Graph Generation with Diffusion Mixture

Jaehyeong Jo [1 *] Dongki Kim [1 *] Sung Ju Hwang [1 2]

Abstract

Generation of graphs is a major challenge for real-world tasks that require understanding the complex nature of their non-Euclidean structures. Although diffusion models have achieved notable success in graph generation recently, they are ill-suited for modeling the topological properties of graphs since learning to denoise the noisy samples does not explicitly learn the graph structures to be generated. To tackle this limitation, we propose a generative framework that models the topology of graphs by explicitly learning the final graph structures of the diffusion process. Specifically, we design the generative process as a mixture of endpoint-conditioned diffusion processes which is driven toward the predicted graph that results in rapid convergence. We further introduce a simple parameterization of the mixture process and develop an objective for learning the final graph structure, which enables maximum likelihood training. Through extensive experimental validation on general graph and 2D/3D molecule generation tasks, we show that our method outperforms previous generative models, generating graphs with correct topology with both continuous (e.g. 3D coordinates) and discrete (e.g. atom types) features. Our code is available at https://github.com/harryjo97/GruM.

1. Introduction

Generation of graph-structured data has emerged as a crucial task for real-world problems such as drug discovery (Simonovsky & Komodakis, 2018), protein design (Ingraham et al., 2019), and program synthesis (Brockschmidt et al., 2019). To tackle the challenge of learning the underlying distribution of graphs, deep generative models have been proposed, including models based on generative adversarial networks (GANs) (De Cao & Kipf, 2018; Martinkus et al., 2022), recurrent neural networks (RNNs) (You et al., 2018), and variational autoencoders (VAEs) (Jin et al., 2018).

Recently, diffusion models have achieved state-of-the-art performance on the generation of graph-structured data (Niu et al., 2020; Jo et al., 2022; Hoogeboom et al., 2022). These models learn the generation process as the time reversal of the forward process, which corrupts the graphs by gradually adding noise that destroys its topological properties. Since the generative process is derived from the unknown score function (Song et al., 2021) or noise (Ho et al., 2020), existing graph diffusion models aim to estimate them in order to denoise the data from noise, which are commonly referred to as the *denoising diffusion models* (Figure 1 (a)).

Despite their success, learning the score or noise is fundamentally ill-suited for the generation of graphs. In contrast to other types of data such as images, the key to generating valid graphs is accurately modeling the discrete structures that determine the topological properties such as connectivity or clusteredness. However, learning the score or noise does not explicitly model these features, as it aims to gradually denoise the corrupted structures. Thereby it is challenging for the diffusion models to recover the topological properties, which leads to failure cases even for small graphs. A way to more accurately generate graphs with correct topology would be directly learning the final graph and its structure, instead of learning how to denoise a noisy version of the original graphs.

However, predicting the final graph structure of the diffusion process is difficult since the prediction would be highly inaccurate in the early steps of the diffusion process, and such an inaccurate prediction may lead the process in the wrong direction resulting in invalid results. Few existing works (Hoogeboom et al., 2021; Austin et al., 2021; Vignac et al., 2023) based on denoising diffusion models aim to predict the probability of the final states by parameterizing the denoising process, but it is only applicable to categorical data with a finite number of states and thus cannot generate graphs with continuous features, which is not suitable for tasks such as 3D molecule generation.

To address these limitations of existing graph diffusion models, we propose a novel framework that explicitly models the

[*]Equal contribution [1]Korea Advanced Institute of Science and Technology (KAIST) [2]DeepAuto.ai. Correspondence to: Jaehyeong Jo <harryjo97@kaist.ac.kr>, Dongki Kim <cleverki@kaist.ac.kr>, Sung Ju Hwang <sjhwang82@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

graph topology, by learning the prediction of the resulting graph of the generative process which is represented as a weighted mean of graph data (Figure 1 (b)). Specifically, we construct a diffusion process via the mixture of endpoint-conditioned Ornstein-Uhlenbeck processes for which the drift drives the process in the direction of the predicted graph, differing from the denoising diffusion process used in previous works (Section 3.1). In order to model the mixture of the diffusion process, we develop a simple parameterization of the graph generative model with respect to the prediction of the final graph. We further derive an efficient training objective for learning the graph prediction, which guarantees to maximize the likelihood of our generative model (Section 3.2). Thanks to its ability to capture accurate graph structures, our framework achieves fast convergence to the correct graph topology in an early sampling step (Figure 1 (c) and Figure 3 (Right)).

We experimentally validate our method on diverse real-world graph generation tasks. We first validate it on general graph generation benchmarks with synthetic and real-world graphs, on which it outperforms previous deep graph generative models including graph diffusion models, by being able to generate valid graphs with correct topologies. We further validate our method on 2D and 3D molecule generation tasks to demonstrate its ability to generate graphs with both the continuous and discrete features, on which ours generates a significantly larger number of valid and stable molecules compared to the state-of-the-art baselines. Our main contributions can be summarized as follows:

- We observe that previous diffusion models cannot accurately model the graph structures as they learn to denoise at each step without considering the topology of the graphs to be generated.
- To fix such a myopic behavior of previous diffusion models, we propose a new graph generation framework that captures the graph structures by directly predicting the final graph of the diffusion process modeled by a mixture of endpoint-conditioned diffusion processes.
- We develop a simple parameterization of the graph generative model for modeling the mixture process and present a simulation-free training objective for graph prediction.
- Our method significantly outperforms previous graph diffusion models on the generation of diverse real and synthetic graphs, as well as on 2D/3D molecule generation tasks, by being able to generate graphs with accurate topologies, and both the discrete and continuous features.

2. Related Work

Diffusion Models Diffusion models have been shown to successfully generate high-quality samples from diverse data domains such as images (Dhariwal & Nichol, 2021; Saharia et al., 2022) and videos (Ho et al., 2022). Despite their success, existing diffusion models for graphs (Niu et al., 2020; Jo et al., 2022) often fail to generate graphs with correct structures since they learn to estimate the score or noise for the denoising process which does not explicitly capture the final graph and its structure. To address these limitations, we propose a graph diffusion framework that models the generative process as a mixture of diffusion processes, which learns to predict the final graph with valid topology instead of predicting the denoising function at each step. This promotes our generative process to be driven toward the prediction of the final graph, resulting in generation of valid graphs with correct topology.

Diffusion Bridge Process A line of recent works has improved the generative framework of diffusion models by leveraging the diffusion bridge processes, i.e., processes conditioned to the endpoints. Schrödinger Bridge (Bortoli et al., 2021b; Chen et al., 2022) aims to find both the forward and the backward process that transforms two distributions back and forth using iterative proportional fittings that require heavy computations. More recent works (Peluchetti, 2021; Wu et al., 2022; Liu et al., 2023; Jo & Hwang, 2023) consider learning the generation process as a mixture of diffusion processes instead of reversing the noising process as in denoising diffusion models, which we describe in detail in Appendix A.11. However, previous works aim to approximate the drifts of the diffusion processes which cannot accurately capture the discrete structure of graphs as it does not explicitly learn the graph structures to be generated. Moreover, learning the drift could be problematic as the drift of the diffusion process diverges near the terminal time. Instead, we present a new approach to parameterizing the mixture process with the prediction of the final graph which allows it to model valid graph topology.

Graph Generative Models Deep generative models for graphs either generate nodes and edges in an autoregressive manner or all the nodes and edges at once using VAE (Jin et al., 2018), RNN (You et al., 2018), normalizing flow (Zang & Wang, 2020; Shi et al., 2020; Luo et al., 2021), and GAN (De Cao & Kipf, 2018; Martinkus et al., 2022). However, these models show poor performance due to restrictive model architectures for modeling the likelihood or their inability to model the permutation equivariant nature of graphs. Recently, diffusion models for graphs (Jo et al., 2022; Hoogeboom et al., 2022; Vignac et al., 2023) have made large progress, but either fail to capture the graph topology or are not applicable to general tasks due to the architectural restriction of the framework. In our work, we introduce a diffusion framework that predicts the final graph structure instead of denoising noisy graphs. Our method largely outperforms existing models (Jo et al., 2022; Vignac et al., 2023; Hoogeboom et al., 2022) on generation tasks including general graphs as well as 2D and 3D molecules.

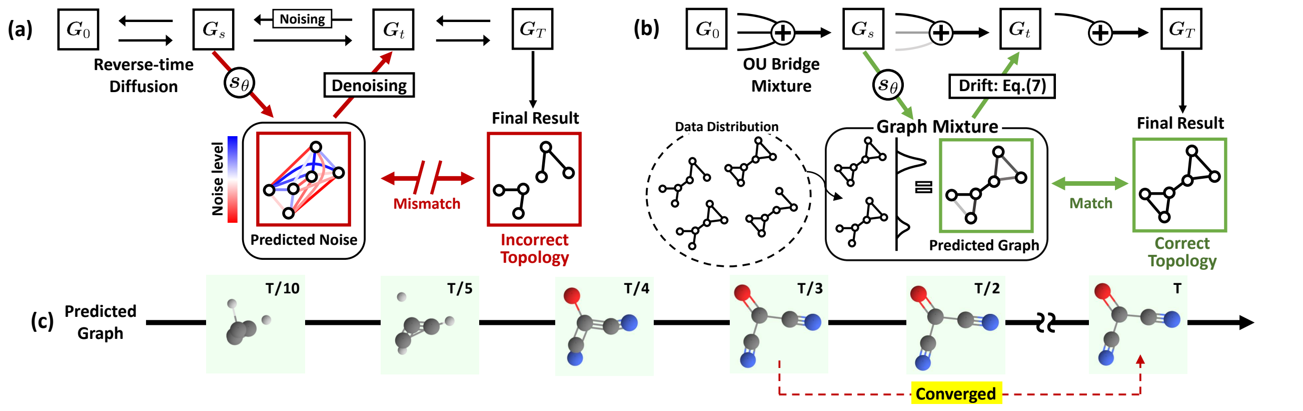


Figure 1: **Illustration of the graph generative process.** (a: Denoising diffusion model, b: GruM (ours), c: Graph mixture) For GruM, we design the generative process as a mixture of endpoint-conditioned diffusion processes (Eq. (3)), namely the OU bridge mixture (Eq. (6)), which is driven toward the graph mixture (**green**) by its drift (Eq. (8)). Our GruM in (b) successfully generates graphs with valid topology by predicting the final result via learning the graph mixture as a weighted mean of data (Eq. (1)). The predicted graph of GruM converges in an early stage to the correct topology as visualized in (c). In contrast, previous denoising diffusion models in (a) often fail to capture the correct topology as they learn the score or noise for denoising (**red**), without explicit knowledge of final graph structure.

3. Graph Diffusion Mixture

In this section, we present our graph generation framework Graph Diffusion Mixture (GruM), for modeling valid topology of graphs using a mixture of diffusion processes.

3.1. Designing Graph Generative Process

The key to generating graph-structured data is understanding the underlying topology of graphs which is crucial to determining its validity, since a slight modification in the edges may significantly change its structure and the attributes, for example, planarity or molecular properties. However, previous diffusion models fail to do so as their objective is to denoise the noisy graphs, in which the topology is only implicitly captured (Figure 1 (a)) from the noisy structure. To overcome the limitation, we propose a graph diffusion framework that can directly learn the accurate graph structures and capture valid topology.

Throughout the paper, we represent a graph with N nodes as a pair $\boldsymbol{G}=(\boldsymbol{X},\boldsymbol{A})$ where $\boldsymbol{X}\in\mathbb{R}^{N\times F}$ is the node features of feature dimension F and $\boldsymbol{A}\in\mathbb{R}^{N\times N}$ is the weighted adjacency matrix that defines the connection between nodes.

Graph Mixture Our goal is to directly predict the final graph of the diffusion process that transports a prior distribution to the data distribution Π^*. To be specific, for a graph diffusion process represented as a trajectory of random variables $\{\boldsymbol{G}_\tau=(\boldsymbol{X}_\tau,\boldsymbol{A}_\tau)\}_{\tau\in[0,T]}$, we aim to predict the terminus of the process $\boldsymbol{G}_T$ in Π^* given the current state $\boldsymbol{G}_t$. However, identifying the exact $\boldsymbol{G}_T$ at the early stage of the process is problematic since the prediction based on $\boldsymbol{G}_t$ of almost no information would be highly inaccurate, and could lead the process in the wrong direction.

To address this problem, we present a different approach to predicting the probable graph, which we define as a weighted mean of all the possible final results (Figure 1 (b)). Since the probability of a graph $\boldsymbol{g}$ being the final result is equal to the transition probability of the process denoted as $p_{T|t}(\boldsymbol{g}|\cdot)$, we define the probable graph given the current state $\boldsymbol{G}_t$ via the expectation of the graphs as follows:

$$\boldsymbol{D}(\boldsymbol{G}_t,t)=\int \boldsymbol{g}\cdot p_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\,\mathrm{d}\boldsymbol{g}, \tag{1}$$

which we refer to as the *graph mixture* of the process, visualized in Figure 1. In order to explicitly model this, we construct a generative process driven toward the graph mixture using a mixture of diffusion processes, which we describe in the following paragraphs.

Ornstein-Uhlenbeck Bridge Process As a building block of our generative framework, we leverage diffusion processes with fixed endpoints, namely the *diffusion bridge* processes. We propose to use a family of bridge processes, namely the *Ornstein-Uhlenbeck* (OU) bridge process that enables simulation-free training for our generative model while providing flexibility for modeling the complex generative process for graphs.

Given an OU process $\mathbb{Q}$ modeled by the following SDE:

$$\mathbb{Q}:\ \mathrm{d}\boldsymbol{G}_t=\alpha\sigma_t^2\boldsymbol{G}_t\mathrm{d}t+\sigma_t\mathrm{d}\mathbf{W}_t, \tag{2}$$

where α is a constant, σ_t is a noise schedule, and $\mathbf{W}_t$ is the standard Wiener process, the OU bridge process $\mathbb{Q}^{\boldsymbol{g}}$ is the process $\mathbb{Q}$ pinned at a fixed terminal point g. Using the Doob's h-transform (Doob & Doob, 1984), we can derive the OU bridge process $\mathbb{Q}^{\boldsymbol{g}}$ as follows (we provide detailed derivation of the bridge process in Appendix A.1):

$$\mathrm{d}\boldsymbol{G}_t = \Bigg[\underbrace{\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t\right)}_{\eta^{\boldsymbol{g}}(\boldsymbol{G}_t,t)}\Bigg]\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \quad (3)$$

where the scalar functions u_t and v_t are defined as follows:

$$u_t = \exp\left(\alpha\int_t^T \sigma_\tau^2 \mathrm{d}\tau\right), \; v_t = \frac{1}{2\alpha}\left(1-u_t^{-2}\right). \quad (4)$$

The endpoint of $\mathbb{Q}^{\boldsymbol{g}}$ is fixed to $\boldsymbol{G}_T = \boldsymbol{g}$, since the drift $\eta^{\boldsymbol{g}}(\cdot,t)$ of the process forces the trajectory $\boldsymbol{G}_t$ in the direction of $\boldsymbol{g}$. Although there exists a more general class of bridge processes with non-linear drift (see Appendix A.1), they have intractable transition probability and require expensive SDE simulation to obtain trajectories. In contrast, the OU bridge processes yield tractable transition probabilities due to their affine nature and allow the training of our generative model to be simulation-free, which we further discuss in Section 3.2. Note that the Brownian bridge process used in previous works (Wu et al., 2022; Liu et al., 2022) is a special case of the OU bridge process when $\alpha \to 0$ (see Appendix A.1). Especially, we can write the OU bridge process of Eq. (3) for graphs $\boldsymbol{g} = (\boldsymbol{x},\boldsymbol{a})$ as a system of SDEs:

$$\begin{cases} \mathrm{d}\boldsymbol{X}_t = \left[\alpha_1\sigma_{1,t}^2\boldsymbol{X}_t + \frac{\sigma_{1,t}^2}{v_{1,t}}\left(\frac{\boldsymbol{x}}{u_{1,t}} - \boldsymbol{X}_t\right)\right]\mathrm{d}t + \sigma_{1,t}\mathrm{d}\mathbf{W}_{1,t} \\ \mathrm{d}\boldsymbol{A}_t = \left[\alpha_2\sigma_{2,t}^2\boldsymbol{A}_t + \frac{\sigma_{2,t}^2}{v_{2,t}}\left(\frac{\boldsymbol{a}}{u_{2,t}} - \boldsymbol{A}_t\right)\right]\mathrm{d}t + \sigma_{2,t}\mathrm{d}\mathbf{W}_{2,t} \end{cases} \quad (5)$$

With the OU bridge processes in hand, we develop a framework for predicting the final graph via the graph mixture.

Diffusion Mixture for Graph Generation As the graph mixture in Eq. (1) is a weighted mean of the final graphs, conceptually, this can be modeled by aggregating the endpoint-conditioned processes with respect to the weights from the graph mixture. Inspired by the diffusion mixture framework (Peluchetti, 2021; Wu et al., 2022; Liu et al., 2022), we design the generation process by mixing the OU bridge processes with the endpoints from the data distribution, where we leverage the *diffusion mixture representation* (Brigo, 2008; Peluchetti, 2021). This yields the SDE representation of a mixture process as a weighted mean of the SDEs of the diffusion processes (we provide a formal definition of the mixture representation in Appendix A.2).

Specifically, we mix a collection of OU bridge processes $\{\mathbb{Q}^{\boldsymbol{g}} : \boldsymbol{g} = (\boldsymbol{x},\boldsymbol{a}) \sim \Pi^*\}$ to construct a generation process, for which the *mixture process* is modeled by the SDE:

$$\mathbb{Q}^{\Pi^*}: \begin{cases} \mathrm{d}\boldsymbol{X}_t = \eta_1(\boldsymbol{X}_t,\boldsymbol{A}_t,t)\mathrm{d}t + \sigma_{1,t}\mathrm{d}\mathbf{W}_{1,t} \\ \mathrm{d}\boldsymbol{A}_t = \eta_2(\boldsymbol{X}_t,\boldsymbol{A}_t,t)\mathrm{d}t + \sigma_{2,t}\mathrm{d}\mathbf{W}_{2,t} \end{cases} \quad (6)$$

with $\boldsymbol{G}_0 = (\boldsymbol{X}_0,\boldsymbol{A}_0)$ following an arbitrary prior distribution Γ and the drifts η_1 and η_2 defined as follows:

$$\begin{pmatrix}\eta_1(\boldsymbol{X}_t,\boldsymbol{A}_t,t)\\ \eta_2(\boldsymbol{X}_t,\boldsymbol{A}_t,t)\end{pmatrix} = \int \begin{pmatrix}\eta_1^{\boldsymbol{x}}(\boldsymbol{X}_t,t)\\ \eta_2^{\boldsymbol{a}}(\boldsymbol{A}_t,t)\end{pmatrix} \frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) \quad (7)$$

for $\boldsymbol{G}_t = (\boldsymbol{X}_t,\boldsymbol{A}_t)$, where $p_t^{\boldsymbol{g}}$ is the marginal density of the bridge process $\mathbb{Q}^{\boldsymbol{g}}$, and $p_t(\cdot) := \int p_t^{\boldsymbol{g}}(\cdot)\Pi^*(\mathrm{d}\boldsymbol{g})$ is the marginal density of the mixture process. Notably, the terminal distribution of the mixture process $\mathbb{Q}^{\Pi^*}$ is equal to the data distribution Π^* by construction. We refer to this mixture process as the *OU bridge mixture*.

Remarkably, the mixture process $\mathbb{Q}^{\Pi^*}$ can be explicitly represented in terms of the graph mixture. We derive a parameterization of $\mathbb{Q}^{\Pi^*}$ from the SDE representation of the OU bridge process in Eq. (3) as follows (see Appendix A.3 for the derivation):

$$\begin{aligned} \eta_1(\boldsymbol{X}_t,\boldsymbol{A}_t,t) &= \alpha_1\sigma_{1,t}^2\boldsymbol{X}_t + \frac{\sigma_{1,t}^2}{v_{1,t}}\left(\frac{\boldsymbol{D}_X(\boldsymbol{X}_t,\boldsymbol{A}_t,t)}{u_{1,t}} - \boldsymbol{X}_t\right) \\ \eta_2(\boldsymbol{X}_t,\boldsymbol{A}_t,t) &= \alpha_2\sigma_{2,t}^2\boldsymbol{A}_t + \frac{\sigma_{2,t}^2}{v_{2,t}}\left(\frac{\boldsymbol{D}_A(\boldsymbol{X}_t,\boldsymbol{A}_t,t)}{u_{2,t}} - \boldsymbol{A}_t\right) \end{aligned} \quad (8)$$

where $\boldsymbol{D}_X$ and $\boldsymbol{D}_A$ are defined as a weighted mean of the node features and the adjacency matrices respectively:

$$\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) := \begin{pmatrix}\boldsymbol{D}_X(\boldsymbol{G}_t,t)\\ \boldsymbol{D}_A(\boldsymbol{G}_t,t)\end{pmatrix} = \int \begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{a}\end{pmatrix} \frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) \quad (9)$$

Notice that from the definition of the transition distribution, we can derive the following identity:

$$\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) = \int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) = \int \boldsymbol{g}\cdot p_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\mathrm{d}\boldsymbol{g},$$

which shows that $\boldsymbol{D}^{\Pi^*}(\cdot,t)$ coincides with the graph mixture of $\mathbb{Q}^{\Pi^*}$ as in Eq. (1). As a result, $\boldsymbol{D}^{\Pi^*}(\cdot,t)$ acts as the prediction of the final graph at time t, where $\boldsymbol{D}_X$ and $\boldsymbol{D}_A$ are the predicted node features and adjacency matrices, respectively, in the form of a weighted mean of data.

In the view of the graph mixture as the weighted mean from $\boldsymbol{D}^{\Pi^*}$, it converges to the final graph of the mixture process since the marginal density $p_t^{\boldsymbol{g}}$ of the bridge process converges to one if $\boldsymbol{g}$ corresponds to the final graph while the probability becomes zero otherwise. This convergence is achieved at an early stage as visualized in Figure 1 (c) and Appendix E.2, where we further analyze the convergence behavior with respect to the coefficient α and the noise schedule σ_t in Appendix D.2.

A key observation is that the drift of the OU bridge mixture in Eq. (8) highly resembles the drift of the OU bridge process in Eq. (3), except that the final graph $\boldsymbol{g}$ is replaced by the graph mixture. From this observation, we can see that the trajectory of the mixture process is guided by the drift in the direction of $\boldsymbol{D}^{\Pi^*}(\cdot,t)$, driven toward the graph mixture that converges to a graph in the data distribution Π^*. Therefore, if we could estimate the graph mixture of this process, we can build a generative model upon the mixture process without relying on score function or noise, where the graph structures and the topological attributes can be explicitly modeled by the graph mixture.

Before introducing the training objective for learning the graph mixture, we discuss the difference between our framework and the denoising diffusion models. Our generative process is modeled by the mixture of bridge processes that describes the exact transport from the prior distribution to the data distribution by construction, whereas the time reversal of denoising diffusion models is not an exact transport to the data distribution for finite time (Franzese et al., 2023). We provide further discussion on the difference in the characteristics of our mixture process and the denoising diffusion processes in Appendix A.10.

3.2. Generation Framework Using Graph Mixture

Training Objectives Our goal is to design a generative model that explicitly learns the graph topology. To this end, we leverage the OU bridge mixture parameterized by the graph mixture, where we estimate the graph mixture using a neural network $\boldsymbol{s}_\theta(\cdot, t)$ that corresponds to directly learning the graph structures. In particular, we show that estimating the graph mixture guarantees to maximize the likelihood of our generative model. For the rest of the section, we represent the system of SDEs of Eq. (6) as a SDE with respect to $\boldsymbol{G}_t$ for notational simplicity.

We propose to define the generative model $\mathbb{P}^\theta$ to approximate the mixture process $\mathbb{Q}^{\Pi^*}$ as follows:

$$\begin{aligned}
&\mathbb{P}^\theta : \mathrm{d}\boldsymbol{G}_t = \eta_\theta(\boldsymbol{G}_t, t)\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \\
&\eta_\theta(\boldsymbol{G}_t, t) = \alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_t\right),
\end{aligned} \tag{10}$$

where $\boldsymbol{s}_\theta$ is desired to estimate the graph mixture $\boldsymbol{D}^{\Pi^*}$.

In order to model the drift η_θ, we provide a tractable objective for estimating the graph mixture, which guarantees to maximize the likelihood of our generative model $\mathbb{P}^\theta$. Leveraging the Girsanov theorem (Øksendal, 2003), we upper-bound the KL divergence between Π^* and the terminal distribution of $\mathbb{P}^\theta$ denoted as p_T^θ as follows (see Appendix A.5 for a detailed derivation of the objective):

$$\begin{aligned}
&D_{KL}(\Pi^* \| p_T^\theta) \leq D_{KL}(\mathbb{Q}^{\Pi^*} \| \mathbb{P}^\theta) \\
&= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t)\right\|^2 \mathrm{d}t\right] + C_1 \\
&= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_T\right\|^2 \mathrm{d}t\right] + C_2,
\end{aligned} \tag{11}$$

where $\gamma_t \coloneqq \sigma_t/(u_t v_t)$, C_1 and C_2 are constants independent of θ, and the expectation is computed over the samples $\boldsymbol{G}$ from the OU bridge mixture $\mathbb{Q}^{\Pi^*}$.

During training, $\boldsymbol{G}_t$ from the mixture process $\mathbb{Q}^{\Pi^*}$ can be easily obtained without simulating the SDE. Notice that $\boldsymbol{G}_t$ follows the distribution $p_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0, \boldsymbol{G}_T)$, which is the distribution of the mixture process $\mathbb{Q}^{\Pi^*}$ at time t given the endpoints $\boldsymbol{G}_0$ and $\boldsymbol{G}_T$. By construction, the OU bridge mixture with fixed endpoints $\boldsymbol{G}_0$ and $\boldsymbol{G}_T$ coincides with the OU process (Eq. (2)) with these fixed endpoints, and $p_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0, \boldsymbol{G}_T)$ corresponds to the marginal probability of the OU process with the fixed endpoints $\boldsymbol{G}_0$ and $\boldsymbol{G}_T$.

Using the Bayes theorem, we derive that the distribution $p(\boldsymbol{G}_t|\boldsymbol{G}_0, \boldsymbol{G}_T)$ is also Gaussian that results from the product of Gaussian distributions, where the mean μ_t^* and the covariance $\boldsymbol{\Sigma}_t^*$ have analytical forms as follows (see Appendix A.6 for the derivation):

$$\begin{aligned}
\mu_t^* &= \frac{\sinh(\varphi_T - \varphi_t)}{\sinh(\varphi_T)}\boldsymbol{G}_0 + \frac{\sinh(\varphi_t)}{\sinh(\varphi_T)}\boldsymbol{G}_T, \\
\boldsymbol{\Sigma}_t^* &= \frac{1}{\alpha}\frac{\sinh(\varphi_T - \varphi_t)\sinh(\varphi_t)}{\sinh(\varphi_T)}\mathbf{I},
\end{aligned} \tag{12}$$

where $\varphi_t \coloneqq \alpha\int_0^t \sigma_\tau^2 \mathrm{d}\tau$. Thereby the training of GruM is simulation-free, and our approach is 17.5 times faster compared to the training that relies on expensive SDE simulation (Wu et al., 2022).

In particular, Eq. (12) shed light on the connection of the OU bridge mixture and the stochastic interpolant (Albergo et al., 2023) between the distributions Γ and Π^* as follows:

$$\begin{aligned}
\boldsymbol{G}_t =& \frac{\sinh(\varphi_T - \varphi_t)}{\sinh(\varphi_T)}\boldsymbol{G}_0 + \frac{\sinh(\varphi_t)}{\sinh(\varphi_T)}\boldsymbol{G}_T \\
&+ \left(\frac{1}{\alpha}\frac{\sinh(\varphi_T - \varphi_t)\sinh(\varphi_t)}{\sinh(\varphi_T)}\right)^{1/2}\boldsymbol{Z}.
\end{aligned} \tag{13}$$

where $\boldsymbol{G}_0 \sim \Gamma$, $\boldsymbol{G}_T \sim \Pi^*$, and $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, respectively.

Note that the goal of Eq. (11) is to model the drift of the OU bridge mixture parametrized by the graph mixture, where $\boldsymbol{s}_\theta$ is trained to estimate the graph mixture instead of the exact graph $\boldsymbol{G}_T$, and we refer to this objective as the *graph mixture matching*. Learning the graph mixture not only allows us to directly model the structures of the final graph and their topological properties, but further guarantees our generative model to closely approximate the data distribution. Additionally, we discuss the difference between learning the graph mixture and the training objectives of denoising diffusion models in Appendix A.9 and A.10. We summarize the training process in Algorithm 1 and provide the details in Appendix B.2.

Sampling from GruM Using the trained model $\boldsymbol{s}_\theta$ to compute the drift η_θ of the parameterized mixture process in Eq. (10), we generate samples by simulating $\mathbb{P}^\theta$ from time

Algorithm 1 Training

```
Input: Model $s_\theta$, constant $\epsilon$
For each epoch:
1:  Sample graph $G$ from the training set
2:  $N \leftarrow$ number of nodes of $G$
3:  Sample $t \sim [0, T-\epsilon]$ and $G_0 \sim \mathcal{N}(0, \mathbf{I}_N)$
4:  Sample $G_t \sim p_{t|0,T}(G_t|G_0, G)$                ▷ Eq. (13)
5:  $\gamma_t \leftarrow \sigma_t / u_t v_t$
6:  $\mathcal{L}_\theta \leftarrow \gamma_t^2 \| s_\theta(G_t, t) - G \|^2$
7:  Update $\theta$ using $\mathcal{L}_\theta$
```

Algorithm 2 Sampling

```
Input: Trained model $s_\theta$, number of sampling steps $K$, diffusion step size $\mathrm{d}t$
1:  Sample number of nodes $N$ from the training set.
2:  $G_0 \sim \mathcal{N}(0, \mathbf{I}_N)$                    ▷ Start from noise
3:  $t \leftarrow 0$
4:  for $k = 1$ to $K$ do
5:      $\eta \leftarrow \alpha\sigma_t^2 G_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t} s_\theta(G_t, t) - G_t\right)$
6:      $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
7:      $G_{t+\mathrm{d}t} \leftarrow \eta \mathrm{d}t + \sigma_t \sqrt{\mathrm{d}t}\mathbf{w}$   ▷ Euler-Maruyama Step
8:      $t \leftarrow t + \mathrm{d}t$
9:  end for
10: $G \leftarrow \mathtt{quantize}(G_t)$           ▷ Quantize if necessary
11: Return: Graph $G$
```

$t = 0$ to $t = T$ with initial samples drawn from the prior distribution. Note that we generate the node features and the adjacency matrices simultaneously using the system of SDEs in the form of Eq. (6), and solving the SDEs is similar to that of denoising diffusion models which does not require additional time. We summarize the sampling process in Algorithm 2 and describe the details in Appendix B.4.

Advantages of Our Framework We conclude this section by explaining the advantages of our framework. First, GruM can directly model the graph topology by predicting the graph structures via the graph mixture, instead of implicitly capturing via noise or score. Furthermore, our framework is not restricted to the type of data to be generated, allowing it to be applicable to both continuous and discrete data, for example, 3D molecules with both discrete atom types and continuous coordinates.

From the perspective of the model hypothesis space, learning the graph mixture is considerably easier compared to previous objectives such as learning the score function or the drift of the diffusion process. While the graph mixture is supported inside the bounded data space, the score function or the drift tends to diverge near the terminal time which could be problematic for the model to learn. Furthermore, we can exploit the inductive bias of the graph data for learning the graph mixture, which is critical as it dramatically reduces the hypothesis space. To be specific, we can leverage the prior knowledge of the graph representation such as one-hot encoding or the categorical type by adding an additional function at the last layer of the model s_θ, for instance, softmax function for the one-hot encoded node features and the sigmoid function for the 0-1 adjacency matrices (we provide more details in Appendix B.3). We experimentally verify these advantages in Section 4.4.

4. Experiments

4.1. General Graph Generation

We validate GruM on general graph generation tasks to show that it can generate valid graph topology.

Datasets and Metrics We evaluate the quality of generated graphs on three synthetic and real datasets used as benchmarks in previous works (Martinkus et al., 2022; Vignac et al., 2023): **Planar**, Stochastic Block Model (**SBM**), and **Proteins** (Dobson & Doig, 2003). We follow the evaluation setting of Martinkus et al. (2022) using the same data split. We measure the maximum mean discrepancy (MMD) of four graph statistics between the set of generated graphs and the test set: degree (**Deg.**), clustering coefficient (**Clus.**), count of orbits with 4 nodes (**Orbit**), and the eigenvalues of the graph Laplacian (**Spec.**). To verify that the model truly learns the distribution, we report the percentage of valid, unique, and novel (**V.U.N.**) graphs for which the validness is defined as satisfying the specific property of each dataset. We provide further details in Appendix C.1.

Baselines We compare our method against the following graph generative models: **GraphRNN** (You et al., 2018) an autoregressive model based on RNN, **GRAN** (Liao et al., 2019) an autoregressive model with attention, **SPECTRE** (Martinkus et al., 2022) a one-shot model based on GAN, **EDP-GNN** (Niu et al., 2020) a score-based model for adjacency matrix, **GDSS** (Jo et al., 2022) and **ConGress** (Vignac et al., 2023) a continuous diffusion model, and **DiGress** (Vignac et al., 2023), a discrete diffusion model. We provide the details of training and sampling of our GruM in Appendix B and describe further implementation details including the hyperparameters in Appendix C.1.

Results Table 1 shows that our method outperforms all the baselines on all datasets. Especially, ours achieves the highest validity (V.U.N.) metric, as it accurately learns the underlying topology of the graphs. Notably, our method outperforms DiGress by a large margin in V.U.N., even though we do not use specific prior distributions or structural feature augmentation that are utilized in DiGress. We provide an ablation study on the model architecture in Appendix D.2 to validate that the superior performance of GruM comes from its ability to accurately model the graph topology by

Table 1: **Generation results on the general graph datasets.** Best results are highlighted in bold, where smaller MMD and larger V.U.N. indicate better results. Hyphen(-) denotes out-of-resources that take more than 2 weeks.

	Planar					SBM					Proteins			
	Synthetic, $\lvert V\rvert = 64$					Synthetic, $44 \leq \lvert V\rvert \leq 187$					Real, $100 \leq \lvert V\rvert \leq 500$			
	Deg.	Clus.	Orbit	Spec.	V.U.N.	Deg.	Clus.	Orbit	Spec.	V.U.N.	Deg.	Clus.	Orbit	Spec.
Training set	0.0002	0.0310	0.0005	0.0052	100.0	0.0008	0.0332	0.0255	0.0063	100.0	0.0003	0.0068	0.0032	0.0009
GraphRNN	0.0049	0.2779	1.2543	0.0459	0.0	0.0055	0.0584	0.0785	0.0065	5.0	0.0040	0.1475	0.5851	0.0152
GRAN	0.0007	0.0426	0.0009	0.0075	0.0	0.0113	0.0553	0.0540	0.0054	25.0	0.0479	0.1234	0.3458	0.0125
SPECTRE	0.0005	0.0785	0.0012	0.0112	25.0	0.0015	0.0521	**0.0412**	0.0056	52.5	0.0056	0.0843	**0.0267**	0.0052
EDP-GNN	0.0044	0.3187	1.4986	0.0813	0.0	0.0011	0.0552	0.0520	0.0070	35.0	-	-	-	-
GDSS	0.0041	0.2676	0.1720	0.0370	0.0	0.0212	0.0646	0.0894	0.0128	5.0	0.0861	0.5111	0.732	0.0748
ConGress	0.0048	0.2728	1.2950	0.0418	0.0	0.0273	0.1029	0.1148	-	0.0	-	-	-	-
DiGress	**0.0003**	0.0372	**0.0009**	0.0106	75	0.0013	0.0498	0.0434	0.0400	74	-	-	-	-
GruM (Ours)	0.0005	**0.0353**	**0.0009**	**0.0062**	**90.0**	**0.0007**	**0.0492**	0.0448	**0.0050**	**85.0**	**0.0019**	**0.0660**	0.0345	**0.0030**

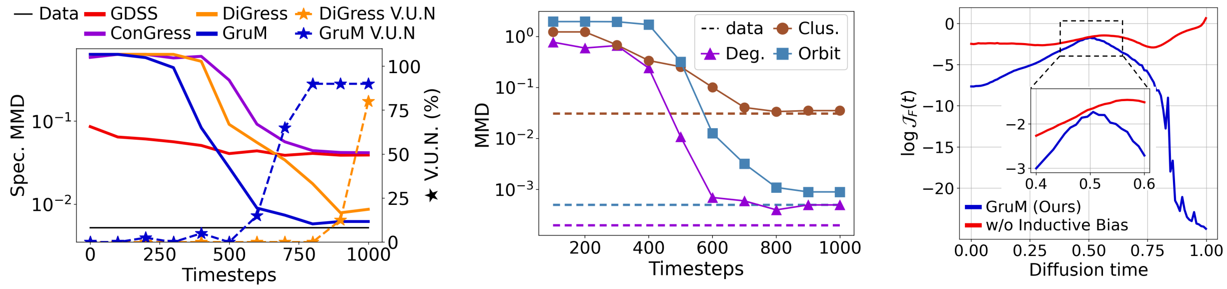


Figure 2: **(Left) Topology analysis**. We compare Spec. MMD and V.U.N of the graph mixture from GruM against the implicit prediction computed from GDSS, ConGress, and DiGress which we provide details in Appendix C.1. **(Middle) MMD between the test set and the graph mixture of GruM** through the generative process. **(Right) The complexity of GruM** with and without using the inductive bias, measured by the Frobenius norm of the Jacobian of the models.

predicting the graph mixture. We provide the visualization of the generated graphs and the generative process of GruM in Appendix E, showing that it can accurately capture the attributes of each dataset.

Topology Analysis To show how learning the graph mixture results in graphs with correct topology, we conduct an analysis of the graph mixture. Figure 2 (Left) demonstrates that GruM can achieve the spectral property of the final graph at an early stage by explicitly modeling the topology via learning the graph mixture. In contrast, GDSS and ConGress fail to recover the spectral properties as they implicitly model the topology via predicting the noise or score functions. Further, ours recovers the spectral property faster than DiGress, resulting in graphs with higher validity. In particular, we observe that the V.U.N. of the estimated graph mixture increases after achieving the desired spectral property, resulting in 90% V.U.N. This shows that predicting the final graph allows us to better capture the global topologies. Moreover, we plot the MMD results of GruM through the generative process in Figure 2 (Middle), which demonstrates that the local characteristics of the predicted graph rapidly converge to that of the graphs from the training set.

4.2. 2D Molecule Generation

We further validate GruM on 2D molecule generation tasks to show that it can accurately generate graphs with both the node features and the topologies of the target graphs.

Datasets and Metrics We evaluate the quality of generated 2D molecules on two molecule datasets used as benchmarks in Jo et al. (2022): **QM9** (Ramakrishnan et al., 2014) and **ZINC250k** (Irwin et al., 2012). Following the evaluation setting of Jo et al. (2022), we evaluate the models with four metrics: **Validity** is the percentage of the valid molecules among the generated without any posthoc correction. **FCD** (Preuer et al., 2018) measures the distance between the sets of molecules in the chemical space. **NSPDK MMD** (Costa & De Grave, 2010) evaluates the quality of the graph structure compared to the test set. **Scaffold similarity** (Scaf.) evaluates the ability to generate similar substructures. We provide more details in Appendix C.2.

Baselines We compare to the following molecular graph generative models: **MoFlow** (Zang & Wang, 2020) is a one-shot flow-based model. **GraphAF** (Shi et al., 2020) and **GraphDF** (Luo et al., 2021) are autoregressive flow-based model. **EDP-GNN**, **GDSS**, **ConGress**, and **DiGress** are

Table 2: **Generation results on the 2D molecule datasets.** We report the mean of 3 different runs. Best results are highlighted in bold. We provide the results of uniqueness, novelty and variance in Appendix D.1.

	QM9 ($\|V\| \leq 9$)				ZINC250k ($\|V\| \leq 38$)			
Method	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑
Training set	100.0	0.0398	0.0001	0.9719	100.0	0.0615	0.0001	0.8395
MoFlow (Zang & Wang, 2020)	91.36	4.467	0.0169	0.1447	63.11	20.931	0.0455	0.0133
GraphAF (Shi et al., 2020)	74.43	5.625	0.0207	0.3046	68.47	16.023	0.0442	0.0672
GraphDF (Luo et al., 2021)	93.88	10.928	0.0636	0.0978	90.61	33.546	0.1770	0.0000
EDP-GNN (Niu et al., 2020)	47.52	2.680	0.0046	0.3270	82.97	16.737	0.0485	0.0000
GDSS (Jo et al., 2022)	95.72	2.900	0.0033	0.6983	97.01	14.656	0.0195	0.0467
DiGress (Vignac et al., 2023)	98.19	**0.095**	0.0003	0.9353	94.99	3.482	0.0021	0.4163
GruM (Ours)	**99.69**	0.108	**0.0002**	**0.9449**	**98.65**	**2.257**	**0.0015**	**0.5299**

	QM9 ($\|V\| \leq 29$)		GEOM-DRUGS ($\|V\| \leq 181$)	
Method	Atom Stab.(%)	Mol. Stab.(%)	Atom Stab.(%)	Mol. Stab.(%)
G-Schnet (Gebauer et al., 2019)	95.7	68.1	-	-
EN-Flow (Satorras et al., 2021)	85.0	4.9	75.0	0.0
GDM (Hoogeboom et al., 2022)	97.0	63.2	75.0	0.0
EDM (Hoogeboom et al., 2022)	98.7 $\pm$0.1	82.0 $\pm$0.4	81.3	0.0
Bridge (Wu et al., 2022)	98.7 $\pm$0.1	81.8 $\pm$0.2	81.0 $\pm$0.7	0.0
Bridge+Force (Wu et al., 2022)	98.8 $\pm$0.1	84.6 $\pm$0.2	82.4 $\pm$0.7	0.0
GruM (Ours)	**98.81** $\pm$0.03	**87.34** $\pm$0.19	**82.96** $\pm$0.12	**0.51** $\pm$0.03

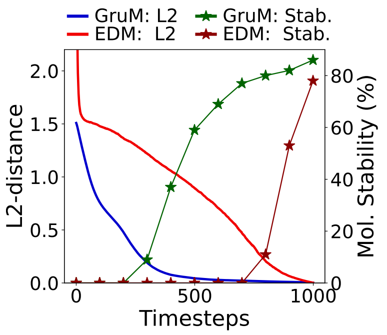


Figure 3: **(Left) Generation results on the 3D molecule datasets.** Best results are highlighted in bold which is the average of 3 different runs. The baseline results are taken from Hoogeboom et al. (2022) and Wu et al. (2022). **(Right) Convergence of the generative process.** We compare the convergence of the graph mixture from GruM and the implicit prediction computed from EDM. We measure the convergence (L2 distance) and report the molecule stability of the predictions.

diffusion models previously explained. We describe further implementation details in Appendix C.2.

Results Table 2 shows that our method achieves the highest validity on all datasets verifying that GruM can generate valid molecules without correction. Further, ours outperforms the baselines in FCD and NSPDK metrics demonstrating that the molecules synthesized by GruM are similar to the molecule from the training set in both chemical and graph-structure aspects. Especially, ours achieves the highest scaffold similarity indicating that it is able to generate similar substructures from that of the training set. We visualize the generated molecules in Appendix E.1.

4.3. 3D Molecule Generation

To show that GruM is able to generate graphs with both continuous and discrete features, we validate it on 3D molecule generation tasks, which come with discrete atom types and continuous coordinates.

Datasets and Metrics We evaluate the generated 3D molecules on two standard molecule datasets used as benchmarks in Hoogeboom et al. (2022): **QM9** (Ramakrishnan et al., 2014) (up to 29 atoms) and **GEOM-DRUGS** (Axelrod & Gomez-Bombarelli, 2022) (up to 181 atoms). Following Hoogeboom et al. (2022), both datasets include hydrogen atoms. For GEOM-DRUGS, we select 30 conformations for each molecule with the lowest energy. We evaluate the quality of the generated molecules with two stability metrics: **Atom stability** is the percentage of the atoms with valid valency. **Molecule stability** is the percentage of the generated molecules that consist of stable atoms. We provide more details in Appendix C.3.

Baselines We compare GruM against 3D molecule generative models: **G-Schnet** (Gebauer et al., 2019) is an autoregressive model based on the 3d point sets. **EN-Flow** (Satorras et al., 2021) is a flow-based model. **GDM** and **EDM** (Hoogeboom et al., 2022) are denoising diffusion models. **Bridge** (Wu et al., 2022) is a diffusion model based on the diffusion mixture that learns to approximate the drift and **Bridge+Force** (Wu et al., 2022) adds physical force to the drift. For ours, we follow the training setting of Hoogeboom et al. (2022) using the same architecture of EGNN (Satorras et al., 2021). We describe further implementation details in Appendix C.3.

Results As shown in the table of Figure 3, our method yields the highest atom stability compared to all the baselines on both datasets. Furthermore, ours achieves higher molecule stability since we directly model the topology by learning the graph mixture. Moreover, GruM outperforms Bridge+Force (Wu et al., 2022) even though GruM does not require task-dependent prior force while trained in a simulation-free manner. Notably, our method achieves non-zero molecule stability in the GEOM-DRUGS dataset consisting of large molecules with up to 181 atoms. We visualize the generated molecules and the generative process of GruM in Appendix E, demonstrating that we can predict the final molecule at an early stage of the process leading to stable molecules. We further observe that GruM generates $\times 1.5$ more number of connected molecules compared to EDM as shown in Table 8 of the Appendix.

Stability Analysis To further investigate the superior performance of our framework in generating more stable molecules, we conduct an analysis of the convergence and stability. Figure 3 (Right) shows the convergence of the predicted graph from GruM and the implicit prediction from EDM computed from the estimated noise. We observe that for GruM, the predicted graphs converge rapidly to the final result. After the convergence, the stability of GruM increases as it has sufficient steps to calibrate the details to produce valid molecules, which is visualized in the generative process of Figure 19 of the Appendix. As for EDM, the implicit predictions converge slowly since EDM does not explicitly learn the information of the final result, which leads to lower stability. This analysis shows that learning the final graph is significantly superior in capturing the correct topology compared to previous diffusion models.

4.4. Further Analysis

We conduct an analysis to investigate the advantages of our framework explained in Section 3.2.

Exploiting Inductive Bias To validate that exploiting the inductive bias of the graph data is critical, we compare GruM against a variant of it without an additional function at the last layer in the model. Figure 2 (Right) shows the complexity of the models s_θ trained on the Planar dataset, where the transformation at the last layer significantly reduces the model complexity for predicting the final graph. Especially, the larger complexity gap at the late stage of the diffusion process suggests that exploiting the inductive bias is crucial for learning valid structures and their topology.

Comparison with Learning Drift To verify that learning the graph mixture as in our framework is superior to learning the drift, we compare with Bridge (Wu et al., 2022) which models the drift of the mixture process. Table 3 shows that ours outperforms Bridge, especially for the molecule stability, since learning the drift is challenging due to its diverging nature and unable to model the topology directly. We further validate that learning the drift performs poorly on general graph generation tasks and fails to generate the correct topology in Appendix D.2.

Early stopping for the generative process In Figure 2 (Left) and (Middle), the V.U.N. and the MMD results of DruM in the Planar dataset demonstrate that the estimated destination mixture converges to the exact destination at early sampling steps, accurately capturing both the global topology and local graph characteristics. This allows us to early-stop the diffusion process, which reduces the generation time by up to 20% on this task. The generation results on SBM and Proteins datasets in Section D.2 of the Appendix show a similar tendency.

5. Conclusion

In this work, we proposed a new diffusion-based graph generation framework, GruM, that explicitly models the topology of the graphs. Unlike existing graph diffusion models that learn to denoise, our framework learns to predict the final graph of the generative process through the graph mixture, thereby accurately capturing the valid graph structure and its topological features. Specifically, we construct the generation process as a mixture of diffusion bridges, which differs from the denoising diffusion process, where the drift drives the generation process toward the predicted graph that converges in an early stage. We extensively validated our framework on diverse graph generation tasks, including 2D/3D molecular generation, on which ours significantly outperforms previous graph generation methods. A promising future direction would be the generalization to domains other than graphs where the topology of the data is important, such as proteins and manifolds.

Impact Statement This paper presents work whose goal is to advance the field of deep generative models, specifically for graph-structured data. We believe that our work can accelerate the discovery of feasible drugs and improve the quality of human lives by recommending drug candidates in silico, which reduces time-consuming wet lab experiments performed by experts. However, one might maliciously use our framework to generate toxic substances or narcotics harmful to humans or the environment.

Acknowledgement This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075 Artificial Intelligence Graduate School Program(KAIST)), Google Research Grant and Google Cloud Research Credits program with the award (XKCV-N0JU-8K3R-65LK, 40KR-GJX5-XH4X-PU3L), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00713), and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00256259).

References

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv:2303.08797*, 2023.

Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing System*, 2021.

Axelrod, S. and Gomez-Bombarelli, R. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):1–14, 2022.

Bemis, G. W. and Murcko, M. A. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

Bortoli, V. D., Doucet, A., Heng, J., and Thornton, J. Simulating diffusion bridges with score matching. *arXiv:2111.07243*, 2021a.

Bortoli, V. D., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. In *Advances in Neural Information Processing Systems*, 2021b.

Brigo, D. The general mixture-diffusion sde and its relationship with an uncertain-volatility option model with volatility-asset decorrelation. *arXiv:0812.4052*, 2008.

Brockschmidt, M., Allamanis, M., Gaunt, A. L., and Polozov, O. Generative code modeling with graphs. In *International Conference on Learning Representations*, 2019.

Campbell, A., Benton, J., De Bortoli, V., Rainforth, T., Deligiannidis, G., and Doucet, A. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 2022.

Chen, T., Liu, G., and Theodorou, E. A. Likelihood training of schrödinger bridge using forward-backward sdes theory. In *International Conference on Learning Representations*, 2022.

Corlay, S. Properties of the ornstein-uhlenbeck bridge. *arXiv preprint arXiv:1310.5617*, 2013.

Costa, F. and De Grave, K. Fast neighborhood subgraph pairwise distance kernel. In *International Conference on Machine Learning*, 2010.

De Cao, N. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *arXiv:2105.05233*, 2021.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

Doob, J. L. and Doob, J. *Classical potential theory and its probabilistic counterpart*, volume 549. Springer, 1984.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv:2012.09699*, 2020.

Franzese, G., Rossi, S., Yang, L., Finamore, A., Rossi, D., Filippone, M., and Michiardi, P. How much is enough? a study on diffusion times in score-based generative models. *Entropy*, 25(4):633, 2023.

Gebauer, N. W. A., Gastegger, M., and Schütt, K. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. In *Advances in Neural Information Processing Systems*, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Ho, J., Salimans, T., Gritsenko, A. A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *arXiv:2204.03458*, 2022.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, 2021.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, 2022.

Ingraham, J., Garg, V. K., Barzilay, R., and Jaakkola, T. S. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems*, 2019.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Jin, W., Barzilay, R., and Jaakkola, T. S. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, 2018.

Jo, J. and Hwang, S. J. Generative modeling on manifolds through mixture of riemannian diffusion processes. *arxiv:2310.07216*, 2023.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, 2022.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in Neural Information Processing Systems*, 2021.

Landrum, G. et al. Rdkit: Open-source cheminformatics software, 2016. *URL http://www. rdkit. org/, https://github. com/rdkit/rdkit*, 2016.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. *Advances in Neural Information Processing Systems*, 2019.

Liu, X., Wu, L., Ye, M., and Liu, Q. Let us build bridges: Understanding and extending diffusion generative models. *arXiv:2208.14699*, 2022.

Liu, X., Wu, L., Ye, M., and Liu, Q. Learning diffusion bridges on constrained domains. In *International Conference on Learning Representations*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

Luo, Y., Yan, K., and Ji, S. Graphdf: A discrete flow model for molecular graph generation. *International Conference on Machine Learning*, 2021.

Martinkus, K., Loukas, A., Perraudin, N., and Wattenhofer, R. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, 2022.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 2019.

Peluchetti, S. Non-denoising forward-time diffusions. *Openreview*, 2021.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,*, pp. 3942–3951. AAAI Press, 2018.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, S. K. S., Lopes, R. G., Ayan, B. K., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 2022.

Satorras, V. G., Hoogeboom, E., Fuchs, F., Posner, I., and Welling, M. E(n) equivariant normalizing flows. In *Advances in Neural Information Processing Systems*, 2021.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*, 2018.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Särkkä, S. and Solin, A. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2019.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

Wu, L., Gong, C., Liu, X., Ye, M., and Liu, Q. Diffusion-based molecule generation with informative prior bridges. In *Advances in Neural Information Processing Systems*, 2022.

Ye, M., Wu, L., and Liu, Q. First hitting diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, 2018.

Zang, C. and Wang, F. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 617–626, 2020.

Øksendal, B. *Stochastic Differential Equations*. Universitext. Springer Berlin Heidelberg, 2003.

Appendix

Organization The Appendix is organized as follows: In Section A, we provide the derivations of the results from the main paper. In Section B, we explain the details of our generative framework including the training objectives, the sampling method, and the model architectures. In Section C, we provide experimental details for the generation tasks and further present additional experimental results in Section D. In Section E, we visualize the generated graphs and molecules, with visualized generative processes. Finally, in Section F, we discuss the limitations of our work.

A. Derivations

A.1. Diffusion bridge processes

Here we derive the Ornstein-Uhlenbeck (OU) bridge process using Doob's h-transform (Doob & Doob, 1984) and show that the Brownian bridge process is a special case of the OU bridge process. We further discuss a general class of bridge processes and explain the advantage of the OU bridge process.

Ornstein-Uhlenbeck bridge process First, we consider the simple case when the reference process is given as a standard OU process without a time-dependent diffusion coefficient:

$$\hat{\mathbb{Q}} \;:\; \mathrm{d}\boldsymbol{G}_t = \alpha \boldsymbol{G}_t \mathrm{d}t + \mathrm{d}\mathbf{W}_t, \tag{14}$$

where α is a constant. Then the Doob's h-transform on $\hat{\mathbb{Q}}$ yields the representation of an endpoint-conditioned process $\hat{\mathbb{Q}}^{\boldsymbol{g}} := \hat{\mathbb{Q}}(\cdot|\boldsymbol{G}_T = \boldsymbol{g})$ defined by the following SDE:

$$\hat{\mathbb{Q}}^{\boldsymbol{g}} \;:\; \mathrm{d}\boldsymbol{G}_t = \Big[\alpha\boldsymbol{G}_t + \nabla_{\boldsymbol{G}_t} \log \hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)\Big]\mathrm{d}t + \mathrm{d}\mathbf{W}_t, \tag{15}$$

where $\hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)$ is the transition probability from time t to T of the standard OU process in Eq. (14). Since the standard OU process has a linear drift, the transition probability is Gaussian, i.e. $\hat{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t) = \mathcal{N}(\boldsymbol{g}; \mu_t, \boldsymbol{\Sigma}_t)$, where the mean μ_t and the covariance $\boldsymbol{\Sigma}_t$ satisfies the following ODEs (derived from the results of Eq.(5.50) and Eq.(5.51) of Särkkä & Solin (2019)):

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}t} = \alpha\mu_t \;\;,\;\; \frac{\mathrm{d}\boldsymbol{\Sigma}_t}{\mathrm{d}t} = \mathbf{I} + 2\alpha\boldsymbol{\Sigma}_t. \tag{16}$$

The ODE with respect to $\boldsymbol{\Sigma}_t$ can be modified as:

$$\frac{\mathrm{d}}{\mathrm{d}t} e^{-2\alpha t}\boldsymbol{\Sigma}_t = e^{-2\alpha t}\mathbf{I}, \tag{17}$$

which give the following closed-form solutions:

$$\mu_t = \hat{u}_t \boldsymbol{G}_t \;\;,\;\; \boldsymbol{\Sigma}_t = \frac{1}{2\alpha}\left(\hat{u}_t^2 - 1\right)\mathbf{I} \;\text{ for }\; \hat{u}_t = e^{\alpha(T-t)}. \tag{18}$$

Therefore, the SDE representation of the standard OU bridge process with fixed endpoint $\boldsymbol{g}$ is given as follows:

$$\hat{\mathbb{Q}}^{\boldsymbol{g}} \;:\; \mathrm{d}\boldsymbol{G}_t = \left[\alpha\boldsymbol{G}_t + \frac{2\alpha}{1-\hat{u}_t^{-2}}\left(\frac{\boldsymbol{g}}{\hat{u}_t} - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \mathrm{d}\mathbf{W}_t. \tag{19}$$

Now we derive the bridge process for the general OU process with a time-dependent diffusion coefficient defined by the following SDE:

$$\mathbb{Q}\!: \;\mathrm{d}\boldsymbol{G}_t = \alpha\sigma_t^2\boldsymbol{G}_t\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t, \tag{20}$$

where σ_t is a scalar function. Since the time change (Section 8.5. of Øksendal (2003)) with $\beta_t = \int_0^t \sigma_\tau^2 \mathrm{d}\tau$ of $\hat{\mathbb{Q}}$ in Eq. (14) is equivalent to $\mathbb{Q}$ of Eq. (20), the transition probability $\tilde{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t)$ of the general OU process satisfies the following:

$$\tilde{p}_{T|t}(\boldsymbol{g}|\boldsymbol{G}_t) = \hat{p}_{\beta_T|\beta_t}(\boldsymbol{g}|\boldsymbol{G}_t) \tag{21}$$

Thereby, the OU bridge process conditioned on the endpoint $\boldsymbol{g}$ is defined by the following SDE:

$$\mathbb{Q}^{\boldsymbol{g}} \;:\; \mathrm{d}\boldsymbol{G}_t = \left[\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{22}$$

where the scalar function u_t and v_t are given as:

$$u_t = e^{\alpha(\beta_T - \beta_t)} = \exp\left(\alpha\int_t^T \sigma_\tau^2 \mathrm{d}\tau\right) \;,\; v_t = \frac{1}{2\alpha}(1 - u_t^{-2}). \tag{23}$$

Note that the OU bridge process, also known as the constrained OU process, was studied theoretically in previous works (Corlay, 2013; Peluchetti, 2021; Bortoli et al., 2021a). However, we are the first to validate the effectiveness of the OU bridge processes for modeling the generative process through extensive experiments, especially for the generation of graphs in diverse tasks including the generation of general graphs as well as 2D and 3D molecular graphs.

Brownian bridge process We show that the Brownian bridge process is a special case of the OU bridge process. When the constant α of the OU bridge process approaches 0, the scalar function u_t converges to 1 that leads to the convergence of v_t as follows:

$$v_t = \frac{1}{2\alpha}(1 - u_t^{-2}) = \frac{1}{2\alpha}\left(1 - e^{-2\alpha(\beta_T - \beta_t)}\right) \to \beta_T - \beta_t,$$

which is due to the Taylor expansion of the exponential function. Therefore, the OU bridge process for $\alpha \to 0$ is modeled by the following SDE:

$$\mathbb{Q}^{\boldsymbol{g}}_{bb} \;:\; \mathrm{d}\boldsymbol{G}_t = \frac{\sigma_t^2}{\beta_T - \beta_t}(\boldsymbol{g} - \boldsymbol{G}_t)\,\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{24}$$

which is equivalent to the SDE representation of the Brownian bridge process. Compared to the OU bridge process in Eq. (22), the Brownian bridge process has a simpler SDE representation with less flexibility for designing the generative process as the process is solely determined by the noise schedule σ_t.

Note that the Brownian bridge is an endpoint-conditioned process with respect to a reference Brownian Motion defined by the following SDE:

$$\mathrm{d}\boldsymbol{G}_t = \sigma_t \mathrm{d}\mathbf{W}_t, \tag{25}$$

which is a diffusion process without drift, and also a special case of the OU process that is used for the reference process of the OU bridge process.

More bridge processes Wu et al. (2022) proposes an approach for designing a more general class of diffusion bridges using the Lyapunov function method. Starting from a simple Brownian bridge $\mathbb{Q}^{\boldsymbol{g}}_{bb}$, we can create a new bridge process by adding an extra drift term as follows:

$$\mathbb{Q}^{\boldsymbol{g}}_{bb,f} \;:\mathrm{d}\boldsymbol{G}_t = \left[\underbrace{\sigma_t f_t(\boldsymbol{G}_t)}_{\text{extra drift}} + \frac{\sigma_t^2}{\beta_T - \beta_t}(\boldsymbol{g} - \boldsymbol{G}_t)\right]\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t, \tag{26}$$

$$\text{for } \; f_t \; \text{ satisfying } \quad \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\boldsymbol{g}}_{bb,f}}[\|f_t(\boldsymbol{G}_t)\|^2] < \infty. \tag{27}$$

$\mathbb{Q}^{\boldsymbol{g}}_{bb,f}$ of Eq. (26) is still a bridge process with endpoint $\boldsymbol{g}$ since the drift of the Brownian bridge (i.e. Eq. (24)) dominates the extra drift term due to the condition of Eq. (27). Moreover, Wu et al. (2022) introduces problem-dependent prior f inspired by physical energy functions.

These general bridge processes could be used for our framework to construct a mixture process for modeling the generative process, as described in Section 3.1. If the explicit SDE representation for the general bridges is accessible, the mixture process can be represented by leveraging the diffusion mixture representation, and further the Brownian bridge could be replaced with the OU bridge process.

However, in contrast to constructing the generative process as a mixture of the OU bridge processes, using the mixture of the general bridge processes results in difficulty during training; Training a generative model that approximates the mixture of the general bridge processes requires expensive SDE simulation due to the intractable transition probability. We show through extensive experiments that for our approach, the family of OU bridge processes is sufficient to model the complex generation process while the generative model can be trained in a simulation-free manner.

A.2. Diffusion mixture representation

In this section, we provide the formal definition of the diffusion mixture representation (Brigo, 2008; Peluchetti, 2021). Consider a collection of diffusion processes $\{\mathbb{Q}^\lambda : \lambda \in \Lambda\}$ defined by the SDEs:

$$\mathbb{Q}^\lambda : \mathrm{d}\boldsymbol{Z}^\lambda_t = \eta^\lambda(\boldsymbol{Z}_t, t)\mathrm{d}t + \sigma^\lambda_t \mathrm{d}\mathbf{W}^\lambda_t \;\;,\;\; \boldsymbol{Z}^\lambda_0 \sim p^\lambda_0 \tag{28}$$

where $\mathbf{W}^\lambda_t$ are independent standard Wiener processes and p^λ_0 are the initial distributions. Denote p^λ_t as the marginal density of the process $\mathbb{Q}^\lambda$. Further, define the mixture of marginal densities and the mixture of initial distributions with respect to a mixing distribution $\mathcal{L}$ on the collection Λ as follows:

$$p_t(z) = \int_\Lambda p^\lambda_t(z)\mathcal{L}(\mathrm{d}\lambda) \;\;,\;\; p_0(z) = \int_\Lambda p^\lambda_0(z)\mathcal{L}(\mathrm{d}\lambda), \tag{29}$$

Then there exists a diffusion process that induces a marginal density p_t, and the diffusion process is modeled by the following SDE:

$$\mathbb{Q}^\mathcal{L} : \mathrm{d}\boldsymbol{Z}_t = \eta(\boldsymbol{Z}_t, t)\mathrm{d}t + \sigma_t \mathrm{d}\mathbf{W}_t \;\;,\;\; \boldsymbol{Z}_0 \sim p_0, \tag{30}$$

where the drift and diffusion coefficients are given as the weighted mean of the corresponding coefficients of $\mathbb{Q}^\lambda$ as follows:

$$\eta(z,t) = \int_\Lambda \eta^\lambda(z,t)\frac{p^\lambda_t(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda) \;\;,\;\; \sigma^2_t = \int_\Lambda (\sigma^\lambda_t)^2 \frac{p^\lambda_t(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda). \tag{31}$$

Below, we provide a proof of this statement.

proof. It is enough to show that p_t defined in Eq. (29) is the solution to the corresponding Fokker-Planck equation of Eq. (30), which is given as follows:

$$\frac{\partial q_t(z)}{\partial t} = -\nabla_z \cdot \left(q_t(z)\eta(z,t) - \frac{1}{2}\sigma^2_t \nabla_z q_t(z) \right), \tag{32}$$

where q_t denotes the marginal density of Eq. (30). Using the definition of Eq. (29) and the corresponding Fokker-Planck equations with respect to $\mathbb{Q}^\lambda$ for $\lambda \in \Lambda$, we derive the following result:

$$\begin{aligned}
\frac{\partial p_t(z)}{\partial t} &= \frac{\partial}{\partial t}\int_\Lambda p^\lambda_t(z)\mathcal{L}(\mathrm{d}\lambda) = \int_\Lambda \frac{\partial}{\partial t} p^\lambda_t(z)\mathcal{L}(\mathrm{d}\lambda) \\
&= \int_\Lambda \left[-\nabla_z \cdot \left(\eta^\lambda(z,t)p^\lambda_t(z) - \frac{1}{2}(\sigma^\lambda_t)^2 \nabla_z p^\lambda_t(z) \right) \right] \mathcal{L}(\mathrm{d}\lambda) \\
&= -\nabla_z \cdot \int_\Lambda \left[\eta^\lambda(z,t)p^\lambda_t(z) - \frac{1}{2}(\sigma^\lambda_t)^2 \nabla_z p^\lambda_t(z) \right] \mathcal{L}(\mathrm{d}\lambda) \\
&= -\nabla_z \cdot \left(p_t(z)\int_\Lambda \eta^\lambda(z,t)\frac{p^\lambda_t(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda) - \frac{1}{2}\nabla_z \left[p_t(z)\int_\Lambda (\sigma^\lambda_t)^2 \frac{p^\lambda_t(z)}{p_t(z)}\mathcal{L}(\mathrm{d}\lambda) \right] \right) \\
&= -\nabla_z \cdot \left(p_t(z)\eta(z,t) - \frac{1}{2}\sigma^2_t \nabla_z p_t(z) \right),
\end{aligned} \tag{33}$$

which proves that p_t is the solution to the Fokker-Planck equation of Eq. (32).

A.3. OU bridge mixture

Now we use the diffusion mixture representation described in Appendix A.2 to derive the OU bridge mixture. Consider a mixture of the collection of OU bridge processes with endpoints in the data distribution, i.e. $\{\mathbb{Q}^{\boldsymbol{g}} : \boldsymbol{g} \sim \Pi^*\}$. We mix this collection of processes with the data distribution Π^* as the mixing distribution, which is represented by the following SDE:

$$\begin{aligned}\mathbb{Q}^{\Pi^*} : \mathrm{d}\boldsymbol{G}_t &= \left[\int \left(\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{\boldsymbol{g}}{u_t} - \boldsymbol{G}_t\right)\right)\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g})\right]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t \\ &= \left[\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}) - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t \\ &= \left[\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t \end{aligned} \tag{34}$$

where $p_t(z) = \int p_t^{\boldsymbol{g}}(z)\Pi^*(\mathrm{d}\boldsymbol{g})$ is used for the second equality and the definition of the graph mixture (Eq. (9)) is used for the last equality.

A.4. Reverse-time diffusion process of the OU bridge mixture

Here we derive the reverse-time diffusion process of GruM, i.e. the time reversal of the OU bridge mixture. Since the generative process of GruM transports the prior distribution Γ to the data distribution Π^*, the time reversal of GruM transports Π^* to Γ. We show that it has a similar SDE representation as Eq. (34).

We derive the reverse process of the OU bridge mixture by constructing a mixture of the reverse processes of each OU bridge process. To be precise, for the mixture process $\mathbb{Q} := \int \mathbb{Q}^{\boldsymbol{g}}\mathrm{d}\Pi^*$, the reverse process of $\mathbb{Q}$ denoted as $\overline{\mathbb{Q}}$ is equal to the mixture process $\int \overline{\mathbb{Q}}^{\boldsymbol{x}}\mathrm{d}\Gamma$ where $\overline{\mathbb{Q}}^{\boldsymbol{x}}$ is the reverse process of the bridge process $\mathbb{Q}^{\boldsymbol{g}}$ with starting point $\boldsymbol{x}$. For the simplicity of the representation, we first derive the time-reversal of general bridge processes, where the reference process is given as

$$\mathrm{d}\boldsymbol{G}_t^{ref} = \mu(\boldsymbol{G}_t^{ref}, t) + \sigma_t\mathbf{W}_t, \tag{35}$$

with the marginal density denoted as $\tilde{p}_t$. In order to obtain the reverse-time diffusion process, we leverage the reverse-time SDE representation (Anderson, 1982; Song et al., 2021) as follows:

$$\mathrm{d}\overline{\boldsymbol{G}}_t^{ref} = \Big[-\mu(\overline{\boldsymbol{G}}_t^{ref}, T{-}t) + \sigma_{T-t}^2\nabla_{\overline{\boldsymbol{G}}_t^{ref}}\log\tilde{q}_t(\overline{\boldsymbol{G}}^{ref})\Big]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t, \tag{36}$$

where $\tilde{q}_t = \tilde{p}_{T-t}$ is the marginal density of the process $\{\overline{\boldsymbol{G}}_t^{ref}\}_{t\in[0,T]}$. Then the bridge process of Eq. (36) with fixed end point $\boldsymbol{x} \sim \Gamma$ can be derived by using the Doob's h-transform (Doob & Doob, 1984) as follows:

$$\overline{\mathbb{Q}}^{\boldsymbol{x}} : \mathrm{d}\overline{\boldsymbol{G}}_t = \Big[-\mu(\overline{\boldsymbol{G}}_t, T{-}t) + \sigma_{T-t}^2\nabla_{\overline{\boldsymbol{G}}_t}\log\tilde{q}_t(\overline{\boldsymbol{G}}_t) + \sigma_{T-t}^2\nabla_{\overline{\boldsymbol{G}}_t}\log\tilde{q}_{T|t}(\boldsymbol{x}|\overline{\boldsymbol{G}}_t)\Big]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t, \tag{37}$$

which is a reverse process for the conditioned process of $\boldsymbol{G}^{ref}$ with starting point $\boldsymbol{x}$ and endpoint $\boldsymbol{g} \sim \Pi^*$ fixed. Here using the fact that $\tilde{q}_t = \tilde{p}_{T-t}$, we can see that

$$\tilde{q}_t(y)\tilde{q}_{T|t}(x|y) = \tilde{q}(\overline{\boldsymbol{G}}_T{=}x, \overline{\boldsymbol{G}}_t{=}y) = \frac{\tilde{q}(\overline{\boldsymbol{G}}_T{=}x, \overline{\boldsymbol{G}}_t{=}y)}{\tilde{q}_T(x)}\tilde{q}_T(x) = \tilde{p}_{T-t|0}(y|x)\tilde{q}_T(x), \tag{38}$$

and since $\nabla_{\overline{\boldsymbol{G}}_t}\log\tilde{q}_T(\boldsymbol{x}) = 0$ for fixed $\boldsymbol{x}$, Eq. (37) can be simplified as follows:

$$\overline{\mathbb{Q}}^{\boldsymbol{x}} : \mathrm{d}\overline{\boldsymbol{G}}_t = \Big[-\mu(\overline{\boldsymbol{G}}_t, T{-}t) + \sigma_{T-t}^2\nabla_{\overline{\boldsymbol{G}}_t}\log\tilde{p}_{T-t|0}(\overline{\boldsymbol{G}}_t|\boldsymbol{x})\Big]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t. \tag{39}$$

Finally, the mixture of the bridge processes $\{\overline{\mathbb{Q}}^{\boldsymbol{x}} : \boldsymbol{x} \sim \Gamma\}$ can be derived using the diffusion mixture representation as follows:

$$\overline{\mathbb{Q}} : \mathrm{d}\overline{\boldsymbol{G}}_t = \left[-\mu(\overline{\boldsymbol{G}}_t, t) + \sigma_{T-t}^2\int\nabla_{\overline{\boldsymbol{G}}_t}\log\tilde{p}_{T-t|0}(\overline{\boldsymbol{G}}_t|\boldsymbol{x})\frac{q_t^{\boldsymbol{x}}(\overline{\boldsymbol{G}}_t)}{q_t(\overline{\boldsymbol{G}}_t)}\Gamma(\mathrm{d}\boldsymbol{x})\right]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t, \tag{40}$$

where $q_t^{\boldsymbol{x}}$ is the marginal density of $\overline{\mathbb{Q}}^{\boldsymbol{x}}$ and q_t is the marginal density of the mixture process $\overline{\mathbb{Q}}$ defined as $q_t(\cdot) := \int q_t^{\boldsymbol{x}}(\cdot)\Gamma(\mathrm{d}\boldsymbol{x})$.

Using the result of Eq. (40), now we can derive the time reversal of the OU bridge mixture by setting $\mu(z,t) = \alpha\sigma_t^2 z$. Since the transition distributions of the OU process satisfy the following (we provide closed-form mean and covariance of the transition distribution in Eq. (56)):

$$\tilde{p}_{T-t|0}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z};\ \bar{u}_t\boldsymbol{x},\ \bar{u}_t^2\bar{v}_t\mathbf{I}\right) \quad \text{for} \quad \bar{u}_t = \exp\left(\alpha\int_0^{T-t}\sigma_\tau^2 \mathrm{d}\tau\right),\ \bar{v}_t = \frac{1}{2\alpha}\left(1-\bar{u}_t^{-2}\right), \tag{41}$$

the log gradient of the transition distribution can be computed as follows:

$$\nabla_{\boldsymbol{z}} \log \tilde{p}_{T-t|0}(\boldsymbol{z}|\boldsymbol{x}) = -\frac{1}{\bar{u}_t^2\bar{v}_t}\left(\boldsymbol{z} - \bar{u}_t\boldsymbol{x}\right). \tag{42}$$

Thereby, the reverse-time diffusion process of the OU bridge mixture is given by:

$$\overline{\mathbb{Q}} : \mathrm{d}\overline{\boldsymbol{G}}_t = \left[-\alpha\sigma_{T-t}^2\overline{\boldsymbol{G}}_t + \frac{\sigma_{T-t}^2}{\bar{u}_t^2\bar{v}_t}\left(\bar{u}_t\boldsymbol{D}^\Gamma(\overline{\boldsymbol{G}}_t,t) - \overline{\boldsymbol{G}}_t\right)\right]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t, \quad \overline{\boldsymbol{G}}_0 \sim \Pi^*, \tag{43}$$

where $\boldsymbol{D}^\Gamma(\cdot,t)$ is the graph mixture of $\overline{\mathbb{Q}}$ defined as follows:

$$\boldsymbol{D}^\Gamma(\overline{\boldsymbol{G}}_t,t) = \int \boldsymbol{x}\frac{q_t^{\boldsymbol{x}}(\overline{\boldsymbol{G}}_t)}{q_t(\overline{\boldsymbol{G}}_t)}\Gamma(\mathrm{d}\boldsymbol{x}). \tag{44}$$

Since $\overline{\mathbb{Q}}$ describes the diffusion process from the data distribution to the prior distribution, it can be considered a perturbation process. Further, we can observe that the time reversal of the OU bridge mixture is non-linear with respect to $\overline{\boldsymbol{G}}_t$ in general, and completely different from the forward process (i.e. perturbation process) of denoising diffusion models, i.e. the VESDE or VPSDE (Song et al., 2021).

Note that the reverse process of the OU bridge mixture perfectly transports the data distribution Π^* to the arbitrary prior distribution Γ in the sense that the terminal distribution exactly matches Γ for finite terminal time T. On the other hand, the forward process of denoising diffusion models, for example, VPSDE (Song et al., 2021), does not perfectly transport the data distribution to the prior distribution. The terminal distribution of the forward process is approximately Gaussian but not exactly a Gaussian distribution for finite T, although the mismatch is small for sufficiently large T. This is because the forward process requires infinite T in order to decouple the prior distribution Γ from the data distribution Π^*.

In conclusion, the generative process of GruM is different from denoising diffusion models which naturally follows from the fact that the time reversal of the OU bridge mixture is different from the forward processes of denoising diffusion models.

A.5. Derivation of the graph mixture matching objective

We provide the derivation of our graph mixture matching objective, corresponding to Eq. (11). First, we leverage the Girsanov theorem (Øksendal, 2003) to upper bound the KL divergence between the data distribution Π^* and the terminal distribution of $\mathbb{P}^\theta$ denoted as p_T^θ:

$$D_{KL}(\Pi^*\|p_T^\theta) \le D_{KL}(\mathbb{Q}^{\Pi^*}\|\mathbb{P}^\theta) \tag{45}$$

$$= D_{KL}(\mathbb{Q}_0^{\Pi^*}\|\mathbb{P}_0^\theta) + \mathbb{E}_{\mathbb{Q}^{\Pi^*}}\left[\log\frac{\mathrm{d}\mathbb{Q}^{\Pi^*}}{\mathrm{d}\mathbb{P}^\theta}\right] \tag{46}$$

$$= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[-\log p_0^\theta(\boldsymbol{G}_0) + \frac{1}{2}\int_0^T \left\|\sigma_t^{-1}\left(\eta_\theta(\boldsymbol{G}_t,t) - \eta(\boldsymbol{G}_t,t)\right)\right\|^2 \mathrm{d}t\right] + C \tag{47}$$

$$= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[-\log p_0^\theta(\boldsymbol{G}_0) + \frac{1}{2}\int_0^T \gamma_t^2\left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\right\|^2 \mathrm{d}t\right] + C, \tag{48}$$

where p_0^θ is a predetermined prior distribution that is easy to sample from, for instance, Gaussian distribution, and C is a constant independent of θ. Note that the first inequality is known as the data processing inequality. The expectation in Eq. (48) corresponds to Eq. (11).

Furthermore, the expectation of Eq. (48) can be written as follows:

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\right\|^2 \mathrm{d}t\right] \\
&= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \Big(\boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\Big) + \Big(\boldsymbol{G}_T - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\Big)\right\|^2 \mathrm{d}t\right] \\
&= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\right\|^2 \mathrm{d}t\right] + \mathcal{E} + \mathcal{E}^T + C_1, \qquad (49)
\end{aligned}
$$

where $\mathcal{E}$ and C_1 are defined as:

$$
\begin{aligned}
\mathcal{E} &= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \Big(\boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\Big)^T \Big(\boldsymbol{G}_T - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\Big)\mathrm{d}t\right], \\
C_1 &= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\|\boldsymbol{G}_T - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\right\|^2 \mathrm{d}t\right].
\end{aligned} \qquad (50)
$$

From the definition of the graph mixture (Eq. (9)), the following identity holds for all $t \in [0,T]$:

$$
\mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}} \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) = \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}} \boldsymbol{G}_T, \qquad (51)
$$

which gives the following result:

$$
\begin{aligned}
\mathcal{E} &= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \Big(\boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\Big)^T \Big(\boldsymbol{G}_T - \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t)\Big)\mathrm{d}t\right] \qquad (52) \\
&= \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \Big(\boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\Big)^T \Big(\boldsymbol{G}_T - \boldsymbol{G}_T\Big)\mathrm{d}t\right] = 0 \qquad (53)
\end{aligned}
$$

Therefore, we can conclude that minimizing Eq. (48) is equivalent to minimizing the following loss:

$$
\mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_t^2 \left\| \boldsymbol{s}_\theta(\boldsymbol{G}_t,t) - \boldsymbol{G}_T\right\|^2 \mathrm{d}t\right] \qquad (54)
$$

which corresponds to the graph mixture matching presented in Eq. (11).

A.6. Analytical computation of graph mixture matching

In order to practically use the graph mixture matching (Eq. (11)), we provide the analytical form of the distribution $p_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$. Notice that by construction, the OU bridge mixture with a fixed starting point $\boldsymbol{G}_0$ and an endpoint $\boldsymbol{G}_T$ coincides with the reference OU process in Eq. (20) with a fixed starting point $\boldsymbol{G}_0$ and an endpoint $\boldsymbol{G}_T$. Thereby, $p_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$ is equal to $\tilde{p}_{t|0,T}(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$ where $\tilde{p}$ denotes the marginal probability of the reference OU process of Eq. (20). Using the Bayes theorem, we can derive the following:

$$
\tilde{p}(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T) = \frac{\tilde{p}(\boldsymbol{G}_t,\boldsymbol{G}_T|\boldsymbol{G}_0)}{\tilde{p}(\boldsymbol{G}_T|\boldsymbol{G}_0)} = \frac{\tilde{p}(\boldsymbol{G}_T|\boldsymbol{G}_t,\boldsymbol{G}_0)\,\tilde{p}(\boldsymbol{G}_t|\boldsymbol{G}_0)}{\tilde{p}(\boldsymbol{G}_T|\boldsymbol{G}_0)} = \frac{\tilde{p}(\boldsymbol{G}_T|\boldsymbol{G}_t)\,\tilde{p}(\boldsymbol{G}_t|\boldsymbol{G}_0)}{\tilde{p}(\boldsymbol{G}_T|\boldsymbol{G}_0)}, \qquad (55)
$$

where the last equality is due to the Markov property of the OU process. Note that the transition distributions of the reference OU process are Gaussian with the mean and the covariance as follows:

$$
\begin{aligned}
&\tilde{p}_{b|a}(\boldsymbol{G}_b|\boldsymbol{G}_a) = \mathcal{N}(\boldsymbol{G}_b; u_{b|a}\boldsymbol{G}_a, u_{b|a}^2 v_{b|a}\mathbf{I}) \;\; \text{for} \;\; 0 \le a < b \le T, \\
&\text{where} \;\; u_{b|a} := \exp\left(\alpha\int_a^b \sigma_\tau^2 \mathrm{d}\tau\right) \;\; , \;\; v_{b|a} := \frac{1}{2\alpha}\left(1 - u_{b|a}^{-2}\right).
\end{aligned} \qquad (56)
$$

Therefore, the distribution $p(\boldsymbol{G}_t|\boldsymbol{G}_0,\boldsymbol{G}_T)$ is also Gaussian resulting from the product of Gaussian distributions, where the mean μ_t^* and the covariance $\boldsymbol{\Sigma}_t^*$ have analytical forms as follows:

$$
\mu_t^* = \frac{v_{T|t}}{u_{t|0}v_{T|0}}\boldsymbol{G}_0 + \frac{v_{t|0}}{u_{T|t}v_{T|0}}\boldsymbol{G}_T \;\; , \;\; \boldsymbol{\Sigma}_t^* = \frac{v_{T|t}v_{t|0}}{v_{T|0}}\mathbf{I}. \qquad (57)
$$

The mean and the covariance can be simplified by using the hyperbolic sine function as follows:

$$\mu_t^* = \frac{\sinh(\varphi_T - \varphi_t)}{\sinh(\varphi_T)} \boldsymbol{G}_0 + \frac{\sinh(\varphi_t)}{\sinh(\varphi_T)} \boldsymbol{G}_T, \quad \boldsymbol{\Sigma}_t^* = \frac{1}{\alpha} \frac{\sinh(\varphi_T - \varphi_t)\sinh(\varphi_t)}{\sinh(\varphi_T)} \mathbf{I}, \tag{58}$$

where $\varphi_t \coloneqq \alpha\beta_t = \alpha \int_0^t \sigma_\tau^2 \mathrm{d}\tau$.

A.7. GruM as a stochastic interpolant

Recently, Albergo et al. (2023) introduced the concept of *stochastic interpolant* which unifies the framework for diffusion models from the perspective of continuous-time stochastic processes.

From the results of Eq. (58), we can represent the OU bridge mixture as a stochastic interpolant between the distributions Γ and Π^* as follows:

$$\boldsymbol{G}_t = \frac{\sinh(\varphi_T - \varphi_t)}{\sinh(\varphi_T)} \boldsymbol{G}_0 + \frac{\sinh(\varphi_t)}{\sinh(\varphi_T)} \boldsymbol{G}_T + \left(\frac{1}{\alpha} \frac{\sinh(\varphi_T - \varphi_t)\sinh(\varphi_t)}{\sinh(\varphi_T)} \right)^{1/2} \boldsymbol{Z}. \tag{59}$$

where $\boldsymbol{G}_0$, $\boldsymbol{G}_T$, and $\boldsymbol{Z}$ are random variables sampled independently from the distributions Γ, Π^*, and $\mathcal{N}(\mathbf{0}, \mathbf{I})$, respectively. Eq. (59) shows that $\boldsymbol{G}_t$ is linear in both the starting point $\boldsymbol{G}_0 \sim \Gamma$ and the endpoint $\boldsymbol{G}_T \sim \Pi^*$. Note that our proposed graph mixture matching is different from the loss introduced in Albergo et al. (2023), as graph mixture matching does not require estimation of the score function. Additionally, we further derive the score function of our GruM in Section A.9.

A.8. Understanding the informative prior as regularizing the graph mixture

Wu et al. (2022) introduces incorporating prior information into the generative process, for example injecting physical and statistical information. To be specific, given a generative process:

$$\mathrm{d}\boldsymbol{G}_t = \eta(\boldsymbol{G}_t, t)\mathrm{d}t + \sigma_t \mathbf{W}_t,$$

Wu et al. (2022) modifies the drift by adding a prior function $f(\cdot, t)$ as follows:

$$\mathrm{d}\boldsymbol{G}_t = \Big(\underbrace{\sigma_t f(\boldsymbol{G}_t, t) + \eta(\boldsymbol{G}_t, t)}_{\eta_R(\boldsymbol{G}_t, t)} \Big) \mathrm{d}t + \sigma_t \mathbf{W}_t, \tag{60}$$

where $f(\cdot, t)$ is designed to be a force defined as $f(\cdot, t) = -\nabla \boldsymbol{E}(\cdot)$ where $E(\cdot)$ is a problem-dependent energy function. Although Wu et al. (2022) shows that incorporating prior information is beneficial for the generation of stable molecules or realistic 3D point clouds, how this modification leads to better performance was not fully explained.

Notably, from the perspective of our framework, we can interpret the incorporation of the prior information as modifying the generative path toward an energy-regularized result. To be precise, given a generative process modeled by the OU bridge mixture as in Eq. (34), adding the prior function $f(\cdot, t)$ to the drift can be written as follows:

$$\eta_R(\boldsymbol{G}_t, t) = \alpha\sigma_t^2 \boldsymbol{G}_t + \frac{\sigma_t^2}{v_t} \left[\frac{1}{u_t} \left(\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) + \frac{u_t v_t}{\sigma_t} f(\boldsymbol{G}_t, t) \right) - \boldsymbol{G}_t \right], \tag{61}$$

which is equivalent to regularizing the graph mixture with the weighted prior function as follows:

$$\boldsymbol{D}_R^{\Pi^*}(\boldsymbol{G}_t, t) \coloneqq \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) + \frac{u_t v_t}{\sigma_t} f(\boldsymbol{G}_t, t). \tag{62}$$

Since the weight of the prior function converges to 0 through the generative process:

$$\frac{u_t v_t}{\sigma_t} = \frac{\exp\left(\alpha \int_t^T \sigma_\tau^2 \mathrm{d}\tau\right) - \exp\left(-\alpha \int_t^T \sigma_\tau^2 \mathrm{d}\tau\right)}{2\alpha\sigma_t} \to 0 \quad \text{as} \quad t \to T,$$

we can see that $\boldsymbol{D}_R^{\Pi^*}$ converges to the original graph mixture $\boldsymbol{D}^{\Pi^*}$ where the convergence is determined by the prior function. By defining $f(\cdot, t) = -\nabla \boldsymbol{E}(\cdot)$ where $\boldsymbol{E}$ is an energy function, for example, potential energy for the 3D molecules or Riesz energy for the 3D point cloud, the regularized graph mixture has the following representation:

$$\boldsymbol{D}_R^{\Pi^*}(\boldsymbol{G}_t, t) = \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t, t) - \frac{u_t v_t}{\sigma_t} \nabla \boldsymbol{E}(\boldsymbol{G}_t). \tag{63}$$

Thereby, $\boldsymbol{D}_R^{\Pi^*}$ follows a path that minimizes the energy function $\boldsymbol{E}$ through the generative process. Therefore, the generative process is guided toward the regularized graph mixture which results in samples that achieve desired physical properties, for instance, stable 3D-structured molecules or point clouds.

A.9. Associated probability flow ODE of GruM

Since we have derived the reverse-time diffusion process of the OU bridge mixture in Section A.4, we can further derive its associated probability flow ODE (Song et al., 2021), i.e. a deterministic process that shares the same marginal density with the OU bridge mixture.

First, the OU bridge mixture is modeled by the following SDE:

$$\mathrm{d}\boldsymbol{G}_t = \left[\alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) - \boldsymbol{G}_t\right)\right]\mathrm{d}t + \sigma_t\mathrm{d}\mathbf{W}_t,$$

where the scalar functions u_t and v_t, and the graph mixture $\boldsymbol{D}^{\Pi^*}$ are defined as:

$$u_t = \exp\left(\alpha\int_t^T \sigma_\tau^2 \mathrm{d}\tau\right),\quad v_t = \frac{1}{2\alpha}(1-u_t^{-2}),\quad \boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) = \int \boldsymbol{g}\frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)}\Pi^*(\mathrm{d}\boldsymbol{g}).$$

Then using the results of Section A.4, the reverse-time diffusion process of the OU bridge mixture is modeled by the following SDE:

$$\mathrm{d}\overline{\boldsymbol{G}}_t = \left[-\alpha\sigma_{T-t}^2\overline{\boldsymbol{G}}_t + \frac{\sigma_{T-t}^2}{\bar{u}_t^2\bar{v}_t}\left(\bar{u}_t\boldsymbol{D}^{\Gamma}(\overline{\boldsymbol{G}}_t,t) - \overline{\boldsymbol{G}}_t\right)\right]\mathrm{d}t + \sigma_{T-t}\mathrm{d}\mathbf{W}_t,$$

where the scalar functions $\bar{u}_t$ and $\bar{v}_t$, and the reversed graph mixture $\boldsymbol{D}^{\Gamma}$ are defined as:

$$\bar{u}_t = \exp\left(\alpha\int_0^{T-t}\sigma_\tau^2\mathrm{d}\tau\right),\quad \bar{v}_t = \frac{1}{2\alpha}\left(1-\bar{u}_t^{-2}\right),\quad \boldsymbol{D}^{\Gamma}(\overline{\boldsymbol{G}}_t,t) = \int \boldsymbol{x}\frac{q_t^{\boldsymbol{x}}(\overline{\boldsymbol{G}}_t)}{q_t(\overline{\boldsymbol{G}}_t)}\Gamma(\mathrm{d}\boldsymbol{x}).$$

From the relation between the diffusion process and its reverse-time diffusion process (for instance, Eq. (35) and Eq. (36)), the score function of the OU bridge mixture can be computed as follows:

$$\nabla_{\boldsymbol{G}_t}\log p_t(\boldsymbol{G}_t) = \frac{1}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) - \boldsymbol{G}_t\right) + \frac{1}{\bar{u}_{T-t}^2\bar{v}_{T-t}}\left(\bar{u}_{T-t}\boldsymbol{D}^{\Gamma}(\boldsymbol{G}_t,T-t) - \boldsymbol{G}_t\right). \tag{64}$$

Therefore, the associated probability flow ODE can be derived as follows:

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{G}_t}{\mathrm{d}t} &= \alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) - \boldsymbol{G}_t\right) - \frac{1}{2}\sigma_t^2\nabla_{\boldsymbol{G}_t}\log p_t(\boldsymbol{G}_t) &(65)\\ &= \alpha\sigma_t^2\boldsymbol{G}_t + \frac{\sigma_t^2}{2v_t}\left(\frac{1}{u_t}\boldsymbol{D}^{\Pi^*}(\boldsymbol{G}_t,t) - \boldsymbol{G}_t\right) - \frac{\sigma_t^2}{2\bar{u}_{T-t}^2\bar{v}_{T-t}}\left(\bar{u}_{T-t}\boldsymbol{D}^{\Gamma}(\boldsymbol{G}_t,T-t) - \boldsymbol{G}_t\right). &(66)\end{aligned}$$

To practically use the probability flow ODE as a generative model, the graph mixtures $\boldsymbol{D}^{\Pi^*}(\cdot,t)$ and $\boldsymbol{D}^{\Gamma}(\cdot,t)$ should be approximated by the neural networks $\boldsymbol{s}_\theta(\cdot,t)$ and $\boldsymbol{s}_\phi(\cdot,t)$, respectively. $\boldsymbol{s}_\theta$ can be trained using the graph mixture matching (Eq. (54)). $\boldsymbol{s}_\phi$ also can be trained in a similar way where the trajectories are sampled from the reverse-time process of the OU bridge mixture.

In particular, from the result of Eq. (64), we can see that learning the score function of the mixture process is not interchangeable with learning the graph mixture since the score function additionally requires the knowledge of the reversed graph mixture $\boldsymbol{D}^{\Gamma}$. Our mixture process differs from the denoising diffusion processes for which learning the score function is equivalent to recovering clean data from its corrupted version (Kingma et al., 2021). The difference originates from the difference in the construction of the generative process, where denoising diffusion processes are derived by reversing the forward noising processes while our mixture process is built as a mixture of bridge processes without relying on the time-reversal approach. We further discuss the difference between our framework and the denoising diffusion models in Section A.10.

A.10. Comparison with Denoising Diffusion Models

Here we explain in detail the difference between our framework and previous denoising diffusion models.

Comparison of the generative processes The main difference with the denoising diffusion models (Ho et al., 2020; Song et al., 2021) is in the different generative processes. While denoising diffusion models derive the generative process by reversing the forward noising process, our method constructs the generative process using the mixture of OU bridge processes described in Eq. (3) which does not rely on the time-reversal approach. Due to the difference in the generative process, our method demonstrates two distinct properties: First, the mixture process defines an exact transport from an arbitrary prior distribution to the data distribution by construction. In contrast, denoising diffusion processes are not an exact transport to the data distribution since the forward noising processes require infinitely long diffusion time in order to guarantee convergence to the prior distribution (Franzese et al., 2023).

Furthermore, our framework does not suffer from the restrictions of denoising diffusion models. Denoising diffusion models require p_{prior} to be approximately independent of the data distribution Π^*, e.g. Gaussian, as the perturbation process decouples p_{prior} from Π^*, and further this decoupling requires infinitely long diffusion time T. On the contrary, our framework does not have any constraints on the prior distribution p_{prior} and does not require large T, since the OU bridge mixture can be defined between two arbitrary distributions for any $T > 0$, where its drift forces the process to the terminal distribution regardless of the initial distribution. Therefore, the OU bridge mixture provides flexibility for our generative framework in choosing the prior distribution and the finite terminal time while maintaining the generative process to be an exact transport from the prior to the data distribution.

Comparison of the training objectives We further compare our training objective in Eq. (54) with the training objectives of denoising diffusion models. First, we clarify that learning the graph mixture is not equivalent to learning the score function for the mixture process of GruM. As derived in Eq. (64) of Section A.9, the score function of the OU bridge mixture additionally requires the knowledge of the reversed graph mixture $\boldsymbol{D}^{\Gamma}$, thus learning the score function needs to predict not only the graph mixture but also the reversed graph mixture. In contrast, the training objectives of denoising diffusion models are interchangeable (Kingma et al., 2021), i.e., learning the score function of the denoising diffusion process is equivalent to recovering clean data from its corrupted version. This difference in the training objective originates from the difference in the generative process, which we have discussed in detail in the previous paragraphs.

Furthermore, our training objective differs from the objectives of previous works (Saharia et al., 2022) that aim to recover clean data from its corrupted version. While our method learns the graph mixture, i.e. the probable graph represented as the weighted mean of data, Saharia et al. (2022) aims to predict the exact final result which could be problematic as the prediction would be highly inaccurate in early steps which may lead the process in the wrong direction. It should be noted that the goal of Eq. (54) is to estimate the graph mixture, i.e. the weighted mean of data, not to predict the exact graph as in Saharia et al. (2022). This is because Eq. (54) is derived from Eq. (11) which minimizes the difference between our model prediction and the ground truth graph mixture. We emphasize that learning graph mixture (Eq. (11)) is only feasible for the OU bridge mixture and cannot be used for denoising diffusion models due to the difference in the generative processes.

In the perspective of the mathematical formulation of the training objective, Eq. (54) differs from the objective of Saharia et al. (2022) in two parts: (1) The computation of the expectation for the squared error loss term is different. The expectation is computed by sampling from the trajectory of the diffusion process, where our GruM uses the OU bridge mixture while previous works use the denoising diffusion process. These two processes are not the same and therefore result in different objectives. (2) The weight function in the loss is different. The weight function γ_t of Eq. (11) is different from the weight function used in denoising diffusion models, and γ_t is derived to guarantee that minimizing Eq. (54) is equivalent to minimizing the KL divergence between the data distribution and the terminal distribution of our approximated process.

Another line of works on discrete diffusion models (Austin et al., 2021; Hoogeboom et al., 2021; Vignac et al., 2023) aims to predict the probabilities of each state of the final data to be generated. Since these works predict the probabilities, they are limited to data with a finite number of states and cannot be applied to data with continuous features. In contrast, our method directly predicts the weighted mean of the data (i.e., graph mixture) instead of the probabilities, which can be applied to data with continuous features, for example, 3D molecules as well as the discrete data, which we experimentally validate to be effective. It is worth noting that our GruM is a continuous diffusion model, and thereby our framework can leverage the advanced sampling strategies that reduce the sampling time or improve sample quality (Campbell et al., 2022), whereas the discrete diffusion models are forced to use a simple ancestral sampling strategy.

Table 3: **Comparison of graph diffusion models.**

	Explicitly model graph topology	Simulation-free training	Arbitrary prior distribution	Does not require large diffusion time T	Learning object is bounded	Model prediction
EDM (Hoogeboom et al., 2022)	✗	✓	✗	✗	✓	Noise
GDSS (Jo et al., 2022)	✗	✓	✗	✗	✗	Score
Bridge (Wu et al., 2022)	✗	✗	✓	✓	✗	Drift
GruM (Ours)	✓	✓	✓	✓	✓	Data

A.11. Additional explanation on relevant works

Related works on diffusion bridges Recent works (Peluchetti, 2021; Wu et al., 2022; Ye et al., 2022; Liu et al., 2023) introduce learning the generative process using a mixture of diffusion bridge processes, instead of learning to reverse the noising process as in denoising diffusion models. Peluchetti (2021) introduces a diffusion mixture representation that constructs a generation process as a mixture of the bridge processes. Wu et al. (2022) injects physical information into the process by adding informative prior to the drift, while Ye et al. (2022) and Liu et al. (2023) extend the bridge process to constrained domains.

Related works on graph generative models Recently, graph diffusion models (Niu et al., 2020; Jo et al., 2022; Hoogeboom et al., 2022; Vignac et al., 2023) have made large progress on generating general graphs as well as molecular graphs. EDP-GNN (Niu et al., 2020) aims to generate the adjacency matrix by learning the score function of the denoising diffusion process, while GDSS (Jo et al., 2022) proposes a framework for generating the nodes and edges simultaneously by learning the joint score function that captures the node-edge dependency. However, learning the score function is ill-suited for modeling the graph topology as it does not explicitly consider the graph structures, and further could be problematic due to the diverging score function. On the other hand, discrete diffusion model (Vignac et al., 2023) proposes to model the noising process as successive graph edits, preserving the discrete structure during the diffusion process. However, this is not a desirable solution for real-world graph generation tasks since it applies to graphs with categorical node and edge attributes, and cannot be alone applied to graphs with continuous features, such as the 3D coordinates of atoms. We summarize the comparison between closely related graph diffusion models in Table 3.

B. Details of GruM

In this section, we provide the details of the training and sampling methods of GruM, describe the models used in our experiments, and further discuss the hyperparameters of GruM.

B.1. Overview

We provide the pseudo-code of the training and sampling of our generative framework in Algorithm 3 and 4, respectively. We further provide the implementation details of the training and sampling for each generation task in Section C.

B.2. Training objectives

Random permutation The general graph datasets, namely Planar and SBM, contain only 200 graphs. Thus to ensure the permutation invariant nature of the graph dataset, we apply random permutation to the graphs of the training set during training. To be specific, for a graph data $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{A})$ in the training set and random permutation matrix $\boldsymbol{P}$, we use the permuted data $(\boldsymbol{P}^T\boldsymbol{X}, \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P})$ for training, where $\boldsymbol{P}^T$ denotes the transposed matrix. We empirically found that this leads to more stable training.

Simplified loss We provide the explicit form of simplified loss explained in Section 3.2, which uses constant loss coefficient c instead of the time-dependent γ_t as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{G}\sim\mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T c^2\left\|\boldsymbol{s}_\theta(\boldsymbol{G}_t, t) - \boldsymbol{G}_T\right\|^2 \mathrm{d}t\right]. \tag{67}$$

We empirically found that using this loss is beneficial for the generation of continuous features such as eigenvectors of the graph Laplacian or 3D coordinates.

Algorithm 3 Training of GruM

```
Input: Model $s_\theta$, constant $\epsilon$
For each epoch:
 1: Sample graph $G$ from the training set
 2: $N \leftarrow$ number of nodes of $G$
 3: Sample $t \sim [0, T-\epsilon]$ and $G_0 \sim \mathcal{N}(0, \mathbf{I}_N)$
 4: Sample $G_t \sim p_{t|0,T}(G_t|G_0, G)$                ▷ Section A.6
 5: $\gamma_t \leftarrow \sigma_t / u_t v_t$
 6: $\mathcal{L}_\theta \leftarrow \gamma_t^2 \|s_\theta(G_t, t) - G\|^2$          ▷ Eq. (54)
 7: Update $\theta$ using $\mathcal{L}_\theta$
```

Algorithm 4 Sampling of GruM

```
Input: Trained model $s_\theta$, number of sampling steps $K$, diffusion step size $\mathrm{d}t$
 1: Sample number of nodes $N$ from the training set.
 2: $G_0 \sim \mathcal{N}(0, \mathbf{I}_N)$                  ▷ Start from noise
 3: $t \leftarrow 0$
 4: for $k = 1$ to $K$ do
 5:     $\eta \leftarrow \alpha\sigma_t^2 G_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}s_\theta(G_t, t) - G_t\right)$
 6:     $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
 7:     $G_{t+\mathrm{d}t} \leftarrow \eta\mathrm{d}t + \sigma_t\sqrt{\mathrm{d}t}\mathbf{w}$       ▷ Euler-Maruyama
 8:     $t \leftarrow t + \mathrm{d}t$
 9: end for
10: $G \leftarrow$ quantize($G_t$)            ▷ Quantize if necessary
11: Return: Graph $G$
```

Algorithm 5 PC Sampler for GruM

```
Input: Trained models $s_\theta$ and $s_\phi$ (described in Section A.9), number of sampling steps $K$, number of correction steps per prediction $M$, diffusion step size $\mathrm{d}t$, score-to-noise ratio $r$
Output: Sampled graph $G$
 1: Sample number of nodes $N$ from the training set.
 2: $G_0 \sim \mathcal{N}(0, \mathbf{I}_N)$                  ▷ Start from noise
 3: $t \leftarrow 0$
 4: for $k = 1$ to $K$ do
 5:     $\eta \leftarrow \alpha\sigma_t^2 G_t + \frac{\sigma_t^2}{v_t}\left(\frac{1}{u_t}s_\theta(G_t, t) - G_t\right)$
 6:     $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
 7:     $\tilde{G}_t \leftarrow \eta\mathrm{d}t + \sigma_t\sqrt{\mathrm{d}t}\mathbf{w}$               ▷ Predictor
 8:     for $m = 1$ to $M$ do                    ▷ Corrector loop
 9:         $D, \bar{D} \leftarrow s_\theta(\tilde{G}_t, t),\ s_\phi(\tilde{G}_t, T-t)$
10:         $s \leftarrow$ Compute_Score($D, \bar{D}, \tilde{G}_t$)      ▷ Eq.(64)
11:         $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_N)$
12:         $\epsilon \leftarrow 2\left(r\|\mathbf{w}\|_2 / \|s\|_2\right)^2$
13:         $\tilde{G}_t \leftarrow$ Corrector($\tilde{G}_t, s, \epsilon$)
14:     end for
15:     $G_{t+\mathrm{d}t} \leftarrow \tilde{G}_t$
16:     $t \leftarrow t + \mathrm{d}t$
17: end for
18: $G \leftarrow$ quantize($G_t$)            ▷ Quantize if necessary
19: Return: Graph $G$
```

Attributed graphs Especially for the generation of attributed graphs $G = (X, A)$, the graph mixture matching for X and A can be derived from Eq. (11). Specifically, for the model $s_\theta(\cdot, t) = (s_\theta^X(\cdot, t), s_\theta^A(\cdot, t))$, we use the following objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{G \sim \mathbb{Q}^{\Pi^*}}\left[\frac{1}{2}\int_0^T \gamma_{1,t}^2 \left\|s_\theta^X(G_t, t) - X_T\right\|^2 \mathrm{d}t + \frac{\lambda}{2}\int_0^T \gamma_{2,t}^2 \left\|s_\theta^A(G_t, t) - A_T\right\|^2 \mathrm{d}t\right] \tag{68}$$

where λ is the hyperparameter. We use $\lambda = 5$ for all our experiments and empirically observed that changing λ did not make much difference for sufficient training epochs.

B.3. Model architecture

For the general graph and 2D molecule generation tasks, we leverage the graph transformer network introduced in Dwivedi & Bresson (2020) and Vignac et al. (2023). The node features and the adjacency matrices are updated with multiple attention layers with global features obtained by the self-attention-based FiLM layers (Perez et al., 2018). We additionally use the higher-order adjacency matrices following Jo et al. (2022). For general graph generation tasks, we add the sigmoid function to the output of the model since the entries of the weighted mean of the data are supported in the interval $[0, 1]$. For 2D molecule generation tasks, we apply the softmax function to the output of the node features to model the one-hot encoded atom types, and further apply the sigmoid function to the output of the adjacency matrices. Note that we do not use the structural augmentation as in Vignac et al. (2023). For the 3D molecule generation task, we use EGNN (Satorras et al., 2021) to model the E(3) equivariance of the molecule data. We additionally add a softmax function at the last layer to model the one-hot encoded atom types.

B.4. Sampling from GruM

Sampling from the generative model requires solving the SDE of Eq. (10). If our model s_θ can closely approximate the graph mixture, a simple Euler-Maruyama method would be enough to simulate the generative model, which is true for most of the experiments. Since s_θ is trained on the marginal density p_t, G_t outside of p_t could cause incorrect predictions that lead to an undesired result. To address the limitation, we may leverage the predictor-corrector (PC) sampling method introduced in Song et al. (2021). Using the corrector method such as Langevin dynamics (Song et al., 2021), we force $\boldsymbol{G}_t$ to be drawn from p_t which ensures a more accurate estimation of the graph mixture. The score function to be used for the corrector can be computed as in Eq. (64) of Section A.9. We provide the pseudo-code of the predictor-only sampler and the PC sampler for our GruM in Algorithm 4 and 5. Note that our GruM does not require additional time for sampling compared to the denoising diffusion models, since the generation is equivalent to solving the SDE where the drift is computed each step from the forward pass of the model.

B.5. Hyperparameters of GruM

The generative process of GruM modeled as the OU bridge mixture is uniquely determined with the noise schedule σ_t and constant α. Through our experiments, we use $\alpha = -1/2$ and a decreasing linear noise schedule, starting from σ_0^2 and ends in σ_0^2 defined as follows:

$$\sigma_t^2 = (1-t)\sigma_0^2 + t\sigma_1^2 \;\; \text{for} \;\; 0 < \sigma_1 < \sigma_0 < 1 \tag{69}$$

We perform a grid search for the hyperparameters σ_0 and σ_1 in $\{0.4, 0.6, 0.8, 1.0\}$ and $\{0.1, 0.2, 0.3\}$, respectively, where the search space slightly differs for each generation task.

C. Experimental Details

C.1. General graph generation

Datasets and evaluation metrics We evaluate the quality of generated graphs on three graph datasets used as benchmarks in Martinkus et al. (2022): **Planar** graph dataset consists of 200 synthetic planar graphs where each graph has 64 nodes. We determine that a graph is a valid Planar graph if it is connected and planar. **Stochastic Block Model** (SBM) graph dataset consists of 200 synthetic stochastic block model graphs with the number of communities uniformly sampled between 2 and 5, where the number of nodes in each community is uniformly sampled between 20 and 40. The probability of the inter-community edges and the intra-community edges are 0.3 and 0.05, respectively. We determine that a graph is a valid SBM graph if it has the number of communities between 2 and 5, the number of nodes in each community between 20 and 40, and further using the statistical test introduced in Martinkus et al. (2022). **Proteins** graph dataset (Dobson & Doig, 2003) consists of 918 real protein graphs with up to 500 nodes in each graph. The protein graph is constructed by considering each amino acid as a node and connecting two nodes if the corresponding amino acids are less than 6 Angstrom. For our experiments, we use the datasets provided by Martinkus et al. (2022).

We follow the evaluation setting of Liao et al. (2019) using total variation (TV) distance for measuring MMD which is considerably fast compared to using the earth mover's distance (EMD) kernel, especially for large graphs. Moreover, we use the V.U.N. metric following Martinkus et al. (2022) that measures the proportion of valid, unique, and novel graphs among the generated graphs, where the validness is defined as satisfying the specific property of the dataset explained above. The baseline results are taken from Vignac et al. (2023) or obtained by running the open-source codes.

Implementation details We follow the standard setting of Martinkus et al. (2022) using the same data split and evaluation procedures. We report the baseline results taken from Martinkus et al. (2022) and Vignac et al. (2023), or the results obtained from running the open-source codes using the hyperparameters given by the original work. We could not report the results of EDP-GNN (Niu et al., 2020) and DiGress (Vignac et al., 2023) on the Proteins dataset as they took more than 2 weeks. For the diffusion models including our proposed method, we set the diffusion steps to 1000 for a fair comparison.

For our proposed GruM, we train our model for 30,000 epochs for all datasets using a constant learning rate with AdamW optimizer (Loshchilov & Hutter, 2017) and weight decay 10^{-12}, applying the exponential moving average (EMA) to the parameters (Song & Ermon, 2020). We perform the hyperparameter search explained in Section B.5 for the lowest MMD and highest V.U.N. results. We initialize the node features with the eigenvectors of the graph Laplacian of the adjacency matrices, which we further scale with constant. During training (Algorithm 3), we sample the noise for the adjacency matrices to be symmetric with zero diagonals. During generation (Algorithm 4), we generate both the node features and adjacency matrices starting from noise, and we quantize the entries of the resulting adjacency matrices. Empirically, we found that the entries of the resulting sample lie very close to the desired values 0 and 1, for which the L1 distance between the resulting sample and the quantized sample is smaller than 10^{-2}.

In Figure 2 (Left), we measure the Spec. MMD and V.U.N. of our method and the baselines as follows: For GruM we evaluate the predicted graph mixture. For DiGress, we evaluate the prediction obtained from the predicted probability of each state. For GDSS and ConGress, we evaluate the implicit prediction computed from the estimated score or noise following Eq. (16) of Hoogeboom et al. (2022). The Spec. MMD and the V.U.N. are measured after quantizing the predicted adjacency matrix with thresholding at 0.5.

C.2. 2D molecule generation

Datasets and evaluation metrics We evaluate the quality of generated 2D molecules on two molecule datasets used as benchmarks in Jo et al. (2022). **QM9** (Ramakrishnan et al., 2014) dataset consists of 133,885 molecules with up to 9 heavy atoms of four types. **ZINC250k** (Irwin et al., 2012) dataset consists of 249,455 molecules with up to 38 heavy atoms of 9 types. Molecules in both datasets have 3 edge types, namely single bond, double bond, and triple bond. For our experiments, we follow the standard procedure (Shi et al., 2020; Luo et al., 2021; Jo et al., 2022) of kekulizing the molecules using the RDKit library (Landrum et al., 2016) and removing the hydrogen atoms from the molecules in the QM9 and ZINC250k datasets.

We evaluate the models with four metrics: **Validity** is the percentage of the valid molecules among the generated without any post-hoc correction such as valency correction or edge resampling. **Fréchet ChemNet Distance** (FCD) (Preuer et al., 2018) measures the distance between the feature vectors of generated molecules and the test set using ChemNet, evaluating the chemical properties of the molecules. **Neighborhood subgraph pairwise distance kernel** (NSPDK) MMD (Costa & De Grave, 2010) measures the MMD between the generated molecular graphs and the molecular graphs from the test set, assessing the quality of the graph structure. **Scaffold similarity** (Scaf.) measures the cosine similarity of the frequencies of Bemis-Murcko scaffolds (Bemis & Murcko, 1996), evaluating the ability to generate similar substructures. See Section C.2 for more details. Among these, FCD and NSPDK MMD metrics measure the distribution similarity between the test dataset and generated samples while scaffold similarity evaluates the similarity of the generated scaffolds. The baseline results are taken from Jo et al. (2022) or obtained by running open-source codes.

Implementation details We report the results of the baselines taken from Jo et al. (2022), except for the results of the Scaffold similarity (Scaf.) and the results of DiGress, which we obtained by running the open-source codes. Especially, the 2D molecule generation results of DiGress on the QM9 dataset are different compared to the results reported in its original paper, since we have used the preprocessed dataset following the setting of Jo et al. (2022) for a fair comparison with other baselines. We set the diffusion steps to 1000 for all the diffusion models.

For our GruM, we train our model $\boldsymbol{s}_\theta$ with a constant learning rate with AdamW optimizer and weight decay 10^{-12}, applying the exponential moving average (EMA) to the parameters. We perform the hyperparameter search similar to that of the general graph generation tasks. Especially for GruM, we follow Jo et al. (2022) by using the adjacency matrices in the form of $\boldsymbol{A} \in \{0,1,2,3\}^{N\times N}$ where N is the maximum number of atoms in a molecule and each entries indicating the bond types: 0 for no bond, 1 for the single bond, 2 for the double bond and 3 for the triple bond. Further, we scale the entries with a constant scale of 3 in order to bound the input of the model in the interval $[0,1]$, and rescale the final sample of the generation process to recover the bond types. We also sample the noise for the adjacency matrices to be symmetric with zero diagonals during training. We quantize the entries of the resulting adjacency matrices to obtain the discrete bond types $\{0,1,2,3\}$. Empirically, we found that the entries of the resulting sample lie very close to the desired bond types $\{0,1,2,3\}$, for which the L1 distance between the resulting sample and the quantized sample is approximately 0.

C.3. 3D molecule generation

Datasets and evaluation metrics We evaluate the quality of generated 3D molecules on two molecule datasets used as benchmarks in Hoogeboom et al. (2022). **QM9** (Ramakrishnan et al., 2014) dataset consists of 133,885 molecules with up to 29 atoms of five types including hydrogen atoms. The node features of the QM9 dataset include the one-hot representation of atom type and atom number. **GEOM-DRUGS** (Axelrod & Gomez-Bombarelli, 2022) dataset consists of 430,000 molecules with up to 181 atoms of fifteen types including hydrogen atoms. GEOM-DRUGS dataset contains different conformations for each molecule. Among many conformations, the 30 lowest energy conformations for each molecule are retained. For the GEOM-DRUGS dataset, we utilize the one-hot representation of atom type without the atom number. To evaluate the generated molecules, we measure the **atom and molecule stability** by predicting the bond type between atoms with the standard bond lengths and then checking the valency.

Implementation details We follow the standard setting of Hoogeboom et al. (2022) for a fair comparison: for the QM9 experiment, we use EGNN with 256 hidden features and 9 layers and train the model, and for the GEOM-DRUGS experiment, we use EGNN with 256 hidden features and 4 layers and train the model. We report the results of the baselines taken from Hoogeboom et al. (2022) and Wu et al. (2022). In Figure 3 (Right), we compute the implicit prediction using the

Table 4: **2D molecule generation results on the QM9 dataset.** The baseline results are taken from Jo et al. (2022) or obtained by running the open-source codes. Best results are highlighted in bold.

Method	Valid (%)↑	FCD ↓	NSPDK ↓	Scaf. ↑	Uniq (%) ↑	Novelty (%) ↑
MoFlow (Zang & Wang, 2020)	91.36 ±1.23	4.467 ±0.595	0.017 ±0.003	0.1447 ±0.0521	98.65 ±0.57	**94.72** ±0.77
GraphAF‘(Shi et al., 2020)	74.43 ±2.55	5.625 ±0.259	0.021 ±0.003	0.3046 ±0.0556	88.64 ±2.37	86.59 ±1.95
GraphDF (Luo et al., 2021)	93.88 ±4.76	10.928 ±0.038	0.064 ±0.000	0.0978 ±0.1058	98.58 ±0.25	98.54 ±0.48
EDP-GNN (Niu et al., 2020)	47.52 ±3.60	2.680 ±0.221	0.005 ±0.001	0.3270 ±0.1151	**99.25** ±0.05	86.58 ±1.85
GDSS (Jo et al., 2022)	95.72 ±1.94	2.900 ±0.282	0.003 ±0.000	0.6983 ±0.0197	98.46 ±0.61	86.27 ±2.29
DiGress (Vignac et al., 2023)	98.19 ±0.23	**0.095** ±0.008	0.0003 ±0.000	0.9353 ±0.0025	96.67 ±0.24	25.58 ±2.36
GruM (ours)	**99.69** ±0.19	0.108 ±0.006	**0.0002** ±0.000	**0.9449** ±0.0054	96.90 ±0.15	24.15 ±0.80

Table 5: **2D molecule generation results on the ZINC250k dataset.** The baseline results are taken from Jo et al. (2022) or obtained by running the open-source codes. Best results are highlighted in bold.

Method	Valid (%)↑	FCD ↓	NSPDK ↓	Scaf. ↑	Uniq (%) ↑	Novelty (%) ↑
MoFlow (Zang & Wang, 2020)	63.11 ±5.17	20.931 ±0.184	0.046 ±0.002	0.0133 ±0.0052	**99.99** ±0.01	**100.00** ±0.00
GraphAF (Shi et al., 2020)	68.47 ±0.99	16.023 ±0.451	0.044 ±0.005	0.0672 ±0.0156	98.64 ±0.69	99.99 ±0.01
GraphDF (Luo et al., 2021)	90.61 ±4.30	33.546 ±0.150	0.177 ±0.001	0.0000 ±0.0000	99.63 ±0.01	**100.00** ±0.00
EDP-GNN (Niu et al., 2020)	82.97 ±2.73	16.737 ±1.300	0.049 ±0.006	0.0000 ±0.0000	99.79 ±0.08	**100.00** ±0.00
GDSS (Jo et al., 2022)	97.01 ±0.77	14.656 ±0.680	0.019 ±0.001	0.0467 ±0.0054	99.64 ±0.13	**100.00** ±0.00
DiGress (Vignac et al., 2023)	94.99 ±2.55	3.482 ±0.147	0.0021 ±0.0004	0.4163 ±0.0533	99.97 ±0.01	99.99 ±0.01
GruM (ours)	**98.65** ±0.25	**2.257** ±0.084	**0.0015** ±0.0003	**0.5299** ±0.0441	99.97 ±0.03	99.98 ±0.02

estimated noise following Eq. (16) of Hoogeboom et al. (2022).

For our GruM, we train our model s_θ for 1,300 epochs with batch size 256 for the QM9 experiment, and for 13 epochs with batch size 64 for the GEOM-DRUGS experiment. We apply EMA to the parameters of the model with a coefficient of 0.999 and use AdamW optimizer with learning rate 10^{-4} and weight decay 10^{-12}. The 3D coordinates and charge values are scaled as $\times 4$ and $\times 0.1$, respectively, and we use the simplified loss with a constant $c = 100$. We perform the hyperparameter search with smaller values for coordinates in $\{0.1, 0.2, 0.3\}$ and higher values for node features in $\{0.6, 0.7, 0.8, 0.9, 1.0\}$. For the generation, we use the Euler-Maruyama predictor.

C.4. Computing resources

For all experiments, we use NVIDIA GeForce RTX 3090 and 2080 Ti and implement the source code with PyTorch (Paszke et al., 2019).

D. Additional Experimental Results

D.1. 2D molecule generaation

We provide the standard deviation results in Table 4 and Table 5. We additionally report the following two metrics: **Novelty** is the proportion of the molecules generated that are valid and not in the training set, and **Uniqueness** is the proportion of the molecules generated that are valid and unique, where valid molecules are the ones that do not violate the chemical valency rule.

D.2. Further analysis

Comparison with learning the drift To verify that learning the graph mixture as in our GruM is superior compared to learning the drift, we additionally report the generation result of the variant of GruM which learns the drift, similar to Wu et al. (2022), on the Planar dataset. Table in Figure 4 shows that learning the drift, denoted as Drift in the table, performs poorly generating only 15% valid, novel, and unique graphs. The generated Planar graphs in Figure 4 demonstrate that learning the drift fails to capture the correct topology.

Further, to verify why learning the drift fails to capture the correct topology, we compare the complexity of the models for learning different objectives. As shown in Figure 8, the complexity of learning the drift (Drift) is significantly higher than learning the graph

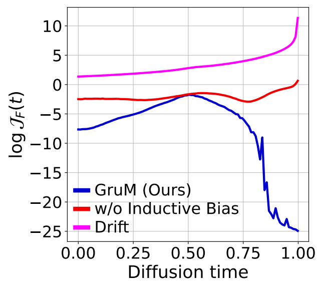


Figure 8: Model complexity comparison of GruM and Drift.

Figure 4: **(Left) Generation results on the Planar dataset.** Best results are highlighted in bold, where smaller MMD and larger V.U.N. indicate better results. **(Right) Generated graphs by learning the drift.** Visualized graphs are randomly sampled without curation.

	Planar				
	Deg.	Clus.	Orbit	Spec.	V.U.N.
Training set	0.0002	0.0310	0.0005	0.0052	100.0
GraphRNN (You et al., 2018)	0.0049	0.2779	1.2543	0.0459	0.0
SPECTRE (Martinkus et al., 2022)	0.0005	0.0785	0.0012	0.0112	25.0
EDP-GNN (Niu et al., 2020)	0.0044	0.3187	1.4986	0.0813	0.0
GDSS (Jo et al., 2022)	0.0041	0.2676	0.1720	0.0370	0.0
DiGress (Vignac et al., 2023)	**0.0003**	0.0372	**0.0009**	0.0106	75
Drift	0.0008	0.0845	0.0075	0.0126	15
GruM (Ours)	0.0005	**0.0353**	**0.0009**	**0.0062**	**90.0**

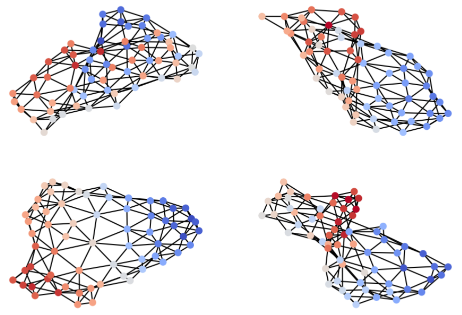
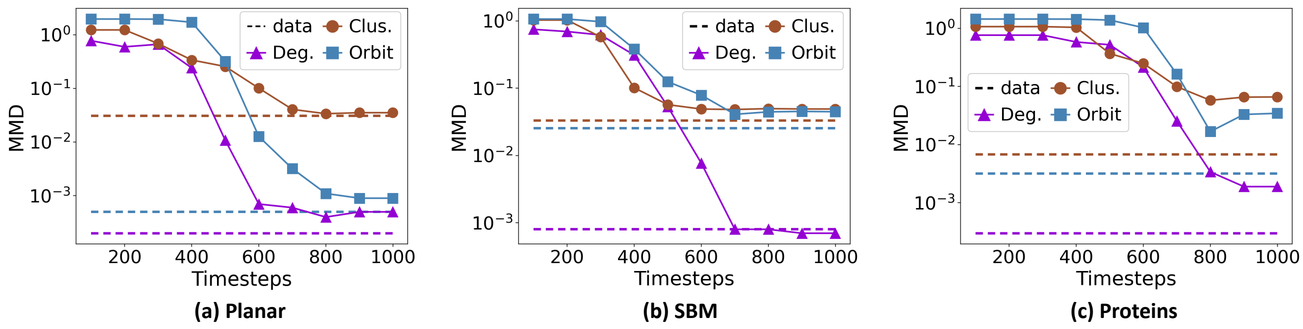


Figure 5: **MMD results of graph mixture of GruM** through the generative process. Dotted lines indicate MMDs of training set.



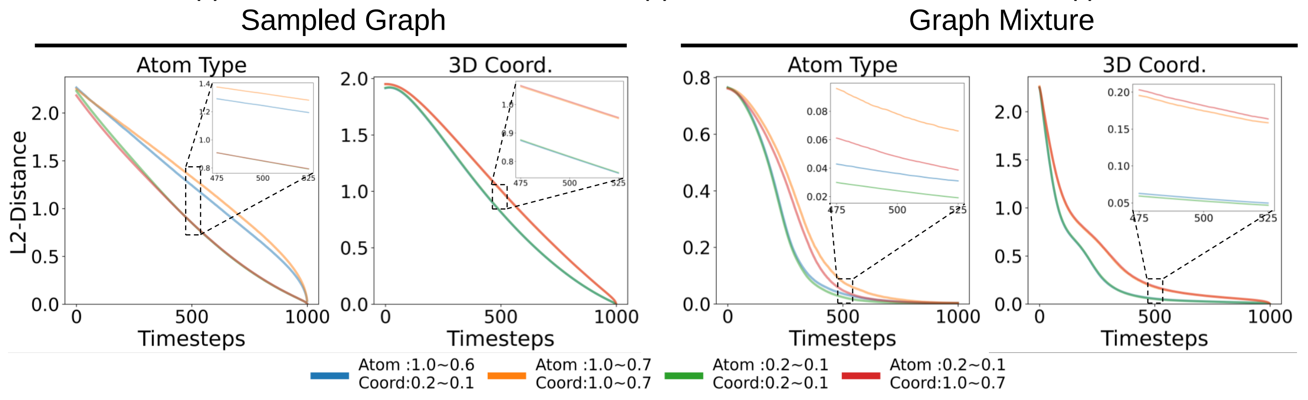


Figure 6: **Convergence of sampled graphs and graph mixtures with varying σ_0 and σ_1 values.**

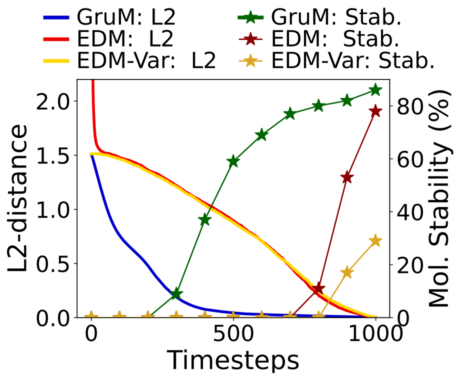
mixture (GruM) for all time steps. Moreover, learning the drift is much harder compared to learning the graph mixture without exploiting the graph structure (w/o Inductive Bias). In particular, the complexity gap dramatically increases at the late stage of the diffusion process, because the drift diverges approaching the terminal time while the graph mixture is supported inside the data space, as discussed in Section 3.2.

Early Stopping for Generative Process In Figure 2 (Left) and (Middle), the V.U.N. and the MMD results of GruM in the Planar dataset demonstrate that the estimated graph mixture converges to the final result at early sampling steps, accurately capturing both the global topology and local graph characteristics. This allows us to early-stop the diffusion process, which reduces the generation time by up to 20% on this task.

We provide additional MMD results of the generative processes in Figure 5, which show that the estimated graph mixture

Figure 7: The experimental results for the variant of EDM where it aims to predict the final result (EDM-Var.). **(Left) Generation results on the 3D molecule QM9 datasets.** Best results are highlighted in bold where the higher stability indicates better results. **(Right) Convergence of the generative process.** We compare the convergence of the graph mixture from GruM, the implicit prediction computed from the estimated noise of EDM, and the predicted result of EDM-Var. We measure the convergence with L2 distance and further visualize the molecule stability of the predictions through the generative process.

	QM9 ($\|V\| \leq 29$)	
Method	Atom Stab.(%)	Mol. Stab.(%)
G-Schnet (Gebauer et al., 2019)	95.7	68.1
GDM (Hoogeboom et al., 2022)	97.0	63.2
EDM (Hoogeboom et al., 2022)	98.7	82.0
Bridge (Wu et al., 2022)	98.7	81.8
Bridge+Force (Wu et al., 2022)	98.8	84.6
EDM-Var.	94.02	35.95
GruM (Ours)	**98.81**	**87.34**



converges to the final result around 800 diffusion steps for all datasets.

Role of the diffusion coefficient We can observe that the generative process of GruM is uniquely determined by the constant α and the diffusion coefficient σ_t. These two coefficients control the convergence behavior of the diffusion process: large α and small σ_t lead to a drift with a large norm that forces the trajectory to converge quickly. Here, we demonstrate the effect of the diffusion coefficient σ_t on the convergence of the generative process. Figure 6 (Sampled Graph) shows that the smaller values of σ_t (i.e. 0.2$\sim$0.1) lead to faster convergence of the trajectory to the final result, compared to the larger σ_t. This is due to the fast convergence of each bridge process with small σ_t. Especially, as shown in Figure 6 (Graph Mixture), large σ_t for the continuous feature (i.e., 3D coordinates) leads to slower convergence of the graph mixture since it destroys the topology of graphs and makes it hard to predict the final result.

Graph prediction through EDM Additionally, we compare our GruM with the variant of EDM (Hoogeboom et al., 2022) which learns to predict the final result of the denoising process instead of learning the noise. Table of Figure 7 shows the generation result of this variant, denoted as EDM-Var., on the 3D molecule QM9 dataset. EDM-Var. exhibits the lowest atom stability and extremely low molecule stability of less than 40%, which performs significantly worse than GruM as well as the original EDM. This is because EDM-Var. depends on a single deterministic prediction during the generative process, and the inaccurate prediction of the final result at the early step of the generative process leads the process in the wrong direction resulting in invalid molecules, as discussed in the Introduction and Section 3.1.

On the other hand, our GruM predicts the final grpah of the generative process using the graph mixture which represents the probable graph as a weighted mean of the data, thereby guiding the process in the right direction resulting in valid molecules with correct topology. We further provide the convergence results of EDM-Var. in Figure 7, which demonstrates that the prediction of GruM converges significantly faster than that of EDM and EDM-Var. The inaccurate prediction of EDM-Var. results in slower convergence and low molecule stability.

Analysis on the model architecture As shown in Table 6 and 7, GDSS using graph transformer architecture shows improved performance over original GDSS but is still outperformed by our GruM with a large margin in V.U.N, FCD, and NSPDK. These results verify that the superior performance of GruM comes from its ability to accurately model the topology of the final graph to be generated.

Table 6: **General graph generation results with GDSS using graph transformer.**

	Planar					SBM				
	Synthetic, $\|V\| = 64$					Synthetic, $44 \leq \|V\| \leq 187$				
	Deg.	Clus.	Orbit	Spec.	V.U.N.	Deg.	Clus.	Orbit	Spec.	V.U.N.
Training set	0.0002	0.0310	0.0005	0.0052	100.0	0.0008	0.0332	0.0255	0.0063	100.0
GDSS	0.0041	0.2676	0.1720	0.0370	0.0	0.0212	0.0646	0.0894	0.0128	5.0
GDSS+Transformer	0.0036	0.1206	0.0525	0.0137	5.0	0.0411	0.0565	0.0706	0.0074	27.5
ConGress	0.0048	0.2728	1.2950	0.0418	0.0	0.0273	0.1029	0.1148	-	0.0
DiGress	**0.0003**	0.0372	**0.0009**	0.0106	75	0.0013	0.0498	0.0434	0.0400	74
GruM (Ours)	0.0005	**0.0353**	**0.0009**	**0.0062**	**90.0**	**0.0007**	**0.0492**	0.0448	**0.0050**	**85.0**

Table 7: **2D molecule generation results with GDSS using graph transformer.**

	QM9 $(\|V\| \leq 9)$				ZINC250k $(\|V\| \leq 38)$			
Method	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑	Valid (%)↑	FCD↓	NSPDK↓	Scaf.↑
Training set	100.0	0.0398	0.0001	0.9719	100.0	0.0615	0.0001	0.8395
GDSS	95.72	2.900	0.0033	0.6983	97.01	14.656	0.0195	0.0467
GDSS+Transformer	99.68	0.737	0.0024	0.9129	96.04	5.556	0.0326	0.3205
DiGress	98.19	**0.095**	0.0003	0.9353	94.99	3.482	0.0021	0.4163
GruM (Ours)	**99.69**	0.108	**0.0002**	**0.9449**	**98.65**	**2.257**	**0.0015**	**0.5299**

E. Visualization

In this section, we visualize the generated graphs and molecules of GruM, and further provide visualization of the diffusion processes for diverse generation tasks.

E.1. Generated samples of GruM

General graphs Graphs from the training set and the generated graphs of GruM are visualized in Figure 9, 10, and 11. The visualized graphs are randomly selected from the training set and the generated graph set. Note that we visualize the entire graph for the Proteins dataset, unlike Martinkus et al. (2022) which visualizes the largest connected component since it fails to consistently generate connected graphs. For GruM, we found that 92% of the generated Proteins graphs are connected.

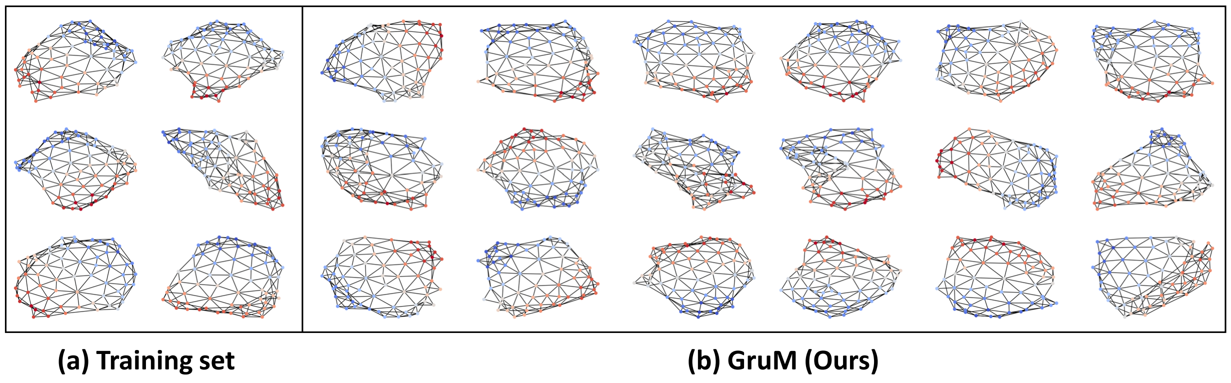


Figure 9: **Visualization of graphs from the Planar dataset and the generated graphs of GruM.**

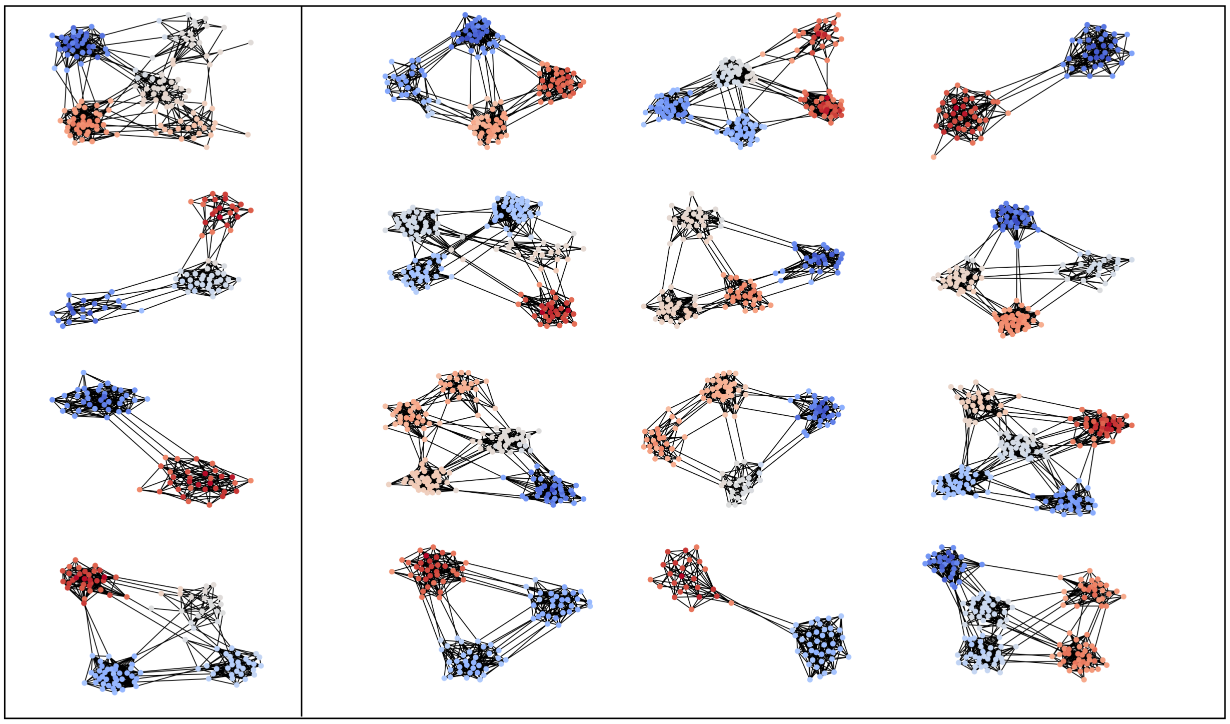


Figure 10: **Visualization of graphs from the SBM dataset and the generated graphs of GruM.**

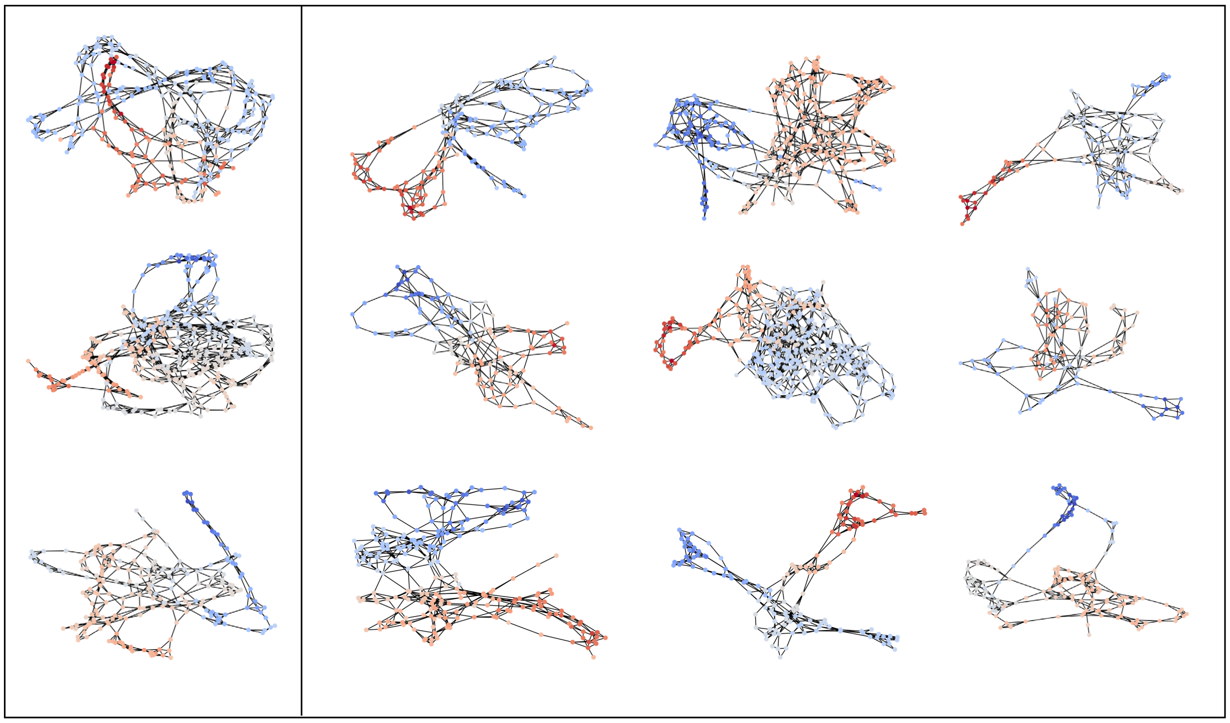


Figure 11: **Visualization of graphs from the Proteins dataset and the generated graphs of GruM.**

2D molecules We provide the visualization of the molecules from the training set and the generated 2D molecules in Figure 12 and 13. These molecules are randomly selected from the training set and the generated molecule set.

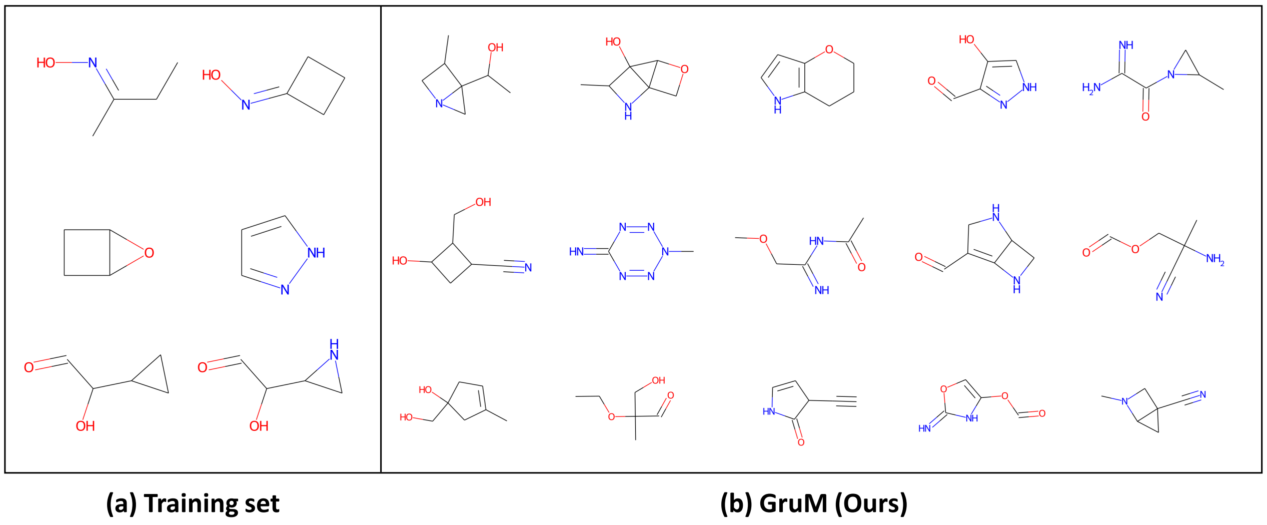


Figure 12: **Visualization of molecules from the QM9 dataset and the generated molecules of GruM for the 2D molecule generation experiment.**

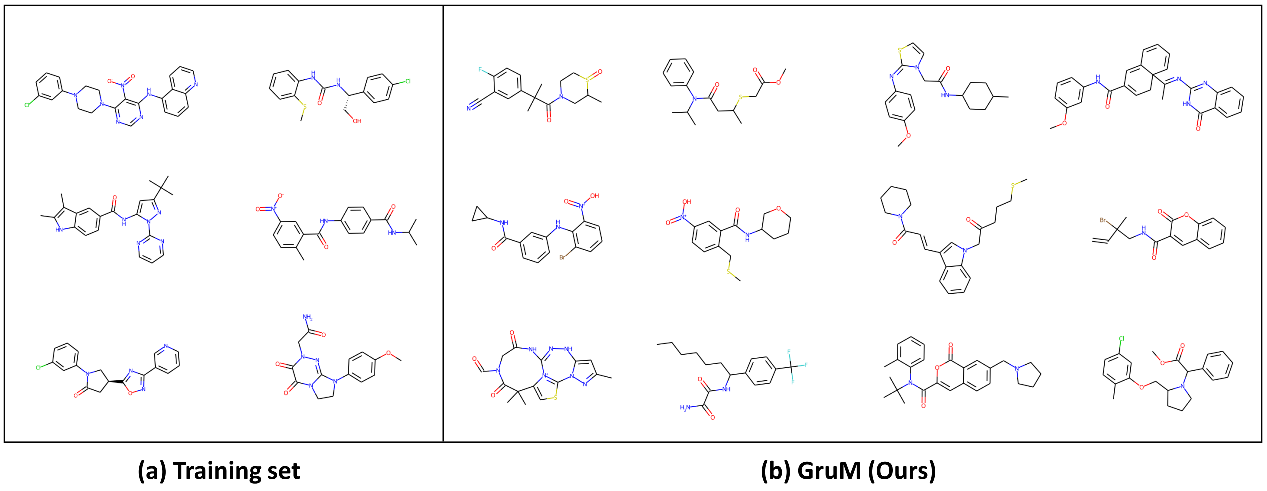


Figure 13: **Visualization of the molecules from the ZINC250k dataset and the generated molecules of GruM for the 2D molecule generation experiment.**

3D molecules We visualize the generated molecules for the 3D molecule generation experiment in Figure 14 and 15. Note that the visualized molecules are all stable. For the GEOM-DRUGS experiment, we observe that a few of the generated molecules are not connected as pointed out in Hoogeboom et al. (2022). To measure how many graphs are connected, we report the fraction of the connected graphs, taking the average of 3 different runs. Table 8 shows that GruM can generate a significantly larger number of connected molecules compared to EDM (Hoogeboom et al., 2022).

Table 8: Fraction of connected graphs on GEOM-DRUGS experiment.

Methods	Connected (%)
EDM	37.70 ±0.79
GruM (Ours)	**56.57** ±0.31

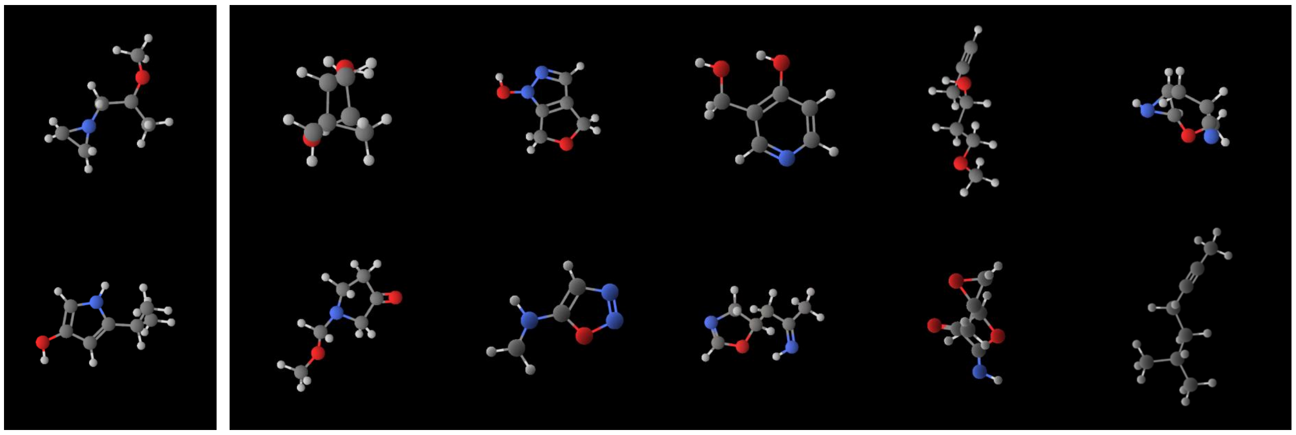


Figure 14: **Visualization of the molecules from the QM9 dataset and the generated molecules of GruM for the 3D molecule generation experiment.**

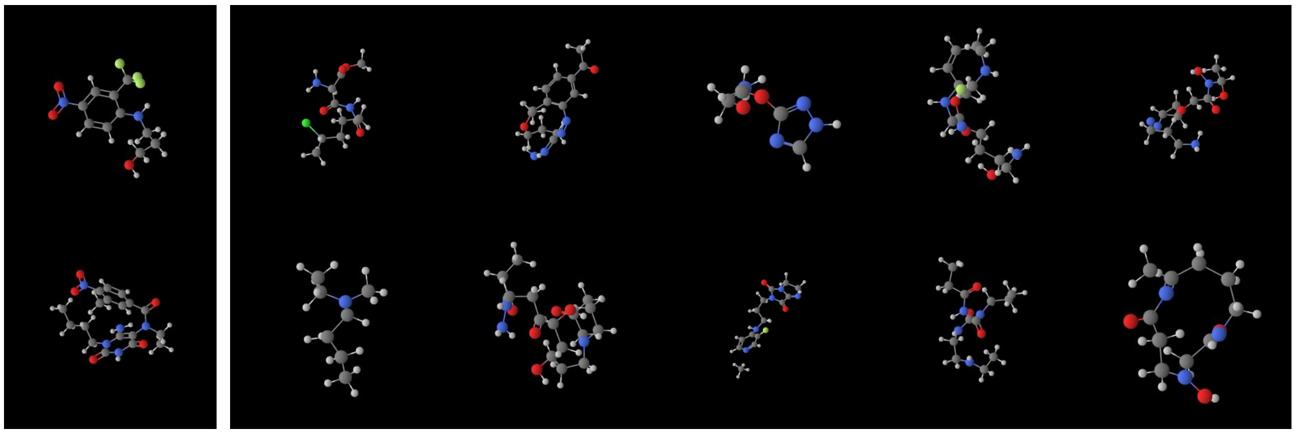


Figure 15: **Visualization of the molecules from the GEOM-DRUGS dataset and the generated molecules of GruM for the 3D molecule generation experiment.**

E.2. Generative process of GruM

Here we visualize the generative process of GruM. We visualize the generative process of general graphs in Figure 16, 17, and 18. We also visualize the generative process of the 3D molecules in Figure 19. We further provide the animation of the generative process in https://github.com/harryjo97/GruM.

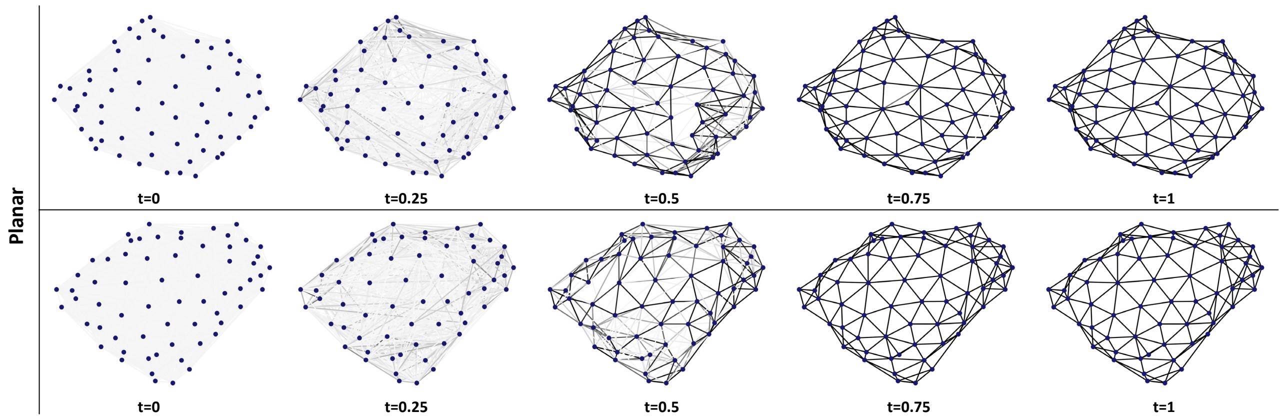


Figure 16: **Visualization of the generative process of GruM.** We visualize the graph mixture from GruM on the Planar dataset.

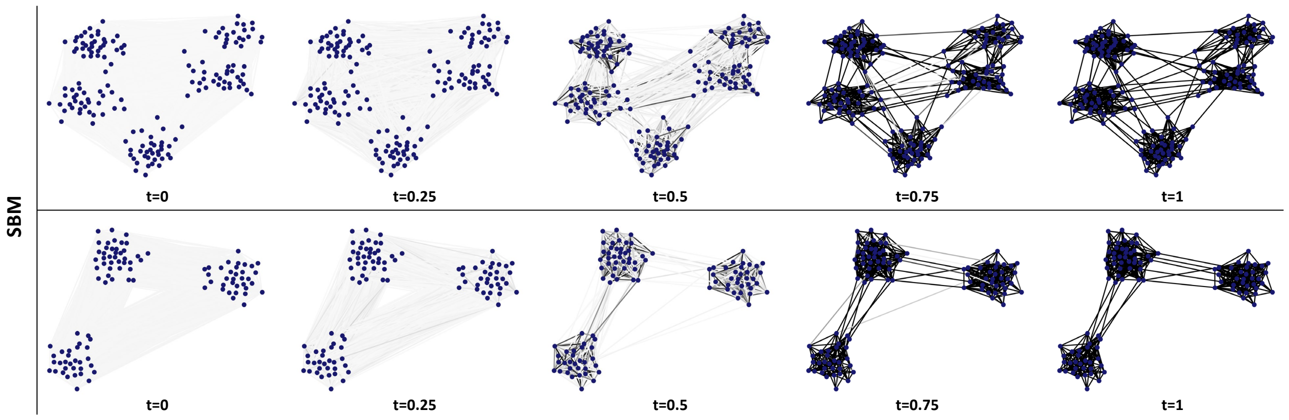


Figure 17: **Visualization of the generative process of GruM.** We visualize the graph mixture from GruM on the SBM dataset.

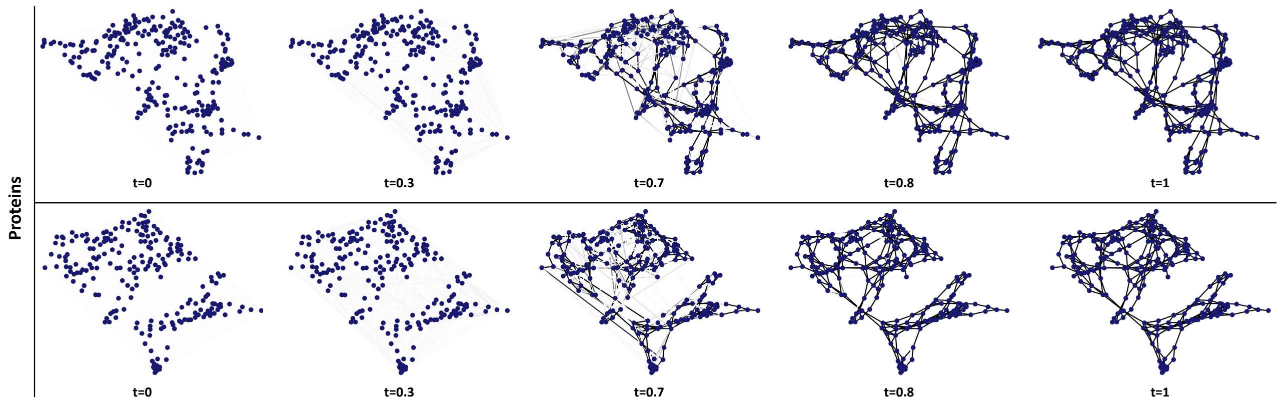


Figure 18: **Visualization of the generative process of GruM.** We visualize the graph mixture from GruM on the Proteins dataset.

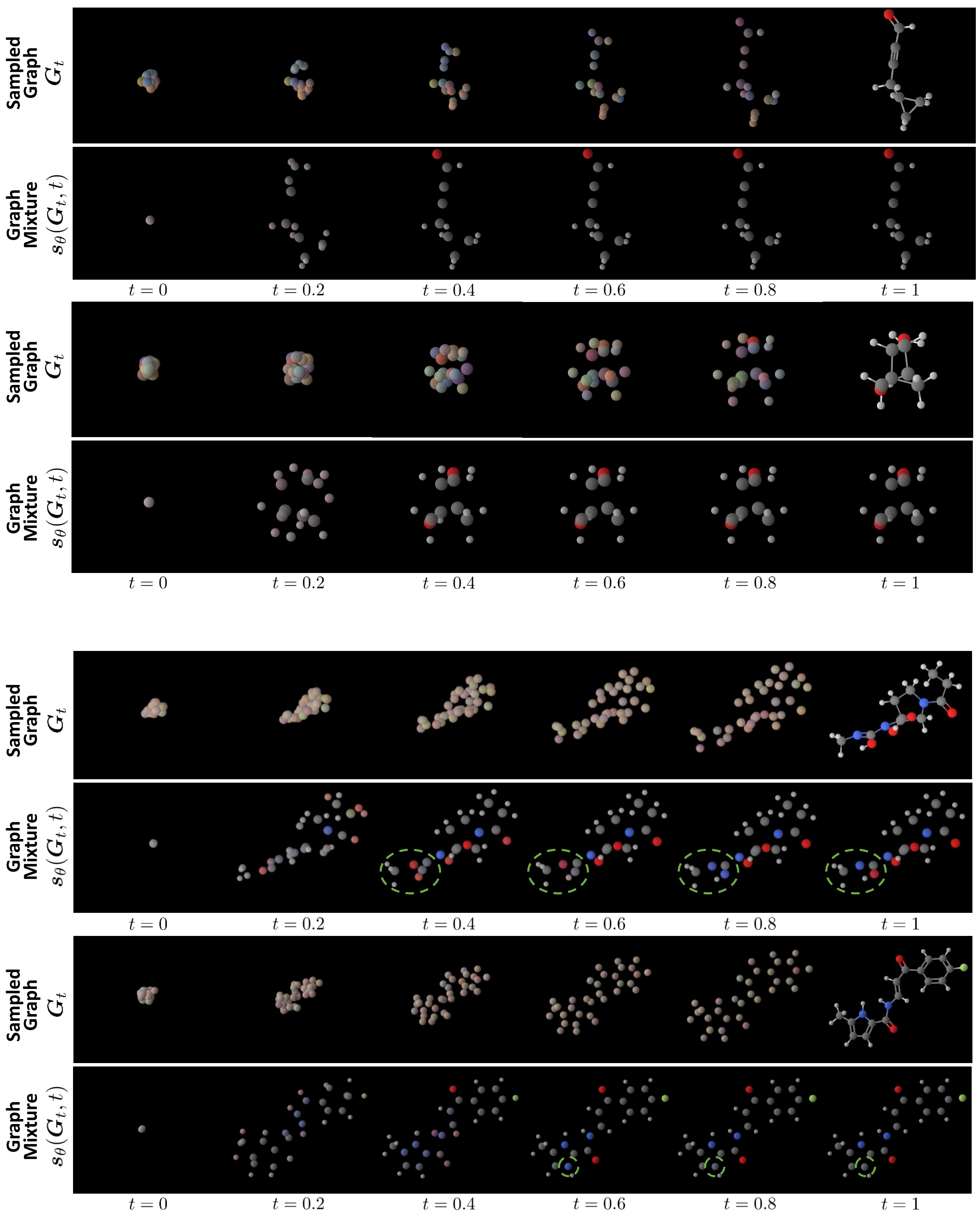


Figure 19: **Visualizations of the 3D molecule generative process** of GruM on QM9 dataset (Top) and GEOM-DRUGS dataset (Bottom). For each dataset, we visualize the trajectory $\boldsymbol{G}_t$ in the first row, and we visualize the estimated graph mixtures from GruM in the second row. Note that the visualized molecules are stable. The atom types and the 3D coordinates of the atoms inside the green circles are calibrated after the convergence of the graph mixtures, where the convergence is achieved at an early stage.

F. Limitation

Limitation We proposed a novel diffusion-based graph generation framework that directly predicts the final graph of the generative process as a weighted mean of data, thereby accurately capturing the valid structures and the topological properties. We have shown that our framework is able to generate graphs with correct topology for diverse graph generation tasks, including 2D/3D molecular generation, on which ours significantly outperforms previous graph generation methods. While GruM shows superior performance, future work would benefit from improving our framework.

First, the likelihood of the generative process of GruM cannot be directly computed from the training objective. In order to compute the likelihood, one could derive an associated probability flow ODE of GruM as described in Section A.9, but this requires training an additional model for estimating the reverse graph mixture.

Furthermore, the proposed framework is focused on unconditional graph generation tasks. We could design a conditional framework of GruM by training a model $\boldsymbol{s}_\theta(\boldsymbol{G}_t, t, \boldsymbol{c})$ for a given condition (i.e., class label) $\boldsymbol{c}$ for estimating the $\boldsymbol{c}$-conditional graph mixture defined as follows:

$$\boldsymbol{D}^{\Pi^*_{\boldsymbol{c}}}(\boldsymbol{G}_t, t) \coloneqq \int \boldsymbol{g} \frac{p_t^{\boldsymbol{g}}(\boldsymbol{G}_t)}{p_t(\boldsymbol{G}_t)} \Pi^*_{\boldsymbol{c}}(\mathrm{d}\boldsymbol{g}), \quad \Pi^*_{\boldsymbol{c}} \coloneqq \{\boldsymbol{g} : \boldsymbol{g} \sim \Pi^* \text{ with label } \boldsymbol{c}\}. \tag{70}$$

Intuitively, the generative process of the modified OU bridge mixture, for which the graph mixture is replaced by $\boldsymbol{D}^{\Pi^*_{\boldsymbol{c}}}(\boldsymbol{G}_t, t)$ is guided by the conditional graph mixture that terminates in the conditioned distribution $\Pi^*_{\boldsymbol{c}}$. We leave this conditional framework as future work.