum_{i\neq j} \log\left(p^{\theta,(i,j)}_{1|t}\left(e_1^{(i,j)} \mid G_t\right)\right)\right], \tag{41}$$

where the expectation is taken over time t, sampled from a predefined distribution over $[0,1]$ (e.g., uniform); $G_1 \sim \boldsymbol{p}_1(G_1)$ is a clean graph from the dataset; and $G_t \sim \boldsymbol{p}_t(G_t|G_1)$ is its noised version at time t [52]. The hyperparameter $\lambda \in \mathbb{R}^+$ controls the relative weighting between node and edge reconstruction losses.

Then, DFM models the denoising process as a Continuous-Time Markov Chain (CTMC). Starting from an initial distribution $\boldsymbol{p}_0$, the generative process evolves according to:

$$\boldsymbol{p}_{t+\Delta t|t}(G_{t+\Delta t} \mid G_t) = \boldsymbol{\delta}(G_t, G_{t+\Delta t}) + \boldsymbol{R}_t(G_t, G_{t+\Delta t})\,\mathrm{d}t, \tag{42}$$

where $\boldsymbol{R}_t$ is the CTMC rate matrix. In practice, this update is approximated over a finite interval Δt, in an Euler method step, with the rate matrix estimated from the network predictions $\mathbf{p}^{\theta}_{1|t}(\cdot \mid G_t)$ via:

$$\boldsymbol{R}^{\theta}_t(G_t, G_{t+\Delta t}) = \sum_n \delta(G_t^{\backslash(n)}, G_{t+\Delta t}^{\backslash(n)})\,\mathbb{E}_{p^{\theta,(n)}_{1|t}(x_1^{(n)}|G_t)}\left[R_t^{(n)}(x_t^{(n)}, x_{t+\Delta t}^{(n)} \mid x_1^{(n)})\right] \tag{43}$$

$$+ \sum_{i\neq j} \delta(G_t^{\backslash(i,j)}, G_{t+\Delta t}^{\backslash(i,j)})\,\mathbb{E}_{p^{\theta,(i,j)}_{1|t}(e_1^{(i,j)}|G_t)}\left[R_t^{(i,j)}(e_t^{(i,j)}, e_{t+\Delta t}^{(i,j)} \mid e_1^{(i,j)})\right], \tag{44}$$

where the $G^{\setminus(d)}$ denote the full graph G except variable d (node or edge). The Knonecker deltas ensure that each variable-specific rate matrix, $R_t^{(n)}$ or $R_t^{(i,j)}$, is applied independently at each node and edge, respectively. Finally, each node-specific rate matrix is defined as:

$$R_t(x_t^{(n)}, x_{t+\Delta t}^{(n)} \mid x_1^{(n)}) = \frac{\text{ReLU}\left[\partial_t p_{t|1}(x_{t+\Delta t}^{(n)} \mid x_1^{(n)}) - \partial_t p_{t|1}(x_t^{(n)} \mid x_1^{(n)})\right]}{X_t^{>0}\; p_{t|1}(x_t^{(n)} \mid x_1^{(n)})}, \tag{45}$$

for each node $x^{(n)}$, where $X_t^{>0} = \left|\left\{x_t^{(n)} : p_{t|1}(x_t^{(n)} \mid x_1^{(n)}) > 0\right\}\right|$. An analogous definition applies for edge rate matrices.

I.3 Discrete diffusion for graph generation

Among discrete diffusion-based models for undirected graphs, DIGRESS [63], grounded in the structured discrete diffusion framework [3], has been particularly influential. This approach mostly differs from the DFM-based formulation described in Appendix I.2, as it operates in discrete time, with the denoising process modeled as a Discrete-Time Markov Chain (DTMC), in contrast to the continuous-time formulation used in DFM.

The forward, or noising, process is modeled as a Markov noise process q, which generates a sequence of progressively noised graphs G_t, for $t = 1, \dots, T$. At each timestep, node and edge tensors are perturbed using categorical transition matrices $[\mathbf{Q}_t^X]^{(i,j)} = q(x_t = j \mid x_{t-1} = i)$ and $[\mathbf{Q}_t^E]^{(i,j)} = q(e_t = j \mid e_{t-1} = i)$, respectively. This process induces structural changes such as edge addition or deletion, and edits to node and edge categories. The transition probabilities can be summarized as:

$$q(G_t \mid G_{t-1}) = \left(\mathbf{X}_{t-1}\mathbf{Q}_t^X,\, \mathbf{E}_{t-1}\mathbf{Q}_t^E\right) \quad \text{and} \quad q(G_t \mid G) = \left(\mathbf{X}\prod_{i=1}^{t}\mathbf{Q}_i^X,\, \mathbf{E}\prod_{i=1}^{t}\mathbf{Q}_i^E\right). \tag{46}$$

In practice, these noise matrices are implemented based on the *marginal* noise model, defined as:

$$\mathbf{Q}_t^X = \alpha^t\mathbf{I} + (1-\alpha^t)\mathbf{1}_X\mathbf{m}_X^\top \quad \text{and} \quad \mathbf{Q}_t^E = \alpha^t\mathbf{I} + (1-\alpha^t)\mathbf{1}_E\mathbf{m}_E^\top,$$

where α^t transitions from 1 to 0 with t according to the popular cosine scheduling [46]. The vectors $\mathbf{1}_X \in \{1\}^X$ and $\mathbf{1}_E \in \{1\}^{E+1}$ are filled with ones, and $\mathbf{m}_X \in \Delta^X$ and $\mathbf{m}_E \in \Delta^{E+1}$ are vectors filled with the marginal node and edge distributions, respectively[3].

This noising process can be rewritten using the analogous notation to the one used for DFM in Equation (47) as:

$$p_{t'|1}^X(x_{t'}^{(n)} \mid x_1^{(n)}) = \bar{\alpha}(t')\,\delta(x_{t'}^{(n)}, x_1^{(n)}) + (1-\bar{\alpha}(t'))\,p_{\text{noise}}^X(x_{t'}^{(n)}), \tag{47}$$

with $t' = 1 - \frac{t}{T}$ and $\bar{\alpha}(t') = \cos\left(\frac{\pi}{2}\frac{1-t'+s}{1+s}\right)^2$, with a small s. Importantly, this function is only evaluated in the discrete values of time considered for the DTMC associated to the diffusion model. For the remaining of this section, we consider the transformed variable t' as the reference time variable.

In the reverse, or denoising, process, a clean graph is progressively built leveraging the denoising neural network predictions $\boldsymbol{p}_{1|t'}^\theta(\cdot|G_t')$, analogously to the DFM setting. In particular, the model begins from a noise sample G_0 and iteratively predicts the clean graph G_1 by modeling node and edge distributions conditioned on the full graph structure. The reverse transition is defined as:

$$p_{t'+\Delta t'|t'}^\theta(G_{t'+\Delta t'} \mid G_{t'}) = \prod_{i=1}^{N} p_{t'+\Delta t'|t'}^\theta(x_{t'+\Delta t'}^{(n)} \mid G_{t'}) \prod_{1\le i<j\le N} p_{t'+\Delta t'|t'}^\theta(e_{t'+\Delta t'}^{(i,j)} \mid G_{t'}). \tag{48}$$

with each of the node and edge denoising terms are computer through the following marginalization:

$$\boldsymbol{p}_{t'+\Delta t'|t'}^\theta(x_{t'+\Delta t'}^{(n)} \mid G_{t'}) = \sum_{x_1^{(n)} \in \{1,\dots,X\}} \boldsymbol{p}_{t'+\Delta t'|1,t'}(x_{t'+\Delta t'}^{(n)}|x_1^{(n)}, G_{t'})\; p_{1|t'}^{\theta,(n)}(x_1^{(n)}|G_{t'}), \tag{49}$$

[3]We denote the probability simplex of the state-space of cardinality Z by Δ^Z

and similarly for edges. To compute the missing posterior term in Equation (49), $p_{t'+\Delta t'|1,t'}(x^{(n)}_{t'+\Delta t'}|x^{(n)}_1, G_{t'})$, we equate it to the posterior term of the forward process:

$$\boldsymbol{p}_{t'+\Delta t'|1,t'}(x^{(n)}_{t'+\Delta t'}|x^{(n)}_1, G_{t'}) = \begin{cases} \frac{\mathbf{x}^{(n)}_{t'}(\mathbf{Q}^X_{t'})^\top \odot \mathbf{x}^{(n)}_1 \bar{\mathbf{Q}}^X_{t'+\Delta t'}}{\mathbf{x}^{(n)}_{t'} \bar{\mathbf{Q}}^X_{t'} \mathbf{x}^{(n)}_1} & \text{if } q(x^{(n)}_{t'}|x^{(n)}_1) > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{50}$$

where $\mathbf{x}^{(n)}_{t'}$ and $\mathbf{x}^{(n)}_1$ denote the vectorized versions of $x^{(n)}_{t'}$ and $x^{(n)}_1$, respectively. Crucially, and contrarily to DFM, the sampling strategy with discrete diffusion is fixed at training time, which restricts the design space of this generative framework.

J Visualizations

In this section, we present visualizations of the digraphs generated with DIRECTO for the different synthetic and real-world datasets and combinations of positional encodings reported in the main paper. To allow for a fair comparison between original and generated digraphs, we additionally report digraphs from the train splits of the original datasets.

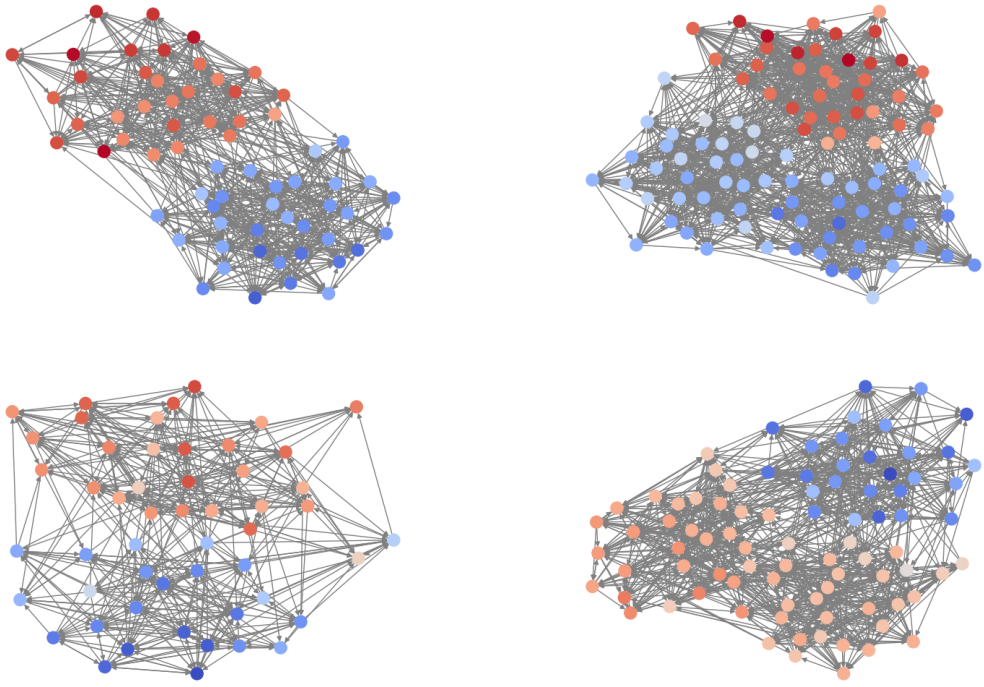


Figure 6: Visualizations of original synthetic digraphs from the train splits of the SBM dataset.

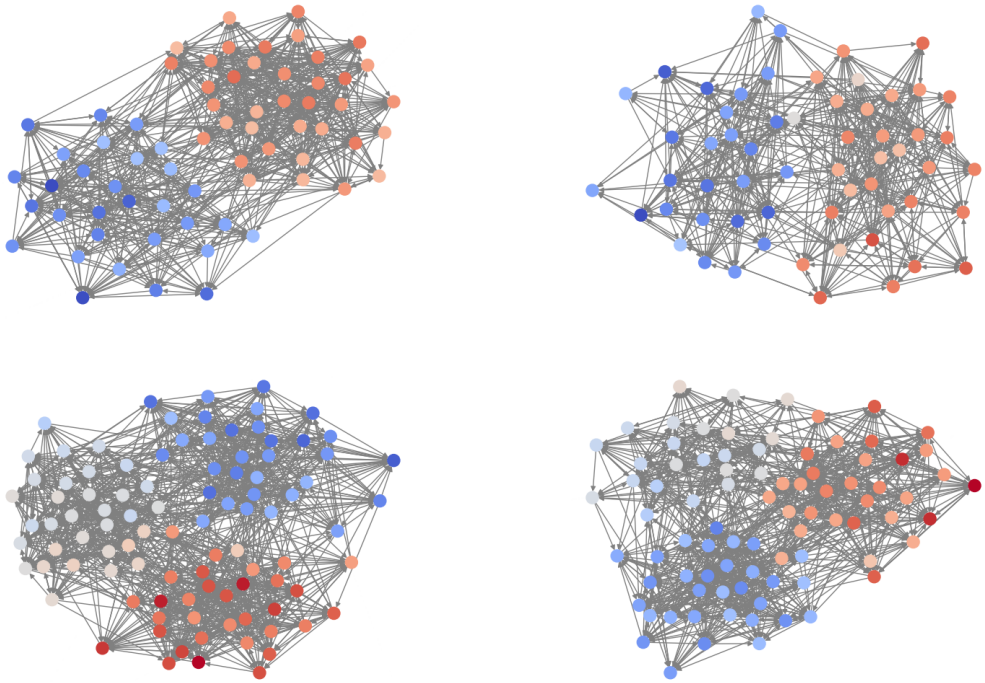


Figure 7: Visualizations of four generated digraphs for the SBM dataset with RRWP positional encoding.

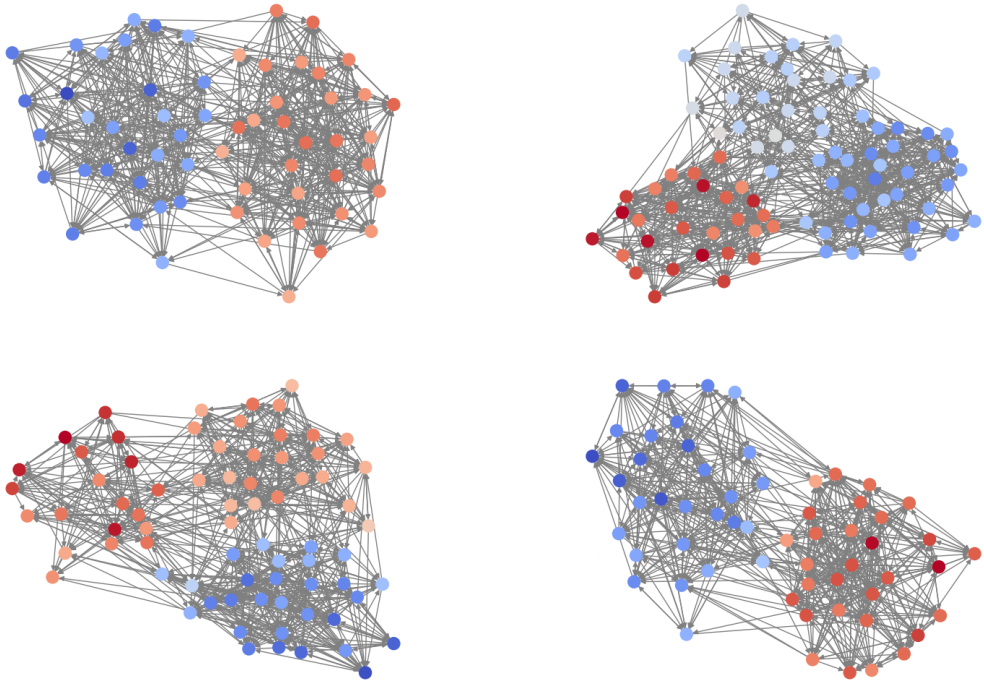


Figure 8: Visualizations of four generated digraphs for the SBM dataset with MagLap ($Q = 10$) positional encoding.

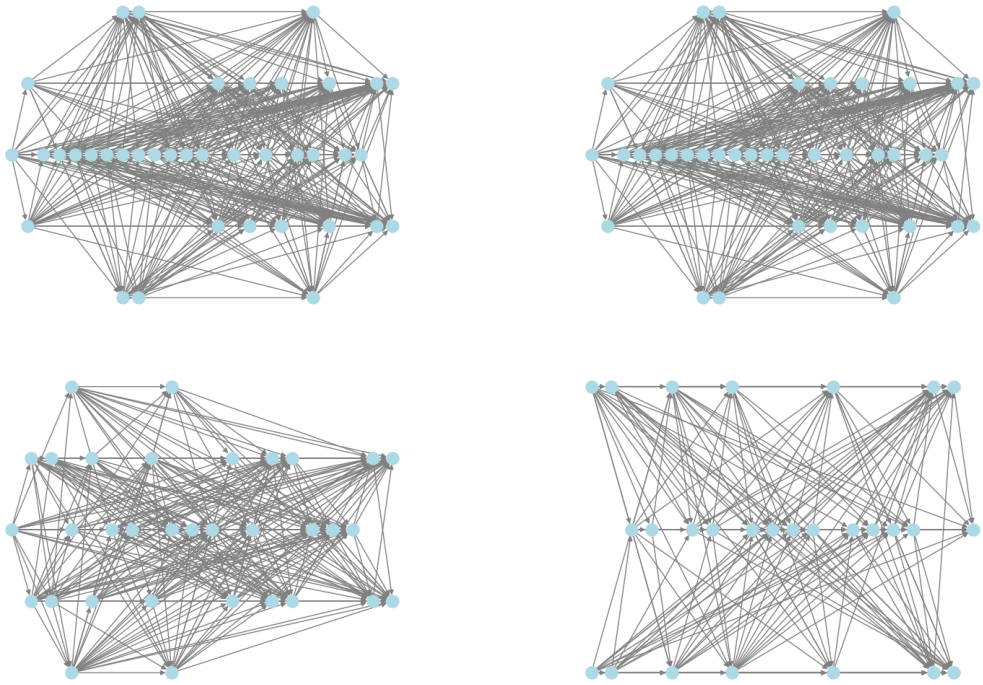


Figure 9: Visualizations of original synthetic digraphs from the train splits of the ER-DAG datasets. Nodes are in topological ordering to highlight the acyclic structure.

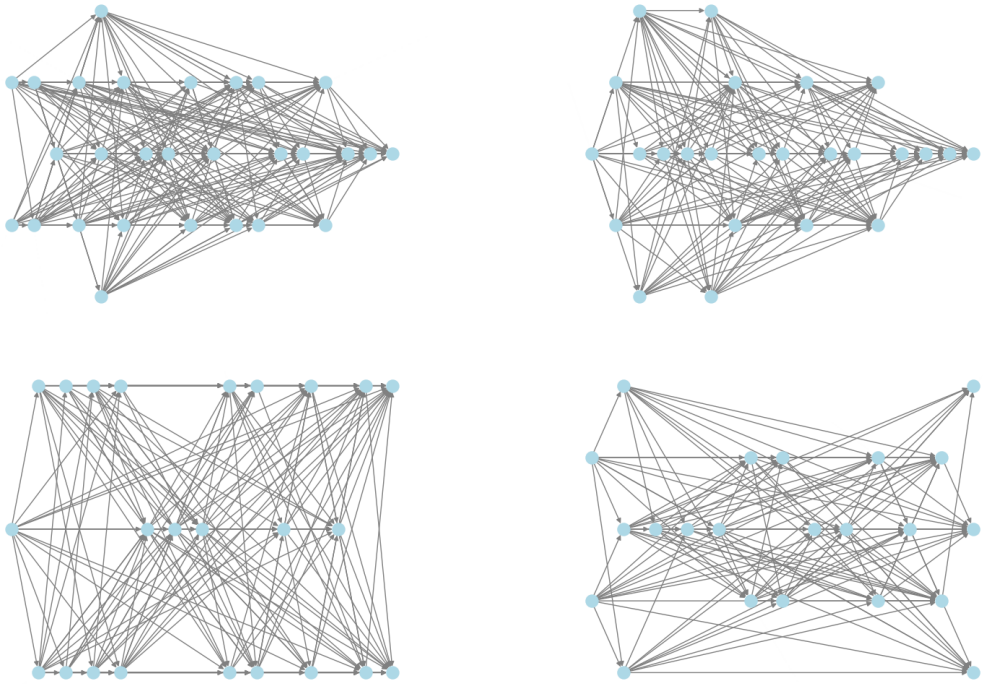


Figure 10: Visualizations of four generated digraphs for the ER-DAG dataset with RRWP positional encoding. Nodes are in topological ordering to highlight the acyclic structure.

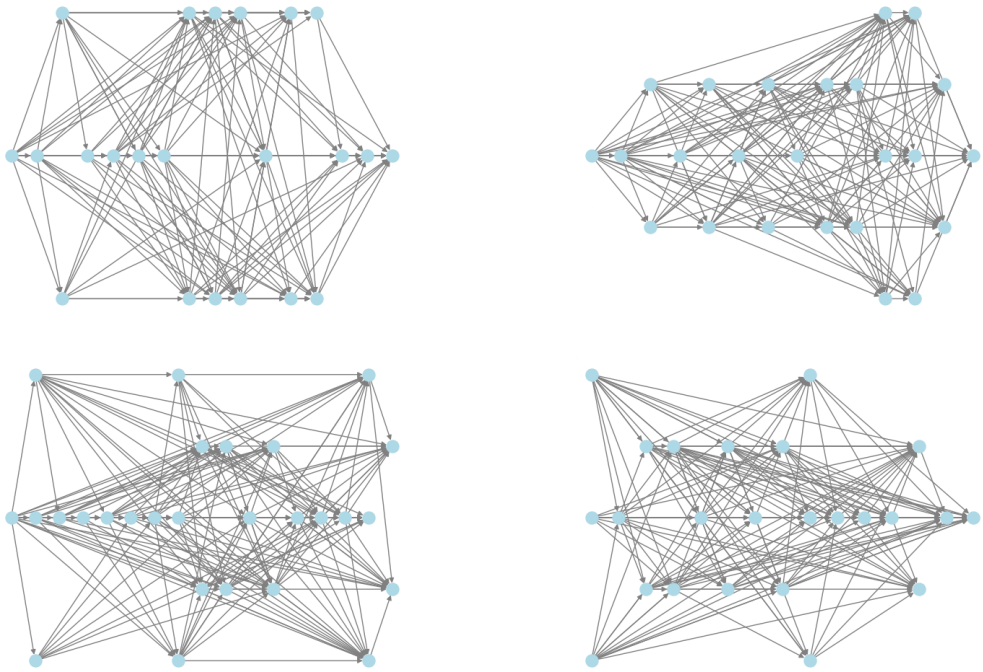


Figure 11: Visualizations of four generated digraphs for the ER-DAG dataset with MagLap ($Q = 10$) positional encoding. Nodes are in topological ordering to highlight the acyclic structure.

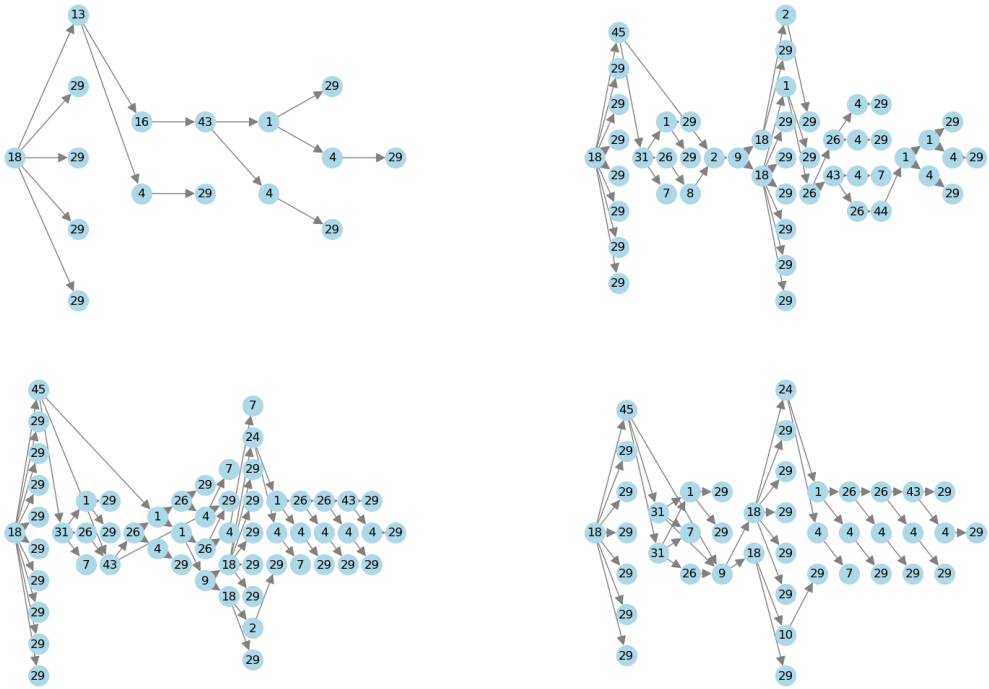


Figure 12: Visualizations of original real-world digraphs from the train splits of the TPU Tiles dataset. Nodes are in topological ordering to highlight the acyclic structure.

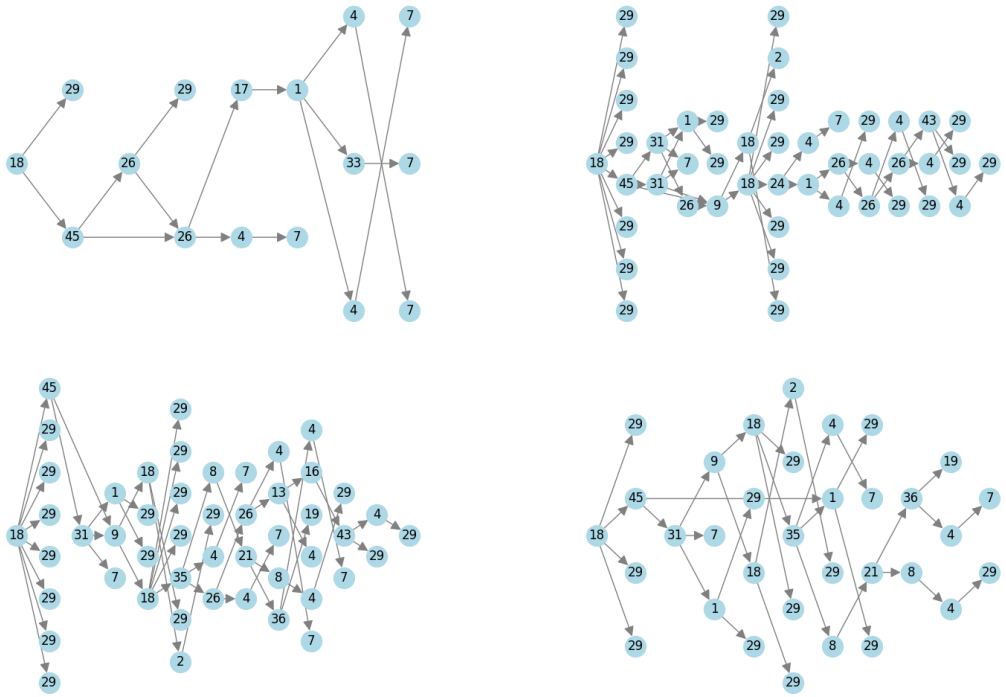


Figure 13: Visualizations of four generated dgraphs for the TPU Tiles dataset with RRWP positional encoding. Nodes are in topological ordering to highlight the acyclic structure.

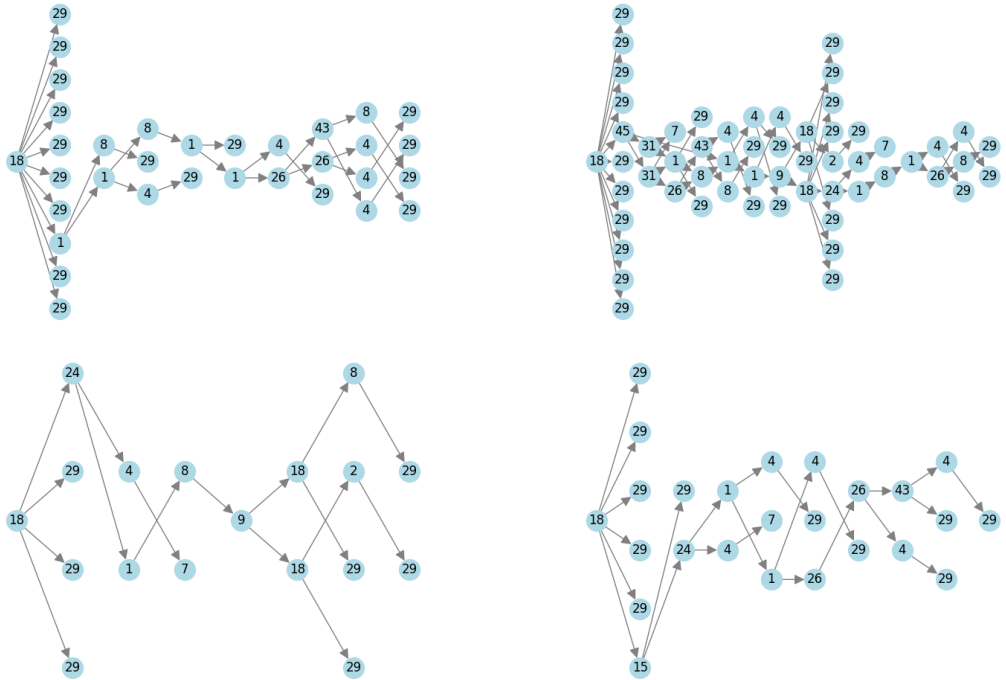


Figure 14: Visualizations of four generated digraphs for the TPU Tiles dataset with MagLap ($Q = 5$) positional encoding. Nodes are in topological ordering to highlight the acyclic structure.

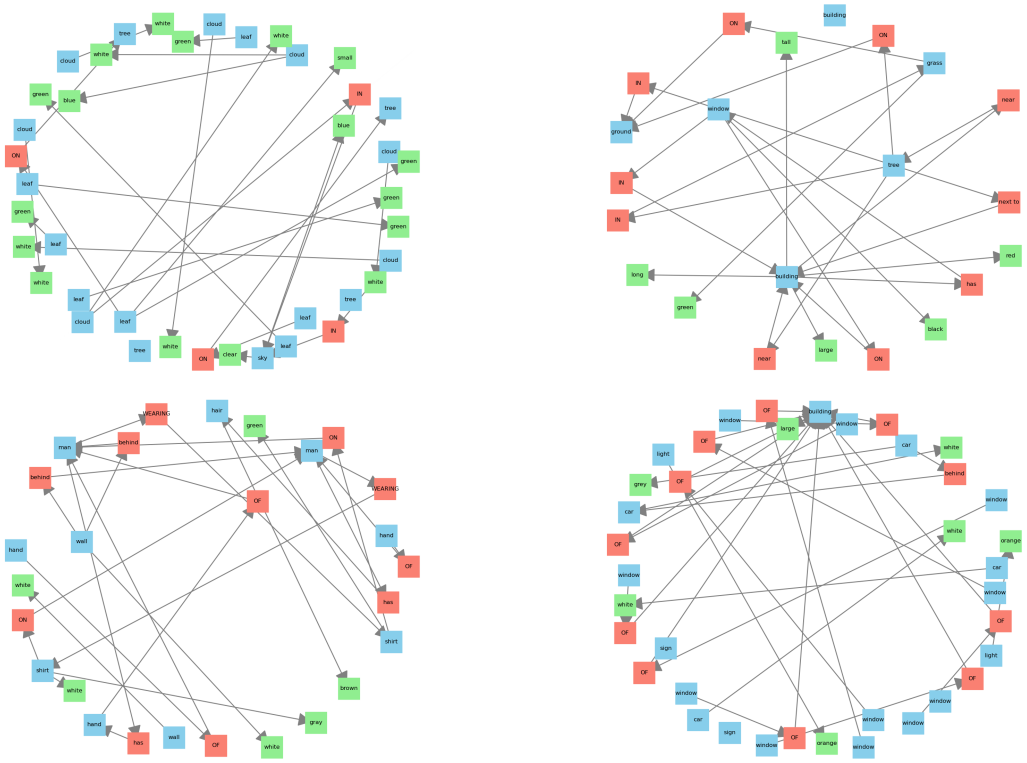


Figure 15: Visualizations of original real-world digraphs from the train splits of the Visual Genome dataset. Nodes represent objects (blue), relationships (red), and attributes (green).

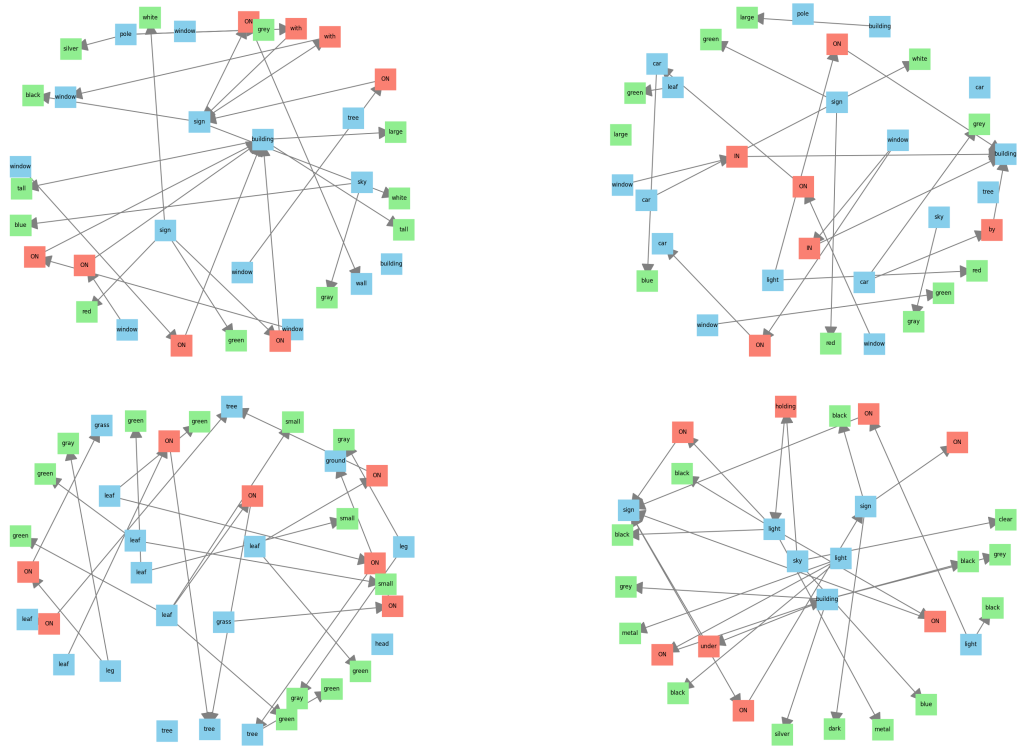


Figure 16: Visualizations of four generated digraphs for the Visual Genome dataset with RRWP positional encoding. Nodes represent objects (blue), relationships (red), and attributes (green).

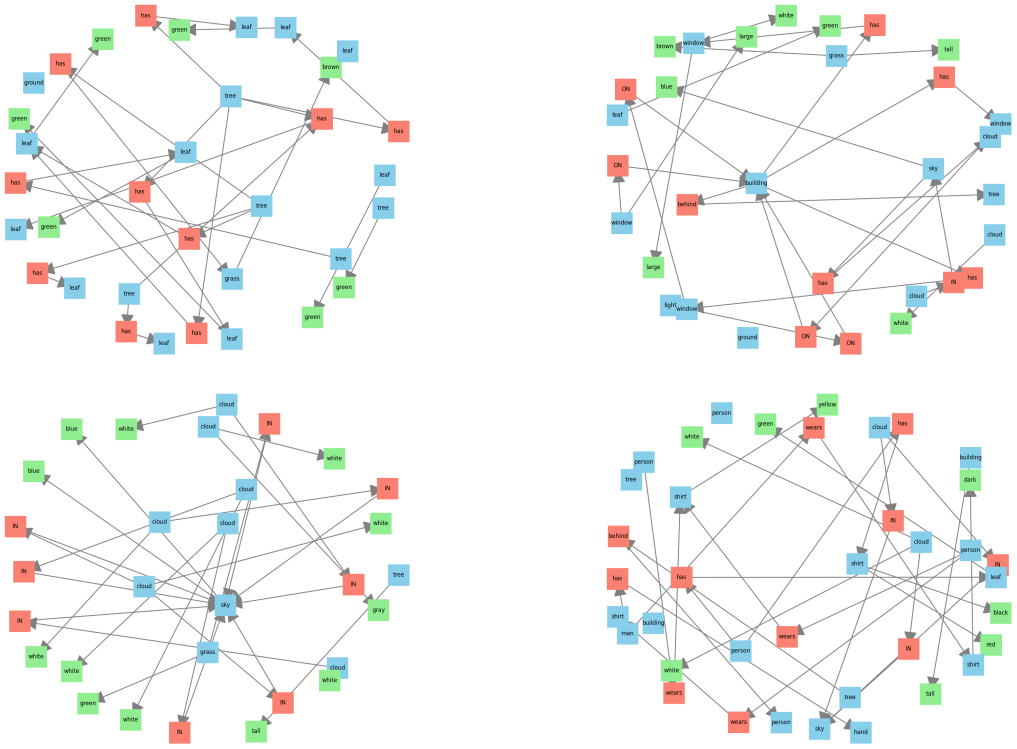


Figure 17: Visualizations of generated digraphs for the Visual Genome dataset with MagLap ($Q = 5$) positional encoding. Nodes represent objects (blue), relationships (red), and attributes (green).

K Impact statement and limitations

The objective of this work is to advance methodologies for the generation of directed graphs by introducing architectural mechanisms that explicitly model directionality and asymmetric relationships. Directed graphs are central to a wide range of applications, including causal inference, traffic modeling, and biological network analysis. Accordingly, improvements in directed graph generation have the potential to support progress in scientific research, decision-making systems, infrastructure design, or causal discovery.

The proposed framework enhances the expressiveness of generative models to directed graphs, and, at the moment, we do not identify any immediate societal risks associated with its deployment in its current form. Moreover, although the method is capable of generating structured and semantically meaningful graphs, the scale and complexity of these outputs remain limited, which may difficult direct applicability in high-impact domains such as real-time clinical diagnostics or large-scale policy modeling at this stage.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: claims made in the abstract and introduction (Section 1) are supported in Sections 2 and **??** and through experimentation in Section 3. Additionally, the supplementary material (appendices) further support our contributions.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: limitations to our work are discussed in the conclusion (Section 5) and further analyzed in Appendix K. In particular, we analyze computational efficiency and scalability of our method as their main limitations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [NA]

Justification: our paper does not provide any new theoretical results. Theoretical results from previous works are properly credited and referenced.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: the proposed method is introduced in Section 2, with further details and algorithms provided in Appendix D and F. For the experimentation, the datasets and metrics are introduced in Section **??**. Details on the datasets (inluding splits, dataset statistics and details on generation of the syntethic data) as well as the evaluation metrics are available in Appendix E. Finally, details on the experimental setup are provided in Appendix G. In addition, to make our results reproducible, we anonymously share our codebase as part of the supplementary material.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: the code provided in the supplementary material contains the proposed datasets and scripts to download all additional datasets used. It includes clear instructions for setting up the environment and running the commands needed to reproduce all experiments presented in the paper.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: experimental details on the datasets implemented and generated, as well as the splits used are available in Appendix E. Details on the hyperparameters and other model details are provided in Appendix G.2.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: we report our results of our generations as the mean $\pm$ standard deviation across 5 different sampling runs whenever the computational resources allowed.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: experimental details and information on the computational resources used are detailed in Appendix G.3.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: our submission is anonymized and respects the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: we discuss potential positive and negative societal impacts of our work in Appendix K.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [Yes]

Justification: the release of our model does not pose any direct safety risk.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: we acknowledge all sources of code and data used in our work, with citations provided both in the paper and the accompanying codebase.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: for the introduced datasets we properly cite the sources of the data, and provide further details on all datasets in Appendix E.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: experiments in this paper do not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

Answer: [NA]

Justification: experiments in this paper do not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: our method does not involve the usage of LLMs as part of the core research.

Guidelines:

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