TRAIN SMALL, DEPLOY LARGE: ZERO-SHOT GNN TRANSFER ACROSS GRAPH SCALES

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) can operate on graphs of different sizes, yet weights learned at one resolution do not necessarily remain effective at another. We investigate whether training on a smaller replica allows zero-shot deployment on the full-resolution graph. We construct these replicas using geometric renormalization (GR), a previously proposed scale transformation designed to preserve latent geometry and self-similar multiscale organization. Across synthetic networks and eight real-world node-classification datasets, GCN, GraphSAGE, and GAT models trained on graphs compressed by factors up to 32 often retain much of the full-resolution accuracy while reducing training cost. Matched-compression randomization controls show that this effect cannot be explained by graph-size reduction alone: randomizing topology, latent geometry, or both breaks the relation between scales and substantially weakens transfer. CKA and orthogonal Procrustes analyses reveal cross-scale alignment of learned representations, while Jensen–Shannon comparisons show similar predictive trajectories during training. Comparisons with established coarsening methods show that GR remains competitive in transfer accuracy while generally preserving measured structural properties more faithfully. Together, these results identify self-similarity as a structural basis for cross-scale GNN transfer and position GR as a reusable training surrogate and controlled framework for studying cross-scale learning in graphs.

1 INTRODUCTION

Message-passing GNNs share local propagation and aggregation rules across nodes, allowing the same architecture to operate on graphs of different sizes Kipf & Welling (2017); Hamilton et al. (2017). This architectural flexibility does not ensure that weights learned at one resolution remain effective at another as neighborhood statistics and the relations among topology, features, and labels may change. We study zero-shot transfer between two resolutions of a graph. Given an original graph G_0 and a smaller replica G_ℓ obtained using the previously proposed geometric renormalization (GR) transformation García-Pérez et al. (2018), we train on G_ℓ and deploy the learned weights on G_0 without retraining. The question is whether G_ℓ is less expensive yet structurally faithful enough to support effective transfer across scales.

Existing work on GNN transferability mainly concerns generalization across distinct graph instances. Graphon analyses establish transferability under assumptions on graph convergence and filters Ruiz et al. (2020), whereas size-generalization results show that local structural changes can prevent small-to-large transfer Yehudai et al. (2021). Neither line determines how to construct a smaller training representation of a fixed graph. Graph coarsening addresses this setting more directly: Huang et al. Huang et al. (2021) train GCN and APPNP models on a coarsened graph and transfer their parameters to the original graph. Many methods preserve Laplacian properties, such as spectral or cut structure Loukas (2019); Chen et al. (2022), but these do not directly guarantee preservation of standard message passing. Joly and Keriven Joly & Keriven (2024) address this mismatch with a propagation operator designed for coarsened graphs. These studies establish coarse-to-full training as a scalable strategy but leave its mechanism unresolved: what relation between resolutions makes the same learned weights effective at both?

We hypothesize that self-similar multiscale organization provides this relation. Many real networks are shaped by a latent space Serrano et al. (2008); Boguna et al. (2021), often represented using

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.

hyperbolic geometry Krioukov et al. (2010), in which pairwise distances govern connection probabilities. This framework explains small-world distances, heterogeneous degree distributions, high clustering, and self-similar organization Serrano et al. (2008); Serrano & Boguñá (2022). GR merges nearby nodes in the similarity subspace and rescales their connections, producing progressively lower-resolution graphs designed to preserve the original network's multiscale organization García-Pérez et al. (2018). For networks compatible with the underlying model, these graphs are statistically equivalent replicas. GR is therefore not merely compression but an explicit geometry-preserving scale transformation. This yields a falsifiable prediction: GNNs should transfer along the GR flow, whereas matched-compression randomization of topology, latent geometry, or both should disrupt the relation between resolutions and weaken transfer.

We test this prediction with GCN, GraphSAGE, and GAT on synthetic networks from HyperBench Aliakbarisani et al. (2026) and eight real-world node-classification datasets. We evaluate transfer accuracy on G_0, representation alignment using centered kernel alignment (CKA) and orthogonal Procrustes (OP) similarity Kornblith et al. (2019), and predictive-trajectory alignment using Jensen–Shannon divergence. We compare GR with matched-compression randomization controls and established graph-coarsening methods.

Models trained on GR graphs up to $32\times$ smaller often retain much of the full-resolution accuracy while reducing training cost. Randomizing topology, latent geometry, or both substantially weakens transfer, showing that size reduction alone does not explain the effect. CKA and OP reveal aligned representations, while Jensen–Shannon comparisons show similar predictive trajectories across resolutions. GR also remains competitive with established coarsening methods in transfer accuracy while generally preserving structural properties more faithfully. Together, these results identify self-similarity as an effective structural basis for cross-scale GNN transfer providing evidence of cross-resolution compatibility over the tested scales, but do not establish exact scale equivariance.

Our contributions are:

- We formulate zero-shot transfer between resolutions of the same graph as cross-scale learning, distinct from transfer across graph instances and graph compression alone.
- Across synthetic networks, eight real-world datasets, three GNN architectures, and compression ratios up to $32\times$, we evaluate GR graphs as training surrogates for full-resolution deployment and compare them with established coarsening methods.
- Matched-compression controls show that randomizing topology, latent geometry, or both substantially weakens transfer by disrupting the self-similar relation between resolutions.
- We assess cross-resolution compatibility through transfer accuracy, CKA and OP representation alignment, and Jensen–Shannon analysis of predictive trajectories.
- We release *cuMercator*, a GPU implementation of Mercator that accelerates hyperbolic embedding by more than $400\times$ for networks with $10{,}000$ nodes, with greater speedups for larger networks.

2 Related work

Graph coarsening is widely used to reduce graph size or construct hierarchical representations. Learned pooling methods, including DiffPool Ying et al. (2018) and Graph U-Nets Gao & Ji (2019), build task-dependent coarse graphs within the model through node assignments, edge contractions, or geometric reductions Grattarola et al. (2022); Limbeck et al. (2026); Feng & Weber (2024). Preprocessing methods instead coarsen the input before training: Huang et al. Huang et al. (2021) used coarsening to scale GNN training, while Joly and Keriven Joly & Keriven (2024) showed that classical spectral coarsening need not preserve message passing and proposed operators with message-passing guarantees. These approaches focus primarily on efficient training, hierarchical pooling, or signal preservation on the coarse graph.

Network renormalization instead relates graph structure across scales Gabrielli et al. (2025). Song, Havlin, and Makse Song et al. (2005) introduced box-covering renormalization and showed that many real networks exhibit self-similar organization. Hidden-metric-space models later connected this self-similarity to latent geometry Serrano et al. (2008); Krioukov et al. (2010). Geometric

renormalization builds on this connection by embedding a graph in the hyperbolic plane and merging nearby nodes in the similarity subspace, producing a multiscale flow that preserves connection probabilities and key topological observables García-Pérez et al. (2018). It has been applied to empirical systems such as the human connectome Zheng et al. (2020) and extended to weighted networks Zheng et al. (2024). Other schemes define scales without hidden geometry, including Laplacian renormalization for heterogeneous networks Villegas et al. (2023), higher-order Laplacian renormalization for simplicial and hypergraph structures Nurisso et al. (2025), and geometry-free multiscale renormalization based on hidden variables Garuccio et al. (2023). GR is distinguished by its direct connection to analytically tractable hidden-metric-space models, making its coarse-graining and rescaling rules interpretable as a renormalization flow in the similarity subspace rather than an algorithmic graph contraction. These methods provide principled graph-scale transformations but have not been used to study the scale dependence of learned GNN weights.

A related literature examines how graph structure affects GNN performance and transferability. Performance can depend on degree distributions, joint topological and feature structure, and spectral characteristics (Li et al., 2023; Fesser & Weber, 2025). Yehudai et al. Yehudai et al. (2021) showed that GNNs may fail to generalize from small to large graph instances when local structures differ across sizes. Graphon-based analyses provide transferability guarantees for graph sequences sampled from a common limiting object Ruiz et al. (2020; 2023), while recent work shows that standard message-passing GNNs can be discontinuous across graph resolutions Koke et al. (2026). This literature establishes that cross-scale GNN weight transfer is nontrivial but does not identify which coarse representation of a fixed graph can reliably train weights for its full-resolution counterpart.

3 BACKGROUND

Random geometric graphs describe complex networks by embedding nodes in a latent metric space, where geometric distances encode similarity and link formation emerges from a trade-off between node popularity and similarity Serrano et al. (2008), with emerging hyperbolic geometry Krioukov et al. (2010). The canonical model in this framework is the $\mathbb{S}^1$ model Serrano et al. (2008), which places nodes on a circle of radius $R = N/2\pi$, representing the similarity space, and connects every pair i, j with probability

$$p_{ij} = \frac{1}{1 + \chi_{ij}^{\beta}}, \qquad \chi_{ij} \equiv \frac{R\Delta\theta_{ij}}{\mu\kappa_i\kappa_j}. \tag{1}$$

where μ controls the average degree of the network, β its clustering coefficient, and $\Delta\theta_{ij}$ is the angular distance between nodes i and j. The hidden degrees κ reflect node popularity or importance. The $\mathbb{S}^1$ model is isomorphic to the purely geometric $\mathbb{H}^2$ model Krioukov et al. (2010), in which nodes are placed in a two-dimensional hyperbolic disk by mapping each hidden degree κ_i to a radial coordinate $r_i = 2\ln(2R/(\mu\kappa_0\kappa_i))$, and the connection probability depends only on the hyperbolic distance Serrano & Boguñá (2022).

Geometric renormalization (GR) unveils the self-similarity of complex networks to construct scaled-down representations García-Pérez et al. (2018). It operates by merging neighboring nodes in the similarity subspace into supernodes and rescaling connections, averaging over short-range interactions and progressively preserving longer-range connections. The procedure is shown in Figure 1a. Given an input network G_0 at layer 0, we first construct its hyperbolic map using the embedder Mercator García-Pérez et al. (2019); Jankowski et al. (2023), obtaining the hidden degree (or radial position) and angular position of each node, (κ_i, θ_i). Then, we partition the nodes into nonoverlapping blocks of r consecutive nodes along the circle and coarse-grain each block into a supernode, independently of whether the nodes within the block are connected. Each supernode is placed within the angular region spanned by its corresponding block, thereby preserving the circular order of nodes in the original embedding throughout the renormalization process. Finally, all links between nodes belonging to two different supernodes, if any, are rescaled into a single link between those supernodes. The GR transformation produces a smaller graph that retains the original network properties, but it typically increases the average degree. We therefore prune links according to the underlying metric model to restore the original average degree and generate a replica of the original network. In Appendix A.1, we provide a detailed description of the algorithm.

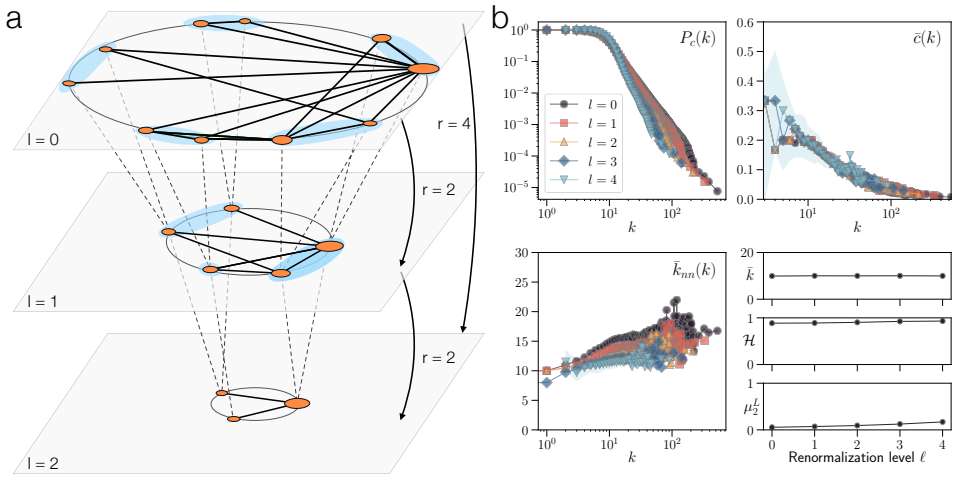

Figure 1: (**a**) Geometric renormalization (GR). Starting from the original network at $\ell = 0$, each layer applies a renormalization step of resolution r. Orange nodes lie on the circle, with sizes proportional to the logarithm of their hidden degrees; solid lines denote intralayer edges, blue regions denote coarse-graining blocks, and dashed lines map nodes to their supernodes at $\ell + 1$. Supernodes at $\ell + 1$ are linked if any pair of their constituent nodes is linked at ℓ. Under composition, GR forms an Abelian semigroup: repeated transformations at resolution r are equivalent to one transformation at higher resolution. When the number of nodes is not divisible by r, the final supernode contains fewer than r nodes, as shown at $\ell = 1$. Reproduced from García-Pérez et al. (2018). (**b**) Topological validation of GR for synthetic networks in the small-world regime over levels $\ell = 0, \ldots, 4$. The top-left, top-right, and bottom-left panels show the complementary cumulative degree distribution $P_c(k)$, clustering spectrum $\bar{c}(k)$, and average nearest-neighbor degree $\bar{k}_{nn}(k)$, respectively; solid lines and shaded regions denote medians and interquartile ranges (IQRs). The three bottom-right panels show the mean degree $\bar{k}$, homophily ratio $\mathcal{H}$, and spectral gap μ_2^L as functions of ℓ.

4 METHODS

4.1 GR-BASED TRANSFER METHODOLOGY

In this work, we use synthetic networks generated with HypBench Aliakbarisani et al. (2026) and a set of real networks. Figure 2 shows the schematic pipeline. Starting from the input graph G_0, we first embed the network into the hyperbolic disk using Mercator. For synthetic networks, this embedding step is not required because the ground-truth coordinates are known. We then apply GR to construct a coarse-grained network G_ℓ, which is used for GNN training.

To generate scaled-down replicas of a given input network, we use the geometric renormalization procedure as described in Section 3. Because nodes are divided into train and test sets, the merging

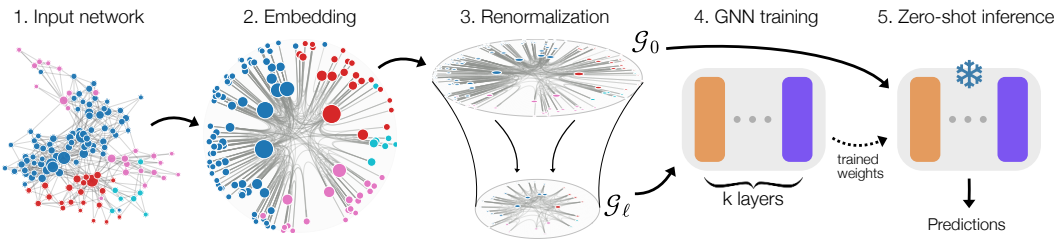

Figure 2: **Overview of the train-small, deploy-large protocol**. The input graph G_0 is embedded and renormalized to obtain a scaled-down graph G_ℓ. A GNN is trained on G_ℓ, and the learned weights are transferred without retraining to G_0 for zero-shot inference.

procedure must avoid label leakage. Each supernode label is assigned by majority vote over the training labels within the block; test labels are ignored. In the event of a tie, one of the tied labels is selected at random. Each GR step uses a block size of r. Unless stated otherwise, we use $r = 2$ in all experiments. The feature vector of each supernode is obtained by averaging the feature vectors of its constituent nodes. Throughout, ℓ denotes the renormalization level, r the block size used in each GR step, and N_0/N_ℓ the compression ratio. We use *resolution* to distinguish the original graph G_0 from its coarse-grained representations G_ℓ.

After training the GNN on G_ℓ for node classification, we evaluate the model directly on G_0 without any retraining: node representations are computed on G_0 using the weights learned on G_ℓ, and predictions are obtained for every node in the original test set. We refer to the resulting accuracy, computed on the original test nodes of G_0, as the *transfer test accuracy*. We average results over 10 random train/test splits per network (60/40 ratio) and, for synthetic networks, over 10 independently generated network realizations per configuration.

4.2 Representational similarity measures

To characterize cross-resolution transfer, we compare representations learned independently on the original graph G_0 and its GR counterpart G_ℓ. For each renormalization level ℓ, we train a GNN f_ℓ on $G_\ell = (V_\ell, E_\ell)$ using labels induced from G_0. Because the graphs have different node sets, we lift the coarse representations to V_0 using the hierarchical map $\phi_\ell \to V_\ell$, which assigns each original node to its supernode. Consequently, original nodes merged into the same supernode receive the same lifted representation.

We compare the original and lifted representations using linear centered kernel alignment (CKA) Kornblith et al. (2019); Cortes et al. (2012); Cristianini et al. (2001) and orthogonal Procrustes (OP) similarity Ding et al. (2021); Williams et al. (2021). CKA compares the induced node–node geometry, whereas OP measures alignment up to rotation or reflection. Both range from 0 to 1, with higher values indicating greater similarity. Formal definitions and further details are provided in Appendix A.2.

4.3 Functional trajectory preservation during training

To compare predictive dynamics across resolutions, we compare the class-probability distributions produced during training on the original graph G_0 and a coarse graph G_ℓ. Let $P_{j\to i}(t)$ denote the collection of node-wise categorical distributions over class labels obtained at training step t by a model trained on G_j and evaluated on G_i.

First, we compute the *training-graph* divergence, $D_{\mathrm{JS}}^{\mathrm{TG}} = D_{\mathrm{JS}}\left(P_{0\to 0}(t), P_{\ell\to 0}(t)\right)$, where D_{JS} is the Jensen-Shannon divergence. Small values of $D_{\mathrm{JS}}^{\mathrm{TG}}$ indicate that the renormalized graph acts as a good training surrogate for the original graph in output space. Second, we examine the *evaluation-graph* divergence, $D_{\mathrm{JS}}^{\mathrm{EG}} = D_{\mathrm{JS}}\left(U_\ell P_{\ell\to\ell}(t), P_{\ell\to 0}(t)\right)$, quantifying how much a model trained on G_ℓ changes when evaluated on G_0 rather than G_ℓ. Here, U_ℓ denotes the coarse-to-fine lifting operator that maps coarse-node predictions from G_ℓ back to the node set of G_0. Further definitions and implementation details are provided in Appendix A.3.

5 Experimental setup

Random Baselines. Following the topology/geometry disentanglement strategy of Zheng et al. Zheng et al. (2020), we use three random-renormalization baselines: *Random*, which randomizes both topology and node coordinates before renormalization, and *Random-T* and *Random-G*, which randomize only topology or geometry, respectively, while preserving the other. The main text compares GR with *Random*; results for the partial baselines are reported in Appendix A.9.3.

Parameters of HypBench. Using the HypBench Aliakbarisani et al. (2026) benchmark, we generate node-attributed synthetic networks with the $\mathbb{S}^1/\mathbb{H}^2$ and bipartite-$\mathbb{S}^1/\mathbb{H}^2$ models. The unipartite graphs have $N = 2^{17} = 131{,}072$, $\langle k\rangle = 10$, $\beta = 1.5$, $N_c = 6$, and homophily parameter $\alpha = 6$. We use degree exponents $\gamma = 2.5$ (ultra-small-world, USW) and $\gamma = 3.5$ (small-world, SW) Boguñá et al. (2020), whose average shortest-path lengths scale as $\mathcal{O}(\log\log N)$ and $\mathcal{O}(\log N)$, respectively.

Node features are generated from bipartite graphs with $N_f = 1000$, $\gamma_n = \gamma_f = 2.5$ for node and feature vertices, and average node degree $\langle k_n \rangle = 10$.

Real networks. For real networks, we consider eight benchmark datasets: Cora, PubMed Sen et al. (2008); Yang et al. (2016), Computers, Photo, CS, Physics Shchur et al. (2018), WikiCS Mernyei & Cangea (2020), and Flickr Zeng et al. (2020). Table 2 summarizes the main properties of each dataset. For each network, we infer β using Mercator García-Pérez et al. (2019) and estimate the exponent γ of the degree distribution following Voitalov et al. (2019). Based on the classification in Boguñá et al. (2020), networks with $\beta < 2$ and $2 < \gamma < 3$ are categorized as ultra-small-world (USW), whereas networks with $\beta < 2$ and $\gamma > 3$ are categorized as small-world (SW). We used the original dataset features.

GNN training details. We compare three GNN models: GCN Kipf & Welling (2017), GraphSAGE Hamilton et al. (2017), and GAT Veličković et al. (2018), each implemented with two layers and 32 hidden dimensions using the PyTorch Geometric library Fey & Lenssen (2019). We train the models using the Adam optimizer with an initial learning rate of 0.01 and a cosine learning-rate scheduler. We also evaluated SGD, but found that it led to weaker performance on some real-world networks. Further details are provided in Appendix A.8.

Hyperbolic embeddings of real networks. Mercator García-Pérez et al. (2019); Jankowski et al. (2023) embeds a graph in the $\mathbb{S}^1/\mathbb{H}^2$ model by maximum likelihood, requiring only the graph and returning node coordinates (θ_i, κ_i) and model parameters (β, μ). Its $\mathcal{O}(N^2)$ complexity limits large-network embedding. We therefore introduce *cuMercator*, which accelerates inference with CUDA kernels and approximates non-neighbor likelihood contributions by negative sampling. Computational benchmarks and large-network embedding estimates are provided in Appendix A.6. Table 3 reports embedding time for real networks.

6 Results

We assess whether synthetic and real networks preserve key structural properties along the GR flow. We then evaluate the transfer test accuracy of GNNs trained on renormalized networks and compare learned representations and predictive trajectories across renormalization levels.

6.1 Zero-shot GNN transfer on synthetic networks

Figure 1b presents the topological validation of geometric renormalization for synthetic networks in the small-world regime. The degree distribution, clustering spectrum, and average nearest-neighbor-degree spectrum are preserved during coarse-graining. Global properties, such as the average degree and homophily level, are also maintained. The spectral gap is slightly larger for smaller networks, as expected from its asymptotic vanishing in the large-network limit Kiwi & Mitsche (2018). Similar results are obtained in the ultra-small-world regime, see Figure 12 in Appendix A.9.1.

In Figure 3, we summarize our results for synthetic networks in the small-world regime. Panel a shows the transfer test accuracy across different renormalization levels. In most cases, it closely matches the test accuracy obtained on the original graph G_0. For GCN and GraphSAGE, performance remains high even at a compression factor of $2^5 = 32$, whereas GAT exhibits a slight decrease. In contrast, the random baseline declines in performance at higher renormalization levels. However, performance does not fall to the level of random chance because the models continue to leverage the original node features, which are not randomized. Reducing the network size also substantially improves training time, as shown in Fig. 3b. For example, for GraphSAGE, training on a coarse-grained network with 4,096 nodes instead of the original network with 131,072 nodes yields a 20-fold speedup, see also Figure 3c for the network size at each renormalization level. Figures 3d and 3e report the representational-similarity results. GNNs trained independently at different graph resolutions converge to similar hidden representations, indicating that geometric renormalization preserves the learned representation space across scales. Both linear CKA and OP similarity remain high even under strong compression, whereas random merging produces almost no similarity across scales. Figures 3f,g show the alignment of prediction dynamics across graph resolutions. The training-graph divergence $D_{\mathrm{JS}}^{\mathrm{TG}}$ (Fig. 3f) remains close to zero throughout training for GR, indicating that a model trained on the renormalized graph produces prediction distributions on G_0 that closely match those of a model trained directly on G_0. In contrast, the random baseline leads to a

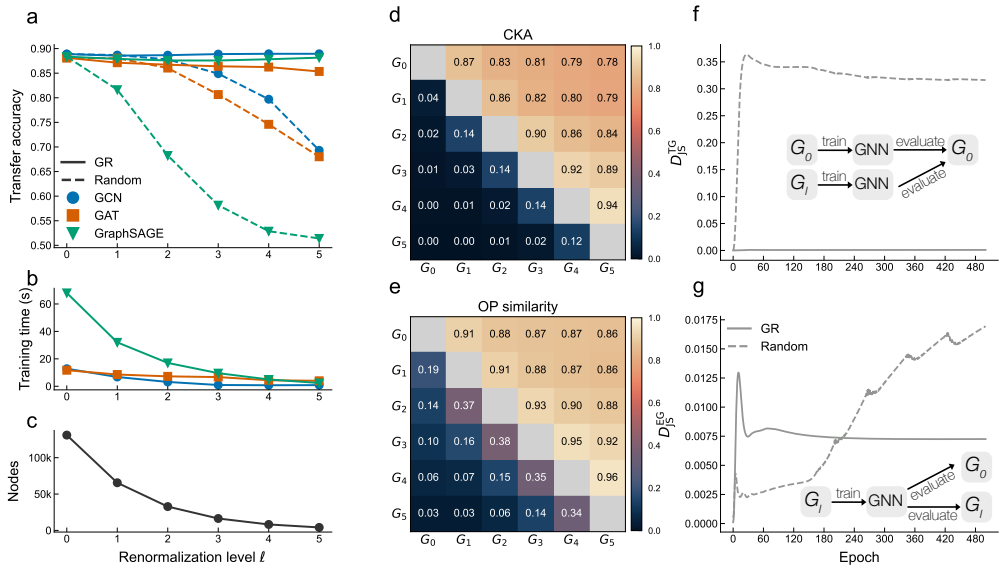

Figure 3: **Zero-shot GNN transfer on synthetic networks**. (**a**) Transfer test accuracy versus renormalization level ℓ for different GNN architectures. (**b**) GNN training time and (**c**) node count across levels. **(d, e)** Cross-level representation similarity measured by linear CKA and orthogonal Procrustes, respectively (higher is better); upper- and lower-triangular entries show GR and the random baseline, respectively. **(f, g)** Training alignment of output prediction distributions measured by Jensen–Shannon divergence D_{JS} (lower is better): $D_{\mathrm{JS}}^{\mathrm{TG}}$ in **f** and $D_{\mathrm{JS}}^{\mathrm{EG}}$ in **g**. In **a**, **f**, and **g**, solid and dashed lines denote GR and random renormalization, respectively. Panels **f** and **g** use a GCN and GR block size $r = 32$ to construct G_ℓ. All results are for the synthetic network in the small-world regime.

rapid and sustained increase in $D_{\mathrm{JS}}^{\mathrm{TG}}$, showing that random coarse graphs do not preserve the output-space training trajectory. The evaluation-graph divergence $D_{\mathrm{JS}}^{\mathrm{EG}}$ (Fig. 3g) remains small for both GR and the random baseline, indicating that models trained on coarse graphs are relatively stable when evaluated across graph resolutions. However, this stability alone is not sufficient to preserve predictive performance: despite similarly low $D_{\mathrm{JS}}^{\mathrm{EG}}$, only GR maintains high transfer test accuracy across renormalization levels, whereas the random baseline leads to a pronounced accuracy degradation (Fig. 3a). Together, these results suggest that GR preserves the functional training trajectory.

6.2 Zero-shot GNN transfer on real networks

We apply the same methodology to a set of real-world networks. Figure 4 summarizes the results for the Photo dataset, while results for the remaining datasets are reported in Appendix A.9.2. We also report in Appendix A.9.4 the topological validation of the GR transformation. Overall, we observe qualitatively similar behavior to that seen in the synthetic networks. Under geometric renormalization, transfer test accuracy remains relatively high across renormalization levels (Fig. 4a). Among the architectures considered, the simpler GCN model appears to be the most suitable for weight transfer, outperforming both GAT and GraphSAGE. Although the transfer accuracy of GraphSAGE decreases by approximately 10%, this reduction is accompanied by a roughly fourfold speedup in training time. Transfer accuracy is lower on the real-world networks than on the synthetic networks, possibly because topology, node features, and labels have more complex dependencies Jankowski et al. (2024).

The representation-similarity measures remain higher under GR than under the random baseline (Fig. 4d,e), indicating that the GR graphs better preserve the learned representation structure. As in the synthetic case, representational similarity does not fall to negligible levels under the random baseline because the models continue to leverage the original node features, which are not randomized. The functional trajectories show a qualitatively similar pattern to the synthetic networks

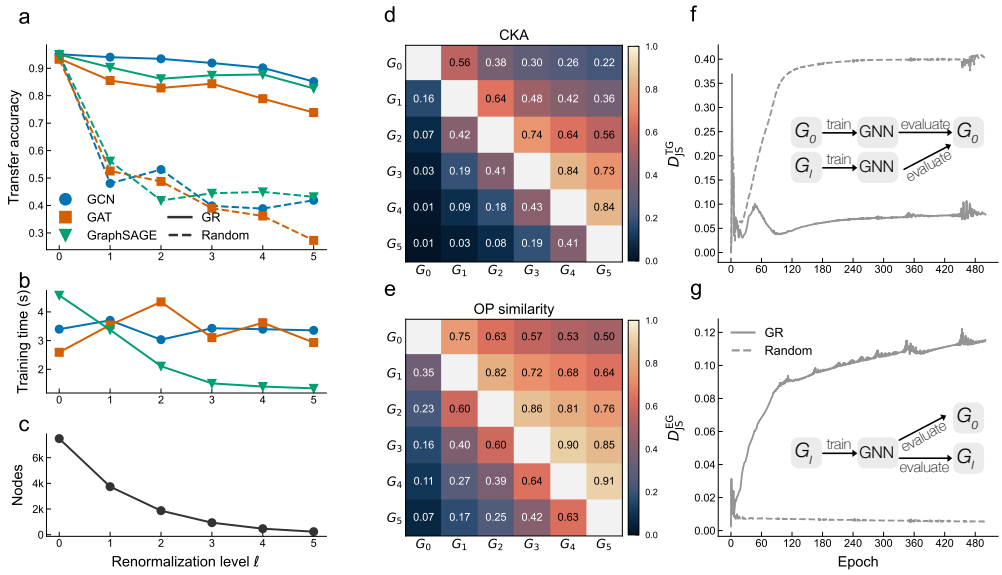

Figure 4: **Zero-shot GNN transfer on the Photo dataset**. (**a**) Transfer test accuracy and (**b**) training time across renormalization levels for the three GNN architectures; (**c**) node count at each level. (**d, e**) Cross-level representation similarity measured by linear CKA and orthogonal Procrustes, respectively (higher is better); upper- and lower-triangular entries show GR and the random baseline. (**f, g**) Predictive-trajectory alignment measured by the training-graph and evaluation-graph Jensen–Shannon divergences, $D_{\mathrm{JS}}^{\mathrm{TG}}$ and $D_{\mathrm{JS}}^{\mathrm{EG}}$, respectively (lower is better). In **a**, **f**, and **g**, solid and dashed lines denote GR and random renormalization. Results are averaged over 10 train/test splits.

(Fig. 4f,g): GR preserves the training-graph trajectory more effectively than the random baseline, while the evaluation-graph divergence remains relatively small. In Table 1, we show the transfer test accuracy in the 8 considered datasets.

Table 1: **Transfer test accuracy for eight real-world datasets.** The GCN is trained on the coarse-grained graphs G_ℓ obtained via GR, and the weights are directly applied, without retraining, to the original graph G_0. We report the mean $\pm$ standard deviation of transfer test accuracy over 10 random train/test splits. The bottom row shows the compression ratio N_0/N_ℓ at each renormalization level, obtained with a fixed block size $r = 2$.

	Original	Coarse-grained				
	$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
Computers	0.92 ± 0.01	0.90 ± 0.00	0.88 ± 0.01	0.87 ± 0.00	0.84 ± 0.01	0.80 ± 0.03
Cora	0.87 ± 0.01	0.80 ± 0.01	0.77 ± 0.01	0.77 ± 0.01	0.76 ± 0.01	0.71 ± 0.02
CS	0.94 ± 0.00	0.90 ± 0.00	0.87 ± 0.00	0.87 ± 0.00	0.88 ± 0.01	0.88 ± 0.01
Flickr	0.52 ± 0.00	0.49 ± 0.02	0.49 ± 0.01	0.50 ± 0.00	0.49 ± 0.00	0.50 ± 0.00
Photo	0.95 ± 0.00	0.94 ± 0.00	0.93 ± 0.01	0.92 ± 0.01	0.90 ± 0.01	0.85 ± 0.02
Physics	0.96 ± 0.00	0.94 ± 0.00	0.92 ± 0.00	0.92 ± 0.01	0.92 ± 0.01	0.91 ± 0.01
PubMed	0.88 ± 0.00	0.85 ± 0.00	0.85 ± 0.00	0.85 ± 0.01	0.84 ± 0.01	0.83 ± 0.00
WikiCS	0.83 ± 0.00	0.81 ± 0.00	0.79 ± 0.01	0.75 ± 0.01	0.68 ± 0.01	0.64 ± 0.01
Compression ratio	$1\times$	$2\times$	$4\times$	$8\times$	$16\times$	$32\times$

Other coarse-graining methods have been proposed in the literature. We compare GR with the Laplacian renormalization group (LRG) Villegas et al. (2023), two edge-contraction pooling methods, MagEdgePool and SpreadEdgePool Limbeck et al. (2026), and the coarse-graining method introduced in Huang et al. Huang et al. (2021). We repeat the zero-shot weight-transfer evaluation for each method and network compression rate. Specifically, we measure the difference between the transfer and original test accuracies and the combined topology preservation error.

The latter quantifies how well the coarse-graining procedure preserves key topological properties, including the degree distribution, clustering spectrum, average nearest-neighbor degree, average clustering coefficient, and spectral gap. See Appendix A.5 for more details. Figure 5 shows that GR produces the smallest accuracy gap, indicating that it maintains high transfer test accuracy even at high compression rates. It also achieves the lowest combined topological error, demonstrating superior preservation of the original network structure. The other methods generally perform worse except for lower compression rates. In Figure 7 we plot results for other datasets and GNN architectures. And in Figures 8–10, we separate the combined topology-preservation error into its individual components. The results suggest that high transfer accuracy does not require preserving every network property, though determining which properties matter and why remains an open theoretical question. Together, these results show that GR provides a faithful coarse-grained version of the original graph.

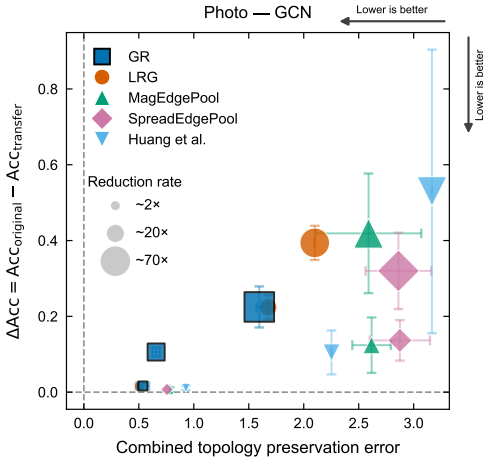

Figure 5: Topology-preservation error versus transfer-accuracy gap for different coarse-graining methods on the Photo dataset. Color denotes the method and marker size the compression rate, with larger markers indicating smaller coarse graphs. Results are averaged over 10 train/test splits.

7 Discussion

We study zero-shot cross-scale GNN transfer through geometric renormalization (GR). Parameters learned on substantially smaller GR replicas can be deployed directly on the original graph, retaining much of the full-resolution predictive performance while reducing training cost. Although geometric embedding dominates the end-to-end cost, it is computed only once per network; the resulting coarse-grained graphs can therefore be reused across hyperparameters and architectures, amortizing this cost over downstream experiments (Appendix A.7). GR is a previously proposed scale transformation designed to preserve latent geometry and self-similar multiscale organization, yielding statistically equivalent replicas when the network is compatible with the underlying model. Matched-compression randomization controls show that randomizing topology, latent geometry, or both disrupts this organization and substantially weakens transfer, ruling out graph-size reduction alone as the explanation.

Against established coarse-graining baselines, GR generally achieves comparable or smaller transfer-accuracy gaps while more faithfully preserving the measured structural properties across datasets and architectures. CKA and orthogonal Procrustes analyses reveal cross-scale alignment of learned representations, while Jensen–Shannon comparisons show similar predictive trajectories during training. Together, these results identify preserved self-similarity as an effective structural basis for cross-scale GNN transfer, but do not establish scale equivariance.

The present study focuses on a controlled experimental setting. We aggregate node features by a simple mean, leaving geometry-aware feature renormalization as a natural extension. To isolate cross-scale behavior, we use the same architecture and hyperparameters at every level; whether hyperparameters selected on a coarse graph remain optimal on G_0 is outside the present scope. The current GR formulation uses a one-dimensional similarity space, while higher-dimensional representations may reveal additional transfer behavior Ferrà Marcús et al. (2026); Almagro et al. (2022); Jankowski et al. (2023). *cuMercator* substantially accelerates embedding, although this one-time step remains the dominant component of the end-to-end cost. Finally, our experiments focus on homophilic node classification; extensions to heterophilic graphs, graph classification Koke et al. (2026), link prediction, and other graph-learning tasks remain open for future work.

Overall, our results show that preserving self-similar organization can support zero-shot GNN transfer across the tested graph scales, positioning GR as both a practical training surrogate and a controlled framework for studying learning across graph resolutions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript editing, proofreading, LaTeX checks, and refining the articulation and framing of the paper. We also used generative AI to assist with code generation, including code used in the preparation of figures. AI-generated or AI-assisted code was independently reviewed, checked, and tested for correctness by two authors before being incorporated into the work. Generative AI was used as an assistive tool rather than as an autonomous source of scientific conclusions. The authors reviewed AI-assisted text for accuracy, consistency with the underlying work, and appropriate presentation, and manually verified the correctness and suitability of AI-assisted code and figure-generation procedures. We reviewed and verified all AI suggestions and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The Methods and Experimental Setup sections specify the GR transfer protocol, feature and label aggregation, randomization controls, datasets, synthetic-network parameters, model architectures, training hyperparameters, data splits, and number of experimental repetitions. Appendix A.1 details geometric renormalization; Appendices A.2 and A.3 define the representation and trajectory metrics; and Appendices A.9.2, A.5, and A.6 provide dataset properties, comparisons with other coarse-graining methods, and computational details for *cuMercator*. The anonymous repository with code is available at `https://anonymous.4open.science/r/gnn-zero-shot-weight-transfer-02C8`.

REFERENCES

Roya Aliakbarisani, Robert Jankowski, M. Ángeles Serrano, and Marián Boguñá. Hypbench: Hyperbolic benchmark for graph neural network performance. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2026. doi: 10.1109/TNNLS.2026.3697597.

Pedro Almagro, Marián Boguñá, and M Ángeles Serrano. Detecting the ultra low dimensionality of real networks. *Nature Communications*, 13(1):6096, 2022.

Marian Boguñá, Dmitri Krioukov, Pedro Almagro, and M. Ángeles Serrano. Small worlds and clustering in spatial networks. *Physical Review Research*, 2(2):023040, Apr 2020. doi: 10.1103/PhysRevResearch.2.023040.

Marian Boguna, Ivan Bonamassa, Manlio De Domenico, Shlomo Havlin, Dmitri Krioukov, and M Ángeles Serrano. Network geometry. *Nature Reviews Physics*, 3(2):114–135, 2021.

Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. *Advances in Neural Information Processing Systems*, 34:1556–1568, 2021.

Amy Feng and Melanie Weber. Graph pooling via ricci flow. *Transactions on Machine Learning Research*, 2024.

Aina Ferrà Marcús, Robert Jankowski, Meritxell Vila-Miñana, Carles Casacuberta, and M. Ángeles Serrano. Chordless cycle filtrations for dimensionality detection in complex networks via topological data analysis. *Nature Communications*, 17(1):6105, 2026.

Lukas Fesser and Melanie Weber. Performance heterogeneity in graph neural networks: Lessons for architecture design and preprocessing. *arXiv preprint arXiv:2503.00547*, 2025.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Andrea Gabrielli, Diego Garlaschelli, Subodh P Patil, and M Ángeles Serrano. Network renormalization. *Nature Reviews Physics*, 7(4):203–219, 2025.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2083–2092. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gao19a.html.

Guillermo García-Pérez, Marián Boguñá, and M Ángeles Serrano. Multiscale unfolding of real networks by geometric renormalization. *Nature Physics*, 14(6):583–589, 2018.

Guillermo García-Pérez, Antoine Allard, M Ángeles Serrano, and Marián Boguñá. Mercator: uncovering faithful hyperbolic embeddings of complex networks. *New Journal of Physics*, 21(12): 123033, 2019.

Elena Garuccio, Margherita Lalli, and Diego Garlaschelli. Multiscale network renormalization: Scale-invariance without geometry. *Physical Review Research*, 5(4):043101, 2023.

Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE transactions on neural networks and learning systems*, 35(2):2708–2718, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21)*, pp. 675–684, 2021. doi: 10.1145/3447548.3467256.

Robert Jankowski, Antoine Allard, Marian Boguna, and M Ángeles Serrano. The d-mercator method for the multidimensional hyperbolic embedding of real networks. *Nature Communications*, 14(1): 7585, 2023.

Robert Jankowski, Pegah Hozhabrierdi, Marián Boguñá, and M Ángeles Serrano. Feature-aware ultra-low dimensional reduction of real networks. *npj Complexity*, 1(1):13, 2024.

Antonin Joly and Nicolas Keriven. Graph coarsening with message-passing guarantees. *Advances in Neural Information Processing Systems*, 37:114902–114927, 2024.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Marcos Kiwi and Dieter Mitsche. Spectral gap of random hyperbolic graphs and related parameters. *The Annals of Applied Probability*, 28(2):941 – 989, 2018. doi: 10.1214/17-AAP1323. URL https://doi.org/10.1214/17-AAP1323.

Christian Koke, Yuesong Shen, Abhishek Saroha, Marvin Eisenberger, Bastian Rieck, Michael Bronstein, and Daniel Cremers. Graph neural networks are not continuous across graph resolutions. *arXiv preprint arXiv:2605.31315*, 2026.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82(3):036106, 2010.

Ting Wei Li, Qiaozhu Mei, and Jiaqi Ma. A metadata-driven approach to understand graph neural networks. *Advances in Neural Information Processing Systems*, 36:15320–15340, 2023.

Katharina Limbeck, Lydia Mezrag, Guy Wolf, and Bastian Rieck. Geometry-aware edge pooling for graph neural networks. *Advances in Neural Information Processing Systems*, 38:157770–157808, 2026.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature Communications*, 8(1):1615, 2017.

Marco Nurisso, Marta Morandini, Maxime Lucas, Francesco Vaccarino, Tommaso Gili, and Giovanni Petri. Higher-order laplacian renormalization. *Nature Physics*, 21(4):661–668, 2025.

Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33:1702–1712, 2020.

Luana Ruiz, Luiz FO Chamon, and Alejandro Ribeiro. Transferability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 71:3474–3489, 2023.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

M Ángeles Serrano and Marián Boguñá. *The Shortest Path to Network Geometry: A Practical Guide to Basic Models and Applications*. Cambridge University Press, 2022.

M Ángeles Serrano, Dmitri Krioukov, and Marián Boguná. Self-similarity of complex networks and hidden metric spaces. *Physical Review Letters*, 100(7):078701, 2008.

Oleksandr Shchur, Max Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. doi: 10.48550/arXiv.1811.05868.

Chaoming Song, Shlomo Havlin, and Hernan A Makse. Self-similarity of complex networks. *Nature*, 433(7024):392–395, 2005.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Pablo Villegas, Tommaso Gili, Guido Caldarelli, and Andrea Gabrielli. Laplacian renormalization group for heterogeneous networks. *Nature Physics*, 19(3):445–450, 2023.

Ivan Voitalov, Pim Van Der Hoorn, Remco Van Der Hofstad, and Dmitri Krioukov. Scale-free networks well done. *Physical Review Research*, 1(3):033034, 2019.

Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. *Advances in neural information processing systems*, 34:4738–4750, 2021.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pp. 11975–11986. PMLR, 2021.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)*, 2020.

Muhua Zheng, Antoine Allard, Patric Hagmann, Yasser Alemán-Gómez, and M Ángeles Serrano. Geometric renormalization unravels self-similarity of the multiscale human connectome. *Proceedings of the National Academy of Sciences*, 117(33):20244–20253, 2020.

Muhua Zheng, Guillermo García-Pérez, Marián Boguñá, and M Ángeles Serrano. Geometric renormalization of weighted networks. *Communications Physics*, 7(1):97, 2024.

A Appendix

A.1 Geometric renormalization in detail

Here, we describe the renormalizability of the $\mathbb{S}^1$ model. For more details, such as the semigroup structure of the coarse-graining step and the solution in the power-law approximation, please refer to the Supplementary Information of García-Pérez et al. (2018). The renormalized networks remain maximally congruent with the hidden metric space model by assigning a new hidden degree $\kappa_i^{(l+1)}$ to supernode i in layer $l+1$ as a function of the hidden degrees of nodes it contains in layer l according to

$$\kappa_i^{(l+1)} = \left(\sum_{j=1}^{r} \left(\kappa_j^{(l)} \right)^{\beta} \right)^{1/\beta} \tag{2}$$

as well as an angular coordinate $\theta_i^{(l+1)}$ given by

$$\theta_i^{(l+1)} = \left(\frac{\sum_{j=1}^{r} \left(\theta_j^{(l)} \kappa_j^{(l)} \right)^{\beta}}{\sum_{j=1}^{r} \left(\kappa_j^{(l)} \right)^{\beta}} \right)^{1/\beta} . \tag{3}$$

The global parameters also need to be rescaled as $\mu^{(l+1)} = \mu^{(l)}/r$, $\beta^{(l+1)} = \beta^{(l)}$, and $R^{(l+1)} = R^{(l)}/r$. This implies that the probability $p_{ij}^{(l+1)}$ for two supernodes i and j to be connected in layer $l+1$ maintains its original form (Eq. 1). This applies both to the model and to real networks as long as they admit a good embedding. During the renormalization flow, the average degree increases according to $\langle k \rangle^{(l+1)} = r^{\nu} \langle k \rangle^{(l)}$, where ν is a scaling exponent determined by the connectivity structure of the original network. We therefore adjust the average degree of the downscaled network replicas. Specifically, to match the average degree of a renormalized network to that of the original network, we prune links according to the underlying metric model with which networks at all layers are congruent.

A.2 Representational similarity measures

Let $n = |V_0|$ and let d denote the common representation dimension of all models, which use the same GNN architecture. We write

$$Z_0 \in \mathbb{R}^{n \times d}, \qquad Z_\ell \in \mathbb{R}^{|V_\ell| \times d} \tag{4}$$

for the node representations learned on the original and coarse-grained graphs. The coarse-scale representations are lifted to the original node set so that $Z_\ell^\uparrow \in \mathbb{R}^{n\times d}$. We center each feature across nodes, obtaining X_0 and $X_\ell^\uparrow$.

We compute linear CKA as

$$\mathrm{CKA}(X_0, X_\ell^\uparrow) = \frac{\|X_0^\top X_\ell^\uparrow\|_F^2}{\|X_0^\top X_0\|_F \, \|(X_\ell^\uparrow)^\top X_\ell^\uparrow\|_F}, \tag{5}$$

and normalized Orthogonal Procrustes similarity as

$$\mathrm{OP}(X_0, X_\ell^\uparrow) = \frac{\|X_0^\top X_\ell^\uparrow\|_*}{\|X_0\|_F \|X_\ell^\uparrow\|_F}, \tag{6}$$

Here, $\|\cdot\|_F$ is the Frobenius norm, i.e., the square root of the sum of squared matrix entries, and $\|\cdot\|_*$ is the nuclear norm, i.e., the sum of the singular values.

For nonzero representation matrices, both measures are in $[0, 1]$. A value of 1 indicates maximal similarity under the corresponding criterion, whereas 0 indicates no shared linear structure. CKA compares the geometry induced among nodes and is invariant to isotropic scaling and orthogonal transformations. OP measures the maximal similarity after rotating or reflecting one representation.

A.3 Functional trajectory diagnostics

Representation-similarity measures characterize learned embeddings but do not directly determine whether training on a graph-renormalization replica follows the same predictive trajectory as training on the original graph. We therefore compare model outputs throughout training rather than gradients in parameter space, which may be sensitive to parameterization and training noise.

Let $Y_v \in \{1, \ldots, C\}$ denote the class-label random variable for node v, where C is the number of classes. Let $\omega_t^{(j)}$ denote the parameters of a GNN after training for t steps on graph G_j. For a node v in the evaluation graph G_i, define

$$P_{j\to i}^{(v)}(t) = \mathrm{softmax}\Big(f_{\omega_t^{(j)}}(G_i)_v\Big). \tag{7}$$

We use $P_{j\to i}(t)$ to denote the collection of these node-level distributions over the evaluated node set. In particular, $P_{0\to 0}(t)$ corresponds to training and evaluation on G_0, $P_{\ell\to\ell}(t)$ to training and evaluation on G_ℓ, and $P_{\ell\to 0}(t)$ to training on G_ℓ followed by evaluation on G_0.

$$D_{\mathrm{JS}}^{\mathrm{TG}}(t) = D_{\mathrm{JS}}\left(P_{0\to 0}(t), P_{\ell\to 0}(t)\right). \tag{8}$$

This quantity isolates the effect of replacing G_0 with G_ℓ during training while keeping the evaluation graph fixed. Small values indicate that G_ℓ acts as an effective training surrogate for G_0 in output space.

$$D_{\mathrm{JS}}^{\mathrm{EG}}(t) = D_{\mathrm{JS}}\left(U_\ell P_{\ell\to\ell}(t), P_{\ell\to 0}(t)\right). \tag{9}$$

This quantity keeps the trained model fixed and compares its predictions at the coarse and original graph resolutions. The operator U_ℓ maps each coarse-node prediction to the corresponding nodes of G_0. Small values therefore indicate that the coarse-trained model transfers stably to the original graph. Both divergences are evaluated throughout training. Together, they distinguish the quality of G_ℓ as a training surrogate from the stability of evaluating a coarse-trained model at the original graph resolution.

A.4 Properties of real networks

A.5 Comparison with other coarse-graining methods

We compare geometric renormalization (GR) with the Laplacian renormalization group (LRG), which identifies appropriate spatiotemporal scales in heterogeneous networks Villegas et al. (2023), with two edge-contraction pooling methods, MagEdgePool and SpreadEdgePool, which iteratively

Table 2: Properties of the real datasets. N: number of nodes; E: number of edges; N_f: number of node features; N_c: number of classes; $\bar{k}$: average degree; $\mathcal{H}$: homophily ratio; γ: degree-distribution exponent; β: clustering-coefficient parameter of the $\mathbb{S}^1/\mathbb{H}^2$ model, inferred with Mercator; $\bar{c}$: average clustering coefficient; Regime: structural classification of the network as ultra-small-world (USW) or small-world (SW), following Boguñá et al. (2020), based on the inferred values of β and γ.

Dataset	N	E	N_f	N_c	$\bar{k}$	$\mathcal{H}$	γ	β	$\bar{c}$	Regime
Computers	13,381	245,778	767	10	36.7	0.78	2.96	1.46	0.36	USW
Cora	2,485	5,069	1433	7	4.1	0.81	3.08	1.54	0.29	SW
CS	18,333	81,894	6805	15	8.9	0.81	6.11	1.72	0.37	SW
Flickr	89,250	449,878	500	7	10.1	0.32	2.64	1.01	0.08	USW
Photo	7,487	119,043	745	8	31.8	0.83	2.99	1.65	0.41	USW
Physics	34,493	247,962	8415	5	14.4	0.93	5.0	1.71	0.38	SW
PubMed	19,717	44,324	500	3	4.5	0.8	4.18	1.12	0.11	SW
WikiCS	11,311	215,554	300	10	38.1	0.65	3.72	1.61	0.5	SW

contract the least important edges and are computed before GNN training Limbeck et al. (2026), and with the method of Huang et al. Huang et al. (2021).

For LRG, we first fix the network's temporal resolution parameter τ. For GR, we choose the resolution parameter r so that the final size of the coarse-grained network matches the target compression level. We use the corresponding pooling ratio for MagEdgePool and SpreadEdgePool. For Huang et al.'s method, we set the correct coarsening ratio. This method partitions nodes into supernodes to construct a smaller weighted graph on which the GNN is trained, before transferring the learned parameters back to the original graph for inference.

We first compare the topological properties of the coarse-grained networks produced by each method. In Figure 6a-f, we show the degree distribution, clustering spectrum, average nearest-neighbor degree spectrum, and global network properties, including the average degree, average clustering coefficient, and spectral gap. Overall, GR preserves both local and global network properties more accurately than the other methods. We then evaluate transfer accuracy (Fig. 6g). GR and the Huang et al. method yield the smallest drop in transfer accuracy compared with a GNN trained on the original PubMed network.

In Figure 5, we compute the combined topology-preservation error as the sum of the normalized discrepancies across all topology metrics except average degree, because GR explicitly preserves the average degree through network pruning García-Pérez et al. (2018). This combined topology preservation error is defined as the sum of these normalized topology errors as

$$\mathrm{Err}_{\mathrm{topo}}(G_\ell, G_0) = \widetilde{\mathrm{Err}}_{P(k)} + \widetilde{\mathrm{Err}}_{c(k)} + \widetilde{\mathrm{Err}}_{k_{\mathrm{nn}}(k)} + \widetilde{\mathrm{Err}}_{\bar{c}} + \widetilde{\mathrm{Err}}_{\lambda_2}, \tag{10}$$

where $\widetilde{\mathrm{Err}}_i$ is the normalized error for topology statistic i, obtained by dividing the corresponding raw error by the maximum raw error of the same statistic across all coarse-grained graphs included in the comparison. Lower values indicate better preservation of the original topology. The degree-distribution term $\mathrm{Err}_{P(k)}$ is the Anderson–Darling statistic between the node-degree samples of G_0 and G_ℓ. The spectrum terms $\mathrm{Err}_{c(k)}$ and $\mathrm{Err}_{k_{\mathrm{nn}}(k)}$ are root-mean-square errors (RMSE) between the original and coarse-grained degree-dependent clustering spectrum $c(k)$ and average-neighbor-degree spectrum $k_{\mathrm{nn}}(k)$, respectively. The mean-clustering term is the relative error $\mathrm{Err}_{\bar{c}} = \frac{|\bar{c}_\ell - \bar{c}_0|}{\bar{c}_0}$, and the spectral-gap term is the absolute log-ratio error $\mathrm{Err}_{\lambda_2} = \left|\log \frac{\lambda_{2,\ell}+\epsilon}{\lambda_{2,0}+\epsilon}\right|$, where λ_2 is the second-smallest eigenvalue of the normalized graph Laplacian and $\epsilon = 10^{-6}$.

Figure 7 extends the analysis in Figure 5 to three additional datasets and three GNN architectures. In most cases, GR achieves the lowest combined topology-preservation error. However, a low topology-preservation error does not necessarily correspond to a small difference between the original and transfer test accuracies. For instance, on Pubmed, Huang et al.'s method exhibits the highest topology-preservation error while still preserving test accuracy well.

We further investigate this effect in Figure 8, where we plot each topology-preservation error separately. The errors in the degree distribution and average neighbor degree show little correlation with

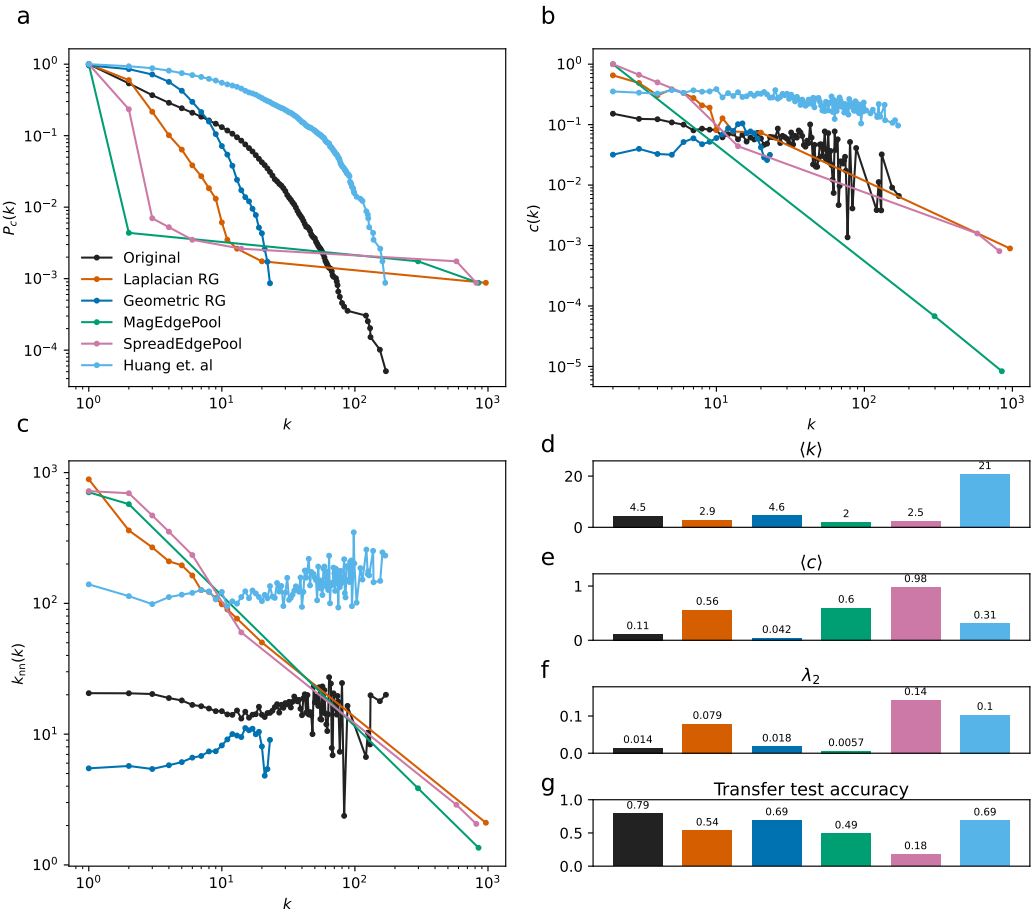

Figure 6: Comparison of coarse-graining methods on the PubMed dataset. We report: (a) the complementary cumulative degree distribution, (b) the clustering spectrum, (c) the average-neighbor-degree spectrum, (d) the average degree, (e) the average clustering coefficient, (f) the spectral gap, and (g) the transfer test accuracy. For LRG, we use $\tau = 1.2$, which yields approximately 1100 nodes, compressing the original network by a factor of approximately 17. For GR, MagEdgePool, SpreadEdgePool, and Huang et al.'s method, we match this number of nodes by varying the resolution parameter r for GR, the pooling ratio for MagEdgePool and SpreadEdgePool, and the coarsening ratio for Huang et al.'s method. For the transfer task, we train a two-layer GCN with 32 hidden dimensions.

transfer performance. In contrast, the spectral gap error (panel f) appears to be more correlated with transfer performance. However, further work is needed to establish this relationship. In Figure 9, we present the results for Cora using GAT, while Figure 10 shows the corresponding results for Pubmed using GraphSAGE.

Finally, we compare the computational complexity of the methods. Assuming that the input graph is sparse, i.e., $E = \mathcal{O}(N)$, the GR coarse-graining procedure scales linearly with the network size, $\mathcal{O}(N)$. However, GR first requires a hyperbolic embedding of the network using Mercator, whose standard time complexity is $\mathcal{O}(N^2)$. As shown in Appendix A.6, GPU acceleration can substantially reduce the practical runtime of this embedding step. LRG requires $\mathcal{O}(N^3)$ time due to the eigendecomposition and matrix exponential operations. MagEdgePool and SpreadEdgePool have time complexity $\mathcal{O}(N^3)$ on sparse graphs when approximate diffusion distances are used. For Huang et al.'s method using Variation Neighborhoods, the coarsening complexity is $\widetilde{\mathcal{O}}(N\Delta^2)$ for sparse graphs when the spectral subspace dimension and number of coarsening levels are fixed, reducing to near-linear $\widetilde{\mathcal{O}}(N)$ for bounded maximum degree Δ.

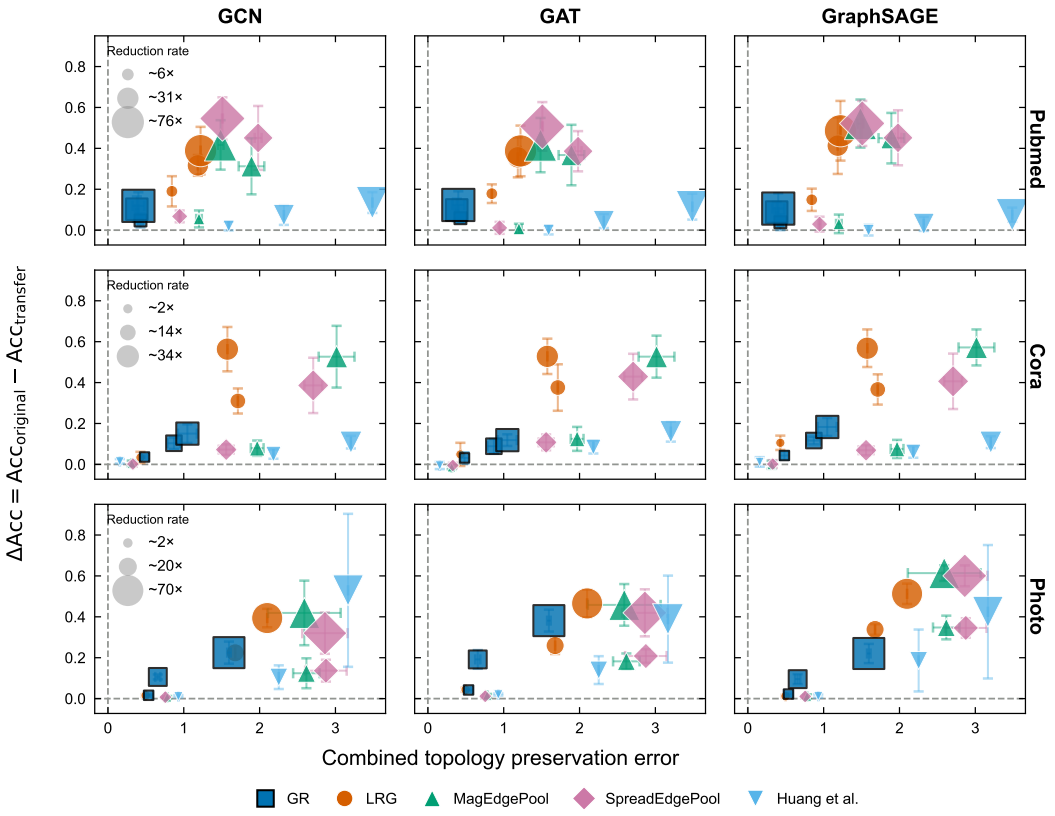

Figure 7: Combined topology preservation error as a function of the difference between the transfer and original test accuracies for different coarse-graining methods. Each row corresponds to one dataset and each column to one GNN architecture. Marker color denotes the coarse-graining method, while marker size represents the compression rate. Larger markers correspond to smaller coarse-grained networks. Results are averaged over 10 train-test splits.

A.6 Computational analysis of Mercator on GPU

Mercator García-Pérez et al. (2019); Jankowski et al. (2023) is a reliable method for embedding complex networks into their underlying hyperbolic latent geometry. The algorithm combines machine-learning techniques with maximum-likelihood (ML) optimization to infer node coordinates in the hyperbolic disk by maximizing the agreement between the observed network topology and the geometric model. The original implementation has a time complexity that scales quadratically with the network size.

In this work, we accelerate the ML stage by moving the most computationally expensive operations to a GPU. In Figure 11a, we report the time required to embed networks of different sizes. The GPU implementation, called *cuMercator*, provides substantial speedup compared with the CPU version. However, the asymptotic time complexity remains $\mathcal{O}(N^2)$. To reduce this cost, we introduce negative sampling into the ML calculation. Instead of computing the log-likelihood contribution from all non-neighbors at each newly proposed position, we sample only S non-neighbors. This significantly reduces the cost of evaluating the non-neighbor contribution. In Figure 11b, we compare the embedding time obtained with exact ML calculations against the negative-sampling approximation with $S = 64$. We observe a substantial reduction in runtime while retaining high-quality embeddings. Negative sampling may prevent the final log-likelihood from reaching the exact optimum. Nevertheless, Figure 11c shows that the resulting c-score Muscoloni et al. (2017) remains very high, indicating strong agreement between the ordering of nodes in the original and inferred angular coordinates. These changes allow us to embed very large networks efficiently and use the inferred node coordinates for geometric renormalization.

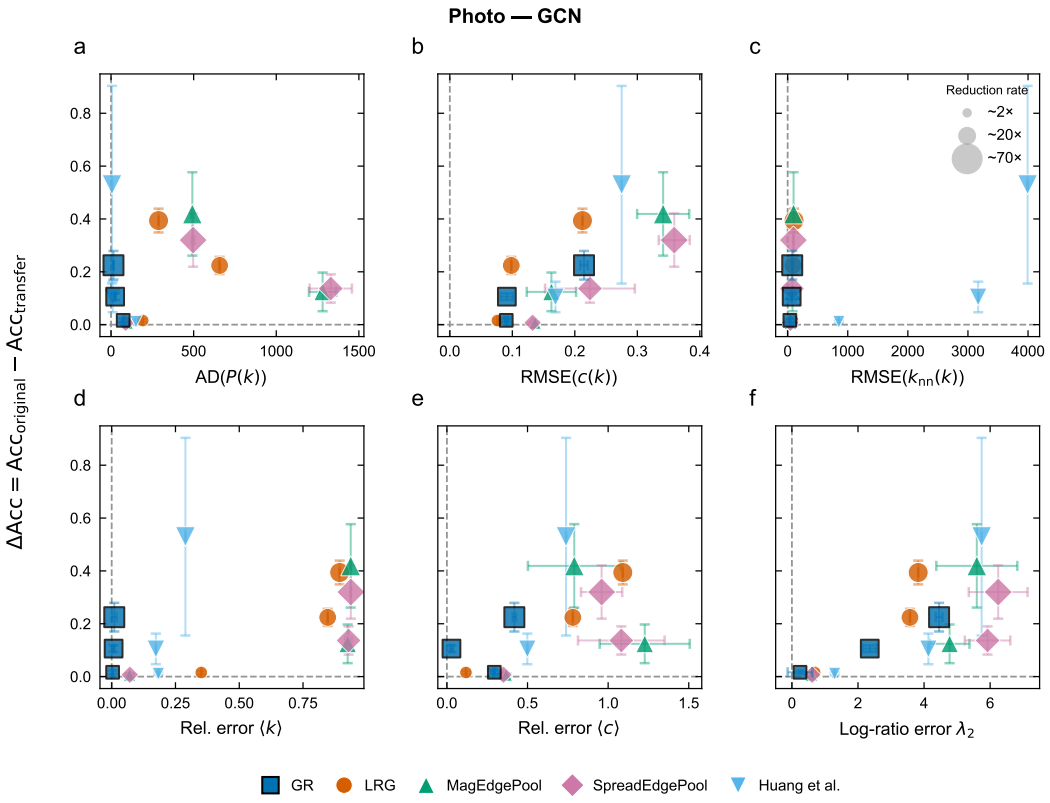

Figure 8: Topology preservation errors as a function of the difference between the transfer and original test accuracies for different coarse-graining methods for Photo trained on GCN model. (**a**) Anderson-Darling statistics between degree distributions, (**b**) Root-mean-square-error (RMSE) between clustering spectra, (**c**) RMSE between average neighbor degree spectra, (**d**) relative error in average degrees, (**e**) relative error in mean clustering coefficient, (**f**) absolute log-ratio error in spectral gaps. Marker color denotes the coarse-graining method, while marker size represents the compression rate. Larger markers correspond to smaller coarse-grained networks. Results are averaged over 10 train-test splits.

A.7 End-to-end computational cost

Table 3 reports embedding time (computed once per dataset) and renormalization time (computed once per dataset and level). Since both quantities are independent of the GNN architecture, these values apply uniformly across all GNN architectures considered in this work. The embedding time dominates the preprocessing cost; we report it on both a single H200 GPU and a single A100 GPU. In contrast, renormalization is comparatively inexpensive across all datasets and levels, on the order of a few seconds at most, and tends to decrease with level as the coarse-grained graphs shrink, though not strictly monotonically in every case due to the iterative pruning step.

Importantly, this one-time embedding cost is amortized across all renormalization levels and GNN architectures: once a network is embedded, the same coordinates are reused to generate coarse-grained graphs at every level and to train any model on them, without recomputing the embedding. This amortization is particularly relevant for workloads that require many repeated training runs on the same graph, such as sweeping over hyperparameter configurations or training model ensembles, which could be performed on the reusable coarse-grained replicas at a lower cost than at the original scale.

For K repeated training runs, the total computational costs are $T_{\text{full}} = KT_F$ for training on the full graph and $T_\ell = T_E + T_R + KT_C$ for training on the coarse-grained graph. Here, T_F and T_C denote the training time for a single model on the full and coarse-grained graphs, respectively, while T_E

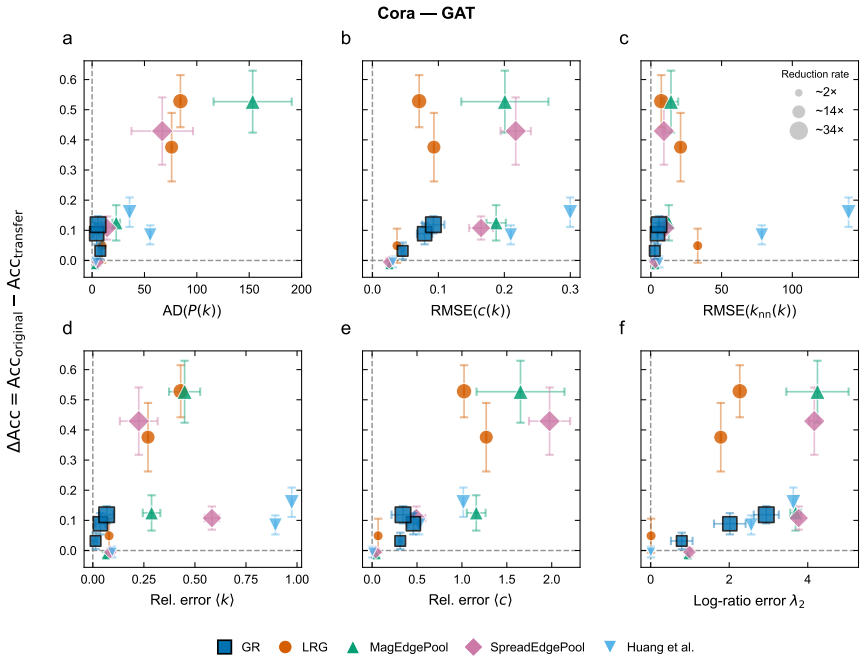

Figure 9: Results for Cora on GAT. See caption of Figure 8.

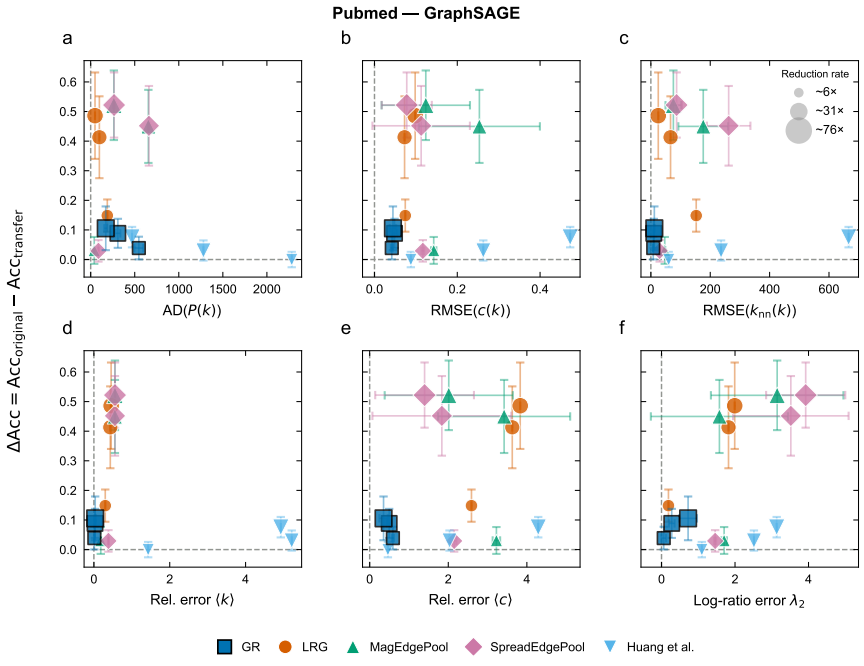

Figure 10: Results for PubMed on GraphSAGE. See caption of Figure 8.

is the embedding time and T_R is the renormalization time (Table 3). The coarse-grained approach becomes computationally advantageous when $T_E + T_R + KT_C < KT_F$, which yields the break-even number of training runs $K_b = \frac{T_E+T_R}{T_F-T_C}$. We report the corresponding K_b values for five real datasets in Table 4. In general, the break-even point is on the order of $10^2 - 10^3$ training runs. Such a number of runs is commonly required in a modest hyperparameter search; for example, evaluating

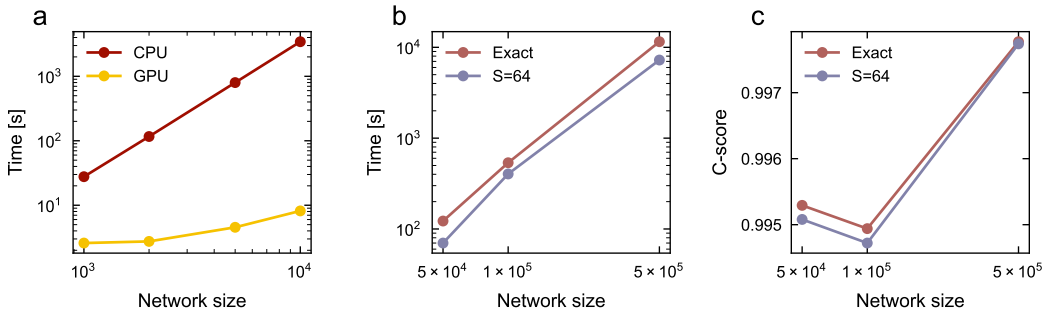

Figure 11: Computational performance of Mercator. (**a**) Embedding time as a function of network size for the CPU and GPU implementations. (**b**) Embedding time on the GPU using either exact maximum-likelihood (ML) calculations or negative sampling with $S = 64$. (**c**) c-score between the original and inferred angular coordinates for embeddings obtained with exact ML and negative sampling. Model parameters are $\beta = 2$, $\gamma = 10$, and $\langle k \rangle = 10$. All experiments were performed on a single H200 GPU.

Table 3: Embedding and renormalization time (s), by dataset and renormalization level.

Dataset	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	Embed. (H200)	Embed. (A100)
CS	0.520	0.282	0.306	0.117	0.076	22.040	187.065
Computers	0.495	0.290	0.166	0.431	0.085	84.060	568.858
Cora	0.055	0.029	0.016	0.010	0.007	5.050	18.180
Flickr	2.583	1.391	1.250	1.152	2.369	409.900	2002.180
Photo	0.322	0.204	0.118	0.057	0.036	45.140	617.801
Physics	1.040	0.626	0.406	0.248	0.199	58.270	450.587
PubMed	0.447	0.235	0.132	0.120	0.072	19.620	221.545
WikiCS	0.320	0.231	0.135	0.145	0.095	104.840	787.615

five different numbers of layers, two model architectures, and ten random seeds already requires $5 \times 2 \times 10 = 100$ training runs.

Table 4: Break-even number of repeated GNN training runs K_b required to amortize the one-time embedding and renormalization costs at $\ell = 5$.

Dataset	GCN	GAT	GraphSAGE
CS	3344	513	8
Flickr	430	675	102
Physics	94	206	6
PubMed	417	371	116
WikiCS	1228	432	137

As a concrete illustration of this general result, Table 5 reports the end-to-end cost of a configuration-sweep workload on Physics and PubMed. For each entry, we train GCN over a grid of 4 depths and 4 hidden-dimension sizes, with 50 random realizations per configuration, for a total of 800 training runs. The *Train* ($\ell = 0$) column reports the cost of running this entire grid on the original graph G_0. The remaining columns report the one-time hyperbolic embedding and renormalization costs to reach level 5, the cost of running the same grid on the renormalized graph G_5, and their total. Because the embedding and renormalization overhead is paid only once and amortized across all 800 runs, the total cost at level 5 is lower than training on the original graph for both datasets, with savings consistent with the K_b values reported in Table 4: since $K = 800$ runs exceeds K_b for both datasets, the coarse-grained approach remains advantageous, though the margin is narrower for PubMed. The *Embedding* column reflects the overhead of loading CUDA, which is far larger than

Table 5: End-to-end total time (s) for GCN on Physics and PubMed, for a grid of 4 depths × 4 hidden-dimension sizes with 50 realizations each (800 training runs per entry). *Total ($\ell = 5$)* sums *Embedding* + *Renorm ($\ell = 5$)* + *Train ($\ell = 5$)*.

Dataset	Train ($\ell = 0$)	Embedding	Renorm ($\ell = 5$)	Train ($\ell = 5$)	Total ($\ell = 5$)
Physics	5665.47	450.59	0.20	2397.96	2848.74
PubMed	3738.76	221.54	0.07	3309.20	3530.82

Table 6: Peak GPU memory usage (GB) for GCN on Physics and PubMed, for the same grid as Table 5. *Max ($\ell = 5$)* is the larger of *Embedding* and *Train ($\ell = 5$)*.

Dataset	Train ($\ell = 0$)	Embedding	Train ($\ell = 5$)	Max ($\ell = 5$)
Physics	2.84	0.41	1.25	1.25
PubMed	0.26	0.41	0.07	0.41

the memory needed for Mercator's own data structures at the network sizes considered here; we report it for completeness rather than as a size-scaling benchmark.

A.8 Choice of optimizer

In this work, we used Adam as the optimizer for GNN training. We also tested stochastic gradient descent (SGD) as a baseline. While SGD led to convergence on the synthetic networks, its performance was weaker on some real-world networks. Therefore, we used Adam in all experiments to ensure stable training and high accuracy across all considered networks. It is worth noting one limitation of using Adam when analyzing training trajectories. With SGD on synthetic networks, we were able to extract raw gradients and directly compare the gradients of GNNs trained on the original and coarse-grained graphs. With Adam, however, the raw gradients were noisier and less stable for comparison. For this reason, we instead compared the softmax prediction trajectories, which provided a more stable basis for analyzing training alignment.

A.9 Additional results

A.9.1 Synthetic networks

Figure 12 shows the topological validation of geometric renormalization in the ultra-small-world (USW) regime. Similarly to the small-world regime, the topological properties are preserved. Figure 13 summarizes the zero-shot transfer results.

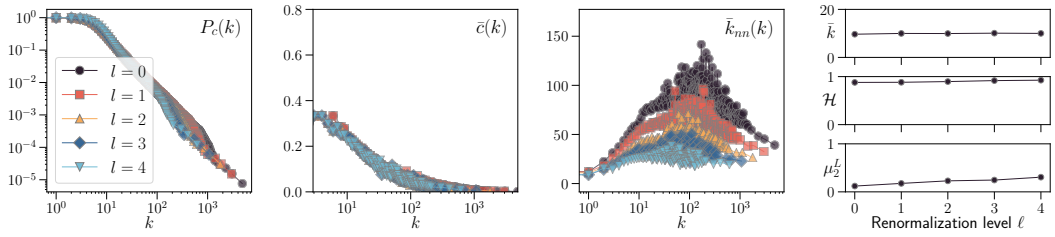

Figure 12: Topological validation of geometric renormalization in the ultra-small-world (USW). See the caption of Fig. 1 for panel details. Across all renormalization levels, these topological and mesoscale properties remain stable, confirming that geometric renormalization preserves the structural organization of the network in this regime as well.

A.9.2 Real networks

Figures 14–20 show transfer accuracy and training time across renormalization levels for the remaining real-world datasets.

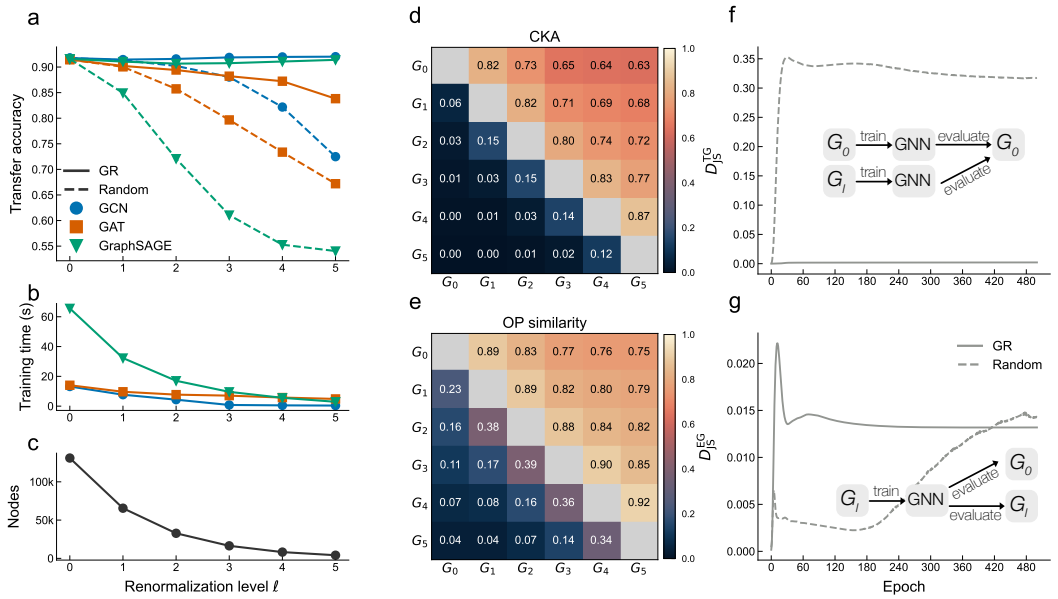

Figure 13: Zero-shot GNN transfer on synthetic networks in the USW regime. See the caption of Figure 3 for more details.

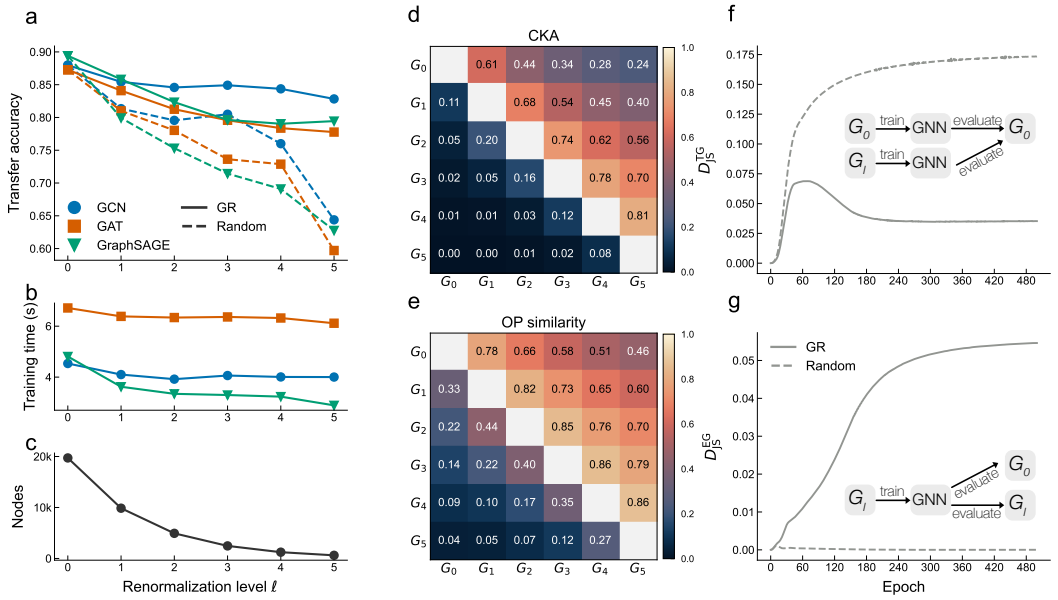

Figure 14: **Zero-shot GNN transfer on PubMed dataset**. See the caption of Fig. 3 for more detail.

A.9.3 Comparison with Random-T and Random-G baselines

In the main text, we compared geometric renormalization (GR) against a fully randomized baseline (*Random*), in which both the network topology and the geometric coordinates used to merge nodes are randomized before coarse-graining. To disentangle the individual contribution of these two factors, we here report results for two partial baselines: *Random-T*, in which only the topology is randomized while node coordinates are kept as inferred by Mercator, and *Random-G*, in which only the geometric coordinates are randomized while the original topology is preserved. Comparing GR against these two variants isolates whether preserving topology, preserving geometry, or their combination is responsible for the transfer performance observed in the main text.

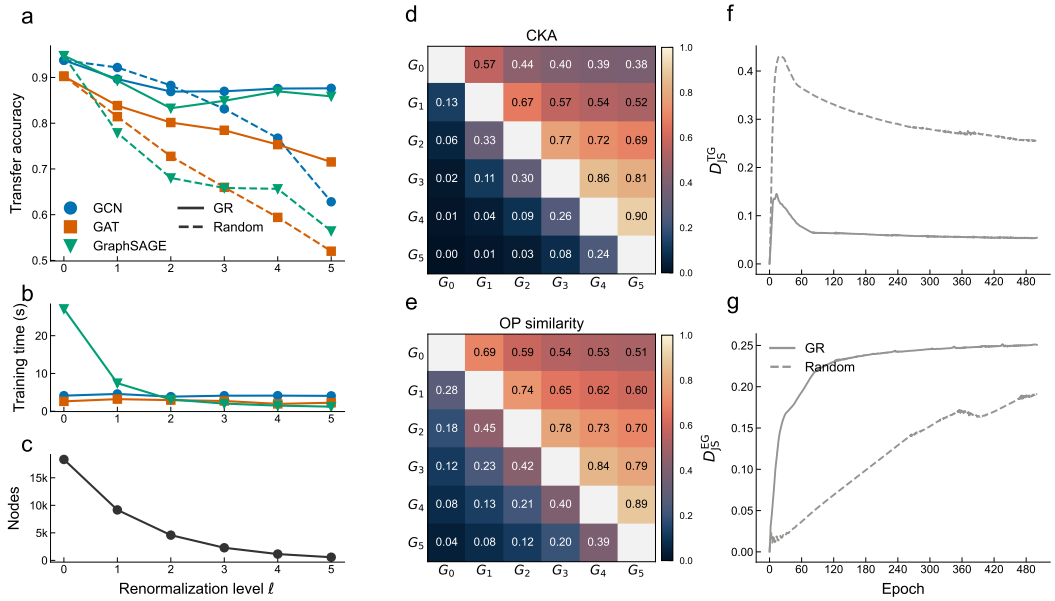

Figure 15: **Zero-shot GNN transfer on CS dataset**. See the caption of Fig. 3 for more detail.

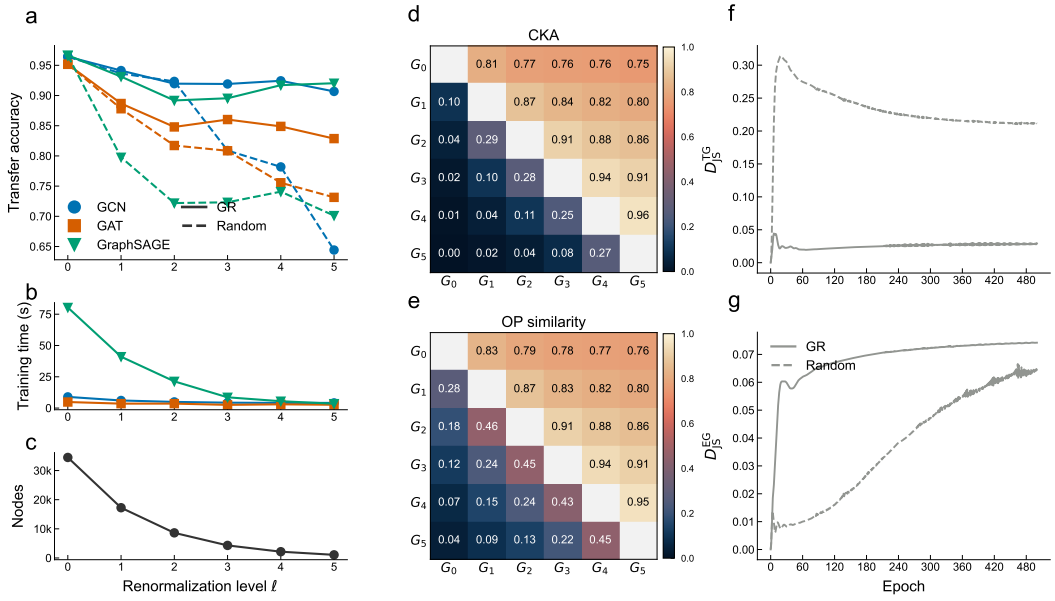

Figure 16: **Zero-shot GNN transfer on Physics dataset**. See the caption of Fig. 3 for more detail.

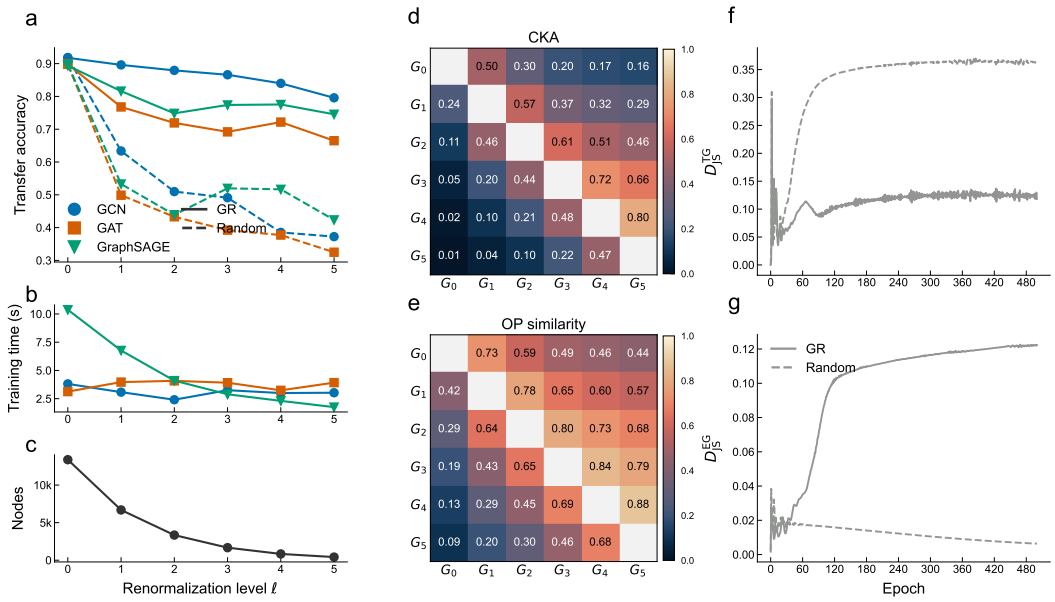

Figure 17: **Zero-shot GNN transfer on Computers dataset**. See the caption of Fig. 3 for more detail.

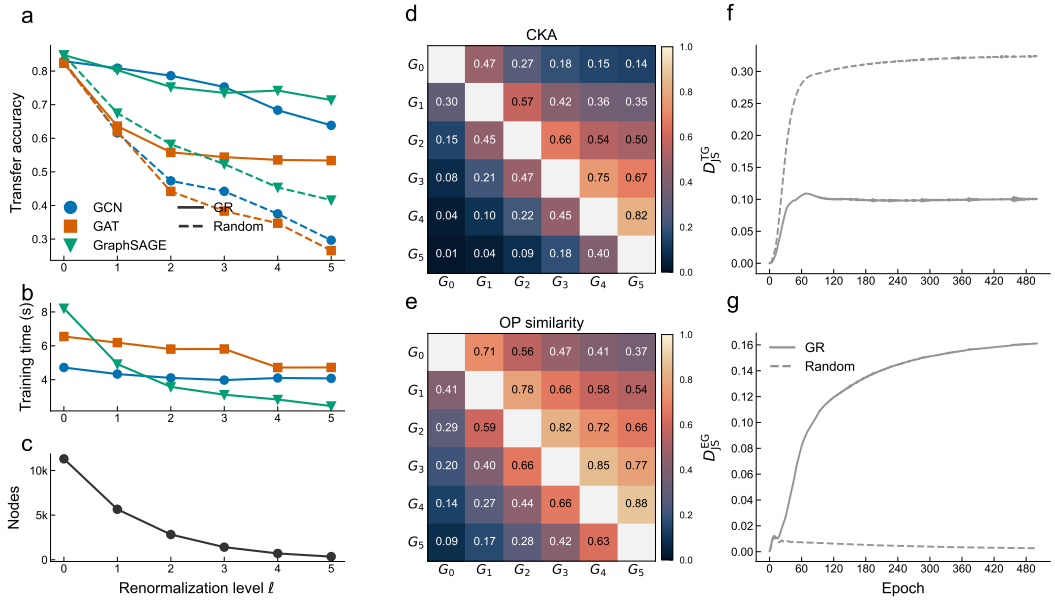

Figure 18: **Zero-shot GNN transfer on WikiCS dataset**. See the caption of Fig. 3 for more detail.

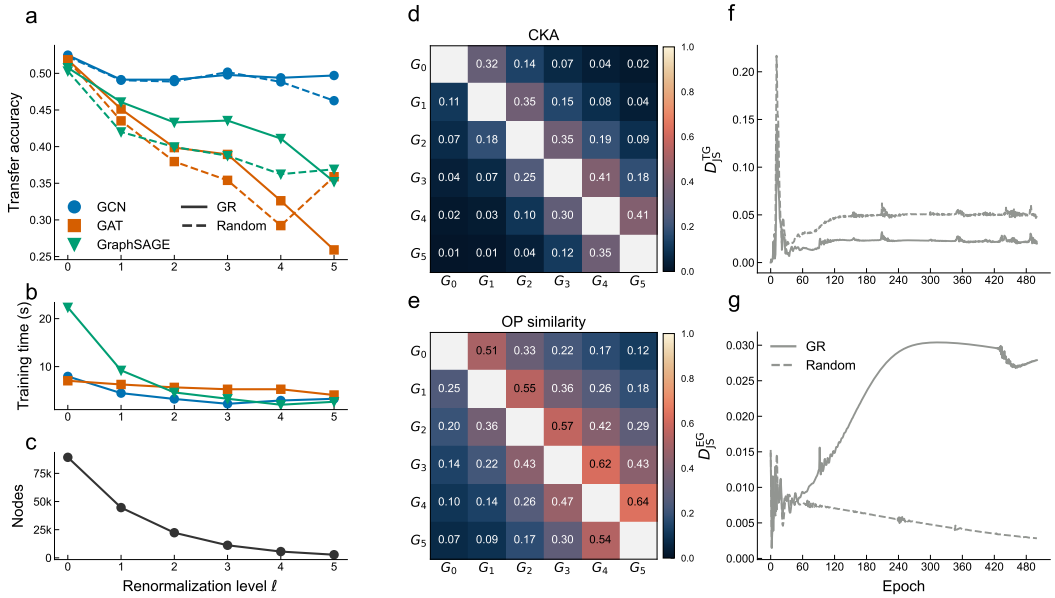

Figure 19: **Zero-shot GNN transfer on Flickr dataset**. See the caption of Fig. 3 for more detail.

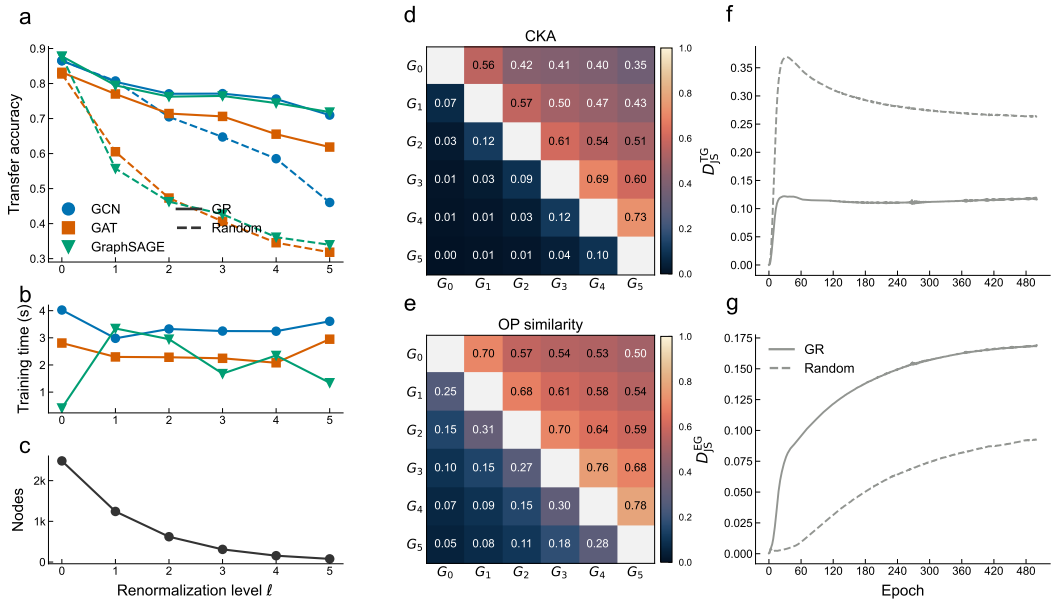

Figure 20: **Zero-shot GNN transfer on Cora dataset**. See the caption of Fig. 3 for more detail.

Figure 21 shows the zero-shot transfer accuracy for the synthetic networks in both the ultra-small-world (USW) and small-world (SW) regimes, comparing GR against Random-T and Random-G.

Figure 22 shows the zero-shot transfer accuracy for each of the eight real-world datasets, comparing GR against Random-T and Random-G.

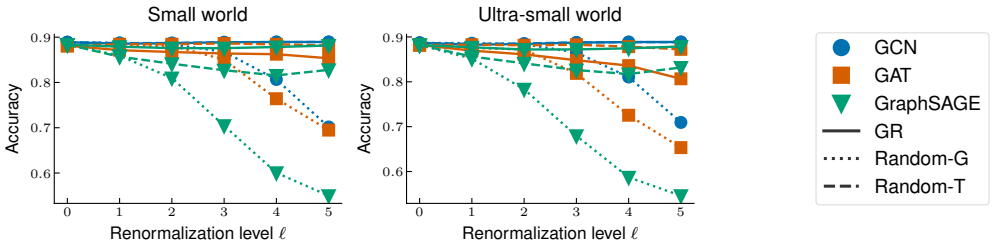

Figure 21: **Zero-shot GNN transfer accuracy for SW and USW regimes, against Random-T and Random-G renormalization methods**. See the caption of Fig. 3 for details on models and renormalization methods shown.

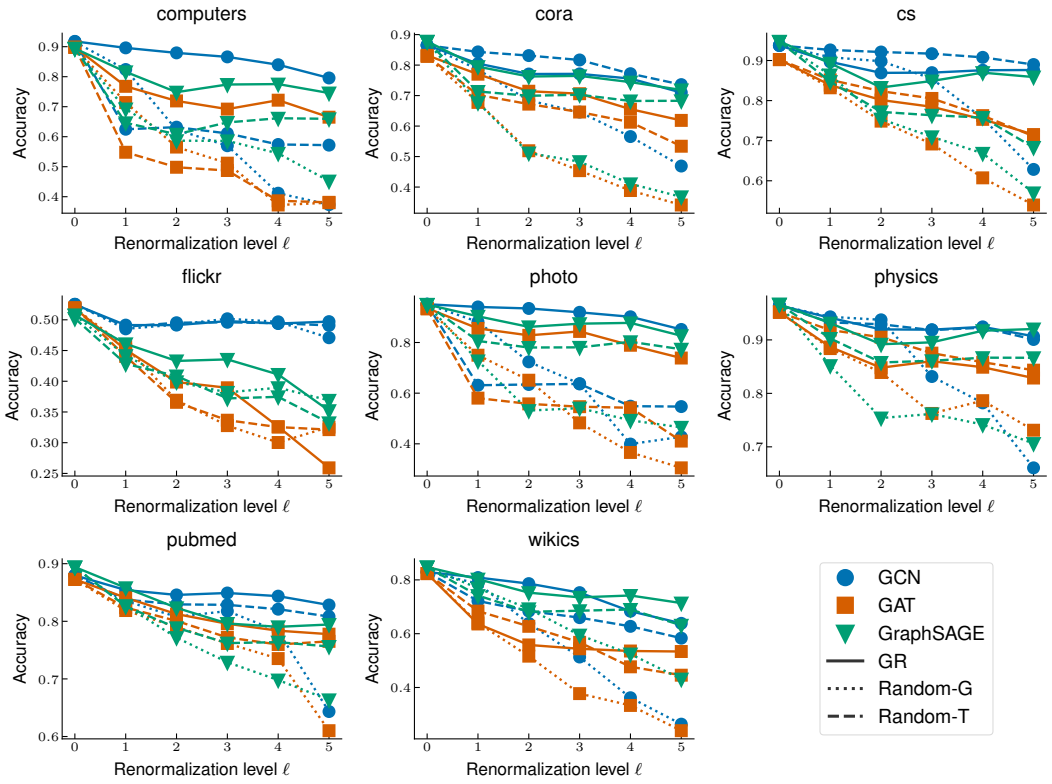

Figure 22: **Zero-shot GNN transfer accuracy across all real-world datasets, against Random-T and Random-G renormalization methods**. Each panel corresponds to one dataset. See the caption of Fig. 3 for details on models and renormalization methods shown.

Interestingly, the three architectures respond differently to the Random-T baseline. We hypothesize that this reflects how each model aggregates neighborhood information. One possible explanation is that each architecture aggregates neighborhood information differently: GraphSAGE separates self and neighbor transformations, GCN applies a shared transformation across both, and GAT learns attention weights over neighbors, which may make them differently sensitive to randomized topology.

A.9.4 TOPOLOGICAL VALIDATION

Figures 23 and 24 show the topological validation of GR across all real-world networks considered.

Figures 25 and 26 further show the topological validation of the SW regime under the random renormalization variants (Random, Random-T, and Random-G), the latter broken down by renormalization level $l = 1, 2, 3, 4$.

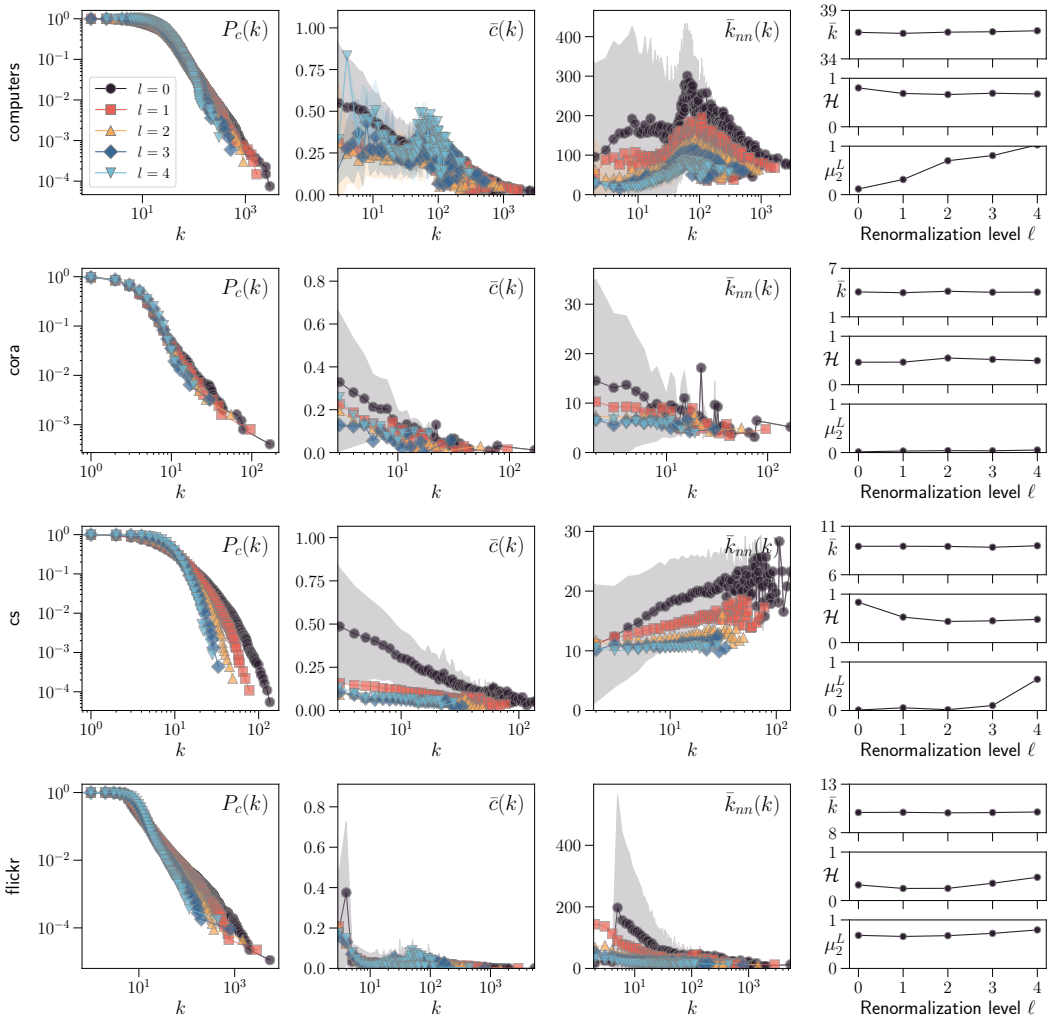

Figure 23: Topological validation of GR for the real-world datasets considered (part 1 of 2).

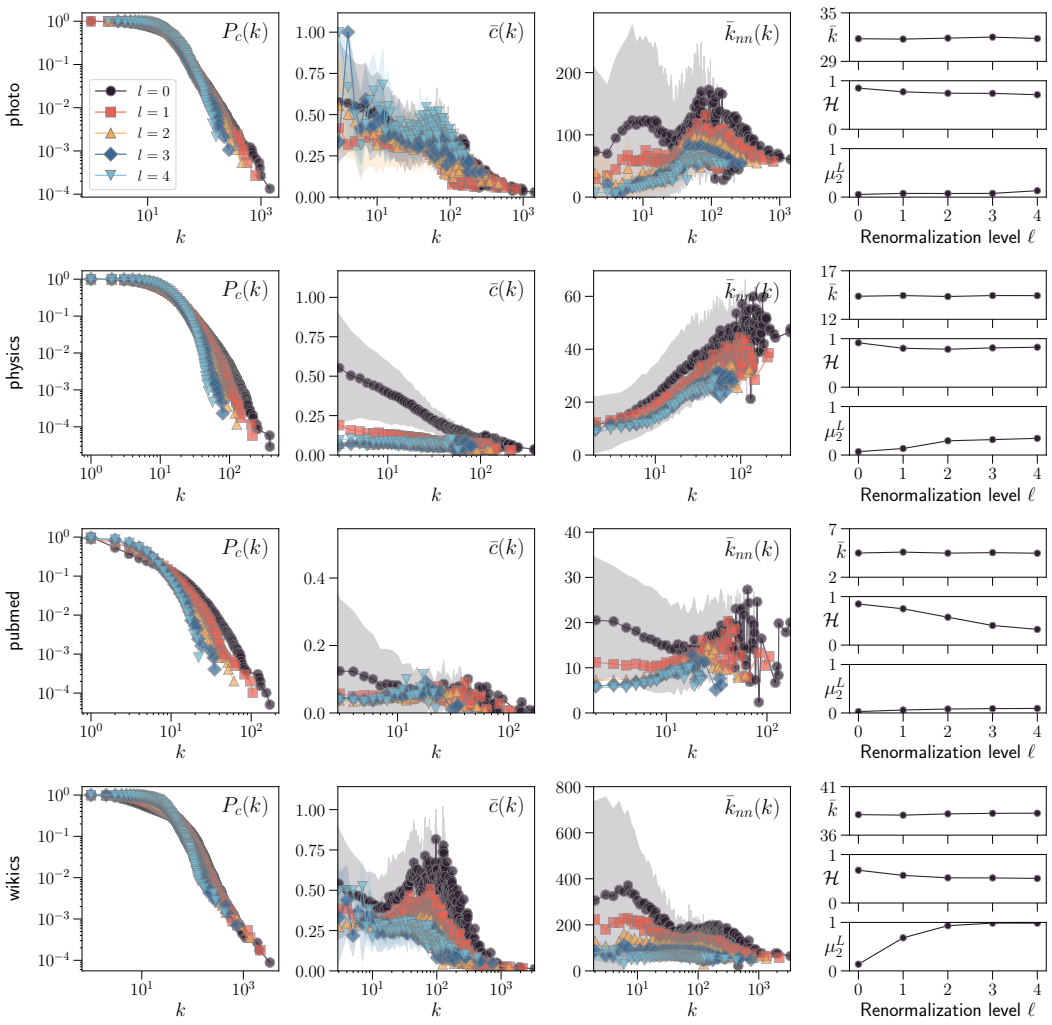

Figure 24: Topological validation of GR for the real-world datasets considered (part 2 of 2).

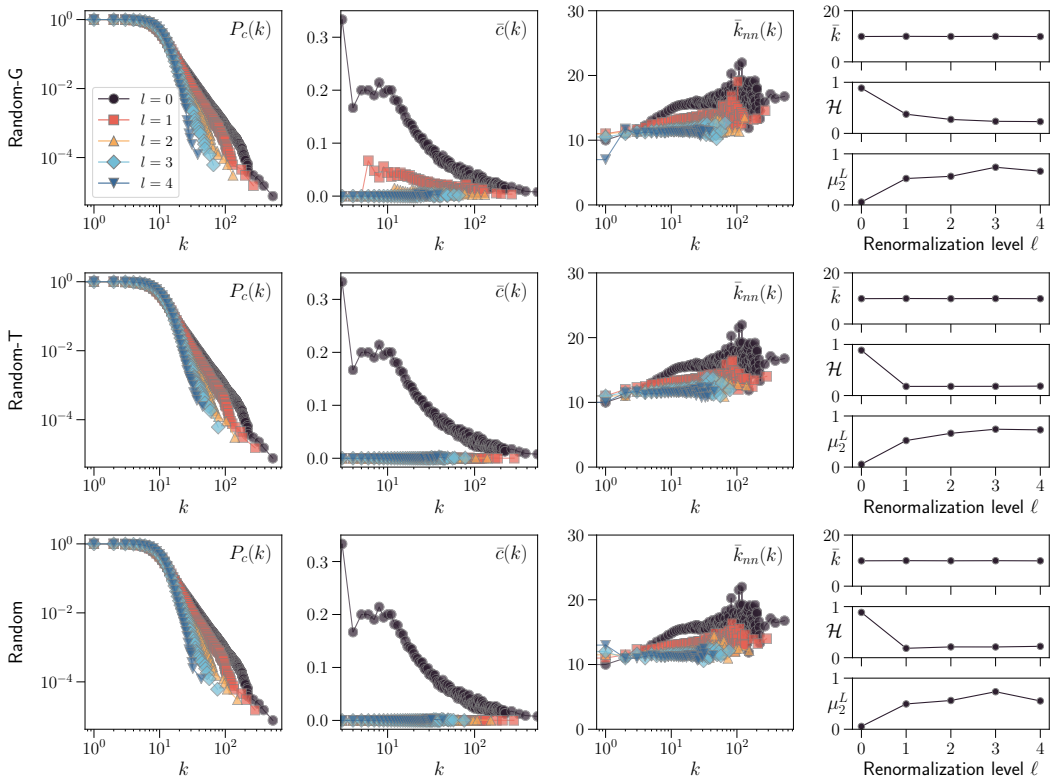

Figure 25: Topological validation of the SW regime under three random variants of the renormalization method.

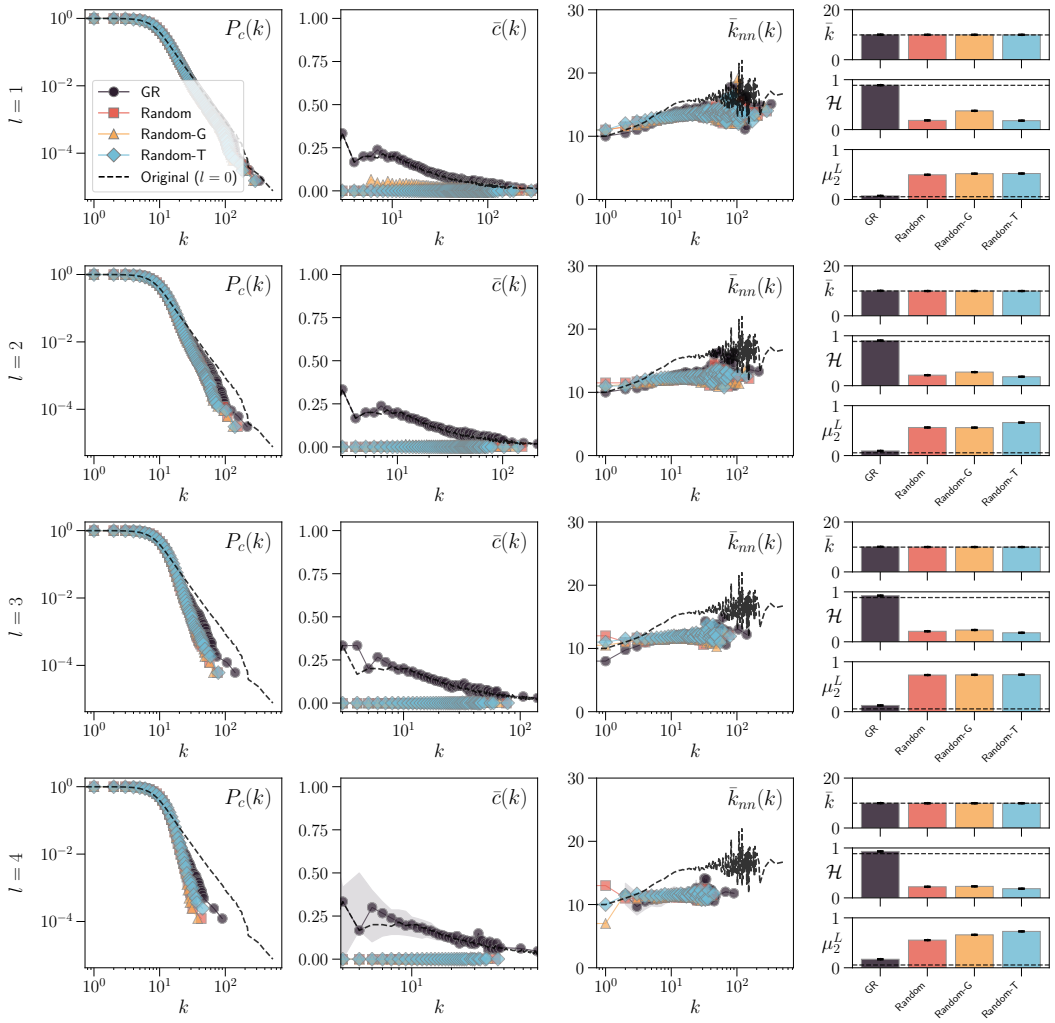

Figure 26: Topological validation of the SW regime under GR and three random variants-Random, Random-G, and Random-T of the renormalization method, shown for renormalization levels $l = 1, 2, 3, 4$. Dashed black lines indicate the reference values of the original (non-renormalized) network at $l = 0$.