# EFFICIENT SYNTHETIC NETWORK GENERATION VIA LATENT EMBEDDING RECONSTRUCTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Network data are ubiquitous across the social sciences, biology, and information systems. Generating realistic synthetic network data has broad applications from network simulation to scientific discovery. However, many existing black-box approaches for network generation tend to overfit observed data while overlooking characteristic network structure, and incur substantial computational overhead at scale. These practical challenges call for synthetic network generation methods that are both efficient and capable of capturing structural properties of networks. In this paper, we introduce Synthetic Network Generation via Latent Embedding Reconstruction (SyNGLER), a general and efficient framework for synthetic network generation that builds on latent space network models. Given an observed network, SyNGLER first learns low-dimensional latent node embeddings via a latent space network model and then reconstructs the latent space by building a distribution-free generator over these embeddings. For generation, SyNGLER first samples (or resamples) node embeddings from the generator in the latent space and then produces synthetic networks using the latent space network model. Through the latent space framework, SyNGLER preserves unique characteristics in networks such as sparsity and node degree heterogeneity, while allowing for efficient training with lower computational cost than many existing deep architectures. We provide theoretical guarantees by developing consistency results regarding the distance between the true and synthetic edge distributions. Empirical studies further demonstrate the effectiveness of SyNGLER, where SyNGLER efficiently produces networks that better preserve key network characteristics such as network moments and degree distributions compared with existing approaches.

## 1 INTRODUCTION

Graph network data capture interactions among entities in complex systems. Examples include social networks (Traud et al., 2012), molecular interaction networks (Gómez-Bombarelli et al., 2018), and brain connectivity networks (Bullmore & Sporns, 2009). Generating realistic synthetic network data (Zhu et al., 2022) has broad applications, spanning drug discovery (Li et al., 2018a), material discovery (Merchant et al., 2023), and image recognition (Xie et al., 2019). Designing efficient generative models that produce realistic network data while preserving characteristic structural network properties remains a long-standing and active research challenge.

Recent years have witnessed a growing line of work on data-driven graph network generation using deep learning. For example, Li et al. (2018b) proposed an autoregressive generation scheme, in which a graph neural network (GNN; Scarselli et al. (2008)) sequentially adds nodes and edges based on the current graph. You et al. (2018) later adopted recurrent neural networks (Schmidt, 2019) that summarize nodes and edges and generate, at each step, the next node and its associated edges. Liao et al. (2019) introduced a block-wise autoregressive model with graph attention mechanism (Veličković et al., 2017), reducing serial computation while preserving long-range dependencies. Nevertheless, for large graphs, training and sampling in deep autoregressive models remain computationally heavy due to sequential modeling of a large graph (Salha et al., 2021). Another line of research develops diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) for graphs. Early methods (Niu et al., 2020; Jo et al., 2022) applied continuous diffusion processes directly in adjacency-matrix space, which neglected discreteness in graphs. Vignac et al. (2022) and Haefeli et al. (2022) studied discrete Markov processes over adjacency matrices. However, while operating

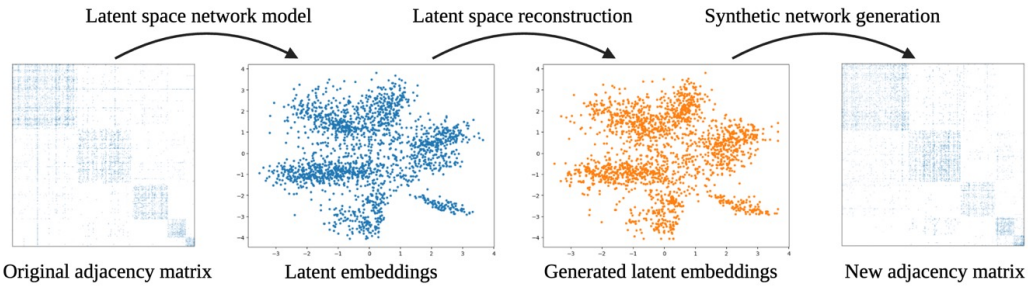



Figure 1: An illustrative SyNGLER pipeline using the YOUTUBE dataset (Yang & Leskovec, 2012) with a two-dimensional latent space. From left to right: observed network in the form of an adjacency matrix; learned latent embeddings; synthetic embeddings from the generator in the latent space; synthetic network.

on a discrete state space, applying diffusion directly in adjacency-matrix space still overlooks the low-rank structure often present in large-scale network data (Luo et al., 2023). Vahdat et al. (2021) and Rombach et al. (2022) combined diffusion modeling with encoder-decoder architectures by applying diffusion in a continuous latent space. The resulting latent-diffusion approach has been used for molecular and protein graph generation (Xu et al., 2023; Fu et al., 2024) and extended to general graph generation (Zhou et al., 2024), covering both conditional and unconditional settings. Nevertheless, these methods typically rely on variational training to connect the graphs and the latent space, which becomes computationally demanding for large-scale networks. Overall, existing methods tend to face computational challenges on large graphs due to deep neural network training on large-dimensional data and/or variational procedures, while characteristic network structure is often neglected, underscoring the need for graph generation models that can capture complex network structure while staying computationally tractable and scalable to large graphs.

In this work, we introduce Synthetic Network Generation via Latent Embedding Reconstruction (SyNGLER), an efficient synthetic network generation framework that leverages latent space network models (Hoff et al., 2002; Ma et al., 2020) to address these challenges. During training, SyNGLER first fits a likelihood-based latent space model to the observed network to learn a set of low-dimensional node embeddings. Using these embeddings, it then trains a distribution-free generator in the latent space. For generation, SyNGLER samples (or resamples) node embeddings from the generator and produces synthetic networks from these node embeddings via the latent space model. An illustrative pipeline is in Figure 1. Via the latent space approach, SyNGLER efficiently learns a set of low-dimensional node embeddings and requires only lightweight generative model training in the latent space, thereby reducing computational cost. Moreover, the geometry of in the latent space enables SyNGLER to preserve key structural properties of the network that reflect latent node-node interactions. We provide a theoretical analysis of SyNGLER and establish its consistency and generalization guarantees under a hierarchical latent space network model. Extensive experiments on synthetic and real-world datasets further demonstrate its strong performance in comparison to existing approaches, with significantly reduced computational cost.

The remainder of the paper is organized as follows. In Section 2, we formally introduce the SyNGLER framework. Section 3 presents the theoretical results. Section 4 reports empirical results on simulated and real-world data. Section 5 concludes the paper with a discussion. Additional numerical results, experimental details, and proofs are provided in the Appendix.

## 2 SYNTHETIC NETWORK GENERATION VIA LATENT EMBEDDING RECONSTRUCTION

### 2.1 SETUP AND SYNGLER

Given an observed network with $n$ nodes, our goal is to train a generative model that can produce networks that preserves key structural properties of the original network. The synthetic node set may comprise the original nodes, newly generated nodes with distributional characteristics similar to the

originals, or any mixture of the two. Specifically, we represent the observed network by an adjacency matrix $A \in \mathbb{R}^{n\times n}$ with $A_{ij} = A_{ji}$ for $i \neq j$ and $A_{ii} = 0$ for all $i \in [n]$, where the observation $A_{ij}$ can be binary observations in $\{0, 1\}$, count-valued observations in $\mathbb{N}$, or general continuous observations in $\mathbb{R}$. Given the observation $A$, our goal is to generate a synthetic network $\widetilde{A} \in \mathbb{R}^{n\times n}$.

We introduce Synthetic Network Generation via Latent Embedding Reconstruction (SyNGLER) to achieve this goal. During training, SyNGLER first learns a set of node embeddings from the observed network using a likelihood model $p(\cdot \mid \cdot)$ compatible with edge types of $A_{ij}$, and then reconstructs the latent space by training a distribution-free generator over the learned embeddings. For data generation, SyNGLER samples a set of node embeddings from the latent generator and produces synthetic networks using these embeddings via $p(\cdot \mid \cdot)$. Algorithm 1 summarizes the procedure.

**Algorithm 1** Synthetic Network Generation via Latent Embedding Reconstruction

1: **Input:** Latent dimension $r$, input network $A \in \{0, 1\}^{n\times n}$.
2: Fit the likelihood model to get $(\hat{Z}, \hat{\alpha}, \hat{\rho}) \in \mathbb{R}^{n\times r} \times \mathbb{R}^n \times \mathbb{R}$;
3: Train a generative model using the fitted data: $\texttt{Sampler} = \texttt{GenModel}(\{(\hat{z}_i, \hat{\alpha}_i)\}_{i=1}^n)$;
4: **for** each $i = 1, \ldots, n$ **do**
5: Sample $(\tilde{z}_i, \tilde{\alpha}_i) \in \mathbb{R}^{r+1}$ from $\texttt{Sampler}$.
6: **end for**
7: **for** each pair of nodes $(i, j)$ with $1 \leq i < j \leq n$ **do**
8: Independently generate the edge observation $\widetilde{A}_{ij} = \widetilde{A}_{ji}$ from the conditional model $p(\cdot \mid \tilde{z}_i^\top \tilde{z}_j + \tilde{\alpha}_i + \tilde{\alpha}_j + \hat{\rho}_n)$
9: **end for**
10: **Output:** Generated network $\widetilde{A}$.

In Algorithm 1, $\hat{Z} = (\hat{z}_1, \ldots, \hat{z}_n)^\top$ denotes the learned latent node embeddings, $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^\top$ the learned node degree parameters, and $\hat{\rho}$ the learned global sparsity parameter. $\texttt{GenModel}$ denotes the generator architecture in latent space. We specify the likelihood $p(\cdot \mid \cdot)$ and formally introduce the model parameters in Section 2.2, and we detail $\texttt{GenModel}$ in Section 2.3.

### 2.2 Latent Space Network Models and Network Embedding

Latent space network models (Hoff et al., 2002; Ma et al., 2020; Li et al., 2025) provide a flexible and efficient network embedding framework. Associate each node $i$ with a latent position $z_i \in \mathbb{R}^r$ and a degree parameter $\alpha_i \in \mathbb{R}$. Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{R}^n$ and define $\Phi \in \mathbb{R}^{n\times(r+1)}$, where each row of $\Phi$ is given by $\Phi_i = (z_i^\top, \alpha_i)^\top \in \mathbb{R}^{r+1}$, $i = 1, \ldots, n$. Let $\rho_n \in \mathbb{R}$ denote the global sparsity parameter that is allowed to diverge with $n$ and controls network sparsity (or, for weighted edges, overall signal strength). Latent space network models specify that given $\Phi$, each edge observation $A_{ij} = A_{ji}$ for $1 \leq i < j \leq n$ is independently generated from:

$$\mathbb{P}(A_{ij} \in \mathcal{A} \mid \Phi, \rho_n) = \int_{\mathcal{A}} p(a \mid {z_i}^\top z_j + \alpha_i + \alpha_j + \rho_n)\, d\mu(a), \tag{1}$$

where $p(\cdot \mid \cdot)$ is the conditional density of $A_{ij}$ given $\Phi$ and $\rho_n$, $\mu$ is some measure on $\mathbb{R}$ (e.g., counting measures for discrete-valued edges or Lebesgue measures for continuous-valued edges), and $\mathcal{A}$ is the corresponding Borel set. Given $\Phi$ and $\rho_n$, we use $\mathbb{P}_{A|\Phi,\rho_n}$ to denote the conditional distribution of $A$ given $\Phi$ and $\rho_n$. We consider the following assumptions on the latent embedding distribution.

**Assumption 2.1.** *We assume that $\{(z_i, \alpha_i)\}_{i=1}^n$ are independently sampled from some distribution $\mathbb{P}_0$ on $\mathbb{R}^{r+1}$ such that:*

*(i) the conditional distribution of $\alpha_i$ given $z_i$ follows that $\mathbb{P}_0(\alpha \mid z) = \mathbb{P}_0(\alpha \mid Uz)$ for any $U \in \mathbb{R}^{r\times r}$ such that $U^\top U = I_r$;*

*(ii) there exists $R > 0$ with $\mathbb{E}_{\mathbb{P}_0}[\phi] = 0_{r+1}$ and $\operatorname{supp}(\mathbb{P}_0) \subset B_R$, where $B_R = \{\phi \in \mathbb{R}^{r+1} : \|\phi\|_2 \leq R\}$;*

*(iii) the expectation of $z$ and $\alpha$ under $\mathbb{P}_0$ is zero, i.e. $\mathbb{E}_{\mathbb{P}_0}[\alpha] = 0$ and $\mathbb{E}_{\mathbb{P}_0}[z] = 0_r$;*

*(iv) the sparsity parameter $\rho_n^*$ satisfies that $\rho_n^* > (-1 + \kappa) \log n$ for some universal constant $\kappa > 0$.*

Assumption (i) ensures that the choice of coordinates for the latent positions does not affect the conditional distribution of the degree parameter. Assumption (ii) requires the latent positions and degree parameters to be bounded to ensure well-conditioning in latent space, Assumption (iii) guarantees the identifiability of the latent parameters, since joint shifts in $\alpha$ and $z$ could leave the edge distribution unchanged, and Assumption (iv) on $\rho_n^*$ guarantees that the observed network $A$ is not too sparse to enable valid estimation (Li et al., 2025).

**Network embedding for general edge types.** Based on the observed edge type of $A_{ij}$, the likelihood model $p(\cdot \mid \cdot)$ can be chosen flexibly. Define $\pi_{ij} = z_i^\top z_j + \alpha_i + \alpha_j + \rho_n$, which determines the distribution of $A_{ij}$. If $A_{ij}$'s are binary, one can choose the Bernoulli model as further introduced in the next paragraph. If $A_{ij}$'s are continuously measured, one can use a Gaussian noise model $p(a_{ij} \mid \pi_{ij}) = (2\pi\sigma^2)^{-1/2} \exp\{-(a_{ij} - \pi_{ij})^2/(2\sigma^2\}$ for some $\sigma^2 > 0$. Once the conditional model is determined, we can fit the latent space model by maximizing the following likelihood function over the parameters $(Z, \alpha, \rho)$, where $Z = (z_1^\top, z_2^\top, \ldots, z_n^\top)^\top$ is the latent position matrix with rows $z_1, z_2, \ldots, z_n$ and $\alpha = (\alpha_1, \ldots, \alpha_n)$ is the vector of degree parameters:

$$(\hat{Z}, \hat{\alpha}, \hat{\rho}) = \underset{\substack{(Z,\alpha)\in\mathbb{R}^{n\times(r+1)},\rho\in\mathbb{R}:\\ Z^\top Z \text{ diagonal}, Z^\top \mathbf{1}_n = \mathbf{0}_r, \alpha^\top \mathbf{1}_n = 0}}{\arg\max} \sum_{1\le i<j\le n} \log p(A_{ij} \mid z_i^\top z_j + \alpha_i + \alpha_j + \rho). \tag{2}$$

The constraint in Eq. (2) ensures identifiability of the fitted latent positions and degree parameters and is designed based on (iii) in Assumption 2.1. Although this problem is nonconvex, it can be solved efficiently via projected gradient descent, with provable convergence guarantees (Ma et al., 2020). We leave the details of the optimization algorithms in Appendix B.1.

**Sparse networks for binary edges.** For binary networks, let $\mu = \delta_0 + \delta_1$ (Dirac measures at 0 and 1) and specify $p(a_{ij} \mid \pi_{ij}) = \exp\{a_{ij}\pi_{ij}\}\,(1 + \exp\{\pi_{ij}\})^{-1}$, i.e., $\mathbb{P}(A_{ij} = 1 \mid \pi_{ij}) = \sigma(\pi_{ij})$, where $\sigma(\cdot) = \exp(\cdot)/\big(1 + \exp(\cdot)\big)$. Recall that $\pi_{ij} = z_i^\top z_j + \alpha_i + \alpha_j + \rho_n$, and with $\rho_n \to -\infty$, the edge probability $\Pr(A_{ij} = 1 \mid \pi_{ij}) = \sigma(\pi_{ij}) \to 0$. This is consistent with the fact that many large-scale networks are sparse, and $\sigma(\pi_{ij}) \asymp \exp(\rho_n)$ controls the sparsity level. We remark that, provided that $\exp(\rho_n)$ decays no faster than $n^{-1+\epsilon}$ for some $\epsilon > 0$ as $n \to \infty$, the solution to Eq. (2) remains theoretically tractable; see Li et al. (2025).

### 2.3 Latent Embedding Reconstruction

In this section, we introduce two implementations of `GenModel` in Algorithm 1. Both implementations reconstruct the latent space from the learned embeddings. The first approach resamples from the empirical distribution on $\hat{\Phi}$; this is suitable when the synthetic network should include some original nodes from the observed network. The second approach trains a distribution-free generative model on the learned embeddings, and we use a score-based generator as an example and for implementation in this work. The choice of `GenModel` is flexible and can be adapted to practitioners' needs.

**Resampling based latent space reconstruction.** Given the learned embeddings $\hat{\Phi}$, we set `GenModel` to be the uniform distribution over the discrete set $\hat{\Phi}$. At each call, this sampler returns one of the row vectors of $\hat{\Phi}$, with replacement. The resampled latent embeddings, denoted $\tilde{\Phi}$, are used later to construct the synthetic network. The idea of resampling latent embeddings in networks has been used for bootstrap inference of network statistics (see Levin & Levina, 2019), but it has not been systematically studied for network generation tasks. Although straightforward to implement, direct resampling from $\hat{\Phi}$ has several limitations. First, resampling $\hat{\Phi}$ exposes the latent characteristics of existing nodes, which may carry sensitive, although unobserved, information about individuals in the original network into downstream network generation. Moreover, sampling with replacement can easily lead to duplicate embeddings in $\tilde{\Phi}$, which is unrealistic because latent positions are typically continuous and distinct. In this scenario, we suggest dropping the duplicate embeddings and preserving any nodes that must remain in the network as needed.

**Score-based generator in the latent space.** To sample novel node embeddings that are close in distribution to the learned embeddings, here we use a score-based generative model (Song et al., 2020), in which a forward noising process gradually adds noise to the input data, and a backward denoising

process recovers the original data from pure noise using information from the forward process. In our setup, a key difference is that we train the score network based on the learned embeddings. Specifically, let $s_\theta(x,t) : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ be a black-box prediction model parameterized by $\theta$. We use xgboost, following ForestDiffusion (Jolicoeur-Martineau et al., 2024), to approximate the score function. Consider a variance-preserving Ornstein-Uhlenbeck (OU) (Maller et al., 2009) forward process. Given the fitted embeddings $\hat{\Phi}$, the forward process $\phi_t$ follows

$$d\phi_t = -\phi_t\, dt + \sqrt{2}\, dB_t,$$

where $\phi_0$ is randomly sampled from the row vectors of $\hat{\Phi}$ and $B_t$ is a standard Wiener process. Given the forward process, the parameter $\theta$ in $s_\theta(x,t)$ is optimized by minimizing the denoising score-matching objective (Song et al., 2020) constructed using the learned embeddings (Wu et al., 2025):

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_{t\sim\mathcal{U}[0,1],\, z\sim\mathcal{N}(0,I_{r+1})}\left[\frac{1}{n}\sum_{i=1}^{n}\left\| s_\theta\big(e^{-t}\hat{\phi}_i + \sqrt{1-e^{-2t}}\, z,\, t\big) + \frac{z}{\sqrt{1-e^{-2t}}}\right\|^2\right].$$

To sample from the generator, we simulate the following process initialized at $\tilde{\phi}^{(0)} \sim \mathcal{N}(0, I_{r+1})$, using the trained network $s_{\hat{\theta}}$:

$$\tilde{\phi}^{(k+1)} = \tilde{\phi}^{(k)} + h\Big(\tilde{\phi}^{(k)} + 2\, s_{\hat{\theta}}(\tilde{\phi}^{(k)}, 1-kh)\Big) + \sqrt{2h}\, \xi_k, \quad \xi_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_{r+1}).$$

When $T$ is large (equivalently, $h$ is small) and $s_\theta$ sufficiently approximates the true score, the distribution of $\tilde{\phi}^{(\lceil 1/h \rceil)}$ is close to the distribution of the learned embeddings. This approach enables us to sample novel node embeddings, thereby including novel nodes in the synthetic network.

## 3 Theoretical Analysis

In this section, we study the distance between the distributions of the synthetic and the original network. Throughout this section, we treat the global sparsity parameter $\rho_n^*$ as a fixed parameter, where a superscript $*$ indicates true parameters, and consider the data generating model where the observed network $A$ is generated from the latent space model in Eq. (1), with global sparsity parameter $\rho_n^*$ and some $\Phi^*$ whose rows are independent realizations of $\mathbb{P}_0$ from Assumption 2.1. Note that the distribution of the synthetic network $\tilde{A}$ is based on the model trained on the observed network $A$. We denote the distribution of $\tilde{A}$ given $A$ as $\mathbb{P}^A_{\tilde{A}}$, which can be viewed as a random measure where the randomness comes from the observed network $A$ generated under model Eq. (1) with embeddings $\Phi^*$ and sparsity parameter $\rho_n^*$. Similarly, we define $\mathbb{P}^A_{\tilde{\Phi}}$ as the random measure for the group of latent embeddings on $\mathbb{R}^{n\times(r+1)}$. We also denote the individual distribution of $\tilde{\phi}$ given $A$ as $\mathbb{P}^A_{\tilde{\phi}}$. Similarly, we denote the marginal distribution of each $\hat{\phi}_i$ as $\mathbb{P}_{\hat{\phi}}$, where the subscript $i$ in $\hat{\phi}_i$ is omitted due to the exchangeability of the nodes. Our first theorem decomposes the Kullback-Leibler (KL) divergence between $\mathbb{P}_A$ and $\mathbb{P}^A_{\tilde{A}}$.

**Theorem 3.1.** *The average KL divergence between the distribution of $A$ and $\tilde{A}$ given $A$ admits the following decomposition:*

$$\frac{1}{n^2} d_{\mathrm{KL}}(\mathbb{P}_A \,\|\, \mathbb{P}^A_{\tilde{A}}) = E_\rho + E_\Phi + E_{gen}. \tag{3}$$

*where the three error terms are defined as follows:*

$$\begin{aligned}
E_\rho &= \mathbb{E}_{\mathbb{P}_{A|\Phi}\times\mathbb{P}_0^{\otimes n}}\left[\log\frac{p(A_{12}\,|\, z_1^\top z_2 + \alpha_1 + \alpha_2 + \rho_n^*)}{p(A_{12}\,|\, z_1^\top z_2 + \alpha_1 + \alpha_2 + \hat{\rho})}\right];\\
E_\Phi &= \min_{\mathcal{T}} \frac{1}{n}\left(\mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}}\left[\log\frac{\mathbb{P}_0^{\mathcal{T}}}{\mathbb{P}_{\hat{\phi}}}(\phi)\right] + \mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}}\left[\log\frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi)\right] - \mathbb{E}_{\mathbb{P}_{\hat{\phi}}}\left[\log\frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi)\right]\right);\\
E_{gen} &= \frac{1}{n} d_{\mathrm{KL}}(\mathbb{P}_{\hat{\phi}} \,\|\, \mathbb{P}^A_{\tilde{\phi}}).
\end{aligned} \tag{4}$$

*Here the minimization in Eq. (4) is over all orthogonal transforms $\mathcal{T} : \Phi = (Z, \alpha) \mapsto (ZU, \alpha)$, where $U$ is a $r$-dimensional rotation matrix, and $\mathbb{P}_0^{\mathcal{T}}$ is the distribution of $\mathcal{T}(\phi)$ when $\phi \sim \mathbb{P}_0$.*

The first term in Theorem 3.1 is referred to as $E_\rho$ since $p(A_{ij} \mid z_i^\top z_j + \alpha_i + \alpha_j + \rho_n^*)$ differs from $p(A_{ij} \mid z_i^\top z_j + \alpha_i + \alpha_j + \hat{\rho})$ only in the global sparsity parameter $\rho_n$. The second term is denoted as $E_\Phi$, since it primarily depends on the distance between $\mathbb{P}_{\hat{\phi}}$ and $\mathbb{P}_0$. The third term $E_{\text{gen}}$ denotes the KL divergence between the conditional distributions of $\hat{\phi}$ and $\tilde{\phi}$ given $A$, and is determined by the generative model in Algorithm 1. Error analysis of such terms has been considered in the literature, for example, Chen et al. (2022; 2023a) for score-based generative models. In this analysis, we focus on characterizing the first two terms in Eq. (3), while noting that the third term concerns the generative error in a low-dimensional space rather than in the original large-scale network space.

In the sequel, we analyze $E_\rho$ and $E_\Phi$ under Assumption 2.1 and the asymptotic regime where $n \to \infty$. We first have the following theorem on $E_\rho$.

**Theorem 3.2.** *Suppose that the log likelihood for each edge $l_a(\pi) = \log p(a \mid \pi)$ follows that $\sup_{a,|\pi|\le 2R^2} |l_a'(\pi)| < M$, and each $l'_{A_{ij}}(z_i^{*\top} z_j^* + \alpha_i^* + \alpha_j^* + \rho_n^*)$ is independently bounded random variables, then it holds that $E_\rho = O_p((w_n \cdot n)^{-1/2} \log n)$, where $w_n = e^{\rho_n^*}$.*

Analyzing the second term is challenging since it involves the marginal distribution of the estimated latent positions, which is practically hard to be compared with the true distribution $\mathbb{P}_0$. Following Wu et al. (2025), we use the technique of discretizing the underlying distribution to understand the approximation error between $\mathbb{P}_{\hat{\phi}}$ and $\mathbb{P}_0^{\mathcal{T}}$ for some transform $\mathcal{T}$. Suppose that $\mathbb{P}_0$ is a continuous distribution of the latent embeddings with density $p_0$ that follows Assumption 2.1. Since the support of $\mathbb{P}_0$ is bounded, we discretize the support of $\mathbb{P}_0$ into the following grid:

$$\mathcal{G}_{\gamma_n} = \{\phi \in \mathbb{R}^{r+1} : \phi_i/\gamma_n \in \mathbb{Z}, |\phi_i| \le R + \gamma_n\},$$

where $\{\gamma_n\}$ is a sequence of discretization scales that goes to zero. Then for any $\phi \in \mathcal{G}_{\gamma_n}$, we define the following mass function:

$$q_{\gamma_n}(\phi) = \int_{\mathbb{R}^{r+1}} \prod_{i \le r+1} \mathbb{1}\{\phi_i' \in [\phi_i - \gamma_n/2, \phi_i + \gamma_n/2\} p_0(\phi') d\phi'.$$

Using the linearity of expectation, we conclude that $\sum_{\phi \in \mathcal{G}_{\gamma_n}} q_{\gamma_n}(\phi) = 1$. Therefore, $q_{\gamma_n}$ is a probability mass function. We claim that, as long as the original density function is sufficiently smooth, this discretized mass function is able to capture the structure of the original density function well. To this end, we define the projection operator associated with the grid as $\text{proj}_{\mathcal{G}_{\gamma_n}}(\phi) = \arg\min_{\phi' \in \mathcal{G}_{\gamma_n}} \|\phi' - \phi\|_2$, and consider $p_{\gamma_n}(\phi) = q_{\gamma_n}(\text{proj}_{\mathcal{G}_{\gamma_n}}(\phi)) {\gamma_n}^{-(r+1)}$. Then we have the following theorem.

**Theorem 3.3.** *Suppose that $p_0 : \mathbb{R}^{r+1} \to \mathbb{R}_{\ge 0}$ is L-Lipschitz, then it holds that*

$$\left|p_0(\phi) - p_{\gamma_n}(\phi)\right| \le L\gamma_n\sqrt{r+1}.$$

This theorem indicates that sampling from the original distribution is almost the same as sampling from $q_{\gamma_n}$. We consider $\Phi^\dagger = (\phi_1^\dagger, \ldots, \phi_n^\dagger)^\top \in \mathbb{R}^{n \times (r+1)}$ where the rows are independent realizations from $q_{\gamma_n}$. We denote the corresponding empirical mass function as $\hat{q}_{\gamma_n}(\phi) = n^{-1}\sum_{i \le n} \mathbb{1}\{\phi_i^\dagger = \phi\}$ for $\phi \in \mathcal{G}_{\gamma_n}$. Then the following result indicates that $\hat{q}_{\gamma_n}$ is close to $q_{\gamma_n}$.

**Lemma 3.1.** *For any small $\delta > 0$, it holds that $\max_{\phi \in \mathcal{G}_{\gamma_n}} |q_{\gamma_n}(\phi) - \hat{q}_{\gamma_n}(\phi)| \le \sqrt{\log(1/\delta) + \log(R/\gamma_n)}/\sqrt{n}$ with probability $1 - \delta$.*

When $\Phi^*$ is replaced by $\Phi^\dagger$, we need to modify the estimators in Eq. (2) accordingly using the projection operator $\text{proj}_{\mathcal{G}_{\gamma_n}}$. Given $\widehat{\Phi}$ and a rotation transform $\mathcal{T}$, we define the corresponding empirical distribution on the grid $\mathcal{G}_{\gamma_n}$ as follows:

$$\check{q}_{\gamma_n}(\phi) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{\phi_i = \text{proj}_{\mathcal{G}_{\gamma_n}}(\mathcal{T}\hat{\phi}_i)\}, \quad \phi \in \mathcal{G}_{\gamma_n}.$$

Next theorem shows that $\check{q}_{\gamma_n}$ is close to $\hat{q}_{\gamma_n}$, which is a direct consequence of the uniform consistency of the estimated latent embeddings.

**Theorem 3.4.** *Suppose that each $\phi$ is sampled from a discrete mass function $q_{\gamma_n}$ on $\mathcal{G}_{\gamma_n}$. Then, for any $\gamma_n = \Omega\big((w_n \cdot n)^{-1/2+\epsilon}\big)$ for some $\epsilon > 0$, there exists a rotation transform $\mathcal{T}$, such that*

$$\mathbb{P}\Big( \max_{\phi \in \mathcal{G}_{\gamma_n}} |\hat{q}_{\gamma_n}(\phi) - \check{q}_{\gamma_n}(\phi)| = 0 \Big) \to 1.$$

*Note that here the randomness comes from the realizations of the observed network $A$.*

This result shows that the marginal distribution of any single point from $\{\mathrm{proj}_{\gamma_n}(\hat{\phi}_i)\}_{i \le n}$ is close to the discretized distribution $q_{\gamma_n}$, up to a rotation. Combining with the uniform convergence result in Li et al. (2025), we conclude that $\mathbb{P}_{\hat{\phi}}$ is close to $\mathbb{P}_0^{\mathcal{T}}$ for some rotation transform $\mathcal{T}$.

## 4 Experiments

In this section, we empirically evaluate the effectiveness and efficiency of the proposed SyNGLER framework using both simulated and real-world network datasets.

**Simulated network dataset.** We consider simulated networks with sizes and latent dimensions $(n, r) \in \{500, 1000, 1500\} \times \{2, 3, 4\}$. For each $(n, r)$, we independently sample latent node embeddings $\{z_i^*\}_{i=1}^n$ from a truncated Gaussian mixture in $\mathbb{R}^r$ and degree parameters $\{\alpha_i^*\}_{i=1}^n$ from a uniform distribution. Based on the sampled latent embeddings, we generate a network from Eq. (1) with the binary logistic model. Each result is based 200 Monte Carlo repetitions. Please see more details in Appendix B.2.

**Real-world datasets.** We use four large real-world networks: (i) the user-user friendship network from the Yelp Open Dataset (Yelp, 2024); (ii) the YouTube social network dataset (Yang & Leskovec, 2012); (iii) the DBLP co-authorship network (Yang & Leskovec, 2012); and (iv) the PolBlogs network (Adamic & Glance, 2005). Details regarding the preprocessing of these datasets are in Appendix B.4.

**Baselines and implementations.** We consider two variants of our method: the diffusion-based SyNG-D and the resampling-based SyNG-R, as defined in Section 2.3. For baselines, VGAE (Kipf & Welling, 2016) is compared across all experiments. In the real-data evaluations, we also compare to GRAN (Liao et al., 2019), EDGE (Chen et al., 2023b), and the classical Erdős–Rényi model (Erdos & Rényi, 1960). Implementation details for our methods are provided in Appendix B.4. The implementations of the baselines on simulated and real-world datasets are provided separately in Appendix B.5 and Appendix B.6, respectively.

### 4.1 Generation Performance

**Evaluation metrics.** To evaluate the quality of the synthetic networks, we consider the distance between characteristic network statistics of the synthetic and observed networks. Specifically, we evaluate triangle density (Tri.), clustering coefficient (Clus.), eigenvalue distributions (Eig.), and degree centrality (DegC.). For Tri. and Clus., which are single values for each network, we compute the root-mean-square error (RMSE) and bias relative to the observed network. For Eig. and DegC., which are vectors for each network, we compute the maximum mean discrepancy (MMD), the Kolmogorov-Smirnov (KS) statistic, and the energy distance. More details are in Appendix B.3.

**Results.** Table 1 summarizes the results for simulated networks. Our methods exhibit overall superior performance compared to the baseline VGAE model, reasonably due to the fact that the data are simulated from latent space network models with a mixture distribution over latent embeddings. In the relatively large-scale setting with $(n, r) = (1500, 4)$, we note that SyNG-D outperforms SyNG-R on all metrics, indicating the effectiveness of generating novel node embeddings while capturing network structures. Here, we report only the results for $(n, r) \in \{(500, 2), (1500, 4)\}$ in Table 1 due to page constraints. Full experimental results on simulated networks are provided in Appendix B.5.

Real-world datasets allow a fairer comparison. For each method, we select the configuration that yields the best average performance across all four metrics. The results are summarized in Table 2. EDGE and GRAN ran out of memory on the Yelp dataset on a single NVIDIA GeForce RTX 4090 with memory of 24GB, and are marked "-" at the corresponding entries. This reflects that these methods

Table 1: Results on simulated networks. Eigenvalues and degree centrality are compared using energy distance (En.) and Kolmogorov-Smirnov distance (KS). Best performances are in **bold**.

| $(n, r)$ | Method | Tri. ($\times 10^{-3}$) | | Clus. ($\times 10^{-3}$) | | Eig. ($\times 10^{-2}$) | | DegC. ($\times 10^{-2}$) | |
|---|---|---|---|---|---|---|---|---|---|
| | | RMSE | Bias | RMSE | Bias | En. | KS | En. | KS |
| $(500, 2)$ | SyNG-D | 4.54 | 1.75 | 6.16 | **3.44** | **2.78** | **4.67** | **2.72** | **6.88** |
| | SyNG-R | **4.33** | **1.65** | **6.12** | 3.68 | 2.99 | 5.24 | 2.73 | 7.20 |
| | VGAE | 6.85 | -5.45 | 25.9 | -24.9 | 101 | 96.5 | 17.2 | 45.6 |
| $(1500, 4)$ | SyNG-D | **2.65** | **0.194** | **3.21** | **-0.293** | **1.55** | **2.76** | **1.51** | **3.83** |
| | SyNG-R | **2.65** | 0.605 | 3.37 | 1.28 | 2.46 | 4.22 | 1.53 | 4.00 |
| | VGAE | 6.54 | -5.68 | 25.3 | -24.0 | 110 | 98.7 | 17.5 | 45.5 |

are computationally expensive at scale and pose challenges when computational resources are limited. For most metrics, SyNG-D and SyNG-R produce networks with better-preserved characteristics. In Figure 2, we visualize the YouTube network alongside synthetic networks produced by different methods. SyNG-D and SyNG-R preserve the clustering patterns in the observed network evidently, whereas other methods do not. More results and details are in Appendix B.6.

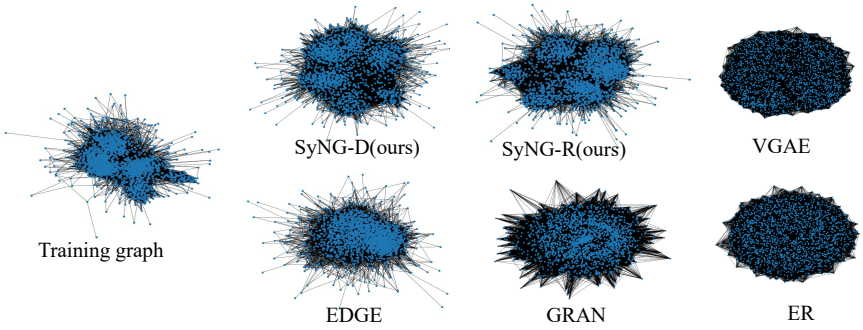



Figure 2: Visualization of the YouTube network and synthetic networks by different methods.

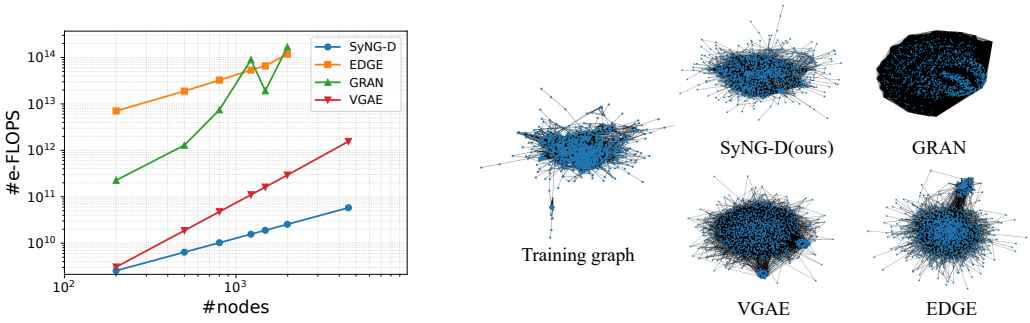



Figure 3: Efficiency comparison between different models. Left: number of e-FLOPS versus the number of nodes in the observed graph. Right: Synthetic networks from different methods on the DBLP dataset.

### 4.2 Efficiency

**Evaluation metrics and configuration.** We compare training efficiency between SyNG-D and the baseline methods through a computational workload metric we define as the equivalent floating-point operations (e-FLOPs). To evaluate the training efficiency of different methods, the e-FLOPs

Table 2: Results on real-world networks. Eigenvalues and degree centrality are evaluated using MMD. Each entry is the average over 200 generated networks. Best performances are in **bold**.

(a) YouTube

| Method | Tri ($\times 10^{-4}$) | Clus ($\times 10^{-2}$) | Eig ($\times 10^{-2}$) | DegC ($\times 10^{-2}$) |
|---|---|---|---|---|
| E-R | 1.05 | 16.3 | 25.96 | 69.22 |
| VGAE | 0.559 | 8.0 | 17.20 | 27.83 |
| GRAN | 1.13 | 10.1 | 7.88 | 7.86 |
| EDGE | **0.785** | 14.4 | 4.44 | 7.84 |
| SyNG-D | 0.967 | 2.15 | **2.61** | 2.13 |
| SyNG-R | 1.11 | **2.08** | 4.47 | **1.10** |

(b) DBLP

| Method | Tri ($\times 10^{-4}$) | Clus ($\times 10^{-1}$) | Eig ($\times 10^{-1}$) | DegC ($\times 10^{-1}$) |
|---|---|---|---|---|
| E-R | 7.96 | 8.94 | 4.409 | 8.716 |
| VGAE | **0.578** | **0.136** | 1.196 | 2.797 |
| GRAN | 7.99 | 8.92 | 1.757 | 2.973 |
| EDGE | 2.20 | 1.33 | 1.387 | 0.992 |
| SyNG-D | 1.58 | 0.622 | 0.879 | 0.926 |
| SyNG-R | 1.43 | 0.378 | **0.799** | **0.748** |

(c) Yelp

| Method | Tri ($\times 10^{-4}$) | Clus ($\times 10^{-2}$) | Eig ($\times 10^{-2}$) | DegC ($\times 10^{-2}$) |
|---|---|---|---|---|
| E-R | 8.12 | 14.5 | 24.53 | 77.36 |
| VGAE | 7.41 | 10.3 | 17.58 | 33.74 |
| GRAN | - | - | - | - |
| EDGE | - | - | - | - |
| SyNG-D | 2.00 | 1.77 | **2.50** | 6.72 |
| SyNG-R | **0.778** | **0.756** | 4.69 | **0.62** |

(d) PolBlogs

| Method | Tri ($\times 10^{-4}$) | Clus ($\times 10^{-2}$) | Eig ($\times 10^{-2}$) | DegC ($\times 10^{-2}$) |
|---|---|---|---|---|
| E-R | 3.22 | 20.4 | 40.10 | 79.24 |
| VGAE | 2.07 | 3.26 | 31.10 | 35.33 |
| GRAN | 1.57 | 11.4 | 9.04 | 13.86 |
| EDGE | 0.790 | 5.69 | 4.69 | **0.00** |
| SyNG-D | **0.710** | **1.90** | **2.58** | 0.97 |
| SyNG-R | 0.830 | 2.45 | 3.01 | 0.92 |

metric counts the total number of floating-point operations or their approximate equivalents during the training process. Note that SyNG-D and the baseline methods consist of neural nets and tree structures in their model architecture. For neural nets, e-FLOPs counts floating-point operations, whereas for tree structures, e-FLOPs counts node visits. We use e-FLOPs to evaluate the efficiency of different methods, since they are less dependent on the hardware environment. For the more straightforward wall-clock training time, we present the results along with the hardware environment used by each method in Appendix B.7. In addition to four real-world datasets, we also compare these methods on a group of simulated networks with sizes of 200, 500, and 800. For a fair comparison, we keep the latent dimension of all models at 4. Specifically, for SyNG-D and VGAE, the latent dimension corresponds to the dimension of the latent space. For GRAN, it is the output dimension of the attention layers. For EDGE, it is the dimension of the hidden layer in the score network.

**Results.** Figure 3 summarizes the results. We have several observations. In terms of e-FLOPs, SyNG-D attains the lowest cost across all settings and remains stable as network size grows. In terms of synthetic network quality, SyNG-D best preserves the overall structure of the observed networks; VGAE, while computationally comparable to SyNG-D, does not match its quality. Third, GRAN and EDGE require substantially longer training times, yet their performance is not as satisfactory, especially for GRAN. Overall, SyNG-D is both effective and efficient for this task, producing realistic synthetic networks with a computationally lightweight training process. Additional results regarding efficiency comparisons, along with further details, are provided in Appendix B.7.

## 5 Conclusion

In this work, we address the challenge of synthesizing realistic networks at scale while preserving salient structural properties using Synthetic Network Generation via Latent Embedding Reconstruction (SyNGLER), a general and efficient framework that learns low-dimensional latent node embeddings from a single observed network and then trains a distribution-free generator in the learned latent space. By separating representation learning via a likelihood-based latent space approach from generative modeling, SyNGLER preserves structural information with latent space geometry where lightweight generators suffice, enabling fast training and sampling. Theoretical and empirical results both demonstrate the effectiveness of SyNGLER. Future research directions include incorporating richer supervision for conditional generation (e.g., node/edge attributes and constraints), extending to directed, dynamic, and multilayer networks, and developing rigorous privacy-preserving training and release mechanisms.

## REFERENCES


Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43, 2005.

Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

Sourav Chatterjee. Matrix estimation by universal singular value thresholding. 2015.

Minshuo Chen, Kaixuan Huang, Tuo Zhao, and Mengdi Wang. Score approximation, estimation and distribution recovery of diffusion models on low-dimensional data. In *International Conference on Machine Learning*, pp. 4672–4712. PMLR, 2023a.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

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

Fan Chung and Mary Radcliffe. On the spectra of general random graphs. *the electronic journal of combinatorics*, pp. P215–P215, 2011.

Paul Erdos and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

Cong Fu, Keqiang Yan, Limei Wang, Wing Yee Au, Michael Curtis McThrow, Tao Komikado, Koji Maruhashi, Kanji Uchino, Xiaoning Qian, and Shuiwang Ji. A latent diffusion model for protein structure generation. In *Learning on graphs conference*, pp. 29–1. PMLR, 2024.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

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

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In *International conference on artificial intelligence and statistics*, pp. 1288–1296. PMLR, 2024.

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

Keith Levin and Elizaveta Levina. Bootstrapping networks with latent space structure. *arXiv preprint arXiv:1907.10821*, 2019.

Jinming Li, Shihao Wu, Chengyu Cui, Gongjun Xu, and Ji Zhu. Statistical inference on latent space models for network data. *arXiv preprint arXiv:2312.06605v3*, 2025.

Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):33, 2018a.

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

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

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, 2023.

Zhuang Ma, Zongming Ma, and Hongsong Yuan. Universal latent space model fitting for large networks with edge covariates. *Journal of Machine Learning Research*, 21(4):1–67, 2020.

Ross A Maller, Gernot Müller, and Alex Szimayer. Ornstein–uhlenbeck processes and extensions. *Handbook of financial time series*, pp. 421–437, 2009.

Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International conference on artificial intelligence and statistics*, pp. 4474–4484. PMLR, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Guillaume Salha, Romain Hennequin, Jean-Baptiste Remy, Manuel Moussallam, and Michalis Vazirgiannis. Fastgae: Scalable graph autoencoders with stochastic subgraph decoding. *Neural Networks*, 142:1–19, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

Shihao Wu, Junyi Yang, Gongjun Xu, and Ji Zhu. Denoising diffused embeddings: a generative approach for hypergraphs. *arXiv preprint arXiv:2501.01541*, 2025.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1284–1293, 2019.

Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pp. 1–8, 2012.

Yelp. Yelp open dataset. Online resource, 2024. URL https://business.yelp.com/data/resources/open-dataset/.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. *Advances in Neural Information Processing Systems*, 37:61963–61999, 2024.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pp. 47–1. PMLR, 2022.

# APPENDIX

We provide experimental details, additional numerical results, and proofs of the theoretical results in the appendix. Appendix A collects proofs of the theoretical results in Section 3. Appendix B contains complete experimental details. Specifically, Appendix B.1 covers the details of the estimation algorithms. Appendix B.2 specifies the setup for the simulated networks and provides details on the real-world networks, as well as their preprocessing pipelines. Appendix B.4 contains implementation details for both SyNGLER and the baselines, including the device environments on which they are implemented and the hyperparameter configurations used. Appendix B.5 and Appendix B.6 collect additional results for the simulated networks and the real-world networks, respectively. Appendix B.7 provides additional numerical results on the efficiency analysis of different methods.

## A PROOFS IN SECTION 3

**Lemma A.1** (Theorem 5 in Chung & Radcliffe (2011)). *Let $X_1, \dots, X_m$ be independent random $n \times n$ Hermitian matrices. Assume $\|X_i - \mathbb{E}X_i\| \le M$ for all $i$, and put*

$$\nu^2 \;=\; \Big\|\sum_{i=1}^{m} \mathrm{Var}(X_i)\Big\|.$$

*Let $X = \sum_{i=1}^{m} X_i$. Then for any $a > 0$,*

$$\mathbb{P}\big(\,\|X - \mathbb{E}X\| > a\,\big) \;\le\; 2n \exp\!\left(-\frac{a^2}{2\nu^2 + \frac{2}{3}Ma}\right).$$

**Lemma A.2.** *Under the model in Eq. (1) and Assumption 2.1, let $\partial l^* \in \mathbb{R}^{n\times n}$ be the matrix where each entry is $\partial l^*_{ij} = l'_{A_{ij}}(\pi^*_{ij})$. Then we have that*

$$\|\partial l^*\| \;=\; O_p\big(w_n^{1/2}\, n^{1/2}\, (\log n)^{1/2}\big).$$

*Proof of Lemma A.2.* Let $E^{i,j}$ be the $n \times n$ matrix with 1 in the $(i,j)$ and $(j,i)$ positions and 0 elsewhere. Denote $p^*_{ij} = \mathbb{E}_A[l'_{A_{ij}}(\pi^*_{ij})$. To use Lemma A.1, write $\partial l^*$ as the sum of matrices $A^{i,j}$ defined as

$$A^{i,j} = (A_{ij} - p^*_{ij})\, E^{i,j}, \qquad 1 \le i < j \le n,$$

so that $\partial l^* = \sum_{i=1}^{n}\sum_{j=i+1}^{n} A^{i,j}$. Note that $\|A^{i,j}\| \le 1$, $\mathbb{E}[A^{i,j}] = 0_{n\times n}$, and

$$\mathbb{E}\big[(A^{i,j})^2\big] = \big(p^*_{ij} - (p^*_{ij})^2\big)\,(E^{i,i} + E^{j,j}).$$

Let

$$\nu^2 = \left\|\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathbb{E}\big[(A^{i,j})^2\big]\right\| = \left\|\sum_{i=1}^{n}\sum_{j=i+1}^{n} \big(p^*_{ij} - (p^*_{ij})^2\big)(E^{i,i} + E^{j,j})\right\|.$$

Then

$$\begin{aligned}\nu^2 &= \left\|\sum_{i=1}^{n}\Big(\sum_{j=i+1}^{n}\big(p^*_{ij} - (p^*_{ij})^2\big)\Big)E^{i,i} + \sum_{j=2}^{n}\Big(\sum_{i=1}^{j-1}\big(p^*_{ij} - (p^*_{ij})^2\big)\Big)E^{j,j}\right\| \\ &\le 2\max\left\{\max_{i=1,\dots,n}\sum_{j=1}^{n}\big(p^*_{ij} - (p^*_{ij})^2\big)\right\} \;\le\; 2n\max_{i,j}\big(p^*_{ij} - (p^*_{ij})^2\big) \;\le\; 2n\max_{i,j} p^*_{ij}.\end{aligned}$$

For $\epsilon' > 0$, we set $a = \sqrt{5n\max_{i,j} p^*_{ij}\,\log\big(2n/\epsilon'\big)}$. By Assumption 2.1, for sufficiently large $n$, it holds that

$$n\max_{i,j} p^*_{ij} \;\ge\; 2\sqrt{\tfrac{5}{3}\, n\max_{i,j} p^*_{ij}\,\log\big(2n/\epsilon'\big)}.$$

Applying Lemma A.1, we obtain

$$\mathbb{P}(\|\partial l^*\| > a) \le 2n \exp\left\{ -\frac{5n \max_{i,j} p^*_{ij} \log(2n/\epsilon')}{4n \max_{i,j} p^*_{ij} + 2\sqrt{\frac{5}{3} n \max_{i,j} p^*_{ij} \log(2n/\epsilon')}} \right\}$$
$$\le 2n \exp\{-\log(2n/\epsilon')\} = \epsilon'.$$

Noting that $\max_{i,j} p^*_{ij} \asymp w_n$ as $n \to \infty$, there exists $M' > 0$ such that, for any $\epsilon' > 0$,

$$\mathbb{P}\Big( \|\partial l^*\| \ge M' \sqrt{n w_n \log(n/\epsilon')} \Big) \le \epsilon',$$

which concludes the proof. □

*Proof of Theorem 3.1.* Note that the marginal distribution of the network $A$ can be expressed as the product of the conditional distribution and the marginal distribution of the latent positions. Because of the identifiability issue of $Z$ in model Eq. (1), we consider any transform $\mathcal{T} : \Phi = (Z, \alpha) \mapsto (ZU, \alpha) \in \mathbb{R}^{n \times (r+1)}$, where $U$ is a $r$-dimensional rotation matrix. The transform $\mathcal{T}$ applied on $\Phi$ does not change the conditional distribution of $A$. We define $\mathbb{P}_{\mathcal{T}\Phi^*}$ as the distribution of $\mathcal{T}\Phi^*$, then

$$\begin{aligned} d_{\mathrm{KL}}(\mathbb{P}_A \,\|\, \mathbb{P}^A_{\tilde{A}}) &= \mathbb{E}_{\mathbb{P}_{A \mid \Phi} \times \mathbb{P}_0^{\otimes n}} \Big[ \log \frac{\mathbb{P}_{A|\mathcal{T}\Phi^*}}{\mathbb{P}_{\tilde{A}|\tilde{\Phi}}}(A \mid \Phi) + \log \frac{\mathbb{P}_{\mathcal{T}\Phi^*}}{\mathbb{P}^A_{\tilde{\Phi}}}(\Phi) \Big] \\ &= \mathbb{E}_{\mathbb{P}_{A \mid \Phi} \times \mathbb{P}_0^n} \Big[ 2 \sum_{i<j} \log \frac{p(A_{ij} \mid z_i^\top z_j + \alpha_i + \alpha_j + \rho^*)}{p(\widetilde{A}_{ij} \mid z_i^\top z_j + \alpha_i + \alpha_j + \hat{\rho})} \Big] + d_{\mathrm{KL}}(\mathbb{P}_{\mathcal{T}\Phi^*} \,\|\, \mathbb{P}^A_{\tilde{\Phi}}) \\ &= n^2 \, \mathbb{E}_{\mathbb{P}_{A_{12} \mid \phi_1, \phi_2}} \Big[ \log \frac{p(A_{12} \mid z_1^\top z_2 + \alpha_1 + \alpha_2 + \rho^*)}{p(A_{12} \mid z_1^\top z_2 + \alpha_1 + \alpha_2 + \hat{\rho})} \Big] + d_{\mathrm{KL}}(\mathbb{P}_{\mathcal{T}\Phi^*} \,\|\, \mathbb{P}^A_{\tilde{\Phi}}). \end{aligned}$$

Here the last equality holds because of the exchangeability of the distributions of $(\phi_1, \phi_2, \ldots, \phi_n)$. We decompose the K-L divergence in the right-hand side as follows

$$\begin{aligned} d_{\mathrm{KL}}(\mathbb{P}_{\mathcal{T}\Phi^*} \,\|\, \mathbb{P}^A_{\tilde{\Phi}}) &= n \cdot \mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}} \Big[ \log \frac{\mathbb{P}_0^{\mathcal{T}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi) \Big] \\ &= n \cdot \mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}} \Big[ \log \frac{\mathbb{P}_0^{\mathcal{T}}}{\mathbb{P}_{\hat{\phi}}}(\phi) + \log \frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi) \Big] \\ &= n \cdot \Big( \mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}} \Big[ \log \frac{\mathbb{P}_0^{\mathcal{T}}}{\mathbb{P}_{\hat{\phi}}}(\phi) \Big] + \mathbb{E}_{\mathbb{P}_0^{\mathcal{T}}} \Big[ \log \frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi) \Big] - \mathbb{E}_{\mathbb{P}_{\hat{\phi}}} \Big[ \log \frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi) \Big] \\ &\qquad + \mathbb{E}_{\mathbb{P}_{\hat{\phi}}} \Big[ \log \frac{\mathbb{P}_{\hat{\phi}}}{\mathbb{P}^A_{\tilde{\phi}}}(\phi) \Big] \Big). \end{aligned}$$

Here $\mathbb{P}_{\hat{\phi}}$ is the marginal distribution of $\hat{\phi}_i$ for each $i$, which is the same for all $i$ because of the exchangeability. Minimizing over all transform $\mathcal{T}$ concludes the proof of the theorem. □

*Proof of Theorem 3.2.* We denote $\theta_{ij} = z_i^\top z_j + \alpha_i + \alpha_j$. The Lipschitz continuity of $l_{ij}(\pi) = \log p(A_{ij} \mid \pi)$ in $\pi$ implies that

$$\big| \log p(A_{ij} \mid \theta_{ij} + \rho^*) - \log p(A_{ij} \mid \theta_{ij} + \hat{\rho}) \big| \le M |\rho^* - \hat{\rho}|.$$

Now it suffices to bound $|\rho^* - \hat{\rho}|$. Let $\pi_{ij} = z_i^\top z_j + \alpha_i + \alpha_j + \rho$ and $\hat{\pi}$ be its estimated version. Applying Taylor's expansion to each $l_{ij}$ at $\pi^*_{ij}$ yields that

$$\sum_{i<j} l_{ij}(\hat{\pi}_{ij}) = \sum_{i<j} l_{ij}(\pi^*_{ij}) + \sum_{i<j} l'_{ij}(\pi^*_{ij})(\hat{\pi}_{ij} - \pi^*_{ij}) + \frac{1}{2} \sum_{i<j} l''_{ij}(\xi_{ij})(\hat{\pi}_{ij} - \pi^*_{ij})^2,$$

where each $\xi_{ij} \in [\min\{\hat{\pi}_{ij}, \pi^*_{ij}\}, \max\{\hat{\pi}_{ij}, \pi^*_{ij}\}]$. Using the optimality condition, it holds that $\sum_{i<j} l_{ij}(\hat{\pi}_{ij}) \geq \sum_{i<j} l_{ij}(\pi^*_{ij})$. And therefore

$$\begin{aligned}\sum_{i<j} l'_{ij}(\pi^*_{ij})(\hat{\pi}_{ij} - \pi^*_{ij}) &\geq \sum_{i<j} -l''_{ij}(\xi_{ij})(\hat{\pi}_{ij} - \pi^*_{ij})^2 \\ &\geq \frac{1}{4} w_n \cdot e^{-2R^2} \sum_{i<j} |\hat{\pi}_{ij} - \pi^*_{ij}|^2. \end{aligned} \tag{5}$$

To facilitate the matrix inequalities, we define $\partial l^*, \Pi^*, \hat{\Pi} \in \mathbb{R}^{n\times n}$ such that $\partial l^*_{ij} = l'_{A_{ij}}(\pi^*_{ij})$, $\Pi^*_{ij} = \pi^*_{ij} = z_i^{*\top} z^*_j + \alpha^*_i + \alpha^*_j + \rho^*_n$, and $\hat{\Pi}_{ij} = \hat{\pi}_{ij} = \hat{z}_i^\top \hat{z}_j + \hat{\alpha}_i + \hat{\alpha}^*_j + \hat{\rho}$. Then, we can upper bound the left-handed-side in Eq. (5) as

$$\begin{aligned}\sum_{i<j} l'_{ij}(\pi^*_{ij})(\hat{\pi}_{ij} - \pi^*_{ij}) &\leq |\langle \partial l^*, \hat{\Pi} - \Pi^* \rangle| \\ &\leq \sqrt{2r+3} \cdot \|\partial l^*\| \cdot \|\hat{\Pi} - \Pi^*\|_{\mathrm{F}}. \end{aligned} \tag{6}$$

Last inequality holds since $\hat{\Pi} - \Pi^*$ has rank at most $2r+3$. Invoking Lemma A.2, we have that $\|\partial l^*\| = O_p(w_n^{1/2} n^{1/2} \log n)$. Combining Eqs. (5) and (6) yields that $\|\hat{\Pi} - \Pi^*\|_{\mathrm{F}} = O_p(w_n^{-1/2} n^{1/2} \log n)$.

Before obtaining the estimation error of $\hat{\rho}$, we need to involve an identifiability transform over $Z^*, \alpha^*$, since their sample average is not necessarily zero. We consider $Z^\dagger = Z^* - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top Z^*$ and $\alpha^\dagger = \alpha^* + \frac{1}{2n} Z^* Z^{*\top} \mathbf{1}_n - \frac{1}{n}\mathbf{1}_n^\top \alpha^* \cdot \mathbf{1}_n - \frac{1}{2n}\mathbf{1}_n^\top Z^* Z^{*\top}\mathbf{1}_n \cdot \mathbf{1}_n$, and $\rho^\dagger = \rho^* + n^{-1}\mathbf{1}_n^\top \alpha^*$. Then, it is evident that

$$Z^{\dagger\top} Z^\dagger + \alpha^\dagger \mathbf{1}_n^\top + \mathbf{1}_n \alpha^{\dagger\top} + \rho^\dagger \mathbf{1}_n\mathbf{1}_n^\top = Z^* Z^{*\top} + \alpha^* \mathbf{1}_n^\top + \mathbf{1}_n \alpha^{*\top} + \rho^* \mathbf{1}_n\mathbf{1}_n^\top.$$

Additionally, we have that $Z^{\dagger\top}\mathbf{1}_n = 0_n$ and $\mathbf{1}_n^\top \alpha^\dagger = 0$.

On the other hand, we can expand $\|\hat{\Pi} - \Pi^*\|_{\mathrm{F}}^2$ as

$$\begin{aligned}\|\hat{\Pi} - \Pi^*\|_{\mathrm{F}}^2 &= \|\widehat{Z}\widehat{Z}^\top - Z^\dagger Z^{\dagger\top}\|_{\mathrm{F}}^2 + \|(\hat{\alpha} - \alpha^\dagger)\mathbf{1}_n^\top + \mathbf{1}_n(\alpha^\dagger - \hat{\alpha})^\top\|_{\mathrm{F}}^2 + n^2|\hat{\rho} - \rho^\dagger|^2 \\ &\quad + 2\langle \widehat{Z}\widehat{Z}^\top - Z^\dagger Z^{\dagger\top}, (\hat{\alpha} - \alpha^\dagger)\mathbf{1}_n^\top + \mathbf{1}_n(\hat{\alpha} - \alpha^\dagger)^\top \rangle \\ &\quad + 2\langle \widehat{Z}\widehat{Z}^\top - Z^\dagger Z^{\dagger\top}, (\hat{\rho} - \rho^\dagger)\mathbf{1}_n\mathbf{1}_n^\top \rangle \\ &\quad + 2\langle (\hat{\alpha} - \alpha^\dagger)\mathbf{1}_n^\top + \mathbf{1}_n(\hat{\alpha} - \alpha^\dagger)^\top, (\hat{\rho} - \rho^\dagger)\mathbf{1}_n\mathbf{1}_n^\top \rangle \\ &= \|\widehat{Z}\widehat{Z}^\top - Z^\dagger Z^{\dagger\top}\|_{\mathrm{F}}^2 + 2n\|\hat{\alpha} - \alpha^\dagger\|^2 + n^2|\hat{\rho} - \rho^\dagger|^2 \\ &\geq n^2|\hat{\rho} - \rho^\dagger|^2. \end{aligned}$$

Here the second line holds because $Z^{\dagger\top}\mathbf{1}_n = 0_n$ and $\mathbf{1}_n^\top \alpha^\dagger = 0$. In conclusion, we have that $|\hat{\rho} - \rho^\dagger| = O_p(w_n^{-1/2} n^{-1/2} \cdot \log n)$. On the other hand, we have that $|\rho^\dagger - \rho^*| = O(n^{-1/2})$ because of the boundedness and i.i.d. condition of $\alpha^*$. Therefore, we have that $|\hat{\rho} - \rho^*| = O_p(w_n^{-1/2} n^{-1/2} \log n)$.

□

*Proof of Lemma 3.1.* For fixed $\phi \in \mathcal{G}$, using Hoeffding's inequality yields that

$$\mathbb{P}\Big(|q_\gamma(\phi) - \hat{q}_\gamma(\phi)| \geq t\Big) \leq 2\exp(-2nt^2).$$

Using the union bound over all $(R/\gamma)^{r+1}$ points in $\mathcal{G}_\gamma$ yields that

$$\mathbb{P}\Big(\max_{\phi\in\mathcal{G}_\gamma} |q_\gamma(\phi) - \hat{q}_\gamma(\phi)| \geq t\Big) \leq 2(R/\gamma)^{r+1}\exp(-2nt^2).$$

Therefore, setting $t = \sqrt{(r+1)\log(R/\gamma) + \log(2/\delta)}/\sqrt{2n}$ yields the desired result. □

*Proof of Theorem 3.4.* Using Theorem 3.3 in Li et al. (2025), we have for any $\epsilon', \delta > 0$, there exists a constant $M > 0$ and a rotation matrix $U$ such that

$$\mathbb{P}\Big( \max_i \|\hat{\phi}_i - U\phi_i^*\|_2 > M(w_n n)^{-1/2+\epsilon'} \Big) < \delta,$$

Therefore, we have that $\mathbb{P}(\max_i \|\hat{\phi}_i - U\phi_i^*\|_2 \le \gamma_n) \to 1$ with $\gamma_n = \Omega((w_n n)^{-1/2+\epsilon})$ and fixed $\epsilon > 0$. On the event $\{\|\hat{\phi}_i - U\phi_i^*\|_2 \le \gamma_n\}$, we have that $\check{q}_{\gamma_n} - q_{\gamma_n} = 0$. □

# B Supplemental Materials for Experiments

## B.1 Deferred Algorithms

**Estimation in the latent space model.** Suppose that we observe a network $A$, and we want to fit a latent space network model on $A$ with proper conditional model $p(\cdot \mid \cdot)$ and candidate latent dimension $r$. We use the following Algorithm 2 to solve Eq. (2).

**Algorithm 2** Projected Gradient Descent

**Require:** Network observation $A \in \mathbb{R}^{n\times n}$, model $p(\cdot|\cdot)$, stepsizes $\eta_Z, \eta_\alpha > 0$, number of iterations $N \in \mathbb{N}$;

1: **for** $i = 0$ **to** $N - 1$ **do**
2: $\quad \Pi \leftarrow ZZ^\top + \alpha\mathbf{1}^\top + \mathbf{1}\alpha^\top$;
3: $\quad Z \leftarrow Z + 2\eta_Z \, \partial_\pi \, p(A|\Pi)Z$;
4: $\quad \alpha \leftarrow \alpha + 2\eta_\alpha \, \partial_\pi \, p(A|\Pi)\mathbf{1}_n$;
5: $\quad Z \leftarrow \big(Z - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top Z\big)R$, where $R \in \mathbb{R}^{r\times r}$ is the orthonormal matrix such that $n^{-1}(Z - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top Z)^\top(Z - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top Z)$.
6: **end for**
7: **return** $\hat{Z} = Z, \hat{\alpha} = \alpha - \alpha^\top\mathbf{1}_n/n, \hat{\rho} = \alpha^\top\mathbf{1}_n$.

The convergence of Algorithm 2 in the well-specified setting can be found in Ma et al. (2020). In practice, we need to use a proper initialization for $Z$ and $\alpha$. And we use the output of universal singular value thresholding (USVT) (Chatterjee, 2015) as the initialization of $Z$ and $\alpha$. The detail of this initialization algorithm can be found in Ma et al. (2020).

## B.2 Datasets Details

**Simulated Datasets.** In the simulated datasets evaluation, we consider $(n, r) \in \{500, 1000, 1500\} \times \{2, 3, 4\}$. For each $(n, r)$ pair and each replicate $t = 1, \ldots, 200$, we generate an undirected simple graph $A \in \{0, 1\}^{n\times n}$ as follows.

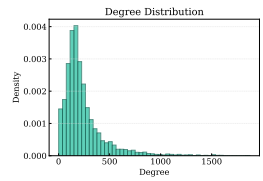
We first draw the degree parameters $\alpha_i \overset{\text{i.i.d.}}{\sim} \text{Unif}([-1/2, 1/2])$ for $i = 1, \ldots, n$ and set $\alpha = (\alpha_1, \ldots, \alpha_n)$ . Let $\widetilde{z}_i \in \mathbb{R}^r$ be i.i.d. realizations of $\text{proj}_{[-2/\sqrt{r},2/\sqrt{r}]^r} \# \mathcal{N}_r(0, I_r/r)$ (i.e., a scaled Gaussian distribution truncated to $[-2/\sqrt{r}, 2/\sqrt{r}]^r$). We then independently draw two centers $v^{(1)}, v^{(2)} \in \mathbb{R}^r$ from $\text{Unif}([-1, 1]^r)$. For each node, we independently sample a label $L_i$ with $\mathbb{P}(L_i = 1) = \mathbb{P}(L_i = 2) = 1/2$ for $i = 1, \ldots, n$. Finally, we set $z_i' = \widetilde{z}_i + v^{(L_i)}$ and $z_i = z_i' \cdot (n^{-1}\| \sum_i z_i' z_i'^\top \|_{\text{F}})^{-1/2}$.

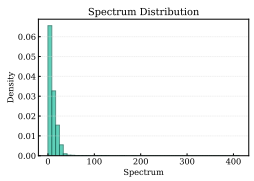
Given the latent positions and the degree parameters, we generate the network edges. We set the sparsity parameter $\rho^* = 0$. For each pair of nodes $1 \le i < j \le n$, we calculate $p_{ij} = \sigma(\alpha_i + \alpha_j + z_i^\top z_j)$. Then we independently sample $A_{ij} = A_{ij} \sim \text{Bernoulli}(p_{ij})$ for $i < j$ and set $A_{ii} = 0$ for all $i \le n$.

**Real-world datasets.** We evaluate on four networks spanning thousands to millions of nodes. For Yelp, YouTube, and DBLP, whose full graphs are extremely large and highly sparse, we construct tractable training sets by extracting high-degree nodes and then taking the largest connected component (LCC).

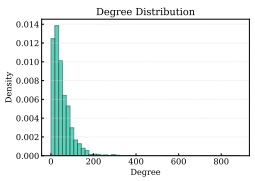
Table 3: Dataset statistics for Yelp, YouTube, DBLP and PolBlogs.

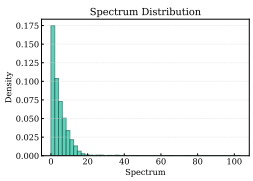
| Dataset | Original Dataset | | Subgraph | | Statistics | | |
|---|---|---|---|---|---|---|---|
| | Nodes | Edges | Nodes | Edges | Density | Clustering Coef. | Triangle Density |
| Yelp | 906,179 | 7,305,874 | 4,530 | 541,655 | 0.0527 | 0.1976 | 0.0010 |
| YouTube | 1,134,890 | 2,987,624 | 1,991 | 51,756 | 0.0261 | 0.1891 | 0.0002 |
| DBLP | 317,080 | 1,049,866 | 1,481 | 18,901 | 0.0172 | 0.9116 | 0.0008 |
| PolBlogs | 1,490 | 19,090 | 1,222 | 16,714 | 0.0224 | 0.2259 | 0.0003 |



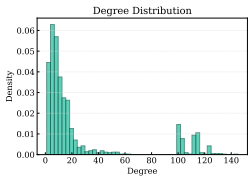



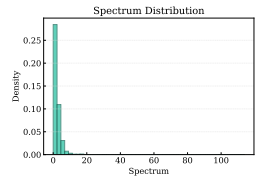

(a) Yelp dataset.



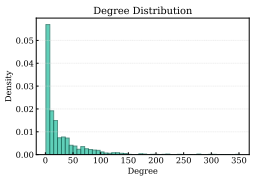



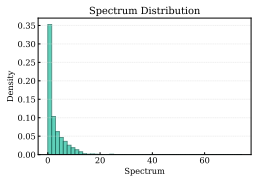

(b) YouTube dataset.




(c) DBLP dataset.




(d) PolBlogs dataset..

Figure 4: Degree and eigenvalue distributions for four real-world datasets.

In the Yelp and YouTube datasets, nodes represent users and an undirected edge between users represents a social tie (friendship/subscription). In the DBLP dataset, nodes represent authors, and an edge connects two authors if they have coauthored at least one paper. In the PolBlogs dataset, nodes represent U.S. political blogs from the 2004 election blogosphere and two blogs are connected if one of them contains a link to the other.

Since the Yelp, YouTube, and DBLP datasets are very large and contain many low-degree nodes, we sample induced subgraphs for tractable evaluation. Our general procedure is to rank nodes by degree, take the induced subgraph on the top-$k$ nodes (for $k$ between 1,000 and 5,000), and then extract the largest connected component (LCC). For Yelp, we select the top $0.5\%$ of users by degree, yielding an LCC of 4,530 nodes and 541,655 edges. For YouTube and DBLP, we take the top 2,000 and 1,500 nodes, resulting in LCCs of 1,991 and 1,481 nodes, respectively. To avoid out-of-memory (OOM) issues for some baseline methods, we cap most subgraphs at $\leq 2,000$ nodes. The PolBlogs network is relatively smaller, so we use its full LCC of 1,222 nodes and 16,714 edges. For all networks, we symmetrize edges and remove self-loops. Key statistics for the original and the extracted graphs are in Table 3, with degree and eigenvalue(spectrum) distributions shown in Figure 4.

### B.3 Evaluation Details

**Metrics for similarity.** We assess the quality of the generated networks by comparing some metrics that capture both numerical and structural aspects of a network. For the *numerical* characteristics, we use the triangle density and the global clustering coefficient. For the *structural* characteristics, we consider the distribution of degree centralities and the eigenvalues of the adjacency matrix. For any network adjacency $A$, we consider the following numerical characteristics:

- The triangle density: $\mathsf{TD}(A) = \mathsf{NT}(A)/\binom{n}{3}$ where $\mathsf{NT}(A) = \frac{1}{6}\operatorname{tr}(A^3)$ is the number of triangles in the graph;
- The global clustering coefficient: $\mathsf{GC}(A) = 3\,\mathsf{NT}(A)/\sum_{i=1}^{n}\binom{d_i}{2}$ where $d_i = \sum_{j\neq i} A_{ij}$ is the degree of node $i$.

For each input network and generative model, we generate $S = 200$ independent networks $\widetilde{A}_1, \ldots, \widetilde{A}_S$ and compute the empirical distribution of each numerical characteristic. Specifically, for a numerical characteristic $f$ with $f \in \{\mathsf{TD}, \mathsf{GC}\}$ and a collection of generated networks $\{\widetilde{A}^{(s)}\}_{s=1}^{S}$, we compute

$$\mathrm{RMSE}_f = \Big(\frac{1}{S}\sum_{s=1}^{S}\big(f(A) - f(\widetilde{A}^{(s)})\big)^2\Big)^{1/2},$$

$$\mathrm{MAE}_f = \frac{1}{S}\sum_{s=1}^{S}\big|f(A) - f(\widetilde{A}^{(s)})\big|,$$

$$\mathrm{Bias}_f = \frac{1}{S}\sum_{s=1}^{S}\big(f(\widetilde{A}^{(s)}) - f(A)\big).$$

For the structural characteristics, we consider the following:

- The degree centrality: $\mathsf{DC}(A) = (d_1, \ldots, d_n)$, where $d_i = \sum_{j\neq i} A_{ij}$ is the degree of node $i$;
- The eigenvalues: $\mathsf{EV}(A) = (\lambda_1, \ldots, \lambda_n)$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of $A$.

For two vectors, we consider the Wasserstein distance 1-distance, the Kolmogorov–Smirnov distance, energy distance and maximum mean discrepancy (MMD) as follows:

$$W^1(u,v) = \frac{1}{n}\sum_{i=1}^{n}|u_{(i)} - v_{(i)}|,$$

$$\mathrm{KS}(u,v) = \sup_{x\in\mathbb{R}}\Big|\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{u_i \leq x\} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{v_i \leq x\}\Big|,$$

$$\mathrm{ED}(u,v) = \frac{2}{n^2}\sum_{i,j=1}^{n}|u_i - v_j| - \frac{1}{n^2}\sum_{i,j=1}^{n}|u_i - u_j| - \frac{1}{n^2}\sum_{i,j=1}^{n}|v_i - v_j|,$$

$$\mathrm{MMD}(u,v) = \frac{1}{n^2}\sum_{i,j=1}^{n}k(u_i,u_j) + \frac{1}{n^2}\sum_{i,j=1}^{n}k(v_i,v_j) - \frac{2}{n^2}\sum_{i,j=1}^{n}k(u_i,v_j),$$

where $u_{(1)} \leq u_{(2)} \leq \cdots \leq u_{(n)}$ are the order statistics of $u$, and $k(x,y) = \exp(-|x-y|^2/2)$ is the standard Gaussian RBF kernel. For a structural characteristic $f$ with $f \in \{\mathsf{DC}, \mathsf{EV}\}$ and a discrepancy metric $d$ with $d \in \{W^1, \mathrm{KS}, \mathrm{ED}, \mathrm{MMD}\}$, we compute the average distance between the original network and the generated networks as $\bar{d}_f = \frac{1}{S}\sum_{s=1}^{S} d\big(f(A), f(\widetilde{A}^{(s)})\big)$.

**Evaluation pipeline.** For single input network $A$, we generate $S = 200$ independent networks and calculate the above metrics. In the simulated dataset setting, we repeat this procedure for 200 input networks and report the Monte Carlo mean and standard deviation of each metric. In the real-world dataset setting, we directly report the averaged metrics for each input network and its associated standard deviation.

### B.4 Implementation Details

**Implementation of SyNGLER.** For the SyNG-D, we use `ForestDiffusion` (Jolicoeur-Martineau et al., 2024) to construct the score approximation. Table 4 lists all the hyperparameters for `ForestDiffusion` throughout our experiments.

**Experimental environment.** All experiments are conducted on NVIDIA GeForce RTX 4090 (24 GB) GPUs and 384 CPU cores.

The `ForestDiffusion` module is parallelized on the CPU and executes entirely on host cores. We deploy VGAE and SyNG-D models on CPUs. EDGE and GRAN are deployed on a single GPU, according to the default configuration in the original codebase.

Table 4: `ForestDiffusion` Hyperparameters For Data Generation.

| Category | Hyperparameter | Simulated Data | Real-world data | Description |
|---|---|---|---|---|
| ForestDiffusion | $n_t$ | 50 | 100 | Number of diffusion time steps. |
| | `duplicate K` | 100 | 100 | Sample duplication factor for training data. |
| | `diffusion type` | `vp` | `vp` | Use variance-preserving (VP) diffusion. |
| XGBoost | `max depth` | 7 | 7 | Maximum tree depth. |
| | `number of estimators` | 100 | 100 | Number of boosting trees. |
| | `eta` | 0.3 | 0.3 | Learning rate. |
| | `tree method` | `hist` | `hist` | Histogram-based tree construction. |
| | `regression lambda` | 0.0 | 0.0 | L2 regularization parameter. |
| | `regression alpha` | 0.0 | 0.0 | L1 regularization parameter. |
| | `subsample` | 1.0 | 1.0 | Row subsampling ratio per tree. |

**Implementation for baselines.** For the VGAE, EDGE and GRAN, we use the codebases hosted in gae, graph-generation-EDGE, GRAN, respectively. For each baseline method, we adopt the default models in the corresponding codebase without further clarification. In the real data experiments, we include the results for each method with several different configurations. For our SyNG-D and SyNG-R, we vary the number of the latent dimension $r$ from 2 to 6. For VGAE, we vary the number of the embeddings consecutively from 2 to 6, and include the default setting 16. For GRAN, we choose the dimension of the hidden layer from 128, 256 and 512. Other methods remain their default configurations.

### B.5 Evaluation Results on Simulated Network

**Structural characteristics.** The following Table 5 and Table 6 summarize the discrepancies between the numerical statistics of the generated networks and those of the input networks.

Table 5: Averaged distance between the *degree centralities* of the original network and the generated output.

| | | $n = 500$ | | | $n = 1000$ | | | $n = 1500$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Metric | Method | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 |
| W1-dist | SyNG-D | **0.008** ± 0.002 | **0.008** ± 0.002 | **0.008** ± 0.002 | **0.005** ± 0.001 | **0.005** ± 0.002 | **0.006** ± 0.002 | **0.004** ± 0.001 | **0.004** ± 0.001 | **0.004** ± 0.001 |
| | SyNG-R | **0.008** ± 0.002 | **0.008** ± 0.002 | **0.008** ± 0.002 | **0.005** ± 0.001 | 0.006 ± 0.002 | **0.006** ± 0.002 | **0.004** ± 0.001 | **0.004** ± 0.001 | **0.004** ± 0.001 |
| | VGAE | 0.049 ± 0.002 | 0.049 ± 0.003 | 0.050 ± 0.003 | 0.049 ± 0.002 | 0.050 ± 0.003 | 0.050 ± 0.003 | 0.050 ± 0.002 | 0.051 ± 0.002 | 0.051 ± 0.002 |
| KS-dist | SyNG-D | **0.069** ± 0.010 | **0.068** ± 0.010 | **0.070** ± 0.013 | **0.047** ± 0.007 | **0.047** ± 0.009 | **0.048** ± 0.009 | **0.038** ± 0.006 | **0.038** ± 0.007 | **0.038** ± 0.007 |
| | SyNG-R | 0.072 ± 0.009 | 0.072 ± 0.009 | 0.073 ± 0.012 | 0.049 ± 0.006 | 0.049 ± 0.008 | 0.051 ± 0.008 | 0.039 ± 0.005 | 0.040 ± 0.006 | 0.040 ± 0.006 |
| | VGAE | 0.456 ± 0.023 | 0.453 ± 0.024 | 0.457 ± 0.028 | 0.451 ± 0.019 | 0.451 ± 0.020 | 0.452 ± 0.021 | 0.452 ± 0.016 | 0.453 ± 0.016 | 0.455 ± 0.017 |
| Energy-dist | SyNG-D | **0.027** ± 0.006 | **0.027** ± 0.006 | **0.028** ± 0.007 | **0.019** ± 0.004 | **0.019** ± 0.005 | **0.019** ± 0.005 | **0.015** ± 0.003 | **0.015** ± 0.004 | **0.015** ± 0.004 |
| | SyNG-R | **0.027** ± 0.005 | 0.028 ± 0.005 | **0.028** ± 0.007 | **0.019** ± 0.004 | **0.019** ± 0.005 | **0.019** ± 0.005 | **0.015** ± 0.003 | **0.015** ± 0.004 | **0.015** ± 0.004 |
| | VGAE | 0.172 ± 0.006 | 0.172 ± 0.007 | 0.173 ± 0.007 | 0.172 ± 0.005 | 0.173 ± 0.006 | 0.173 ± 0.006 | 0.173 ± 0.005 | 0.175 ± 0.005 | 0.175 ± 0.005 |

Table 6: Averaged distance between the *eigenvalues* of the original network and the generated output.

| | | $n = 500$ | | | $n = 1000$ | | | $n = 1500$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| eigenvalues Distance | Method | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 |
| W1-dist | SyNG-D | **0.037** ± 0.004 | **0.043** ± 0.009 | **0.066** ± 0.037 | **0.024** ± 0.003 | **0.026** ± 0.004 | **0.036** ± 0.013 | **0.019** ± 0.002 | **0.020** ± 0.003 | **0.025** ± 0.005 |
| | SyNG-R | 0.040 ± 0.004 | 0.059 ± 0.009 | 0.098 ± 0.018 | 0.025 ± 0.003 | 0.035 ± 0.005 | 0.056 ± 0.015 | 0.020 ± 0.002 | 0.025 ± 0.003 | 0.038 ± 0.006 |
| | VGAE | 0.954 ± 0.051 | 0.970 ± 0.067 | 0.955 ± 0.078 | 0.955 ± 0.046 | 0.980 ± 0.069 | 0.991 ± 0.076 | 0.952 ± 0.050 | 0.980 ± 0.070 | 0.983 ± 0.082 |
| KS-dist | SyNG-D | **0.047** ± 0.006 | **0.048** ± 0.011 | **0.062** ± 0.029 | **0.032** ± 0.003 | **0.032** ± 0.006 | **0.038** ± 0.015 | **0.027** ± 0.003 | **0.026** ± 0.004 | **0.028** ± 0.006 |
| | SyNG-R | 0.052 ± 0.006 | 0.065 ± 0.012 | 0.096 ± 0.022 | 0.035 ± 0.003 | 0.042 ± 0.008 | 0.059 ± 0.018 | 0.028 ± 0.002 | 0.032 ± 0.005 | 0.042 ± 0.009 |
| | VGAE | 0.965 ± 0.001 | 0.961 ± 0.002 | 0.961 ± 0.004 | 0.981 ± 0.001 | 0.982 ± 0.001 | 0.981 ± 0.002 | 0.987 ± 0.001 | 0.988 ± 0.001 | 0.987 ± 0.001 |
| Energy-dist | SyNG-D | **0.028** ± 0.004 | **0.029** ± 0.007 | **0.041** ± 0.021 | **0.018** ± 0.002 | **0.018** ± 0.004 | **0.023** ± 0.011 | **0.015** ± 0.002 | **0.014** ± 0.003 | **0.016** ± 0.004 |
| | SyNG-R | 0.030 ± 0.004 | 0.039 ± 0.008 | 0.064 ± 0.016 | 0.019 ± 0.002 | 0.023 ± 0.005 | 0.036 ± 0.012 | **0.015** ± 0.001 | 0.017 ± 0.003 | 0.025 ± 0.006 |
| | VGAE | 1.015 ± 0.023 | 1.040 ± 0.031 | 1.048 ± 0.036 | 1.055 ± 0.020 | 1.078 ± 0.030 | 1.092 ± 0.032 | 1.070 ± 0.022 | 1.089 ± 0.028 | 1.099 ± 0.033 |

**Numerical characteristics.** The following Table 7 and Table 8 summarize the distances between the structural characteristics of the generated networks and the input networks.

Table 7: Similarity between the the *triangle densities* of the original network and generated network.

| $n$ | Method | RMSE | | | MAE ($\pm$ std) | | | Bias ($\pm$ std) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 |
| 500 | SyNG-D | 0.0045 | **0.0044** | 0.0050 | 0.0037 $\pm$ 0.0026 | **0.0047** $\pm$ 0.0034 | 0.0050 $\pm$ 0.0049 | 0.0018 $\pm$ 0.0042 | **0.0010** $\pm$ 0.0043 | 0.0022 $\pm$ 0.0066 |
| | SyNG-R | **0.0043** | **0.0044** | **0.0049** | **0.0035** $\pm$ 0.0026 | 0.0051 $\pm$ 0.0037 | **0.0039** $\pm$ 0.0030 | **0.0017** $\pm$ 0.0040 | 0.0013 $\pm$ 0.0042 | **0.0018** $\pm$ 0.0046 |
| | VGAE | 0.0068 | 0.0073 | 0.0071 | 0.0058 $\pm$ 0.0036 | 0.0061 $\pm$ 0.0039 | 0.0058 $\pm$ 0.0041 | -0.0054 $\pm$ 0.0042 | -0.0057 $\pm$ 0.0045 | -0.0052 $\pm$ 0.0049 |
| 1000 | SyNG-D | 0.0031 | **0.0032** | **0.0033** | 0.0032 $\pm$ 0.0017 | **0.0031** $\pm$ 0.0025 | 0.0035 $\pm$ 0.0029 | **0.0009** $\pm$ 0.0029 | **0.0005** $\pm$ 0.0032 | **0.0006** $\pm$ 0.0045 |
| | SyNG-R | **0.0030** | 0.0043 | 0.0034 | **0.0025** $\pm$ 0.0017 | 0.0034 $\pm$ 0.0026 | **0.0027** $\pm$ 0.0020 | **0.0009** $\pm$ 0.0028 | 0.0019 $\pm$ 0.0038 | 0.0010 $\pm$ 0.0032 |
| | VGAE | 0.0066 | 0.0067 | 0.0068 | 0.0058 $\pm$ 0.0032 | 0.0059 $\pm$ 0.0033 | 0.0059 $\pm$ 0.0033 | -0.0056 $\pm$ 0.0035 | -0.0056 $\pm$ 0.0038 | -0.0056 $\pm$ 0.0038 |
| 1500 | SyNG-D | 0.0031 | **0.0031** | **0.0032** | 0.0025 $\pm$ 0.0014 | **0.0024** $\pm$ 0.0019 | **0.0025** $\pm$ 0.0021 | 0.0008 $\pm$ 0.0030 | **0.0003** $\pm$ 0.0025 | **-0.0003** $\pm$ 0.0032 |
| | SyNG-R | **0.0024** | 0.0032 | 0.0034 | **0.0020** $\pm$ 0.0013 | 0.0025 $\pm$ 0.0020 | 0.0027 $\pm$ 0.0020 | **0.0004** $\pm$ 0.0023 | 0.0013 $\pm$ 0.0030 | 0.0013 $\pm$ 0.0031 |
| | VGAE | 0.0071 | 0.0065 | 0.0065 | 0.0065 $\pm$ 0.0029 | 0.0059 $\pm$ 0.0028 | 0.0058 $\pm$ 0.0031 | -0.0065 $\pm$ 0.0029 | -0.0057 $\pm$ 0.0032 | -0.0057 $\pm$ 0.0033 |

Table 8: Similarity between the the *global clustering coefficients* of the original network and generated network.

| $n$ | Method | RMSE | | | MAE ($\pm$ std) | | | Bias ($\pm$ std) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 | r=2 | r=3 | r=4 |
| 500 | SyNG-D | 0.0062 | **0.0058** | **0.0070** | **0.0051** $\pm$ 0.0034 | **0.0047** $\pm$ 0.0034 | **0.0050** $\pm$ 0.0049 | **0.0034** $\pm$ 0.0051 | **0.0021** $\pm$ 0.0054 | **0.0022** $\pm$ 0.0066 |
| | SyNG-R | **0.0061** | 0.0063 | **0.0070** | **0.0051** $\pm$ 0.0034 | 0.0051 $\pm$ 0.0037 | 0.0057 $\pm$ 0.0041 | 0.0037 $\pm$ 0.0049 | 0.0036 $\pm$ 0.0052 | 0.0043 $\pm$ 0.0056 |
| | VGAE | 0.0259 | 0.0260 | 0.0258 | 0.0249 $\pm$ 0.0072 | 0.0245 $\pm$ 0.0086 | 0.0240 $\pm$ 0.0093 | -0.0249 $\pm$ 0.0072 | -0.0245 $\pm$ 0.0086 | -0.0240 $\pm$ 0.0094 |
| 1000 | SyNG-D | **0.0040** | **0.0039** | **0.0045** | **0.0032** $\pm$ 0.0023 | **0.0031** $\pm$ 0.0025 | **0.0035** $\pm$ 0.0029 | **0.0017** $\pm$ 0.0036 | **0.0007** $\pm$ 0.0039 | **0.0006** $\pm$ 0.0045 |
| | SyNG-R | **0.0040** | 0.0043 | 0.0052 | **0.0032** $\pm$ 0.0024 | 0.0034 $\pm$ 0.0026 | 0.0041 $\pm$ 0.0032 | 0.0020 $\pm$ 0.0035 | 0.0019 $\pm$ 0.0038 | 0.0026 $\pm$ 0.0045 |
| | VGAE | 0.0259 | 0.0257 | 0.0261 | 0.0250 $\pm$ 0.0070 | 0.0243 $\pm$ 0.0085 | 0.0247 $\pm$ 0.0085 | -0.0250 $\pm$ 0.0070 | -0.0243 $\pm$ 0.0085 | -0.0247 $\pm$ 0.0085 |
| 1500 | SyNG-D | 0.0031 | **0.0031** | **0.0032** | 0.0025 $\pm$ 0.0017 | **0.0024** $\pm$ 0.0019 | **0.0025** $\pm$ 0.0021 | **0.0008** $\pm$ 0.0030 | **0.0003** $\pm$ 0.0031 | **-0.0003** $\pm$ 0.0032 |
| | SyNG-R | **0.0030** | 0.0032 | 0.0034 | **0.0024** $\pm$ 0.0017 | 0.0025 $\pm$ 0.0020 | 0.0027 $\pm$ 0.0020 | 0.0010 $\pm$ 0.0028 | 0.0013 $\pm$ 0.0030 | 0.0013 $\pm$ 0.0031 |
| | VGAE | 0.0271 | 0.0246 | 0.0253 | 0.0263 $\pm$ 0.0064 | 0.0235 $\pm$ 0.0074 | 0.0240 $\pm$ 0.0082 | -0.0263 $\pm$ 0.0064 | -0.0235 $\pm$ 0.0074 | -0.0240 $\pm$ 0.0082 |

### B.6 Evaluation Results on Real-world Datasets

In this subsection, we list all the experiment results on the real-world datasets. For each dataset, we present three tables detailing the generation quality. The first table evaluates the similarity of degree centrality distributions, the second assesses the similarity of eigenvalue distributions, and the third reports on numerical characteristics such as global clustering coefficient and triangle density.

**YouTube dataset.** Tables 9 to 11 summarize the generation quality on the YouTube dataset.

Table 9: Generation quality in degree centralities distribution similarity on YouTube dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | **0.0015 ± 0.0007** | **0.0376 ± 0.0143** | **0.0001 ± 0.0001** | **0.0213 ± 0.0180** |
| | 3 | 0.0023 ± 0.0009 | 0.0561 ± 0.0167 | 0.0001 ± 0.0001 | 0.0446 ± 0.0202 |
| | 4 | 0.0032 ± 0.0010 | 0.0742 ± 0.0191 | 0.0003 ± 0.0002 | 0.0671 ± 0.0215 |
| | 5 | 0.0049 ± 0.0009 | 0.1179 ± 0.0178 | 0.0007 ± 0.0002 | 0.1149 ± 0.0190 |
| | 6 | 0.0058 ± 0.0009 | 0.1392 ± 0.0187 | 0.0010 ± 0.0003 | 0.1380 ± 0.0198 |
| SyNG-R | 2 | 0.0013 ± 0.0006 | **0.0286 ± 0.0113** | **0.0000 ± 0.0000** | **0.0110 ± 0.0143** |
| | 3 | 0.0013 ± 0.0006 | 0.0291 ± 0.0116 | 0.0000 ± 0.0000 | 0.0121 ± 0.0147 |
| | 4 | 0.0013 ± 0.0006 | 0.0291 ± 0.0111 | 0.0000 ± 0.0000 | 0.0118 ± 0.0150 |
| | 5 | 0.0013 ± 0.0006 | 0.0298 ± 0.0116 | 0.0000 ± 0.0000 | 0.0115 ± 0.0147 |
| | 6 | 0.0013 ± 0.0006 | 0.0300 ± 0.0121 | 0.0000 ± 0.0000 | 0.0116 ± 0.0152 |
| VGAE | 2 | 0.0089 ± 0.0001 | 0.2374 ± 0.0044 | 0.0020 ± 0.0000 | 0.3385 ± 0.0046 |
| | 3 | 0.0089 ± 0.0001 | 0.2349 ± 0.0040 | 0.0020 ± 0.0000 | 0.3339 ± 0.0055 |
| | 4 | **0.0080 ± 0.0001** | **0.2010 ± 0.0043** | **0.0015 ± 0.0000** | **0.2783 ± 0.0041** |
| | 5 | 0.0091 ± 0.0001 | 0.2440 ± 0.0039 | 0.0021 ± 0.0000 | 0.3496 ± 0.0036 |
| | 6 | 0.0082 ± 0.0001 | 0.2033 ± 0.0043 | 0.0016 ± 0.0000 | 0.2878 ± 0.0036 |
| | 16 | 0.0085 ± 0.0001 | 0.2176 ± 0.0043 | 0.0018 ± 0.0000 | 0.3103 ± 0.0040 |
| GRAN | 128 | 0.0078 ± 0.0038 | 0.1538 ± 0.0547 | 0.0012 ± 0.0017 | 0.1602 ± 0.0424 |
| | 256 | 0.0571 ± 0.0041 | 0.5759 ± 0.0300 | 0.0387 ± 0.0048 | 0.6202 ± 0.0320 |
| | 512 | **0.0076 ± 0.0020** | **0.0924 ± 0.0200** | **0.0008 ± 0.0004** | **0.0786 ± 0.0172** |
| EDGE | – | **0.0041 ± 0.0005** | **0.0782 ± 0.0086** | **0.0004 ± 0.0001** | **0.0784 ± 0.0095** |
| E-R | – | **0.0141 ± 0.0001** | **0.4708 ± 0.0044** | **0.0067 ± 0.0001** | **0.6922 ± 0.0040** |

**DBLP dataset.** The following Tables 12 to 14 present the full experimental results for the DBLP dataset.

**Yelp dataset.** Here we provide the detailed evaluation for the Yelp dataset. The results for degree centrality, eigenvalue distribution, and other numerical characteristics are shown in the Tables 15 to 17 respectively.

**PolBlogs dataset.** Finally, we present the comprehensive results for the PolBlogs dataset. The subsequent Tables 18 to 20 detail the performance of each method in capturing the structural and numerical properties of the original network.

Table 10: Generation quality in eigenvalue distribution similarity on YouTube dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | 0.3111 ± 0.0552 | 0.0303 ± 0.0067 | 0.0109 ± 0.0051 | 0.0261 ± 0.0094 |
| | 3 | 0.2296 ± 0.0400 | 0.0198 ± 0.0056 | 0.0049 ± 0.0026 | 0.0084 ± 0.0097 |
| | 4 | **0.1901 ± 0.0398** | **0.0142 ± 0.0038** | **0.0028 ± 0.0015** | **0.0007 ± 0.0030** |
| | 5 | 0.3003 ± 0.0759 | 0.0199 ± 0.0056 | 0.0075 ± 0.0042 | 0.0019 ± 0.0051 |
| | 6 | 0.4025 ± 0.0820 | 0.0278 ± 0.0068 | 0.0142 ± 0.0068 | 0.0114 ± 0.0115 |
| SyNG-R | 2 | 0.4337 ± 0.0683 | 0.0440 ± 0.0070 | 0.0241 ± 0.0086 | 0.0447 ± 0.0089 |
| | 3 | 0.3756 ± 0.0627 | 0.0394 ± 0.0068 | 0.0189 ± 0.0071 | 0.0392 ± 0.0095 |
| | 4 | 0.3299 ± 0.0586 | 0.0359 ± 0.0064 | 0.0150 ± 0.0059 | 0.0340 ± 0.0091 |
| | 5 | 0.3081 ± 0.0559 | 0.0355 ± 0.0061 | 0.0135 ± 0.0055 | 0.0322 ± 0.0088 |
| | 6 | **0.2935 ± 0.0539** | **0.0354 ± 0.0058** | **0.0126 ± 0.0051** | **0.0311 ± 0.0088** |
| VGAE | 2 | 1.2692 ± 0.0092 | 0.1514 ± 0.0010 | 0.2348 ± 0.0039 | 0.1848 ± 0.0012 |
| | 3 | 1.2646 ± 0.0087 | 0.1510 ± 0.0010 | 0.2332 ± 0.0037 | 0.1843 ± 0.0012 |
| | 4 | **1.1502 ± 0.0085** | **0.1406 ± 0.0010** | **0.1941 ± 0.0033** | **0.1720 ± 0.0013** |
| | 5 | 1.2868 ± 0.0096 | 0.1532 ± 0.0011 | 0.2415 ± 0.0041 | 0.1875 ± 0.0012 |
| | 6 | 1.1740 ± 0.0085 | 0.1422 ± 0.0011 | 0.2016 ± 0.0034 | 0.1743 ± 0.0013 |
| | 16 | 1.2084 ± 0.0092 | 0.1456 ± 0.0011 | 0.2136 ± 0.0038 | 0.1791 ± 0.0012 |
| GRAN | 128 | 0.9723 ± 0.4615 | **0.0697 ± 0.0289** | 0.1140 ± 0.1250 | **0.0708 ± 0.0324** |
| | 256 | 3.9987 ± 0.2321 | 0.2398 ± 0.0119 | 1.4200 ± 0.1533 | 0.2576 ± 0.0139 |
| | 512 | **0.9434 ± 0.1651** | 0.0759 ± 0.0098 | **0.0987 ± 0.0328** | 0.0788 ± 0.0085 |
| EDGE | – | **0.3357 ± 0.0388** | **0.0389 ± 0.0085** | **0.0136 ± 0.0049** | **0.0444 ± 0.0072** |
| E-R | – | **1.9528 ± 0.0101** | **0.2099 ± 0.0010** | **0.5537 ± 0.0059** | **0.2596 ± 0.0009** |

Table 11: Generation quality in numerical characteristics on YouTube dataset.

| Method | Config | Global clustering coefficient | | | Triangle density | | |
|---|---|---|---|---|---|---|---|
| | | RMSE | MAE | Bias | RMSE | MAE | Bias |
| SyNG-D | 2 | 0.022788 | 0.021504 | -0.021504 | 0.000097 | 0.000092 | 0.000092 |
| | 3 | 0.016919 | 0.014596 | -0.014099 | 0.000089 | 0.000084 | 0.000084 |
| | 4 | 0.013366 | 0.010770 | -0.009493 | 0.000083 | 0.000078 | 0.000078 |
| | 5 | 0.014951 | 0.012186 | -0.010930 | **0.000058** | **0.000052** | **0.000052** |
| | 6 | **0.014220** | **0.011899** | **-0.010130** | 0.000041 | 0.000035 | 0.000035 |
| SyNG-R | 2 | 0.022261 | 0.020758 | -0.020731 | **0.000115** | **0.000111** | **0.000111** |
| | 3 | **0.010665** | **0.008717** | **-0.004048** | 0.000138 | 0.000134 | 0.000134 |
| | 4 | 0.011996 | 0.009619 | 0.006626 | 0.000153 | 0.000149 | 0.000149 |
| | 5 | 0.015688 | 0.012999 | 0.011945 | 0.000161 | 0.000157 | 0.000157 |
| | 6 | 0.018381 | 0.015688 | 0.015182 | 0.000165 | 0.000161 | 0.000161 |
| VGAE | 2 | 0.117986 | 0.117984 | -0.117984 | 0.000081 | 0.000081 | -0.000081 |
| | 3 | 0.116633 | 0.116631 | -0.116631 | 0.000080 | 0.000080 | -0.000080 |
| | 4 | **0.080046** | **0.080040** | **-0.080040** | **0.000056** | **0.000056** | **-0.000056** |
| | 5 | 0.120663 | 0.120662 | -0.120662 | 0.000083 | 0.000083 | -0.000083 |
| | 6 | 0.092410 | 0.092407 | -0.092407 | 0.000065 | 0.000065 | -0.000065 |
| | 16 | 0.098745 | 0.098741 | -0.098741 | 0.000069 | 0.000069 | -0.000069 |
| GRAN | 128 | 0.113408 | 0.113203 | -0.113203 | **0.000085** | **0.000056** | **0.000054** |
| | 256 | **0.053022** | **0.052876** | **-0.052876** | 0.001505 | 0.001495 | 0.001495 |
| | 512 | 0.101078 | 0.101039 | -0.101039 | 0.000113 | 0.000108 | 0.000108 |
| EDGE | – | **0.144481** | **0.138077** | **-0.137958** | **0.000078** | **0.000075** | **-0.000050** |
| E-R | – | **0.163006** | **0.163006** | **-0.163006** | **0.000105** | **0.000105** | **-0.000105** |

Table 12: Generation quality in degree centralities distribution similarity on DBLP dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | 0.0019 ± 0.0008 | **0.0751 ± 0.0170** | **0.0002 ± 0.0001** | **0.0926 ± 0.0226** |
| | 3 | **0.0018 ± 0.0006** | 0.0913 ± 0.0162 | **0.0002 ± 0.0001** | 0.1034 ± 0.0191 |
| | 4 | 0.0022 ± 0.0007 | 0.1223 ± 0.0140 | 0.0003 ± 0.0001 | 0.1286 ± 0.0172 |
| | 5 | 0.0033 ± 0.0009 | 0.1735 ± 0.0178 | 0.0005 ± 0.0002 | 0.1782 ± 0.0197 |
| | 6 | 0.0044 ± 0.0010 | 0.2191 ± 0.0183 | 0.0009 ± 0.0002 | 0.2224 ± 0.0196 |
| SyNG-R | 2 | **0.0015 ± 0.0007** | **0.0725 ± 0.0181** | **0.0001 ± 0.0001** | **0.0748 ± 0.0281** |
| | 3 | **0.0015 ± 0.0007** | 0.0729 ± 0.0178 | **0.0001 ± 0.0001** | 0.0754 ± 0.0278 |
| | 4 | **0.0015 ± 0.0007** | 0.0731 ± 0.0182 | **0.0001 ± 0.0001** | 0.0768 ± 0.0281 |
| | 5 | 0.0016 ± 0.0007 | 0.0737 ± 0.0177 | **0.0001 ± 0.0001** | 0.0781 ± 0.0273 |
| | 6 | 0.0016 ± 0.0007 | 0.0746 ± 0.0174 | **0.0001 ± 0.0001** | 0.0798 ± 0.0267 |
| VGAE | 2 | 0.0049 ± 0.0001 | **0.2328 ± 0.0084** | 0.0010 ± 0.0000 | 0.2840 ± 0.0096 |
| | 3 | **0.0029 ± 0.0000** | 0.2653 ± 0.0066 | **0.0009 ± 0.0000** | **0.2797 ± 0.0059** |
| | 4 | 0.0035 ± 0.0001 | 0.3202 ± 0.0081 | 0.0012 ± 0.0000 | 0.3275 ± 0.0059 |
| | 5 | 0.0035 ± 0.0001 | 0.3164 ± 0.0083 | 0.0012 ± 0.0000 | 0.3231 ± 0.0067 |
| | 6 | 0.0032 ± 0.0000 | 0.2959 ± 0.0071 | 0.0011 ± 0.0000 | 0.3015 ± 0.0053 |
| | 16 | 0.0032 ± 0.0000 | 0.2960 ± 0.0078 | 0.0010 ± 0.0000 | 0.2957 ± 0.0060 |
| GRAN | 128 | 0.0148 ± 0.0025 | 0.3785 ± 0.0882 | 0.0041 ± 0.0014 | 0.5133 ± 0.0962 |
| | 256 | 0.0125 ± 0.0003 | 0.4609 ± 0.0119 | 0.0054 ± 0.0002 | 0.6118 ± 0.0141 |
| | 512 | **0.0106 ± 0.0001** | **0.2207 ± 0.0080** | **0.0031 ± 0.0001** | **0.2973 ± 0.0086** |
| EDGE | – | **0.0023 ± 0.0012** | **0.0790 ± 0.0106** | **0.0002 ± 0.0002** | **0.0992 ± 0.0232** |
| E-R | – | **0.0157 ± 0.0001** | **0.6626 ± 0.0069** | **0.0102 ± 0.0002** | **0.8716 ± 0.0061** |

Table 13: Generation quality in eigenvalue distribution similarity on DBLP dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | 0.3021 ± 0.0319 | **0.0807 ± 0.0063** | 0.0235 ± 0.0043 | 0.0879 ± 0.0059 |
| | 3 | 0.2127 ± 0.0276 | 0.0914 ± 0.0069 | **0.0157 ± 0.0021** | 0.0769 ± 0.0049 |
| | 4 | **0.1730 ± 0.0167** | 0.1056 ± 0.0072 | 0.0165 ± 0.0020 | **0.0767 ± 0.0052** |
| | 5 | 0.2559 ± 0.0431 | 0.1320 ± 0.0084 | 0.0299 ± 0.0057 | 0.0964 ± 0.0075 |
| | 6 | 0.3803 ± 0.0456 | 0.1628 ± 0.0085 | 0.0562 ± 0.0089 | 0.1281 ± 0.0083 |
| SyNG-B | 2 | 0.2893 ± 0.0333 | 0.0626 ± 0.0049 | 0.0202 ± 0.0050 | 0.0799 ± 0.0067 |
| | 3 | 0.2254 ± 0.0327 | 0.0575 ± 0.0054 | 0.0135 ± 0.0039 | 0.0651 ± 0.0071 |
| | 4 | 0.1698 ± 0.0287 | 0.0500 ± 0.0056 | 0.0083 ± 0.0025 | 0.0492 ± 0.0066 |
| | 5 | 0.1312 ± 0.0299 | 0.0459 ± 0.0050 | 0.0052 ± 0.0018 | 0.0366 ± 0.0068 |
| | 6 | **0.1187 ± 0.0315** | **0.0406 ± 0.0050** | **0.0040 ± 0.0015** | **0.0273 ± 0.0071** |
| VGAE | 2 | 1.2381 ± 0.0156 | 0.2351 ± 0.0028 | 0.3767 ± 0.0101 | 0.2622 ± 0.0033 |
| | 3 | 0.4036 ± 0.0056 | 0.0979 ± 0.0035 | 0.0347 ± 0.0017 | 0.1196 ± 0.0039 |
| | 4 | 0.3828 ± 0.0096 | 0.0673 ± 0.0027 | 0.0281 ± 0.0018 | 0.0683 ± 0.0028 |
| | 5 | 0.3756 ± 0.0116 | 0.0654 ± 0.0029 | 0.0269 ± 0.0021 | 0.0653 ± 0.0026 |
| | 6 | **0.3512 ± 0.0077** | **0.0610 ± 0.0021** | **0.0230 ± 0.0013** | **0.0643 ± 0.0032** |
| | 16 | 0.3575 ± 0.0093 | 0.0628 ± 0.0026 | 0.0248 ± 0.0016 | 0.0654 ± 0.0032 |
| GRAN | 128 | **0.6396 ± 0.1610** | 0.1608 ± 0.0204 | 0.1492 ± 0.0620 | 0.1874 ± 0.0223 |
| | 256 | 1.7553 ± 0.0482 | 0.3056 ± 0.0053 | 0.7524 ± 0.0329 | 0.3538 ± 0.0062 |
| | 512 | 0.6951 ± 0.0236 | **0.1491 ± 0.0055** | **0.1181 ± 0.0100** | **0.1757 ± 0.0066** |
| EDGE | – | **0.5500 ± 0.1337** | **0.1223 ± 0.0249** | **0.0972 ± 0.0423** | **0.1387 ± 0.0296** |
| E-R | – | **2.2549 ± 0.0136** | **0.3768 ± 0.0016** | **1.2143 ± 0.0126** | **0.4409 ± 0.0016** |

Table 14: Generation quality in numerical characteristics on DBLP dataset.

| Method | Config | Global clustering coefficient | | | Triangle density | | |
|---|---|---|---|---|---|---|---|
| | | RMSE | MAE | Bias | RMSE | MAE | Bias |
| SyNG-D | 2 | 0.062213 | 0.058698 | -0.058698 | **0.000158** | **0.000126** | **0.000046** |
| | 3 | 0.053395 | **0.049389** | **-0.049389** | 0.000161 | 0.000137 | -0.000087 |
| | 4 | 0.055118 | 0.052253 | -0.052253 | 0.000181 | 0.000157 | -0.000139 |
| | 5 | **0.052696** | 0.049473 | -0.049473 | 0.000234 | 0.000213 | -0.000207 |
| | 6 | 0.057088 | 0.053737 | -0.053737 | 0.000301 | 0.000286 | -0.000286 |
| SyNG-B | 2 | 0.037820 | 0.032761 | -0.032515 | **0.000143** | **0.000112** | **0.000011** |
| | 3 | 0.032168 | 0.026907 | -0.026373 | 0.000144 | 0.000112 | 0.000019 |
| | 4 | 0.026335 | 0.020965 | -0.019649 | 0.000145 | 0.000112 | 0.000025 |
| | 5 | 0.022729 | 0.017575 | -0.015066 | 0.000146 | 0.000113 | 0.000028 |
| | 6 | **0.021397** | **0.016556** | **-0.013237** | 0.000146 | 0.000113 | 0.000031 |
| VGAE | 2 | 0.632837 | 0.632827 | -0.632827 | 0.000665 | 0.000665 | -0.000665 |
| | 3 | **0.013590** | **0.013382** | **-0.013382** | 0.000058 | 0.000058 | -0.000058 |
| | 4 | 0.029395 | 0.029349 | 0.029349 | 0.000028 | 0.000028 | 0.000028 |
| | 5 | 0.030082 | 0.030021 | 0.030021 | 0.000026 | 0.000026 | 0.000026 |
| | 6 | 0.033185 | 0.033145 | 0.033145 | 0.000011 | 0.000010 | 0.000010 |
| | 16 | 0.031045 | 0.031000 | 0.031000 | **0.000005** | **0.000005** | **0.000005** |
| GRAN | 128 | **0.876328** | **0.876311** | **-0.876311** | **0.000678** | **0.000673** | **-0.000673** |
| | 256 | 0.889293 | 0.889279 | -0.889279 | 0.000793 | 0.000793 | -0.000793 |
| | 512 | 0.892119 | 0.892116 | -0.892116 | 0.000799 | 0.000799 | -0.000799 |
| EDGE | – | **0.132680** | **0.108559** | **-0.108559** | **0.000220** | **0.000177** | **-0.000177** |
| E-R | – | **0.894349** | **0.894349** | **-0.894349** | **0.000796** | **0.000796** | **-0.000796** |

Table 15: Generation quality in degree centralities distribution similarity on Yelp dataset.

| Method | Config | W1 | KS | Energy | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | **0.0015 ± 0.0005** | **0.0249 ± 0.0074** | **0.0054 ± 0.0019** | **0.0128 ± 0.0087** |
| | 3 | 0.0021 ± 0.0010 | 0.0355 ± 0.0126 | 0.0077 ± 0.0034 | 0.0236 ± 0.0143 |
| | 4 | 0.0024 ± 0.0010 | 0.0454 ± 0.0112 | 0.0098 ± 0.0033 | 0.0346 ± 0.0109 |
| | 5 | 0.0036 ± 0.0011 | 0.0602 ± 0.0116 | 0.0142 ± 0.0037 | 0.0503 ± 0.0115 |
| | 6 | 0.0044 ± 0.0013 | 0.0772 ± 0.0142 | 0.0181 ± 0.0044 | 0.0672 ± 0.0136 |
| SyNG-B | 2 | **0.0013 ± 0.0006** | 0.0202 ± 0.0080 | **0.0047 ± 0.0022** | 0.0065 ± 0.0094 |
| | 3 | **0.0013 ± 0.0006** | **0.0198 ± 0.0078** | **0.0047 ± 0.0022** | **0.0060 ± 0.0090** |
| | 4 | **0.0013 ± 0.0006** | 0.0199 ± 0.0082 | **0.0047 ± 0.0023** | 0.0062 ± 0.0092 |
| | 5 | 0.0014 ± 0.0006 | 0.0205 ± 0.0080 | 0.0048 ± 0.0022 | 0.0067 ± 0.0094 |
| | 6 | **0.0013 ± 0.0006** | **0.0198 ± 0.0078** | **0.0047 ± 0.0022** | 0.0062 ± 0.0091 |
| VGAE | 2 | **0.0165 ± 0.0000** | **0.2212 ± 0.0023** | **0.0579 ± 0.0002** | **0.3223 ± 0.0016** |
| | 3 | 0.0170 ± 0.0000 | 0.2314 ± 0.0022 | 0.0598 ± 0.0002 | 0.3332 ± 0.0019 |
| | 4 | 0.0171 ± 0.0000 | 0.2350 ± 0.0021 | 0.0603 ± 0.0002 | 0.3367 ± 0.0017 |
| | 5 | 0.0169 ± 0.0000 | 0.2281 ± 0.0022 | 0.0597 ± 0.0002 | 0.3338 ± 0.0016 |
| | 6 | 0.0179 ± 0.0000 | 0.2447 ± 0.0020 | 0.0634 ± 0.0002 | 0.3549 ± 0.0014 |
| | 16 | 0.0173 ± 0.0000 | 0.2337 ± 0.0021 | 0.0609 ± 0.0002 | 0.3374 ± 0.0018 |
| E-R | – | **0.0281 ± 0.0000** | **0.5717 ± 0.0018** | **0.1209 ± 0.0003** | **0.7736 ± 0.0013** |

Table 16: Generation quality in eigenvalue distribution similarity on Yelp dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | 1.2877 ± 0.0902 | 0.0529 ± 0.0039 | 0.2932 ± 0.0239 | 0.0611 ± 0.0045 |
| | 3 | 1.0882 ± 0.1008 | 0.0443 ± 0.0046 | 0.2428 ± 0.0273 | 0.0508 ± 0.0053 |
| | 4 | 0.9242 ± 0.0833 | 0.0376 ± 0.0039 | 0.2034 ± 0.0228 | 0.0426 ± 0.0046 |
| | 5 | 0.7666 ± 0.0816 | 0.0302 ± 0.0040 | 0.1620 ± 0.0225 | 0.0336 ± 0.0050 |
| | 6 | **0.6392 ± 0.0923** | **0.0234 ± 0.0046** | **0.1279 ± 0.0251** | **0.0250 ± 0.0062** |
| SyNG-B | 2 | 1.3240 ± 0.0944 | 0.0551 ± 0.0041 | 0.3035 ± 0.0251 | 0.0636 ± 0.0047 |
| | 3 | 1.2229 ± 0.0933 | 0.0514 ± 0.0041 | 0.2816 ± 0.0249 | 0.0590 ± 0.0048 |
| | 4 | 1.0975 ± 0.0903 | 0.0474 ± 0.0040 | 0.2553 ± 0.0241 | 0.0539 ± 0.0047 |
| | 5 | 1.0018 ± 0.0894 | 0.0437 ± 0.0038 | 0.2333 ± 0.0237 | 0.0493 ± 0.0047 |
| | 6 | **0.9405 ± 0.0875** | **0.0421 ± 0.0037** | **0.2209 ± 0.0232** | **0.0469 ± 0.0046** |
| VGAE | 2 | 2.4428 ± 0.0057 | 0.1401 ± 0.0004 | 0.6508 ± 0.0018 | **0.1728 ± 0.0004** |
| | 3 | 2.4566 ± 0.0053 | 0.1410 ± 0.0004 | 0.6557 ± 0.0017 | 0.1755 ± 0.0004 |
| | 4 | 2.4662 ± 0.0057 | 0.1415 ± 0.0004 | 0.6582 ± 0.0018 | 0.1763 ± 0.0004 |
| | 5 | 2.4762 ± 0.0057 | 0.1416 ± 0.0004 | 0.6596 ± 0.0017 | 0.1749 ± 0.0004 |
| | 6 | 2.5127 ± 0.0054 | 0.1436 ± 0.0004 | 0.6707 ± 0.0017 | 0.1792 ± 0.0003 |
| | 16 | **2.4206 ± 0.0057** | **0.1400 ± 0.0004** | **0.6487 ± 0.0018** | 0.1758 ± 0.0004 |
| E-R | – | **3.8301 ± 0.0067** | **0.2001 ± 0.0003** | **1.0215 ± 0.0018** | **0.2453 ± 0.0003** |

Table 17: Generation quality in numerical characteristics on Yelp dataset.

| Method | Config | Global clustering coefficient | | | Triangle density | | |
|---|---|---|---|---|---|---|---|
| | | RMSE | MAE | Bias | RMSE | MAE | Bias |
| SyNG-D | 2 | 0.025551 | 0.025188 | -0.025188 | **0.000124** | **0.000108** | **-0.000105** |
| | 3 | 0.026478 | 0.026080 | -0.026080 | 0.000171 | 0.000157 | -0.000156 |
| | 4 | 0.020121 | 0.019630 | -0.019630 | 0.000159 | 0.000147 | -0.000146 |
| | 5 | 0.018293 | 0.017720 | -0.017720 | 0.000182 | 0.000172 | -0.000171 |
| | 6 | **0.017662** | **0.017043** | **-0.017043** | 0.000200 | 0.000189 | -0.000188 |
| SyNG-B | 2 | 0.027935 | 0.027581 | -0.027581 | 0.000149 | 0.000135 | -0.000134 |
| | 3 | 0.023972 | 0.023582 | -0.023582 | 0.000133 | 0.000117 | -0.000115 |
| | 4 | 0.015615 | 0.014984 | -0.014984 | 0.000101 | 0.000085 | -0.000073 |
| | 5 | 0.009946 | 0.008985 | -0.008902 | 0.000083 | 0.000069 | -0.000043 |
| | 6 | **0.007562** | **0.006486** | **-0.006140** | **0.000078** | **0.000064** | **-0.000030** |
| VGAE | 2 | 0.103578 | 0.103578 | -0.103578 | **0.000736** | **0.000736** | **-0.000736** |
| | 3 | 0.105472 | 0.105471 | -0.105471 | 0.000745 | 0.000745 | -0.000745 |
| | 4 | 0.105552 | 0.105552 | -0.105552 | 0.000746 | 0.000746 | -0.000746 |
| | 5 | 0.107737 | 0.107737 | -0.107737 | 0.000749 | 0.000749 | -0.000749 |
| | 6 | 0.112563 | 0.112563 | -0.112563 | 0.000767 | 0.000767 | -0.000767 |
| | 16 | **0.102531** | **0.102531** | **-0.102531** | 0.000741 | 0.000741 | -0.000741 |
| E-R | – | **0.144825** | **0.144825** | **-0.144825** | **0.000812** | **0.000812** | **-0.000812** |

Table 18: Generation quality in degree centralities distribution similarity on PolBlogs dataset.

| Method | Config | W1 | KS | Energy | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | **0.0018 ± 0.0008** | **0.0453 ± 0.0097** | **0.0001 ± 0.0001** | **0.0097 ± 0.0144** |
| | 3 | 0.0022 ± 0.0011 | 0.0502 ± 0.0134 | **0.0001 ± 0.0001** | 0.0195 ± 0.0215 |
| | 4 | 0.0028 ± 0.0013 | 0.0574 ± 0.0164 | 0.0002 ± 0.0002 | 0.0318 ± 0.0250 |
| | 5 | 0.0044 ± 0.0014 | 0.0829 ± 0.0204 | 0.0004 ± 0.0002 | 0.0649 ± 0.0244 |
| | 6 | 0.0052 ± 0.0015 | 0.0998 ± 0.0211 | 0.0006 ± 0.0003 | 0.0831 ± 0.0250 |
| SyNG-B | 2 | **0.0018 ± 0.0010** | **0.0447 ± 0.0092** | **0.0001 ± 0.0001** | **0.0092 ± 0.0145** |
| | 3 | **0.0018 ± 0.0010** | 0.0449 ± 0.0111 | **0.0001 ± 0.0001** | 0.0093 ± 0.0150 |
| | 4 | **0.0018 ± 0.0010** | 0.0457 ± 0.0123 | **0.0001 ± 0.0001** | 0.0094 ± 0.0150 |
| | 5 | **0.0018 ± 0.0010** | 0.0468 ± 0.0130 | **0.0001 ± 0.0001** | 0.0106 ± 0.0162 |
| | 6 | **0.0018 ± 0.0010** | 0.0488 ± 0.0149 | **0.0001 ± 0.0001** | 0.0112 ± 0.0167 |
| VGAE | 2 | 0.0093 ± 0.0001 | 0.3538 ± 0.0049 | 0.0024 ± 0.0001 | 0.3542 ± 0.0061 |
| | 3 | 0.0097 ± 0.0001 | 0.3624 ± 0.0053 | 0.0026 ± 0.0001 | 0.3716 ± 0.0051 |
| | 4 | 0.0094 ± 0.0001 | 0.3565 ± 0.0052 | 0.0024 ± 0.0001 | 0.3602 ± 0.0060 |
| | 5 | **0.0092 ± 0.0001** | **0.3531 ± 0.0051** | **0.0023 ± 0.0001** | **0.3533 ± 0.0064** |
| | 6 | 0.0097 ± 0.0001 | 0.3635 ± 0.0053 | 0.0026 ± 0.0001 | 0.3742 ± 0.0057 |
| | 16 | 0.0098 ± 0.0001 | 0.3656 ± 0.0049 | 0.0026 ± 0.0001 | 0.3782 ± 0.0056 |
| GRAN | 128 | **0.0056 ± 0.0019** | **0.1588 ± 0.0270** | **0.0008 ± 0.0005** | **0.1386 ± 0.0201** |
| | 256 | 0.0225 ± 0.0031 | 0.3558 ± 0.0315 | 0.0064 ± 0.0015 | 0.3378 ± 0.0361 |
| | 512 | 0.0930 ± 0.0067 | 0.6337 ± 0.0238 | 0.0636 ± 0.0072 | 0.6741 ± 0.0269 |
| EDGE | – | **0.0006 ± 0.0000** | **0.0491 ± 0.0044** | **0.0000 ± 0.0000** | **0.0000 ± 0.0000** |
| E-R | – | **0.0183 ± 0.0001** | **0.5670 ± 0.0056** | **0.0104 ± 0.0001** | **0.7924 ± 0.0059** |

Table 19: Generation quality in eigenvalue distribution similarity on PolBlogs dataset.

| Method | Config | W1 dist. | KS dist. | Energy dist. | MMD |
|---|---|---|---|---|---|
| SyNG-D | 2 | 0.2060 ± 0.0574 | 0.0482 ± 0.0094 | 0.0092 ± 0.0051 | 0.0258 ± 0.0137 |
| | 3 | 0.2022 ± 0.0401 | 0.0502 ± 0.0103 | 0.0090 ± 0.0038 | 0.0315 ± 0.0116 |
| | 4 | **0.2013 ± 0.0488** | 0.0441 ± 0.0096 | **0.0074 ± 0.0032** | 0.0252 ± 0.0122 |
| | 5 | 0.2482 ± 0.0683 | **0.0322 ± 0.0077** | 0.0080 ± 0.0046 | 0.0151 ± 0.0118 |
| | 6 | 0.2973 ± 0.0857 | 0.0332 ± 0.0089 | 0.0119 ± 0.0072 | **0.0143 ± 0.0136** |
| SyNG-B | 2 | 0.2400 ± 0.0787 | **0.0504 ± 0.0110** | **0.0124 ± 0.0082** | **0.0301 ± 0.0159** |
| | 3 | 0.2336 ± 0.0705 | 0.0545 ± 0.0102 | 0.0133 ± 0.0076 | 0.0364 ± 0.0134 |
| | 4 | 0.2272 ± 0.0674 | 0.0558 ± 0.0106 | 0.0135 ± 0.0079 | 0.0386 ± 0.0132 |
| | 5 | 0.2192 ± 0.0663 | 0.0568 ± 0.0099 | 0.0134 ± 0.0077 | 0.0398 ± 0.0120 |
| | 6 | **0.2182 ± 0.0619** | 0.0569 ± 0.0098 | 0.0136 ± 0.0073 | 0.0415 ± 0.0117 |
| VGAE | 2 | 1.2339 ± 0.0104 | 0.2733 ± 0.0018 | 0.4220 ± 0.0072 | 0.3111 ± 0.0020 |
| | 3 | 1.2677 ± 0.0121 | 0.2767 ± 0.0018 | 0.4405 ± 0.0082 | 0.3157 ± 0.0021 |
| | 4 | 1.2410 ± 0.0112 | 0.2741 ± 0.0018 | 0.4265 ± 0.0074 | 0.3123 ± 0.0020 |
| | 5 | **1.2263 ± 0.0117** | **0.2727 ± 0.0018** | **0.4188 ± 0.0075** | **0.3110 ± 0.0022** |
| | 6 | 1.2671 ± 0.0108 | 0.2772 ± 0.0018 | 0.4416 ± 0.0074 | 0.3162 ± 0.0019 |
| | 16 | 1.2787 ± 0.0114 | 0.2781 ± 0.0018 | 0.4470 ± 0.0077 | 0.3180 ± 0.0020 |
| GRAN | 128 | **0.4770 ± 0.1785** | 0.1091 ± 0.0115 | **0.0463 ± 0.0288** | **0.0904 ± 0.0098** |
| | 256 | 1.2251 ± 0.1586 | **0.1089 ± 0.0118** | 0.1703 ± 0.0431 | 0.0939 ± 0.0126 |
| | 512 | 3.5235 ± 0.1906 | 0.2455 ± 0.0124 | 1.2293 ± 0.1278 | 0.2684 ± 0.0141 |
| EDGE | – | **0.2793 ± 0.0415** | **0.0821 ± 0.0117** | **0.0190 ± 0.0059** | **0.0469 ± 0.0126** |
| E-R | – | **2.0281 ± 0.0109** | **0.3442 ± 0.0012** | **0.9320 ± 0.0092** | **0.4010 ± 0.0011** |

Table 20: Generation quality in numerical characteristics on PolBlogs dataset.

| Method | Config | Global clustering coefficient | | | Triangle density | | |
|---|---|---|---|---|---|---|---|
| | | RMSE | MAE | Bias | RMSE | MAE | Bias |
| SyNG-D | 2 | 0.018976 | 0.015124 | 0.011476 | 0.000071 | **0.000055** | **0.000013** |
| | 3 | **0.015564** | **0.012312** | 0.005437 | **0.000068** | 0.000056 | -0.000033 |
| | 4 | 0.018539 | 0.014519 | 0.009024 | 0.000078 | 0.000066 | -0.000043 |
| | 5 | 0.016557 | 0.013288 | **0.004908** | 0.000109 | 0.000100 | -0.000098 |
| | 6 | 0.019115 | 0.014940 | 0.009254 | 0.000119 | 0.000109 | -0.000107 |
| SyNG-B | 2 | **0.024512** | **0.020038** | **0.018285** | **0.000083** | **0.000062** | **0.000039** |
| | 3 | 0.026816 | 0.022261 | 0.021210 | 0.000085 | 0.000064 | 0.000044 |
| | 4 | 0.028811 | 0.024436 | 0.023662 | 0.000087 | 0.000066 | 0.000049 |
| | 5 | 0.029078 | 0.024992 | 0.024354 | 0.000089 | 0.000068 | 0.000051 |
| | 6 | 0.031319 | 0.027414 | 0.026902 | 0.000092 | 0.000071 | 0.000055 |
| VGAE | 2 | 0.037207 | 0.037138 | -0.037138 | 0.000210 | 0.000210 | **-0.000207** |
| | 3 | 0.044932 | 0.044882 | -0.044882 | 0.000220 | 0.000220 | -0.000220 |
| | 4 | 0.040265 | 0.040203 | -0.040203 | 0.000214 | 0.000214 | -0.000214 |
| | 5 | **0.032565** | **0.032483** | **-0.032483** | **0.000207** | **0.000207** | **-0.000207** |
| | 6 | 0.048727 | 0.048684 | -0.048684 | 0.000222 | 0.000222 | -0.000222 |
| | 16 | 0.045418 | 0.045365 | -0.045365 | 0.000222 | 0.000222 | -0.000222 |
| GRAN | 128 | 0.114164 | 0.113753 | -0.113753 | **0.000157** | **0.000149** | **-0.000141** |
| | 256 | 0.108178 | 0.107799 | -0.107799 | 0.000442 | 0.000422 | 0.000422 |
| | 512 | **0.007568** | **0.006001** | **0.000342** | 0.006494 | 0.006445 | 0.006445 |
| EDGE | – | **0.056941** | **0.054337** | **-0.054337** | **0.000079** | **0.000075** | **-0.000075** |
| E-R | – | **0.203573** | **0.203573** | **-0.203573** | **0.000322** | **0.000322** | **-0.000322** |

### B.7 Supplementary for the Efficiency Comparison

In this section, we begin with a note on e-FLOPs and then present a comparison of training and sampling time across methods, specifying the device environment used for each.

**A note on e-FLOPs.** The definition of e-FLOPs considers two types of operations: the float-point operations on neural nets, and node visit operations on trees. These two operations are not directly comparable, as node visits involve comparison and branching which are different operations than the float-point operations on neural nets. In Section 4.2, we use e-FLOPs for a comparison which is less dependent on the implemented device environments. As a supplementary, we study the wall-block training and sampling time of each methods with their implemented device environments specified.

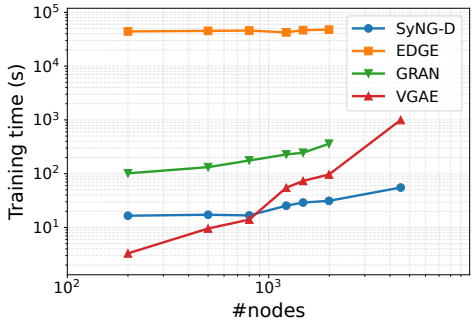



Figure 5: Wall-clock training time of different methods for datasets of different sizes.

**Evaluation metrics and configuration.** We compare training and sampling efficiency between SyNG-D and the baseline methods through the time they spend during training and sampling. SyNG-D and VGAE are trained on CPUs, while GRAN and EDGE are trained on a single NVIDIA GeForce RTX 4090 with memory of 24GB. For each dataset, we train each model using the default training schedule, sample 128 networks from each model, and record the wall-clock training time and average sampling time.

**Results and discussion.** Figure 5 presents the training time comparisons between our method and the baselines. In particular, our model can be trained in tens of seconds even for graphs with up to 5,000 nodes, whereas deep learning based methods typically require on the order of hundreds to thousands of seconds. As network size increases, the training time for our method grows at a moderate rate, while the training time for VGAE increases much faster. This indicates that the computational cost of SyNG-D remains relatively stable as network size grows, demonstrating its advantages on large-scale networks. The training time of EDGE and GRAN remains high across all dataset sizes.

For sampling speed, on networks with fewer than 1,000 nodes, both SyNG-D and VGAE complete a single draw in under 0.1 s on average, whereas EDGE and GRAN require a few seconds. SyNG-D requires only a small number of diffusion steps to generate high-quality latent embeddings. For larger networks with 5,000 nodes, SyNG-D samples a synthetic network in only a few seconds, significantly faster than GRAN and EDGE, which require tens of seconds. The sampling time of SyNG-D is also more stable as network size increases, compared with the other methods.

These results highlight the scalability of SyNG-D and its advantage for large graphs.