Table 3: Statistics of the datasets used in graph feature imputation

Dataset	Nodes	Train/Test Features	Feature Density	Edges	Edge Density
Douban	3,000	123,202/13,689	1.52%	2,690	0.03%
Ciao	7,317	39,279/16,892	0.77%	111,781	0.21%
Cora	2,708	88,590/9,842	2.54%	5,429	0.15%
Citeseer	3,327	189,298/21,032	1.70%	4,732	0.08%
Amaphoto	7,650	35,640/3,958	0.69%	143,663	0.49%
Amacomp	13,752	66,150/7,350	0.69%	287,209	0.30%

Table 4: Statistics of the datasets used in graph structure generation

Datasets	MUTAG	PTC-MR	ZINC
No.Graphs	188	344	250k
Avg.Nodes	17.9	14.3	23.1
Avg.Edges	19.8	14.7	23.9

A Proof of Proposition 1

Proof. Based on the independent white noise model (7) and (8), we have

$$\begin{aligned} x' &= U\text{diag}(g_c^{-1}(\lambda))U^\top \sigma^{-1}(\hat{h}) \\ &= U\text{diag}(g_c^{-1}(\lambda))U^\top \sigma^{-1}\left(\sigma(U\text{diag}(g_c(\lambda))U^\top x) + \epsilon\right) \\ &= x + U\text{diag}(g_c^{-1}(\lambda))U^\top \sigma^{-1}(\epsilon). \end{aligned}$$

Hence,

$$(x' - x) = U\text{diag}(g_c^{-1}(\lambda))U^\top \sigma^{-1}(\epsilon). \tag{14}$$

In general, we cannot find an exact formula for the variance of a nonlinear function. Here, we consider a more relevant case, that is, a random variable ϵ is fairly closed to the mean (i.e., the standard error is sufficiently small), then we can approximate the nonlinear transformation $\sigma^{-1}(\epsilon)$ with a first-order Taylor expansion. Under the condition $\sigma^{-1}(\epsilon) \approx \sigma^{-1}(\mathbf{0}) + \nabla\sigma^{-1}(\mathbf{0})^T\epsilon$ where $\sigma^{-1}(\cdot)$ is a element-wise function. Plugging into (14), we have

$$\begin{aligned} (x' - x) &\approx U\text{diag}(g_c^{-1}(\lambda))U^\top(\sigma^{-1}(\mathbf{0}) + \nabla\sigma^{-1}(\mathbf{0})^T\epsilon) \\ x' &\approx U\text{diag}(g_c^{-1}(\lambda))U^\top\nabla\sigma^{-1}(\mathbf{0})^T\epsilon + U\text{diag}(g_c^{-1}(\lambda))U^\top\sigma^{-1}(\mathbf{0}) + x \end{aligned} \tag{15}$$

Hence, we can observe that the variance of x' is almost identical to the variance of $U\text{diag}(g_c^{-1}(\lambda))U^\top\nabla\sigma^{-1}(\mathbf{0})^T\epsilon$ in an asymptotic sense.

$$\begin{aligned} \text{Var}(x') &\approx (\partial\sigma^{-1})^2(0)\text{Var}(U\text{diag}(g_c^{-1}(\lambda))U^\top\epsilon) \\ &\geq \text{Var}(U\text{diag}(g_c^{-1}(\lambda))U^\top\epsilon). \end{aligned} \tag{16}$$

The last inequality holds as our assumption on the inverse activation function. We will elaborate more details and explain why it is a reasonable condition.

Fact 2. *If z is a Gaussian random variable, i.e., $z \sim \mathcal{N}(\mu, \Sigma)$, then $Az \sim \mathcal{N}(A\mu, A\Sigma A^T)$.*

$$\begin{aligned} \Delta &= U\text{diag}(g_c^{-1}(\lambda))U^\top\epsilon \\ &= U\text{diag}(g_c^{-1}(\lambda))\xi \text{ (denote } \xi = U^\top\epsilon). \end{aligned}$$

Here, the distribution of ξ is same as ϵ's due to the orthogonal matrix U and the rotation invariance of the Gaussian distribution. We refer the reader to Fact 2 for details. Hence, note that $y = \text{diag}(g_c^{-1}(\lambda)\xi$, we have

$$y \sim \mathcal{N}(0, \sigma^2\text{diag}\left([(1-\lambda_1)^{-2}, \cdots, (1-\lambda_N)^{-2}]\right)).$$

$$\text{Var}(x') = \text{Var}(\Delta) = \sigma^2 \sum_{i=1}^{N} \frac{1}{(1-\lambda_i)^2}.$$

Since the eigenvalue of the normalized Laplacian matrix is bounded, that is, $\lambda_i \in [0, 2], \forall i \in [N]$, we have $1 - \lambda_i \in [-1, 1]$ and $\frac{1}{(1-\lambda_i)^2} \in [1, +\infty)$. It was worthwhile mentioning that the equality holds only when $\lambda_i = 0$ or $2, \forall i \in [N]$. Therefore,

$$\text{Var}(x') = \sigma^2 \sum_{i=1}^{N} \frac{1}{(1-\lambda_i)^2} \geq N\sigma^2 = \text{Var}(\epsilon).$$

Regarding the condition $\partial\sigma^{-1}(0) \geq 1$, we can consider three representative examples to corroborate our assumption, i.e.,

- Sigmoid function; $\sigma(x) = \frac{1}{1+e^{-x}}$ and its inverse operator $\sigma^{-1}(x) = -\ln\left(\frac{1}{x} - 1\right)$. Thus, we have
$$\partial\sigma^{-1}(x) = \frac{1}{x - x^2},$$
and $(\partial\sigma^{-1}(x))^2 \to +\infty$ when x goes to zero.

- Hyperbolic tangent (Tanh) activation function; $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The inverse hyperbolic tangent is given as
$$\sigma^{-1}(x) = \frac{1}{2}[\ln(1+x) - \ln(1-x)].$$
Computing its gradient, we have
$$\partial\sigma^{-1}(x) = \frac{1}{1-x^2} \geq 1.$$

- Leaky ReLu;
$$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ \frac{x}{\alpha} & \text{if } x \leq 0, \end{cases} \Rightarrow \sigma^{-1}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0, \end{cases}$$
where α is a constant belonging to $(1, +\infty)$. Therefore, the subgradient of σ^{-1} is larger or equal to 1.

□

Remark 3. *The last equality is extremely hard to hold in practice and indeed to corroborate our empirical finding that the inverse operator will amplify the noise.*

This concludes the proof of proposition 1. Next, we show proposition 1 still holds if GCN model is parameterized.

Parameterized Model We consider

$$\hat{h} = \sigma(WU\text{diag}(g_c(\lambda))U^\top x) + \epsilon, \tag{17}$$

where ϵ is the white Gaussian noise (i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbb{I}_N)$). Under the condition that W is orthogonal, the inverse operator admits

$$x' = U\text{diag}(g_c^{-1}(\lambda))U^\top W^T \sigma^{-1}(\hat{h}). \tag{18}$$

Therefore,

$$\begin{aligned} x' &= U\text{diag}(g_c^{-1}(\lambda))U^\top W^T \sigma^{-1}(\hat{h}) \\ &= U\text{diag}(g_c^{-1}(\lambda))U^\top W^T \sigma^{-1}\left(\sigma(WU\text{diag}(g_c(\lambda))U^\top x) + \epsilon\right) \\ &= x + U\text{diag}(g_c^{-1}(\lambda))U^\top W^T \sigma^{-1}(\epsilon). \end{aligned}$$

It was worth noting that the product of orthogonal matrices is orthogonal. By the similar argument in Proposition 1, we can also conclude that the proposed inverse operation may introduce undesirable noise into the output graph.

B Comparisons with Additional Baselines

We additionally provide the comparison results of our method with three discrete matrix completion models: sRGCNN [29], GC-MC [4], and GraphRec [12]. We adapt the same setting as in Section 4.1 and the results are shown in Table 5. Ours yields the smallest imputation error on all datasets compared with the three matrix completion models. Notice that the three matrix completion models are specially designed for discrete matrix completion, where ours is applicable to both continuous and discrete feature values.

C Graph Autoencoder Framework

C.1 Encoder

Our encoder consists of two layers of Graph Convolutional Networks (GCN) [21], to produce smoothed representations of the input graphs.

Convolution

$$H = \text{GCN}(A, X), \tag{19}$$

where $H \in \mathbb{R}^{N \times v}$ denotes smoothed node representations.

C.2 Decoder

Our decoder consists of one layer of GDN, to produce fine-grained graphs.

Deconvolution

$$X' = \text{GDN}(A, H), \tag{20}$$

where $X' \in \mathbb{R}^{N \times d}$ denotes the recovered graph features.

C.3 The loss function

The reconstruction loss is a feature reconstruction loss.

$$\mathcal{L} = f(X, X'), \tag{21}$$

where $f(\cdot, \cdot)$ denotes a differential distance metric, e.g., $f(\cdot, \cdot) = \text{MSE}(\cdot, \cdot)$, and $\text{MSE}(\cdot, \cdot)$ represents Mean Squared Error.

D Detailed Model Configuration

All the experiments are done on a single machine with 32 cores and 300G memories. We don't use GPU as it involves a lot of sparse matrix multiplication.

For graph feature imputation, We use the graph autoencoder framework specified in Appendix C. The recovered graph features X' is used to infer the potential missing features. We train our model using Adam optimizer with a learning rate of $\{0.005, 0.002\}$. The output dimension of the first layer is $\{256, 512\}$. The output dimension of the second layer is 128. The number of epochs is chosen from $\{100, 200\}$. We use full-batch size. We stack the output of the first GCN layer and the second GCN layer in our encoders, and use left normalization to preprocess adjacency matrix. We don't use feature Dropout as it does not improve the performance. We use DropEdge [33]. We use grid search to choose the hyper-parameters. The selected parameters are reported in Table 6.

For graph generation, We strictly follow the architectures of VGAE [20] and Graphite [15] and add one feature reconstruction loss. The overall reconstruction loss is a sum of structure reconstruction term and feature reconstruction term,

$$\log p(G|Z) = \log p(A|Z) + \log p(X|Z), \tag{22}$$

where $p(A|Z)$ is the default structure reconstruction term such as a simple inner product of latent variables in VGAE , i.e., $\prod_i \prod_j p(A_{ij}|z_i, z_i)$. $\log p(X|Z)$ is computed with GDN and defined as

Table 5: RMSE test comparison with matrix completion methods on graph feature imputation

Datasets	**Ciao**	**Douban**	**Cora**	**Citeseer**	**Amaphoto**	**Amacomp**
sRGCNN [29]	1.183	0.801	0.550	0.551	0.519	0.509
GC-MC [4]	1.061	**0.734**	0.434	0.420	0.402	0.405
GraphRec [12]	1.062	0.754	0.437	0.425	0.417	0.417
OURS	**1.011**	**0.734**	**0.415**	**0.399**	**0.391**	**0.393**

Table 6: Summary of chosen hyper-parameters on graph feature imputation

Datasets	**Ciao**	**Douban**	**Cora**	**Citeseer**	**Amaphoto**	**Amacomp**
Dim of the first layer	256	256	512	256	256	256
Dim of the second layer	128	128	64	128	128	128
Learning rate	0.005	0.005	0.002	0.005	0.005	0.005
Epochs	200	200	200	200	100	100
DropEdge rate	1	0.5	0.5	0.5	1	1

in Eq. 21. As both VGAE and Graphite can only deal with homogeneous edge attribute, we don't differentiate bond types in ZINC dataset, i.e., we only evaluate if there is a bond between two given atoms. The number of epochs, initial learning rate and hyperparameters are all set to be the same with VGAE [20] or Graphite [15], for a fair comparison.The effect of orders of Maclaurin Series in GDN are reported in Table 7.

E Derivative of Inverse Operator

$$g_c^{-1}(\lambda_i) = \frac{1}{1-\lambda_i} = \sum_{n=0}^{\infty} \frac{\left(\frac{1}{1-\lambda_i}\right)^{(n)}_{\lambda_i=0}}{n!}\lambda_i^n = \sum_{n=0}^{\infty} \frac{(-1)^n n!(-1)^n}{n!}\lambda_i^n = \sum_{n=0}^{\infty} \lambda_i^n$$

where (n) denotes the n-th order derivative. Thus, Eq. 9 can be obtained as:

$$U\text{diag}(\sum_{n=0}^{\infty}\lambda_1^n, \ldots, \sum_{n=0}^{\infty}\lambda_N^n)U^\top = U\sum_{n=0}^{\infty}\Lambda^n U^\top \tag{23}$$

Table 7: Summary of the effect of orders of Maclaurin Series in GDN on the performance of graph structure generation

	Datasets	**MUTAG**			**PTC-MR**			**ZINC**		
	-	$\log p(A\|Z)$	AUC	AP	$\log p(A\|Z)$	AUC	AP	$\log p(A\|Z)$	AUC	AP
VGAE + GDN	First-order	**-1.002**	0.738	0.464	**-1.351**	0.493	0.386	**-0.999**	0.474	0.239
	Second-order	-1.008	0.766	0.502	-1.356	0.559	0.443	-1.001	0.611	0.345
	Third-order	-1.026	**0.823**	**0.611**	-1.351	**0.760**	**0.602**	-1.006	**0.858**	**0.611**
Graphite + GDN	First-order	**-1.003**	0.741	0.470	-1.355	0.529	0.398	-1.001	0.478	0.241
	Second-order	-1.014	0.774	0.501	-1.358	0.574	0.450	-1.003	0.632	0.379
	Third-order	-1.024	**0.818**	**0.608**	**-1.347**	**0.773**	**0.613**	**-0.998**	**0.838**	**0.567**