GRAPH CONVOLUTIONAL ATTENTION: A SPECTRAL PERSPECTIVE ON GRAPH DENOISING AND DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Denoising graphs is a fundamental problem in graph learning and the core operation of graph diffusion models. Attention-based architectures such as graph transformers have shown promise for this task, yet the mechanisms through which attention denoises graphs remain poorly understood. We show that, under a graph denoising objective, standard attention scores learn an average spectral denoising filter across the training distribution and are suboptimal when the graph spectra vary. To address this limitation, we introduce spectral attention scores, which are attention scores that depend on the input graph spectrum. We characterize the improvement over standard attention scores in terms of the spectral diversity of the graph distribution. We then propose graph convolutional attention (GCA), a practical realization of spectral attention that uses graph-filtered queries and keys. For asymptotically large stochastic block models, we show that GCA matches the optimal denoising loss of spectral attention under suitable conditions. We also show that softmax can further reduce denoising error by approximately projecting noisy eigenvectors onto the clean eigenspace. Across synthetic and real datasets, replacing standard query–key projections with GCA improves graph denoising and diffusion, with denoising gains correlated with spectral diversity. Furthermore, replacing the DiGress architecture with a GCA-based architecture within the DiGress generative framework yields comparable generation quality with significantly faster inference.

1 INTRODUCTION

Denoising is a fundamental problem in machine learning (Milanfar & Delbracio, 2025), both in its own right and as the central learning primitive behind denoising diffusion models (Song et al., 2020a;b). Graph denoising introduces a unique challenge: the object to be denoised is not merely a signal on a known domain, but the domain itself. In image and audio denoising, the underlying grid or temporal axis is fixed, so models can exploit geometry through convolutions (Milanfar, 2012). In graphs, by contrast, the geometry and spectral content are corrupted and must themselves be recovered. Classical approaches address graph denoising through regularized optimization (Wu et al., 2017; Segarra et al., 2017), but the parametric perspective has received comparatively little attention. Interest has grown recently, driven by graph diffusion models (Jo et al., 2022; Vignac et al., 2022), where the dominant approach relies on graph transformers that take as input the eigenvectors of the graph, and optionally, additional structural features. Yet, a theoretical understanding of how such parametric models learn to denoise graphs remains limited. In this work, we aim to provide insight into the mechanisms underlying parametric graph denoising through theoretical analysis.

We proceed in four steps. First, we (i) analyze the attention-score matrix, the outer product of linear query-key projections, which underlies standard attention-based graph denoisers. We show that it learns an average of the spectral properties across the graph distribution and is therefore suboptimal when the spectra vary. To address this limitation, we (ii) introduce spectral attention, a general class whose score matrices depend arbitrarily on the input eigenvalues. We show that its minimum denoising loss is lower than that achieved using standard attention scores by an amount determined by the spectral diversity of the graph distribution, i.e., how diverse the spectra are across the distribution. Spectral attention, however, is an abstract class. To turn the insight into something practical, we (iii) introduce graph convolutional attention (GCA), in which the attention-score matrix becomes a pointwise function of the eigenvalues. We show that GCA scores are approximated arbitrarily well by products of graph filters, and that, for asymptotically large stochastic block models (SBMs),

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

GCA matches the optimal loss of spectral attention under suitable conditions. Finally, we (iv) study the softmax applied to the attention-score matrix. Assuming that the clean eigenvalues have been recovered, we show that softmax can further denoise the graph by approximately projecting the noisy eigenvectors onto the clean eigenspace.

Experimentally, across real and synthetic datasets, a graph transformer using GCA in place of the standard query-key projections outperforms the standard graph transformer at denoising, and the per-dataset improvement correlates strongly with a surrogate for the spectral diversity metric, validating the theory. For diffusion models, GCA reduces training loss and, on several datasets, sample quality both qualitatively and under standard metrics. Finally, replacing the DiGress architecture with a simpler GCA-based architecture within the same generative framework achieves comparable graph generation quality while reducing inference cost and enabling faster graph generation.

We summarize our contributions as follows:

- **C1 Theoretical analysis of attention-based denoising.** We show that standard attention scores learn an average of the spectral properties across the graph distribution and are suboptimal when those properties vary. We introduce spectral attention and characterize its improvement in minimum denoising loss through the spectral diversity of the distribution.
- **C2 Graph convolutional attention (GCA).** We propose GCA, an attention mechanism whose score matrix is approximated arbitrarily well by graph convolutional filters. For asymptotically large stochastic block models (SBMs), we show that, under suitable conditions, GCA matches the optimal loss of spectral attention.
- **C3 Role of softmax.** In an idealized setting where the clean eigenvalues are known, we show that softmax further improves denoising by approximately projecting the noisy graph onto the clean eigenspace.
- **C4 Empirical validation.** Across real and synthetic datasets, replacing standard query–key projections with GCA improves graph denoising. The gains across datasets correlate with a surrogate for spectral diversity and extend to graph diffusion, where GCA reduces training loss and improves sample quality on several datasets.
- **C5 Implications for modern diffusion models.** Replacing the DiGress architecture with a simpler GCA-based architecture within the same generative framework yields comparable graph generation quality across the evaluated datasets, while reducing inference time.

1.1 Literature review

Graph denoising. Existing graph denoising methods can be broadly divided into optimization-based and learned approaches. The former recover a clean graph by solving an inverse problem regularized with structural priors such as edge-error models (Wu et al., 2017; Josephs et al., 2021), spectral consistency (Segarra et al., 2017; Rey et al., 2023; Tenorio et al., 2024), or low-rank, sparse, and smooth graph structure (Zhou et al., 2024; Wang et al., 2024; Lyu et al., 2024). The latter replace instance-wise optimization with a parametric denoiser, either as a preprocessing tool or as a jointly trained module for downstream GNNs (Fatemi et al., 2021; Dai et al., 2022; Yang et al., 2024), or as the core component of diffusion and flow-based generative models that iteratively remove graph corruption (Li et al., 2021; Jo et al., 2022; Vignac et al., 2022). Overall, prior work spans from hand-crafted regularization to fully learned graph restoration, but most methods are not designed to explicitly exploit spectral structure within the denoising mechanism itself.

Spectral attention. A parallel line of work incorporates graph spectral information into Transformer architectures. Early methods use Laplacian eigenvectors or eigenvalues as spectral features or positional encodings for attention-based models (He et al., 2020; Kreuzer et al., 2021; Dwivedi & Bresson, 2021; Pengmei & Li, 2024), while later approaches address eigenvector ambiguities through invariant spectral encoders such as SignNet and BasisNet (Lim et al., 2022). More recent models integrate spectral structure directly into the attention mechanism or the learned operator. GIST reweights attention using inner products of random filtered spectral features (Rigotti et al., 2026), whereas Specformer, PolyFormer, and Eigenformer use Transformer blocks to parametrize spectral filters using attention (Bo et al., 2023; Ma et al., 2024; Garg, 2024). These works show that spectral information can be embedded into attention in several ways, but they are mainly aimed at representation learning and prediction rather than graph denoising.

2 DENOISING GRAPHS WITH A LINEAR MODEL

Problem Setup. We begin by introducing the graph denoising problem as it is fundamental to graph diffusion. In particular, Sections 2 and 3 analyze how attention-based neural networks perform denoising from a spectral perspective. We show in this section that the attention scores learn to denoise the eigenvalues of a noisy graph, while, in Section 3, we show that the softmax nonlinearity helps remove noise from the eigenvectors. Regarding the setup, we consider unweighted, undirected graphs $\mathcal{G}$ sampled from a distribution over graphs which we denote by $\mathcal{D}_{\mathcal{G}}$. For a given graph $\mathcal{G}$, we denote by $A \in \mathbb{R}^{n\times n}$ the symmetric adjacency matrix. We denote the noisy adjacency by $\widetilde{A} = A + \mathcal{E}$, where $\mathcal{E} \in \mathbb{R}^{n\times n}$ is a symmetric noise matrix sampled from a noise distribution. We note that depending on the noising method this distribution could depend on the input graph (flipping edge values) or it could be independent (Gaussian noise). We write the eigendecomposition of the adjacency as $A = U\Lambda U^\top$, and that of the noisy adjacency as $\widetilde{A} = \widetilde{U}\widetilde{\Lambda}\widetilde{U}^\top$.

2.1 EIGENVALUE INFORMATION IMPROVES DENOISING.

Consider an arbitrary denoising function $g(\cdot) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ that takes the noisy adjacency as input and returns a prediction of the denoised graph. We define the following loss functional characterizing the denoising capability of g over different graphs and noise samples from the distribution:

$$\ell(g) = \mathbb{E}_{A,\,\mathcal{E}} \left\| g(\widetilde{A}) - A \right\|_F^2 . \tag{1}$$

A common candidate for implementing the denoising map is attention (Vignac et al., 2022; Qin et al., 2024). For a sequence of d-dimensional input tokens $X \in \mathbb{R}^{n\times d}$, standard attention computes

$$a(X) = \sigma(XW_Q W_K^\top X^\top)XW_V, \tag{2}$$

where σ is the row-wise softmax activation function, and $W_Q \in \mathbb{R}^{d\times d_k}$, $W_K \in \mathbb{R}^{d\times d_k}$, and $W_V \in \mathbb{R}^{d\times d_v}$ are learnable query, key, and value matrices, respectively. In graph denoising, attention-based estimators of the adjacency can be built as an outer product of representations (Vignac et al., 2022)

$$g(\widetilde{A}) = a(\widetilde{U})a(\widetilde{U})^\top = \sigma(\underbrace{\widetilde{U}W_Q W_K^\top \widetilde{U}^\top}_{f(\widetilde{A};W_Q,W_K)}) \cdot \widetilde{U}W_V W_V^\top \widetilde{U}^\top \cdot \sigma(\widetilde{U}W_Q W_K^\top \widetilde{U}^\top)^\top. \tag{3}$$

In this view, attention acts on $\widetilde{U}$ to recover a spectral representation of the clean graph A. We analyze this mechanism in two stages. First, we study the attention-score matrix $f(\widetilde{A}; W_Q, W_K) = \widetilde{U}W_Q W_K^\top \widetilde{U}^\top$. This isolates the effect of the scores from the softmax and the value projections: we ask how well they can approximate A by learning its representation in the noisy eigenbasis $\widetilde{U}$. Second, we study in the next section how the softmax and the outer product of the resulting representations can further reduce the error associated with the noisy eigenvectors. Together, these stages distinguish the spectral information learned by the attention scores from the additional denoising provided by the rest of the attention-based estimator. To analyze the first stage, we consider three classes of simplified denoising maps associated with the attention scores:

1. **Standard Attention Scores:** We consider f from the standard function class
$$\mathcal{F}_{\text{A}} = \left\{ f(\widetilde{A}; W_Q, W_K) = \widetilde{U}W_Q W_K^\top \widetilde{U}^\top \mid W_Q \in \mathbb{R}^{n\times d_k}, W_K \in \mathbb{R}^{n\times d_k} \right\}. \tag{4}$$
2. **Spectral Attention Scores:** The broader class of functions $f \in \mathcal{F}_{\text{SA}}$ where attention depends on the eigenvalues of the noisy input graph via any arbitrary function $h(\cdot)$
$$\mathcal{F}_{\text{SA}} = \left\{ f(\widetilde{A}; h(\cdot)) = \widetilde{U}h(\widetilde{\Lambda})\widetilde{U}^\top \mid h(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n\times n} \right\}. \tag{5}$$
3. **Graph Convolutional Attention Scores:** We consider the class of score functions $f \in \mathcal{F}_{\text{GCA}}$ where attention is replaced by a function $\eta(\cdot)$ acting point-wise on the eigenvalues
$$\mathcal{F}_{\text{GCA}} = \left\{ f(\widetilde{A}; \eta(\cdot)) = \widetilde{U}\eta(\widetilde{\Lambda})\widetilde{U}^\top \mid \eta(\cdot) : \mathbb{R} \to \mathbb{R} \right\}, \tag{6}$$
where, with a slight abuse of notation, $\eta(\widetilde{\Lambda})$ denotes the pointwise application of η to the entries of the diagonal matrix $\widetilde{\Lambda}$.

We now evaluate the denoising maps f in the classes defined above under the loss in (1), replacing the full attention-based estimator g with f. Propositions 1 and 2 characterize the minimum achievable loss for standard and spectral attention scores, respectively. Their proofs are given in Appendices C.1 and C.2.

Proposition 1. *For some constant $c \geq 0$, the minimum loss achievable with standard attention scores, $\ell_A^\star := \min_{f \in \mathcal{F}_A} \ell(f)$, is given by*

$$\ell_A^\star = \operatorname{tr}\left(\operatorname{Cov}(\operatorname{vec}(B))\right) + c, \tag{7}$$

where $B := \widetilde{U}^\top A \widetilde{U}$. Furthermore, the minimizer is given by $f(\cdot\,; W^\star)$ where

$$W^\star = \mathbb{E}_{A,\,\mathcal{E}}[B]. \tag{8}$$

Proposition 2. *For some constant $c \geq 0$, the minimum loss achievable with spectral attention scores, $\ell_{SA}^\star := \min_{f \in \mathcal{F}_{SA}} \ell(f)$, is given by*

$$\ell_{SA}^\star = \operatorname{tr}\left(\operatorname{Cov}(\operatorname{vec}(B) \mid \widetilde{\Lambda})\right) + c, \tag{9}$$

where $B := \widetilde{U}^\top A \widetilde{U}$. Furthermore, the minimizer is given by $f(\cdot\,; h^\star)$ where

$$h^\star(\widetilde{\Lambda}) = \mathbb{E}_{A,\,\mathcal{E}}\left[B \;\middle|\; \widetilde{\Lambda}\right]. \tag{10}$$

By the law of total variance, conditioning can only reduce variance, so $\ell_{\mathrm{SA}}^\star$ is smaller or equal to $\ell_{\mathrm{A}}^\star$. We formalize this in Theorem 1, proven in Appendix C.3.

Theorem 1. *The improvement in the minimum achievable loss by the spectral attention scores compared to the standard attention scores is given by*

$$\ell_A^\star - \ell_{SA}^\star = \mathrm{SD}(\mathcal{D}_\mathcal{G}) := \mathbb{E}\left[\left\|\mathbb{E}[B \mid \widetilde{\Lambda}] - \mathbb{E}[B]\right\|_F^2\right] = \operatorname{tr}(\operatorname{Cov}(\mathbb{E}[\operatorname{vec}(B) \mid \widetilde{\Lambda}])), \tag{11}$$

where we define $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ as the "spectral diversity" which depends on the graph data distribution and on the noise distribution.

Remark 1. *The spectral diversity $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ essentially quantifies how much information we gain about the clean graph by observing the noisy spectrum. For more intuition on this, if we assume the noise is sufficiently small so that the noisy eigenvectors are close to the clean ones, then from (11) we can write $\operatorname{tr}(\operatorname{Cov}(\mathbb{E}[\operatorname{vec}(B) \mid \widetilde{\Lambda}])) \approx \operatorname{tr}(\operatorname{Cov}(\mathbb{E}[\Lambda \mid \widetilde{\Lambda}]))$ which is akin to the variance of the expected value of Λ having observed $\widetilde{\Lambda}$. At one extreme, if the graph distribution is degenerate, i.e., its support is only a single graph, then $\mathbb{E}[\Lambda \mid \widetilde{\Lambda}] = \mathbb{E}[\Lambda]$, and its variance and therefore the spectral diversity are 0. At the other extreme, the more Λ varies across graphs, the more $\mathbb{E}[\Lambda \mid \widetilde{\Lambda}]$ can vary, leading to larger variance and spectral diversity.*

2.2 From Spectral Attention to Graph Convolutional Attention

The spectral attention class is rather abstract as it includes any arbitrary function of the eigenvalues. GCA on the other hand, is practically realizable with graph convolutional filters. In this section we aim to bridge the gap between these two classes in certain settings. We begin by defining a graph convolutional filter bank acting on input graph features $X \in \mathbb{R}^{n \times d}$ as

$$h_{\text{filt}}(X) = \sum_{l=0}^{L-1} S^l X H^{(l)}, \tag{12}$$

where S is the graph-shift operator (e.g., adjacency or Laplacian of the graph) and $H^{(l)} \in \mathbb{R}^{d \times d}$ is a set of learnable weights. In our case the input features are the eigenvectors of the graph, i.e., $X = U$, thus when $S = A$, Eq. (12) simplifies to

$$h_{\text{filt}}(U) = \sum_{l=0}^{L-1} U \Lambda^l U^\top U H^{(l)} = U \left(\sum_{l=0}^{L-1} \Lambda^l H^{(l)}\right). \tag{13}$$

By replacing the query and key projections with graph convolutional filter banks as in (13) we get the following function of the adjacency

$$f(A;H) = U\left(\sum_{l=0}^{L-1}\Lambda^l H_Q^{(l)}\right)\left(\sum_{l=0}^{L-1}\Lambda^l H_K^{(l)}\right)^\top U^\top, \tag{14}$$

where $H_K^{(l)}$ and $H_Q^{(l)}$ are diagonal matrices. Eq. (14) reduces to a product of graph filters. The connection between graph convolutional filters and the GCA class defined in (6) can be made formal. We formalize this connection in Lemma 1 (proof in Appendix C.4).

Lemma 1. *A graph convolutional filter bank, i.e., $h_{filt}(\Lambda) = \left(\sum_{l=0}^{L-1}\Lambda^l H_Q^{(l)}\right)\left(\sum_{l=0}^{L-1}\Lambda^l H_K^{(l)}\right)^\top$, for sufficiently large L, can approximate any function acting pointwise on the eigenvalues $\eta(\lambda) : \mathbb{R}\to\mathbb{R}$, if η is continuous on any compact interval $[a,b]\subset\mathbb{R}$.*

With the preceding discussion and Lemma 1 we have established a practical way to realize the functions in $\mathcal{F}_{\text{GCA}}$. Still in general there could be a gap between the minimum achievable loss of the graph convolutional attention model and the spectral attention model because the minimizer of the loss in $\mathcal{F}_{\text{SA}}$ is not necessarily a pointwise function of the eigenvalues. In other words since $\mathcal{F}_{\text{GCA}}\subset\mathcal{F}_{\text{SA}}$, it follows that $\ell^\star_{\text{SA}}\le\ell^\star_{\text{GCA}}$. Thus, it is not obvious whether GCA also improves upon standard attention scores. In order to bridge this gap we show in this section that under certain conditions we have $\ell^\star_{\text{GCA}} = \ell^\star_{\text{SA}}$.

Stochastic Block Model (SBM). Consider the general SBM of size n with k communities. A given graph sampled from this model can be written as

$$A = \mathbb{E}[A] + \underbrace{(A - \mathbb{E}[A])}_{\Delta}, \tag{15}$$

where $\mathbb{E}[A]$ is a deterministic rank k matrix with block structure and the matrix $\Delta := A - \mathbb{E}[A]$ encodes the randomness. Now consider our noisy observation $\widetilde{A} = A + \mathcal{E}$ which we can write as

$$\widetilde{A} = \mathbb{E}[A] + \Delta + \mathcal{E}. \tag{16}$$

Assumption 1. *We assume the variance of Δ is small compared to the eigenvalues of $\mathbb{E}[A]$. In this setting we can ignore Δ and focus on recovering the low rank signal $A\approx\mathbb{E}[A]$ from the noisy observation $\mathbb{E}[A]+\mathcal{E}$. We further consider $\mathcal{E}$ to be a standard Wigner matrix scaled by v.*

A Wigner matrix is a symmetric random matrix with independent zero-mean entries on and above the diagonal. This assumption is not overly restrictive in our setting, since it captures independent entrywise perturbations of a graph. Gaussian noise with independent entries is a Wigner perturbation. Additionally, edge-flipping noise in the SBM setting, can be decomposed into a zero-mean Wigner-type component plus a deterministic bias term (See C.8 for proof). Hence the edge-flip setting also reduces to a low-rank deterministic matrix plus a Wigner-noise matrix. See Appendix C.5 for proof.

Theorem 2 (Outlier Shrinkage)**.** *Under the setting of Assumption 1, we have*

$$\lim_{n\to\infty}\mathbb{E}\left[\widetilde{U}^\top A\widetilde{U}\middle|\widetilde{\Lambda}\right] = \eta_{out}(\widetilde{\Lambda}), \tag{17}$$

where $\eta_{out}(\cdot):\mathbb{R}\to\mathbb{R}$ is a function acting pointwise on the noisy eigenvalues given by

$$\eta_{out}(\widetilde{\lambda}) = \begin{cases}\sqrt{\widetilde{\lambda}^2 - 4v^2} & \text{if } |\widetilde{\lambda}| > 2v\\ 0 & \text{if } |\widetilde{\lambda}|\le 2v.\end{cases} \tag{18}$$

As a result, $\eta_{out}(\cdot)\in\mathcal{F}_{GCA}$.

Remark 2. *It follows immediately from Theorem 2 and Lemma 1 that in the asymptotic regime considered above, $\ell^\star_{GCA} = \ell^\star_{SA}$. Consequently, in this regime GCA attains the optimal spectral-attention loss and therefore improves on standard attention scores by SD$(\mathcal{D}_\mathcal{G})$.*

3 How Softmax Can Help Denoise Further

In Section 2, we showed how a model linear in the input tokens $\widetilde{U}$ denoises the eigenvalues using graph-convolutional filters. We now turn from the attention scores f to the full denoising map g, and examine how the softmax operation and the outer product of the resulting representations can further improve denoising. We will see that, from a spectral perspective, these operations reduce the error associated with the noisy *eigenvectors*, complementing the eigenvalue denoising studied in Section 2.

Problem Setup. Let $A \in \mathbb{R}^{n\times n}$ be block diagonal with k blocks of sizes $m_1, \dots, m_k$ respectively. All entries within the blocks equal one and all others equal zero. Hence, A has rank k; we denote by U_k the top k eigenvectors. We denote by $\widehat{A} = \widetilde{U}\Lambda\widetilde{U}^\top$ the matrix with clean eigenvalues and noisy eigenvectors. Given the attention-based estimator introduced in (3), we consider the ideal case where the spectral denoising step studied in Section 2 is exact, and set $W_V W_V^\top = \frac{1}{\alpha} W_Q W_K^\top = \Lambda$. With this choice of W_Q, W_K, W_V, the attention-based denoising map from (3), simplifies into an analytically tractable surrogate, which we use to study denoising under the assumption that the clean eigenvalues (and hence $\widehat{A}$) are known

$$g_{\text{sm}}(\widehat{A};\alpha) = \sigma(\alpha\widehat{A}) \cdot \widehat{A} \cdot (\sigma(\alpha\widehat{A}))^\top, \tag{19}$$

where σ applies a row-wise softmax operation. The model in (19) isolates the role of the softmax and the outer product, with the parameter α controlling how sharp the softmax is. We define the α-dependent denoising loss as

$$\ell_{\text{soft}}(\alpha) = \mathbb{E}_{\mathcal{E}} \left\| g_{\text{sm}}(\widehat{A};\alpha) - A \right\|_F^2, \tag{20}$$

where $\mathcal{E}$ is the noise matrix. Our goal is to show that the minimum of this loss is smaller than the loss when we have the clean eigenvalues with the noisy eigenvectors. We denote the latter as ℓ_{basis}, since it is essentially the loss caused by the mismatch between the noisy eigenbasis $\widetilde{U}$ and the original U:

$$\ell_{\text{basis}} = \mathbb{E}_{\mathcal{E}} \left\| \widehat{A} - A \right\|_F^2. \tag{21}$$

Next, we show that the softmax denoiser approximates projection onto the clean eigenbasis, which itself improves over ℓ_{basis}. Concretely, we proceed in two steps. First, in Lemma 2 we show that projection onto the clean eigenbasis strictly improves over ℓ_{basis}. Then, in Theorem 3 we show that for a suitable choice of α, the softmax denoiser approximates this projection well enough to inherit the improvement. The connection between softmax and the aforementioned projection becomes clearer by noting that by replacing $\widehat{A}$ in the softmax denoiser with the clean adjacency matrix A, we get

$$\lim_{\alpha\to\infty} \sigma(\alpha A) = P_U, \tag{22}$$

where $P_U = U_k U_k^\top$ is the projection matrix onto the span of the top-k eigenvectors of A. This motivates studying the loss attained by projecting onto the clean eigenbasis

$$\ell_{\text{proj}} = \mathbb{E}_{\mathcal{E}} \left\| P_U \widehat{A} P_U^\top - A \right\|_F^2. \tag{23}$$

The following lemma shows that this projection strictly reduces the loss, since it removes the components of the noise caused by the eigenbasis rotating outside of the clean subspace. The proof can be found in Appendix C.6

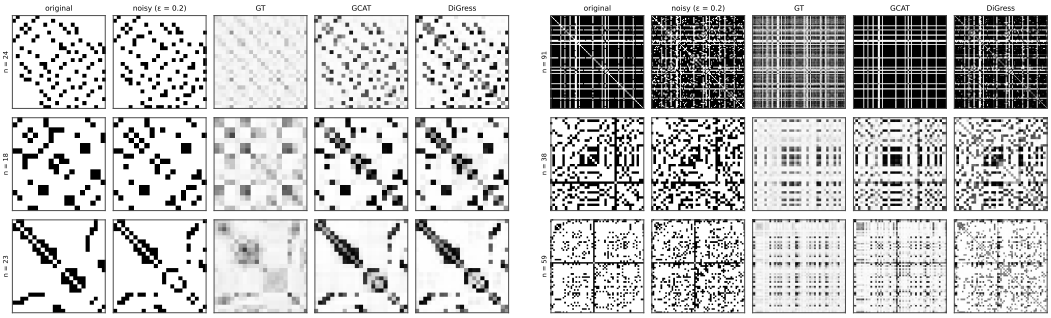
Lemma 2 (Projection improves denoising)**.** *Projecting onto the principal eigenbasis of the original graph strictly reduces the loss, i.e.,*

$$\ell_{\text{proj}} < \ell_{\text{basis}}. \tag{24}$$

When denoising we do not have access to the clean adjacency matrix A, only to $\widehat{A}$. The softmax denoiser $\sigma(\alpha\widehat{A})$ can therefore be viewed as an *approximate* projection: when $\widehat{A}$ is close to A, applying softmax with sufficiently large α yields a matrix close to P_U. The gap between $\ell_{\text{soft}}(\alpha)$ and ℓ_{proj} depends on the choice of α: for very small or very large α the gap is large, but the following theorem shows that there exist intermediate choices of α for which the gap is smaller than the improvement gained by projection i.e., $\ell_{\text{basis}} - \ell_{\text{proj}}$, which in turn means that the softmax denoiser strictly improves upon ℓ_{basis}. The proof can be found in Appendix C.7.

Theorem 3. *Assume that the noise is sufficiently small, i.e.,* $\mathbb{E}\left[\|\mathcal{E}\|_F^2\right] \ll \min\left(1, \frac{m^2}{\log^2 k}\right)$ *and that* $m_i = m$ *for* $i = 0, ..., k$*. Then, there exists* $\alpha^\star > 0$ *such that*

$$\ell_{\text{soft}}(\alpha^\star) - \ell_{\text{proj}} < \ell_{\text{basis}} - \ell_{\text{proj}}. \tag{25}$$



(a) ENZYMES (b) COLLAB

Figure 1: Denoising at $\varepsilon = 0.2$. Each row is a validation graph: its clean adjacency, the corrupted input shared by all three models, and the edge probabilities each model predicts (darker is higher).

4 Numerical Experiments

Our experiments aim to assess whether the theoretical advantages of GCA translate into improved graph denoising and generation. We first compare a standard graph transformer (GT) with a graph convolutional attention transformer (GCAT), that is, a transformer that uses GCA in its attention layers. We examine whether the denoising gains follow the spectral diversity predicted by Theorem 1 and whether they extend to graph generation. Finally, we compare GCAT with the DiGress architecture in terms of generation quality and inference time, showing that GCAT achieves comparable quality with faster inference.

Models. Our baseline, GT, is a graph transformer with L multi-head attention blocks. Its input tokens are the top k eigenvectors of the noisy adjacency matrix $\widetilde{A}$, without additional node or edge features. The adjacency logits are obtained from a learned linear combination of the per-head attention scores in the final block. GCAT uses the same architecture and differs only in the query and key projections. At layer ℓ, GT uses

$$Q^{(\ell)} = X^{(\ell)} W_Q^{(\ell)}, \qquad K^{(\ell)} = X^{(\ell)} W_K^{(\ell)}, \tag{26}$$

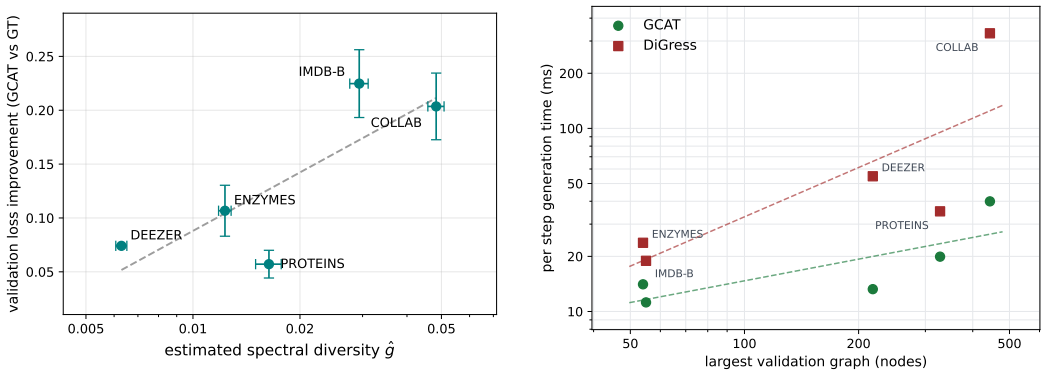
whereas GCAT uses

$$Q^{(\ell)} = \sum_{p=0}^{P-1} \widetilde{A}^p X^{(\ell)} H_{Q,p}^{(\ell)}, \qquad K^{(\ell)} = \sum_{p=0}^{P-1} \widetilde{A}^p X^{(\ell)} H_{K,p}^{(\ell)}. \tag{27}$$

Thus, GCAT forms queries and keys using learnable graph filters on the noisy input. Architectural details are given in Appendix B.2.1. For the generative experiments, we use the diffusion framework of DiGress (Vignac et al., 2022) with three denoising architectures: GT, GCAT, and the original DiGress architecture. We match their parameter counts and provide no additional structural features to any of the models, allowing us to compare the effect of the denoising architecture within the same generative framework.

Datasets. We use real benchmark datasets from PyTorch Geometric's `TUDataset` collection except one. All graphs are treated as undirected and unattributed: we keep only the adjacency structure and discard any node or edge labels supplied by the source dataset. We use five collections of variable-size unattributed graphs, all from `TUDataset` except DEEZER-EGO-NETS. PROTEINS (Borgwardt et al., 2005) has $1{,}113$ protein-structure graphs (up to 620 nodes) with secondary-structure elements as nodes. ENZYMES (Borgwardt et al., 2005) has 600 enzyme tertiary-structure graphs (up to 126 nodes) across six EC classes. IMDB-BINARY (Yanardag & Vishwanathan, 2015) provides $1{,}000$ movie co-appearance ego networks (up to 136 nodes, small and dense), and COLLAB (Yanardag & Vishwanathan, 2015) provides $5{,}000$ scientific-collaboration ego networks from three physics communities (up to 492 nodes, heavy-tailed). DEEZER-EGO-NETS (Rozemberczki & Sarkar, 2020) contains $9{,}629$ Deezer user ego networks with mutual-friendship edges (up to 363 nodes).

Experiments and metrics. We assess denoising through validation loss and compare the predicted edge probabilities with the clean adjacency matrices. To relate the denoising results to Theorem 1,



(a) Improvement against spectral diversity

(b) Generation cost against graph size

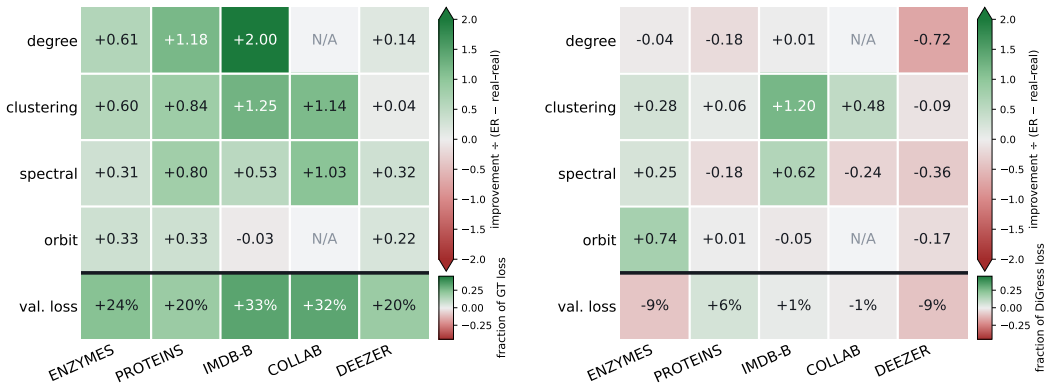
Figure 2: **(a)** Validation-loss improvement of GCAT over GT against the estimated spectral diversity $\widehat{\mathrm{SD}}(\mathcal{D}_{\mathcal{G}})$ (Appendix B.6); error bars are standard deviations over seeds and the dashed line a least-squares fit in $\log \widehat{\mathrm{SD}}(\mathcal{D}_{\mathcal{G}})$ ($r = 0.81$). **(b)** Time per reverse-diffusion step against the largest validation graph, for capacity-matched models (≈670k parameters); dashed lines are log–log fits, $n^{0.39}$ for GCAT and $n^{0.89}$ for DiGress.

we estimate spectral diversity using the KNN procedure described in Appendix B.6. For generation, following (You et al., 2018), we compare the distributions of node degrees, clustering coefficients, four-node orbits, and normalized Laplacian eigenvalues using MMD^2. For each feature f, let $m^{(f)}(M)$ denote the median MMD^2 between graphs generated by model M and real validation graphs over 25 scoring repeats. We compute a real-real reference $\mathrm{rr}^{(f)}$ from two disjoint samples of validation graphs. We also compute an Erdős-Rényi reference $\mathrm{er}^{(f)}$ using graphs drawn from $G(n, p)$ with the node counts and edge densities of the second validation sample. We normalize each score as

$$s^{(f)}_{\mathrm{norm}}(M) = \frac{m^{(f)}(M) - \mathrm{rr}^{(f)}}{\mathrm{er}^{(f)} - \mathrm{rr}^{(f)}}.$$

We then report the difference $\Delta^{(f)}_{\mathrm{norm}} = s^{(f)}_{\mathrm{norm}}(\mathrm{GT}) - s^{(f)}_{\mathrm{norm}}(\mathrm{GCAT})$, so positive values favor GCAT. We mark a comparison as N/A when the Erdős-Rényi median does not exceed the 90th percentile of the real-real scores, since the references are then insufficiently separated. For the three-seed GT-GCAT comparison, we average model scores across seeds before taking their difference. Sampling and kernel details are given in Appendix B.4.

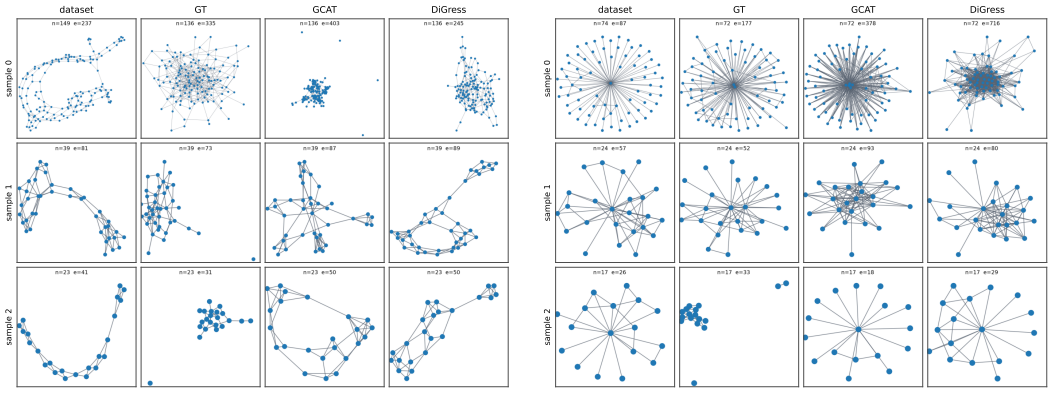
Results. GCAT achieves lower denoising validation loss than GT across the five datasets. The examples in Figure 1 illustrate this difference on ENZYMES and COLLAB. Given the same corrupted graph, GCAT concentrates its predicted edge probabilities more closely on the structure of the clean graph. Moreover, Figure 2a shows that datasets with higher estimated spectral diversity tend to exhibit larger denoising gains. The correlation is consistent with Theorem 1, although PROTEINS falls below the overall trend. The denoising gains also carry over to generation. Figure 3a shows that GCAT improves on GT across most combinations of dataset and graph statistic. When compared with the DiGress architecture at matched capacity, the results are mixed rather than uniformly favoring either architecture. As shown in Figure 3b, GCAT performs better on ENZYMES and IMDB-BINARY, DiGress performs better on DEEZER-EGO-NETS, and the two are broadly comparable on COLLAB and PROTEINS. Figure 4 provides examples of generated graphs alongside validation graphs. However, Figure 2b reveals an important difference in inference cost. While GCAT and the DiGress architecture achieve comparable generation quality, GCAT requires less time per reverse-diffusion step across the evaluated graph sizes, with the gap increasing for larger graphs. Reporting time per step isolates the cost of evaluating the denoiser from the number of diffusion steps. Both architectures use attention over node pairs, but the DiGress architecture also updates edge representations throughout the network, increasing the cost of each evaluation. Thus, GCAT is more scalable, i.e., it achieves comparable generation quality with faster inference in the evaluated setting.



(a) GCAT against GT

(b) GCAT against DiGress

Figure 3: Generation quality; green means the first-named model is closer to the data. The MMD rows show the difference in reference-normalized MMD^2 (Appendix B.4), N/A where the two references do not separate; below the rule, on its own scale, is the validation-loss improvement as a fraction of the baseline's loss. **(a)** GCAT against GT, mean of three seeds. **(b)** GCAT against DiGress at matched capacity.



(a) PROTEINS

(b) DEEZER-EGO-NETS

Figure 4: Samples from each model beside validation graphs of matching size; rows are independent draws.

5 Conclusions

We studied graph denoising from a spectral perspective to understand how attention-based architectures operate in graph diffusion models. Under a denoising objective, we showed that standard attention scores learn an average spectral rule across the training distribution and become suboptimal when graph spectra vary. We introduced spectral attention scores, characterized their advantage in terms of spectral diversity, and proposed graph convolutional attention (GCA) as a practical realization based on graph-filtered queries and keys. For asymptotically large SBMs, GCA matches the optimal loss of spectral attention under suitable conditions. We further showed that softmax provides additional denoising (for a suitable α) by approximately projecting noisy eigenvectors onto the clean eigenspace. Across synthetic and real datasets, replacing standard query–key projections with GCA improved graph denoising, with gains correlated with spectral diversity. These gains also extended to graph diffusion. Furthermore, replacing the DiGress architecture with a simpler GCA-based architecture within the same generative framework achieved comparable graph generation quality with faster inference. Overall, our results suggest that spectral adaptation is a key ingredient for effective graph denoising and generative diffusion, and provide both a theoretical explanation and a practical architectural principle for designing stronger graph diffusion models.

AI USE STATEMENT

In this work, we used generative AI tools for assist in the writing of proofs, providing feedback on experiment methodology and implementation, supporting qualitative and thematic data analysis and interpreting results. We have not used generative AI tools for generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or assist with translation, clean and reformat dataset.

Additionally, we used generative AI tools for create or edit software code, summarize or analyse existing literature, sourcing/searching for information, edit a research paper to improve readability and identify relevant literature. We have reviewed all AI-assisted work. We have double-checked all the code generated by an AI, we have used it only for polishing text selectively, reviewing the output carefully, we have verified the proposed relevant literature by manually checking the papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Florent Benaych-Georges and Raj Rao Nadakuditi. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices, 2010. URL https://arxiv.org/abs/0910.2120.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint arXiv:2303.01028*, 2023.

Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. In *Proceedings of the Thirteenth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pp. 47–56, 2005. doi: 10.1093/bioinformatics/bti1007.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 181–191, 2022.

Hao Deng and Bo Liu. Gadpn: Graph adaptive denoising and perturbation networks via singular value decomposition. *arXiv preprint arXiv:2601.08230*, 2026.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34: 22667–22681, 2021.

Soheil Feizi, Daniel Marbach, Muriel Médard, and Manolis Kellis. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature biotechnology*, 31(8): 726–733, 2013.

H Robert Frost. Denoising deterministic networks using iterative fourier transforms. *arXiv preprint arXiv:2602.00790*, 2026.

Ayush Garg. Graph transformers without positional encodings. *arXiv preprint arXiv:2401.17791*, 2024.

Ran He, Karthik Gopinath, Christian Desrosiers, and Herve Lombaert. Spectral graph transformer networks for brain surface parcellation. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 372–376. IEEE, 2020.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 66–74, 2020.

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

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Nathaniel Josephs, Wenrui Li, and Eric D Kolaczyk. Network recovery from unlabeled noisy samples. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pp. 1268–1273. IEEE, 2021.

Charilaos I Kanatsoulis, Evelyn Choi, Stephanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. *arXiv preprint arXiv:2502.01122*, 2025.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in neural information processing systems*, 34:21618–21629, 2021.

Jia Li, Mengzhou Liu, Honglei Zhang, Pengyun Wang, Yong Wen, Lujia Pan, and Hong Cheng. Mask-gvae: Blind denoising graphs via partition. In *Proceedings of the Web Conference 2021*, pp. 3688–3698, 2021.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 4257–4267, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d0921d442ee91b896ad95059d13df618-Abstract.html.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Jiayi Luo, Qingyun Sun, Haonan Yuan, Xingcheng Fu, and Jianxin Li. Robust graph learning against adversarial evasion attacks via prior-free diffusion-based structure purification. In *Proceedings of the ACM on Web Conference 2025*, pp. 2098–2110, 2025.

Hanbaek Lyu, Yacoub H Kureh, Joshua Vendrow, and Mason A Porter. Learning low-rank latent mesoscale structures in networks. *Nature Communications*, 15(1):224, 2024.

Jiahong Ma, Mingguo He, and Zhewei Wei. Polyformer: Scalable node-wise filters via polynomial graph transformer. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2118–2129, 2024.

Félix Marcoccia, Cédric Adjih, and Paul Mühlethaler. Netdiff: Deep graph denoising diffusion for ad hoc network topology generation. *arXiv preprint arXiv:2410.08238*, 2024.

Peyman Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE signal processing magazine*, 30(1):106–128, 2012.

Peyman Milanfar and Mauricio Delbracio. Denoising: a powerful building block for imaging, inverse problems and machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 383(2299), 2025.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021. URL https://proceedings.mlr.press/v139/nichol21a.html.

Zihan Pengmei and Zimu Li. Technical report: The graph spectral token–enhancing graph transformers with spectral information. *arXiv preprint arXiv:2404.05604*, 2024.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*, 2024.

Samuel Rey, Victor Tenorio, Sergio Rozada, Luca Martino, and Antonio G Marques. Over-parametrized deep encoder-decoder schemes for inputs and outputs defined over graphs. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 855–859. IEEE, 2021.

Samuel Rey, Victor M Tenorio, and Antonio G Marqués. Robust graph filter identification and graph denoising from signal observations. *IEEE Transactions on Signal Processing*, 71:3651–3666, 2023.

Mattia Rigotti, Nicholas Thumiger, and Thomas Frick. Gist: Gauge-invariant spectral transformers for scalable graph neural operators. *arXiv preprint arXiv:2603.16849*, 2026.

Sergio Rozada, Yiming Qin, Manuel Madeira, Pascal Frossard, and Alejandro Ribeiro. DiPhon: Diffusion on graphons for scalable graph generation. *arXiv preprint arXiv:2607.07232*, 2026.

Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1325–1334. ACM, 2020. doi: 10.1145/3340531.3411866.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3rd edition, 1976.

Santiago Segarra, Antonio G Marques, Gonzalo Mateos, and Alejandro Ribeiro. Network topology inference from spectral templates. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):467–483, 2017.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

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

Victor M Tenorio, Samuel Rey, and Antonio G Marques. Robust graph neural network based on graph denoising. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*, pp. 578–582. IEEE, 2023.

Victor M Tenorio, Samuel Rey, and Antonio G Marques. Blind deconvolution of sparse graph signals in the presence of perturbations. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 9406–9410. IEEE, 2024.

Puja Trivedi, Ryan Rossi, David Arbour, Tong Yu, Franck Dernoncourt, Sungchul Kim, Nedim Lipka, Namyong Park, Nesreen K Ahmed, and Danai Koutra. Leveraging graph diffusion models for network refinement tasks. *arXiv preprint arXiv:2311.17856*, 2023.

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

Han Wang, Zhili Qin, Zejun Sun, Qinli Yang, and Junming Shao. Unsupervised graph denoising via feature-driven matrix factorization. *Information Sciences*, 661:120156, 2024.

Yun-Jhong Wu, Elizaveta Levina, and Ji Zhu. Generalized linear models with low rank effects for network data. *arXiv preprint arXiv:1705.06772*, 2017.

Hui Xu, Liyao Xiang, Jiahao Yu, Anqi Cao, and Xinbing Wang. Speedup robust graph structure learning with low-rank information. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2241–2250, 2021.

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1365–1374. ACM, 2015. doi: 10.1145/2783258.2783417.

Tianmeng Yang, Jiahao Meng, Min Zhou, Yaming Yang, Yujing Wang, Xiangtai Li, and Yunhai Tong. You can't ignore either: Unifying structure and feature denoising for robust graph learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4178–4182, 2024.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5694–5703. PMLR, 2018. URL `http://proceedings.mlr.press/v80/you18a.html`.

Bingxin Zhou, Ruikun Li, Xuebin Zheng, Yu Guang Wang, and Junbin Gao. Graph denoising with framelet regularizers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 7606–7617, 2024.

A Limitations and Future Directions

The limitations of this work concern the scope of the experiments and the assumptions of the theoretical analysis. Our experiments cover a limited set of graph datasets. Testing GCA on more real and synthetic graphs would help identify settings in which it is effective and cases in which it fails. We also use eigenvectors as input features in most of our analysis. Studying how GCA interacts with other positional encodings, such as R-PEARL (Kanatsoulis et al., 2025), is a natural next step. Our theoretical results for GCA rely in part on stochastic block models. Extending the analysis to other graph families could clarify whether the same spectral mechanism explains denoising gains more generally. It would also be useful to understand theoretically why the DiGress architecture achieves generation quality comparable to GCAT despite their different designs. Finally, the generative comparison is limited in its number of training seeds, datasets, and hyperparameter settings. Its results should therefore be viewed as preliminary. A broader evaluation is needed to determine how consistently GCAT matches the generation quality of DiGress while reducing inference time.

B Experimental Details and Additional Results

B.1 Datasets

We evaluate on a mix of **real benchmark graphs** drawn from PyTorch Geometric's `TUDataset` collection and **synthetic SBMs** sampled with controllable block-size heterogeneity. All graphs are treated as undirected and unattributed: we keep only the adjacency structure and discard any node or edge labels supplied by the source dataset. Unless stated otherwise, we use a $70/10/20$ train/validation/test split with a fixed seed.

B.1.1 Real datasets

We use five collections of variable-size unattributed graphs.

- **PROTEINS** (Borgwardt et al., 2005), from TUDataset: $1\,113$ protein-structure graphs in which nodes are secondary-structure elements connected when they are neighbours in the amino-acid sequence or in 3D space. Graph size ranges up to 620 nodes, with most graphs in the few-tens to low-hundreds range.
- **ENZYMES** (Borgwardt et al., 2005), from TUDataset: 600 enzyme tertiary-structure graphs spanning the six EC top-level classes. Graph size ranges up to 126 nodes, typically ≈ 30–60 nodes per graph.
- **IMDB-BINARY** (Yanardag & Vishwanathan, 2015), from TUDataset: $1\,000$ movie-collaboration ego networks in which nodes are actors/actresses and edges connect those that appeared together in a film. Graph size ranges up to 136 nodes with a strongly skewed distribution toward smaller, dense ego networks.
- **COLLAB** (Yanardag & Vishwanathan, 2015), from TUDataset: $5\,000$ scientific-collaboration ego networks built from three physics communities. Graph size ranges up to 492 nodes, with heavy-tailed size and degree distributions.
- **DEEZER-EGO-NETS** (Rozemberczki & Sarkar, 2020): $9\,629$ ego networks of Deezer users, with edges representing mutual-friend relationships in the ego's neighbourhood. Graph size ranges up to 363 nodes; the distribution is dominated by ego networks of a few dozen nodes.

In all five cases we drop the original node and edge attributes and predict the binary adjacency only. This matches the unconditional graph-generation setting used in prior work (You et al., 2018; Liao et al., 2019; Vignac et al., 2022). Splits use the PyG-default ordering with a fixed seed of 42 and a $70/10/20$ train/val/test ratio. Every batch is padded to the dataset's published per-graph maximum ($n_{\max}$ in Table 1), which also bounds the eigendecomposition. Table 1 summarises the validation split that every generative evaluation in this paper is scored against.

Table 1: The five datasets of the generative comparison. "$n_{\max}$" is the published per-graph maximum used for padding; the last three columns describe the validation split, which supplies the reference graphs for every MMD^2 and the graphs on which validation losses are measured. Because a sampling batch is padded to its largest member, the maximum validation size, not the mean, sets the per-step memory and time of DiGress's $n \times n$ edge tensors.

Dataset	# graphs	$n_{\max}$	val. graphs	mean n (val.)	max n (val.)
ENZYMES	600	126	60	27.4	54
PROTEINS	1 113	620	111	38.5	328
IMDB-BINARY	1 000	136	100	19.0	55
COLLAB	5 000	492	500	74.2	444
DEEZER-EGO-NETS	9 629	363	962	23.2	218

B.2 Architectural details

B.2.1 Graph Transformer and Graph Convolutional Attention Transformer

Shared pipeline. Both models read the noisy adjacency $A \in \{0,1\}^{n\times n}$ and nothing else: no node attributes, no hand-crafted structural features, and no timestep. We extract the k eigenvectors of A with the largest absolute eigenvalues, $V \in \mathbb{R}^{n\times k}$ (sign-fixed so that the first non-zero entry of each column is positive), and lift them to the model dimension d:

$$X^{(0)} = V\,W_{\text{in}} + \mathbf{1}\,b_{\text{in}}^{\top}, \qquad W_{\text{in}} \in \mathbb{R}^{k\times d}. \tag{28}$$

Each of the first $L-1$ blocks applies a residual multi-head attention update with $\mathrm{SD}(\mathcal{D}_{\mathcal{G}})$ heads of dimension $d_k = d_v = d/h$:

$$X^{(\ell+1)} = \mathrm{LN}\Big(X^{(\ell)} + \mathrm{MHA}^{(\ell)}\big(A,\, X^{(\ell)}\big)\Big), \qquad \ell = 0,\dots,L-2, \tag{29}$$

where, given queries, keys, and values $Q^{(\ell)}, K^{(\ell)}, V^{(\ell)}$, each head computes

$$S_i^{(\ell)} = \frac{Q_i^{(\ell)} {K_i^{(\ell)}}^{\top}}{\sqrt{d_k}}, \qquad \mathrm{Attn}_i^{(\ell)} = \mathrm{softmax}\big(S_i^{(\ell)}\big)\,V_i^{(\ell)}, \qquad i = 1,\dots,h, \tag{30}$$

and the heads are concatenated and mixed by an output projection $W_O \in \mathbb{R}^{d\times d}$. The final block produces no node features; its per-head scores $S^{(L-1)} \in \mathbb{R}^{h\times n\times n}$ are mixed into the edge logits by a learned linear combination,

$$\widehat{A} = \sum_{i=1}^{h} w_i\, S_i^{(L-1)}, \qquad w \in \mathbb{R}^h, \tag{31}$$

and $\widehat{A}_{uv}$ is read as the log-odds of an edge between u and v. The value projection is a plain linear map in both models; the two models differ only in the Q/K projections inside each block.

Graph Transformer (GT)

$$Q^{(\ell)} = X^{(\ell)} W_Q^{(\ell)}, \qquad K^{(\ell)} = X^{(\ell)} W_K^{(\ell)}. \tag{32}$$

Graph Convolutional Attention Transformer (GCAT)

$$Q^{(\ell)} = \mathrm{LN}\left(\sum_{p=0}^{P-1} \bar{A}^{\,p}\, X^{(\ell)}\, W_{Q,p}^{(\ell)}\right), \qquad K^{(\ell)} = \mathrm{LN}\left(\sum_{p=0}^{P-1} \bar{A}^{\,p}\, X^{(\ell)}\, W_{K,p}^{(\ell)}\right). \tag{33}$$

The queries and keys become learnable polynomials of the noisy adjacency, allowing two nodes to attend on the basis of their P-hop neighbourhoods. With $P = 2$ taps, used throughout, each projection is $XW_0 + \bar{A}XW_1$. The filter operator $\bar{A}$ is the raw adjacency A in the default GCAT; the capacity-matched wide variant (Appendix B.2.3) uses the symmetrically normalised $\bar{A} = D^{-1/2}AD^{-1/2}$ (degrees clamped at one), which keeps the neighbourhood term degree-invariant and is what allows the wider model to train stably on the dense datasets.

Sizes. The default configuration has $k = 16$, $d = 64$, $\mathrm{SD}(\mathcal{D}_\mathcal{G}) = 4$ heads and $L = 2$ blocks (one feature block and the score read-out), giving 25 796 parameters for GT and 42 692 for GCAT; the 17k difference is the second filter tap in Q and K together with the LayerNorms on their outputs.

B.2.2 DiGress (small)

The DiGress baseline is the upstream XEy graph transformer of Vignac et al. (2022) at reduced width, which we call DIGRESS_SMALL: a stack of $L = 4$ blocks that jointly update node features X, edge features E, and a global vector y, with FiLM-style cross-modulation between the three streams. Hidden widths are $d_X = 128$, $d_E = 32$, $d_y = 32$ with 4 attention heads, feed-forward widths $(d_{\mathrm{ff}X}, d_{\mathrm{ff}E}, d_{\mathrm{ff}y}) = (128, 32, 128)$ and input/output MLP widths $(64, 32, 64)$, for 678 468 parameters. The attention block is upstream's: the per-pair tensor $Y = (Q \odot K/\sqrt{d_f})\,(E_1 + 1) + E_2$ is FiLM-modulated by the edge stream, softmaxed per channel over the key axis, and also written back as the new edge features; y modulates both X and E and is updated from pooled node and edge summaries.

Inputs. To keep the comparison to GT and GCAT about the attention mechanism rather than the feature set, the baseline receives the same information they do. Its node input is a constant two-class one-hot placeholder (the graphs are unattributed) concatenated with the same $k = 16$ largest-magnitude eigenvectors of the noisy adjacency; its edge input is the two-class one-hot noisy adjacency; and y holds the normalised timestep t/T. DiGress's hand-crafted structural augmentation (cycle counts, connected-component indicators, Laplacian eigenvalues) is switched off. The Q/K projections are the standard linear ones.

B.2.3 Capacity-matched variants

GCAT and DIGRESS_SMALL differ by a factor of 16 in parameter count. To separate the effect of the architecture from that of capacity, we train two further arms in which each model is scaled to the other's size (Table 2). For GCAT the width d is raised from 64 to 256, which scales every projection and the filter taps; for DIGRESS_SMALL d_X is lowered from 128 to 32 with d_E, d_y, the feed-forward and the input/output MLP widths scaled in the preset's proportions (a quarter, a quarter, equal, half). The wide GCAT uses the symmetrically normalised filter operator described above; everything else, including the training protocol, is unchanged.

Table 2: The four arms used in the cross-model comparisons. Parameter counts are exact and dataset-independent.

Arm	Model	Parameters	Change from default
GT	spectral, standard Q/K	25 796	—
GCAT	spectral, convolutional Q/K	42 692	—
GCAT-256	spectral, convolutional Q/K	662 276	$d = 256$, $\bar{A} = D^{-1/2}AD^{-1/2}$
DIGRESS_SMALL	DiGress, standard Q/K	678 468	—
DIGRESS_SMALL-32	DiGress, standard Q/K	44 820	$(d_X, d_E, d_y) = (32, 8, 8)$

B.3 Training protocol

All generative runs share one protocol (Table 3); the two model families differ only in the diffusion horizon T and in the accumulation needed to fit DiGress's edge tensors into memory on the three large-graph datasets. Each run trains for 10 000 optimiser steps at 16 graphs per step, which is the same amount of data on every dataset. Validation loss is measured every 1 000 steps on the full validation split with one (t, noise) draw per graph, a checkpoint is written at every validation, and the checkpoint with the lowest validation loss is the one every generative evaluation samples from. GT and GCAT are trained with three seeds (7, 17, 27); DIGRESS_SMALL with seed 7; the two capacity-matched arms with seeds 7 and 17. All runs use bf16 mixed precision on a single GPU.

Memory. DiGress keeps a hidden feature vector per node pair, so a training step's memory grows with the square of the largest graph in the batch. On a 24 GB GPU DIGRESS_SMALL at 16 graphs

Table 3: Hyperparameters shared across all generative runs. The diffusion horizon is the one setting that differs between the two model families; loss and sample comparisons across families are therefore made on the normalised axis t/T.

Block	Setting	Value
Architecture	GT / GCAT (App. B.2.1)	$k = 16$, $d = 64$, SD$(\mathcal{D}_\mathcal{G}) = 4$, $L = 2$, $P = 2$
	DIGRESS_SMALL (App. B.2.2)	$L = 4$, $(d_X, d_E, d_y) = (128, 32, 32)$, 4 heads
	dropout	0.0
	timestep conditioning	GT / GCAT: none; DiGress: t/T appended to y
Diffusion	forward process	categorical (two edge classes)
	noise schedule	cosine-iDDPM (Nichol & Dhariwal, 2021)
	timesteps T	500 (GT / GCAT), 1 000 (DiGress)
	limit distribution	empirical edge marginal of the training split
	training timesteps	$t \sim \mathcal{U}\{0, \dots, T\}$ per graph
	sampler	ancestral
Optimisation	optimiser	AdamW with AMSGrad
	learning rate	2×10^{-4}, constant
	weight decay	10^{-12}
	gradient clipping / EMA	none / none
	loss	masked categorical cross-entropy, edge weight $\lambda_E = 5$
	precision	bf16 mixed
Schedule	optimiser steps	10 000
	graphs per optimiser step	16
	gradient accumulation (DiGress)	4×4 on PROTEINS and COLLAB, 2×8 on DEEZER
	validation cadence	every 1 000 steps, full validation split
	checkpoint selection	lowest validation loss
Data	split	70/10/20, seed 42
	padding	to the dataset's $n_{\max}$ (Table 1)

per batch peaks at 9.5 GB when padded to 200 nodes and runs out of memory at 350; PROTEINS, COLLAB and DEEZER contain training graphs of 504, 445 and 363 nodes. On those three datasets the DiGress runs therefore accumulate gradients over micro-batches (4×4, 4×4 and 2×8 graphs), which keeps 16 graphs per optimiser step and the same data and splits as every other cell. GT and GCAT carry only per-node features and need under 1 GB on every dataset.

B.4 Evaluation protocol for the improvement heatmaps

The heatmaps of Section 4 compare a *treatment* arm against a *base* arm on the same dataset: GCAT against GT (within the spectral family, three seeds), and GCAT-256 against DIGRESS_SMALL (across families, seed 7). Every cell is a normalised MMD^2 improvement, and a separate row reports the relative change in validation loss. Both are computed as follows.

Regenerated samples. For every run we reload the best-validation checkpoint and draw 24 graphs in two batches of 12 under fixed generation seeds, with node counts sampled from the training split's size distribution. These regenerated graphs, not the 12 sampled during training, are what the heatmaps score.

Statistics and kernels. We compute the SPECTRE-style MMD^2 with a Gaussian kernel on the total-variation distance between per-graph descriptors: degree histograms, clustering-coefficient histograms, the histogram of normalised-Laplacian eigenvalues ("spectral"), and ORCA orbit counts ("orbit"). The heatmaps use one bandwidth per dataset and metric, the *reference-median* bandwidth σ_{ref}: the median pairwise TV distance among the descriptors of the validation pool. This replaces the fixed bandwidths of the sweep's in-training evaluation (degree 1, clustering 0.1, spectral 1, orbit 30), whose scale is arbitrary relative to a dataset's own spread. On the five datasets the two choices agree in sign on 16 of the 17 cells they both define in the GT–GCAT heatmap and on 15 of 17 in the cross-model one. The exception common to both is the IMDB-BINARY orbit cell, which is clearly

positive at the fixed bandwidth and indistinguishable from zero at the reference one; the other is a PROTEINS orbit cell within ± 0.01 of zero at either bandwidth.

Two nulls from one protocol. An MMD^2 value on its own does not say whether a model is close to the data, because the statistic has a finite-sample floor that depends on the dataset and the metric. We therefore score every model against two nulls drawn under exactly the same protocol. In each of 25 repeats we draw 12 validation graphs as the reference set A, 12 disjoint validation graphs B, and a set E of Erdős–Rényi graphs matched one-for-one to B in node count and edge density ($\mathrm{SD}(\mathcal{D}_{\mathcal{G}})(n, p)$ with $p = m/\binom{n}{2}$), and take 12 of the model's 24 regenerated graphs. $\mathrm{MMD}^2(A, B)$ is the *real–real* reference (two samples from the data), $\mathrm{MMD}^2(A, E)$ is the *density-matched ER* reference (a sample that has the right size and density and nothing else), and $\mathrm{MMD}^2(A, \text{model})$ is the model's score. Each is the median over the 25 repeats, and the reference, ER and model draws use separate random streams so that adding a model never moves another model's score.

Cells. Writing rr and er for the two reference medians and $m(M)$ for the median MMD^2 of model M, we first score each model by

$$s_{\mathrm{norm}}(M) = \frac{m(M) - \mathrm{rr}}{\mathrm{er} - \mathrm{rr}}. \tag{34}$$

The real–real reference gives a score of zero, and the density-matched ER reference gives a score of one. A model can score outside this range. The denominator scales each model's distance from the real–real reference by the gap that this metric resolves between the two references. The heatmap cell is the difference between the base and treatment scores:

$$\Delta_{\mathrm{norm}} = s_{\mathrm{norm}}(\text{base}) - s_{\mathrm{norm}}(\text{treat}) = \frac{m(\text{base}) - m(\text{treat})}{\mathrm{er} - \mathrm{rr}}. \tag{35}$$

Thus $+1$ means the treatment moved the model one full "ER gap" closer to the data than the base, and green cells favour the treatment. For the three-seed comparison we average the per-seed model scores before differencing. A cell is marked N/A when the ER median does not exceed the 90th percentile of the real–real draws, in which case the statistic cannot distinguish the dataset from density-matched noise and has no power to rank models on it. Colours are clipped at ± 2; the printed values are exact.

Loss rows. Within the spectral family (GCAT against GT) the loss row reports the validation loss of the final checkpoint as a fraction of the base arm, $(\mathcal{L}_{\mathrm{GT}} - \mathcal{L}_{\mathrm{GCAT}})/\mathcal{L}_{\mathrm{GT}}$, averaged over the three seeds. A single validation pass draws one (t, noise) pair per graph and has a standard deviation of about 0.05 (spectral) to 0.10 (DiGress) in the loss, which is comparable to some of the effects; the row therefore re-evaluates each checkpoint over the whole validation split $\lceil 3\,000/n_{\mathrm{val}} \rceil$ times (at most 50) with a fresh seed per pass, for $\approx 3\,000$ $(\text{graph}, t, \text{noise})$ draws per run and a standard error of about 0.02. Across families the total loss is not comparable, because the node term is identically zero for DiGress (its node class is constant) while the families also use different horizons. The cross-model loss row therefore reports the *edge* cross-entropy alone, measured per decile of t/T with every validation graph's t drawn uniformly inside the decile and at least three passes per decile (600–2 900 draws), and averages the ten decile means; the row shows $(\mathcal{L}^E_{\text{DiGress}} - \mathcal{L}^E_{\text{GCAT-256}})/\mathcal{L}^E_{\text{DiGress}}$. Note that the MMD^2 cells score the best-validation checkpoint while the loss rows read the final (step 10 000) checkpoint.

B.5 Sampling cost

Table 4 reports the wall-clock cost of sampling at matched capacity. Per reverse step GCAT-256 is 1.7–1.8× faster than DIGRESS_SMALL on ENZYMES, PROTEINS and IMDB-B, and 4–8× faster on DEEZER and COLLAB. The gap tracks the largest graph in a batch rather than the mean, because DiGress carries an $n \times n \times d_E$ hidden edge tensor through every block and a batch is padded to its largest member, whereas GT and GCAT carry per-node features only. Width changes neither picture by much: scaling GCAT from 43k to 662k parameters adds 10–40% to its sampling time, and shrinking DiGress from 678k to 45k removes 20–45%, because the n^2 tensors remain.

B.6 Spectral diversity metric

The x-axis of Figure 2a is a non-parametric estimate $\widehat{\mathrm{SD}}(\mathcal{D}_{\mathcal{G}})$ of the improvement gap of Theorem 1, computed on the validation split of each dataset and normalised per graph so that it is measured

Table 4: Sampling cost at matched capacity (≈670k parameters). Each model generates 12 graphs on one A5000 during validation; we report *wall*, the end-to-end wall-clock time for that batch, and *per-step*, the same time divided by the number of reverse diffusion steps ($T = 500$ for GCAT, $T = 1000$ for DIGRESS_SMALL), which removes the differing sampling horizons and isolates the per-step cost of one denoiser call. Ratios are DiGress / GCAT, so values above 1 favour GCAT. Times are medians over validation cycles, averaged over seeds (GCAT-256: seeds 7 and 17; DiGress: seed 7).

	nodes (val.)		GCAT-256 (662k)		DIGRESS_SMALL (678k)		ratio	
Dataset	mean	max	wall (s)	per-step (ms)	wall (s)	per-step (ms)	wall	per-step
ENZYMES	27.4	54	7.0	14.1	23.7	23.7	3.4×	1.7×
PROTEINS	38.5	328	10.0	19.9	35.2	35.2	3.5×	1.8×
IMDB-B	19.0	55	5.6	11.2	18.9	18.9	3.4×	1.7×
COLLAB	74.2	444	20.0	39.9	330.5	330.5	16.5×	8.3×
DEEZER	23.2	218	6.6	13.2	54.8	54.8	8.3×	4.1×

per adjacency entry, the unit of the validation loss on the y-axis. The estimand, the per-graph normalisation, the plug-in estimator, the frame-alignment step that makes the cross-graph average meaningful, the noise model, the bias floor of the plug-in and the permutation null that measures it are detailed below, followed by the hyperparameters of the figure.

Population estimand. For each graph i in a dataset with n_i nodes, let A_i be its clean adjacency, $\widetilde{A}_i$ a noisy observation, and $(\widehat{V}_{k,i}, \widetilde{\Lambda}_{k,i})$ the top-k eigenpairs of $\widetilde{A}_i$ (the k largest eigenvalues and their eigenvectors). Define

$$B_i \;=\; \frac{1}{n_i}\,\widehat{V}_{k,i}^{\top} A_i \widehat{V}_{k,i} \;\in\; \mathbb{R}^{k\times k}, \tag{36}$$

the clean adjacency expressed in the noisy top-k eigenbasis, scaled by the graph's own size (the reason for the factor $1/n_i$ is given next). Theorem 1 identifies the gap between the optimal attention-score losses with $\mathrm{SD}(\mathcal{D}_\mathcal{G}) = \mathrm{tr}(\mathrm{Cov}\,\mathbb{E}[B \mid \widetilde{\Lambda}_k]) = \mathbb{E}\|\mathbb{E}[B \mid \widetilde{\Lambda}_k] - \mathbb{E}[B]\|_F^2$, the variance across graphs of the optimal spectral predictor $f^\star(\widetilde{\Lambda}_k) = \mathbb{E}[B \mid \widetilde{\Lambda}_k]$. Its ratio to the total variance $\mathrm{tr}(\mathrm{Cov}\,B) = \mathbb{E}\|B - \mathbb{E}[B]\|_F^2$ is the fraction of variance explained by the noisy spectrum, the R^2 of $f^\star$ (Propositions 1–2); the theorem states the loss gap itself in terms of $\mathrm{SD}(\mathcal{D}_\mathcal{G})$, and $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ is what the figure plots.

Normalisation by graph size. Without the factor $1/n_i$, because $\widehat{V}_{k,i}$ has orthonormal columns,

$$\|\widehat{V}_{k,i}^{\top} A_i \widehat{V}_{k,i}\|_F^2 \;=\; \|\widehat{V}_{k,i}\widehat{V}_{k,i}^{\top} A_i \widehat{V}_{k,i}\widehat{V}_{k,i}^{\top}\|_F^2, \tag{37}$$

so $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ is a squared error on the rank-k part of the $n_i \times n_i$ adjacency, summed over all n_i^2 entries of the graph. The validation loss it is compared against is a cross-entropy averaged over the node pairs of each graph and then over graphs, so the matching quantity is the per-graph gap divided by n_i^2, averaged over graphs. Scaling B_i by $1/n_i$ before any averaging achieves exactly this, since every term of $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ below is quadratic in B. A single dataset-level divisor such as $n_{\max}^2$ does not: graph sizes within a dataset vary by up to a factor of eight (PROTEINS and DEEZER-EGO-NETS span 10 to 84 nodes), the mean size is between 19 and 58 against band maxima of 54 to 124 (Table 5), and dividing by $n_{\max}^2$ instead understates $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ by a factor of 6 to 20 depending on the dataset. The conditioning vector $\widetilde{\Lambda}_{k,i}$ is left on its natural scale; the normalisation acts on the regression target only.

Plug-in estimator. The population quantities are inaccessible and each is replaced by a finite-sample plug-in. Given N graphs with paired observations $\{(B_i, \widetilde{\Lambda}_{k,i})\}_{i=1}^N$, the optimal predictor is estimated by leave-one-out kNN regression on the noisy eigenvalues,

$$\widehat{B}_i \;=\; \frac{1}{m}\sum_{j\in\mathcal{N}_m(i)} B_j, \tag{38}$$

where $\mathcal{N}_m(i)$ is the set of m graphs other than i whose sorted eigenvalue vectors $\widetilde{\Lambda}_{k,j}$ are closest to $\widetilde{\Lambda}_{k,i}$ in Euclidean distance. The plug-in estimates of $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ and of the total variance use the same sample,

$$\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G}) \;=\; \frac{1}{N}\sum_{i=1}^{N}\|\widehat{B}_i - \bar{B}\|_F^2, \qquad \widehat{\mathrm{tr}(\mathrm{Cov}\, B)} \;=\; \frac{1}{N}\sum_{i=1}^{N}\|B_i - \bar{B}\|_F^2, \tag{39}$$

with $\bar{B} = \frac{1}{N}\sum_i B_i$. The estimator is non-parametric: no analytic form for $f^\star$ is assumed. $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ is the reported quantity; the ratio $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})/\widehat{\mathrm{tr}(\mathrm{Cov}\, B)}$ is recorded beside it but, being scale-free, does not carry the per-entry information the comparison with a loss requires.

Frame alignment. The matrix B_i is not invariant to the gauge of $\widehat{V}_{k,i}$: replacing $\widehat{V}_{k,i}$ by $\widehat{V}_{k,i}Q_i$ for any $Q_i \in O(k)$ maps $B_i \to Q_i^\top B_i Q_i$. Averaging matrices across different per-graph frames adds incommensurable quantities and inflates $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$, because Q_i is partly predictable from $\widetilde{\Lambda}_{k,i}$ (both derive from the same noisy decomposition). We therefore align every $\widehat{V}_{k,i}$ by orthogonal Procrustes (reflections allowed) against a single dataset-wide reference frame, the extrinsic Grassmannian mean of the clean top-k eigenbases (the top-k left singular vectors of the $n \times Nk$ matrix formed by stacking the clean eigenvector blocks side by side). With one common frame, the residual rotation cannot co-vary with $\widetilde{\Lambda}_k$, and the convention artefact that would otherwise feed a spurious signal into the estimator is closed off.

Noise model. Noise is the DiGress corruption with $\bar{\alpha} = 1 - \varepsilon$: every edge of the upper triangle is flipped independently with probability $\varepsilon/2$ and mirrored, so the limit $\varepsilon = 1$ is a uniformly random graph. For the figure, ε is not fixed but drawn from the diffusion schedule of the generative runs: each graph draws its own timestep $t_i \sim \mathcal{U}\{1, \ldots, T\}$ and is corrupted at $\varepsilon_i = 1 - \bar{\alpha}_{t_i}$ of the cosine-iDDPM schedule (Nichol & Dhariwal, 2021), so that $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ is estimated under the same noise marginal that the runs' validation loss averages over (Table 3; the schedule is the runs' own, with $T = 500$, and the mean drawn level is ≈ 0.52). Two differences from the runs' forward process remain: the runs' limit distribution is the empirical edge marginal rather than uniform, and GT and GCAT are not conditioned on t, which is why $\widetilde{\Lambda}_{k,i}$ alone, without ε_i, is the conditioning vector. Fixed levels $\varepsilon \in \{0.05, 0.1, 0.2\}$ were computed alongside: the five datasets rank identically at every level and every k, and the Pearson correlation with the gain stays between $+0.77$ and $+0.82$ across all twelve (ε, k) combinations, with $\rho = +0.60$ throughout.

Padding. Every graph of a dataset is zero-padded to the maximum of its size band so that the eigenvector blocks share an ambient dimension and the Fréchet reference is well defined. Corruption is confined to each graph's real $n_i \times n_i$ block; the padding rows receive no noise, as in training, where the node mask is re-imposed after the forward process.

Bias floor. Under the null $B \perp \widetilde{\Lambda}_k$, the neighbours $\mathcal{N}_m(i)$ are effectively a random subsample of the B_j, so $\mathrm{Var}(\widehat{B}_i) \approx \mathrm{Var}(B)/m$ and $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ acquires a floor

$$\mathbb{E}\big[\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G}) \;\big|\; B \perp \widetilde{\Lambda}_k\big] \;\approx\; \Big(\frac{1}{m} - \frac{1}{N-1}\Big)\,\widehat{\mathrm{tr}(\mathrm{Cov}\, B)}, \tag{40}$$

which for $(m, N) = (10, 200)$ is 0.095 of the total variance. Comparing $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ directly to zero is therefore uninformative.

Permutation null. We measure the floor on each dataset by shuffling the conditioning vectors with a uniformly random permutation π and recomputing the same plug-in, $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})_{\text{null}} = \widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})\big(\{(B_i, \widetilde{\Lambda}_{k,\pi(i)})\}\big)$. The shuffle preserves both marginals and breaks only the joint, so its expectation matches the kNN bias to leading order; the measured nulls are within 16% of the formula above on every dataset. Table 5 reports $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})_{\text{null}}$ beside $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$. It is 20–40% of $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ on four of the five datasets and 76% on DEEZER-EGO-NETS, whose $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ is therefore the one value close to the estimator's resolution. The figure plots the uncalibrated $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$; the calibrated margin $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G}) - \widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})_{\text{null}}$ gives $r = +0.71$ in place of $+0.81$ with the same ordering.

Hyperparameters. The figure uses $k = 8$ and $m = 10$ neighbours; $k \in \{4, 16\}$ was computed alongside and changes neither the ranking nor, beyond the range quoted above, the correlation. Graphs are the validation split of each dataset (the split every validation loss is measured on), restricted to the size band of Table 5 and capped at $N = 200$ by a seeded random subsample, because the bias floor depends on N and an uncapped comparison across datasets would compare estimator bias as much as spectra. Three seeds (42, 123, 2 024) control the subsample, the corruption draws and the per-graph timesteps; each point of the figure is the mean over the three seeds and its horizontal error bar their standard deviation. Eigendecompositions run in float64.

The gain axis. The y-axis is the mean over the three training seeds (7, 17, 27) of the difference in validation edge cross-entropy between GT and GCAT after 10 000 optimiser steps, on the full validation split with one (t, noise) draw per graph (Appendix B.3); the vertical error bar is the standard deviation over seeds. This gain is the difference between two trained models, whereas $\mathrm{SD}(\mathcal{D}_\mathcal{G})$ is the gap between the two Bayes-optimal predictors of Theorem 1; the comparison tests whether the two co-vary across datasets, not whether they agree in scale.

Table 5: Per-dataset settings and values behind Figure 2a ($k = 8$, $m = 10$, ε drawn from the cosine schedule with $T = 500$). N is the number of validation graphs after banding and the 200-graph cap; "mean n_i" is their mean size before padding. $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$ and its permutation null are means over the three estimation seeds ($\pm$ one standard deviation), the gain a mean over the three training seeds. All of $\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})$, its null and the gain are per adjacency entry.

Dataset	band $[n_{\min}, n_{\max}]$	N	mean n_i	$\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G}) \times 10^2$	$\widehat{\mathrm{SD}}(\mathcal{D}_\mathcal{G})_{\text{null}} \times 10^2$	gain (GT − GCAT)
ENZYMES	$[10, 54]$	59	27.8	1.23 ± 0.05	0.39	0.107 ± 0.024
PROTEINS	$[10, 84]$	89	31.8	1.64 ± 0.14	0.33	0.057 ± 0.013
IMDB-BINARY	$[12, 55]$	100	19.0	2.94 ± 0.17	1.17	0.225 ± 0.031
COLLAB	$[32, 124]$	200	57.6	4.83 ± 0.25	1.13	0.204 ± 0.031
DEEZER-EGO-NETS	$[11, 84]$	200	20.2	0.63 ± 0.02	0.48	0.074 ± 0.001

B.7 Compute resources

Hardware. Every experiment in this paper was run on a single workstation with two NVIDIA RTX A5000 GPUs (24 GB GDDR6 each), 64 physical / 128 logical CPU cores and ≈ 270 GB host RAM. Each training run is a single-GPU job; the two cards were used as two independent workers. The improvement-gap estimation (Appendix B.6) is CPU-only; the sweep behind Table 5 (5 datasets × 3 seeds × 4 noise levels × 3 values of k × real/null × raw/per-entry) completes in ≈ 3 minutes.

Training cost. A 10 000-step GT or GCAT run takes 7–11 minutes on the four small-graph datasets and 18–22 minutes on COLLAB; the wide GCAT-256 adds under 10% to these, since per-step cost is set by the padded graph size rather than the width. A DIGRESS_SMALL run takes 19–25 minutes on ENZYMES and IMDB-BINARY, 41 minutes on DEEZER, 97 on PROTEINS and 132 on COLLAB, the last three under gradient accumulation; the narrow DIGRESS_SMALL-32 is 20–45% faster. The GT and GCAT runs reported here (six datasets, three seeds, two arms) consumed ≈ 7.4 GPU-hours, the six DIGRESS_SMALL runs ≈ 6.6 GPU-hours, and the 24 capacity-matched runs ≈ 11.4 GPU-hours. Including the synthetic-SBM datasets, the filter-tap and normalisation ablations and the DiGress attention ablations that informed the analysis but are not reported, the sweeps behind this paper total ≈ 120 GPU-hours. Re-evaluating the 150 checkpoints of the loss rows with ≈ 3 000 draws each took 42 minutes on one GPU.

C Proofs

C.1 Proof of Proposition 1

We begin by expanding the loss using the fact that $\|M\|_F^2 = \mathrm{tr}(M^\top M)$:

$$\ell(W) = \mathbb{E}\,\mathrm{tr}\Big[(\widetilde{U} W \widetilde{U}^\top)^2 - 2A\,\widetilde{U} W \widetilde{U}^\top + A^2\Big] \tag{41}$$

Further, using $\widetilde{U}^\top \widetilde{U} = I$ and the cyclic property of trace:

- $\mathrm{tr}[(\widetilde{U}W\widetilde{U}^\top)^2] = \mathrm{tr}[\widetilde{U}W^2\widetilde{U}^\top] = \mathrm{tr}[W^2]$
- $\mathrm{tr}[A\,\widetilde{U}W\widetilde{U}^\top] = \mathrm{tr}[W\,\widetilde{U}^\top A\widetilde{U}]$

So the loss reduces to:

$$\ell(W) = \mathbb{E}\left[\mathrm{tr}(W^2) - 2\,\mathrm{tr}(W\,B) + \mathrm{tr}(A^2)\right], \quad \text{where } B = \widetilde{U}^\top A\widetilde{U} \tag{42}$$

which we can write as:

$$\ell(W) = \mathbb{E}\left[\mathrm{tr}(W^2) - 2\,\mathrm{tr}(W\,B) + \mathrm{tr}(B^2) + \mathrm{tr}(A^2 - B^2)\right] \tag{43}$$

$$\ell(W) = \mathbb{E}\left[\|W - B\|_F^2 + \|A\|_F^2 - \|B\|_F^2\right] \tag{44}$$

The minimization reduces to:

$$W^* = \underset{W}{\mathrm{argmin}}\ \mathbb{E}\left[\|W - B\|_F^2\right] = \mathbb{E}_{A,\,\mathcal{E}}[B] \tag{45}$$

Define $c := \mathbb{E}\left[\|A\|_F^2 - \|B\|_F^2\right]$. Then the minimum achievable loss is:

$$\ell_{\mathrm{A}} := \ell(W^*) = \mathbb{E}\left[\|\mathbb{E}[B] - B\|_F^2\right] + c = \mathrm{tr}(\mathrm{Cov}(\mathrm{vec}(B))) + c \tag{46}$$

C.2 Proof of Proposition 2

$$\ell(h) = \mathbb{E}\left[\left\|\widetilde{U}h(\widetilde{\Lambda})\widetilde{U}^\top - A\right\|_F^2\right] \tag{47}$$

Similarly to the linear case, we can write the loss as:

$$\ell(h) = \mathbb{E}\left[\left\|h(\widetilde{\Lambda}) - B\right\|_F^2 + \|A\|_F^2 - \|B\|_F^2\right] \tag{48}$$

The minimization reduces to:

$$h^*(\cdot) = \underset{g}{\mathrm{argmin}}\ \mathbb{E}\left[\left\|h(\widetilde{\Lambda}) - B\right\|_F^2\right] \tag{49}$$

Apply the tower property (law of iterated expectations):

$$\mathbb{E}\Big[\|h(\widetilde{\Lambda}) - B\|_F^2\Big] = \mathbb{E}_{\widetilde{\Lambda}}\Big[\mathbb{E}\Big[\|h(\widetilde{\Lambda}) - B\|_F^2 \;\Big|\; \widetilde{\Lambda}\Big]\Big] \tag{50}$$

Since $h(\widetilde{\Lambda})$ is deterministic given $\widetilde{\Lambda}$, the inner conditional expectation is a pointwise squared-error problem for each realization of $\widetilde{\Lambda}$. The minimizer of $\mathbb{E}[\|c - B\|_F^2 \mid \widetilde{\Lambda}]$ over a constant c is the conditional mean:

$$h^*(\widetilde{\Lambda}) = \mathbb{E}\Big[B \;\Big|\; \widetilde{\Lambda}\Big] = \mathbb{E}\Big[\widetilde{U}^\top A\,\widetilde{U} \;\Big|\; \widetilde{\Lambda}\Big] \tag{51}$$

In contrast to the minimizer in the linear attention class which was the unconditional mean $W^* = \mathbb{E}[B]$,

with spectral attention the minimizer is $h^*(\widetilde{\Lambda}) = \mathbb{E}[B \mid \widetilde{\Lambda}]$, i.e., the conditional mean given the noisy eigenvalues. Then the minimum achievable loss becomes:

$$\ell^\star_{\mathrm{SA}} := \ell(h^\star) = \mathbb{E}_{A,\,\mathcal{E}}\Big[\|B - \mathbb{E}[B \mid \widetilde{\Lambda}]\|_F^2\Big] + c \tag{52}$$

$$\Rightarrow \ell^\star_{\mathrm{SA}} = \mathbb{E}_{\widetilde{\Lambda}}\left[\underbrace{\mathbb{E}_{B|\widetilde{\Lambda}}\Big[\|B - \mathbb{E}[B \mid \widetilde{\Lambda}]\|_F^2\Big]}_{\mathrm{tr}\left(\mathrm{Cov}(\mathrm{vec}(B)|\widetilde{\Lambda})\right)}\right] + c \tag{53}$$

C.3 Proof of Theorem 1

By the law of total variance:

$$\mathrm{Cov}(\mathrm{vec}(B)) = \mathbb{E}_{\widetilde{\Lambda}}\left[\mathrm{Cov}(\mathrm{vec}(B) \mid \widetilde{\Lambda})\right] + \mathrm{Cov}(\mathbb{E}[\mathrm{vec}(B) \mid \widetilde{\Lambda}]) \tag{54}$$

Taking the trace of both sides, we have:

$$\ell_{\mathrm{A}}^{\star} = \ell_{\mathrm{SA}}^{\star} + \mathrm{tr}(\mathrm{Cov}(\mathbb{E}[\mathrm{vec}(B) \mid \widetilde{\Lambda}])) \tag{55}$$

Thus, the improvement is:

$$\ell_{\mathrm{A}}^{\star} - \ell_{\mathrm{SA}}^{\star} = \mathrm{tr}(\mathrm{Cov}(\mathbb{E}[\mathrm{vec}(B) \mid \widetilde{\Lambda}])) \tag{56}$$

which can also be written as:

$$\ell_{\mathrm{A}}^{\star} - \ell_{\mathrm{SA}}^{\star} = \mathbb{E}\left[\left\|\mathbb{E}[B \mid \widetilde{\Lambda}] - \mathbb{E}[B]\right\|_F^2\right] \tag{57}$$

C.4 Proof of Lemma 1

We start by setting

$$\sum_{l=0}^{K-1} \widetilde{\Lambda}^l H_Q^{(l)} = \mathbf{I}.$$

This choice is feasible, for instance by taking $H_Q^{(0)} = \mathbf{I}$ and $H_Q^{(l)} = \mathbf{0}$ for all $l \geq 1$. Next, choose each $H_K^{(l)}$ to be a constant diagonal matrix, i.e., $H_K^{(l)} = h_k^{(l)}\mathbf{I}$. Under these choices, Eq. (14) reduces to the spectral filter

$$f(U) = U\left(\sum_{l=0}^{L-1} h_k^{(l)} \Lambda^l\right)^{\top} U^{\top}.$$

Since Λ is diagonal, the matrix

$$\sum_{l=0}^{L-1} h_k^{(l)} \Lambda^l$$

acts independently on each eigenvalue. Equivalently, it implements the pointwise polynomial filter

$$w(\lambda) = \sum_{l=0}^{L-1} h_k^{(l)} \lambda^l.$$

It remains to show that such a polynomial filter can approximate η_{out}. Although η_{out} is not analytic at the thresholds $\lambda = \pm 2v$, it is continuous on any compact interval $[a, b] \subset \mathbb{R}$. Therefore, by the Weierstrass approximation theorem (Rudin, 1976), for every $\varepsilon > 0$ there exists a real polynomial P such that

$$\sup_{\lambda \in [a,b]} |P(\lambda) - \eta_{\mathrm{out}}(\lambda)| < \varepsilon.$$

Writing

$$P(\lambda) = \sum_{l=0}^{L-1} h_k^{(l)} \lambda^l$$

for sufficiently large L, we obtain a choice of coefficients $\{h_k^{(l)}\}_{l=0}^{L-1}$ such that the corresponding graph filter approximates η_{out} uniformly on $[a, b]$. This completes the proof.

C.5 Proof of Theorem 2

Proof. We first show that for a given graph A, $B := \widetilde{U}^\top A \widetilde{U}$ tends to a deterministic matrix as $n \to \infty$. We begin by writing B as:

$$B = \widetilde{U}^\top A \widetilde{U} = \widetilde{U}^\top U \Lambda U^\top \widetilde{U} \tag{58}$$

Consider that A is low rank therefore we can write:

$$A = \sum_{i=1}^{r} \lambda_i u_i u_i^\top \tag{59}$$

Thus we have:

$$B_{ij} = \sum_{l=1}^{r} \lambda_l (\widetilde{u}_i^\top u_l)(\widetilde{u}_j^\top u_l) \tag{60}$$

According to Lemma 3 we have:

$$\lim_{n\to\infty} B_{ii} = \lim_{n\to\infty} \sum_{l=1}^{r} \lambda_l |\widetilde{u}_i^\top u_l|^2 = \lambda_i \lim_{n\to\infty} |\widetilde{u}_i^\top u_i|^2 + \sum_{l=1, l\neq i}^{r} \lambda_l \lim_{n\to\infty} |\widetilde{u}_i^\top u_l|^2 \tag{61}$$

The sum over $l \neq i$ tends to zero almost surely, thus we have:

$$\lim_{n\to\infty} B_{ii} = \lambda_i \alpha(\lambda_i) = \lambda_i (1 - \frac{v^2}{\lambda_i^2}) = \lambda_i - \frac{v^2}{\lambda_i} \tag{62}$$

for the off-diagonal elements we have:

$$\lim_{n\to\infty} B_{ij} = \lim_{n\to\infty} \sum_{l=1}^{r} \lambda_l (\widetilde{u}_i^\top u_l)(\widetilde{u}_j^\top u_l) = 0, \tag{63}$$

where since $i \neq j$, one of the inner products will almost surely tend to zero regardless of the value of l.

So, in summary we have for a given fixed graph A, B tends to a deterministic matrix as $n \to \infty$:

$$\lim_{n\to\infty} B = \alpha(\Lambda) \tag{64}$$

Furthermore, from Lemma 3 we have the deterministic relationship between the noisy and clean eigenvalues:

$$\rho(\lambda_i) = \widetilde{\lambda}_i = \lambda_i + \frac{v^2}{\lambda_i} \tag{65}$$

Thus we have:

$$\lambda_i = \frac{\widetilde{\lambda}_i + \sqrt{\widetilde{\lambda}_i^2 - 4v^2}}{2} \tag{66}$$

Finally we can write:

$$\alpha(\lambda_i) = \alpha\left(\frac{\widetilde{\lambda}_i + \sqrt{\widetilde{\lambda}_i^2 - 4v^2}}{2}\right) = 1 - \frac{v^2}{\left(\frac{\widetilde{\lambda}_i + \sqrt{\widetilde{\lambda}_i^2 - 4v^2}}{2}\right)^2} = \sqrt{\widetilde{\lambda}_i^2 - 4v^2} \tag{67}$$

Finally we define the function $\eta_{out}(\cdot)$ as:

$$\eta_{out}(\widetilde{\lambda}) = \begin{cases} \sqrt{\widetilde{\lambda}^2 - 4v^2} & \text{if } \widetilde{\lambda} > 2v \\ 0 & \text{if } \widetilde{\lambda} \leq 2v \end{cases} \tag{68}$$

Thus in the limit almost surely we have:

$$\lim_{n\to\infty} B = \eta_{out}(\widetilde{\Lambda}) \tag{69}$$

□

The following is a restatement of Theorem 2.1 and Theorem 2.2 from (Benaych-Georges & Nadakuditi, 2010).

Lemma 3 (Leading-order overlap for spiked Wigner)**.** *Let W_n be an $n \times n$ real symmetric (or complex Hermitian) Wigner matrix with i.i.d. entries (up to symmetry) satisfying $\mathbb{E}[W_{ij}] = 0$ and $\mathbb{E}[W_{ij}^2] = v^2/n$ for $i \neq j$, with finite fourth moment. The empirical spectral distribution of W_n converges almost surely to the semicircle law on $[-2v, 2v]$, with edge $b = 2v$ and Stieltjes transform*

$$g(z) := \int \frac{1}{z-x}\, d\mu_{sc}(x) = \frac{z - \sqrt{z^2 - 4v^2}}{2v^2}, \qquad z \in \mathbb{R},\ |z| > 2v, \tag{70}$$

where the branch of the square root is chosen so that $g(z) \to 0$ as $z \to \infty$.

Let A_n be a deterministic (or random, independent of W_n) symmetric matrix of fixed rank r, with spectral decomposition

$$A_n = \sum_{i=1}^{r} \lambda_i\, u_i u_i^\top = U \Lambda U^\top, \tag{71}$$

where $\lambda_1 > \lambda_2 > \cdots > \lambda_r > 0$ are fixed (distinct) and $u_1, \ldots, u_r \in \mathbb{R}^n$ are orthonormal. (Negative spikes are handled symmetrically.)

Define the perturbed matrix $\widetilde{A}_n := W_n + A_n$, and let $\widetilde{\lambda}_1 \geq \widetilde{\lambda}_2 \geq \cdots \geq \widetilde{\lambda}_n$ be its eigenvalues with corresponding orthonormal eigenvectors $\widetilde{u}_1, \ldots, \widetilde{u}_n$.

Let $r_+ \in \{0, 1, \ldots, r\}$ denote the number of supercritical spikes, i.e., the number of indices i with $\lambda_i > v$. Then the following hold almost surely as $n \to \infty$:

(i) Outlier eigenvalue locations. *For each $i \in \{1, \ldots, r_+\}$,*

$$\widetilde{\lambda}_i \xrightarrow{a.s.} \rho(\lambda_i) := \lambda_i + \frac{v^2}{\lambda_i} = g^{-1}(1/\lambda_i), \tag{72}$$

and $\rho(\lambda_i) > 2v$. For $i > r_+$, $\widetilde{\lambda}_i \to 2v$.

(ii) Eigenvector overlap — diagonal. *For each $i \in \{1, \ldots, r_+\}$,*

$$|\langle \widetilde{u}_i, u_i \rangle|^2 \xrightarrow{a.s.} \alpha(\lambda_i) := \frac{-1}{\lambda_i^2\, g'(\rho(\lambda_i))} = 1 - \frac{v^2}{\lambda_i^2}. \tag{73}$$

(iii) Eigenvector overlap — cross terms. *For $i, j \in \{1, \ldots, r_+\}$ with $i \neq j$,*

$$\langle \widetilde{u}_i, u_j \rangle \xrightarrow{a.s.} 0. \tag{74}$$

C.6 Proof of Lemma 2

Proof. We work in the equal block case: $m_\ell = m$ for all ℓ, $n = km$, $\lambda_{\max} = m$. Write $\widetilde{U}_k = U_k R + U_\perp S$ where $U_\perp \in \mathbb{R}^{n \times (n-k)}$ collects the eigenvectors orthogonal to the principal eigenspace, $R \in \mathbb{R}^{k \times k}$, $S \in \mathbb{R}^{(n-k) \times k}$, and $R^\top R + S^\top S = I_k$ (orthonormality of $\widetilde{U}_k$). The matrix S measures the eigenvector leakage out of the true principal eigenspace.

We first characterize ℓ_{basis} and ℓ_{proj} in terms of S.

Characterization of ℓ_{basis}. Recall that

$$\ell_{\text{basis}} = \mathbb{E}_{\mathcal{E}} \left\| \widehat{A} - A \right\|_F^2. \tag{75}$$

In the equal block case $\Lambda_k = mI_k$, so

$$\widehat{A} - A = m(\widetilde{U}_k \widetilde{U}_k^\top - U_k U_k^\top) = m(\widetilde{P} - P_U) \tag{76}$$

where $\widetilde{P} = \widetilde{U}_k \widetilde{U}_k^\top$ is the projector onto the noisy principal eigenspace. Substituting $\widetilde{U}_k = U_k R + U_\perp S$ and using $U_k^\top U_k = I_k$, $U_k^\top U_\perp = 0$, $U_\perp^\top U_\perp = I_{n-k}$, we expand

$$\widetilde{U}_k \widetilde{U}_k^\top = U_k R R^\top U_k^\top + U_k R S^\top U_\perp^\top + U_\perp S R^\top U_k^\top + U_\perp S S^\top U_\perp^\top, \tag{77}$$

and therefore

$$\widetilde{P} - P_U = U_k(RR^\top - I_k)U_k^\top + U_k R S^\top U_\perp^\top + U_\perp S R^\top U_k^\top + U_\perp S S^\top U_\perp^\top. \tag{78}$$

The four terms in (78) are mutually Frobenius-orthogonal, since their row and column spaces lie in either $\mathrm{range}(U_k)$ or $\mathrm{range}(U_\perp)$, which are orthogonal. Hence

$$\|\widetilde{P} - P_U\|_F^2 = \|RR^\top - I_k\|_F^2 + 2\|RS^\top\|_F^2 + \|SS^\top\|_F^2. \tag{79}$$

Using $R^\top R = I_k - S^\top S$ and the cyclic property of the trace,

$$\|RR^\top - I_k\|_F^2 = \mathrm{tr}((R^\top R)^2) - 2\,\mathrm{tr}(R^\top R) + k = \|S^\top S\|_F^2, \tag{80}$$

$$\|RS^\top\|_F^2 = \mathrm{tr}(R^\top R\, S^\top S) = \mathrm{tr}((I_k - S^\top S)\, S^\top S) = \|S\|_F^2 - \|S^\top S\|_F^2, \tag{81}$$

Substituting into (79) gives

$$\|\widetilde{P} - P_U\|_F^2 = 2\|S\|_F^2. \tag{82}$$

Taking expectations,

$$\ell_{\text{basis}} = m^2\,\mathbb{E}\left[\|\widetilde{P} - P_U\|_F^2\right] = 2m^2\,\mathbb{E}\left[\|S\|_F^2\right] \tag{83}$$

Characterization of ℓ_{proj}. The projected estimator is $P_U\widehat{A}P_U^\top$ with $P_U = U_kU_k^\top$. Since $\widehat{A} = m\,\widetilde{U}_k\widetilde{U}_k^\top$ and $U_k^\top\widetilde{U}_k = R$,

$$P_U\widehat{A}P_U^\top = m\,U_k(U_k^\top\widetilde{U}_k)(\widetilde{U}_k^\top U_k)U_k^\top = m\,U_kRR^\top U_k^\top. \tag{84}$$

The clean matrix is $A = m\,U_kU_k^\top = U_k(mI_k)U_k^\top$, so

$$P_U\widehat{A}P_U^\top - A = m\,U_k(RR^\top - I_k)U_k^\top. \tag{85}$$

Since U_k has orthonormal columns,

$$\|P_U\widehat{A}P_U^\top - A\|_F^2 = m^2\,\|RR^\top - I_k\|_F^2 = m^2\,\|S^\top S\|_F^2, \tag{86}$$

where the last equality is (80). Taking expectations,

$$\ell_{\text{proj}} = m^2\,\mathbb{E}\left[\|S^\top S\|_F^2\right]. \tag{87}$$

The Improvement. Combining (83) and (87),

$$\Delta\ell_{\text{proj}} = \ell_{\text{basis}} - \ell_{\text{proj}} = m^2\,\mathbb{E}\big[2\|S\|_F^2 - \|S^\top S\|_F^2\big]. \tag{88}$$

Since $R^\top R + S^\top S = I_k$ with $R^\top R \succeq 0$, we have $S^\top S \preceq I_k$, so all eigenvalues of $S^\top S$ lie in $[0,1]$. Letting $s_1, \ldots, s_k \in [0,1]$ denote these eigenvalues,

$$2\|S\|_F^2 - \|S^\top S\|_F^2 = \sum_{i=1}^k s_i(2 - s_i) \geq \sum_{i=1}^k s_i = \|S\|_F^2, \tag{89}$$

and thus we have

$$\Delta\ell_{\text{proj}} \geq m^2\,\mathbb{E}\left[\|S\|_F^2\right] > 0. \tag{90}$$

Therefore, projecting onto the true principal eigenspace strictly decreases the denoising loss which concludes the proof. □

C.7 Proof of Theorem 3

Proof. Let $E(\alpha) = \sigma(\alpha\widehat{A}) - P_U$ and $\lambda_{\max} = \max_i m_i = \|\widehat{A}\|_2$. Then we have:

$$\begin{aligned} f(\widehat{A};\alpha) - A &= (P_U + E)\,\widehat{A}\,(P_U + E)^\top - A \\ &= \underbrace{(P_U\widehat{A}P_U^\top - A)}_{\delta_0} + \underbrace{E\widehat{A}P_U^\top + P_U\widehat{A}E^\top + E\widehat{A}E^\top}_{\delta_1} \end{aligned} \tag{91}$$

where we used $P_U A P_U^\top = U_k U_k^\top (U_k \Lambda_k U_k^\top) U_k U_k^\top = U_k \Lambda_k U_k^\top = A$, so that $P_U \widehat{A} P_U^\top - A = P_U(\widehat{A} - A)P_U^\top = \delta_0$. Expanding the squared Frobenius norm:

$$\|f(\widehat{A};\alpha) - A\|_F^2 = \|\delta_0\|_F^2 + 2\langle \delta_0,\, \delta_1\rangle_F + \|\delta_1\|_F^2 \tag{92}$$

Taking expectations and noting $\mathbb{E}[\|\delta_0\|_F^2] = \ell_{\text{proj}}$ gives (93).

$$\ell(\alpha) = \ell_{\text{proj}} + 2\,\mathbb{E}[\langle \delta_0,\, \delta_1\rangle_F] + \mathbb{E}\big[\|\delta_1\|_F^2\big] \tag{93}$$

Applying Cauchy–Schwarz to the cross term:

$$|\mathbb{E}[\langle \delta_0, \delta_1\rangle_F]| \le \mathbb{E}[\|\delta_0\|_F \|\delta_1\|_F] \le \sqrt{\mathbb{E}[\|\delta_0\|_F^2]}\,\sqrt{\mathbb{E}[\|\delta_1\|_F^2]} = \sqrt{\ell_{\text{proj}}}\,\sqrt{\mathbb{E}[\|\delta_1\|_F^2]} \tag{94}$$

Thus we have:

$$\ell(\alpha) \le \ell_{\text{proj}} + 2\sqrt{\ell_{\text{proj}}}\,\sqrt{\mathbb{E}[\|\delta_1\|_F^2]} + \mathbb{E}\big[\|\delta_1\|_F^2\big] \tag{95}$$

It remains to bound $\mathbb{E}[\|\delta_1\|_F^2]$ and show that:

$$2\sqrt{\ell_{\text{proj}}}\,\sqrt{\mathbb{E}[\|\delta_1\|_F^2]} + \mathbb{E}\big[\|\delta_1\|_F^2\big] \;<\; \ell_{\text{basis}} - \ell_{\text{proj}} \tag{96}$$

Bounding $\mathbb{E}[\|\delta_1\|_F^2]$.

Since $\widehat{A} = \widetilde{U}_k \Lambda_k \widetilde{U}_k^\top$ and $P_U = U_k U_k^\top$, denoting $\Omega = \widetilde{U}_k^\top U_k \in \mathbb{R}^{k\times k}$ (which satisfies $\|\Omega\|_2 \le 1$), every term in δ_1 factors through $E\widetilde{U}_k$:

$$E\widehat{A}P_U^\top = (E\widetilde{U}_k)\,\Lambda_k\,\Omega\,U_k^\top, \tag{97}$$

$$P_U\widehat{A}E^\top = U_k\,\Omega^\top\Lambda_k\,(E\widetilde{U}_k)^\top, \tag{98}$$

$$E\widehat{A}E^\top = (E\widetilde{U}_k)\,\Lambda_k\,(E\widetilde{U}_k)^\top. \tag{99}$$

Therefore, by sub-multiplicativity:

$$\|\delta_1\|_F \le \lambda_{\max}\Big(2\|E\widetilde{U}_k\|_F + \|E\widetilde{U}_k\|_F^2\Big) \tag{100}$$

Now we turn to bounding $\|E\widetilde{U}_k\|_F$.

Bounding $\|E\widetilde{U}_k\|_F$. Write $\widetilde{U}_k = U_k + \Delta U_k$. Then we have:

$$\|E\widetilde{U}_k\|_F = \|E(U_k + \Delta U_k)\|_F = \|EU_k\|_F + \|E\Delta U_k\|_F \tag{101}$$

The Following Lemma allows us to bound $\|E\Delta U_k\|_F$:

Lemma 4 (Bound on EU_k)**.** *Define $\eta = \|\widehat{A} - A\|_\infty := \max_{i,j} |\widehat{A}_{ij} - A_{ij}|$ and $\gamma = 1 - 2\eta$. Assume $\gamma > 0$ (i.e., the noise is small enough that in-block and cross-block entries of $\widehat{A}$ are separated). Then*

$$\|EU_k\|_F^2 \;\le\; e^{-2\alpha\gamma} \sum_{\ell=1}^{k} \left(\frac{(n - m_\ell)^2}{m_\ell^2} + m_\ell \sum_{\ell' \neq \ell} \frac{1}{m_{\ell'}} \right). \tag{102}$$

In the equal block case ($m_\ell = m$, $n = km$), this simplifies to

$$\|EU_k\|_F \;\le\; k\sqrt{k-1}\, e^{-\alpha\gamma}. \tag{103}$$

Using Lemma 4 we can write in the equal block case:

$$\|E\widetilde{U}_k\|_F \le k\sqrt{k-1}\, e^{-\alpha\gamma} + \|E\Delta U_k\|_F \tag{104}$$

Now we will write the LHS and RHS of the inequality (96) up to leading order in $\|S\|_F$.

RHS of (96) **to leading order.** From Lemma 2 we recall,

$$\ell_{\text{basis}} = 2m^2\mathbb{E}\left[\|S\|_F^2\right], \qquad \ell_{\text{proj}} = m^2\,\mathbb{E}\left[\|S^\top S\|_F^2\right]. \tag{105}$$

Since the eigenvalues of $S^\top S$ lie in $[0,1]$, $\|S^\top S\|_F^2 \le \|S\|_F^4$. Hence, to leading order in $\|S\|_F$,

$$\ell_{\text{basis}} - \ell_{\text{proj}} = 2m^2\,\mathbb{E}\left[\|S\|_F^2\right] + O\big(\mathbb{E}\|S\|_F^4\big)\,. \tag{106}$$

LHS of (96) **to leading order.** We first bound $\|E\widetilde{U}_k\|_F$. Decompose $E = E_1 + E_2$ with

$$E_1 = \sigma(\alpha A) - P_U, \qquad E_2 = \sigma(\alpha \widehat{A}) - \sigma(\alpha A). \tag{107}$$

E_1 is deterministic (independent of noise) and captures the finite-α bias of the softmax; a direct calculation shows $\|E_1\|_F = O(\sqrt{n}\,e^{-\alpha})$, decaying exponentially in α. The noise term E_2 is controlled by the following Lemma (See Section C.7.2 for proof).

Lemma 5 (Linearized softmax bound)**.** *In the equal-block case, to leading order in* $\|S\|_F$*,*

$$\|E_2\|_F \;\le\; \sqrt{2}\,\alpha\,\|S\|_F + O(\|S\|_F^2). \tag{108}$$

Since $\|E_1\|_F$ decays exponentially in α (and is dominated by $\|E_2\|_F$ for $\alpha\gamma \gg \log k$ and $\|S\|_F > 0$),

$$\|E\|_F \le \|E_1\|_F + \|E_2\|_F = \sqrt{2}\,\alpha\,\|S\|_F + O(\|S\|_F^2). \tag{109}$$

Furthermore, $\|\Delta U_k\|_F = \|S\|_F + O(\|S\|_F^2)$ to leading order (the within-eigenspace component $\|R - I_k\|_F$ is $O(\|S\|_F^2)$). Substituting into (104),

$$\|E\widetilde{U}_k\|_F \le k\sqrt{k-1}\,e^{-\alpha\gamma} + \sqrt{2}\,\alpha\|S\|_F^2 + o(\|S\|_F^2). \tag{110}$$

Assume $\alpha\gamma \gg \log k$ so the exponential term is negligible; then

$$\|E\widetilde{U}_k\|_F = \sqrt{2}\,\alpha\|S\|_F^2 + o(\|S\|_F^2), \tag{111}$$

and the quadratic term $\|E\widetilde{U}_k\|_F^2 = O(\|S\|_F^4)$ is higher order. Plugging into (100) with $\lambda_{\max} = m$,

$$\|\delta_1\|_F \le m\Big(2\|E\widetilde{U}_k\|_F + \|E\widetilde{U}_k\|_F^2\Big) = 2\sqrt{2}\,m\alpha\|S\|_F^2 + o(\|S\|_F^2), \tag{112}$$

so

$$\mathbb{E}\|\delta_1\|_F^2 \le 8m^2\alpha^2\,\mathbb{E}\|S\|_F^4 + o\big(\mathbb{E}\|S\|_F^4\big)\,. \tag{113}$$

Since $\sqrt{\ell_{\text{proj}}} \le m\sqrt{\mathbb{E}\|S\|_F^4}$ (using $\|S^\top S\|_F^2 \le \|S\|_F^4$), the cross term satisfies

$$2\sqrt{\ell_{\text{proj}}}\sqrt{\mathbb{E}\|\delta_1\|_F^2} \le 4\sqrt{2}\,m^2\alpha\,\mathbb{E}\|S\|_F^4 + o\big(\mathbb{E}\|S\|_F^4\big)\,. \tag{114}$$

Combining,

$$\text{LHS of (96)} \;\le\; m^2\alpha\Big(4\sqrt{2} + 8\alpha\Big)\;\mathbb{E}\|S\|_F^4 + o\big(\mathbb{E}\|S\|_F^4\big)\,. \tag{115}$$

Resulting condition. Combining (106) and (115), the bound (96) holds to leading order whenever

$$\alpha\Big(4\sqrt{2} + 8\alpha\Big)\,\frac{\mathbb{E}\|S\|_F^4}{\mathbb{E}\left[\|S\|_F^2\right]} \;<\; 2. \tag{116}$$

Denote $\beta := 2\dfrac{\mathbb{E}\left[\|S\|_F^2\right]}{\mathbb{E}\|S\|_F^4}$. Then the condition (116) becomes

$$\alpha\Big(4\sqrt{2} + 8\alpha\Big) \;<\; \beta. \tag{117}$$

Solving the quadratic $8\alpha^2 + 4\sqrt{2}\,\alpha - \beta = 0$, the condition holds for $\alpha \in (0, \alpha_+)$ where

$$\alpha_+ \;=\; \frac{\sqrt{2}}{4}\Big(-1 + \sqrt{1+\beta}\Big)\,. \tag{118}$$

Compatibility of the two conditions on α. The proof so far rests on two conditions on the parameter α:

- **(A)** $\alpha \in (0, \alpha_+)$, so that the LHS leading-order bound (115) is dominated by the RHS leading-order bound (106) (this is the bound (117)).
- **(B)** $\alpha\gamma \gg \log k$, so that the exponential term $k\sqrt{k-1}\, e^{-\alpha\gamma}$ in (104) is negligible compared to the noise term.

These conditions pull in opposite directions: (A) caps α from above, while (B) requires α to grow at least like $\log k$. To show that they can be met simultaneously, we make the following assumption on the noise.

Assumption 2. *There exists s such that $\|S\|_F \le s$ almost surely, with*

$$s \;\ll\; \min\!\left(\frac{1}{m},\; \frac{1}{\log k}\right). \tag{119}$$

Note that $\eta = \|\widehat{A} - A\|_\infty$ is not an independent quantity: by the entry-wise bound $\|M\|_\infty \le \|M\|_F$ together with $\|\widehat{A} - A\|_F = m\sqrt{2}\|S\|_F$ from (83),

$$\eta \;\le\; \|\widehat{A} - A\|_F \;=\; m\sqrt{2}\,\|S\|_F \;\le\; m\sqrt{2}\,s. \tag{120}$$

Therefore Assumption 2 caps both $\|S\|_F$ *and* η. The two bounds in (119) will turn out to control γ (via $\eta \lesssim ms$) and the exponential term (via $\alpha\gamma \gtrsim 1/s$), making (A) and (B) compatible.

Lower bound on α_+. Since $\|S\|_F^2 \le s^2$ a.s., we have the pointwise bound $\|S\|_F^4 \le s^2\,\|S\|_F^2$. Taking expectations,

$$\mathbb{E}\|S\|_F^4 \;\le\; s^2\,\mathbb{E}\|S\|_F^2 \quad\Longrightarrow\quad \beta \;=\; 2\,\frac{\mathbb{E}\|S\|_F^2}{\mathbb{E}\|S\|_F^4} \;\ge\; \frac{2}{s^2}. \tag{121}$$

Since $= \alpha_+ = \frac{\sqrt{2}}{4}(\sqrt{1+\beta} - 1)$ is monotone increasing in β we can write:

$$\alpha_+ \;\ge\; \tfrac{\sqrt{2}}{4}\Big(\sqrt{1 + 2/s^2} - 1\Big) \;\ge\; \tfrac{\sqrt{2}}{4}\Big(\tfrac{\sqrt{2}}{s} - 1\Big) \;=\; \tfrac{1}{2s} - \tfrac{\sqrt{2}}{4}, \tag{122}$$

where we used $\sqrt{1 + 2/s^2} \ge \sqrt{2}/s$. Thus the cap α_+ is at least of order $1/s$, which is large under Assumption 2.

Lower bound on γ. Combining $\gamma = 1 - 2\eta$ with (120),

$$\gamma \;\ge\; 1 - 2m\sqrt{2}\,s. \tag{123}$$

The condition $ms \ll 1$ from Assumption 2 ensures $2m\sqrt{2}\,s \le 1/2$ for s small enough, hence $\gamma \ge 1/2$.

Choosing $\alpha^\star$ to satisfy both (A) and (B). Set $\alpha^\star := \dfrac{1}{4s}$. We verify each condition:

- **Condition (A):** $\alpha^\star < \alpha_+$. By (122) this reduces to $\frac{1}{4s} < \frac{1}{2s} - \frac{\sqrt{2}}{4}$, i.e., $s < \frac{1}{\sqrt{2}}$, which is automatic since $s \ll 1/\log k$.
- **Condition (B):** $\alpha^\star\gamma \gg \log k$. By (123), once $ms \ll 1$ so that $\gamma \ge 1/2$ we have,
$$\alpha^\star\gamma \;\ge\; \tfrac{1}{4s}\cdot\tfrac{1}{2} \;=\; \tfrac{1}{8s} \;\gg\; \log k, \tag{124}$$
where the last step uses $s \ll 1/\log k$.

This is precisely why both parts of (119) are needed: $ms \ll 1$ guarantees γ stays bounded away from 0, and $s \log k \ll 1$ then makes $\alpha^\star\gamma$ large compared to $\log k$. Hence $\alpha^\star$ lies in $(0, \alpha_+)$ *and* makes the exponential term in (104) negligible, so all the leading-order bounds derived in the proof apply at $\alpha = \alpha^\star$.

Conclusion. At $\alpha = \alpha^\star$, the condition (117) holds strictly, which by (115) and (106) is equivalent to the gap bound (96). Therefore

$$\epsilon(\alpha^\star) \;=\; \ell_{\text{soft}}(\alpha^\star) - \ell_{\text{proj}} \;<\; \ell_{\text{basis}} - \ell_{\text{proj}} \;=\; \Delta\ell_{\text{proj}}, \tag{125}$$

proving the theorem.

Translation to the noise scale. We close by translating Assumption 2 into a condition on the adjacency-level noise $\mathcal{E} := \widetilde{A} - A$. In the equal-block case the spectral gap of A between the k-th principal eigenvalue and the trivial part of the spectrum equals $\Delta = m$. By the Davis–Kahan theorem (in $\sin\Theta$ form),

$$\|S\|_F \;\le\; \frac{\|P_\perp\, \mathcal{E}\, V_k\|_F}{\Delta} \;\le\; \frac{\|\mathcal{E}\|_F}{m}. \tag{126}$$

Squaring and taking expectations,

$$\sigma^2 \;=\; \mathbb{E}\|S\|_F^2 \;\le\; \frac{\mathbb{E}\|\mathcal{E}\|_F^2}{m^2}. \tag{127}$$

Translating the two parts of (119) (with s^2 scaling like $\mathbb{E}\|\mathcal{E}\|_F^2 := \sigma^2$, assuming concentration of $\|S\|_F^2$ around its mean):

- $s \ll 1/m$ becomes $\sigma^2 \ll 1/m^2$, i.e., $\mathbb{E}\|\mathcal{E}\|_F^2 \ll 1$.
- $s \ll 1/\log k$ becomes $\sigma^2 \ll 1/\log^2 k$, i.e., $\mathbb{E}\|\mathcal{E}\|_F^2 \ll m^2/\log^2 k$.

Combining, Assumption 2 holds provided

$$\mathbb{E}\|\mathcal{E}\|_F^2 \;\ll\; \min\!\left(1,\; \frac{m^2}{\log^2 k}\right). \tag{128}$$

which is the assumption stated in the theorem.

For independent edge flips with probability p, $\mathbb{E}[\mathcal{E}_{ij}^2] = p(1-p)$ for each entry, so $\mathbb{E}\|\mathcal{E}\|_F^2 = n(n-1)\,p(1-p) \approx n^2 p$. Substituting and using $n = km$,

$$p \;\ll\; \min\!\left(\frac{1}{n^2},\; \frac{1}{k^2 \log^2 k}\right). \tag{129}$$

□

C.7.1 Proof of Lemma 4

Proof. We bound $(Eu_\ell)_i$ separately for i in-block ($i \in B_\ell$) and cross-block ($i \in B_{\ell'}$, $\ell' \neq \ell$). Write $S = \sigma(\alpha\widehat{A}) = P_U + E$ and $Z_i = \sum_{j'=1}^n e^{\alpha \widehat{A}_{ij'}}$ for the softmax normalizer.

Case 1: $i \in B_\ell$**.** Since $\sum_j E_{ij} = 0$ and $(P_U)_{ij} = 0$ for $j \notin B_\ell$:

$$\sum_{j \in B_\ell} E_{ij} = -\sum_{j \notin B_\ell} E_{ij} = -\sum_{j \notin B_\ell} S_{ij}. \tag{130}$$

For $j \notin B_\ell$: $\widehat{A}_{ij} \le \eta$ (cross-block), so $e^{\alpha \widehat{A}_{ij}} \le e^{\alpha\eta}$. For $j' \in B_\ell$: $\widehat{A}_{ij'} \ge 1 - \eta$ (same-block), so $Z_i \ge m_\ell\, e^{\alpha(1-\eta)}$. Therefore

$$\left|\sum_{j \in B_\ell} E_{ij}\right| \le \frac{(n - m_\ell)\, e^{\alpha\eta}}{m_\ell\, e^{\alpha(1-\eta)}} = \frac{n - m_\ell}{m_\ell}\, e^{-\alpha\gamma} \tag{131}$$

and $|(Ev_\ell)_i| \le \frac{n - m_\ell}{m_\ell^{3/2}}\, e^{-\alpha\gamma}$.

Case 2: $i \in B_{\ell'}$, $\ell' \neq \ell$**.** Now $(P_U)_{ij} = 0$ for all $j \in B_\ell$, so $\sum_{j \in B_\ell} E_{ij} = \sum_{j \in B_\ell} S_{ij}$. For $j \in B_\ell$: $\widehat{A}_{ij} \le \eta$ (cross-block relative to i). For $j' \in B_{\ell'}$: $Z_i \ge m_{\ell'}\, e^{\alpha(1-\eta)}$. Thus

$$\sum_{j \in B_\ell} S_{ij} \le \frac{m_\ell\, e^{\alpha\eta}}{m_{\ell'}\, e^{\alpha(1-\eta)}} = \frac{m_\ell}{m_{\ell'}}\, e^{-\alpha\gamma} \tag{132}$$

and $|(Ev_\ell)_i| \le \frac{\sqrt{m_\ell}}{m_{\ell'}}\, e^{-\alpha\gamma}$.

Combining. We have

$$\|Ev_\ell\|^2 = \sum_{i\in B_\ell}(Ev_\ell)_i^2 + \sum_{\ell'\neq\ell}\sum_{i\in B_{\ell'}}(Ev_\ell)_i^2 \le e^{-2\alpha\gamma}\left(\frac{(n-m_\ell)^2}{m_\ell^2} + m_\ell\sum_{\ell'\neq\ell}\frac{1}{m_{\ell'}}\right). \tag{133}$$

Summing over $\ell = 1,\dots,k$ gives (102). Setting $m_\ell = m$ and $n = km$ yields (103). □

C.7.2 Proof of Lemma 5

Proof. Let $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ denote the row-wise softmax with Jacobian $J(x) = \mathrm{diag}(\sigma(x)) - \sigma(x)\sigma(x)^\top$. For row $i \in B_\ell$ of A, $\sigma(\alpha A_{i,:}) = a\,\mathbf{1}_{B_\ell} + b\,\mathbf{1}_{B_\ell^c}$ with $a = 1/(m + (n-m)e^{-\alpha})$ and $b = a\,e^{-\alpha}$. The eigenvalues of $J(\alpha A_{i,:})$ are 0 (on $\mathbf{1}$), a (multiplicity $m-1$, on sum-zero perturbations within B_ℓ), b (multiplicity $n-m-1$, on sum-zero perturbations within B_ℓ^c), and nab (on $\mathrm{span}\{(n-m)\mathbf{1}_{B_\ell} - m\,\mathbf{1}_{B_\ell^c}\}$). For $\alpha \ge 0$, $a \ge b$ and $a \ge nab$, so

$$\|J(\alpha A_{i,:})\|_2 \;=\; a \;\le\; \tfrac{1}{m}. \tag{134}$$

Taylor-expanding σ around $\alpha A_{i,:}$ row-wise (the second-order remainder is controlled by the uniformly bounded Hessian of softmax), squaring, and summing over rows,

$$\|E_2\|_F^2 \;\le\; \alpha^2 \max_i \|J(\alpha A_{i,:})\|_2^2 \cdot \|\widehat{A}-A\|_F^2 + O(\|\widehat{A}-A\|_F^4) \;\le\; \tfrac{\alpha^2}{m^2}\,\|\widehat{A}-A\|_F^2 + O(\|S\|_F^4). \tag{135}$$

The claim follows from $\|\widehat{A} - A\|_F = m\sqrt{2}\|S\|_F$ in (83). □

C.8 Edge-flipping noise

For Digress noise, we consider the edge-flipping model where $\widetilde{A}_{ij} = A_{ij}$ with prob. $1-\varepsilon$ and $\widetilde{A}_{ij} = 1 - A_{ij}$ with prob. ε, so that $\widetilde{A} = A + \mathcal{E}_1$ with $\mathcal{E}_1 = \widetilde{A} - A$. A direct computation gives $\mathbb{E}[\widetilde{A} \mid A] = (1-2\varepsilon)A + \varepsilon\mathbf{1}\mathbf{1}^\top$, hence

$$\widetilde{A} = (1-2\varepsilon)A + \varepsilon\mathbf{1}\mathbf{1}^\top + \Xi, \tag{136}$$

where $\Xi := \widetilde{A} - \mathbb{E}[\widetilde{A} \mid A]$ has zero mean independent entries and is thus a Wigner-type matrix.

D Related Work

D.1 Optimization-based graph denoising

A large body of work formulates graph denoising as a model-based inverse problem in which the clean graph is obtained as the minimizer of a cost regularized by hand-crafted structural priors, with no learned mapping involved. From a network-statistics angle, (Feizi et al., 2013) invert in closed form a convolutional model of direct and indirect edge effects, while (Wu et al., 2017; Josephs et al., 2021) recover a clean adjacency under Type-I/II edge errors, low-rank parameter matrices, or unlabeled noisy samples, respectively. Within the graph signal processing (GSP) framework, the topology is denoised jointly with a process defined on it: (Segarra et al., 2017) recover a graph shift operator from possibly corrupted spectral templates, and (Rey et al., 2021; 2023; Tenorio et al., 2024) progressively extend this idea to robust filter identification and blind deconvolution under joint perturbations of inputs, filter, and topology. A third and particularly influential family exploits the low-rank, sparse, and smooth nature of real-world graphs, with Pro-GNN (Jin et al., 2020) as the canonical formulation and a number of follow-ups (Xu et al., 2021; Zhou et al., 2024; Wang et al., 2024) that improve scalability and avoid over-smoothing through truncated-SVD reparametrizations, framelet regularizers, or feature-driven matrix factorization. Related approaches obtain the clean graph from low-rank motifs (Lyu et al., 2024) or via iterative Fourier thresholding of the adjacency (Frost, 2026); in the same spirit, (Tenorio et al., 2023) extends the Pro-GNN-style joint optimization by replacing its low-rank and sparsity priors with a GSP-style denoising regularizer that is alternated with gradient updates of the downstream GNN.

D.2 Graph denoising via learned estimators

A complementary family of methods abandons the optimization viewpoint altogether and instead trains a parametric mapping from a noisy graph to a clean one, so that denoising amounts to a forward pass through a neural network rather than the solution of an instance-specific inverse problem. The first group of such methods uses the learned denoiser as a component of, or a preprocessing for, a downstream GNN. GADPN (Deng & Liu, 2026) performs the denoising step itself with a non-learned rank-adaptive SVD whose rank is chosen by Bayesian optimization, and feeds the cleaned adjacency to a standard GNN classifier. UGD (Yang et al., 2024) couples a non-parametric high-order proximity score for edge pruning with a GCN-based graph autoencoder that reconstructs corrupted features, and alternates between the two. RS-GNN (Dai et al., 2022) parametrizes the denoiser as an MLP edge predictor that re-weights the adjacency from pairs of node features and is trained jointly with a downstream GCN, while SLAPS (Fatemi et al., 2021) pairs an MLP-kNN graph generator with a separate GNN denoising autoencoder that recovers masked features through the learned structure. A second group places the parametric denoiser at the core of a denoising diffusion or flow-matching model, where a forward process gradually corrupts a clean graph and the learned denoiser is applied iteratively to invert it. Mask-GVAE (Li et al., 2021) is the earliest of these, with a partition-based GCN variational autoencoder that performs blind denoising without paired supervision. GDSS (Jo et al., 2022) parametrizes the score of a system of SDEs over node features and adjacency with a graph convolution / graph transformer backbone, while DiGress (Vignac et al., 2022) introduces a discrete diffusion in which a graph transformer is trained to predict the clean graph from a noisy input at every step; GruM (Jo et al., 2023), DeFoG (Qin et al., 2024), and DiPhon Rozada et al. (2026) reuse essentially the same graph transformer architecture but replace the noise-prediction objective by, respectively, endpoint prediction in a diffusion mixture and discrete flow matching. NetDiff (Marcoccia et al., 2024) augments the DiGress-style graph transformer with cross-attentive modulation tokens to enforce global topological constraints in ad-hoc network generation, DiffSP (Luo et al., 2025) repurposes such graph diffusion models as adversarial structure purifiers on top of a GCN classifier, and SGDM (Trivedi et al., 2023) operates at the subgraph level and is in fact backbone-agnostic, instantiated on top of DiGress, GDSS or EDGE.

D.3 Spectral graph transformers

A line of research orthogonal to standard message-passing GNNs builds graph neural architectures by routing the spectrum of the graph Laplacian through Transformer-style modules. The earliest works inject the spectrum as input features. SpGAT (He et al., 2020) first constructs spectral node features from the graph Laplacian eigenvectors and eigenvalues with an MLP on top. These features are then used as input for a GCN, which performs node-wise classification by assigning a label to each surface vertex. SAN (Kreuzer et al., 2021) constructs spectral positional features from the graph Laplacian spectrum and adds them to the node features. These enriched features are then processed by a fully connected Transformer. The Graph Transformer of (Dwivedi & Bresson, 2021) is an earlier instance of this same template, using truncated Laplacian eigenvectors directly as a positional encoding for a sparse-attention Transformer. This idea of using spectral information as feature is also investigated in (Pengmei & Li, 2024) by processing Laplacian eigenvalues with an MLP and usesing the result to initialize the Transformer's global `[CLS]` token, with the resulting architecture still being a graph Transformer. A second sub-line removes the eigenvector ambiguities (sign flips, basis rotations within an eigenspace) before they reach the Transformer. SignNet and BasisNet (Lim et al., 2022) are parametric encoders that produce sign- and basis-invariant embeddings of the eigenvectors, and are routinely plugged in as the positional-encoding module of a downstream graph Transformer. GIST (Rigotti et al., 2026) addresses the same problem from a different angle: it builds random spectral features by feeding Gaussian noise through a precomputed graph filter, and then uses the inner product matrix of the resulting features as a reweighting mechanism. Specifically, the architecture is a Transformer whose attention scores are reweighted by this spectral inner product matrix. A third sub-line uses the Transformer not as a pose-processing module of the spectrum but as the parametrization of a graph filter itself. Specformer (Bo et al., 2023) learns a graph filter by feeding the graph eigenvalues into a Transformer, which produces frequency responses, and then converts those responses back into a graph operator using the eigenvectors. This learned operator is then used as the propagation layer inside a GNN, so the rest of the model behaves like a graph neural network for prediction. PolyFormer (Ma et al., 2024) avoids eigendecomposition altogether by representing each node with a sequence of polynomial-basis tokens of the Laplacian, and then learning a node-wise

filter through self-attention over those tokens. Eigenformer (Garg, 2024) pushes this idea further by dispensing with explicit positional encodings and instead designing a spectrum-aware attention mechanism whose biases depend directly on the Laplacian spectrum, so that the Transformer block itself acts as a learnable spectral operator.