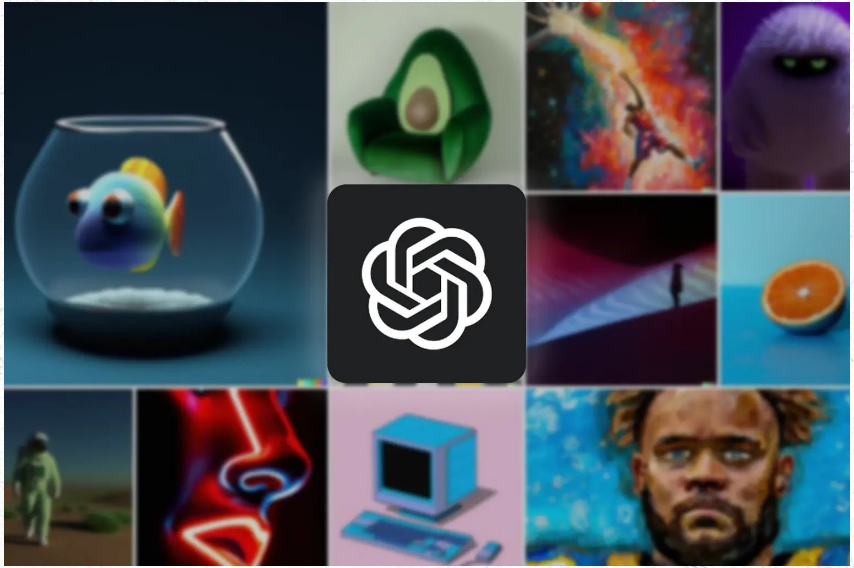
Latent 3D Graph Diffusion

Yuning You[1], Ruida Zhou[2], Jiwoong Park[1], Haotian Xu[1], Chao Tian[1], Zhangyang Wang[3], Yang Shen[1]

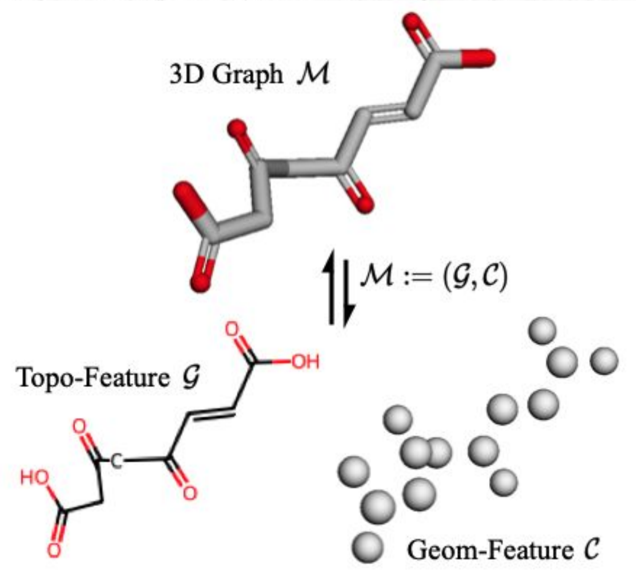
[1]Texas A&M University, [2]University of California, Los Angeles, [3]University of Texas at Austin

Background

Generative AI on Images



Generative AI on 3D Graphs

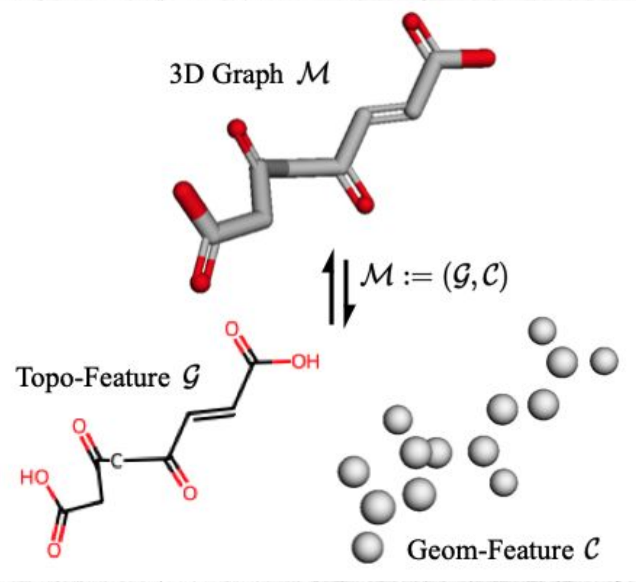


Central Question

In what (latent) space should we learn the distribution of graphs with neural networks?

- Original space: Topology⊗Geometry;
- Symmetry constraint: Data identities are invariant to certain transformations;
- High dimensionality: Data dimensionality depends on graph size;
- Low-dimensional manifold: Data are supposed to distribute in the LD manifold.

Generative AI on 3D Graphs



Answer: What justifies a “good” space for graph generative models?

- We focus on diffusion generative model.
- Assuming there are forward and reverse mappings between the original 3D graph space and a latent space;
- The diffusion model is trained on the latent space to capture distribution, then:

$$\text{3D Graph Diffusion Performance} \leq \text{Latent Space Reconstruction Quality} + \text{Symmetry Preservation} \times \text{Data Dimensionality.}$$

Proposition 2. (3D graph diffusion could benefit from the **lower-dimensional latent space if appropriately constructed**. See proof in Append. A.2) Assume there existing mappings $\overrightarrow{h} : \mathbb{R}^{D'} \to \mathbb{R}^{D''}$, $\overleftarrow{h} : \mathbb{R}^{D''} \to \mathbb{R}^{D'}$ that $D'' < D'$ and $\overleftarrow{h}$ is injective. Assume DGM now is trained in $\mathbb{R}^{D''}$ to model $\overrightarrow{p}_{\mathrm{data}}(\mathbf{z}) = \Pr\{\mathbf{x}_{\mathrm{M}} : \overrightarrow{h}(\mathbf{x}_{\mathrm{M}}) = \mathbf{z}, \mathbf{x}_{\mathrm{M}} \sim p_{\mathrm{data}}\}$ with $p_\theta(\mathbf{z})$, and it is evaluated in $\mathbb{R}^{D'}$ on $\overleftarrow{p}_\theta([\mathbf{x}_{\mathrm{M}}]_{\Pi,\Omega}) = \Pr\{\mathbf{z} : \overleftarrow{h}(\mathbf{z}) \in [\mathbf{x}_{\mathrm{M}}]_{\Pi,\Omega}, \mathbf{z} \sim p_\theta\}$ (as in Propos. 1), and the assumptions in Propos. 1 retain for the score estimator f_θ and mapping distribution. Then, it holds:

$$\mathrm{TV}(\overleftarrow{p}_\theta, \tilde{p}_{\mathrm{data}}) \lesssim \mathrm{TV}(\overleftrightarrow{p}_{\mathrm{data}}, \tilde{p}_{\mathrm{data}}) + \bar{\alpha}(p_\theta, \overrightarrow{h}, \overleftarrow{h}, \Pi, \Omega)\Big(\sqrt{\mathrm{KL}(\overrightarrow{p}_{\mathrm{data}} \| \mathcal{N}_{D''})}e^{-T} + (L\sqrt{D''} + L\mathsf{m} + \varepsilon_{\mathrm{score}})\sqrt{T}\Big), \qquad (3)$$

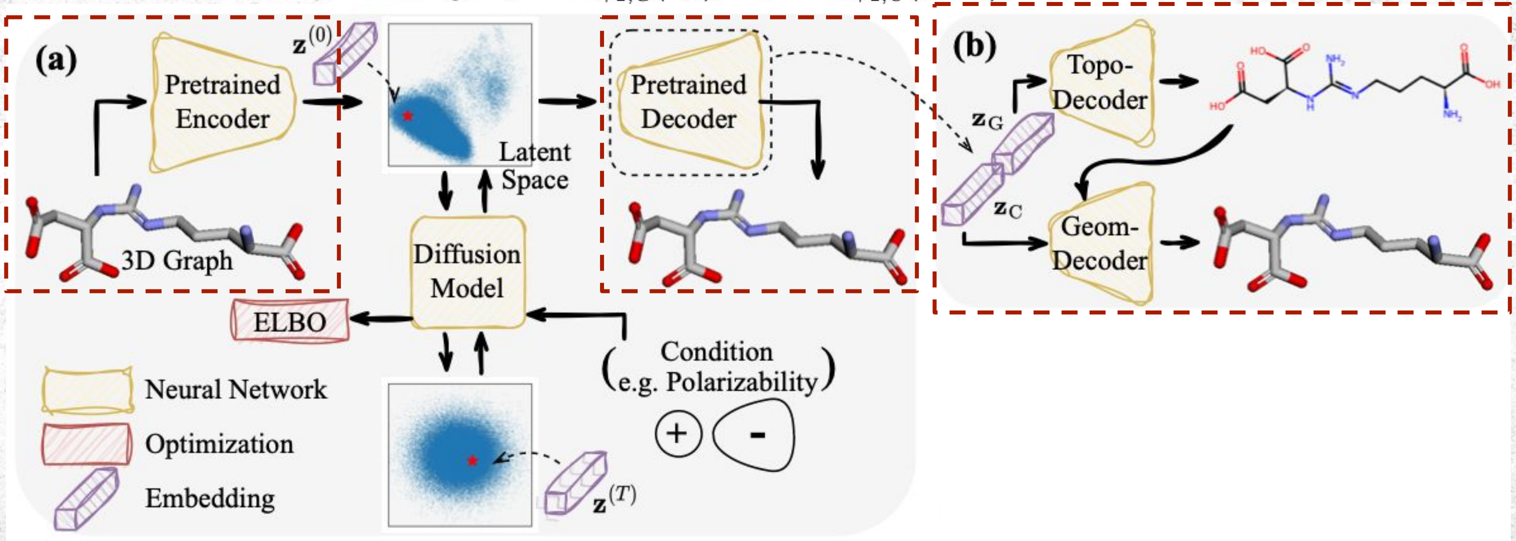
where $\overleftrightarrow{p}_{\mathrm{data}}([\mathbf{x}_{\mathrm{M}}]_{\Pi,\Omega}) = \Pr\{\mathbf{x}'_{\mathrm{M}} : \overleftarrow{h}(\overrightarrow{h}(\mathbf{x}'_{\mathrm{M}})) \in [\mathbf{x}_{\mathrm{M}}]_{\Pi,\Omega}, \mathbf{x}'_{\mathrm{M}} \sim p_{\mathrm{data}}\}$, and $\bar{\alpha}(\cdot)$ depends on both the latent diffusion architecture that $\bar{\alpha}(p_\theta, \overrightarrow{h}, \overleftarrow{h}, \Pi, \Omega) = \alpha(\overleftarrow{p}_\theta, \Pi, \Omega)$ if $\overleftrightarrow{p}_{\mathrm{data}} = p_{\mathrm{data}}$. □

Answer: How to construct a qualified latent space for graph diffusion?

- We develop the framework termed latent 3D graph diffusion.
- Cascaded auto-encoder for 3D graphs:

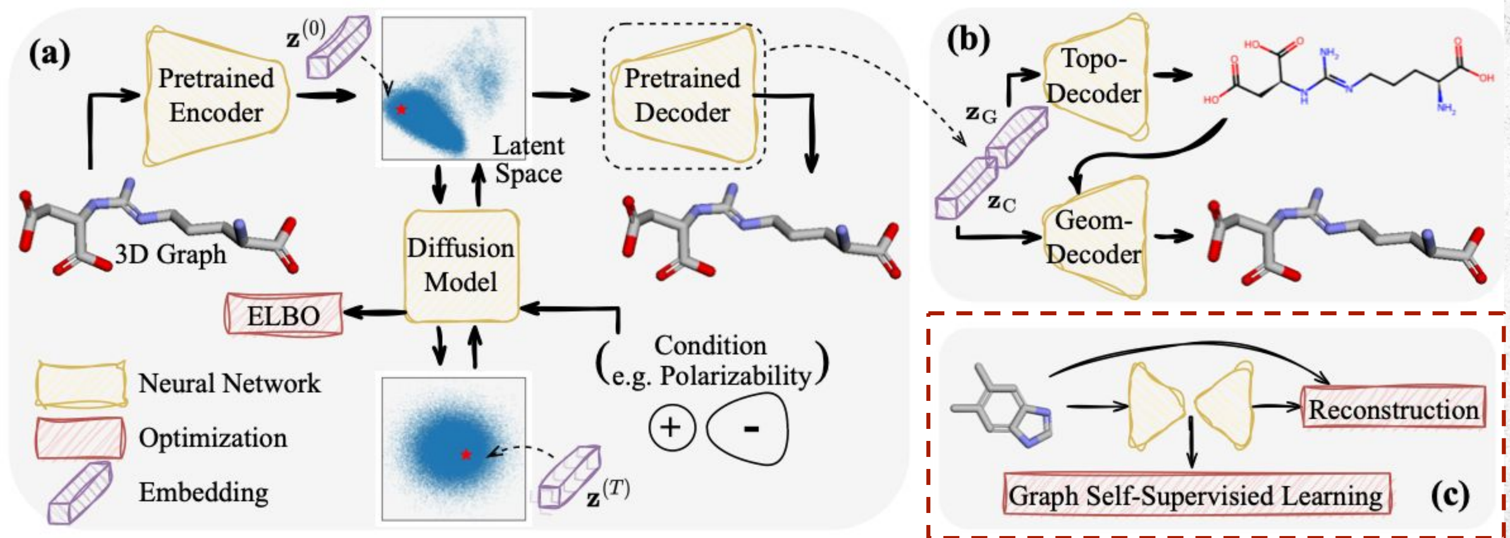
$$\text{Encoding:} \quad \mathbf{z}_{\mathrm{G}} = \overrightarrow{h}_{\phi_{1,\mathrm{G}}}(\mathcal{G}), \quad \mathbf{z}_{\mathrm{C}} = \overrightarrow{h}_{\phi_{1,\mathrm{C}}}(\mathcal{C}), \quad \mathbf{z} = [\mathbf{z}_{\mathrm{G}}; \mathbf{z}_{\mathrm{C}}],$$

$$\text{Decoding:} \quad \bar{\mathcal{G}} = \overleftarrow{h}_{\phi_{2,\mathrm{G}}}(\mathbf{z}_{\mathrm{G}}), \quad \bar{\mathcal{C}} = \overleftarrow{h}_{\phi_{2,\mathrm{C}}}(\bar{\mathcal{G}}, \mathbf{z}_{\mathrm{C}}),$$



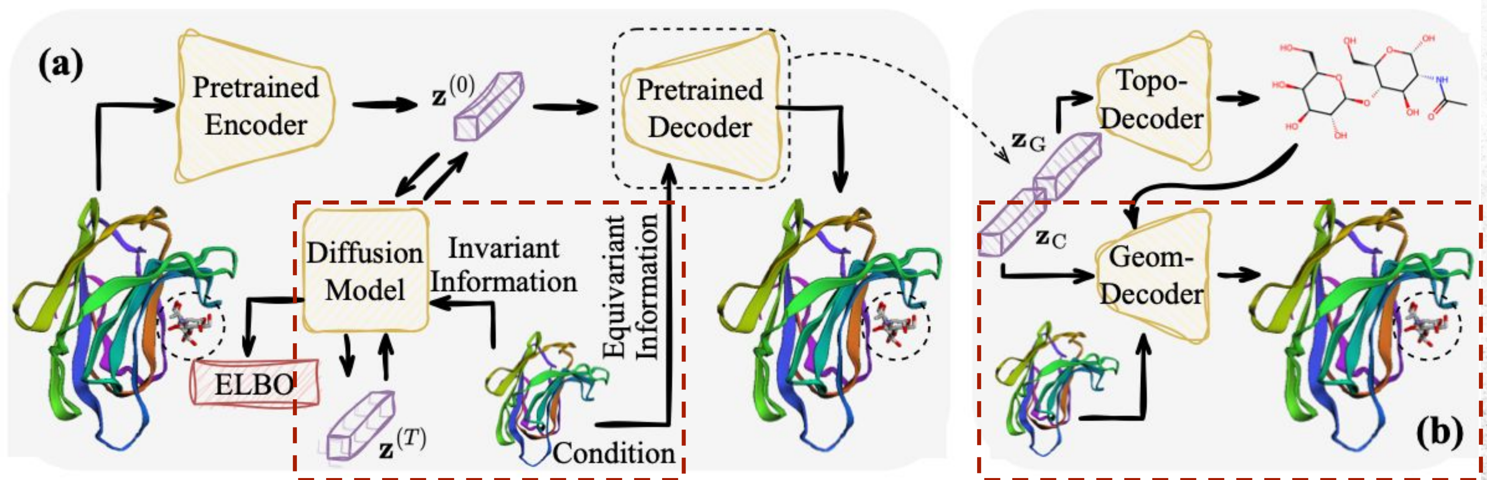
Answer: How to regularize the latent space to introduce domain prior?

- We develop the framework termed latent 3D graph diffusion.
- Graph self-supervised learning regularized auto-encoding:
  - Graph contrastive learning (GraphCL, NeurIPS'20).



Answer: How to extend the framework to conditional generation?

- Equivariance constraint: When condition is a geometric object.
- 1. In/Equi-variant representations for condition inputs;
- 2. Invariant distribution modeling of latent embeddings;
- 3. Equivariant decoding to reconstruct 3D graphs.



Experiments on Unconditional Generation

Table 2: Unconditional generation evaluation on validness of 3D molecules. Valid: proportion of (POF) chemically valid molecules; Valid&Uni: POF chemically valid and unique molecules; AtomSta: POF atoms with correct valency; MolSta: POF molecules without unstable atoms. Numbers(std) in **red** are the best results.

Methods	QM9				Drugs		**Mean**
	Valid	Valid&Uni	AtomSta	MolSta	Valid	AtomSta	
ENF	40.2	39.4	85.0	4.9	–	–	42.37
G-Schnet	85.5	80.3	95.7	68.1	–	–	82.40
GDM	–	–	97.0	63.2	90.8	75.0	81.50
GDM-Aug	90.4	89.5	97.6	71.6	91.8	77.7	86.43
EDM	91.9(0.5)	90.7(0.6)	98.7(0.1)	82.0(0.4)	92.6	81.3	89.53
EDM-Bridge	92.0	90.7	98.8(0.1)	84.6(0.3)	92.8	82.4(0.8)	90.21
GCDM	94.8(0.2)	93.3(0.0)	98.7(0.0)	85.7(0.4)	–	**89.0**(0.8)	92.30
MiDi	97.9	97.0	97.9	84.0	78.0	82.2	89.50
GraphLDM	83.6	82.7	**97.2**	70.5	97.2	76.2	84.56
GraphLDM-Aug	90.5	89.5	97.9	78.7	98.0	79.6	89.03
GeoLDM	93.8(0.4)	92.7(0.5)	**98.9**(0.1)	**89.4**(0.5)	99.3	84.4	93.08
Ours	**100.00**(0.00)	95.27(0.25)	97.57(0.02)	86.87(0.23)	**100.00**(0.00)	80.51(0.08)	**93.37**

Experiments on Conditional Generation

Table 6: Conditional generation on protein binding targets evaluation. Assessment metrics QED/SA & Vina scores are calculated with RDKit (Landrum, 2013) & AutoDock (Huey et al., 2012), respectively.

Methods	QED↑	SA↑	HiAff↑	Vina↓	VDock↓	Vina (Top-10%)↓	Diversity↑
LiGAN	0.39	0.59	21.1%	–	-6.33	–	0.66
GraphBP	0.43	0.49	14.2%	–	-4.80	-7.16	0.79
AR	0.51	0.63	37.9%	-5.75	-6.75	–	0.70
Pocket2Mol	0.56	**0.74**	48.4%	-5.14	-7.15	-8.71	0.69
TargetDiff	0.48	0.58	**58.1**%	-5.47	-7.80	-9.66	0.72
DiffSBDD	0.46	0.55	–	**-7.33**	–	-9.92	0.75
DecompDiff	0.45	0.61	64.4%	-5.67	**-8.39**	–	0.68
Ours	**0.60**	0.71	48.08%	-5.23	-6.85	**-12.34**	**0.80**

Thank You!

https://yyou1996.github.io/