POLARFLOW: SPECTRAL GEOMETRIC FLOW MATCHING FOR DIRECTED GRAPH GENERATION VIA POLAR DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Directed graphs model asymmetric, ordered relationships in biology, transportation, social networks, and electronic design automation, making their generation important for analysis, simulation and design. Recent discrete diffusion and flow-matching methods generate directed graphs effectively, but their edge-wise categorical formulations model every ordered node pair, requiring dense predictions during training and repeated categorical updates during generation. This motivates a fundamental question: can comparable generative quality be achieved through a more computationally efficient formulation that better exploits the structure of directed adjacency matrices? Moreover, existing directed-graph benchmarks cover a limited range of network structures, hindering a comprehensive evaluation of graph generation. We introduce *PolarFlow*, a continuous geometric flow-matching framework based on the polar decomposition of the adjacency matrix and truncated spectral coordinates. Its representation consists of four coupled coordinate components: a Stiefel frame and rotation angles that encode the orthogonal factor, together with a second Stiefel frame and nonnegative eigenvalues that characterize the positive-semidefinite factor. PolarFlow learns a joint velocity field along geometry-aware conditional paths on this product space, reconstructs a continuous adjacency-score matrix, and discretizes it into a binary directed graph. By replacing discrete edge-wise generation with joint continuous transport on this product space, PolarFlow offers a computationally efficient alternative while retaining the global structure of the adjacency matrix. To broaden evaluation, we introduce complementary Circuit and Twitter benchmarks spanning engineered functional and naturally evolved social networks. Across synthetic and real-world datasets, PolarFlow achieves generative quality comparable to competitive discrete methods while substantially reducing training and sampling time.

1 INTRODUCTION

Directed graphs naturally represent systems in which the direction of information, influence, or material flow is semantically consequential, including applications in biology (Wei et al., 2024; Takane et al., 2025; Li et al., 2006), electronic circuits (Liu et al., 2025), transportation (Concas et al., 2022), and social networks (Schweimer et al., 2022). Because edge orientation affects reachability, propagation, and feedback, generating realistic directed graphs supports simulation and data augmentation when observations are scarce or costly, as well as design-space exploration under domain-specific constraints, such as neural architecture search and circuit synthesis (Zhang et al., 2019; Liu et al., 2025; Carballo-Castro et al., 2026; Li et al., 2025).

Existing approaches span three broad strategies. First, *latent autoencoding with autoregressive decoding* encodes a directed acyclic graph (DAG) via asynchronous message passing and decodes it node-by-node (Zhang et al., 2019). Second, *layerwise autoregressive diffusion*, as in LayerDAG (Li et al., 2025), exploits DAG topology to generate each layer and its incoming edges. Third, *one-shot generation over categorical edge variables*, as in Directo (Carballo-Castro et al., 2026), extends discrete flow matching to directed graphs using direction-aware attention and asymmetric positional encodings. A related continuous latent alternative instead applies flow matching to node-

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

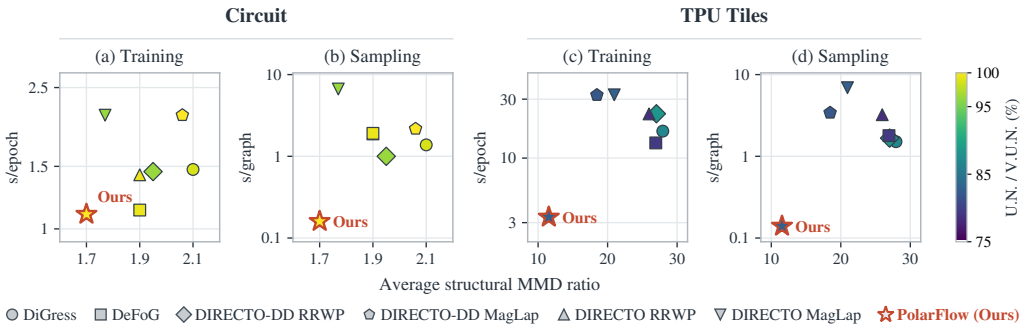

Figure 1: Quality-efficiency comparisons on Circuit and TPU Tiles. Lower-left is better.

level representations learned by a separately trained graph autoencoder (Siraudin & Morris, 2026). Further discussion and comparison of related methods are provided in Appendix A.

Each paradigm entails distinct computational or representational trade-offs. Autoregressive and layer-wise methods generate nodes or topological layers sequentially, limiting parallelism; their reliance on topological structure also restricts them to DAGs. Discrete flow-matching methods avoid this sequential dependence by predicting all edge variables in parallel at each step, but require dense outputs over all $N(N-1)$ ordered node pairs. With $\mathcal{C}_E$ denoting the number of edge categories, including the absent-edge category, each network evaluation produces $\mathcal{O}(N^2\mathcal{C}_E)$ edge logits, accumulating to $\mathcal{O}(TN^2\mathcal{C}_E)$ over T sampling steps, excluding the costs of the denoising backbone and structural encodings. Continuous latent methods reduce repeated adjacency-space prediction but require a separately trained encoder–decoder. These trade-offs motivate a central question: can a continuous representation derived directly from the adjacency matrix capture orientation-sensitive structure and achieve comparable generative quality more *efficiently*?

To answer this question, we seek a representation of directed graphs that enables continuous flow matching while retaining the global structure of the adjacency matrix. We achieve this through polar decomposition, which separates a directed adjacency matrix into orientation and magnitude components, represented in a compact set of spectral and geometric coordinates. Building on this representation, we introduce *PolarFlow*, which learns a continuous velocity field specifying how the orientation and magnitude coordinates evolve together along paths from source to data representations; the adjacency matrix is reconstructed only at the end of the generation process. This design replaces edge-wise categorical prediction and iterative discrete updates with continuous transport in a lower-dimensional structured space, directly targeting the computational bottleneck of existing discrete approaches. Importantly, our goal is not to trade generative quality for efficiency: as shown in Section 5 (previewed in Fig. 1), PolarFlow achieves quality comparable to that of strong discrete flow-matching methods while substantially reducing both training and sampling times.

Our contributions are summarized as follows.

- **(i)** We introduce a compact polar–spectral representation for general directed graphs that separates orientation-sensitive alignment from nonnegative spectral magnitude. Its truncated coordinates comprise two Stiefel frames, rotation angles, and nonnegative eigenvalues.
- **(ii)** We develop geometry-aware conditional flow matching that jointly evolves these four channels, together with a reconstruction and discretization procedure for generating binary directed structures, and a separate process for categorical node-label generation.
- **(iii)** We establish the local perturbation stability of PolarFlow under spectral and geometric regularity, proving that small perturbations of a regularized adjacency endpoint induce proportionally small changes in the conditional flow and learned velocity predictions.
- **(iv)** We introduce new *Circuit and Twitter datasets*, representing engineered functional networks and naturally formed social networks, respectively. We demonstrate that PolarFlow achieves competitive generative quality while delivering substantial empirical efficiency gains across both synthetic and real-world directed graphs.

2 Problem Formulation and Preliminaries

Directed graphs and conditional generation Let $G = (\mathcal{V}, \mathbf{A}, \mathbf{y})$ be a directed graph with vertex set $\mathcal{V} = \{1, \dots, N\}$, binary adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, and optional categorical node labels $\mathbf{y} \in \mathcal{C}^N$, where $\mathcal{C}$ is finite. We use $\mathbf{A}_{ij} = 1$ to indicate the directed edge $i \to j$ and assume $\mathbf{A}_{ii} = 0$ unless stated otherwise. Given training graphs, our goal is to learn the adjacency distribution $p_\phi(\mathbf{A} \mid N)$ conditioned on graph size and, when node labels are available, the label distribution $p_\psi(\mathbf{y} \mid \mathbf{A})$ conditioned on the adjacency matrix, where ϕ and ψ denote the parameters of the structural and label models, respectively. At generation time, N is sampled from the empirical distribution of graph sizes in the training set and held fixed throughout structural generation; for labeled datasets, node labels are then sampled conditional on the generated adjacency matrix.

Polar decomposition and spectral representations Every real square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ admits a right polar decomposition (see Kwak et al. (2024); Ji & Ortega (2026) for the graph signal processing perspective)

$$\mathbf{A} = \mathbf{U}\mathbf{P}, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{N \times N}$ is orthogonal and $\mathbf{P} = (\mathbf{A}^*\mathbf{A})^{1/2} \succeq 0$[1] is symmetric positive semi-definite. The symmetric factor admits the eigendecomposition

$$\mathbf{P} = \mathbf{Q}_p \mathbf{\Lambda}_p \mathbf{Q}_p^*, \qquad \mathbf{\Lambda}_p = \mathrm{diag}(\lambda_{p,1}, \dots, \lambda_{p,N}), \tag{2}$$

where $\mathbf{Q}_p$ is orthogonal and $\lambda_{p,1} \geq \lambda_{p,2} \geq \cdots \geq \lambda_{p,N} \geq 0$. The orthogonal factor $\mathbf{U}$ is normal and also admits an unitary eigendecomposition, whose eigenvalues lie on the complex unit circle. Specifically, we have

$$\mathbf{U} = \mathbf{Q}_u \mathbf{\Lambda}_u \mathbf{Q}_u^*, \tag{3}$$

where the diagonal matrix $\mathbf{\Lambda}_u$ consists of eigenvalues of the form $e^{\mathrm{i}\theta_i}, \theta_i \in [0, 2\pi)$ for $1 \leq i \leq N$, which is either ± 1 or form complex conjugate pairs. Thus, the polar decomposition represents $\mathbf{A}$ through orthonormal/unitary frames, circular rotation angles, and nonnegative spectral values.

Stiefel manifolds and Riemannian flow matching The spectral frames $\mathbf{Q}_p$ and $\mathbf{Q}_u$ consist of orthonormal columns. An orthonormal or unitary k-frame belongs to the Stiefel manifold (Hatcher, 2002; Absil et al., 2008)

$$\mathrm{St}(N,k) = \left\{ \mathbf{S} \in \mathbb{M}^{N \times k} \,:\, \mathbf{S}^*\mathbf{S} = \mathbf{I}_k \right\},$$

with $\mathbb{M}$ either $\mathbb{R}$ (for orthogonal matrices) or $\mathbb{C}$ (for unitary matrices). The full orthonormal basis corresponds to the special case $k = N$. The tangent space at a point $\mathbf{S} \in \mathrm{St}(N,k)$ is

$$\mathcal{T}_{\mathbf{S}}(\mathrm{St}(N,k)) = \left\{ \mathbf{V} \in \mathbb{M}^{N \times k} \,:\, \mathbf{S}^*\mathbf{V} + \mathbf{V}^*\mathbf{S} = 0 \right\}.$$

The Stiefel manifold is equipped with the Riemannian metric $g_{\mathbf{S}}(\mathbf{V}_1, \mathbf{V}_2) = \mathrm{tr}(\mathbf{V}_1^*\mathbf{V}_2)$ for $\mathbf{V}_1, \mathbf{V}_2 \in \mathcal{T}_{\mathbf{S}}(\mathrm{St}(N,k))$. *Note*: for convenience, we do not distinguish real and complex Stiefel manifolds in notation if no confusion arises.

Riemannian flow matching generalizes flow matching to a Riemannian manifold $(\mathcal{M}, g)$, where $\mathcal{M}$ denotes the state space and g_Z defines an inner product on each tangent space $\mathcal{T}_Z\mathcal{M}$. We adopt its conditional formulation, which constructs a path for each source–data endpoint pair. Given a source endpoint Z_0 and a data endpoint Z_1, let $Z_t \in \mathcal{M}$ be a differentiable conditional path satisfying $Z_{t=0} = Z_0$ and $Z_{t=1} = Z_1$ for $t \in [0,1]$. Its velocity $\dot{Z}_t = \mathrm{d}Z_t/\mathrm{d}t \in \mathcal{T}_{Z_t}\mathcal{M}$ provides a tractable regression target for a time-dependent vector field $v_\phi(Z,t) \in \mathcal{T}_Z\mathcal{M}$, with the prediction error measured using g_{Z_t}. Because Z_t and $\mathrm{d}Z_t/\,\mathrm{d}t$ are analytically available during training, the learned ODE need not be integrated and is solved only during generation.

3 Methodology: PolarFlow

In this section, we introduce PolarFlow, a geometry-aware flow-matching framework that maps directed adjacency matrices into four coupled polar–spectral channels: two Stiefel frames, rotation angles, and nonnegative eigenvalues. A shared time-dependent vector field is trained jointly along channel-specific conditional paths in the resulting product space. We establish local perturbation stability, describe reconstruction and discretization to obtain a binary directed adjacency matrix, and use a separate structure-conditioned model to generate node labels for labeled datasets. The complete PolarFlow pipeline is illustrated in Fig. 2.

[1] In this paper, we use * for the Hermitian transpose, which is the usual matrix transpose for real matrices.

3.1 FOUR-CHANNEL POLAR–SPECTRAL REPRESENTATION

Rather than modeling the N^2 entries of a directed adjacency matrix directly, PolarFlow constructs a compact geometric representation with four polar–spectral channels. It obtains these channels by applying the polar decomposition in (1) to separate the orientation and scaling information in the input matrix. For a data adjacency matrix $\mathbf{A}$, the representation is computed from the perturbed matrix $\mathbf{A}_\varepsilon = \mathbf{A} + \mathbf{E}_\varepsilon$, where the perturbation satisfies $\|\mathbf{E}_\varepsilon\|_2 \le \varepsilon$. This perturbation is introduced to mitigate potential rank and spectral degeneracies; setting $\varepsilon = 0$ recovers the unperturbed endpoint.

For the scaling channel, let $\lambda_{p,1} \ge \lambda_{p,2} \ge \cdots \ge \lambda_{p,N} \ge 0$ denote the eigenvalues of $\mathbf{P}$, with corresponding orthonormal eigenvectors $\mathbf{q}_{p,1}, \ldots, \mathbf{q}_{p,N}$. Retaining the k leading eigenpairs gives the rank-k approximation

$$\mathbf{P}_k = \mathbf{Q}_p \mathbf{\Lambda}_p \mathbf{Q}_p^* \qquad \mathbf{Q}_p = [\mathbf{q}_{p,1}, \ldots, \mathbf{q}_{p,k}] \in \mathrm{St}(N,k) \tag{4}$$

where $\mathbf{\Lambda}_p = \mathrm{diag}(\boldsymbol{\lambda}_p^{(k)})$ with $\boldsymbol{\lambda}_p^{(k)} = (\lambda_{p,1}, \cdots, \lambda_{p,k}) \in \mathbb{R}_+^k$. Thus, $\mathbf{Q}_p$ and $\boldsymbol{\lambda}_p^{(k)}$ constitute the two scaling channels. To construct the two orientation channels $\mathbf{Q}_u$ and $\boldsymbol{\theta}^{(k)}$, we use the spectral representation of the orthogonal factor $\mathbf{U}$ in (3). Let $(e^{\mathrm{i}\tilde{\theta}_i}, \tilde{\mathbf{q}}_{u,i})$ with $i = 1, \ldots, N$ denote its eigenpairs, where $\tilde{\theta}_i \in (-\pi, \pi]$ and $\tilde{\mathbf{q}}_{u,i} \in \mathbb{C}^N$. Since $\mathbf{U}$ is real, every nonreal eigenmode occurs together with its complex conjugate, whereas the real eigenvalues $+1$ and -1 correspond to the angles 0 and π, respectively.

Unlike the scaling modes of $\mathbf{P}$, which are naturally ranked by their eigenvalues, all eigenvalues of the orthogonal factor $\mathbf{U}$ have unit modulus. Their magnitudes therefore provide no criterion for constructing a k-dimensional orientation representation compatible with the rank-k scaling representation. To select the k modes most relevant to the product $\mathbf{A}_\varepsilon = \mathbf{UP}$, we evaluate each mode using both its angular deviation from the identity and the scaling magnitude of $\mathbf{P}$ along its direction. This motivates the polar-weighted orientation score $s_i = 2\big|\sin(\widetilde{\theta}_i/2)\big|\widetilde{\mathbf{q}}_{u,i}^* \mathbf{P} \widetilde{\mathbf{q}}_{u,i}$, where $(\cdot)^*$ denotes the conjugate transpose. The angular factor $2|\sin\big(\widetilde{\theta}_i/2\big)| = |e^{\mathrm{i}\widetilde{\theta}_i} - 1|$ measures the deviation of the corresponding eigenvalue from the identity orientation, while $\widetilde{\mathbf{q}}_{u,i}^* \mathbf{P} \widetilde{\mathbf{q}}_{u,i}$ measures the scaling magnitude associated with that mode. Because $\mathbf{P}$ is real and positive semidefinite, $s_i \in \mathbb{R}_{\ge 0}$, and every complex-conjugate pair receives identical scores. The score therefore provides a valid scalar ranking that is consistent with pair-preserving mode selection.

PolarFlow then selects a size-k index set $\mathcal{I}_u = \{i_1, \ldots, i_k\}$ with the highest scores, subject to closure under complex conjugation. In particular, whenever a selected index corresponds to a nonreal eigenvalue, the index of its conjugate mode is also included. Thus, each nonreal conjugate pair contributes two indices to $\mathcal{I}_u$. Using this common index set, the selected eigendirections are represented by a real orthonormal frame and their corresponding eigenangles are collected as

$$\mathbf{Q}_u \in \mathrm{St}(N,k) \qquad \boldsymbol{\theta}^{(k)} = (\tilde{\theta}_{i_1}, \ldots, \tilde{\theta}_{i_k})^\top \in \Theta_k \subseteq \mathbb{T}^k. \tag{5}$$

Here, $\mathbb{T} = \mathbb{R}/(2\pi\mathbb{Z}) \simeq S^1$ denotes the circle of angles modulo 2π, and $\Theta_k \subseteq \mathbb{T}^k$ is the admissible subset induced by the conjugate structure of the retained orientation modes.

Combining the scaling and orientation components, we define the k-dimensional four-channel representation of $\mathbf{A}_\varepsilon$ as $\Phi_k(\mathbf{A}_\epsilon) = \big(\mathbf{Q}_p, \boldsymbol{\lambda}_p^{(k)}, \mathbf{Q}_u, \boldsymbol{\theta}^{(k)}\big) \in \mathcal{M}_k$, where $\mathcal{M}_k = \mathrm{St}(N,k) \times \mathbb{R}_+^k \times \mathrm{St}(N,k) \times \Theta_k$ is the associated geometric product state space. The two Stiefel factors represent the retained scaling and orientation directions, while $\boldsymbol{\lambda}_p^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ contain their respective spectral coordinates.

3.2 COUPLED ENDPOINT DISTRIBUTIONS AND GEOMETRY-AWARE PATHS

PolarFlow constructs its source and target states by applying the same polar–spectral map Φ_k to the corresponding matrix endpoints. For each graph size N, let $\mathbf{A}_0 \in \mathbb{R}^{N \times N}$ be a deterministic source template and let $\mathbf{F}_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. The source matrix endpoint and its four-channel representation are defined as

$$\mathbf{M}_0 = \mathbf{A}_0 + \sigma \mathbf{F}, \qquad \mathbf{Z}_0 = \Phi_k(\mathbf{M}_0), \tag{6}$$

where $\sigma > 0$ controls the source noise level. For a target adjacency matrix $\mathbf{A}_1 \sim p_{\mathrm{data}}(\cdot \mid N)$, the regularized target matrix endpoint and its representation are

$$\mathbf{M}_1 = \mathbf{A}_1 + \mathbf{E}_\varepsilon, \qquad \mathbf{Z}_1 = \Phi_k(\mathbf{M}_1), \qquad \|\mathbf{E}_\varepsilon\| \le \varepsilon \tag{7}$$

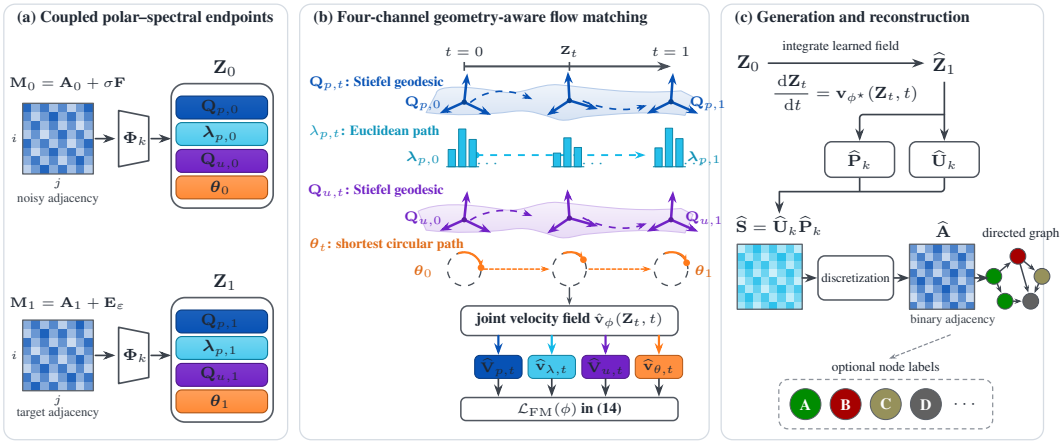

Figure 2: Overview of the proposed PolarFlow framework.

These constructions induce source and target distributions over the four-channel state space. Because every endpoint is obtained from a single matrix through one application of Φ_k, its orientation and scaling channels retain the dependencies induced by the underlying polar decomposition. In particular, PolarFlow does not construct an endpoint by independently sampling the four channel-wise marginals.

To define a conditional path between a sampled source–target pair, we make the four components of each endpoint explicit, i.e, $\mathbf{Z}_\ell = (\mathbf{Q}_{p,\ell}, \lambda_{p,\ell}^{(k)}, \mathbf{Q}_{u,\ell}, \boldsymbol{\theta}_\ell^{(k)}) \in \mathcal{M}_k$ with $\ell \in \{0,1\}$. PolarFlow constructs a conditional path using a shared time $t \sim \mathrm{Unif}([0,1])$. The orientation and scaling frames follow Stiefel geodesics:

$$\mathbf{Q}_{c,t} = \mathrm{Exp}_{\mathbf{Q}_{c,0}}\Big(t \,\mathrm{Log}_{\mathbf{Q}_{c,0}}(\mathbf{Q}_{c,1})\Big), \qquad c \in \{u,p\}, \tag{8}$$

where Exp and Log are the exponential and logarithmic functions between the manifold and its tangent spaces. For the orientation-angle channel, define the componentwise shortest angular displacement as $\boldsymbol{\delta}_\theta^{(k)} = \mathrm{wrap}(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_0)$, where $\mathrm{wrap}(\cdot)$ returns the principal representative modulo 2π. The corresponding circular path is $\boldsymbol{\theta}_t^{(k)} = \mathrm{wrap}(\boldsymbol{\theta}_0^{(k)} + t\boldsymbol{\delta}_{\boldsymbol{\theta}}^{(k)})$, and the spectral-value channel follows a Euclidean linear path $\boldsymbol{\lambda}_{p,t}^{(k)} = (1-t)\boldsymbol{\lambda}_{p,0}^{(k)} + t\boldsymbol{\lambda}_{p,1}^{(k)}$. Since both endpoint vectors are nonnegative, the interpolated spectral values remain in $\mathbb{R}_+^k$. The four channels together define the synchronized intermediate state

$$\mathbf{Z}_t = (\mathbf{Q}_{p,t}, \boldsymbol{\lambda}_{p,t}^{(k)}, \mathbf{Q}_{u,t}, \boldsymbol{\theta}_t^{(k)}) \in \mathcal{M}_k. \tag{9}$$

The corresponding pathwise velocities provide the conditional targets for geometric flow matching. Along the selected geodesic branches, the corresponding channel-wise velocities are

$$\dot{\mathbf{Q}}_{c,t} = \frac{\mathrm{d}\mathbf{Q}_{c,t}}{\mathrm{d}t} \qquad \text{with} \qquad c \in \{u,p\}, \tag{10}$$

while the angle and spectral-value velocities are

$$\dot{\boldsymbol{\theta}}_t^{(k)} = \frac{\mathrm{d}\boldsymbol{\theta}_t^{(k)}}{\mathrm{d}t} = \boldsymbol{\delta}_\theta^{(k)}, \quad \text{and} \quad \dot{\boldsymbol{\lambda}}_{p,t}^{(k)} = \frac{\mathrm{d}\boldsymbol{\lambda}_{p,t}^{(k)}}{\mathrm{d}t} = \boldsymbol{\lambda}_{p,1}^{(k)} - \boldsymbol{\lambda}_{p,0}^{(k)}$$

Collectively, $\mathbf{v}_t(\mathbf{Z}_t|\mathbf{Z}_0,\mathbf{Z}_1) = \big(\dot{\mathbf{Q}}_{p,t}, \dot{\boldsymbol{\lambda}}_{p,t}^{(k)}, \dot{\mathbf{Q}}_{u,t}, \dot{\boldsymbol{\theta}}_t^{(k)}\big)$ defines the geometry-aware conditional velocity target. Using the same time t for all four channels ensures that the components associated with a single endpoint pair evolve synchronously.

3.3 Joint Geometry-Aware Velocity Matching

PolarFlow models the joint evolution of the four polar–spectral channels using a single time-dependent velocity field. For a sampled source–target pair, the geometry-aware path at time t provides the conditional velocity target $\mathbf{v}_t(\mathbf{Z}_t|\mathbf{Z}_0,\mathbf{Z}_1) = \big(\dot{\mathbf{Q}}_{p,t}, \dot{\boldsymbol{\lambda}}_{p,t}^{(k)}, \dot{\mathbf{Q}}_{u,t}, \dot{\boldsymbol{\theta}}_t^{(k)}\big)$. A shared network receives the complete intermediate state $\mathbf{Z}_t$ and time t and jointly predicts the four velocity components:

$$\hat{\mathbf{v}}_\phi(\mathbf{Z}_t,t) = (\widehat{\mathbf{V}}_{p,t}, \hat{\mathbf{v}}_{\lambda,t}, \widehat{\mathbf{V}}_{u,t}, \hat{\mathbf{v}}_{\theta,t}) \in \mathcal{T}_{\mathbf{Z}_t}\mathcal{M}_k. \tag{11}$$

Here $\widehat{\mathbf{V}}_{c,t}$ denotes the velocity prediction associated with the Stiefel component $\mathbf{Q}_{c,t}$ after projection onto $\mathcal{T}_{\mathbf{Q}_{c,t}}\,\mathrm{St}(N,k)$ for $c \in \{p,u\}$. The associated product tangent space is

$$\mathcal{T}_{\mathbf{Z}_t}\mathcal{M}_k = \mathcal{T}_{\mathbf{Q}_{p,t}}\,\mathrm{St}(N,k) \times \mathbb{R}^k \times \mathcal{T}_{\mathbf{Q}_{u,t}}\,\mathrm{St}(N,k) \times \mathcal{T}_{\boldsymbol{\theta}_t^{(k)}}\Theta_k. \tag{12}$$

All four velocity components are predicted from the full state $\mathbf{Z}_t$, rather than from their corresponding channels in isolation. Thus, each prediction may depend on both the scaling and orientation.

Let $e_\phi(\mathbf{A}_\epsilon, t)$ be the *velocity approximation error* $\mathbf{v}_t(\mathbf{Z}_t|\mathbf{Z}_0,\mathbf{Z}_1) - \hat{\mathbf{v}}_\phi(\mathbf{Z}_t,t)$. The network is trained by matching minimizing the expected approximation error for velocity matching:

$$\mathcal{L}_{\mathrm{FM}}(\phi) = \mathbb{E}\|e_\phi(\mathbf{A}_\epsilon,t)\|^2 \tag{13}$$

$$= \mathbb{E}\big[\|\widehat{\mathbf{V}}_{p,t} - \dot{\mathbf{Q}}_{p,t}\|_F^2 + \|\widehat{\mathbf{v}}_{\lambda,t} - \dot{\boldsymbol{\lambda}}_{p,t}^{(k)}\|_2^2 + \|\widehat{\mathbf{V}}_{u,t} - \dot{\mathbf{Q}}_{u,t}\|_F^2 + \|\widehat{\mathbf{v}}_{\theta,t} - \dot{\boldsymbol{\theta}}_t^{(k)}\|_2^2\big] \tag{14}$$

where the expectation is taken over the source–target endpoint sampling procedure and $t \sim \mathrm{Unif}([0,1])$. This is a direct velocity-prediction objective. The endpoints $\mathbf{Z}_0$ and $\mathbf{Z}_1$ are used only to construct $\mathbf{Z}_t$ and its conditional velocity target, while the network receives only (Z_t, t). Unlike discrete graph flow matching, which parameterizes categorical transition dynamics over categorical node-pair variables, PolarFlow learns a continuous vector field over the joint polar–spectral state, thereby modeling the coordinated evolution of global scaling and orientation information.

Theoretical analysis We have the following result regarding the stability of PolarFlow targets. See Appendix B for the proof.

Theorem 1 (Stability of PolarFlow)**.** *Suppose $\mathbf{A}_\varepsilon$ is regular (formally defined in Appendix B) and that, for a fixed source endpoint, the approximation error $e_\phi(\mathbf{A}'_\epsilon, t)$ is locally Lipschitz with respect to $\mathbf{A}'_\varepsilon$ in a neighborhood of $\mathbf{A}_\varepsilon$, uniformly over $t \in [0,1]$. Then there exist constants $r > 0$ and $C > 0$ such that, for every $\widetilde{\mathbf{A}}_\varepsilon$ satisfying $\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\|_2 < r$, the following holds*

$$\sup_{t\in[0,1]} \left\| \widehat{\mathbf{v}}_\phi\left(\mathbf{Z}_t(\widetilde{\mathbf{A}}_\varepsilon), t\right) - \widehat{\mathbf{v}}_\phi\left(\mathbf{Z}_t(\mathbf{A}_\varepsilon), t\right) \right\| \le C \left\| \widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon \right\|_2. \tag{15}$$

Theorem 1 establishes the local perturbation stability of PolarFlow at a spectrally and geometrically regular endpoint: small input perturbations induce proportionally small changes in the conditional paths and velocity targets. This stability extends to the learned velocity field, which is preferred to avoid pathological behavior. It also provides theoretical support for endpoint regularization as a mechanism for stabilizing flow-matching targets. This interpretation is consistent with the generation equality gains obtained under suitable regularization strengths in Appendix E.3.

3.4 Structure Reconstruction and Discretization

For generation, PolarFlow first samples a source matrix $\mathbf{M}_0$ and computes its polar–spectral representation $\mathbf{Z}_0 = \Phi_k(\mathbf{M}_0)$ in (6). Start from $\mathbf{Z}_0$, PolarFlow integrates the ODE induced by the learned velocity field $\mathbf{v}_{\phi^\star}$ from $t = 0$ to $t = 1$, i.e., $\mathrm{d}\mathbf{Z}_t/\,\mathrm{d}t = \mathbf{v}_{\phi^\star}(\mathbf{Z}_t, t)$ with $\mathbf{Z}_{t=0} = \mathbf{Z}_0$, where $\phi^\star$ denotes the learned parameters. This produces the terminal polar–spectral state $\widehat{\mathbf{Z}}_1 = (\widehat{\mathbf{Q}}_p, \widehat{\boldsymbol{\lambda}}_p^{(k)}, \widehat{\mathbf{Q}}_u, \widehat{\boldsymbol{\theta}}^{(k)}) \in \mathcal{M}_k$. The terminal state is first decoded into a continuous matrix representation. The scaling and orientation components are reconstructed respectively as

$$\widehat{\mathbf{P}}_k = \widehat{\mathbf{Q}}_p\,\mathrm{diag}(\widehat{\boldsymbol{\lambda}}_p^{(k)})\widehat{\mathbf{Q}}_p^* \quad \text{and} \quad \widehat{\mathbf{U}}_k = \widehat{\mathbf{Q}}_u\,\mathrm{diag}(e^{\mathrm{i}\boldsymbol{\theta}^{(k)}})\widehat{\mathbf{Q}}_u^*.$$

The continuous directed-adjacency score matrix is then obtained through the polar product $\widehat{\mathbf{S}} = \widehat{\mathbf{U}}_k\widehat{\mathbf{P}}_k \in \mathbb{R}^{N\times N}$. The resulting $\widehat{\mathbf{S}}$ assigns a continuous score to each ordered node pair and is converted into a binary adjacency matrix through the size-conditioned discretization. To obtain a binary adjacency matrix, PolarFlow estimates the size-conditioned edge density ρ_N from the training set, i.e., $\rho_N = \mathbb{E}_{\mathbf{A}\sim p_{\mathrm{data}}(\cdot|N)}\left[\frac{1}{N(N-1)}\sum_{i\neq j} A_{ij}\right]$. The corresponding edge budget is $m_N = \mathrm{round}\,(\rho_N N(N-1))$. PolarFlow constructs $\widehat{\mathbf{A}}$ by assigning ones to the m_N largest off-diagonal entries of $\widehat{\mathbf{S}}$ and zeros to all remaining entries. Self-loops are excluded by setting $\widehat{A}_{ii} = 0$ for all i. The resulting $\widehat{\mathbf{A}} \in \{0,1\}^{N\times N}$ is the binary adjacency matrix of the generated directed graph.

Categorical node labels We model $p_\psi(\mathbf{y} \mid \widehat{\mathbf{A}})$ using a separate masked categorical predictor, with training based on discrete flow matching (Qin et al., 2025) and sampling based on confidence-guided iterative unmasking (Chang et al., 2022). The generated adjacency matrix $\widehat{\mathbf{A}}$ remains fixed throughout label generation; details are provided in Appendix E.4.

4 Expanding Directed-Graph Dataset Diversity

Existing benchmarks for directed-graph generation remain limited in structural diversity (e.g., sparsity) and domain coverage, restricting evaluation to a relatively narrow range of directed connectivity patterns. To broaden this scope, we construct two additional benchmark datasets, *Circuits* and *Twitter*, representing engineered functional systems and naturally evolved social systems, respectively.

Circuits: engineered functional networks. We construct the Circuits dataset from two open-source hardware designs: the Ibex RISC-V CPU core and an Ethernet MAC controller. After compiling the designs into FPGA cell-level netlists of AMD Xilinx primitives, we represent each netlist as a directed cell graph and extract radius-two ego-subgraphs by traversing both incoming and outgoing connections while retaining feedback. Each node denotes a primitive cell instance and carries one of seven categorical cell-type labels, while an edge $u \to v$ indicates that the signal produced by cell u drives an input of cell v. Parallel pin-level connections are collapsed into binary edges, and self-loops are removed. The dataset contains 318 graphs, split into 251 training, 41 validation, and 26 test graphs. Further construction details and dataset statistics are provided in Appendix C.3.

Twitter: naturally evolved social networks. We construct the Twitter dataset from the SNAP *ego-Twitter* collection (McAuley & Leskovec, 2012), a directed social network collected from public follow lists. Each node represents an anonymized user account, and an edge $u \to v$ indicates that user u follows user v, so in- and out-degree respectively characterize received attention and following activity. To prevent node leakage across splits, we partition the source accounts into disjoint training, validation, and test pools before extracting induced subgraphs through breadth-first expansion over both incoming and outgoing neighborhoods. This procedure yields 4,067 weakly connected graphs, divided into 3,447 training, 198 validation, and 422 test graphs. Further construction details and split-wise statistics are provided in Appendix C.2.

Together, Circuits and Twitter broaden evaluation beyond a single directed-graph family by covering complementary connectivity regimes: label-dependent engineered connectivity and heterogeneous asymmetric social structure. Twitter contains no categorical node labels and is therefore evaluated only at the structural level; node-label generation is assessed on labeled datasets.

5 Experiments

5.1 Experimental setup

Datasets We evaluate PolarFlow on the synthetic Stochastic Block Model (SBM) graphs and four real-world datasets: Visual Genome (Krishna et al., 2017), TPU Tiles (Phothilimthana et al., 2023), and our new Circuits and Twitter datasets. We follow the splits of (Carballo-Castro et al., 2026) for established benchmarks. Dataset construction, preprocessing and statistics are detailed in Appendix C.

Baselines We compare PolarFlow against an empirical maximum-likelihood (MLE) baseline over graph sizes, node types, and edge types; DiGress (Vignac et al., 2023), a discrete diffusion model; DeFoG (Qin et al., 2025), a discrete flow-matching model; and four DIRECTO variants (Carballo-Castro et al., 2026) combining DIRECTO or DIRECTO-DD with RRWP or MagLap positional encodings. For TPU Tiles, we also include the DAG-specific generators D-VAE (Zhang et al., 2019) and LayerDAG (Li et al., 2025).

Setup We parameterize the joint velocity field v_ϕ using a residual MLP with a backbone shared across its four components. We randomly initialize the model parameters and train using AdamW optimizer. For each dataset, we report the mean and standard deviation over five independent sampling runs. Implementation and hyperparameter details are provided in Appendix E.2.

Evaluation metrics Following standard protocols (Martinkus et al., 2022; Thompson et al., 2022; Bergmeister et al., 2024), we report the validity–uniqueness–novelty (V.U.N.) ratio and an average ratio based on maximum mean discrepancy (MMD). For labeled datasets, joint node–edge alignment is evaluated using Triplet MMD over dataset-specific semantic-tuple frequency distributions and

radial basis function (RBF)-kernel MMD over GNN embeddings that jointly encode node labels and directed structure. Higher V.U.N. and lower MMD-based values indicate better performance. Full definitions and dataset-specific validity criteria are provided in Appendix D.

5.2 GENERATIVE PERFORMANCE

Structural generation Table 1 reports the average MMD ratio for structural distributional alignment and V.U.N. for sample-level quality. PolarFlow is competitive with strong Directo variants on SBM and Twitter, and achieves the best average ratio on Circuits (1.70) with a perfect uniqueness–novelty score. Its largest gains occur on Visual Genome and TPU Tiles, reducing the best Directo ratios from 6.02 and 18.46 to 3.32 and 11.54, respectively. The V.U.N. gap on Visual Genome is consistent with its strict typed-edge validity criterion: because PolarFlow generates structures and labels in two stages without hard constraint enforcement, structurally well-aligned samples may still fail the graph-level test due to type incompatibilities. Overall, PolarFlow combines strong structural alignment with competitive graph-level performance under the applicable validity, uniqueness, and novelty criteria, while offering substantial training and sampling efficiency advantages over discrete baselines (Table 3). Complete descriptor-level MMD results are provided in Appendix F.

Table 1: Structural performance of directed graph generation methods (mean ± standard deviation over five sampling runs). Best results are bolded.

(a) Average structural MMD ratio (↓)

Model	Circuit	TPU Tiles	Visual Genome	SBM	Twitter
Training set	1.00	1.00	1.00	1.00	1.00
MLE	6.49 ± 0.65	95.54 ± 0.93	17.00 ± 0.60	11.60 ± 0.20	12.59 ± 0.18
D-VAE	–	OOT	–	–	–
LayerDAG	–	135.14 ± 10.23	–	–	–
DiGress	2.10 ± 0.13	27.94 ± 0.67	17.00 ± 0.60	3.90 ± 0.90	1.34 ± 0.02
DeFoG	1.90 ± 0.24	26.93 ± 0.53	10.60 ± 0.80	4.30 ± 0.80	1.48 ± 0.04
DIRECTO-DD RRWP	1.95 ± 0.44	27.02 ± 0.97	15.30 ± 0.80	1.70 ± 0.40	$\mathbf{1.14 \pm 0.21}$
DIRECTO-DD MagLap	2.06 ± 0.38	18.46 ± 0.87	7.60 ± 0.70	$\mathbf{1.50 \pm 0.40}$	1.31 ± 0.10
DIRECTO RRWP	1.90 ± 0.21	25.96 ± 0.42	12.80 ± 0.60	1.80 ± 0.50	1.15 ± 0.04
DIRECTO MagLap	1.77 ± 0.25	20.96 ± 1.35	6.02 ± 0.51	2.00 ± 0.30	1.37 ± 0.11
PolarFlow	$\mathbf{1.70 \pm 0.08}$	$\mathbf{11.54 \pm 0.68}$	$\mathbf{3.32 \pm 0.53}$	2.00 ± 0.30	1.33 ± 0.04

(b) Validity, uniqueness, and novelty (V.U.N.) (%, ↑)

Model	Circuit	TPU Tiles	Visual Genome	SBM	Twitter
Training set	0.00	0.00	0.00	0.00	0.00
MLE	$\mathbf{100.00 \pm 0.00}$	52.27 ± 2.32	90.81 ± 0.92	0.00 ± 0.00	100.00 ± 0.00
D-VAE	–	OOT	–	–	–
LayerDAG	–	$\mathbf{99.93 \pm 1.22}$	–	–	–
DiGress	98.46 ± 1.88	86.63 ± 1.06	96.59 ± 0.55	41.50 ± 5.10	100.00 ± 0.00
DeFoG	99.23 ± 1.54	79.20 ± 0.85	97.81 ± 0.62	37.00 ± 6.60	100.00 ± 0.00
DIRECTO-DD RRWP	96.17 ± 3.21	88.09 ± 2.28	99.75 ± 0.81	81.50 ± 3.20	100.00 ± 0.00
DIRECTO-DD MagLap	$\mathbf{100.00 \pm 0.00}$	81.74 ± 1.69	99.84 ± 0.06	95.50 ± 3.70	100.00 ± 0.00
DIRECTO RRWP	99.23 ± 1.53	78.34 ± 1.17	$\mathbf{99.94 \pm 0.05}$	$\mathbf{99.50 \pm 1.00}$	100.00 ± 0.00
DIRECTO MagLap	96.17 ± 3.21	82.69 ± 0.98	99.08 ± 0.71	96.50 ± 2.50	100.00 ± 0.00
PolarFlow	$\mathbf{100.00 \pm 0.00}$	82.45 ± 1.22	91.65 ± 1.04	92.50 ± 3.80	100.00 ± 0.00

Note. In (b), Circuit and Twitter report the percentage of samples that are both unique and novel, as no binary validity criterion is defined for these datasets. D-VAE and LayerDAG are DAG-specific and are evaluated only on TPU Tiles. OOT indicates that a model could not be run within a reasonable timeframe.

Node-label generation Table 2 reports joint node–edge distributional alignment on the three labeled real-world datasets. PolarFlow achieves the lowest RBF MMD on Visual Genome and remains comparable with DiGress, DeFoG, and the Directo variants on Circuits and TPU Tiles, outperforming MLE on both datasets and LayerDAG on the latter. Structurally faithful sampled adjacencies provide informative conditioning for label generation, potentially reducing train–inference mismatch and benefiting RBF MMD, whose embeddings jointly encode topology and labels. However, the remaining Triplet MMD gaps indicate that structural fidelity alone does not capture label-conditioned connectivity. Additional label-alignment results are available in Appendix F.

Ablation studies (cf. Appendix E.3) We examine two design choices supporting PolarFlow's stability and coherence: (i) applying a small diagonal perturbation to the data endpoint before polar–spectral decomposition to mitigate rank and spectral degeneracies, and (ii) jointly constructing

Table 2: Joint node–edge distributional alignment on three labeled real-world datasets.

	Circuit		TPU Tiles		Visual Genome	
Model	**Triplet MMD↓**	**RBF MMD↓**	**Triplet MMD↓**	**RBF MMD↓**	**Triplet MMD↓**	**RBF MMD↓**
Training set	0.0117	0.0918	0.00061	0.0056	0.0123	0.0257
MLE	0.035 ± 0.002	0.160 ± 0.033	0.091 ± 0.004	0.158 ± 0.013	0.330 ± 0.004	0.121 ± 0.009
D-VAE	–	–	OOT	OOT	–	–
LayerDAG	–	–	0.474 ± 0.011	0.422 ± 0.077	–	–
DiGress	0.018 ± 0.012	0.097 ± 0.011	0.005 ± 0.001	0.016 ± 0.001	0.230 ± 0.010	0.083 ± 0.012
DeFoG	0.021 ± 0.007	0.098 ± 0.005	0.004 ± 0.001	0.017 ± 0.001	$\mathbf{0.016 \pm 0.004}$	0.074 ± 0.002
DIRECTO-DD RRWP	$\mathbf{0.013 \pm 0.005}$	$\mathbf{0.087 \pm 0.018}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.012 \pm 0.001}$	0.027 ± 0.009	0.053 ± 0.011
DIRECTO-DD MagLap	0.015 ± 0.006	0.103 ± 0.018	0.004 ± 0.001	0.015 ± 0.000	0.032 ± 0.005	0.063 ± 0.011
DIRECTO RRWP	0.014 ± 0.006	0.090 ± 0.01	0.002 ± 0.000	0.013 ± 0.001	0.048 ± 0.016	0.065 ± 0.052
DIRECTO MagLap	0.016 ± 0.005	0.090 ± 0.010	0.006 ± 0.001	0.016 ± 0.001	0.035 ± 0.005	0.080 ± 0.007
PolarFlow	0.023 ± 0.009	0.108 ± 0.014	0.009 ± 0.001	0.024 ± 0.002	0.070 ± 0.007	$\mathbf{0.046 \pm 0.007}$

the four source channels from a shared noisy matrix to preserve their cross-channel coupling. The diagonal perturbation generally improves structural distributional alignment, reported in Table 11, providing empirical evidence consistent with the stability analysis in Theorem 1 and indicating that appropriately chosen endpoint regularization can improve generation quality. For source construction, deriving the four channels through the polar–spectral decomposition of a shared source matrix yields lower average MMD ratios than a factorized prior, in which the channels are sampled independently and no common matrix is decomposed at $t = 0$. Moreover, a coupled variant based on a Gaussian-only source matrix yields results close to those of the structured-template variant, suggesting that the improvement stems primarily from cross-channel coupling rather than template-specific structure, shown in Table 12. Complete formulations and results are provided in Appendix E.3.

5.3 COMPUTATIONAL EFFICIENCY

PolarFlow and discrete flow-matching methods differ in how they represent graph structure and perform sampling. Discrete methods predict dense categorical distributions over all ordered node pairs and, during generation, repeatedly evaluate transition probabilities or rates and update categorical edge states. PolarFlow instead evolves four continuous polar–spectral channels through a joint velocity field, reconstructing and discretizing the adjacency matrix only at the endpoint. To assess the practical consequences of these computational profiles, Table 3 reports per-epoch training time and per-graph sampling time. PolarFlow achieves substantially shorter training and sampling times than the evaluated discrete generative methods while maintaining comparable generation quality, an efficiency advantage for sample-intensive tasks such as large-scale evaluation and synthetic-data augmentation. Additional runtime and resource comparisons are provided in Appendix E.5.

Table 3: Comparison of training time (s/epoch), sampling time (s/graph), and storage (GB).

Method	Training Time (s)			Sampling Time (s)			Storage (GB)		
	Twitter	Circuit	TPU Tiles	Twitter	Circuit	TPU Tiles	Twitter	Circuit	TPU Tiles
Digress	21.56	1.47	16.53	1.65	1.38	1.51	8.89	7.89	9.42
DeFoG	16.85	1.13	13.27	1.80	1.91	1.79	8.96	8.15	9.13
DIRECTO-DD RRWP	29.67	1.45	22.86	1.69	1.00	1.67	18.67	17.79	19.67
DIRECTO-DD MagLap	45.19	2.09	32.39	3.77	2.17	3.41	18.63	17.79	20.07
DIRECTO RRWP	29.67	1.42	22.79	3.56	1.88	3.23	18.66	18.27	19.55
DIRECTO MagLap	44.63	2.09	32.52	7.78	6.66	6.91	18.63	18.27	19.93
Ours	3.67	1.10	3.33	0.19	0.16	0.14	1.29	1.03	1.13

6 CONCLUSION

We introduce *PolarFlow*, a continuous geometric framework for directed graph generation based on polar decomposition and truncated spectral coordinates. By jointly modeling the orthogonal and positive-semidefinite factors through four coupled channels, PolarFlow replaces edge-wise categorical generation with geometry-aware flow matching over a structured representation of the adjacency matrix. Experiments across synthetic and real-world datasets, including the newly introduced Circuit and Twitter benchmarks, demonstrate that PolarFlow achieves generative quality comparable to competitive discrete methods while requiring substantially less training and sampling time in practice. These results establish polar–spectral modeling as a promising alternative to discrete edge-wise generation and motivate its broader use in application-specific directed graph design.

AI USE STATEMENT

We acknowledge the use of large language models (LLMs) as a general-purpose assistive tool in preparing this manuscript. Specifically, LLMs were employed to aid in polishing the writing, including refining grammar, improving clarity, and enhancing fluency of expression. They were also used to help check code for potential errors. LLMs were **NOT** used for generating research ideas or producing results. All conceptual contributions, theoretical developments, experimental designs, and interpretations presented in this work are entirely the responsibility of the authors.

ETHICS STATEMENT

This research relies solely on publicly available benchmark datasets, used in accordance with their licenses, and does not involve human subjects or personally identifiable information. No private or sensitive data were accessed. The methods are intended for academic research and pose no foreseeable risks of harm or misuse. We also take care to ensure fair evaluation by following standard protocols and reporting reproducible results. The authors affirm full compliance with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our work. The main text presents the problem formulation, model design, and theoretical analysis, with complete proofs of all theorems and assumptions provided in the appendix. Detailed descriptions of datasets, preprocessing steps, and evaluation protocols are included in the experimental section and supplementary materials. Hyperparameter settings, training details, and additional ablation studies are also reported in either the Appendix or supplementary materials. To further support replication, the source code is provided in the supplementary material. Collectively, these resources enable independent researchers to reproduce both our theoretical and empirical results.

REFERENCES

P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *Proceedings of Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

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

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. In *Proceedings of International Conference on Machine Learning*, volume 213, pp. 5453–5512, 2024.

Alba Carballo-Castro, Manuel Madeira, Yiming Qin, Dorina Thanou, and Pascal Frossard. Generating directed graphs with dual attention and asymmetric encoding. In *Proceedings of International Conference on Learning Representations*, 2026.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Chun-Hui Chen and Ji-Guang Sun. Perturbation bounds for the polar factors. *Journal of Computational Mathematics*, 7(4):397–401, 1989.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *Proceedings of International Conference on Learning Representations*, 2024.

Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9 (1):1–19, 2005.

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):64, 2022.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *Proceedings of International Conference on Machine Learning*, pp. 2302–2312, 2020.

Chandler Davis and William M. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970. doi: 10.1137/0707001.

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

Xingcheng Fu, Yisen Gao, Yuecen Wei, Qingyun Sun, Hao Peng, Jianxin Li, and Xianxian Li. Hyperbolic geometric latent diffusion model for graph generation. In *Proceedings of International Conference on Machine Learning*, volume 235, pp. 14102–14124, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Proceedings of Advances in Neural Information Processing Systems*, volume 37, pp. 133345–133385, 2024.

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

Allen Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002.

Alan J. Hoffman and Helmut W. Wielandt. The variation of the spectrum of a normal matrix. *Duke Mathematical Journal*, 20(1):37–39, 1953.

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

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 2 edition, 2012.

Feng Ji and Antonio Ortega. Digraph signal processing via polar decomposition. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2026.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of International Conference on Machine Learning*, pp. 2323–2332, 2018.

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

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *Proceedings of International Conference on Machine Learning*, volume 900, pp. 22371–22405, 2024.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2017.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *Proceedings of International Conference on Machine Learning*, volume 97, pp. 3499–3508, 2019.

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

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.

Semin Kwak, Laura Shimabukuro, and Antonio Ortega. Frequency analysis and filter design for directed graphs with polar decomposition. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024.

Marc T. Law, Karsten Kreis, and Haggai Maron. Directed graph generation with heat kernels. *Transactions on Machine Learning Research*, 2025.

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.

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.

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. In *Proceedings of International Conference on Learning Representations*, 2025.

Ren-Cang Li. New perturbation bounds for the unitary polar factor. *SIAM Journal on Matrix Analysis and Applications*, 16(1):327–332, 1995.

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

Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. In *Proceedings of International Conference on Learning Representations*, 2021.

Shang Liu, Jing Wang, Wenji Fang, and Zhiyao Xie. SynCircuit: Automated generation of new synthetic RTL circuits can enable big data in circuits. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025.

Aaron Lou, Minkai Xu, and Stefano Ermon. Scaling riemannian diffusion models. In *Proceedings of Advances in Neural Information Processing Systems*, volume 36, 2023.

Shitong Luo, Chence Shi, Minkai Xu, and Jian Tang. Predicting molecular conformation via dynamic graph score matching. In *Proceedings of Advances in Neural Information Processing Systems*, volume 34, pp. 19784–19795, 2021a.

Tianze Luo, Zhanfeng Mo, and Sinno Jialin Pan. Fast graph generation via spectral diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3496–3508, 2024.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. GraphDF: A discrete flow model for molecular graph generation. In *Proceedings of International Conference on Machine Learning*, volume 139, pp. 7192–7203, 2021b.

Changsheng Ma and Xiangliang Zhang. GF-VAE: A flow-based variational autoencoder for molecule generation. In *Proceedings of ACM International Conference on Information & Knowledge Management*, pp. 1181–1190, 2021.

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

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedings of International Conference on Machine Learning*, volume 162, pp. 15159–15179, 2022.

Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Proceedings of Advances in Neural Information Processing Systems*, volume 25, pp. 539–547, 2012.

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pp. 4474–4484, 2020.

Tiago P. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

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. In *Proceedings of Advances in Neural Information Processing Systems*, 2023.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *Proceedings of International Conference on Machine Learning*, 2025.

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. In *Proceedings of the ACM Web Conference*, pp. 1475–1485. Association for Computing Machinery, 2022.

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

Antoine Siraudin and Christopher Morris. Principled latent diffusion for graphs via laplacian autoencoders. *arXiv preprint arXiv:2601.13780*, 2026.

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

Martha Takane, Saúl Bernal-González, Jesús Mauro-Moreno, Gustavo García-López, and Francisco F. De-Miguel. Directed graph theory for the analysis of biological regulatory networks. *Frontiers in Applied Mathematics and Statistics*, 11, 2025.

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W. Taylor. On evaluation metrics for graph generative models. In *Proceedings of International Conference on Learning Representations*, 2022.

Clement Vignac and Pascal Frossard. Top-n: Equivariant set and graph generation without exchangeability. In *Proceedings of International Conference on Learning Representations*, 2022.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *Proceedings of International Conference on Learning Representations*, 2023.

Antoine Wehenkel and Gilles Louppe. Graphical normalizing flows. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, volume 130, pp. 37–45, 2021.

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.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. GeoDiff: A geometric diffusion model for molecular conformation generation. In *Proceedings of International Conference on Learning Representations*, 2022.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Proceedings of Advances on Neural Information Processing Systems*, volume 2531, pp. 79704–79740, 2024.

Ling Yang, Zhilin Huang, Zhilong Zhang, Zhongyi Liu, Shenda Hong, Wentao Zhang, Wenming Yang, Bin Cui, and Luxia Zhang. Graphusion: Latent diffusion for graph generation. *IEEE Transactions on Knowledge and Data Engineering*, 36:6358–6369, 2024.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with an auto-regressive model. In *International Conference on Machine Learning*, volume 80, pp. 5694–5703, 2018.

Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-VAE: A variational autoencoder for directed acyclic graphs. In *Proceedings of Advances in Neural Information Processing Systems*, volume 142, pp. 1588–1600, 2019.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. In *Proceedings of Advances in Neural Information Processing Systems*, 2024a.

Xinyi Zhou, Xing Li, Yingzhao Lian, Yiwen Wang, Lei Chen, Mingxuan Yuan, Jianye Hao, Guangyong Chen, and Pheng-Ann Heng. Seadag: Semi-autoregressive diffusion for conditional directed acyclic graph generation. *arXiv preprint arXiv:2410.16119*, 2024b.

A Related work

Graph generative models. Autoregressive models build a graph through a sequence of node and edge decisions (You et al., 2018; Liao et al., 2019; Dai et al., 2020), while one-shot alternatives include variational autoencoders (Kipf & Welling, 2017; Simonovsky & Komodakis, 2018; Jin et al., 2018; Ma & Zhang, 2021; Vignac & Frossard, 2022), generative adversarial networks (De Cao & Kipf, 2018; Krawczuk et al., 2021), and normalizing flows (Madhawa et al., 2019; Wehenkel & Louppe, 2021; Lippe & Gavves, 2021; Luo et al., 2021b). Diffusion methods first adapted continuous dynamics to graphs (Niu et al., 2020; Luo et al., 2021a; Xu et al., 2022; Jo et al., 2022) and then moved to categorical state spaces and their continuous-time and flow-matching variants (Austin et al., 2021; Haefeli et al., 2022; Vignac et al., 2023; Jo et al., 2024; Xu et al., 2024; Siraudin et al., 2024; Campbell et al., 2024; Gat et al., 2024; Qin et al., 2025). Across these families, the generative variables are node categories and edge categories over all candidate pairs.

Spectral and geometric generation. Latent graph generators reduce the dimension of the visible state through a learned representation (Zhou et al., 2024a; Fu et al., 2024; Yang et al., 2024; Siraudin & Morris, 2026). Spectral methods instead derive coordinates from graph operators. SPECTRE generates leading Laplacian eigenpairs (Martinkus et al., 2022), and GSDM uses low-rank stochastic dynamics on graph spectra (Luo et al., 2024); both focus on undirected graphs with symmetric operators. Riemannian generative models show how conditional paths can respect non-Euclidean domains (Chen & Lipman, 2024; Lou et al., 2023). PolarFlow combines these ideas for an asymmetric adjacency matrix by separating its polar orientation and magnitude.

Directed graph generation. DAG generators usually exploit a topological ordering, including D-VAE (Zhang et al., 2019), GraphPNAS (Li et al., 2022), LayerDAG (Li et al., 2025), and SeaDAG (Zhou et al., 2024b). For general digraphs, DGDK uses a heat-kernel denoising autoencoder (Law et al., 2025), while Directo combines discrete flow matching with asymmetric positional encodings and dual attention (Carballo-Castro et al., 2026), and LG-Flow (Siraudin & Morris, 2026) uses latent diffusion. PolarFlow differs in the generative variables themselves: it transports a global polar–spectral state rather than categorical variables indexed by ordered pairs.

B Discussions and proofs

To start with, we introduce *regularity assumptions* on $\mathbf{A}_\varepsilon = \mathbf{A} + \mathbf{E}_\varepsilon = \mathbf{U}\mathbf{P}$.

For $1 \le k < N$, let $\mathcal{I}_p$ and $\mathcal{I}_u$ index the retained eigenvalues of $\mathbf{P}$ and $\mathbf{U}$, respectively. Define the spectral separations between the retained and discarded spectra of $\mathbf{P}$ and $\mathbf{U}$, respectively, by $\gamma_{p,\varepsilon} := \min_{i\in\mathcal{I}_p, j\notin\mathcal{I}_p} |\lambda_i - \lambda_j|$, and $\gamma_{u,\varepsilon} := \min_{i\in\mathcal{I}_u, j\notin\mathcal{I}_u} \left|e^{\mathrm{i}\theta_i} - e^{\mathrm{i}\theta_j}\right|$. We say that $\mathbf{A}_\epsilon$ is *spectrally regular* if the following holds:

(i) The rank margin is positive, $\alpha_\varepsilon := \sigma_{\min}(\mathbf{A}_\varepsilon) > 0$.

(ii) The relevant spectral gaps are positive, $\gamma_{p,\varepsilon} > 0$ and $\gamma_{u,\varepsilon} > 0$;

(iii) The score-selection margin is positive, $\mu_\varepsilon := s_k - s_{k+1} > 0$.

For example, if both $\mathbf{U}$ and $\mathbf{P}$ do not have repeated eigenvalues, then $\mathbf{A}_\epsilon$ is spectrally regular. In practice, the matrix $\mathbf{A}$ can be irregular, particularly when the graph is sparse. Intuitively, the purpose of continuously perturbing $\mathbf{A}$ matrix with $\mathbf{E}_\epsilon$ is to make $\mathbf{A}_\epsilon$ spectrally regular (with high probability).

Let $\mathbf{Z}_0 \in \mathcal{M}_k$ be a fixed source endpoint and let $\mathbf{Z}_1 = \Phi_k(\mathbf{A}_\epsilon) \in \mathcal{M}_k$ be a target endpoint. We say that $\mathbf{A}_\epsilon$ is *geometrically regular* (w.r.t. $\mathbf{Z}_0$) if the following holds:

(i) There exists a neighborhood $\mathcal{U} \subset \mathcal{M}_k$ of $\mathbf{Z}_1$ such that for $c \in \{p, u\}$, the Riemannian logarithm
$$(\mathbf{Q}_{c,0}, \mathbf{Q}_c) \mapsto \mathrm{Log}_{\mathbf{Q}_{c,0}}(\mathbf{Q}_c)$$
is uniquely defined and smooth on this neighborhood.

(ii) For every retained angle, the endpoint $\theta_1^{(k)}$ satisfies $\theta_1^{(k)} - \theta_0^{(k)} \notin (2\mathbb{Z}+1)\pi$.

Intuitively, spectral regularity ensures the stability of the matrix decomposition, while geometric regularity ensures the local uniqueness and smoothness of the geodesic interpolation on the underlying manifolds.

$\mathbf{A}_\epsilon$ is *regular* if it is both spectrally and geometrically regular.

Proof of Theorem 1. Let $\mathbf{A}_\varepsilon = \mathbf{U}\mathbf{P}$ and $\widetilde{\mathbf{A}}_\varepsilon = \widetilde{\mathbf{U}}\widetilde{\mathbf{P}}$ be the polar decompositions of the two regularized matrices, and set $\delta := \|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\|_2$ and $\alpha_\varepsilon = \sigma_{\min}(\mathbf{A}_\varepsilon) > 0$. We prove the result by first controlling the perturbations of the polar factors and then propagating these bounds through the retained spectral coordinates, the conditional paths, and finally the learned velocity field.

Step 1 Polar-factor stability: We first show that the polar decomposition remains unique in a sufficiently small neighborhood of $\mathbf{A}_\varepsilon$, and then control the variation of its two polar factors. Weyl's singular-value inequality (Horn & Johnson, 2012) gives $\sigma_{\min}(\widetilde{\mathbf{A}}_\varepsilon) \geq \alpha_\varepsilon - \delta$. Hence, by choosing $r < \alpha_\varepsilon/2$, every perturbation with $\delta < r$ satisfies $\sigma_{\min}(\widetilde{\mathbf{A}}_\varepsilon) \geq \frac{\alpha_\varepsilon}{2} > 0$. Consequently, both $\mathbf{A}_\varepsilon$ and $\widetilde{\mathbf{A}}_\varepsilon$ are nonsingular. Their positive polar factors are therefore positive definite, and their orthogonal polar factors are uniquely determined.

Since the matrices are $N \times N$, the spectral and Frobenius norms satisfy $\|\mathbf{X}\|_F \leq \sqrt{N}\|\mathbf{X}\|_2$. The Frobenius-norm perturbation bound of (Chen & Sun, 1989) for positive polar factors thus gives

$$\left\|\widetilde{\mathbf{P}} - \mathbf{P}\right\|_F \leq \sqrt{2}\left\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\right\|_F \leq \sqrt{2N}\,\delta. \tag{16}$$

For the orthogonal polar factors, Theorem 1 of Li (1995) yields

$$\left\|\widetilde{\mathbf{U}} - \mathbf{U}\right\|_F \leq \frac{2\left\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\right\|_F}{\sigma_{\min}(\mathbf{A}_\varepsilon) + \sigma_{\min}(\widetilde{\mathbf{A}}_\varepsilon)} \leq \frac{4\sqrt{N}}{3\alpha_\varepsilon}\,\delta. \tag{17}$$

(16) and (17) show that both polar factors vary Lipschitz-continuously with the regularized matrix. We next transfer the polar-factor bounds to the four PolarFlow endpoint channels.

Step 2 Stability of the retained spectral representation. Having controlled the polar factors, we now transfer these bounds to the retained eigenvalues, eigenangles, and spectral frames. Using $\|\mathbf{X}\|_2 \leq \|\mathbf{X}\|_F$ and the estimates from Step 1, choose $r > 0$ sufficiently small so that every perturbation satisfying $\delta < r$ obeys

$$\begin{aligned}\left\|\widetilde{\mathbf{P}} - \mathbf{P}\right\|_2 \leq \left\|\widetilde{\mathbf{P}} - \mathbf{P}\right\|_F \leq \sqrt{2N}\,\delta < \frac{\gamma_{p,\varepsilon}}{2},\\ \left\|\widetilde{\mathbf{U}} - \mathbf{U}\right\|_2 \leq \left\|\widetilde{\mathbf{U}} - \mathbf{U}\right\|_F \leq \frac{4\sqrt{N}}{3\alpha_\varepsilon}\,\delta < \frac{\gamma_{u,\varepsilon}}{2}.\end{aligned} \tag{18}$$

These inequalities ensure that the retained and discarded parts of each spectrum remain separated under the perturbation. Consequently, the retained invariant subspaces of $\mathbf{P}$ and $\mathbf{U}$ can be matched consistently with their perturbed counterparts, and their dimensions remain unchanged.

For the positive polar factor, recall that $\mathbf{P} = (\mathbf{A}_\varepsilon^*\mathbf{A}_\varepsilon)^{1/2}$. Its eigenvalues are therefore the singular values of $\mathbf{A}_\varepsilon$. Under the common ordering of the retained eigenvalues, Mirsky's singular-value perturbation theorem (Mirsky, 1960) gives

$$\|\widetilde{\boldsymbol{\lambda}}_p^{(k)} - \boldsymbol{\lambda}_p^{(k)}\|_2 \leq \|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\|_F \leq \sqrt{N}\delta$$

Thus, the retained eigenvalue channel is Lipschitz-continuous with respect to the matrix perturbation. The same spectral separation also stabilizes the associated retained invariant subspace. Indeed, the separation between the perturbed retained spectrum and the complementary spectrum of $\mathbf{P}$ is bounded below by

$$\gamma_{p,\varepsilon} - \|\widetilde{\mathbf{P}} - \mathbf{P}\|_2 > \frac{\gamma_{p,\varepsilon}}{2}.$$

To remove the intrinsic basis ambiguity, we align the perturbed frame separately within each retained eigenspace. Let $\nu_1, \ldots, \nu_m$ denote the distinct retained eigenvalues of $\mathbf{P}$, with multiplicities

$m_1, \ldots, m_m$, and let $\mathbf{Q}_{p,\ell}$ and $\widetilde{\mathbf{Q}}_{p,\ell}$ be orthonormal bases for the corresponding unperturbed and perturbed spectral subspaces. For each ℓ, choose

$$\mathbf{R}^{\star}_{p,\ell} \in \operatorname*{arg\,min}_{\mathbf{R}\in \mathrm{O}(m_\ell)} \left\| \widetilde{\mathbf{Q}}_{p,\ell}\mathbf{R} - \mathbf{Q}_{p,\ell} \right\|_F . \tag{19}$$

For a simple eigenvalue, this alignment only fixes the eigenvector sign; for a repeated eigenvalue, it selects the closest orthonormal basis within the corresponding spectral subspace. The positive spectral gaps ensure that the perturbed subspaces have the same dimensions as their unperturbed counterparts. Moreover, for sufficiently small δ, their principal angles are smaller than $\pi/2$, so the cross-Gram matrices $\widetilde{\mathbf{Q}}^{*}_{p,\ell}\mathbf{Q}_{p,\ell}$ are nonsingular and the minimizing orthogonal transformations in (19) are unique and vary continuously with the perturbation. We henceforth replace each $\widetilde{\mathbf{Q}}_{p,\ell}$ by $\widetilde{\mathbf{Q}}_{p,\ell}\mathbf{R}^{\star}_{p,\ell}$ and retain the notation $\widetilde{\mathbf{Q}}_{p}$ for the resulting aligned frame.

With this convention, applying the Davis–Kahan theorem (Davis & Kahan, 1970) and the standard relation between subspace distance and optimally aligned frame distance gives

$$\|\widetilde{\mathbf{Q}}_p - \mathbf{Q}_p\|_F \le \frac{2\sqrt{2}}{\gamma_{p,\varepsilon}} \|\widetilde{\mathbf{P}} - \mathbf{P}\|_F \le \frac{4\sqrt{N}}{\gamma_{p,\varepsilon}} \delta$$

Therefore, both retained channels associated with $\mathbf{P}$ are locally Lipschitz-stable.

We now consider the orthogonal polar factor. Let $z_i = e^{\mathrm{i}\theta_i}$ and $\widetilde{z}_i = e^{\mathrm{i}\widetilde{\theta}_i}$ denote the eigenvalues of $\mathbf{U}$ and $\widetilde{\mathbf{U}}$, respectively. Since both matrices are normal, the Hoffman–Wielandt theorem (Hoffman & Wielandt, 1953) permits the perturbed eigenvalues such that

$$\left(\sum_{i=1}^{N} |\widetilde{z}_i - z_i|^2 \right)^{1/2} \le \|\widetilde{\mathbf{U}} - \mathbf{U}\|_F .$$

Moreover, because $\|\widetilde{\mathbf{U}} - \mathbf{U}\|_F < \gamma_{u,\varepsilon}/2$, the retained and discarded spectra remain separated. Restricting the above inequality to the retained eigenvalues therefore gives

$$\left(\sum_{i\in\mathcal{I}_u} |\widetilde{z}_i - z_i|^2 \right)^{1/2} \le \|\widetilde{\mathbf{U}} - \mathbf{U}\|_F \le \frac{4\sqrt{N}}{3\alpha_\varepsilon}\, \delta.$$

Let $\mathrm{wrap}(x) \in [-\pi, \pi]$ denote the principal value of x module 2π. For each $i \in \mathcal{I}_u$, we have

$$|\widetilde{z}_i - z_i| = 2 \left| \sin\left(\frac{\mathrm{wrap}(\widetilde{\theta}_i - \theta_i)}{2} \right) \right|.$$

since $2\left|\sin\left(\frac{x}{2}\right)\right| \ge \frac{2|x|}{\pi}$ for $x \in [-\pi, \pi]$, we have

$$\left| \mathrm{wrap}\left(\widetilde{\theta}_i - \theta_i \right) \right| \le \frac{\pi}{2} |\widetilde{z}_i - z_i|$$

for each $i \in \mathcal{I}_u$. Applying this inequality over the retained indices gives

$$\left\| \mathrm{wrap}\left(\widetilde{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k \right) \right\|_2 \le \frac{\pi}{2} \left(\sum_{i\in\mathcal{I}_u} |\widetilde{z}_i - z_i|^2 \right)^{1/2} \le \frac{2\pi\sqrt{N}}{3\alpha_\varepsilon}\, \delta. \tag{20}$$

The same spectral separation also stabilizes the associated retained frame. In particular, the perturbed retained spectrum remains separated from the complementary spectrum of $\mathbf{U}$ by at least

$$\gamma_{u,\varepsilon} - \left\| \widetilde{\mathbf{U}} - \mathbf{U} \right\|_2 > \frac{\gamma_{u,\varepsilon}}{2}. \tag{21}$$

Applying the invariant-subspace perturbation bound for normal matrices and using the consistent frame alignment assumed in the theorem yields

$$\left\| \widetilde{\mathbf{Q}}_u - \mathbf{Q}_u \right\|_F \le \frac{2\sqrt{2}}{\gamma_{u,\varepsilon}} \left\| \widetilde{\mathbf{U}} - \mathbf{U} \right\|_F \le \frac{8\sqrt{2N}}{3\alpha_\varepsilon \gamma_{u,\varepsilon}}\, \delta. \tag{22}$$

Finally, we establish the stability of the score-based mode selection. Under the locally consistent frame convention assumed above, each score $s_i = 2\left|\sin\left(\frac{\theta_i}{2}\right)\right| \mathbf{q}_{u,i}^* \mathbf{P} \mathbf{q}_{u,i}$ is locally Lipschitz in its spectral arguments. Indeed, $2|\sin(\theta/2)|$ is 1-Lipschitz with respect to the wrapped angular distance, and $\|\mathbf{q}_i\|_2 = 1$. Consequently,

$$\begin{aligned} |\widetilde{s}_i - s_i| \le & \left\|\widetilde{\mathbf{P}}\right\|_2 \left|\mathrm{wrap}\left(\widetilde{\theta}_i - \theta_i\right)\right| \\ & + 2\left\|\widetilde{\mathbf{P}} - \mathbf{P}\right\|_2 + 4\|\mathbf{P}\|_2 \|\widetilde{\mathbf{q}}_i - \mathbf{q}_i\|_2 . \end{aligned} \tag{23}$$

The spectral estimates established above therefore imply that, within a sufficiently small neighborhood of $\mathbf{A}_\varepsilon$, there exists a finite constant L_s, independent of δ, such that

$$\max_i |\widetilde{s}_i - s_i| \le L_s \delta. \tag{24}$$

Recall that the positive selection margin is $\mu_\varepsilon = s_k - s_{k+1} > 0$ where the scores are ordered in nonincreasing order. Choose the neighborhood radius r to additionally satisfy $r < \frac{\mu_\varepsilon}{2L_s}$. Then, whenever $\delta < r$, every retained index i and discarded index j satisfy

$$\widetilde{s}_i - \widetilde{s}_j \ge s_i - s_j - 2L_s\delta \ge \mu_\varepsilon - 2L_s\delta > 0.$$

Thus, the perturbation does not change the top-k index set.

With the top-k index set preserved and the spectral frames consistently aligned, the preceding channel-wise bounds yield

$$d_\Phi\left(\Phi_k(\mathbf{A}_\varepsilon), \Phi_k(\widetilde{\mathbf{A}}_\varepsilon)\right) \le C\left\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\right\|_2 . \tag{25}$$

Here, $C < \infty$ depends on N, α_ε, $\gamma_{p,\varepsilon}$, and $\gamma_{u,\varepsilon}$, but not on the perturbation magnitude. This proves the claimed local Lipschitz stability of the regularized PolarFlow endpoint.

Step 3: Stability of the conditional velocity targets. Let C denote the endpoint-stability constant established above. Geometric regularity implies that geodesic interpolation on the Stiefel manifolds or on the high-dimensional torus avoids the cut locus of the Log function or the discontinuity of the wrap function. Therefore, in a sufficiently small neighborhood of $\mathbf{Z}_1$, there exist finite constants L_Z and $L_{\dot{Z}}$ such that

$$\sup_{t\in[0,1]} \left\|\mathbf{Z}_t(\widetilde{\mathbf{A}}_\varepsilon) - \mathbf{Z}_t(\mathbf{A}_\varepsilon)\right\| \le L_Z\, d_\Phi\left(\Phi_k(\mathbf{A}_\varepsilon), \Phi_k(\widetilde{\mathbf{A}}_\varepsilon)\right), \tag{26}$$

$$\sup_{t\in[0,1]} \left\|\dot{\mathbf{Z}}_t(\widetilde{\mathbf{A}}_\varepsilon) - \dot{\mathbf{Z}}_t(\mathbf{A}_\varepsilon)\right\| \le L_{\dot{Z}}\, d_\Phi\left(\Phi_k(\mathbf{A}_\varepsilon), \Phi_k(\widetilde{\mathbf{A}}_\varepsilon)\right). \tag{27}$$

Choose r sufficiently small that both conditional paths remain in the neighborhoods on which these Lipschitz constants apply. Combining (27) with the endpoint bound gives

$$\sup_{t\in[0,1]} \left\|\dot{\mathbf{Z}}_t(\widetilde{\mathbf{A}}_\varepsilon) - \dot{\mathbf{Z}}_t(\mathbf{A}_\varepsilon)\right\| \le L_{\dot{Z}} C\left\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\right\|_2 . \tag{28}$$

Thus, the conditional velocity targets vary Lipschitz-continuously with the regularized input matrix.

Step 4: Stability of the learned velocity field. We now transfer the stability of the conditional velocity targets to the learned velocity field. For any regularized matrix $\mathbf{A}'_\varepsilon$ in the neighborhood considered above, recall the velocity approximation error along its conditional path can be expressed as

$$\boldsymbol{e}_\phi(\mathbf{A}'_\varepsilon, t) := \widehat{\mathbf{v}}_\phi\left(\mathbf{Z}_t(\mathbf{A}'_\varepsilon), t\right) - \dot{\mathbf{Z}}_t(\mathbf{A}'_\varepsilon). \tag{29}$$

Assume that this approximation error varies locally Lipschitz-continuously with the regularized matrix, uniformly over $t \in [0,1]$. That is, there exists a finite constant L_e such that

$$\sup_{t\in[0,1]} \left\|\boldsymbol{e}_\phi(\widetilde{\mathbf{A}}_\varepsilon, t) - \boldsymbol{e}_\phi(\mathbf{A}_\varepsilon, t)\right\| \le L_e\left\|\widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon\right\|_2 . \tag{30}$$

By (29), for every $t \in [0,1]$,

$$\begin{aligned} & \widehat{\mathbf{v}}_\phi\left(\mathbf{Z}_t(\widetilde{\mathbf{A}}_\varepsilon), t\right) - \widehat{\mathbf{v}}_\phi\left(\mathbf{Z}_t(\mathbf{A}_\varepsilon), t\right) \\ & \quad = \dot{\mathbf{Z}}_t(\widetilde{\mathbf{A}}_\varepsilon) - \dot{\mathbf{Z}}_t(\mathbf{A}_\varepsilon) + \boldsymbol{e}_\phi(\widetilde{\mathbf{A}}_\varepsilon, t) - \boldsymbol{e}_\phi(\mathbf{A}_\varepsilon, t). \end{aligned} \tag{31}$$

Taking norms and the supremum over t, and then applying (28) and (30), gives

$$\begin{aligned}
&\sup_{t\in[0,1]} \left\| \widehat{\mathbf{v}}_\phi \left(\mathbf{Z}_t(\widetilde{\mathbf{A}}_\varepsilon), t \right) - \widehat{\mathbf{v}}_\phi \left(\mathbf{Z}_t(\mathbf{A}_\varepsilon), t \right) \right\| \\
&\quad \le \sup_{t\in[0,1]} \left\| \dot{\mathbf{Z}}_t(\widetilde{\mathbf{A}}_\varepsilon) - \dot{\mathbf{Z}}_t(\mathbf{A}_\varepsilon) \right\| + \sup_{t\in[0,1]} \left\| \boldsymbol{e}_\phi(\widetilde{\mathbf{A}}_\varepsilon, t) - \boldsymbol{e}_\phi(\mathbf{A}_\varepsilon, t) \right\| \\
&\quad \le (L_{\dot{Z}} C_\Phi + L_e) \left\| \widetilde{\mathbf{A}}_\varepsilon - \mathbf{A}_\varepsilon \right\|_2 . \qquad (32)
\end{aligned}$$

Therefore, taking $C := L_{\dot{Z}} C_\Phi + L_e$ establishes (15) and completes the proof. □

C Dataset descriptions

C.1 Stochastic Block Model (Holland et al., 1983)

We adopt the SBM sampling protocol of Carballo-Castro et al. (2026) without modification, so that our numbers remain directly comparable to the results reported there. Every synthetic graph in our experiments is represented as a binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n\times n}$, with $\boldsymbol{A}_{ij} = 1$ indicating a directed edge from node i to node j. Sampled matrices are then converted into whichever input format the corresponding model expects.

A graph is drawn by first partitioning its nodes into K blocks of sizes $\{n_1, \dots, n_K\}$, where K is sampled uniformly from $\{2, \dots, 5\}$ and each block size is sampled uniformly from $\{20, \dots, 40\}$. Writing $z_i \in \{1, \dots, K\}$ for the block containing node i, each ordered pair is then assigned an edge independently,

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

with $P \in [0,1]^{K\times K}$ collecting the within- and between-block edge probabilities. Because the two orientations of a pair are drawn separately, the sampled graph is directed. We use the same setting as the reference protocol,

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

which yields graphs whose edges concentrate inside blocks while leaving a sparse set of cross-block connections. We sample 200 graphs and split them in $128/32/40$ training, validation, and test sets. Further dataset statistics can be found in Table 4.

Table 4: SBM dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	#Train	#Val	#Test
SBM	44	175	106	340	3008	1426	128	32	40

C.2 Twitter

To construct the Twitter dataset, we first remove self-loops and collapse duplicate edges in the source network. We then partition the source nodes into disjoint training, validation, and test pools before subgraph extraction, ensuring that no account appears in more than one split. Within each pool, candidate induced subgraphs are obtained through breadth-first expansion over both incoming and outgoing neighborhoods, after which only the largest weakly connected component is retained. We keep candidates containing between 64 and 80 nodes, with at least as many edges as nodes and edge reciprocity no greater than 0.42. To limit redundancy within each split, we further discard a candidate if the Jaccard similarity between its node set and that of any previously retained graph exceeds 0.5. Statistics of the resulting dataset are reported in Table 5.

This procedure yields 4,067 connected graphs with an overall mean directed density of 0.118, where density is computed as $|\mathcal{E}|/(|\mathcal{V}|(|\mathcal{V}|-1))$. Split-wise and aggregate statistics are reported in Table 5. Twitter graphs exhibit heterogeneous connectivity on both sides of a directed edge: the mean in- and out-degree Gini coefficients are approximately 0.52 and 0.48, reflecting variations in both account popularity and following activity. Their mean reciprocity is 0.357, compared with 0.166 under an

in/out-degree-preserving null model, indicating that mutual following remains selective but occurs more frequently than expected from the degree sequences alone.

Every retained graph contains at least one directed cycle, and approximately 74% of nodes participate in directed cycles. The graphs also exhibit substantial local clustering and community organization, with mean directed clustering and Louvain modularity of 0.397 and 0.301, respectively. Together, these statistics characterize a naturally evolving social-network regime in which asymmetric connectivity, selective reciprocity, directed cycles, degree heterogeneity, and community structure coexist. Since the processed graph representation contains neither node features nor categorical node labels, Twitter is used exclusively as a structure-only benchmark, with generation and evaluation performed on binary directed adjacency matrices.

Table 5: Twitter dataset statistics. It contains much denser graphs compared with existing datasets TPU Tiles and Visual Genome.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Overall	64	80	71.4	108	2,152	591.3	4,067
Train	64	80	71.4	108	1,959	545.8	3,447
Validation	64	80	70.9	157	1,776	682.5	198
Test	64	80	71.6	125	2,152	920.5	422

C.3 CIRCUIT

Each node represents a primitive cell instance, and its categorical label specifies one of seven cell types appearing in the retained graphs: lookup table (LUT), flip-flop (FD), multiplexer (MUX), constant driver (CONST), carry-chain cell (CARRY), memory cell (MEMORY), or input/output buffer (IOBUF). A directed edge $u \to v$ indicates that a signal produced by cell u drives an input of cell v; hence, edge orientation follows signal flow from driver to load. Parallel pin-level connections between the same pair of cells are represented by a single binary edge, and self-loops are removed. All graphs in the Circuits dataset are weakly connected and sparse, with a mean directed density of 0.044. The node-label distribution is highly imbalanced, with LUT and FD cells together accounting for 91.6% of all nodes. Split-wise and aggregate dataset statistics are reported in Table 6.

Circuit graphs represent engineered systems whose connectivity is constrained by hardware functionality. In particular, a cell's fan-in is limited by its number of input pins, whereas its output may drive many downstream cells, resulting in substantially greater out-degree than in-degree heterogeneity. Directed dependencies are rarely mutual, with an average reciprocity of only 0.012. Approximately 42.7% of the graphs are acyclic; in every remaining graph, each directed cycle contains at least one state-holding cell, reflecting the absence of purely combinational feedback loops in synchronous circuit designs. Connectivity is also strongly conditioned on cell type: only 20 of the 49 possible ordered pairs of the seven observed categories occur, and five pairs (i.e., FD→LUT, LUT→LUT, LUT→FD, CARRY→LUT, and LUT→CARRY), accounting for approximately 86.4% of all edges. Circuits therefore provides a benchmark for modeling constrained and task-specific directed connectivity, where successful generation requires capturing asymmetric signal dependencies and label-conditioned wiring rules rather than only matching aggregate graph statistics.

Table 6: Circuits dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Overall	30	60	44.1	40	193	80.5	318
Train	30	60	43.8	40	193	81.7	251
Validation	30	59	44.7	44	139	73.4	41
Test	30	60	45.3	44	165	80.6	26

C.4 TPU TILES (PHOTHILIMTHANA ET AL., 2023)

Each graph in this collection is a directed acyclic computation graph extracted from a machine learning workload, where nodes are tensor operations and edges carry the data dependencies between them; a

single graph corresponds to one unit of computation, such as a training epoch or an inference pass. A datapoint pairs such a graph with a compiler configuration and records the measured execution time obtained under that configuration, giving $6{,}301$ datapoints in total. The underlying workloads span a range of architecture families, including ResNets, EfficientNets, Masked R-CNNs, and Transformers, so the collection exhibits considerable structural heterogeneity: graph size ranges from a pair of operations to roughly 400 nodes, while the average graph has only 41 nodes and 43 edges. This combination of directedness, acyclicity, and heavy-tailed size makes it a useful test of whether a generator can capture engineered functional dependencies rather than the homogeneous structure of synthetic families.

We adopt the preprocessing pipeline and predefined training/validation/test split of Li et al. (2025). The original collection has a highly dispersed size distribution, with graph sizes ranging from 2 to 394 nodes. Although 90% of the graphs contain at most 80 nodes, the sparsely populated upper tail extends considerably further, with 1% containing more than 113 nodes. At the other extreme, 26% of the graphs contain fewer than 20 nodes and 8% contain fewer than 10. These small graphs provide limited structural information and can yield less stable spectral, wavelet, and clustering statistics. We therefore retain graphs containing between 30 and 80 nodes. The filter is applied independently within each split, preserving the original split assignments. This restriction removes both size extremes and confines graph sizes to a range spanning less than a factor of three, allowing the evaluation to focus on structural fidelity within a representative size regime rather than on behavior at extreme graph sizes. The resulting subset contains $3{,}183$ graphs, comprising $2{,}539$ training, 318 validation, and 326 test graphs, with averages of 50.1 nodes and 53.0 edges per graph. All methods are trained and evaluated on the same filtered subset. Per-split statistics are given in Table 7.

Table 7: TPU Tiles dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Overall	30	80	50	29	116	53	3183
Train	30	80	50	29	116	53.0	2539
Validation	30	80	52	29	102	55.2	318
Test	30	80	48	29	102	50.9	326

C.5 Visual Genome (Krishna et al., 2017)

Visual Genome annotates over $100{,}000$ images with objects, their attributes, and the relationships between them, and is a standard benchmark for scene graph generation, visual question answering, and grounded language understanding. Each annotated scene induces a directed graph in which the three annotation types become typed nodes, labelled by their text, and the visual semantics fix the orientation of every edge: objects point to their attributes and to the relationships they participate in, relationships point back to the objects they take as arguments, and attributes are sinks. The resulting graphs are therefore directed and partly cyclic, with 119 of them ($\sim$37.5%) containing at least one cycle, which makes the dataset a natural test case for methods that cannot assume a topological ordering.

We take the graph construction, vocabulary filtering, and splits directly from Carballo-Castro et al. (2026), so that our results are comparable to those reported there. In brief, that pipeline retains scenes with 20–40 nodes and at least 25 edges, and restricts node labels to the 20 most frequent objects, relationships, and attributes, giving 60 node classes in total. Split statistics are reported in Table 8; the full label vocabulary is given in the original work.

Table 8: Visual Genome dataset statistics.

Dataset	Min. nodes	Max. nodes	Avg. nodes	Min. edges	Max. edges	Avg. edges	# Graphs
Overall	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

D FURTHER DETAILS ON EVALUATION METRICS

D.1 VALIDITY, UNIQUENESS, AND NOVELTY.

Validity is evaluated using dataset-specific criteria. For synthetic datasets, we test whether generated graphs are consistent with the underlying generative distribution; for DAG datasets, validity requires acyclicity. Accordingly, on TPU Tiles, we report the fraction of acyclic samples, whereas on Visual Genome, we verify typed structural constraints requiring edges to point from objects to attributes or relationship nodes, and from relationship nodes to objects. We do not define validity for Circuits or Twitter because neither representation admits a complete binary criterion: circuit functionality cannot be established from extracted cell-level subgraphs, while social-network plausibility is inherently distributional. Their fidelity is instead evaluated through distributional graph statistics. For all datasets, uniqueness denotes the fraction of generated graphs that are distinct up to isomorphism, while novelty denotes the fraction not present in the training set. The V.U.N. ratio measures the proportion of samples that are simultaneously valid, unique, and novel and is reported only for datasets with a defined validity criterion.

D.1.1 VALIDITY METRICS

Stochastic block model Following Carballo-Castro et al. (2026), we assess whether a generated graph is consistent with the target directed SBM through its blockwise connection probabilities. Given a graph $G = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$, we first infer a partition into B non-empty blocks using the minimum-description-length method of Peixoto (2014):

$$\widehat{z} = \arg \min_{z:\mathcal{V}\to 1,\dots,B} \mathcal{L}(G, z),$$

where $\mathcal{L}(G, z)$ denotes the description length. The inferred partition can be further refined through T MCMC steps. Let N_i be the number of nodes assigned to block i and E_{ij} the number of directed edges from block i to block j. The empirical within- and between-block connection probabilities are

$$\widehat{p}_{\text{intra},i} = \frac{E_{ii}}{N_i(N_i-1)+\varepsilon}, \qquad \widehat{p}_{\text{inter},ij} = \frac{E_{ij}}{N_i N_j+\varepsilon}, \quad i \neq j, \tag{33}$$

where $\varepsilon = 10^{-6}$ ensures numerical stability. We compare these estimates with the target probabilities p_{intra} and p_{inter} using

$$W_{ii} = \frac{(\widehat{p}_{\text{intra},i} - p_{\text{intra}})^2}{\widehat{p}_{\text{intra},i}(1-\widehat{p}_{\text{intra},i})+\varepsilon}, \qquad W_{ij} = \frac{(\widehat{p}_{\text{inter},ij} - p_{\text{inter}})^2}{\widehat{p}_{\text{inter},ij}(1-\widehat{p}_{\text{inter},ij})+\varepsilon}, \quad i \neq j. \tag{34}$$

Each statistic is converted into a block-pair compatibility value using a chi-squared reference distribution with one degree of freedom,

$$q_{ij} = 1 - F_{\chi^2_1}(W_{ij}), \qquad s_{\text{SBM}}(G) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} q_{ij}. \tag{35}$$

The resulting graph-level score averages agreement across all ordered block pairs, with larger values indicating closer conformity to the prescribed SBM parameters.

TPU Tiles For the TPU Tiles dataset, validity is determined by acyclicity, consistent with its representation of computational pipelines as directed acyclic graphs (DAGs). We report the proportion of generated graphs containing no directed cycles.

Visual Genome Following Carballo-Castro et al. (2026), we evaluate the typed structural constraints of Visual Genome scene graphs, which contain object, relationship, and attribute nodes. Only three directed edge types are permitted: object-to-relationship, relationship-to-object, and object-to-attribute. We adapt the original binary graph-level criterion by evaluating these constraints at the edge level. Under the binary formulation, a single violating edge makes the entire graph invalid; this all-or-nothing criterion becomes increasingly stringent as the number of edges grows and provides little separation among models on graphs of this size. We therefore define

$$\text{Validity}(G) = \frac{\text{number of type-valid directed edges in } G}{|\mathcal{E}|}. \tag{36}$$

where $|\mathcal{E}|$ denotes the total number of directed edges in G. For each sampling run, we average this score over the generated graphs and report its mean and standard deviation across five runs. Higher values indicate greater compliance with the scene-graph typing rules, and every reference graph has an edge-validity score of one by construction.

D.2 Structural distributional alignment

To assess how closely generated graphs match the data distribution, we compute the Maximum Mean Discrepancy (MMD) (Martinkus et al., 2022; Thompson et al., 2022) between generated and test graphs over in- and out-degree distributions, directed clustering coefficients, spectral, and wavelet features. Spectral features are computed using Chung's directed Laplacian (Chung, 2005), which is computationally simpler than magnetic-Laplacian alternatives. For each descriptor, we normalize the generated–test MMD by the corresponding train–test MMD and average the resulting ratios across descriptors. Lower values indicate better alignment. Descriptor-level results are provided below.

Maximum mean discrepancy metrics Following Martinkus et al. (2022) and its extension to directed graphs by Carballo-Castro et al. (2026), we evaluate structural distributional alignment using Maximum Mean Discrepancy (MMD) over five descriptors: out-degree, in-degree, directed clustering, spectrum, and wavelet features. In- and out-degree distributions are evaluated separately to capture asymmetric connectivity. For the clustering descriptor, we compute the distribution of local directed clustering coefficients using the `clustering()` function in `NetworkX`, which supports directed graphs. For the spectral and wavelet descriptors, we replace the conventional undirected Laplacian with Chung's directed Laplacian (Chung, 2005), which derives spectral information from directed random-walk dynamics while remaining computationally simpler than magnetic-Laplacian alternatives.

For each descriptor, we compute the MMD between the generated and test sets and normalize it by the corresponding MMD between the training and test sets. The reported average ratio is the mean of the five normalized scores. Lower values indicate closer structural alignment, while a ratio of one corresponds to the train–test baseline. We omit the orbit-count metric used in undirected benchmarks because its standard ORCA-based implementation does not support directed graphlets, and graph symmetrization would discard edge-direction information.

D.3 Joint node-edge distributional metrics

Structural statistics alone cannot determine whether generated graphs preserve the dependence between categorical node labels and directed connectivity. For datasets with categorical node labels, we extend the evaluation beyond structural statistics to assess both node-label marginals and their dependence on directed connectivity. First, node-label MMD compares graph-level label histograms to assess alignment of marginal categorical proportions. Second, we extract dataset-specific semantic tuples, such as $(\text{object}, \text{relation}, \text{object})$, and report precision and recall to measure the accuracy and coverage of typed connectivity patterns relative to the test set. We additionally compute Triplet MMD between tuple-frequency histograms to quantify their distributional agreement. Finally, following Thompson et al. (2022), we compute RBF-kernel MMD over GNN-derived graph embeddings that jointly encode node labels and structure, thereby measuring global attributed-graph similarity. These metrics are reported only for datasets with categorical node labels.

Node-label distribution MMD. For each graph, we construct a normalized histogram over node categories and compute the MMD between the histograms of generated and test graphs. This metric measures whether the generated graphs reproduce the distribution of node-type proportions in the reference data.

Multiplicity-aware triplet-based precision and recall. We build on the Triplet Precision and Recall metrics of Carballo-Castro et al. (2026), extending them to account for semantic-tuple multiplicity. For Visual Genome, we extract an $(\text{object}, \text{attribute})$ pair for each attribute node and a $(\text{subject}, \text{relationship}, \text{object})$ triplet for each relationship node. For TPU Tiles and Circuits, each directed edge $u \to v$ contributes $(\ell(u), \texttt{connects}, \ell(v))$. For each tuple type, we compute its average occurrence rate per graph in the generated and test sets. Its matched contribution is the

smaller of these two rates, and summing these contributions over all tuple types gives the total number of matched tuple occurrences. Precision and recall are then defined as

$$\text{Precision} = \frac{\text{matched tuple occurrences}}{\text{generated tuple occurrences}}, \qquad \text{Recall} = \frac{\text{matched tuple occurrences}}{\text{test tuple occurrences}}.$$

Higher precision indicates that a larger proportion of generated tuple occurrences is matched by the test graphs, whereas higher recall indicates that a larger proportion of test tuple occurrences is recovered by the generated graphs. Both metrics range from zero to one, with higher values indicating better agreement, and both equal one only when every tuple type has the same average per-graph occurrence rate in the generated and test graphs.

Triplet MMD. Using the dataset-specific semantic tuples defined above, we construct tuple-frequency representations for the generated and test graphs and compute the MMD between the resulting distributions. This metric evaluates whether label-conditioned connectivity patterns occur with similar relative frequencies in the generated and reference graphs, complementing triplet-based precision and recall, which primarily measure tuple overlap.

Embedding-based RBF MMD. Following Thompson et al. (2022), we obtain graph-level representations using a fixed GNN encoder that incorporates both node labels and directed structure. We then compute MMD with an RBF kernel between the embeddings of generated and test graphs, providing a global measure of joint node–edge distributional alignment.

E Experimental details

E.1 Details on baselines

Maximum likelihood estimation (MLE). Following Carballo-Castro et al. (2026), we include a simple count-based baseline that fits three empirical distributions to the training split and samples from them independently. The first is the distribution over graph size, the second is the marginal over node classes, and the third is the edge-type distribution conditioned on the classes of the two endpoints. A sample is then drawn by first choosing n, then drawing each node class independently, and finally drawing an edge type for every ordered pair conditioned on its endpoint classes. The baseline therefore reproduces first-order statistics by construction while ignoring any higher-order structure.

D-VAE (Zhang et al., 2019). D-VAE encodes and decodes DAGs with an asynchronous message-passing scheme: rather than updating all nodes simultaneously as a standard GNN would, it processes them one at a time in topological order, so that the representation reflects the computation the graph performs and not only its connectivity. Generation mirrors this order, adding nodes sequentially with all new edges oriented toward the node just introduced, which guarantees acyclicity; inputs must be topologically sorted beforehand. We follow the setup of Carballo-Castro et al. (2026), using the authors' released implementation with the hyperparameters recommended in the original paper and converting generated graphs back to our evaluation format. On TPU Tiles, however, the size of the dataset and of its largest graphs made training impractical within our compute budget.

LayerDAG (Li et al., 2025). LayerDAG recasts a DAG as a sequence of bipartite graphs, giving a layer-wise tokenization over which generation proceeds autoregressively; within each step, a diffusion process models the dependencies among nodes that are mutually incomparable, that is, not joined by any directed path. As with D-VAE, inputs must be topologically sorted. We use the authors' released implementation with its default configurations, including the per-component epoch budgets.

DiGress (Vignac et al., 2023). We benchmark against the directed variant of DiGress implemented by Carballo-Castro et al. (2026), obtained by removing the symmetrization operations so that the two orientations of a node pair are modeled separately. In the terminology of that work, this configuration is Directo-DD without dual attention and with Laplacian positional encodings.

DeFoG (Qin et al., 2025). Likewise, we use the directed variant of DEFOG from Carballo-Castro et al. (2026), again obtained by dropping the symmetrization operations. This configuration corresponds to DIRECTO without dual attention and with RRWP positional encoding.

Directo (Carballo-Castro et al., 2026). DIRECTO is the closest baseline to our setting: it applies discrete flow matching directly to general directed graphs, combining dual attention over incoming and outgoing neighbourhoods with asymmetric positional encodings. We benchmark against both reported variants, DIRECTO-DD and DIRECTO, and for each we run the two positional encodings used in the original paper, RRWP and the magnetic Laplacian (MagLap). All four configurations use the authors' released implementation and their published hyperparameters.

E.2 IMPLEMENTATION DETAILS

Network architectures. The joint velocity field v_ϕ is parameterized by a residual MLP with a backbone shared across its four velocity components. The shared backbone consists of an input projection followed by four pre-normalized, two-layer residual blocks with ReLU activations and a hidden dimension of 1,024. For datasets with node labels, the structure-conditioned predictor f_ψ is implemented as a direction-aware GNN with four residual message-passing layers and a hidden dimension of 256. These architectural settings are used across all datasets.

Choice of spectral dimension. For each dataset, we select k by minimizing the validation-set average structural MMD ratio over even-valued candidates not exceeding the dataset's maximum node count $N_{\max}$. Here k specifies the maximum number of retained spectral components. For an n-node graph with $n < k$, we retain the largest even number of components not exceeding n and zero-pad the resulting channels for batching. On Circuits, we evaluate $k \in \{4, 8, 16, 32\}$; $k = 32$ achieves the lowest average structural MMD ratio. Results for the other datasets are reported in Table 9. Across the evaluated settings on Circuits, Visual Genome, and SBM, increasing k lowers the average structural MMD ratio, suggesting that retaining more spectral information improves structural distribution matching.

Table 9: Dataset-specific spectral-rank selection using the validation average structural MMD ratio (↓). Selected settings are bolded.

Dataset	Rank k	Avg. MMD ratio ↓
Circuit	4	3.748 ± 0.098
	8	2.609 ± 0.152
	16	2.453 ± 0.210
	32	$\mathbf{1.766 \pm 0.130}$
Visual Genome	4	38.979 ± 0.286
	8	31.018 ± 1.288
	16	10.723 ± 1.230
	32	$\mathbf{3.206 \pm 0.378}$
SBM	16	15.771 ± 0.604
	32	13.680 ± 0.275
	64	5.627 ± 0.848
	96	$\mathbf{2.694 \pm 0.378}$

Optimization and generation. We train the structural model v_ϕ and, where applicable, the label predictor f_ψ separately using AdamW with a weight decay of 10^{-4}. Following the training protocol of Carballo-Castro et al. (2026), we train v_ϕ for 10,000 epochs on each dataset and retain the checkpoint with the lowest average MMD ratio on the validation set. For f_ψ, we use a fixed 2,000 training epochs, which was sufficient for the training objective to stabilize, and use the final checkpoint for evaluation. During its training, the conditioning adjacency is perturbed through edge-count-preserving random rewiring to reduce sensitivity to discrepancies between ground-truth structures used for training and generated structures encountered at inference. For endpoint construction, we use a dataset-specific source template: a directed path (Chain), a directed cycle (Cycle), or an augmented directed cycle (Aug. Cycle) with additional forward and selected reverse edges. In every case, the four source channels are obtained by decomposing the same perturbed source matrix and therefore jointly correspond to a single matrix at $t = 0$. The ablation in Table 12 indicates that this shared-matrix

coupling is the primary contributor, while the particular template topology has a smaller additional effect. Dataset-specific hyperparameters choosen are summarized in Table 10.

Table 10: Dataset-specific hyperparameters for PolarFlow. Here, k denotes the spectral truncation dimension, ε the diagonal regularization strength applied to the target adjacency, σ the scale of the Gaussian noise added to the source template, and N_{Euler} the number of forward Euler steps used to integrate the generative ODE during sampling. Dashes indicate datasets without node labels.

	Endpoint construction				Structural model (v_ϕ)		Sampling	Label model (f_ψ)	
Dataset	k	ε	σ	source template	learning rate	batch size	N_{Euler}	learning rate	batch size
SBM	96	0	0.1	Aug. cycle	3e-4	16	200	–	–
Twitter	64	0.05	0.3	Aug. cycle	3e-4	32	200	–	–
Circuits	32	0.1	0.1	Chain	3e-4	32	200	1e-3	64
Visual Genome	32	0.1	0.1	Cycle	3e-4	32	200	1e-3	64
TPU Tiles	32	0.3	0.1	Chain	3e-4	32	200	1e-3	64

E.3 Ablation studies

We conduct controlled ablations of two design choices affecting PolarFlow's stability and coherence: (i) applying a small diagonal perturbation to the data endpoint before polar–spectral decomposition to mitigate rank and spectral degeneracies, and (ii) jointly constructing the four source channels from a shared noisy matrix to preserve coupling among the polar factors and their spectral coordinates.

Effect of diagonal endpoint regularization. Motivated by the stability analysis Theorem 1, we instantiate the endpoint perturbation as the diagonal shift $\mathbf{A}_\varepsilon = \mathbf{A} + \varepsilon \mathbf{I}_N$ and evaluate $\varepsilon \in \{0, 0.1, 0.2, 0.3\}$, where $\varepsilon = 0$ corresponds to the unregularized endpoint. The shift is applied before the polar–spectral decomposition and does not introduce self-loops, since diagonal entries are excluded during discretization. Rank deficiency and spectral degeneracies, including repeated eigenvalues and weakly separated spectral modes, can make the polar–spectral endpoint representation non-unique or sensitive to small input perturbations. The diagonal shift may alleviate these issues by increasing the rank margin $\sigma_{\min}(\mathbf{A}_\varepsilon)$ or widening the relevant spectral gaps of $\mathbf{P}$ and $\mathbf{U}$, yielding better-conditioned endpoints and, consequently, more stable conditional paths and velocity targets. As shown in Table 11, every tested nonzero shift achieves a lower mean structural average ratio than the unregularized setting on TPU Tiles and Circuit, while $\varepsilon = 0.1$ yields the best result on Twitter. Overall, these results support the practical value of stability-motivated endpoint regularization, indicating that an appropriately chosen regularization strength can improve generation quality.

Table 11: Effect of the diagonal endpoint shift on structural generation quality via average ratio (↓; mean ± standard deviation). $\varepsilon = 0$ denotes the unregularized setting, and the best results are bolded.

Dataset	$\varepsilon = 0$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$
TPU Tiles	16.44 ± 0.36	$\mathbf{11.67 \pm 0.87}$	14.68 ± 0.81	11.81 ± 0.38
Circuit	3.73 ± 0.11	1.89 ± 0.22	1.88 ± 0.30	$\mathbf{1.81 \pm 0.20}$
Twitter	1.61 ± 0.02	$\mathbf{1.33 \pm 0.04}$	2.26 ± 0.19	1.64 ± 0.10

Effect of coupled source construction. To isolate the contribution of the coupled source distribution, we replace the default source state with an independent prior. PolarFlow constructs all four source channels jointly by applying the polar–spectral decomposition to a single source matrix, thereby preserving their cross-channel coupling. In the independent variant, no matrix is decomposed at $t = 0$; instead, the four channels are sampled separately as

$$\mathbf{Q}_{u,0}, \mathbf{Q}_{p,0} \overset{\text{ind.}}{\sim} \mathrm{Unif}\left(\mathrm{St}(N,k)\right), \quad \boldsymbol{\theta}_0^{(k)} \sim \mathrm{Unif}\left((-\pi,\pi]^k\right), \quad \boldsymbol{\lambda}_0^{(k)} \sim \rho_\lambda \text{ with } \mathrm{supp}(\rho_\lambda) \subseteq \mathbb{R}_+^k,$$

where $\mathbf{Q}_{u,0}$, $\mathbf{Q}_{p,0}$, $\boldsymbol{\theta}_0^{(k)}$, and $\boldsymbol{\lambda}_0^{(k)}$ are mutually independent. Here, ρ_λ denotes the positive prior induced by applying the elementwise transformation $x \mapsto \log(1 + \exp(x))$ to a standard Gaussian vector $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_k)$. This construction respects the geometry of each channel while removing the coupling induced by their derivation from a common source matrix.

As shown in Table 12, the independent prior yields higher average ratios on both Circuit and Visual Genome, indicating degraded structural generation quality. This result highlights the benefit of jointly constructing the four source channels through the polar–spectral decomposition of a common source matrix, thereby preserving their cross-channel consistency. To examine whether the improvement instead arises from the structural prior encoded by the template, we compare the structured-template coupled source with a Gaussian-only coupled variant. For the latter, we set the deterministic template to $\mathbf{A}_0 = \mathbf{0}_{N\times N}$, such that the source matrix reduces to $\mathbf{M}_0 = \sigma\mathbf{F}$. This removes the template-specific structural information while preserving the joint construction of all four channels from the same source matrix. The Gaussian-only variant yields results close to those of the structured-template source and remains substantially better than the independent prior. Taken together, these results suggest that the improvement is driven primarily by the joint construction of the four channels from a shared source matrix, rather than by the particular structure encoded in the template.

Table 12: Effect of source coupling and template choice on structural generation quality, measured by average ratio (↓; mean ± standard deviation). Best results are shown in bold.

Dataset	Independent	Coupled	
		Gaussian only ($\mathbf{A}_0 = \mathbf{0}$)	Structured template
Circuit	2.59 ± 0.21	1.88 ± 0.36	$\mathbf{1.69 \pm 0.22}$ (Chain)
Visual Genome	6.23 ± 0.67	3.87 ± 0.59	$\mathbf{3.32 \pm 0.53}$ (Cycle)

Note: The structured variant uses a directed chain for Circuit to reflect its ordered signal-flow dependencies and a directed cycle for Visual Genome, whose relations are not assumed to follow an acyclic ordering.

E.4 Node-label generation

In directed graphs, categorical node labels often encode structural or functional roles associated with patterns of incoming and outgoing connectivity. To capture this dependence, PolarFlow generates node labels in a second stage conditioned on the graph structure. Given an adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$ and node labels $\mathbf{y} \in \mathcal{C}^N$, this stage models the conditional distribution $p_\psi(\mathbf{y} \mid \mathbf{A})$. During training, $\mathbf{A}$ is a data adjacency matrix; at inference, it is replaced by the generated adjacency matrix $\widehat{\mathbf{A}}$, which remains fixed throughout label generation.

We introduce a label time $\tau \in [0,1]$, independent of the structural-flow time t, and augment the label space $\mathcal{C}$ with a mask state M, giving $\overline{\mathcal{C}} = \mathcal{C} \cup \{\mathsf{M}\}$. For each data label vector $\mathbf{y}$, we define a conditional probability path $q_\tau(\mathbf{y}_\tau \mid \mathbf{y})$ from the all-mask state $\mathbf{y}_0 = (\mathsf{M}, \ldots, \mathsf{M})$ to the data label vector $\mathbf{y}_1 = \mathbf{y}$:

$$q_\tau(\mathbf{y}_\tau \mid \mathbf{y}) = \prod_{i=1}^{N} \left[(1-\tau)\mathbf{1}\{y_{\tau,i} = \mathsf{M}\} + \tau\mathbf{1}\{y_{\tau,i} = y_i\}\right].$$

Thus, at time τ, each node remains masked with probability $1-\tau$ and takes it data label with probability τ. These marginals are coupled monotonically across time: each node switches from M to its endpoint label once and remains fixed thereafter.

Let $\mathcal{I}_\tau = \{i \in \mathcal{V} : y_{\tau,i} = \mathsf{M}\}$ denote the set of masked nodes. Given the partially revealed label vector $\mathbf{y}_\tau$, the fixed adjacency matrix $\mathbf{A}$, a direction-aware graph neural network f_ψ uses separate incoming and outgoing aggregations to predict a distribution $\pi^i_\psi(\cdot \mid \mathbf{y}_\tau, \mathbf{A}, \tau)$ over $\mathcal{C}$ for each masked node $i \in \mathcal{I}_\tau$. The model is trained by minimizing cross-entropy over masked nodes:

$$\mathcal{L}_{\text{label}}(\psi) = -\mathbb{E}\left[\frac{1}{|\mathcal{I}_\tau|}\sum_{i\in\mathcal{I}_\tau} \log \pi^i_\psi\left(y_i \mid \mathbf{y}_\tau, \mathbf{A}, \tau\right)\right]. \tag{37}$$

where the expectation is take over $(\mathbf{A}, \mathbf{y}) \sim p_{\text{data}}$, $\tau \sim \text{Unif}(0,1)$ and $\mathbf{y}_\tau \sim q_\tau(\cdot \mid \mathbf{y})$.

At inference, label generation starts from the all-mask state and remains conditioned on the fixed generated adjacency matrix $\widehat{\mathbf{A}}$. At each iteration, f_ψ predicts a distribution over the final label of every masked node. For each masked node, one label is sampled from its predicted distribution. A stochastic reveal score is then assigned to the resulting node-label pair, with higher-probability samples tending to receive higher scores. The procedure reveals the labels of the highest-scoring

nodes until the next target reveal level is reached. The remaining nodes stay masked, while previously revealed labels remain fixed. The updated partially labeled state is passed back into f_ψ to refresh the predictions for the remaining masked nodes. This process continues until all nodes have been assigned labels, yielding $\hat{\mathbf{y}}$. The complete decoding procedure is provided in Algorithm 1.

Algorithm 1 details the iterative procedure for sampling node labels conditioned on a generated adjacency matrix. Decoding begins from the all-mask state and proceeds for K steps along the nominal cosine label-time grid $\tau_k = 1 - \cos(\pi k/(2K))$. At iteration k, f_ψ predicts a categorical label distribution π_i for each masked node i, conditioned on the current partially labeled state and the fixed adjacency matrix. Following the Gumbel-Top-k construction (Kool et al., 2019), a label $c_i \sim \pi_i$ is sampled for each masked node and assigned the stochastic reveal score

$$s_i = \log \pi_i(c_i) + \eta_k g_i, \qquad g_i \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0,1), \qquad \eta_k = \eta_0 \left(1 - \frac{k+1}{K}\right).$$

Here, η_0 denotes the base scale of the Gumbel perturbation. The next label time τ_{k+1} determines the target number of masked nodes after iteration k. Subject to revealing at least one node whenever masks remain, the procedure commits the sampled labels of the m highest-scoring masked nodes. The other masked nodes remain masked, while labels committed in previous iterations remain unchanged. At the next iteration, f_ψ is applied again to the updated partially labeled state to predict labels for the remaining masked nodes. This process terminates when no masked nodes remain, yielding $\widehat{\mathbf{y}}$.

Algorithm 1 Structure-conditioned node-label decoding

Input: Generated adjacency matrix $\widehat{\mathbf{A}}$; trained label predictor f_ψ; number of decoding steps K; base Gumbel scale η_{base}
Output: Generated label vector $\widehat{\mathbf{y}}$

$\widehat{\mathbf{y}} \leftarrow (\mathsf{M}, \ldots, \mathsf{M})$
Set $\tau_k \leftarrow 1 - \cos\left(\frac{\pi k}{2K}\right)$ for $k = 0, \ldots, K$
for $k = 0, \ldots, K-1$ **do**
  $\mathcal{I} \leftarrow \{i \in \mathcal{V} : \widehat{y}_i = \mathsf{M}\}$
  if $\mathcal{I} = \varnothing$ **then**
    return $\widehat{\mathbf{y}}$
  end if
  $\pi_i \leftarrow \pi_\psi^i(\cdot \mid \widehat{\mathbf{y}}, \widehat{\mathbf{A}}, \tau_k)$ for all $i \in \mathcal{I}$
  $\eta_k \leftarrow \eta_{\text{base}} \left(1 - \frac{k+1}{K}\right)$
  for $i \in \mathcal{I}$ **do**
    Sample a label $c_i \sim \pi_i$ and Gumbel noise $g_i \sim \text{Gumbel}(0,1)$
    $s_i \leftarrow \log \pi_i(c_i) + \eta_k g_i$
  end for
  $m \leftarrow \min\{|\mathcal{I}|, \max\{1, |\mathcal{I}| - \lfloor N(1 - \tau_{k+1}) \rfloor\}\}$ ▷ Number of labels revealed
  $\mathcal{S} \leftarrow$ indices of the m largest scores $\{s_i : i \in \mathcal{I}\}$
  for $i \in \mathcal{S}$ **do**
    $\widehat{y}_i \leftarrow c_i$
  end for
end for
return $\widehat{\mathbf{y}}$

E.5 Resources and runtime

All runtime measurements were obtained on a single NVIDIA A100-SXM4 GPU with 80,GB of memory. Table 13 summarizes the computational requirements of training and sampling across datasets and methods, including the number of training and generated (sampled) graphs, training time per epoch, average sampling time per graph, and training VRAM. For completeness, we additionally include the empirical MLE baseline and, on the DAG dataset TPU Tiles, the DAG-specific D-VAE and LayerDAG baselines.

Table 13: Complete training and sampling resource comparison across datasets and methods. Training time per epoch, and sampling time is averaged over individually generated graphs. VRAM is measured during training.

Model	# Train graphs	Training time (s/epoch)	# Sampled	Sampling time (s/graph)	VRAM (GB)
SBM					
MLE	128	7.20	40	0.01	0.15
DiGress	128	2.81	40	4.33	44.32
DeFoG	128	2.45	40	6.04	47.30
Directo-DD RRWP	128	4.40	40	11.27	74.73
Directo-DD MagLap	128	5.76	40	21.07	74.66
Directo RRWP	128	4.42	40	11.20	66.66
Directo MagLap	128	5.88	40	21.22	74.94
PolarFlow	128	2.66	40	0.13	2.32
Twitter					
MLE	3447	14.15	422	2e-4	1.70
DiGress	3447	21.56	422	1.65	8.89
DeFoG	3447	16.85	422	1.80	8.96
Directo-DD RRWP	3447	29.67	422	1.69	18.67
Directo-DD MagLap	3447	45.19	422	3.77	18.63
Directo RRWP	3447	29.67	422	3.56	18.66
Directo MagLap	3447	44.63	422	7.78	18.63
PolarFlow	3447	3.67	422	0.19	1.29
Circuit					
MLE	251	34.8	26	1e-4	0.74
DiGress	251	1.47	26	1.38	7.89
DeFoG	251	1.13	26	1.91	8.15
Directo-DD RRWP	251	1.45	26	1.00	17.79
Directo-DD MagLap	251	2.09	26	2.17	17.79
Directo RRWP	251	1.42	26	1.88	18.27
Directo MagLap	251	2.09	26	6.66	18.27
PolarFlow	251	1.10	26	0.16	1.03
Visual Genome					
MLE	203	72	63	0.06	0.15
DiGress	203	0.70	63	0.74	3.59
DeFoG	203	0.54	63	0.58	4.04
Directo-DD RRWP	203	0.77	63	0.53	7.99
Directo-DD MagLap	203	1.21	63	1.44	8.01
Directo RRWP	203	0.78	63	0.96	8.69
Directo MagLap	203	1.16	63	2.73	8.72
PolarFlow	203	0.81	63	0.04	1.02
TPU Tiles					
MLE	2539	7.43	326	1e-4	0.05
D-VAE	2539	OOM	OOM	OOM	OOM
LayerDAG	2539	0.34	326	6e-4	1.11
DiGress	2539	16.53	326	1.51	9.42
DeFoG	2539	13.27	326	1.79	9.13
Directo-DD RRWP	2539	22.86	326	1.67	19.67
Directo-DD MagLap	2539	32.39	326	3.41	20.07
Directo RRWP	2539	22.79	326	3.23	19.55
Directo MagLap	2539	32.52	326	6.91	19.93
PolarFlow	2539	3.33	326	0.14	1.13

Note: MLE requires no iterative optimization; its training-time entry therefore reports the total runtime required to fit the empirical distributions. The polar–spectral decompositions of the training graphs are precomputed once on CPU and are excluded from the reported per-epoch training time.

F Additional numerical results

This section provides the complete descriptor-level evaluation results. Tables 14 and 15 report MMD values for the in- and out-degree distributions, directed clustering coefficients, and spectral and wavelet features on the synthetic and real-world datasets, respectively. Table 16 further reports node-label generation performance using node-type MMD, multiplicity-aware triplet precision and recall, Triplet MMD, and RBF MMD.

Structural generation The in- and out-degree descriptors characterize directional connectivity patterns, directed clustering captures local triadic organization, and the spectral and wavelet features summarize global and multiscale graph structure. As shown in Tables 14 and 15, PolarFlow achieves strong descriptor-level alignment, including the lowest in- and out-degree MMDs on Visual Genome and TPU Tiles and the lowest wavelet MMD on Circuits. Its clustering and spectral results on Circuits are also comparable to those of the learned baselines. Across the complete set of descriptors,

PolarFlow obtains the lowest average ratio on all three real-world datasets, indicating strong overall alignment across complementary views of directed graph structure.

Table 14: Directed graph generation performance on structural graphs. Results are presented as mean ± standard deviation across five sampling runs.

Model	In-degree↓	Out-degree↓	Clustering↓	Spectre↓	Wavelet↓	Ratio↓	Valid↑	Unique↑	Novel↑	V.U.N.↑
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	0.0038 ± 0.0018	0.0037 ± 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	0.0022 ± 0.0016	0.0026 ± 0.0019	0.0813 ± 0.0058	0.0048 ± 0.0002	0.0110 ± 0.0019	4.3 ± 0.8	37 ± 6.6	100 ± 0.0	100 ± 0.0	37.0 ± 6.6
Directo-DD RRWP	0.0036 ± 0.0018	0.0036 ± 0.0019	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	0.0038 ± 0.0018	0.0037 ± 0.0019	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
Directo RRWP	0.0028 ± 0.0013	0.0031 ± 0.0014	0.0594 ± 0.0027	0.0035 ± 0.0004	0.0027 ± 0.0009	1.8 ± 0.5	99.5 ± 1.0	100 ± 0.0	100 ± 0.0	99.5 ± 1.0
Directo MagLap	0.0038 ± 0.0010	0.0039 ± 0.0012	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
Ours	0.0035 ± 0.0013	0.0037 ± 0.0014	0.0574 ± 0.0010	0.0042 ± 0.0003	0.0048 ± 0.0012	2.0 ± 0.3	92.5 ± 3.8	100 ± 0.0	100 ± 0.0	92.5 ± 3.8
Twitter										
Training set	0.0034	0.0017	0.0027	0.0103	0.0086	1.0	—	100	0.0	0.0
MLE	0.0091 ± 0.0001	0.0088 ± 0.0001	0.0919 ± 0.0022	0.0511 ± 0.0003	0.1381 ± 0.0008	12.59 ± 0.18	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
DiGress	0.0023 ± 0.0001	0.0035 ± 0.0002	0.0049 ± 0.0000	0.0085 ± 0.0002	0.0144 ± 0.0005	1.34 ± 0.02	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
DeFoG	0.0037 ± 0.0003	0.0024 ± 0.0002	0.0049 ± 0.0000	0.0174 ± 0.0014	0.0088 ± 0.0005	1.48 ± 0.04	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Directo-DD RRWP	0.0041 ± 0.0001	0.0016 ± 0.0001	0.0048 ± 0.0000	0.0089 ± 0.0004	0.0078 ± 0.0003	1.14 ± 0.21	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Directo-DD MagLap	0.0052 ± 0.0005	0.0017 ± 0.0002	0.0049 ± 0.0000	0.0153 ± 0.0026	0.0063 ± 0.0007	1.31 ± 0.11	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Directo RRWP	0.0045 ± 0.0003	0.0017 ± 0.0002	0.0048 ± 0.0000	0.0112 ± 0.0007	0.0048 ± 0.0004	1.15 ± 0.04	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Directo MagLap	0.0052 ± 0.0004	0.0019 ± 0.0003	0.0048 ± 0.0000	0.0153 ± 0.0021	0.0078 ± 0.0008	1.37 ± 0.11	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Ours	0.0030 ± 0.0002	0.0026 ± 0.0003	0.0050 ± 0.0000	0.0079 ± 0.0005	0.0134 ± 0.0011	1.33 ± 0.04	—	100 ± 0.0	100 ± 0.0	100 ± 0.0

Table 15: Directed graph generation performance on real graphs. Results are presented as mean ± standard deviation across five sampling runs. OOT indicates that the model could not be run within a reasonable time frame.

Model	In-degree↓	Out-degree↓	Clustering↓	Spectre↓	Wavelet↓	Ratio↓	Valid↑	Unique↑	Novel↑	V.U.N.↑
Circuit										
Training set	0.0005	0.0105	0.0393	0.0123	0.0089	1.0	—	100	0.0	0.0
MLE	0.0121 ± 0.0029	0.0460 ± 0.0070	0.1701 ± 0.0335	0.1489 ± 0.0088	0.0815 ± 0.0095	6.49 ± 0.65	—	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
DiGress	0.0266 ± 0.0031	0.0117 ± 0.0020	0.0758 ± 0.0128	0.0157 ± 0.0038	0.0143 ± 0.0023	2.10 ± 0.13	—	100.0 ± 0.0	98.46 ± 1.88	98.46 ± 1.88
DeFoG	0.0256 ± 0.0024	0.0196 ± 0.0023	0.0734 ± 0.0143	0.0126 ± 0.0019	0.0115 ± 0.0025	1.90 ± 0.24	—	100.0 ± 0.0	99.23 ± 1.54	99.23 ± 1.54
Directo-DD RRWP	0.0134 ± 0.0026	0.0256 ± 0.0037	0.0678 ± 0.0217	0.0188 ± 0.0104	0.0124 ± 0.0067	1.95 ± 0.44	—	99.23 ± 1.54	96.92 ± 2.88	96.17 ± 3.21
Directo-DD MagLap	0.1241 ± 0.0023	0.0272 ± 0.0040	0.0828 ± 0.0056	0.0201 ± 0.0095	0.0134 ± 0.0062	2.06 ± 0.38	—	100.0 ± 0.0	100.0 ± 0.0	100 ± 0.0
Directo RRWP	0.0108 ± 0.0030	0.0257 ± 0.0030	0.0852 ± 0.0183	0.0179 ± 0.0037	0.0114 ± 0.0022	1.90 ± 0.21	—	100 ± 0.0	99.23 ± 1.53	99.23 ± 1.53
Directo MagLap	0.0115 ± 0.0029	0.0250 ± 0.0023	0.0693 ± 0.0099	0.0149 ± 0.0030	0.0106 ± 0.0030	1.77 ± 0.25	—	99.24 ± 1.53	96.91 ± 2.87	96.17 ± 3.21
Ours	0.0110 ± 0.0008	0.0202 ± 0.0034	0.0657 ± 0.0198	0.0168 ± 0.0065	0.0092 ± 0.0019	1.61 ± 0.22	—	100 ± 0.0	100 ± 0.0	100 ± 0.0
Visual Genome										
Training set	0.003	0.0018	0	0.0072	0.0036	1.0	100	100	0.0	0.0
MLE	0.0474 ± 0.0023	0.0607 ± 0.0036	0.5342 ± 0.0830	0.0535 ± 0.0015	0.0399 ± 0.0017	17.00 ± 0.60	90.81 ± 0.92	100.0 ± 0.0	100.0 ± 0.0	90.81 ± 0.92
DiGress	0.0389 ± 0.0032	0.0654 ± 0.0044	0.0008 ± 0.0008	0.0416 ± 0.0017	0.0225 ± 0.0007	17.00 ± 0.60	96.59 ± 0.55	100.0 ± 0.0	100.0 ± 0.0	96.59 ± 0.55
DeFoG	0.0396 ± 0.0028	0.0447 ± 0.0044	0.0002 ± 0.0002	0.0122 ± 0.0020	0.0089 ± 0.0011	10.60 ± 0.80	97.81 ± 0.62	100.0 ± 0.0	100.0 ± 0.0	97.81 ± 0.62
Directo-DD RRWP	0.0413 ± 0.0037	0.0424 ± 0.0043	0.0000 ± 0.0000	0.0132 ± 0.0053	0.0074 ± 0.0083	15.30 ± 0.80	99.75 ± 0.18	100.0 ± 0.0	100.0 ± 0.0	99.75 ± 0.18
Directo-DD MagLap	0.0359 ± 0.0036	0.0245 ± 0.0022	0.0000 ± 0.0000	0.0163 ± 0.0026	0.0086 ± 0.0011	7.60 ± 0.70	99.84 ± 0.06	100.0 ± 0.0	100.0 ± 0.0	99.84 ± 0.06
Directo RRWP	0.0425 ± 0.0029	0.0494 ± 0.0021	0.0000 ± 0.0000	0.0226 ± 0.0075	0.0228 ± 0.0058	12.80 ± 0.60	99.94 ± 0.05	100.0 ± 0.0	100.0 ± 0.0	99.94 ± 0.05
Directo MagLap	0.0302 ± 0.0040	0.0180 ± 0.0024	0.0000 ± 0.0000	0.0142 ± 0.0021	0.0099 ± 0.0013	6.02 ± 0.51	99.08 ± 0.71	99.37 ± 0.78	99.68 ± 0.63	99.08 ± 0.71
Ours	0.0089 ± 0.0021	0.0055 ± 0.0018	0.1231 ± 0.0446	0.0169 ± 0.0028	0.0105 ± 0.0009	3.32 ± 0.53	91.65 ± 1.04	100.0 ± 0.0	100.0 ± 0.0	91.51 ± 0.85
TPU Tiles										
Training set	0.0008	0.0009	0.0008	0.0009	0.0004	1.0	100	100	0.0	0.0
MLE	0.1067 ± 0.0016	0.0377 ± 0.0015	0.0630 ± 0.0031	0.0681 ± 0.0017	0.0752 ± 0.0013	95.54 ± 0.93	52.27 ± 2.32	100.0 ± 0.0	100.0 ± 0.0	52.27 ± 2.32
D-VAE	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT	OOT
LayerDAG	0.1754 ± 0.0316	0.1377 ± 0.0435	0.0162 ± 0.0049	0.0349 ± 0.0064	0.1103 ± 0.0206	135.14 ± 10.23	100.0 ± 0.0	99.93 ± 1.22	100.0 ± 0.0	99.93 ± 1.22
DiGress	0.0953 ± 0.0012	0.0056 ± 0.0003	0.0016 ± 0.0002	0.0065 ± 0.0010	0.0035 ± 0.0006	27.94 ± 0.67	87.06 ± 1.50	99.88 ± 0.25	99.69 ± 0.27	86.63 ± 1.06
DeFoG	0.0942 ± 0.0016	0.0054 ± 0.0003	0.0015 ± 0.0004	0.0045 ± 0.0004	0.0027 ± 0.0002	26.93 ± 0.53	79.69 ± 0.76	99.82 ± 0.25	99.51 ± 0.31	79.20 ± 0.85
Directo-DD RRWP	0.0963 ± 0.0013	0.0057 ± 0.0002	0.0029 ± 0.0011	0.0028 ± 0.0005	0.0015 ± 0.0002	27.02 ± 0.47	88.09 ± 2.28	93.68 ± 1.79	90.18 ± 2.22	88.09 ± 2.28
Directo-DD MagLap	0.0647 ± 0.0026	0.0035 ± 0.0003	0.0012 ± 0.0006	0.0029 ± 0.0004	0.0018 ± 0.0002	18.46 ± 0.87	83.74 ± 1.63	99.01 ± 0.59	97.91 ± 0.68	81.74 ± 1.69
Directo RRWP	0.0952 ± 0.0013	0.0056 ± 0.0002	0.0006 ± 0.0002	0.0022 ± 0.0001	0.0012 ± 0.0002	25.96 ± 0.42	86.44 ± 1.08	95.83 ± 0.53	92.45 ± 1.56	78.34 ± 1.17
Directo MagLap	0.0651 ± 0.0019	0.0035 ± 0.0002	0.0077 ± 0.0033	0.0043 ± 0.0008	0.0019 ± 0.0003	20.96 ± 1.35	85.95 ± 1.46	98.71 ± 0.76	96.81 ± 0.49	82.69 ± 0.98
Ours	0.0004 ± 0.0002	0.0028 ± 0.0004	0.0143 ± 0.0017	0.0138 ± 0.0012	0.0094 ± 0.0008	11.54 ± 0.68	84.97 ± 1.23	99.81 ± 0.15	97.18 ± 0.36	82.45 ± 1.22

Node-label generation Table 16 reports node-label and joint node–edge alignment on the three labeled real-world datasets. Overall, PolarFlow delivers broadly competitive label-generation performance across the complementary metrics. On Circuits, it obtains a competitive node-type MMD and the highest multiplicity-aware precision among the evaluated generators, with recall remaining within the range of the learned baselines. On Visual Genome, it again achieves the highest precision and the lowest RBF MMD while maintaining competitive node-type alignment. On TPU Tiles, its results remain on the same numerical scale as the general-purpose baselines, with Triplet and RBF MMD substantially improving over MLE and LayerDAG. Collectively, these results demonstrate consistent alignment in node-label marginals, semantic-tuple support, and global node–structure representations across different graph domains.

Because node labels are generated conditionally on the sampled adjacency matrix, stronger structural fidelity may provide a more data-like context for label prediction. This is consistent with the favorable RBF MMD on Visual Genome. Since the RBF embeddings jointly encode topology and labels, this

result is best interpreted as evidence of end-to-end alignment in the two-stage generation process rather than performance attributable to the label generator alone.

Table 16: Directed graph generation performance on node-label distributional alignment metrics for the three real-world datasets. Results are presented as mean ± standard deviation across five sampling runs.

Model	**Node type (MMD)↓**	**Precision↑**	**Recall↑**	**Triplet MMD↓**	**RBF↓**
			Circuit		
Training set	0.0057	0.8816	0.8968	0.0117	0.0918
MLE	0.0535 ± 0.0015	0.7889 ± 0.0284	0.9062 ± 0.0278	0.0347 ± 0.0023	0.1599 ± 0.0334
Digress	0.0951 ± 0.0081	0.8793 ± 0.0523	0.8848 ± 0.0605	0.0177 ± 0.0120	0.0973 ± 0.0107
DeFoG	0.0885 ± 0.0080	0.8682 ± 0.0375	0.8094 ± 0.0471	0.0213 ± 0.0074	0.0983 ± 0.0045
Directo-DD RRWP	0.0088 ± 0.0024	0.8133 ± 0.0454	0.8976 ± 0.0148	0.0130 ± 0.0053	0.0896 ± 0.0184
Directo-DD MagLap	0.0039 ± 0.0032	0.8751 ± 0.0363	0.8675 ± 0.0453	0.0147 ± 0.0061	0.1032 ± 0.0183
Directo RRWP	0.0093 ± 0.0099	0.8421 ± 0.0440	0.8529 ± 0.0634	0.0142 ± 0.0058	0.0896 ± 0.0097
Directo MagLap	0.0079 ± 0.0107	0.8593 ± 0.0492	0.8806 ± 0.0359	0.0163 ± 0.0048	0.0900 ± 0.0103
Ours	0.0076 ± 0.0037	0.8881 ± 0.0464	0.8697 ± 0.0454	0.0231 ± 0.0089	0.1078 ± 0.0137
			Visual Genome		
Training set	0.0116	0.6381	0.6200	0.0123	0.0256
MLE	0.0519 ± 0.0021	0.6042 ± 0.0307	0.1310 ± 0.0064	0.3298 ± 0.0036	0.1206 ± 0.0087
Digress	0.0206 ± 0.0034	0.7066 ± 0.0392	0.2748 ± 0.0178	0.2098 ± 0.0097	0.0830 ± 0.0120
DeFoG	0.0110 ± 0.0019	0.6132 ± 0.0407	0.5309 ± 0.0038	0.0160 ± 0.0036	0.0738 ± 0.0024
Directo-DD RRWP	0.0231 ± 0.0082	0.5583 ± 0.0368	0.5377 ± 0.0291	0.0272 ± 0.0087	0.0533 ± 0.0112
Directo-DD MagLap	0.0288 ± 0.0069	0.5719 ± 0.0207	0.5188 ± 0.0240	0.0319 ± 0.0054	0.0634 ± 0.0111
Directo RRWP	0.0535 ± 0.0149	0.4775 ± 0.0462	0.4707 ± 0.0462	0.0484 ± 0.0158	0.0646 ± 0.0517
Directo MagLap	0.0299 ± 0.0047	0.5008 ± 0.0287	0.4643 ± 0.0246	0.0348 ± 0.0050	0.0802 ± 0.0074
Ours	0.0164 ± 0.0030	0.7040 ± 0.0192	0.4755 ± 0.0213	0.0701 ± 0.0071	0.0458 ± 0.0070
			TPU Tiles		
Training set	0.00016	0.9374	0.9769	0.00061	0.0056
MLE	0.0010 ± 0.0001	0.9051 ± 0.0157	0.9336 ± 0.0054	0.0914 ± 0.0039	0.1578 ± 0.0130
D-VAE	OOT	OOT	OOT	OOT	OOT
LAYERDAG	0.0073 ± 0.0027	0.1057 ± 0.0085	0.1228 ± 0.0299	0.4744 ± 0.0110	0.4224 ± 0.0767
Digress	0.0014 ± 0.0003	0.8946 ± 0.0095	0.9215 ± 0.0055	0.0047 ± 0.0009	0.0160 ± 0.0001
DeFoG	0.0007 ± 0.0002	0.8976 ± 0.0071	0.9296 ± 0.0110	0.0041 ± 0.0006	0.0167 ± 0.0011
Directo-DD RRWP	0.0007 ± 0.0002	0.9159 ± 0.0052	0.9460 ± 0.0051	0.0022 ± 0.0003	0.0126 ± 0.0007
Directo-DD MagLap	0.0010 ± 0.0003	0.8969 ± 0.0111	0.9349 ± 0.0107	0.0038 ± 0.0009	0.0151 ± 0.0004
Directo RRWP	0.0007 ± 0.0002	0.9125 ± 0.0070	0.9389 ± 0.0102	0.0024 ± 0.0004	0.0132 ± 0.0005
Directo MagLap	0.0014 ± 0.0002	0.8968 ± 0.0082	0.9055 ± 0.0071	0.0056 ± 0.0007	0.0163 ± 0.0010
Ours	0.0018 ± 0.0004	0.8723 ± 0.0077	0.9067 ± 0.0046	0.0093 ± 0.0006	0.0237 ± 0.0021