GLAD: Improving Latent Graph Generative Modeling with Simple Quantization

Van Khoa Nguyen, Yoann Boget, Frantzeska Lavda, Alexandros Kalousis

Geneva School for Business Administration (HES-SO)
University of Geneva, 1214 Geneva, Switzerland
{van-khoa.nguyen, yoann.boget, frantzeska.lavda}@etu.unige.ch, alexandros.kalousis@unige.ch

Abstract

Learning graph generative models over latent spaces has received less attention compared to models that operate on the original data space and has so far demonstrated lacklustre performance. We present GLAD a latent space graph generative model. Unlike most previous latent space graph generative models, GLAD operates on a discrete latent space that preserves to a significant extent the discrete nature of the graph structures making no unnatural assumptions such as latent space continuity. We learn the prior of our discrete latent space by adapting diffusion bridges to its structure. By operating over an appropriately constructed latent space we avoid relying on decompositions that are often used in models that operate in the original data space. We present experiments on a series of graph benchmark datasets that demonstrates GLAD as the first equivariant latent graph generative method achieves competitive performance with the state of the art baselines.

Code — https://github.com/v18nguye/GLAD

Introduction

Graph generation has posed a longstanding challenge with very diverse methods proposed over time. Most of them operate directly over the original data space and learn generative models over node and edge features (Shi et al. 2020; Luo, Yan, and Ji 2021; Kong et al. 2023). Considerably less attention has been given to methods that operate over graph latent spaces and the appropriate definition and design of such latent spaces. Some initial efforts (Simonovsky and Komodakis 2018; Jin, Barzilay, and Jaakkola 2018; Liu et al. 2018; Samanta et al. 2020) propose learning the graph distribution over the continuous latent space of variational autoecoders (VAEs) (Kingma and Welling 2013). Yet, such approaches often suffer from high reconstruction errors, have to solve challenging graph-matching problems, and/or rely on heuristic corrections for the generated graphs.

Diffusion-based models have recently emerged as a compelling approach for modeling graph distributions (Niu et al. 2020; Jo, Lee, and Hwang 2022; Vignac et al. 2023). They allow for the natural incorporation of permutation-invariance as an inductive bias within their denoising component; nevertheless, they are not without limitations. The initial graph diffusion methods rely on continuous score functions to model the graph distributions. They learn these score functions by denoising graph structures that have been noised with real-valued noise (Niu et al. 2020; Jo, Lee, and Hwang 2022). Continuous score functions are a poor match for structures that are inherently discrete. A rather limited number of diffusion-based works treat graphs heads-on as discrete objects (Vignac et al. 2023). Moreover, constructing appropriate score functions over the original graph representation is challenging. Common approaches factorise the score function to node- and adjacency-matrix-specific components, which require their own distinct denoising networks. These partial score functions provide a fragmented view of the graph and do not reflect well the true score function making them a poor choice to model graph distributions. Diffusion-based methods typically have a denoising component which operates over the edges. The denoising process takes place over dense, fully-connected, graphs (Niu et al. 2020; Jo, Lee, and Hwang 2022; Vignac et al. 2023), posing significant scalability challenges (Qin, Vignac, and Frossard 2023).

In this work we start from certain desiderata that a graph generative model should satisfy, which will guide the design of our model. Graph generative models should learn graph distributions that are invariant to node permutations. Graphs should be treated in a holistic manner without relying on artificial decompositions to their components. Finally graphs should be treated as discrete objects, as they are. To address these desiderata we propose **G**raph Discrete **La**tent **D**iffusion (GLAD) a generative model that operates on a carefully designed permutation-equivariant discrete latent space that explicily accounts for the discrete nature of graphs. We quantize our latent node embeddings to preserve their discrete nature in the graph space a fact that ensures efficient graph encoding and supports highly accurate reconstructions, compared to existing methods operating on continuous latent spaces. Subsequently, we tailor diffusion bridges, (Liu et al. 2023) proposed for constrained data domains, to model latent graph distributions over the quantized space. Our latent diffusion model relies on a geometric set of latent nodes, where latent node coordinates encode local graph substructures, and does not resort to a component specific decomposition as other diffusion models operating on the graph space do. To the best of our knowledge, GLAD is

the first equivariant latent graph diffusion model that models in a permutation-invariant manner graph distributions. We carefully evaluate GLAD on a set of graph benchmarks that clearly show that it excels in capturing both chemical and structural properties of graphs compared to state-of-the-art graph generative baselines.

Graph Latent Spaces

In this section, we discuss existing graph generative methods, how they structure their latent spaces, and their limitations that motivated the design of the latent space of GLAD.

Continuous-graph latent spaces Simonovsky and Komodakis were probably the first to use variational autoencoders (VAEs), (Kingma and Welling 2013), for graph generative modeling. They establish a global latent representation of graph by pooling its node embeddings. The posterior distribution of the graph encoding lies in a continuous space. The use of pooling breaks the permutation symmetry and requires solving a graph-matching problem in order to compute the graph-reconstruction loss. Graph-matching scales poorly and is inaccurate for large graphs.

Continuous-node latent spaces Subsequent works, (Li, Zhang, and Liu 2018; Samanta et al. 2020), that follow the VAE approach choose to encode graphs as sets of node embeddings. Operating over node embeddings allows for the use of standard l_2 reconstruction loss and removes the need for solving the graph matching problem. Such approaches typically define the posterior graph distribution as the product of the posterior node distributions, with each posterior node distribution being regularised towards the same prior, typically the standard Gaussian prior. As we will empirically demonstrate the result of this regularisation is that the posterior node distributions are indistinguishable between them. We call this indistinguishability latent-node collapse and show that it drastically hinders graph reconstruction. In Figure 1 we provide a cartoon visualisation of the continuous graph and node latent spaces.

Discrete latent spaces Obtaining distinguishable encodings of local graph structures is necessary if one wants to (i) improve the graph reconstruction performance and (ii) ensure that the learnt generative models capture diverse local graph patterns. These objectives can be achieved with an appropriate design of the discrete latent space, where the encodings of the nodes and their neighborhoods (subgraphs) are embedded in a distinguishable manner in the latent space. The main challenge is to preserve the discrete nature of the graphs in a meaningfully manner when we encode them to the latent spaces. There are few notable works that treat latent graph generative modelling as a discrete problem (Luo, Yan, and Ji 2021; Boget, Gregorova, and Kalousis 2024). The former builds an auto-regressive discrete flow over a discrete latent space defined by a modulo shift transform. The latter relies on vector quantization (Van Den Oord, Vinyals et al. 2017) to encode the latent graph representations using codebooks, and learns the discrete latent graph distribution auto-regressively. While these methods have demonstrated a superior performance over their continuous counterparts (Shi et al. 2020), they still belong to the auto-regressive family and require a canonical-node ordering and as a consequence do not model graph distributions in a permutation-invariant manner. In this work, we introduce a novel discrete latent space for graphs, which extends the finite-quantization (Mentzer et al. 2023) to the graph generation setting. Our graph latent space is simple in design, yet effective in performance. Over the graph latent space we learn an equivariant generative model that models graph distributions in a permutation-invariant manner.

Graph Discrete Latent Diffusion Model

We will now introduce GLAD, the first graph equivariant generative model that operates on a discrete latent space of graphs. We first present our mathematical notations, then define a novel discrete graph latent space, and show how to learn a prior over the defined latent space using diffusion bridges.

Notation

We denote a graph by the tuple $\mathcal{G} = (X, E)$; $X = \{x_i\}_{i=1}^N$ is the set of the node feature vectors, $E = \{e_{ij}\}_{i,j \in N}$ is the set of edge feature vectors and N is the number of nodes. We rely on an autoencoder (AE) to embed graphs to a latent space. The encoder (ϕ)-, and decoder (θ)- are equivariant graph neural networks (E-GNNs). In the latent space, we denote by $Z_{raw} = \{z_i\}_{i=1}^N$ the set of node embeddings, where $z_i \in \mathbb{R}^f$ corresponds to the embedding of the i^{th} graph node.

GLAD's Discrete Graph Latent Space

In constructing the latent space of GLAD we want to satisfy the following desiderata: i) ensure that we learn graph distributions that are invariant to permutations, ii) encode nodes and their local structures in a way that preserves their structural specificities, and iii) preserve the discrete nature of the objects we encode, nodes and their neighborhoods, and that of the graph itself.

We use an E-GNN encoder, ϕ, to map a graph to the set of the continuous node embeddings $Z_{raw} = \{z_i\}_{i=1}^N := \phi(\mathcal{G})$. The encoder captures node and node-neighborhood information. We then apply the quantization operator, $\mathcal{F}$, on the continuous node embeddings and map them to a quantized space of the same dimension. We follow (Mentzer et al. 2023) to design the quantisation $\mathcal{F}$, which we apply independently on each latent node dimension. The quantization of the j^{th} dimension of the z_i node embedding is given by:

$$z_{ij}^q = \mathcal{F}(z_{ij}, L_j) = \mathcal{R}(\frac{L_j}{2} \tanh z_{ij}),\ 1 \leqslant j \leqslant f \qquad (1)$$

Where $\mathcal{R}$ is the rounding operator, and L_j is the number of quantization levels on the j^{th} latent node dimension. We will denote by $[L_j]$ the set of pre-defined quantization levels: $[-\mathcal{R}(L_j/2), \cdots, -1, 0, 1, \cdots, \mathcal{R}(L_j/2)]$. The quantization creates a new set of now discrete latent node embeddings $Z_{\mathcal{G}} = \{z_i^q\}_{i=1}^N$ with $z_i^q \in \mathbf{S}$; $\mathbf{S}$ is the discrete latent space which is given by the Cartesian product $\mathbf{S} := \prod_{j=1}^f [L_j] = \{l_1 \times \cdots \times l_f : \forall l_j \in [L_j]\}$, we denote by K the cardinality of $\mathbf{S}$.

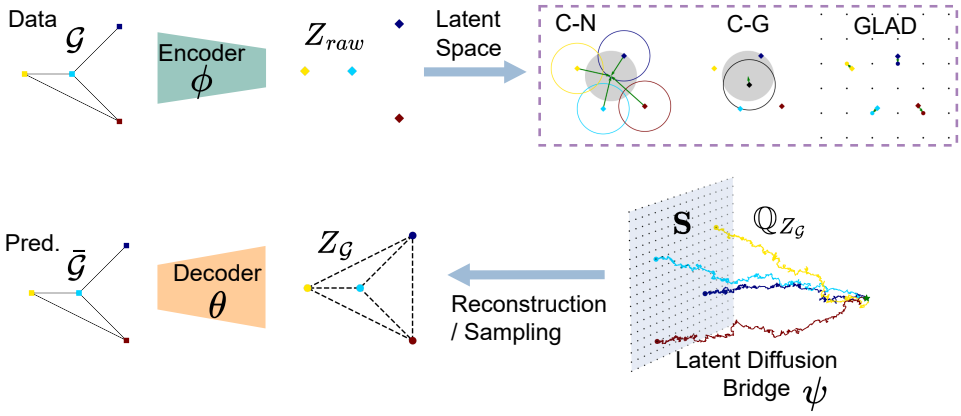

Figure 1: (Top) We encode an annotated graph with node features (colored squares) to a set of node embeddings (colored diamonds). We visualize three studied graph-latent spaces: continuous-graph latent (C-G), continuous-node latent (C-N), and quantised latent (GLAD) spaces. The grey disks denote normal priors, the colored circles denote posterior distributions, and the green arrows denote pushing forces. GLAD distinctively maps the raw-node embeddings to discrete points on a uniformly quantized space (black dots). (Bottom) We decode a fully-connected pseudo latent graph (dashed-edge graph) formed by the set of quantized latent nodes $Z_{\mathcal{G}}$ (colored dots) to the original graph space. We leverage diffusion bridges to learn the graph-latent discrete distribution Π constrained over the quantized-latent space $\mathbf{S}$. The colored paths represent an example of $Z_{\mathcal{G}}$-bridge.

The operator $\mathcal{F}$ is permutation equivariant and so is the mapping from the initial graph $\mathcal{G}$ to its quantized latent representation as long as the ϕ graph encoder is permutation equivariant. Thus for any P permutation matrix the following equivariance relation holds:

$$P^T Z_{\mathcal{G}} = \mathcal{F}(P^T Z) = \mathcal{F}(\phi(P^T E P, P^T X)) \quad (2)$$

We apply an equivariant decoder, $\theta(Z_{\mathcal{G}})$, assuming a fully-connected graph on the quantized latent nodes $Z_{\mathcal{G}}$. Our graph autoencoder is composed exclusively of equivariant components. Therefore, we ensure that decoded distributions are invariant to the graph node permutations. The decoded nodes have a one to one relation to the nodes of the original graph, making it trivial to define a reconstruction loss for the autoencoder. We will now show how to use diffusion bridges to learn the discrete latent graph distribution on the latent space induced by our autoencoder.

Learning Graph Prior with Diffusion Bridges

For a given graph, the quantization maps the set of original node embeddings to points in the discrete latent space $\mathbf{S}$. The latent-graph domain Ω becomes a set structure that we can decompose into a node-wise manner as $\Omega = I_1 \times \cdots \times I_N$, where $I_i = \mathbf{S}$. We extend diffusion bridges (Liu et al. 2023), which learn data distributions on constrained domains, to model our latent-graph structure Ω. To do so we will first define a conditional diffusion process, conditioned on a given latent graph structure $Z_{\mathcal{G}}$. We then build the diffusion bridge for our latent graph distribution Π as the mixture of conditional diffusion processes defined over the training data.

We start by defining a Brownian motion-based non-conditional diffusion process given by the stochastic differential equation (SDE):

$$\mathbb{Q} : dZ_t = \sigma(Z_t, t) dW_t \quad (3)$$

where W_t is a Wiener process and $\sigma : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}$ is a diffusion coefficient.

$Z_{\mathcal{G}}$-conditioned diffusion bridge We will now define a conditional diffusion bridge where the endpoint Z_T of the diffusion process will be the conditioning factor $Z_{\mathcal{G}}$, i.e. $Z_T = Z_{\mathcal{G}}$. We denote the $Z_{\mathcal{G}}$-conditioned bridge by $\mathbb{Q}^{Z_{\mathcal{G}}}$, the dynamics of which can be derived from the *h-transform* (Oksendal 2013):

$$\mathbb{Q}^{Z_{\mathcal{G}}} : dZ_t = \eta^{Z_{\mathcal{G}}}(Z_t, t) dt + \sigma(Z_t, t) dW_t, \quad Z_0 \sim \mathbb{P}_0 \quad (4)$$

where $\mathbb{P}_0$ is the prior distribution of the $Z_{\mathcal{G}}$-conditione bridge. The $Z_{\mathcal{G}}$-conditioned bridge's drift is defined as $\eta^{Z_{\mathcal{G}}}(Z_t, t) = \sigma^2(Z_t, t) \nabla_{Z_t} \log q_{T|t}(Z_{\mathcal{G}}|Z_t)$. $q_{T|t}(Z_{\mathcal{G}}|Z_t)$ is the probability density of obtaining $Z_{\mathcal{G}}$ at time T when we have Z_t at time t; $\nabla_{Z_t} \log q_{T|t}(Z_{\mathcal{G}}|Z_t)$ acts as a steering force, which is central to guiding Z_t towards our specified target, $Z_T = Z_{\mathcal{G}}$.

Π-conditioned diffusion bridge Given the $Z_{\mathcal{G}}$-conditioned bridge definition, we can now construct a bridge process on the discrete latent-graph distribution Π, which is constrained over the latent-graph domain Ω. We call this bridge process a Π-bridge and denote it by $\mathbb{Q}^{\Pi}$. We construct the Π-bridge as a mixture of $Z_{\mathcal{G}}$-conditioned bridges; their end-points are conditioned on latent-graph samples $\{Z^i_{\mathcal{G}}\}$ i.i.d drawn from the latent graph distribution Π. The Π-bridge's dynamics are governed by the SDE:

$$\mathbb{Q}^{\Pi} : dZ_t = \eta^{\Pi}(Z_t, t) dt + \sigma(Z_t, t) dW_t \quad (5)$$

The drift is given by $\eta^{\Pi}(Z_t, t) = \sigma^2(Z_t, t) \mathbb{E}_{\omega \sim q_{T|t,\Omega}(\omega|Z_t)}[\nabla_{Z_t} \log q_{T|t}(\omega|Z_t)]$, where ω is sampled from a transition probability density given by $q_{T|t,\Omega}(\omega|Z_t) = q(Z_T = \omega|Z_t, Z_T \in \Omega)$. We see that the

Algorithm 1: Graph Discrete Latent Diffusion Bridge

Input Encoder ϕ, decoder θ, bridge model ψ
// Stage 1: Learn graph autoencoder (ϕ, θ)
for batch $\mathcal{G} = (X, E)$:
$Z_{raw} \leftarrow \phi(\mathcal{G})$
$Z_{\mathcal{G}} \leftarrow \mathcal{F}(Z_{raw}, L)$
$Z_{\mathcal{G}} \leftarrow$ *straight-through-estimator*$(Z_{\mathcal{G}}, Z_{raw})$
$\mathcal{L}_1 \leftarrow \|X - \theta_X(Z_{\mathcal{G}})\|^2 + \|E - \theta_E(Z_{\mathcal{G}})\|^2$
$\phi, \theta \leftarrow$ *Adam-optim* $(\nabla_{\phi,\theta}(\mathcal{L}_1))$
// Stage 2: Learn bridge model (ψ)
for batch $(Z_{\mathcal{G}}, t \sim [0, T], Z_0 \sim \mathbb{P}_0)$:
$Z_t \leftarrow \frac{\beta_t}{\beta_T}(Z_{\mathcal{G}} + (\beta_T - \beta_t)Z_0) + \xi\sqrt{\beta_t\left(1 - \frac{\beta_t}{\beta_T}\right)}$
$\left[\eta^{Z_{\mathcal{G}}}\right]^{I_i} \leftarrow \sigma_t^2 \frac{Z_{\mathcal{G}}^{I_i} - Z_t^{I_i}}{\beta_T - \beta_t} \quad \forall I_i \in \Omega$
$\left[\eta^{\Pi}\right]^{I_i} \leftarrow \sigma_t^2 \nabla_{Z_t^{I_i}} \log \sum_{s_k \in \mathbf{S}} \exp\left(-\frac{\|Z_t^{I_i} - s_k\|^2}{2(\beta_T - \beta_t)}\right)$
$\mathcal{L}_2 \leftarrow \sum_{I_i \in \Omega} \left\| [\psi(Z_t, t)]^{I_i} - \sigma_t^{-1}\left(\left[\eta^{Z_{\mathcal{G}}}\right]^{I_i} - \left[\eta^{\Pi}\right]^{I_i}\right)\right\|^2$
$\psi \leftarrow$ *Adam-optim* $(\nabla_{\psi}(\mathcal{L}_2))$
// Sampling novel graphs
$t \leftarrow 0, \delta_t \leftarrow \frac{1}{T}, Z_0 \sim \mathbb{P}_0$
while $(t < T)$:
$\xi \sim \mathcal{N}(0, I)$
$Z_{t+1} \leftarrow Z_t + \left(\sigma_t \psi(Z_t, t) + \eta^{\Pi}(Z_t, t)\right)\delta_t + \sigma_t\sqrt{\delta_t}\xi$
$\mathcal{G}_{new} \leftarrow \theta(Z_T)$

discrete latent-graph structure Ω is taken into consideration in the expectation of the steering forces. The marginal distribution, $\mathbb{Q}_T^{\Pi}$, induced by the Π-bridge at $t = T$ will match the latent graph distribution Π by construction, i.e. $\mathbb{Q}_T^{\Pi} = \Pi$.

Model bridge We will train a bridge, $\mathbb{P}^{\psi}$, to fit, and generalise from, the Π-bridge dynamics and generate new latent-graph samples from the Π distribution. The dynamics of $\mathbb{P}^{\psi}$ are given by the parametric SDE:

$$\mathbb{P}^{\psi} : dZ_t = \eta^{\psi}(Z_t, t)dt + \sigma(Z_t, t)dW_t \tag{6}$$

where the drift term is $\eta^{\psi}(Z_t, t) = \sigma(Z_t, t)\psi(Z_t, t) + \eta^{\Pi}(Z_t, t)$; ψ is a neural network. The Π-bridge imputes noise to latent-graph samples at different time steps which we use to learn the model bridge $\mathbb{P}^{\psi}$. We train $\mathbb{P}^{\psi}$ by minimizing the Kullback-Leibler divergence between the two probability path measures $\min_{\psi}\{\mathcal{L}(\psi) := \mathcal{KL}(\mathbb{Q}^{\Pi}||\mathbb{P}^{\psi})\}$. The Girsanov theorem, (Lejay 2018), allows us to compute the $\mathcal{KL}$ divergence with the following closed form solution:

$$\mathcal{L} = \mathbb{E}_{Z_{\mathcal{G}} \sim \Pi, t \sim [0,T], Z_t \sim \mathbb{Q}^{Z_{\mathcal{G}}}}\left[\|\psi(Z_t, t) - \bar{\eta}(Z_t, t)\|^2\right] \tag{7}$$

with $\bar{\eta}(Z_t, t) = \sigma^{-1}(Z_t, t)(\eta^{Z_{\mathcal{G}}}(Z_t, t) - \eta^{\Pi}(Z_t, t))$, where $\eta^{Z_{\mathcal{G}}}$, η^{Π} are introduced in Equation 4 and Equation 5, respectively. Since we use Brownian motion for the non-conditional diffusion $\mathbb{Q}$, we can retrieve its closed-form perturbation kernel and accordingly derive the dynamics of the $Z_{\mathcal{G}}$-bridge's drift as:

$$\eta^{Z_{\mathcal{G}}}(Z_t, t) = \sigma_t^2 \frac{Z_{\mathcal{G}} - Z_t}{\beta_T - \beta_t}, \ \beta_t = \int_0^t \sigma_s^2 ds \tag{8}$$

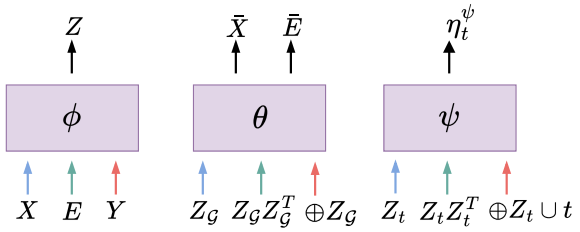

Figure 2: Graph transformer-based architectures. (Left) graph encoder, (Middle) graph decoder, (Right) model bridge drift. We denote $\oplus$ as pooling operator on a set of latent nodes, and $\cup$ as concatenation operator.

Where we simply define the diffusion coefficient σ_t that depends only on the time variable. As our latent-graph structure Ω can be factorized into the latent-structure of nodes, the expectation over Ω , Equation 5, can simplify to one-dimensional integrals over $\mathbf{S}$. And the Π-bridge's drift can be decomposed into the latent-node structures:

$$\eta^{\Pi}(Z_t, t) = \left[\left[\eta^{\Pi}(Z_t^i, t)\right]^{I_i}\right]_{i=1}^{N}$$
$$\left[\eta^{\Pi}(Z_t^i, t)\right]^{I_i} = \sigma_t^2 \nabla_{Z_t^i} \log \sum_{k=1}^{K} \exp\left(-\frac{\|Z_t^i - s_k\|^2}{2(\beta_T - \beta_t)}\right) \tag{9}$$

where $s_k \in \mathbf{S}$, Z_t^i is the feature of i^{th} latent node of Z_t. We apply the drift terms obtained above to Equation 7 to get the closed-form objective.

Training We train GLAD in two stages. First, we train a graph autoencoder that learns a structural mapping from the graph space to the designed latent space. As a technical detail, we note that the rounding operator $\mathcal{R}$ of the quantization is non-differentiable. To ensure end-to-end training of our discrete graph latent space, we use the straight-through estimator (STE) (Bengio, Léonard, and Courville 2013), implemented with a stop-gradient operator $\mathcal{S}$ as $Z_{\mathcal{G}} \mapsto Z + \mathcal{S}(Z_{\mathcal{G}} - Z)$. We freeze the graph-autoencoder while training the model bridge in the second stage. GLAD learns the graph latent distributions constrained over the quantized space defined on the graph-encoder output and the quantization operator. At sampling, we apply a simple discretization scheme Euler–Maruyama on the model bridge. We summarise our training procedure in Algorithm 1.

Architectures We learn data distributions that lie on both graph- and set-based structures in GLAD. As sets are more or less considered as fully-connected graphs. We opt to employ a general-purpose architecture such as graph transformers (Dwivedi and Bresson 2020). Future works could consider a more specific choice of architecture designed to graph- or set-related domains. We adapt the graph transformer (Vignac et al. 2023) and further customize the architecture for our graph encoder ϕ, graph decoder θ, and drift model ψ. We visualize our architectures in Figure 2. The graph encoder takes a node feature matrix X, an edge feature matrix E, and a spectral feature vector Y as inputs. Due

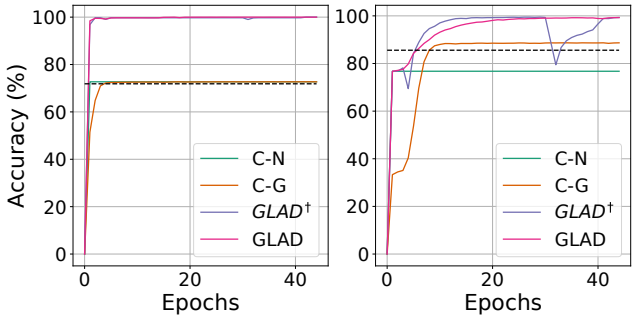

Figure 3: Molecule reconstruction from different latent spaces on QM9. Atom (Left) and bond (Right) type test accuracies: i) continuous-graph latent representation (C-G). ii) continuous-node latent representation (C-N) iii) unquantized discrete-node latent space GLAD†, iv) quantised discrete-node latent space GLAD. Dashed back lines denote the percentage of dominant atom (Carbon) and bond (Single) types.

to the inherent limitations of most message-passing neural networks (Chen et al. 2020), we compute structural features like cycle counts and spectral features of graph-Laplacian decomposition as hand-crafted input features Y. The encoder outputs a set of raw node embeddings that is quantized to obtain a set of quantized latent nodes $Z_{\mathcal{G}}$. The graph decoder takes a pseudo-latent graph, which is formed by the latent nodes $Z_{\mathcal{G}}$, a pseudo-adjacency matrix $Z_{\mathcal{G}} Z_{\mathcal{G}}^T$; in addition, we incorporate a global representation feature to the decoder by pooling the set of latent nodes $\oplus Z_{\mathcal{G}}$. The drift model is similar in the design of graph decoder, however it takes a noisy set of latent nodes Z_t sampled at different time step t from $Z_{\mathcal{G}}$-bridge. The model learns the dynamic of Π-bridge drift by η_t^{ψ}, which structure is identical to $Z_{\mathcal{G}}$.

Experiments

Graph Reconstruction

Setup We claim that our quantized-graph latent space is pivotal to the good performance of GLAD. We now provide empirical evidence to support our claim. We compare the reconstruction performance of GLAD applied on its quantized graph latent space to the following latent spaces and methods: i) continuous-graph latent space (C-G); we train a vanilla VAE on a global structure, pooled latent nodes, to get a continuous-latent representation ii) continuous-node latent space (C-N); similar to i) we also train a vanilla VAE, however applied on a set of latent nodes. Finally in iii) we simply use a set of raw node embeddings without any constraints imposed to the graph-latent space (GLAD†), akin to a standard autoencoder for graphs. We evaluate the quality of the underlying graph latent spaces using their bond-type (X) and atom-type (E) reconstruction accuracies on QM9.

Results We show the reconstruction ability per latent structure in Figure 3. The two continuous latent spaces (C-G, C-N) have considerably lower accuracies compared to the two GLAD variants, which preserve to a different extent the original discrete graph topology. Both GLAD variants converge very fast to nearly 100% reconstruction accuracy. GLAD has a small advantage when it comes to the bond type (E), for which its training is more stable than that of its unquantized sibling GLAD†. The latent spaces of C-G and C-N suffer from the so-called latent-node-collapsing problem. They are not able to well describe different atom local structures. As a result, most reconstructed nodes are assigned to the dominant Carbon atom, roughly corresponding to the 72% of atoms in QM9.

Generic Graph Generation

Setup We measure GLAD's ability to capture the underlying structures of generic graphs on three datasets: (a) ego-small (Sen et al. 2008), (b) community-small, and (c) enzymes (Schomburg et al. 2004). On these graphs, there is no explicit information about nodes available for training. We therefore use node degrees as augmented-node features to the graph-encoder's inputs. We use the same train- and test-splits as the baselines for a fair comparison. We evaluate model performance by computing the maximum-mean discrepancy (MMD) between the generated-set and test-set distributions of the following graph statistics: degree (Deg. $\downarrow$), clustering coefficient (Clus. $\downarrow$), and orbit 4-node occurrences (Orb. $\downarrow$). At each evaluation, we generate an equal number of graphs to the current test set. We adopt the Gaussian Earth Mover's Distance (EMD) kernel to calculate MMD thanks to its computational stability.

Baselines Here under the approach working directly on graph space, we compare to auto-regressive GraphRNN (You et al. 2018), auto-regressive flow GraphAF (Shi et al. 2020), auto-regressive diffusion GraphArm (Kong et al. 2023), continuous diffusion GDSS (Jo, Lee, and Hwang 2022), and discrete diffusion DiGress (Vignac et al. 2023). Under the latent-space direction, we have GraphVAE (Simonovsky and Komodakis 2018), continuous flow GraphNF(Liu et al. 2019), discrete flow GraphDF (Luo, Yan, and Ji 2021), and discrete autoencoder DGAE (Boget, Gregorova, and Kalousis 2024). In all baselines, we use the continuous- and discrete- terms with an implicit indication on how data represented or treated by individual approach.

Results We observe in Table 1 that our model achieves competitive performance compared to the state-of-the-art baselines; it consistently maintains the lowest-average MMD distance across three datasets. Moreover, GLAD significantly outperforms all baselines that make explicit use of latent-space structures, namely GraphVAE, GNF, GraphDF, and DGAE. These accomplishments would highlight two pivotal strengths of GLAD: first, our quantized graph-latent space can encode rich local graph sub-structures, serving as a cornerstone for any latent-driven generative models. Second, the diffusion bridges work seamlessly within the proposed latent structure by design, which is underscored by its capability to learn the set-based latent representation of graphs. While GLAD does not operate directly on the graph space like GDSS and DiGress, which are also

	COMMUNITY-SMALL				EGO-SMALL				ENZYMES			
	Deg. ↓	Clus. ↓	Orb. ↓	AVG	Deg. ↓	Clus. ↓	Orb. ↓	AVG	Deg. ↓	Clus. ↓	Orb. ↓	AVG
GraphRNN	0.080	0.120	0.040	0.080	0.090	0.220	0.003	0.104	0.017	0.062	0.046	0.042
GraphAF	0.180	0.200	0.020	0.133	0.030	0.110	**0.001**	0.047	1.669	1.283	0.266	1.073
GDSS	0.045	0.086	0.007	0.046	0.021	0.024	0.007	0.017	0.026	0.061	0.009	0.032
DiGress	0.047	**0.041**	0.026	0.038	0.015	0.029	0.005	0.016	**0.004**	0.083	0.002	0.030
GraphArm	0.034	0.082	**0.004**	0.040	0.019	0.017	0.010	0.015	0.029	0.054	0.015	0.033
GraphVAE	0.350	0.980	0.054	0.623	0.130	0.170	0.050	0.117	1.369	0.629	0.191	0.730
GNF	0.200	0.200	0.110	0.170	0.030	0.100	**0.001**	0.044	-	-	-	-
GraphDF	0.060	0.120	0.030	0.070	0.040	0.130	0.010	0.060	1.503	1.061	0.202	0.922
DGAE	0.032	0.062	0.005	0.033	0.021	0.041	0.007	0.023	0.020	0.051	0.003	0.025
GLAD	**0.029**	0.047	0.008	**0.028**	**0.012**	**0.013**	0.004	**0.010**	0.012	**0.014**	**0.001**	**0.009**

Table 1: Generation results on the generic graph datasets. We show the mean values of 15 runs for each experiment. The baselines are sourced from (Jo, Lee, and Hwang 2022; Kong et al. 2023). We compare with generative models operating on graph space (top row) and on latent-graph space (bottom row). Hyphen (-) denotes unreproducible results. The 1st and 2nd best results are bolded and underlined, respectively.

	QM9					ZINC250K				
	Val. ↑	Uni. ↑	Nov. ↑	NSPDK ↓	FCD ↓	Val. ↑	Uni. ↑	Nov. ↑	NSPDK ↓	FCD ↓
GraphAF	74.43	88.64	86.59	0.020	5.27	68.47	98.64	100	0.044	16.02
MoFlow	91.36	98.65	94.72	0.017	4.47	63.11	99.99	100	0.046	20.93
GDSS	95.72	98.46	86.27	0.003	2.90	97.01	99.64	100	0.019	14.66
DiGress	99.0	96.66	33.40	0.0005	0.360	91.02	81.23	100	0.082	23.06
GraphArm	90.25	95.62	70.39	0.002	1.22	88.23	99.46	100	0.055	16.26
GraphDF	93.88	98.58	98.54	0.064	10.93	90.61	99.63	100	0.177	33.55
DGAE	92.0	97.61	79.09	0.0015	0.86	77.9	99.94	99.97	0.007	4.4
GLAD	97.12	97.52	38.75	**0.0003**	**0.201**	81.81	100	99.99	**0.002**	**2.54**

Table 2: Generation results on the molecule datasets. We show the mean values of 3 runs for each experiment. The baselines are sourced from (Jo, Lee, and Hwang 2022; Kong et al. 2023). We highlight the most important metrics; the 1st and 2nd best results are bolded and underlined, respectively. We compare with generative models operating on graph space (top row) and on latent-graph space (bottom row).

diffusion-based frameworks, GLAD demonstrates superior performance to capture the underlying topologies of graphs in a holistic manner.

Molecule Graph Generation

Setup We evaluate the capacity of GLAD to capture the complex dependencies between atoms and bonds as they appear in different molecules. We conduct experiments on two standard datasets: QM9 (Ramakrishnan et al. 2014) and ZINC250k (Irwin et al. 2012). Following (Jo, Lee, and Hwang 2022), we remove hydrogen atoms and kekulize molecules by RDKit Landrum et al. (2016). We quantitatively evaluate all methods in terms of validity without post-hoc corrections (Val. ↑), uniqueness (Uni. ↑), and novelty (Nov. ↑) of 10.000 generated molecules. In addition, we compute two more salient metrics that quantify how well the distribution of the generated graphs aligns with the real data distribution, namely Fréchet ChemNet Distance (FCD ↓) and Neighborhood Subgraph Pairwise Distance Kernel (NSPDK ↓). FCD evaluates the distance in the chemical space between generated and training graphs by using the activation of the ChemNet's penultimate layer, while NSPDK measures the MMD distance, showing the similarity of underlying structures between generated and test molecules. These last two metrics show the most relevant comparisons as they directly take into account the distribution of molecular properties, e.g chemical and structural, rather than with only molecule-graph statistics.

Baselines We compare GLAD with generative models that operate on graph spaces, including continuous flow GraphAF (Shi et al. 2020), conditional flow MoFlow (Zang and Wang 2020), continuous diffusion GDSS (Jo, Lee, and Hwang 2022), discrete diffusion DiGress (Vignac et al. 2023), and auto-regressive diffusion GraphArm (Kong et al. 2023). We also compare with generative models that work with discrete latent structures, including auto-regressive flow GraphDF (Luo, Yan, and Ji 2021), and auto-regressive autoencoder DGAE (Boget, Gregorova, and Kalousis 2024).

Prior	QM9				
	Val. ↑	Uni. ↑	Nov. ↑	NSPDK ↓	FCD ↓
$\mathcal{N}(\mathbf{0}, \mathbf{1})$	95.38	97.38	41.54	0.0004	0.330
$\mathbf{0}$	97.12	97.52	38.75	0.0003	0.201
$\mathbf{0}^{\dagger}$	83.12	97.54	62.44	0.0009	1.207

Table 3: Ablations on prior $\mathbb{P}_0$ and quantization $\mathcal{F}$. Generation results when the prior distribution is initialized as a standard normal distribution and a Dirac Δ distribution on the fixed point $\mathbf{0}$; the latter is ablated with and without quantization, denoted by $\mathbf{0}$ and $\mathbf{0}^{\dagger}$ respectively.

Results Table 2 show the generation results on molecules. GLAD is the first one-shot model that learns to generate molecular structures from a discrete latent space. Even though our model does not learn directly on atom- and bond-type structures, GLAD can still generate molecules with good validity scores without corrections. Importantly, GLAD achieves state of the art performance on the two most salient metrics NSPDK and FCD that capture well both structural and chemical properties of molecule distributions, consistently outperforming by significant margins all baselines. It demonstrates that our quantized latent space offers a suitable discrete structure to encode those properties. We additionally compare the latent node distributions between train and sampled molecules in Figure 4, which shows the GLAD's bridge model is able of capturing well the latent node distributions of different datasets.

Ablation Studies

Quantisation effect on generation We study the effect of quantisation by removing the quantization step. Node embeddings remain discrete-, but non-quantised- structures, and they are embedded into a continuous domain, which has the same dimensionality to the quantized one f, denoted as $\Omega_c = [L_{\min}, L_{\max}]^{(N\times f)}$. We retrain our graph autoencoder, drift model , and obtain $L_{\min}, L_{\max}$ that correspond to the min- and max- values of learned latent nodes for the entire training set. The Π-bridge's drift on a continuous domain is computed as:

$$\left[\eta^{\Pi}\right]^h = \sigma_t^2 \nabla_{Z_t^h} \log\left(F(\frac{Z_t^h - L_{\min}}{\sqrt{\beta_T - \beta_t}}) - F(\frac{Z_t^h - L_{\max}}{\sqrt{\beta_T - \beta_t}})\right)$$
$$\forall h, 1 \leqslant h \leqslant (N \times f) \quad (10)$$

Where $Z_t \in [L_{\min}, L_{\max}]^{(N\times f)}$, F is the standard Gaussian Cumulative Distribution Function (CDF). As per Figure 3, the two GLAD variants have a rather similar reconstruction performance, but this is not the case when we compare their generation performance, described in Table 3. There we see that the diffusion bridges can not capture well the graph distribution when they operate on the unquantized space, with a performance drop for most measures (importantly on the two measures capturing molecular distributions, NSPDK and FCD), showing the benefits of the quantised discrete latent space. Quantization acts as a spatial regulariser over the non-quantised structures by constraining latent nodes on high-dimensional discretized grid instead of letting them localized in a infinite-continuous space. We hypothesize this spatial regularisation is non-trivial to construct effective latent structures for graphs.

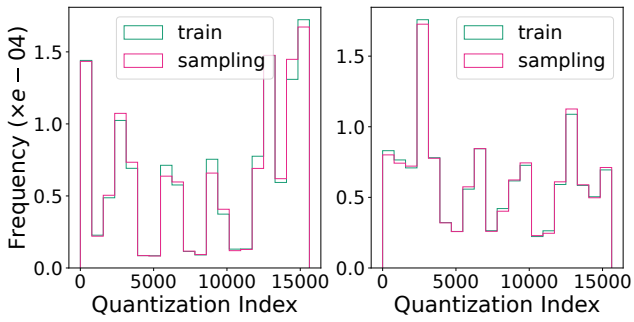

Figure 4: Comparison of latent node distributions on the quantized-grid structure $\mathbf{S}$ of train and sampled molecules, QM9 (Left) and ZINC250k (Right).

Choice of bridge priors Here we evaluate model performance on different priors; a fixed point prior $\mathbb{P}_0 = \mathbf{0}$, which is the central mass of our quantized latent space, and a standard normal distribution prior, $\mathbb{P}_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$. We conduct the experiments on QM9 and give the results in Table 3. As a result of the steering forces of the bridge processes, there are only negligible differences between the two priors making prior selection less of a nuance. With the fixed prior, we obtain generated molecules with higher validity scores, closer to the training distribution in both chemical space (FCD score), structural space (NSPDK score). Using the normal prior we have slightly better performance in terms of novelty score.

Conclusion

We present the first equivariant graph diffusion model that operates over a discrete latent space. By the careful design, our graph-latent structure satisfies certain desiderata that stem from the graph nature, namely it should allow for learning graph-permutation-invariant distributions, being rich representational power, and respecting the inherently discrete nature of graphs. We empirically validate GLAD on a number of benchmarks and show that it achieves state of the art generative performance, surpassing other baselines independently of whether they operate on the original or on latent spaces. We systematically ablated different design choices for graph latent space representation and show that our discrete latent space brings clear performance advantages over continuous alternatives both in terms of reconstruction as well as generation. This paves the way for a more systematic exploration of graph generative modelling in latent spaces, something that until now has received rather limited attention.

Acknowledgements

We acknowledge the financial support of the Swiss National Science Foundation within the LegoMol project (grant no. 207428). The computations were performed at the University of Geneva on Baobab and Yggdrasil HPC clusters.

References

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Boget, Y.; Gregorova, M.; and Kalousis, A. 2024. Discrete Graph Auto-Encoder. *Transactions on Machine Learning Research*.

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J. 2020. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33: 10383–10395.

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

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

Jin, W.; Barzilay, R.; and Jaakkola, T. 2018. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, 2323–2332. PMLR.

Jo, J.; Lee, S.; and Hwang, S. J. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, 10362–10383. PMLR.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kong, L.; Cui, J.; Sun, H.; Zhuang, Y.; Prakash, B. A.; and Zhang, C. 2023. Autoregressive diffusion model for graph generation. In *International Conference on Machine Learning*, 17391–17408. PMLR.

Landrum, G.; et al. 2016. Rdkit: Open-source cheminformatics software, 2016. *URL http://www. rdkit. org/, https://github. com/rdkit/rdkit*, 149(150): 650.

Lejay, A. 2018. The Girsanov theorem without (so much) stochastic analysis. *Séminaire de Probabilités XLIX*, 329–361.

Li, Y.; Zhang, L.; and Liu, Z. 2018. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10: 1–24.

Liu, J.; Kumar, A.; Ba, J.; Kiros, J.; and Swersky, K. 2019. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. 2018. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31.

Liu, X.; Wu, L.; Ye, M.; and qiang liu. 2023. Learning Diffusion Bridges on Constrained Domains. In *The Eleventh International Conference on Learning Representations*.

Luo, Y.; Yan, K.; and Ji, S. 2021. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, 7192–7203. PMLR.

Mentzer, F.; Minnen, D.; Agustsson, E.; and Tschannen, M. 2023. Finite Scalar Quantization: VQ-VAE Made Simple. *arXiv preprint arXiv:2309.15505*.

Niu, C.; Song, Y.; Song, J.; Zhao, S.; Grover, A.; and Ermon, S. 2020. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, 4474–4484. PMLR.

Oksendal, B. 2013. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media.

Qin, Y.; Vignac, C.; and Frossard, P. 2023. Sparse Training of Discrete Diffusion Models for Graph Generation. *arXiv preprint arXiv:2311.02142*.

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

Samanta, B.; De, A.; Jana, G.; Gómez, V.; Chattaraj, P.; Ganguly, N.; and Gomez-Rodriguez, M. 2020. Nevae: A deep generative model for molecular graphs. *Journal of machine learning research*, 21(114): 1–33.

Schomburg, I.; Chang, A.; Ebeling, C.; Gremse, M.; Heldt, C.; Huhn, G.; and Schomburg, D. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1): D431–D433.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shi, C.; Xu, M.; Zhu, Z.; Zhang, W.; Zhang, M.; and Tang, J. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*.

Simonovsky, M.; and Komodakis, N. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, 412–422. Springer.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

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

You, J.; Ying, R.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, 5708–5717. PMLR.

Zang, C.; and Wang, F. 2020. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 617–626.